Universidad de las Ciencias Informáticas

Facultad 3

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

Título: Diseño del subsistema Control de Medios de Transporte Internacional del sistema Gestión Integral de Aduanas.

Autor:
Mariem Salinas Pérez

Tutor:
Ing. Rafael Andrés Céspedes Basteiro

Ciudad de La Habana, junio del 2011
"Año 53 de la Revolución"

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Ernesto Che Guevara

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2011.

Mariem Salinas Pérez	Ing. Rafael A. Céspedes Basteiro
Firma del autor	Firma del tutor

A la memoria de mi abuelo Orestes quien fue partícipe de un sueño que hoy se convierte en realidad.

A mis padres por quienes hago todo para que se sientan orgullosos de mí.

A todos los que siempre creyeron en mí.

Agradecimientos

Todo trabajo, del tipo que sea, tiene además del autor, un conjunto de personas que de una forma u otra han colaborado y dado su apoyo. De ahí que sea tan importante para mí agradecerle a un grupo grande el gran sustento, seguridad y confianza que me han brindado.

A mis padres que son lo más valioso que tengo.

A mi mamá por haber sido mi más grande apoyo durante estos 5 años, gracias a ti me he convertido en la persona que soy hoy, gracias por haber tenido siempre una frase de amor para mí, mami eres lo mejor que me ha dado la vida.

A mi papá muchas gracias por haber estado ahí cada vez que te necesité y por haberme dejado seguir mis sueños, tú has sido mi ejemplo papi. Gracias por depositar toda tu confianza en mí, sencillamente eres el mejor padre del mundo.

A los dos gracias por haber inculcado en mi los deseos de superación profesional.

A mi abuela Sabina quien ha sido para mí, como una madre, gracias por tus desvelos y preocupaciones.

A mi hermana Bebi, gracias por quererme tanto, tú eres mi más valioso tesoro.

A los cuatro les dedico este triunfo porque se cuánto los enorgullece verme graduada.

A mis amigas: Aylín, Odelkis, Maydel, Lilibel, Yeni y Lucrecia no puedo pensar en los mejores momentos que he pasado en la universidad sin que estén ustedes, muchas gracias por haberme dado su amistad.

A mi amigo Yiset que sin su amistad incondicional no hubiera podido superar muchos de los obstáculos durante estos 5 años, gracias.

A mi novio Alexis, gracias por todo el amor que me das, te amo mucho.

Agradecimientos

A todos mis compañeros de universidad y a los profesores que he tenido durante estos 5 años, gracias a ustedes he podido llegar hasta aquí.

Gracias a mi tutor Céspedes por su paciencia.

RESUMEN

Controlar de forma efectiva los medios de transporte internacionales a su entrada y salida del país es una de las responsabilidades de mayor peso en el área de Lucha Contra el Fraude de la Aduana General de la República de Cuba. El objetivo principal del presente trabajo es realizar el diseño de un subsistema que facilite la gestión de los procesos relacionados con el control de medios de transporte internacionales. Para lograr este propósito se utiliza la metodología de desarrollo RUP, la cual a su vez utiliza UML para elaborar todos los artefactos. Como herramienta de modelado se utilizó el Visual Paradigm. Se describen las técnicas y herramientas utilizadas así como la aplicación de patrones para realizar un diseño de calidad.

Para comprender el negocio se realizó un estudio exhaustivo de los artefactos del análisis a partir de los cuales se generaron los artefactos del diseño. Para garantizar la calidad de los artefactos generados se aplicaron métricas que arrojaron resultados positivos.

La puesta en marcha de este sistema permitirá erradicar los problemas existentes a la hora de registrar los controles efectuados a los medios de transporte internacionales a su paso por la frontera, ya que se contará con un solo subsistema que brinde información a los analistas de Lucha Contra el Fraude y que les facilitará la toma de decisiones.

Palabras Clave: *aduana, diseño, MTI, LCF, enfrentamiento, control.*

ÍNDICE

INTRODUCCIÓN

Las actividades comerciales son parte importante dentro de la historia de la humanidad, desde la antigüedad los hombres comerciaban intercambiando productos para de esta forma desarrollarse económicamente. Fue creciendo el ámbito en el cual intercambiar y de esta forma se hizo necesario trasladar la mercancía a través de grandes extensiones de tierra y océano, surgiendo de esta manera lo que llamamos comercio internacional. A su vez a los medios de transporte que se utilizaban para estas labores se les llamó medios de transporte internacional (MTI).

En la actualidad el comercio entre naciones se realiza utilizando medios de transporte modernos que son capaces de atravesar el mundo en pocos días y llevar consigo miles de toneladas de productos. A su vez se puede plantear que los países en su desarrollo son dependientes de las importaciones y exportaciones de productos que puedan realizar, entrando aquí el papel de la aduana como organismo fiscal que media sobre los productos que se importan o exportan desde y hacia los países.

La Aduana General de la República de Cuba (AGR) constituye un órgano de control en la frontera y de fiscalización en la actividad vinculada al comercio exterior. Esta institución, creada el 5 de febrero de 1963, tiene entre sus objetivos la agilización y optimización de sus procesos así como la obtención de información oportuna para la toma de decisiones, para lo cual se propuso la tarea de automatizar sus procesos y sistemas actuales (Hernández Ruiz, 2010). Entre sus misiones está garantizar la seguridad y protección de la sociedad socialista y de la economía nacional, por lo que este organismo creo el área de Lucha Contra el Fraude (LCF) que se encarga de enfrentar las acciones terroristas, de narcotráfico, los contrabandos comerciales o no comerciales y las operaciones que ponen en riesgo el patrimonio cultural y natural del país (Céspedes Basteiro & Rodríguez Pérez, 2009).

Es de destacar que en el mundo los MTI han sido utilizados para acciones que ponen en riesgo la vida de personas; han sido utilizados por terroristas, narcotraficantes y contrabandistas. En ellos se ha transportado de forma ilícita armas, sustancias químicas, explosivos, virus, drogas de cualquier tipo, obras de arte, especímenes de la flora y fauna, así como las propias personas que son capaces de raptar los propios medios de transporte y a su vez a las personas que normalmente viajan en ellos. Todo esto con el objetivo de buscar beneficios personales. Está claro que estas acciones que se pueden

realizar en los MTI afecta en gran medida la paz y tranquilidad de la que deben disfrutar las personas. En esto radica entonces la importancia de los especialistas de enfrentamiento de MTI en la AGR para con su país, estas personas tienen basta experiencia acumulada por años en cuanto al desafío que resulta de detectar los fraudes que pueden existir en los MTI que entran y salen de Cuba.

Para lograr de forma efectiva un enfrentamiento adecuado sobre los medios de transporte es importante para los analistas de LCF contar con información histórica y/o adelantada de los MTI. Surgiendo así en el año 1997 el Sistema Automatizado de Control Mercantil (SACOM) en su versión 1.0 desarrollado por el Centro de Automatización para la Dirección y la Información de la AGR (CADI). Este software está desarrollado en Visual FoxPro y en este se conserva mucha información sobre las infracciones de los MTI; a más de 13 años de su primer uso resulta una herramienta obsoleta que no se adapta a los procesos actuales que realizan los especialistas en el control a los MTI, presentando problemas con su mantenimiento ya que no hay desarrolladores que se dediquen a estas tareas. Este sistema además corre sobre sistemas Windows, esto va en contra de la política tecnológica de la aduana de migrar completamente a software libre, además para poder usarlo se tiene que emular porque todas las computadoras de la aduana tienen instalado el sistema GNU/LINUX en la distribución Ubuntu.

En el año 2005 se estableció el Sistema Único de Aduanas (SUA) donde colaboraron los especialistas del CADI con trabajadores y estudiantes de la Universidad de las Ciencias Informáticas (UCI), en este proyecto se logró un subsistema de despacho de MTI que trata el proceso de tramitación ante la aduana de toda la documentación que declara la carga, provisiones, pasajeros y tripulantes que transporten los MTI. En este sistema resulta engorroso el control aduanero a los MTI porque existe discordancia entre la información que se recibe tanto manual como digital, además de que los trámites y gestión de esta información pasan por muchos intermediarios hasta llegar a su destino, lo que constituye la razón por la cual en ocasiones esta se recibe incompleta o tardía. Es importante aclarar que este sistema no fue realizado para el enfrentamiento y por tanto no responde a las actividades que se realizan para enfrentar a los medios de transporte internacionales.

Ambos sistemas SACOM y SUA presentan la dificultad de no poder ayudar a los especialistas de enfrentamiento en su labor para con los MTI porque les faltan

funcionalidades específicas dirigidas al enfrentamiento, problema que se pretende resolver con la solución informática de Gestión Integral de Aduanas (GINA), desarrollada por la UCI en completa colaboración con el CADI y especialistas en procesos aduaneros de la AGR. GINA tiene como misión informatizar todos los procesos aduaneros, y dentro del propio sistema está concebida la realización de un subsistema de control de MTI que brinde las prestaciones que necesitan los especialistas para enfrentar el ilícito aduanero que ocurre en los medios de transporte internacional. El sistema GINA es directamente desarrollado bajo las condiciones de trabajo y políticas establecidas por la aduana cubana, basado en los problemas y virtudes existentes en los sistemas anteriores de los cuales se toman las buenas ideas y funcionalidades, además de que en este software se vinculan tanto a las nuevas políticas como a las nuevas tecnologías existentes.

GINA se desarrolla en la Facultad 3 en el centro de desarrollo CEIGE (Centro para la informatización y gestión de entidades), en este centro existe un departamento de Desarrollo de Soluciones para la Aduana y este departamento presenta el proyecto Enfrentamiento el cual tiene a su cargo el desarrollo de todos los subsistemas de enfrentamiento del GINA. Este proyecto cuenta con los artefactos de análisis necesarios para el desarrollo efectivo del sistema informático que ayude a resolver los problemas de enfrentamiento relativos al Control de MTI.

De manera particular el presente trabajo pretende abordar esta temática centrándose en el siguiente **problema a resolver:** ¿De qué forma se puede modelar un subsistema que permita automatizar los procesos de gestión relativos al control de los MTI?

Basándose en lo anterior se formula como **objeto de estudio:** La gestión de los procesos relacionados con el enfrentamiento en la Aduana General de la República de Cuba el cual enmarca como **campo de acción**: La gestión de los procesos relacionados con los controles a MTI en la Aduana General de la República.

Para darle solución al problema expuesto anteriormente se propone como **objetivo general:** Realizar el diseño que cumpla con los requisitos planteados en el análisis y permita una correcta implementación de este subsistema para la Aduana General de la República de Cuba.

Como **idea a defender** para esta investigación se plantea que: El diseño de los procesos relacionados con los controles a MTI permitirán una correcta implementación de este

subsistema para la Aduana General de la República de Cuba, ya que satisfará las necesidades de información de enfrentamiento relativas al control de MTI en la misma.

Se definen **objetivos específicos:**

1. Realizar un estudio de los procesos de negocios así como los requisitos descritos en el análisis de Control de MTI.
2. Obtener el modelo de diseño de la solución Control de MTI para el sistema Gestión Integral de Aduanas.
3. Validar técnica y funcionalmente el diseño obtenido.

Con el presente trabajo se espera obtener como **posible resultado**: El modelo de diseño del subsistema Control de Medios de Transporte Internacional.

Se realizó un estudio de los métodos para la investigación científica, en la realización de este trabajo se utilizaron los que se describen a continuación:

Método Histórico – Lógico

Métodos históricos: Analiza todo el trayecto por el cual atraviesa el objeto a estudiar. De esta manera es posible conocer sus etapas y reacción ante los períodos de la historia.

Métodos lógicos: Estudia internamente la lógica de su desarrollo desde el punto de vista histórico para obtener el conocimiento primario en el cual se haya su teoría. Básicamente se manifiestan en forma teórica así es posible relacionar la estructura que ha adquirido el objeto a través de la historia. Lo lógico no repite lo histórico en todos sus detalles, se enfoca solo en lo importante del fenómeno.

La utilización de ambos métodos resulta una relación necesaria para encontrar resultados que no se basen en lo especulativo y muestre bases de razonamiento previo en la investigación. Para esto el método lógico debe enfocarse en lo que se revele en el método histórico.

Método Analítico – Sintético

Se trata de analizar toda la documentación bibliográfica como el contexto donde se desarrolla para poder descomponer en partes pequeñas a las cuales poder realizarle una selección adecuada de lo que se necesita y elaborar las ideas correspondientes que

tributen a todo el objeto de estudio. De esta manera es práctico dividir mentalmente todo el contenido, analizarlo y posteriormente de manera analítica integrarlo

Método Modelación

La modelación es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. Es un modelo que recrea al objeto de estudio por lo que genera una correspondencia objetiva entre ambos elementos. Resultando una vía de investigar la realidad donde se encuentra el problema.

Dado que cada problema resulta único en su concepción, las maneras de modelarlo cambian para una mejor comprensión.

Modelo analógico

Este tipo de modelado solo se enfoca en la estructura de las relaciones y determinadas propiedades fundamentales de toda la realidad. Se establece una analogía entre el sistema real y el modelo, y se estudia el primero, utilizando como medio auxiliar el segundo.

El presente documento consta de 3 capítulos en los cuales se desarrollan aspectos de importancia para lograr el objetivo que se persigue:

Capítulo 1: Fundamentación teórica:

1. Estudio del estado del arte del tema a nivel nacional e internacional.

2. Definición de herramientas y metodologías que se utilizarán en la propuesta de solución.

Capítulo 2: Diseño del Subsistema:

1. Modelación del sistema, se le da forma para que soporte todos los requisitos y restricciones, consolidando una arquitectura que sirva de base para la implementación.

Capítulo 3: Validación de la solución:

1. Aplicación de métricas que validen el correcto funcionamiento del diseño obtenido.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se brinda información acerca del estudio realizado sobre el origen y desarrollo de las aduanas a nivel mundial y específicamente en Cuba, se centra en la descripción de los sistemas que existen actualmente para llevar a cabo el enfrentamiento a las ilegalidades en los Medios de Transporte Internacional. Se incluye la especificación de la metodología, el lenguaje y las herramientas empleadas para el diseño de la solución.

1.1 Estado del arte

1.1.1 Antecedentes internacionales

Las aduanas en el contexto mundial fueron creadas para el control del intercambio comercial entre los países con el fin de proteger los comercios y productos propios de cada uno de ellos. Las funciones de las aduanas pueden clasificarse en específicas y complementarias, estas últimas aplican las leyes o códigos aduaneros, la legislación arancelaria, los tratados y convenios internacionales comerciales y a su vez cuidan que el tráfico se realice conforme a las normas legales, persiguiendo y reprimiendo el ilícito aduanero.

El desarrollo de las industrias, el transporte y el comercio han influido en la evolución y modernización de las aduanas a nivel mundial de un tiempo a la fecha para convertirse de una administración fiscal en una administración económica. La consecuente liberación continua del intercambio internacional de mercancías como resultado de la creación de los bloques económicos ha aumentado y ello ya no puede ser controlado con los instrumentos tradicionales del derecho aduanero, ya que esto trae como consecuencia el entorpecimiento del comercio y por consiguiente de la economía. Actualmente la función de las aduanas del mundo es facilitar el comercio, hacerlo más práctico, expeditivo y funcional.

En el mundo se ha llegado a la conclusión que la opción más recomendable para obviar el control y las precauciones administrativas versus agilización de trámites es la simplificación de las formalidades aduaneras. Esto es el conjunto de operaciones que realizan las aduanas para asegurar el cumplimiento de las prescripciones legales y reglamentarias que las aduanas tienen a su cargo en cuanto a control de personas,

equipajes, mercancías y medios de transporte que crucen la frontera (Rodríguez Arce, 2008).

A consecuencia de la magnitud del comercio internacional, y de los transportes, los gobiernos extranjeros vienen dándose cuenta que la labor de las aduanas debe ser facilitadora y no represiva, esto es labor económica. Las aduanas del mundo actualmente han pasado a convertirse de aduanas fiscales en aduanas económicas, siguiendo esta línea como la mejor política para sus procesos. Esto trae consigo un deterioro del enfrentamiento a las ilegalidades efectuadas en todo el entorno aduanero. Por lo tanto las aduanas extranjeras en estos momentos pueden caracterizarse como aduanas económicas enfocándose en el desarrollo de los procesos comerciales y a su vez en el aumento de la economía; dejando atrás los instrumentos tradicionales para reprimir el ilícito aduanero.

Durante este estudio sobre los softwares involucrados en la gestión aduanera utilizados internacionalmente se analizaron los sistemas SIDUNEA, SOFIA, María y SARA, los cuales se describen a continuación.

SIDUNEA

El Sistema Aduanero Automatizado (SIDUNEA) es la herramienta informática para el control y administración de la gestión aduanera, desarrollada por La Conferencia de las Naciones Unidas sobre el Comercio y el Desarrollo UNCTAD, y que actualmente es usada con éxito en más de 80 países. SIDUNEA permite realizar un seguimiento automatizado de las operaciones aduaneras y controlar efectivamente la recaudación de los impuestos aduaneros, porque este sistema verifica automáticamente los registros, calcula los impuestos y contabiliza todo lo relativo a cada declaración, con la mínima intervención del factor humano subjetivo.

Al ser un sistema multidisciplinario, está especializado en cada área del trabajo aduanero para ser la herramienta de trabajo de todos los clientes de la aduana, sean usuarios internos o externos, privados o públicos. De este modo se convierte en un único lenguaje, seguro y comprensible para todos los actores del proceso. SIDUNEA implementa los estándares internacionales para procesar los datos de comercio exterior ya acordados por la Organización Mundial de Aduanas (OMA) y por la Organización Internacional para la Estandarización (ISO).

El sistema SIDUNEA contiene ocho módulos, a continuación se describen algunas funcionalidades de los que intervienen de una forma u otra en todo el proceso de control del despacho de medios de transporte (Aduana Nacional, 2004c).

MODBRK

El Módulo para el Agente Despachante o MODBRK ha sido diseñado para ser utilizado por un declarante o agente despachante de aduanas. Este trabaja principalmente con la declaración de mercancías. Adicionalmente, contiene opciones de reporte para verificar el estado de bienes declarados bajo regímenes suspensivos, tales como el depósito de aduanas (Aduana Nacional, 2004d).

MODCAR

El Módulo de Transporte o Control de Manifiestos (MODCAR) sirve para la preparación y transmisión de detalles del transporte de carga en formato electrónico. Se utiliza para generar el formato electrónico del Manifiesto de Carga y sus documentos de transporte. Además se utiliza junto a otros módulos del sistema SIDUNEA para el control de la carga, incluyendo el retiro de las mercancías y el manejo de los inventarios de carga (Aduana Nacional, 2004a).

MODCAR permite que los transportistas ingresen al sistema SIDUNEA la información detallada referente a la carga que transportan. De igual manera, la Aduana en éste módulo podrá realizar los controles que le compete; siendo el manifiesto uno de los más importantes.

Este módulo posibilita que la Aduana y los usuarios transportistas cuenten, anticipadamente, con información de la carga transportada facilitando y agilizando el paso de mercancías por fronteras y el cumplimiento de las formalidades aduaneras correspondientes (Aduana Nacional, 2004c).

MODTRS

El Módulo de Tránsitos (MODTRS) ha sido diseñado para permitir un monitoreo y control del movimiento de las mercancías dentro el territorio nacional. Estos movimientos controlados incluyen todas las formas de Tránsito Interno, tales como el tránsito de frontera a frontera (tránsito internacional), de frontera a una aduana interna (importación)

o de una aduana interna a una frontera (exportación). Adicionalmente, pueden ser incluidos los tránsitos de mercancías entre aduanas internas (Aduana Nacional, 2004c).

MODTRS ha sido creado para facilitar la automatización del procesamiento y manejo de todos los tipos de transacción y transmisión que se ejecutan en un proceso de tránsito. Las operaciones del MODTRS son: control de tránsitos, operaciones en Aduana de destino, cierre de un tránsito, y reportes e informes. Con MODTRS, el usuario tiene la facilidad de ver, recuperar, enmendar, guardar, validar, imprimir, transmitir y listar los documentos de tránsito (Aduana Nacional, 2004b).

MODTRB

El módulo MODTRB realiza las mismas funciones que MODTRS, lo que específicamente para Funcionarios de Aduana y otros operadores. Los agentes despachantes de Aduana y otros operadores pueden hacer uso del módulo MODTRB para preparar documentos de tránsito.

Con MODTRB, el usuario tiene la facilidad de capturar, ver, guardar localmente, guardar, recuperar, imprimir, enmendar, borrar y listar los documentos de tránsito. En caso de la presentación de la documentación ya ingresada por el declarante en MODTRB, la Aduana recupera el documento desde el servidor, lo revisa y valida el documento. Adicionalmente, el sistema genera un mensaje que contiene el documento y lo envía a la Aduana de destino, a través del sistema (Aduana Nacional, 2004b).

SIDUNEA es un sistema especializado en el despacho aduanero, no así para los temas de enfrentamiento; no es utilizado por la AGR debido a que es privativo y no se adapta a las regulaciones y normativas de la aduana cubana.

Sistema Integrado Financiero y Administrativo (SOFIA)

SOFIA es un sistema modular e integrado de información financiera, comercial, administrativa y gerencial, totalmente parametrizable que se ajusta a las necesidades específicas de las organizaciones públicas o privadas de cada sector económico. Permite optimizar los procesos financieros y administrativos, de forma eficiente y segura, brindando información en línea, generación automática de transacciones, informes estadísticos e indicadores de gestión para la toma de decisiones gerenciales y operativas oportunas.

Este sistema solo se encarga de la gestión de los procesos relacionados con las finanzas, la administración, el comercio y el presupuesto público, no contiene ningún módulo que implique el control a los MTI, lo cual, además de que el software es privativo imposibilita su utilización.

Sistema Informático María (S.I.M)

María es un sistema de origen francés, fue concebido para registrar las operaciones y no para detectar operaciones fraudulentas. Tiene un diseño por el cual resulta difícil usarlo para el control, debido a que no emite alertas relacionados con las mercancías sospechosas para que sean revisadas a fondo, lo que evidencia la imposibilidad de usarlo para el control a los MTI.

Sistema Automatizado de Rentas Aduaneras (SARA)

Con este sistema es posible consultar ya sea por posición arancelaria o una descripción a cuánto asciende el porcentaje de impuestos que paga una determinada mercancía. Esto es importante pues mediante la utilización del mismo se dispone de la versión más reciente del arancel nacional pudiendo el mismo utilizarse para el elaboración de pre-liquidaciones y cálculos de tributos aduaneros con antelación a la operación de comercio internacional.

Lo anteriormente expuesto evidencia que este sistema no abarca en su alcance el control a los medios de transporte internacional, por lo cual no se valoró su utilización.

1.1.2 Antecedentes Nacionales

En Cuba debido a sus características físico-geográficas, las vías de transporte internacional se reducen a marítima y aérea; por tanto el control aduanero a los MTI se evidencia en puertos, marinas y aeropuertos de la isla.

El área de la aduana Lucha Contra el Fraude se encarga de regular de forma efectiva todos los procesos de control aduanero a los Medios de Transporte Internacional. Las condiciones, trámites y formalidades que se llevan a cabo para la ejecución del despacho y control aduanero a los MTI, a la entrada y salida del país, son los componentes que comprenden dichos procesos. El elemento fundamental de los procesos de control aduanero a los MTI es la recepción de la información adelantada referente a las cargas,

provisiones, pasajeros, tripulantes y otros datos característicos de los MTI. La información adelantada de los MTI está representada por los documentos OMI (Organización Marítima Internacional) y el Manifiesto de Carga si el MTI es marítimo; en el caso aéreo la conforma la Información Adelantada de Carga (ACI), la Información Adelantada de Pasajeros (API) y el Operacional (Hernández Ruiz, 2010).

Debido a las actividades que se realizan en LCF los sistemas informáticos que se han logrado en Cuba, son para dar respuesta a las necesidades que en esta área se plantean. Aquí se han obtenido resultados, tales como:

Sistema Automatizado de Control Mercantil (SACOM)

Este sistema puesto en funcionamiento desde el año 1997 está desarrollado en Visual FoxPro, es un sistema que debe ser instalado en cada PC (computadora personal por sus siglas en inglés) por lo que inhabilita el acceso global al mismo. Para actualizar la información desde los distintos puntos del país se tienen que copiar ficheros de base de datos diariamente desde los servidores FTP[1] de la aduana, esta tarea es tardía y tediosa para los usuarios que la realizan. Tiene como característica que se pueden registrar las infracciones de los medios de transporte internacional pero no presenta información sobre las personas naturales que se transportan en ellos. Otro problema que presenta es que no existe integración con la información adelantada tan importante en el desenvolvimiento de las acciones de enfrentamiento. Se tiene que plantear además que este sistema no presenta en la actualidad ningún tipo de soporte por parte de los desarrolladores del CADI.

Sistema Único de Aduanas (SUA)

Otro software desarrollado en Cuba es el Sistema Único de Aduanas, el cual se realizó en PHP v 5.04, de manera estructurada y sin la utilización de un *framework*[2] de desarrollo. Colaboraron en esta solución especialistas del CADI con estudiantes y profesores de la UCI, logrando de esta forma un módulo de Despacho de Medios de Transporte Internacional. Pero este se presenta de forma incipiente ante las necesidades de los especialistas de enfrentamiento, no presenta ninguna operación para combatir el ilícito

[1] ***FTP:*** Protocolo de Transferencia de Archivos

[2] ***framework*****:** es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado

aduanero y solo le da seguimiento a los procesos relacionados con el despacho de los MTI. Es importante destacar que este módulo nunca se puso en explotación por parte de los aduaneros cubanos por lo que no se abala como una solución factible, aunque si presenta buenas ideas que pueden ser utilizadas en el despacho aduanero.

Resumen de sistemas

Los sistemas analizados en el ámbito internacional se dedican al despacho y fiscalización de mercancías dejando a un lado las operaciones de enfrentamiento, es por esto que resultan escasas sus ideas y funcionalidades para este tipo de actividad. En cuanto a los sistemas nacionales les falta cumplir con los requerimientos actuales y ponerse a la altura de las nuevas tecnologías, las cuales brindan mejoras sustanciales en cuanto a robustez, fiabilidad y flexibilidad. Es por esto que se le da continuidad en el presente documento a la idea de que es necesario incluir en el GINA el desarrollo de un subsistema para el control de los MTI.

1.2 Diseño de software

Siempre que se idea un resultado, y la maravillosa mente humana lo hace realidad mediante la imaginación, intuitivamente se atraviesa por el paso de diseño, donde se trata de concebir de la mejor manera posible el objeto que se desea. Es en el diseño entonces donde la idea se convierte en forma y donde se aterriza el pensamiento inicial. En el campo del software sucede de manera similar, donde el análisis sería la idea y el diseño es la guía que es capaz de decir paso a paso como materializarla.

El término diseño admite varios significados. Así, el “diseño” puede ser una actividad, la “actividad de diseñar”, puede ser un producto, el “resultado de la actividad de diseñar”, o puede ser un calificativo, y en este sentido es muy común referirse a algo como “de diseño”, cuando aporta una geometría, una forma o unas cualidades diferenciadoras que implican un aire de calidad y distinción.

Se debe notar que, de acuerdo con esta significación, el diseño aborda los “elementos básicos”, esto es, los más relevantes o fundamentales. La ordenación de los detalles correspondería a una parte del “diseño”, que sería el “diseño detallado”. También se debe apuntar que el diseño no conlleva necesariamente unas tareas de “cálculo” o de “dimensionamiento preciso”, tareas que sí formarían parte de un diseño detallado o de las propias de una ingeniería (MACAS and FIERRO, 2009).

1.3 Metodología, lenguaje y herramientas de modelado para el desarrollo

1.3.1 Metodología de desarrollo

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software. La metodología de desarrollo que se empleó para la realización del presente trabajo fue RUP.

Proceso Unificado de Desarrollo (RUP)

En esta metodología se han unificado técnicas de desarrollo mediante el Lenguaje Unificado de Modelado (UML). En su modelación define como sus principales elementos el quién hace qué, cuándo y cómo lo hace. RUP está preparado para desarrollar grandes y complejos proyectos y es la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

El ciclo de vida de RUP se caracteriza por:

- Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean. Estos guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo.

- Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Para el caso de una iteración de elaboración, centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración.

RUP representa un ciclo de desarrollo en la vida de un producto de software con 4 fases:

- La fase de Concepción: Define la visión, los objetivos y el alcance del proyecto, tanto desde el punto de vista funcional como del técnico.
- La fase de Elaboración: Completa el análisis de los Casos de Uso y define la arquitectura del sistema.
- La fase de Construcción: Se obtiene la capacidad operacional inicial del producto.
- La fase de Transición: Inicia con una versión “beta” del sistema y culmina con el sistema en fase de producción.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Los cuales son: Modelación del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue, Gestión de configuración y cambios, Gestión de proyectos y por último Entorno, donde los 3 últimos constituyen los flujos de soporte.

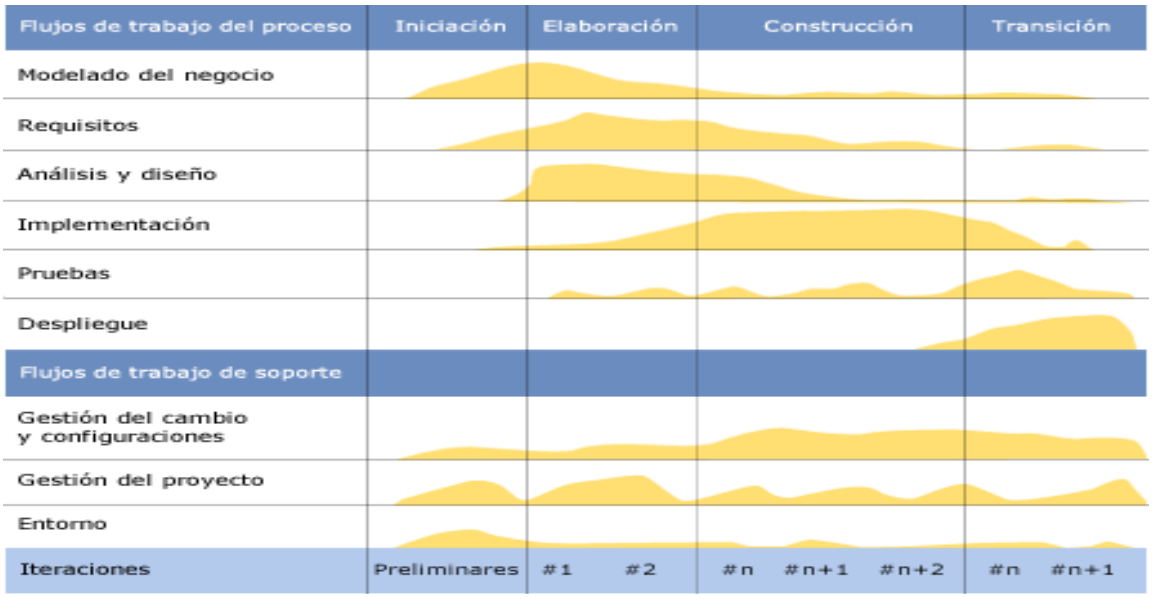

Figura 1: Fases y flujos de trabajo de RUP.

El uso de esta metodología será enfocado mayormente a las fases que propone RUP y a las actividades y artefactos que se generan en cada una de ellas. Actualmente el proyecto Aduana se encuentra trabajando con nuevas adecuaciones que transforman la manera en que RUP propone sus artefactos en busca de mejor eficiencia y mayor rapidez en la

entrega de las soluciones. Por esta razón a pesar de tomar como base esta metodología no se centra el desarrollo de la aplicación en casos de uso sino que es guiado por procesos como se explica en el Trabajo Diploma Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE (Martínez Manso, 2010).

1.3.1.1.1 Fase de diseño

Esta fase definida por RUP recibe como artefactos de entrada los generados en el análisis: el modelo de procesos, la especificación de requisitos y el modelo de casos de uso del sistema. A su vez esta debe entregar como entrada para la fase de implementación los artefactos: diagramas de clases del diseño, de secuencia orientado a actividades, de paquetes del subsistema, el diagrama de despliegue y el modelo de datos.

1.3.1.1.2 Rol del diseñador

El diseñador es el responsable de diseñar partes del sistema, teniendo en cuenta las restricciones de los requerimientos, la arquitectura y el proceso de desarrollo para el proyecto. Él identifica y define las responsabilidades, funcionamientos, atributos, y relaciones de los elementos del diseño. Asegura que el diseño es consistente con la arquitectura del software, y la detalla a un punto donde la aplicación puede proceder. La persona que desempeñe el rol de diseñador debe contar con sólidos conocimientos en cuanto a los requisitos del sistema, la arquitectura del mismo, las técnicas de diseño de software, incluyendo el análisis orientado objeto y el lenguaje unificado de modelado, las tecnologías en las que el sistema será implementado y las líneas bases del proyecto.

1.3.2 Lenguajes de programación

Lenguaje de programación para el lado del servidor:

El uso de uno o varios lenguajes de programación es parte fundamental para el desarrollo de cualquier sistema informático. En el caso de la Aduana se estableció el uso del lenguaje PHP.

PHP es un lenguaje de programación interpretado, creado en 1994 por Rasmus Lerdof. Es utilizado habitualmente para la creación de sitios, contenido dinámico para aplicaciones Web y aplicaciones para servidores. Con frecuencia los scripts PHP se embeben en otros códigos como HTML ampliando las posibilidades del diseñador de

páginas Web. La interpretación y ejecución de estos scripts se hacen en el servidor, el cliente solicitud realizada desde un navegador Web) sólo recibe el resultado y jamás ve el código PHP. Permite conexión con todo tipo de bases de datos como MySQL (Structured Query Language), Postgre SQL, Oracle y Microsoft SQL Server. PHP corre sobre 7 plataformas, funciona en 11 tipos de servidores, ofrece soporte para varios Sistemas Gestores de Bases de Datos (SGBD) y contiene unas 40 extensiones estables, actualmente se encuentra en la versión 5 (PHP, 2001).

Algunas de las más importantes capacidades de PHP son:

- Integración con varias bibliotecas externas, permitiendo generar documentos PDF (Portable Document Format) y Microsoft Office Excel (XLS).
- Ofrece una solución simple y universal de fácil programación para las paginaciones dinámicas.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, permite que los fallos de funcionamiento se encuentren y reparen rápidamente, implicando menos costos.
- Gran número de funciones predefinidas. A diferencia de otros lenguajes de programación, PHP fue diseñado especialmente para el desarrollo de páginas web dinámicas. Por ello, está dotado de un gran número de funciones que simplificará enormemente tareas habituales como descargar documentos, envío de correo electrónico, creación dinámica de imágenes y gráficos en el servidor, procesamiento de información en formularios, manipulación de cookies y sesiones, transporte de información mediante HTTP y análisis de documentos XML.
- Análisis léxico para reconocer el tipo de dato almacenado en una variable, haciéndose automáticamente, permitiéndole al usuario no tener que separar las variables de sus valores.
- Posee un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos desde el cliente evitando por ejemplo, la ocurrencia de la conocida inyección de código SQL.

Lenguaje de programación para el lado del cliente:

JavaScript es el lenguaje que permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web rapidez y vida. Este lenguaje comparte muchos

elementos con otros lenguajes de alto nivel. Hay que tener en cuenta que es muy semejante a otros como C, Java o PHP, tanto en su formato como en su sintaxis, aunque tiene sus propias características definitorias.

Al ser Javascript la tecnología imperante en el desarrollo de la capa de presentación en aplicaciones web es innegable la necesidad de utilizarlo el mismo tiene como característica fundamental que posee un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de aplicaciones basadas en AJAX, una tecnología para construir páginas web dinámicas del lado del cliente. Esto es alcanzado más fácilmente con el uso de un *framework* dedicado a procesar peticiones AJAX. En el artículo donde nació el término "AJAX", muchos autores describen ésta tecnología como "un intermediario... entre el usuario y el servidor". Su meta es proveer este motor AJAX y funciones asociadas al servidor y del lado del cliente. Una librería asiste el trabajo del desarrollador en dos niveles: en el lado del cliente, ofreciendo funciones Javascript para enviar peticiones al servidor y en el lado del servidor, el cual procesa las peticiones, busca información, y la transmite al navegador.

Interfaz de usuario

ExtJS es un *framework* para Javascript muy utilizado en el desarrollo de aplicaciones Web con AJAX. Tiene una librería inmensa que permite configurar la interfaces Web de manera semejante a las aplicaciones desktop.

ExtJS empezó siendo un conjunto de librerías y extensiones para YUI (Yahoo! UserInteface). Con el tiempo se convirtió en un framework independiente y a principios de 2007 se creó una compañía para comercializar y dar soporte del framework Ext. De esta forma ExtJS tiene dos tipos de licencias, LGPL y comercial (PHPBB, 2000).

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.).

ExtJS posee un modelo de componentes jerárquico que facilita la reutilización de cada uno de los componentes que forman esta librería y promueven la extensión de los mismos con vista a desarrollos específicos de alto grado de personalización.

Usar un motor de render como ExtJS conlleva beneficios, tales como:

- Comunicación asíncrona: En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- Eficiencia de la red: El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.
- Existe un balance entre Cliente – Servidor: La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

ExtJS lleva incluidos la mayoría de los controles de los formularios Web incluyendo Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros dentro de su librería de componentes.

1.3.3 Herramientas

1.3.3.1 Herramienta Case

Las Herramientas CASE[3] permiten aumentar la productividad en el desarrollo de software. Reduciendo el coste de las mismas en términos de tiempo y dinero pues mejora la comunicación entre los desarrolladores del proyecto. Además, facilita un mejor intercambio de ideas con el cliente y entre los propios integrantes del equipo de desarrollo, permitiendo modelar los procesos de negocio de las empresas. Aunque no son suficientes si no van acompañadas de las técnicas y metodologías adecuadas, si no se implantan de la forma correcta o si no se complementan por otros aspectos relativos a la calidad. El entorno CASE debe estar integrado con otros aspectos que también persiguen la mejora del software como son las métricas, el modelo de proceso, evaluación de capacidad de desarrollo y calidad.

[3] **CASE:** *Siglas en inglés que se utilizan para referirse a Ingeniería de Software Asistida por Computadora*

Visual Paradigm para UML

Visual Paradigm para UML es una Herramienta CASE que soporta las últimas versiones del Lenguaje Unificado de Modelado y la Notación de Modelado de Procesos de Negocios.

Se integra con las siguientes herramientas Java:

- Eclipse/IBM WebSphere
- JBuilder
- NetBeans IDE
- Oracle JDeveloper
- BEA Weblogic

Visual Paradigm para UML es apoyado por un conjunto de idiomas tanto en la generación del código como en la Ingeniería Inversa por mencionar algunos ejemplos los cuales tienen la capacidad de soporte de Java, C + +, CORBA IDL, PHP, XML Schema, Ada y Python. Además, apoya la generación del código C #, VB. NET, ObjectDefinitionLanguage (ODL), Flash ActionScript, Delphi, Perl, C - Objetivo, y Ruby. Para la realización de la Ingeniería Inversa apoya la clase Java, .NET, .DLL, .exe, JDBC, y archivos de mapeo en Hibernate.

Presenta una alta interoperabilidad pues apoya la importación y exportación de XML de versiones 1.0, 1.2 y 2.1. Para maximizar la interoperabilidad de los productos de Visual Paradigm con otras aplicaciones, se introdujo la importación y exportación de modelos de proyecto desde o hasta un formato XML. Los usuarios y proveedores de tecnología pueden integrar Visual Paradigm en cada uno de sus modelos para utilizarlos en sus soluciones con un mínimo esfuerzo (VISUALPARADIGMGROUP, 1995).

1.3.3.2 Embarcadero ERstudio

La herramienta para el diseño de base de datos utilizada en este trabajo fue PLATINUM ERwin, debido a que es la herramienta utilizada por el equipo multifuncional que dentro del proyecto se encarga de generar la Base de Datosdel sistema GINA.

Esta herramienta brinda productividad en su diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el

modelo físico perfeccionado para las características específicas de la base de datos diseñada, además ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de stored procedure y triggers para los principales tipos de base de datos.

ERwin hace fácil el diseño de una base de datos. Los diseñadores de bases de datos sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad _ relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes.

La migración automática garantiza la integridad referencial de la base de datos. ERwin establece una conexión entre una base de datos diseñada y una base de datos, permitiendo transferencia entre ambas y la aplicación de ingeniería inversa. Usando esta conexión, ERwin genera automáticamente tablas, vistas, índices, reglas de integridad referencial (llaves primarias, llaves foráneas), valores por defecto y restricciones de campos y dominios.

ERwin soporta principalmente bases de datos relacionales SQL y bases de datos que incluyen Oracle, Microsoft SQL Server, Sybase. El mismo modelo puede ser usado para generar múltiples bases de datos, o convertir una aplicación de una plataforma de base de datos a otra. ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables.

Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

1.3.3.3 Gestor de Base de Datos

Oracle es un sistema gestor de base de datos relacional desarrollado por la empresa “Oracle Corporation”, este gestor es multiplataforma y está considerado entre los más completos y potentes, destacado por su soporte de transacciones, estabilidad, escalabilidad.

Una BD Oracle está almacenada físicamente en ficheros, y la correspondencia entre los ficheros y las tablas es posible gracias a las estructuras internas de la BD, que permiten que diferentes tipos de datos estén almacenados físicamente separados.

Características de Oracle:

- Es una herramienta de administración gráfica que es mucho más intuitiva y cómoda de utilizar.
- Ayuda a analizar datos y efectuar recomendaciones concernientes a mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.
- Apoya en el diseño y optimización de modelos de datos.
- Asistir a los desarrolladores con sus conocimientos de SQL y de construcción de procedimientos almacenados y triggers, entre otros.
- Apoya en la definición de estándares de diseño y nomenclatura de objetos.

Documentar y mantener un registro periódico de las mantenciones, actualizaciones de hardware y software, cambios en las aplicaciones y, en general, todos aquellos eventos relacionados con cambios en el entorno de utilización de una base de datos.

1.3.4 Arquitectura de software

La arquitectura de *software*, es la estructura de las estructuras del sistema, la cual comprende los componentes de software, las propiedades de esos componentes visiblemente externos y las relaciones entre ellos. Esta no es más que la organización de los componentes del sistema de forma que quede registrado la manera en la que colaboran y se relacionan entre ellos. Es una vía en la cual el sistema queda modelado desde distintas perspectivas con el objetivo de lograr y establecer como deberá ser construido el futuro sistema (Pressman, 2005).

Los componentes y aspectos de la arquitectura están asociados con los distintos niveles de abstracción de la funcionalidad de los sistemas, esta asociación se manifiesta en forma clara en las distintas etapas del proceso de desarrollo de software. La arquitectura con relación al diseño propone como una actividad conciliatoria entre los requerimientos del problema, en términos de una función, y la factibilidad de una solución en términos de un sistema de software la idea básica de obtener una visión amplia, completa y humana del software, como un producto tanto del conocimiento como de la intuición del diseñador de software.

1.3.4.1 Marco de trabajo

El marco de trabajo utilizado es Symfony este es un *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Este 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.

Symfony está desarrollado completamente con PHP 5. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft y se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Principales Ventajas:

- Fácil de instalar y configurar, en la mayoría de las plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos y suficientemente flexible como para adaptarse a los casos más complejos.
- 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.
- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- Los formularios incluyen validación automatizada y relleno automático de datos (repopulation), lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.
- La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.

Este *framework* implementa internamente el patrón arquitectónico Modelo-Vista-Controlador, el cual se describe a continuación.

Patrón Modelo-Vista-Controlador

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

Modelo: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en contexto del sistema proveen de información al usuario y a la aplicación misma.
Vista: Es la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de una aplicación Web, la "Vista" es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario. (Catalani, 2010).

Contenido de cada capa en Symfony:

- La capa del Modelo: está compuesta por la abstracción de la base de datos, y el acceso a los datos.
- La capa de la Vista: está compuesta por la vista, la plantilla y el layout.
- La capa del Controlador: está compuesta por el controlador frontal y el acción.

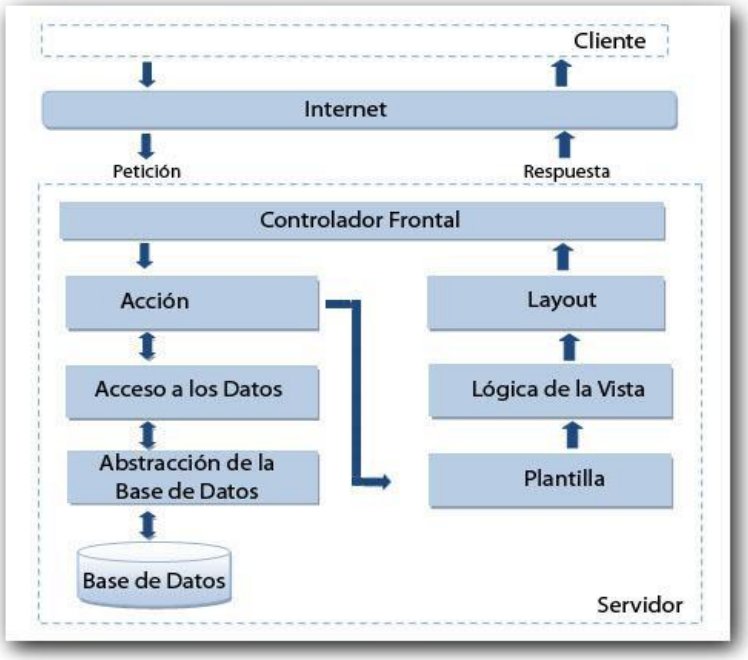

Figura 2: Implementación del patrón MVC según el Framework Symfony.

Tipo de sistema y plataforma

La solución Gestión Integral de Aduanas (GINA) es un sistema Web que se desarrolla sobre el sistema operativo GNU/Linux y es multiplataforma por lo que su funcionamiento puede ser tanto en GNU/Linux como Windows. A su vez son multiplataforma también el lenguaje utilizado (Hypertext Pre-processor (PHP), Preprocesador de Hipertexto por sus siglas en inglés), el marco de trabajo o *framework* de desarrollo (Symfony) y la base de datos (Oracle).

1.3.5 Principios básicos del diseño

El diseñador debe considerar enfoques alternativos juzgando a cada uno en base a los requisitos del problema, los resultados disponibles y los criterios de calidad interna.

- Se deberían poder seguir los pasos de diseño hasta el modelo de análisis
- El diseño no va a reinventar nada que ya esté inventado
- El diseño debería presentar uniformidad e integración
- Debe estructurarse para admitir cambios

- El diseño no es escribir código y escribir código no es diseñar
- Se debería valorar la calidad del diseño mientras se crea, no después de terminado

1.4 Buenas prácticas para el diseño

La mayoría de los softwares fallan. Alrededor del 80% de los proyectos son insatisfactorios ya sea porque están sobre cargados, retrasados, contienen funciones pérdidas o la combinación de ellos. Además el 30% de las aplicaciones informáticas son tan pobremente ejecutadas que deben ser canceladas antes de su terminación. Las aplicaciones actuales desarrolladas con herramientas potentes no son la excepción.

Aún contando con una buena arquitectura todavía es posible tener un mal diseño El problema radica en que muchas aplicaciones son o bien sobre diseñadas o poco diseñadas. Los dos principios básicos son: *Keep it Simple*[4] e *Information hiding*[5]. Para muchos proyectos es importante desarrollar un análisis y diseño orientado a objeto usando UML como se plantea en el libro UML User Guide. Reutilizar es una de las grandes promesas del orientado a objetos, pero es a menudo irrealizable debido los esfuerzos adicionales requeridos para crear capital reusable. La reutilización de código es una de las formas de utilización dentro de las existentes para proveer mejor ganancia productiva (Standish, 2008).

1.5 Patrones de Diseño

1.5.1 GRASP

Los patrones GRASP acrónimo de General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresadas en forma de patrones. Constituyen la base del cómo se diseñará el sistema. Se aplican en los primeros momentos del diseño. Estos patrones son definidos como: “Los patrones GRASP

[4] **Keep it simple**: Una expresión denominada KISS (Keep It Simple, Stupid) para referirse a la premisa de que la simplicidad debe ser la clave para el diseño y se debe evitar la complejidad.

[5] **Information hiding:** Es un principio de segregación de las decisiones del diseño en un programa computacional. Esta medida protege otras partes del programa de excesiva modificación.

constituyen un apoyo para la enseñanza que lo cual ayuda a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable" (LARMAN, 1999).

Los patrones GRASP pueden clasificarse en Experto en información, Creador, Alta cohesión, Bajo acoplamiento y Controlador.

Experto en información

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

Hay que tener en cuenta que esto es aplicable mientras se estén considerando los mismos aspectos del sistema:

- Lógica de negocio
- Persistencia a la base de datos
- Interfaz de usuario

No tiene sentido considerar que una clase se debe escribir a sí misma en base de datos o formatearse para presentarse en una página HTML por el hecho de poseer los datos. Estos son elementos estructuralmente distintos y deben considerarse desde una perspectiva distinta.

Creador

Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- B contiene a A
- B es una agregación (o composición) de A
- B almacena a A
- B tiene los datos de inicialización de A (datos que requiere su constructor)
- B usa a A

El hecho de crear objetos tiene casuísticas particulares:

- Pool de Objetos
- Caches
- Instancias únicas

Estos casos son candidatos para la utilización de otros patrones más concretos (de diseño).

A la hora de crear objetos en distintos lenguajes hay que tener en cuenta sus peculiaridades.

Alta cohesión

Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Ejemplos de una baja cohesión son clases que hacen demasiadas cosas. En todas las metodologías se considera la refactorización. Uno de los elemento a refactorizar son las clases saturadas de métodos.

Ejemplos de buen diseño se producen cuando se crean los denominados "paquetes de servicio" o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia).

Bajo Acoplamiento

Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuánto software se puede extraer de un modo independiente y reutilizarlo en otro proyecto? Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia. Como ejemplo (de mal diseño), en muchos diseños Web se puede ver como se crea un servlet base con capacidades de paginación y se hereda de él para construir los demás. La paginación es un servicio que se podría usar en aplicaciones no Web, por lo que es más adecuado mantener estas capacidades en clases externas. Otro ejemplo clásico se produce cuando se pasan los objetos relacionados con la capa de presentación a la capa de negocio (HttpRequest, HttpResponse).

Controlador

Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

En RUP, al construir el modelo de análisis, existen estereotipos predefinidos que favorecen la separación de entidades, mecanismos de interfaz y mecanismos de control.

En aplicaciones Web, se tiende a separación de la lógica de presentación y de la lógica de negocio. Patrones bien conocidos como MVC o Fachada, son de amplia utilización.

1.5.2 GOF

Los patrones GOF (acrónimo de Gang of Four en español banda de los cuatro), se dividen en tres grupos fundamentales, de creación, estructura y comportamiento. Los primeros se encargan de mostrar una guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones serán normalmente resueltas dinámicamente, decidiendo que clases instanciar o sobre que objetos se delegará responsabilidades. Los de estructura se encargan de describir la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Y los de comportamiento se utilizan para manejar, organizar y combinar comportamientos.

1.6 Conclusiones parciales

En el presente capítulo se demuestra la importancia que tiene para la Aduana cubana y para todo el país un software que permita controlar los procesos de enfrentamiento a las ilegalidades asociadas a los Medios de Transporte Internacional. Se presenta un estudio de los sistemas extranjeros que trabajan con un objetivo similar, el de lograr mayor seguridad en las fronteras, pero en su mayoría son muy costosos y no realizan la totalidad de las funcionalidades requeridas. Se pudo constatar que a nivel nacional no se encuentran soluciones capaces de brindar las funcionalidades necesarias relativas al control efectivo de los MTI y si existen, solo ayudan en una que otra actividad, y lo hacen mediante la utilización de técnicas obsoletas que en la actualidad no son eficientes.

Por tales motivos se propone la realización de este subsistema informático con el uso de las tecnologías para el desarrollo de aplicaciones web, descritas en el capítulo.

CAPÍTULO 2. DISEÑO DEL SUBSISTEMA

2.1 Introducción

En el presente capítulo se describe el diseño del subsistema Control de Medios de Transporte Internacional para el sistema de Gestión Integral de Aduanas. En el mismo se abordan los patrones de diseño utilizados para la solución de la aplicación, las clases del diseño con estereotipos Web y se presentan los diagramas de vistas por escenarios correspondientes a los requisitos que se encuentran definidos en el Trabajo Diploma Análisis del subsistema Control de Medios de Transporte Internacional del sistema Gestión Integral de Aduanas (Hernández Ruiz, 2010) y documentados en el expediente del proyecto enfrentamiento y disponible para su consulta según las regulaciones de seguridad informática establecidas para los proyectos de la UCI, los cuales se listan a continuación:

- Gestionar MTI
  - Crear Ficha Técnica
  - Actualizar Ficha Técnica
- Gestionar marcaje
  - Marcar MTI
  - Eliminar Marcaje
  - Guardar Marcaje
  - Modificar Marcaje
- Insertar Incidencias
  - Insertar infracción MTI
  - Insertar incidencia persona natural
  - Insertar acta de advertencia
- Registrar Resultados
- Intento de entrada al país
- Buscar historial
- Buscar MTI
- Estudio Información Adelantada

Estos requisitos representan la solución informática que cubre las tareas o funciones que ha de realizar el subsistema, conjuntamente se presenta el diseño de la base de datos del

mismo y se describe cómo se adapta el subsistema Control de MTI a la arquitectura definida para el sistema GINA.

Los autores I. Jacobson, G. Booch y J. Rumbaugh en el libro El Proceso Unificado de Desarrollo de Software, plantean: “En el diseño se modela el sistema y se le da forma (incluida la arquitectura) para que soporte todos los requisitos y otras restricciones que se le suponen” (JACOBSON and RUMBAUCH, 2000).

El proceso de diseño de un producto de software es complejo porque hay un elevado número de factores (internos y externos) que deben ser manipulados e incorporados, los objetivos a cumplir pueden ser amplios y hay personas, organizaciones y procesos involucrados (MACAS and FIERRO, 2009).

Como resultado del diseño se pretende adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnología de interfaz de usuario, tecnología de gestión de transacciones, entre otras, crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases, así como ser capaces de descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.

2.2 Modelo de diseño

El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación.

2.3 Clases del diseño

Características de una clase de diseño

Las clases del diseño deben poseer características específicas como las que se describen a continuación:

- Completa y suficiente una clase de diseño debe ser la encapsulación completa de todos los atributos y métodos que se pueden esperar, en forma razonable, que existan para la clase, es decir, que debe contener los métodos aquellos que sean suficientes para lograr el objetivo ni más ni menos.
- Primitivismo, los métodos asociados a una clase de diseño deben enfocarse en el cumplimiento de un servicio para la clase. Una vez que el servicio ha sido implementado con un método, la clase no debe proporcionar otra forma de complementar la misma.
- Cohesión alta, una clase de diseño cohesiva tiene un conjunto de responsabilidades pequeño y enfocado, y aplica atributos y métodos de manera sencilla para implementar dichas responsabilidades.
- Acoplamiento bajo dentro del modelo de diseño es necesario que las clases de diseño colaboren con alguna otra. Sin embargo, la colaboración se debe mantener en un mínimo aceptable. Si un modelo de diseño tiene un acoplamiento alto, el sistema es difícil de implementar, probar y mantener a través del tiempo. En general las clases de diseño dentro de un subsistema deben tener solo un conocimiento limitado de las clases en otros subsistemas. Esta restricción, llamada la Ley de Deméter, sugiere que un método solo debe enviar mensajes a métodos de clases vecinas.

2.3.1 Extensiones para el diseño Web

Los mecanismos de extensibilidad incorporados permiten a UML ser una especie de especificación abierta para cubrir determinados aspectos de modelado. Estos mecanismos permiten extender la notación y semántica de UML.

Los estereotipos son el mecanismo de extensibilidad incorporado más utilizado dentro de UML. Un estereotipo representa una distinción de uso, puede ser aplicado a cualquier elemento de modelado.

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

Estereotipo	Imagen	Descripción
Server page		Representa una página Web dinámica que contiene el código ensamblado por el servidor cada vez que se solicita. Típicamente, una página servidora contiene scripts que se ejecutan en el servidor y que actúan recíprocamente con los recursos del lado del servidor estos son: las bases de datos, los componentes de la lógica del negocio, sistemas externos, y así sucesivamente.
Client page		Representa una instancia de una página cliente. Es una página Web con formato HTML. Las páginas cliente son interpretadas y mostradas por los navegadores del cliente y además pueden contener scripts que se interpretan en el navegador.
Form		Simboliza un formulario, es el elemento encargado de realizar envíos a las páginas servidoras.

Tabla 1: Estereotipos Web

Relaciones que se establecen entre las clases que conforman la extensión UML para Web.

Hasta/ Desde	Server page	Client page	Form
Server page	<<Redirect>>	<<Build>>, <<Redirect>>	--
Client page	<<Link>>, <<Redirect>>	<<Link>> , <<Redirect>>	Contiene
Form	<<Submit>>	Agregado por	--

Tabla 2: Relaciones entre las clases que conforman la extensión UML para Web.

El autor Jim Conallen en su libro “Buildin Web Applications with UML Second Edition”, presenta las siguientes descripciones para los estereotipos utilizados en las relaciones entre las clases.

Estereotipo	Descripción
<<Link>>	Representa una relación entre una página del cliente y un recurso del lado del servidor, o una página Web.
<<Build>>	Representa una relación direccional entre una página servidora y cliente. Esta relación identifica la salida HTML de la ejecución de una página servidora.
<<Submit>>	Relación directa entre un formulario <<HTML form>> y una página servidora. Similar a la relación <<Link>>, pero solo referencia a recursos del lado servidor. Sin embargo, cuando los recursos son pedidos desde el servidor, todos los campos del formulario son enviados al servidor junto con la petición, donde son procesados.
<<Redirect>>	Relación direccional entre páginas clientes, páginas servidoras y unas con otras. Esta asociación indica un comando a la página cliente para realizar petición de otro recurso.
<<Include>>	Asociación direccional desde una <<server page>> a otra <<server page>> o <<client page>>. Esta asociación indica que las páginas incluidas son procesadas mientras que la página se ensambla.

Tabla 3: Especificación de las relaciones.

2.3.2 Clases del diseño con estereotipos Web

Una clase de diseño y sus objetos participan en varias realizaciones de requisitos funcionales. También puede suceder que algunas operaciones, atributos y asociaciones sobre una clase específica sean solo relevantes para una sola realización de un requisito. Para manejar todo esto se utilizan los diagramas de clases conectados a una realización de requisito, mostrando así sus clases participantes, subsistemas y sus relaciones.

2.3.3 Descripción de los Procesos a través de los Diagramas de Clases del Diseño

La utilización del *framework* Symfony y el uso de las librerías ExtJS permiten desarrollar un diseño con una estructura similar para varios procesos y lograr una relación entre cada uno de los diseños y una armonía en la solución visual que debe presentarse. La página servidora Action.class es responsable de las peticiones que realiza el controlador frontal y

sus respectivas manipulaciones para brindar una respuesta correcta y precisa; luego estas se redireccionan al Layout. El Layout carga en el cuerpo la plantilla (como se trata de un menú de procesos esta página carece de contenido visual descriptivo y no requiere de diagramización), luego se construye la página cliente que importa todas las interfaces de usuario del módulo según sea la necesidad, estas interfaces están definidas en el fichero de configuración view.yml y se encuentran representadas dentro del paquete Interfaces JS.

Diagramas de clases del diseño por requisito:

Intento de entrada al país

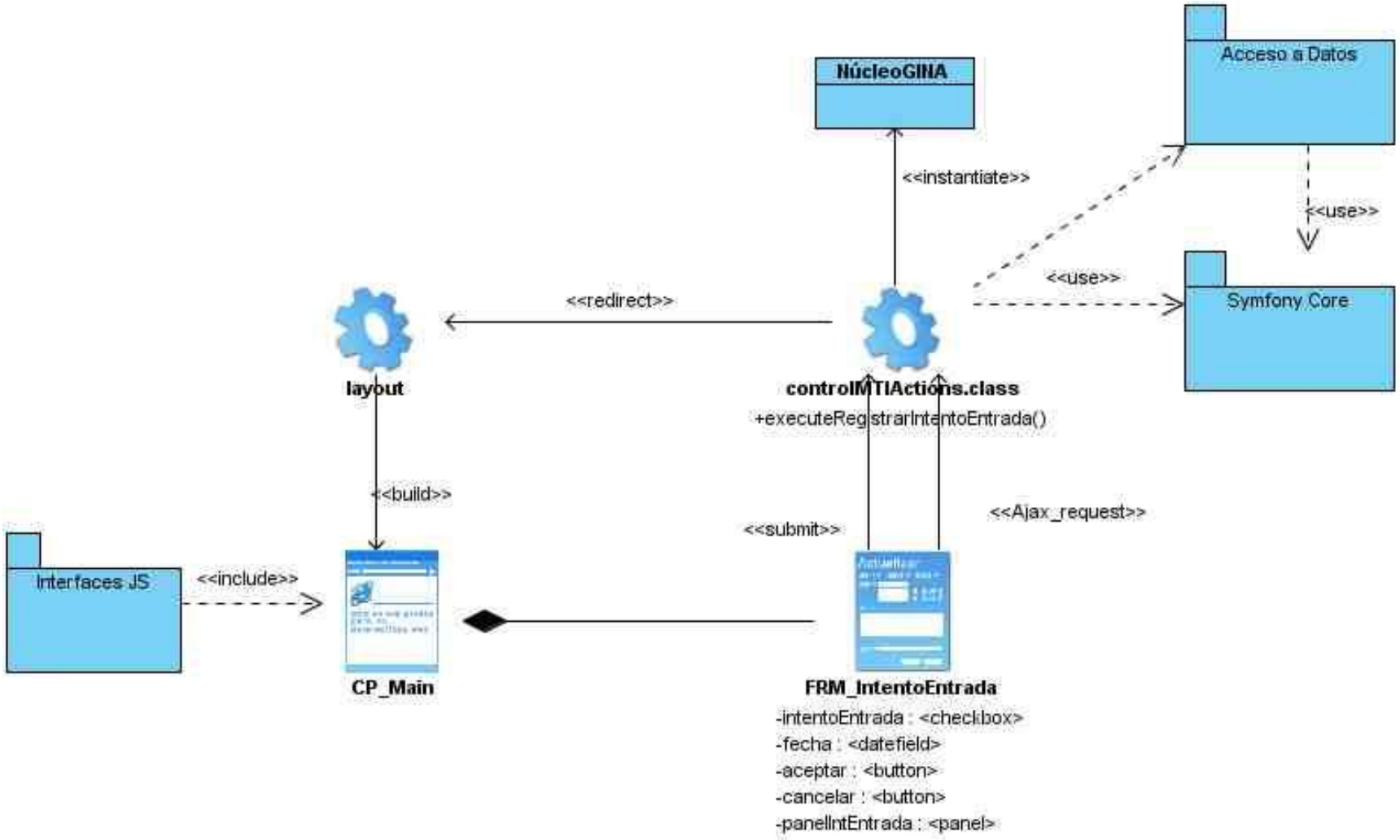

Figura 3: Diagrama de Clases Registrar Intento de Entrada al país.

Buscar MTI

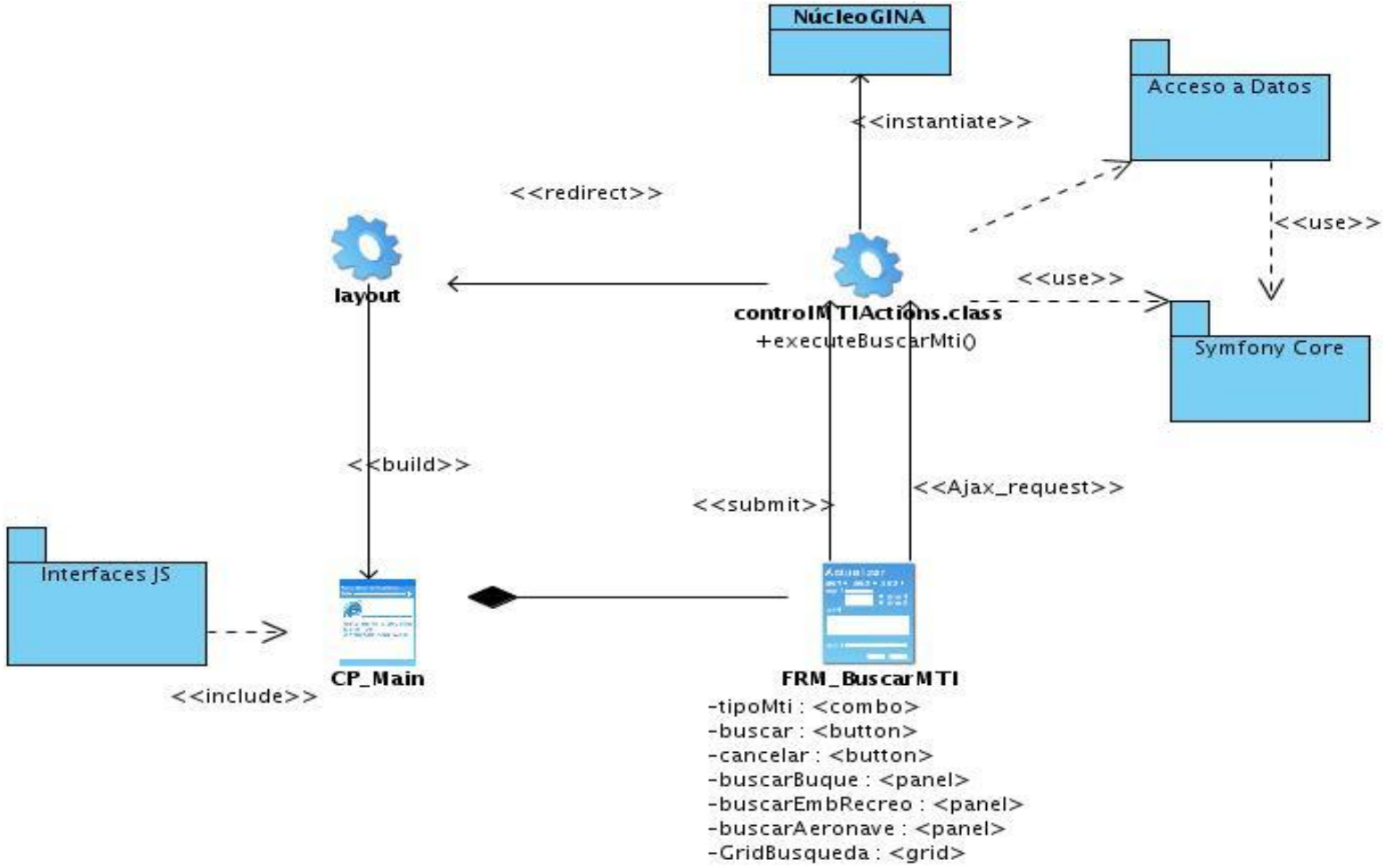

Figura 4: Diagrama de Clases Buscar MTI.

Registrar resultados

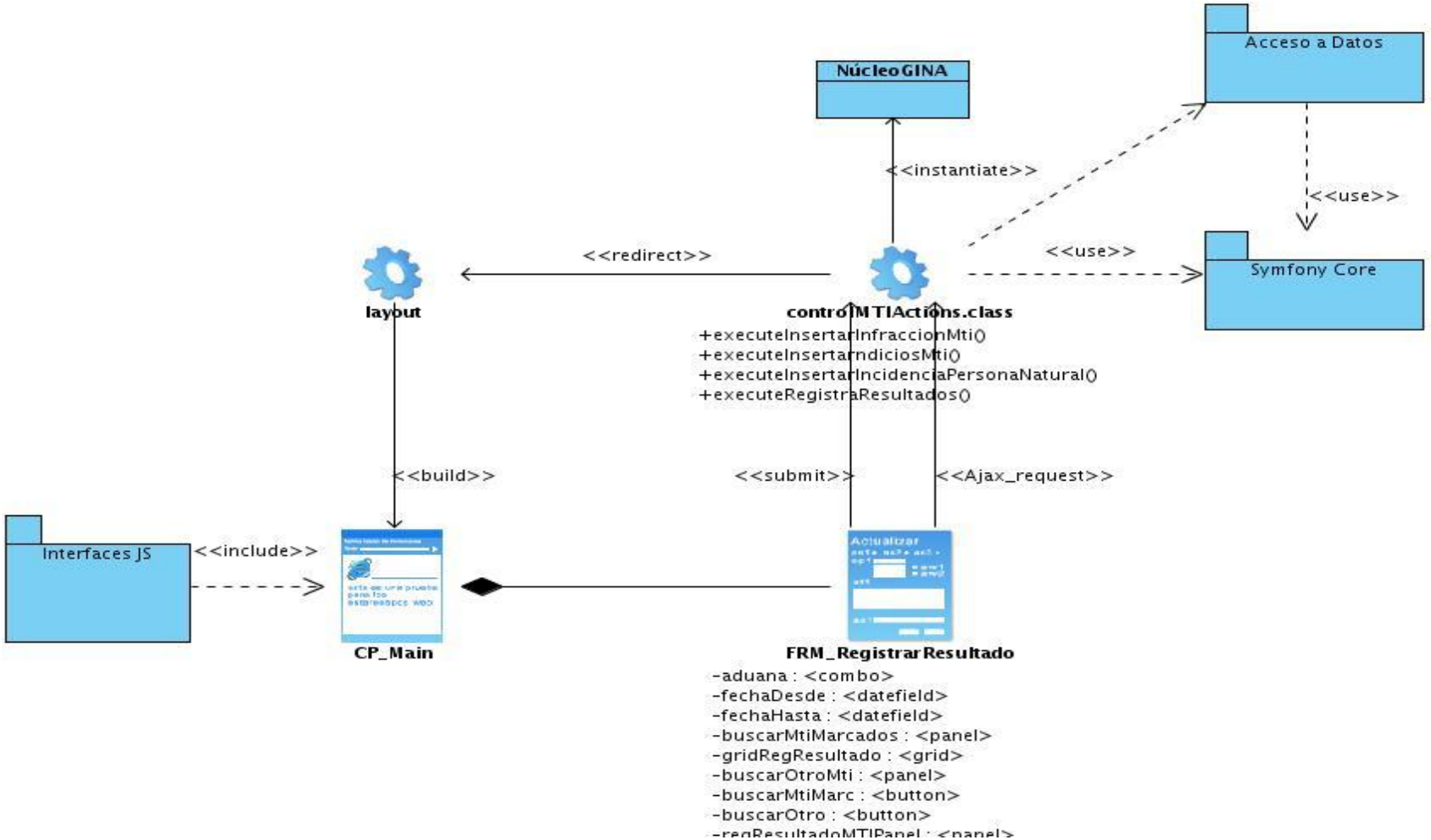

Figura 5: Diagrama de Clases Registrar resultados.

Gestionar Marcajes

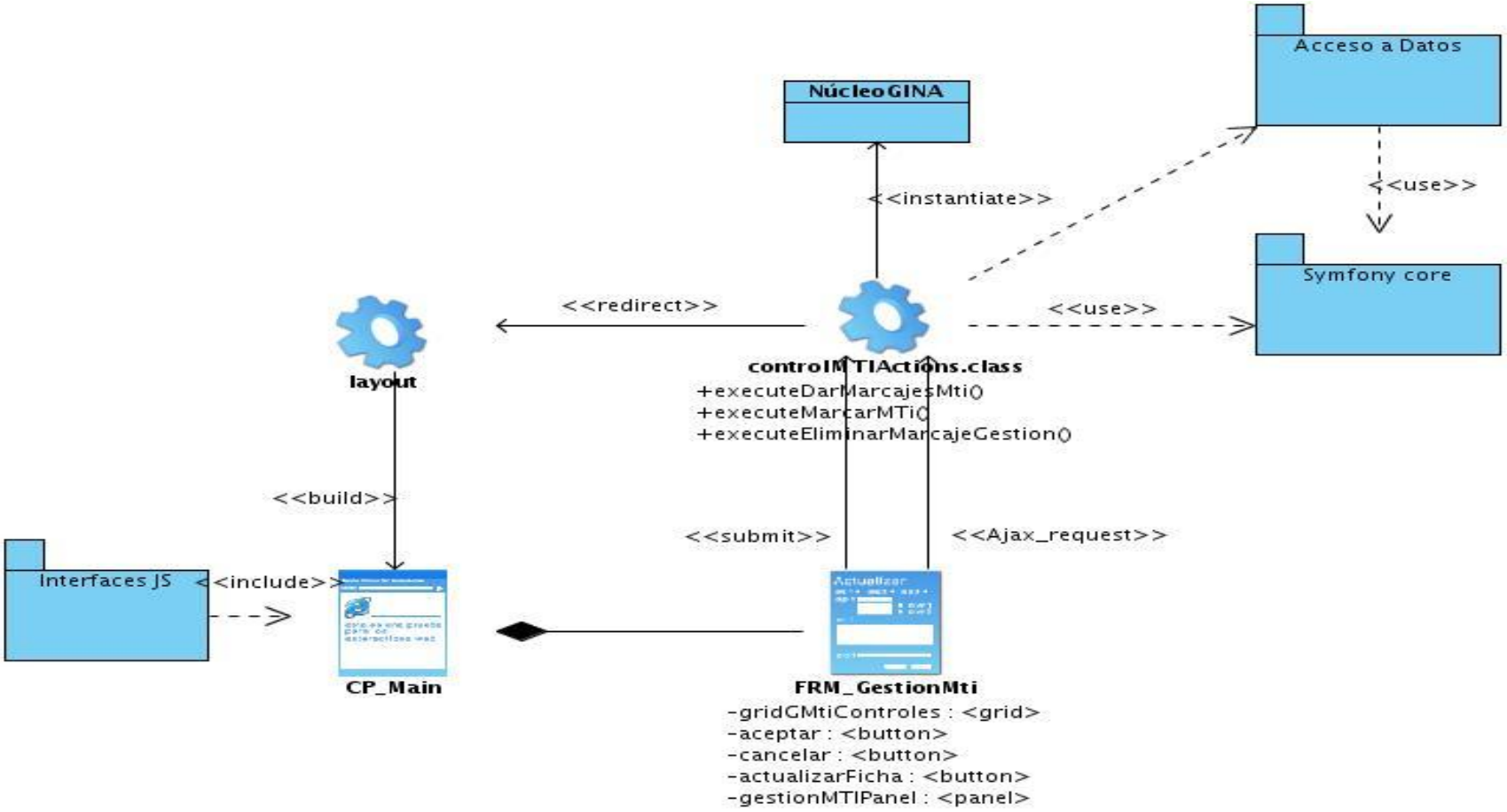

Figura 6: Diagrama de Clases Gestionar marcajes.

Los restantes diagramas de Clases del diseño se encuentran anexados en el Modelo de diseño perteneciente a la documentación del proyecto LCF.

2.3.4 Patrones de diseño utilizados

Para el desarrollo de este subsistema se utilizó el marco de trabajo Symfony, debido a que el mismo automatiza muchas de las funcionalidades más comunes para el desarrollo de software, lo cual proporciona ventajas significativas para los desarrolladores. Este *framework* es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño y arquitectónicos más utilizados en la actualidad, ya que el mismo *framework* los utiliza (Céspedes Basteiro and Rodríguez Pérez, 2009)

2.3.5 Implementación de patrones GRASP

Creador

La clase controlMtiActions contiene las acciones definidas para el subsistema Control de Medios de Transporte Internacional y es en ella misma donde se ejecutan las funciones que dan vida al sistema. En esta clase las acciones se encargan de crear los objetos de las clases que representan las entidades, evidenciando de este modo que la clase controlMtiActions es el “creador” de las entidades.

Experto

Este patrón se pone de manifiesto mediante Propel que es la librería externa que utiliza Symfony para realizar su capa de abstracción al modelo de datos, encapsulando toda la lógica de los datos y generando las clases con todas las funcionalidades comunes de las entidades. Por tanto cada clase creada por Propel a partir de una entidad es experta en manejar su información.

Alta Cohesión

Symfony permite la asignación de responsabilidades con una alta cohesión, por ejemplo la clase controlMtiActions tiene la responsabilidad para definir las acciones sobre las plantillas y colabora con otras para realizar diferentes operaciones e instanciar objetos, es

decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (lcf_dev.php), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Bajo Acoplamiento

En el diseño de las clases se aplicó este patrón logrando un modelo con las relaciones necesarias que impulsan la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a resultados negativos. De esta forma se tiene un diseño de clases más independiente que reduce el impacto al cambio.

2.3.6 Utilización de patrones GOF

Instancia única (Singleton)

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Es el caso del controlador frontal, donde hay una llamada a la función sfContext::getInstance() que garantiza que siempre se acceda a la misma instancia.

Fábrica abstracta (Abstract Factory)

Se utilizó este patrón para trabajar con objetos de distintas familias de manera que no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el *framework* necesita por ejemplo crear un nuevo objeto, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Decorador (Decorator)

Añade funcionalidad a una clase, dinámicamente. El archivo layout.php, que 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 contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

2.3.7 Diagrama de paquetes

En el Lenguaje Unificado de Modelado (UML), las abstracciones que organizan un modelo son denominados paquetes. Un paquete es un mecanismo de propósito general para organizar los elementos en grupos, los paquetes ayudan a organizar los elementos en los modelos con el fin de comprenderlos con mayor facilidad, estos también permiten controlar el acceso a sus contenidos para controlar las líneas de separación de la arquitectura del sistema como pueden ser: los elementos estructurales, los elementos de comportamiento, e incluso otros elementos de agrupación pueden incluirse en un paquete.

Se pueden emplear los paquetes para presentar diferentes vistas de la arquitectura del sistema. Un paquete es una parte de un modelo, cada parte del modelo debe pertenecer a un paquete, UML no impone una regla para componer los paquetes, estos ofrecen un mecanismo general para la organización de los modelos y/o subsistemas agrupando elementos de modelado, cada paquete corresponde a un submodelo (subsistema) del modelo (sistema), por lo que un paquete puede contener otros elementos, incluyendo clases, interfaces, componentes, nodos, colaboraciones, casos de uso, diagramas e incluso otros paquetes.

Existe una dependencia entre dos elementos del diagrama si los cambios a la definición de un elemento pueden causar cambios al otro. La dependencia entre paquetes puede ser por acceso o por importación. El acceso ocurre cuando un paquete accede a elementos de otro paquete sin ser necesario que realice cambios en el mismo y la importación ocurre cuando es necesario contener todos los elementos del otro paquete para poder realizar cambios en los mismos.

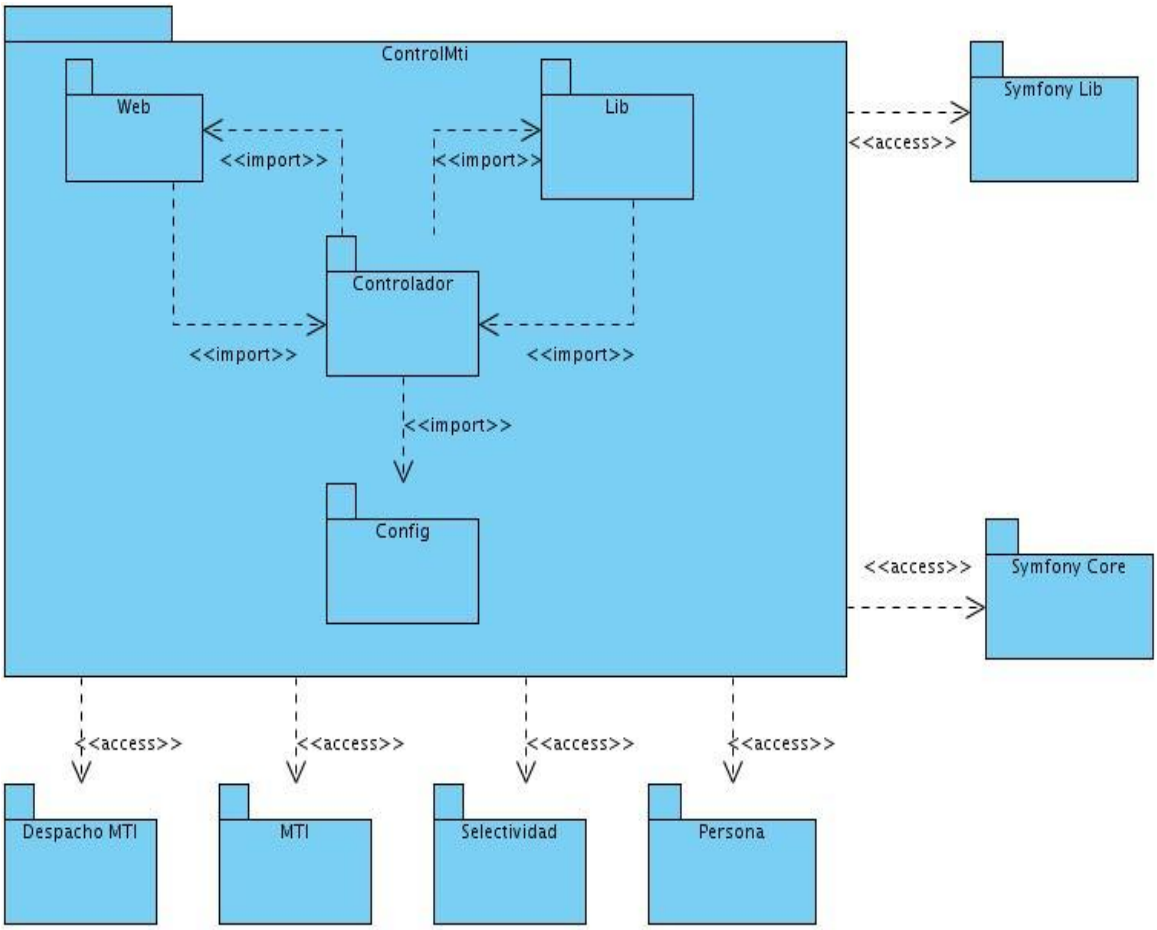

Figura 7: Diagrama de paquetes del subsistema Control MTI.

2.3.8 Diagrama de paquetes Acceso a Datos e Interfaces JS

El paquete "Acceso a Datos" contiene tres paquetes Objetos, Objetos Peer y Propel. Dentro de estos paquetes están las clases necesarias para efectuar la conexión y el intercambio de información de la aplicación con la 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*. Por su parte el paquete Objetos tiene las clases de la lógica de la aplicación que contienen los atributos y métodos necesarios que mediante el uso de Propel permiten la inserción y actualización de información persistente.

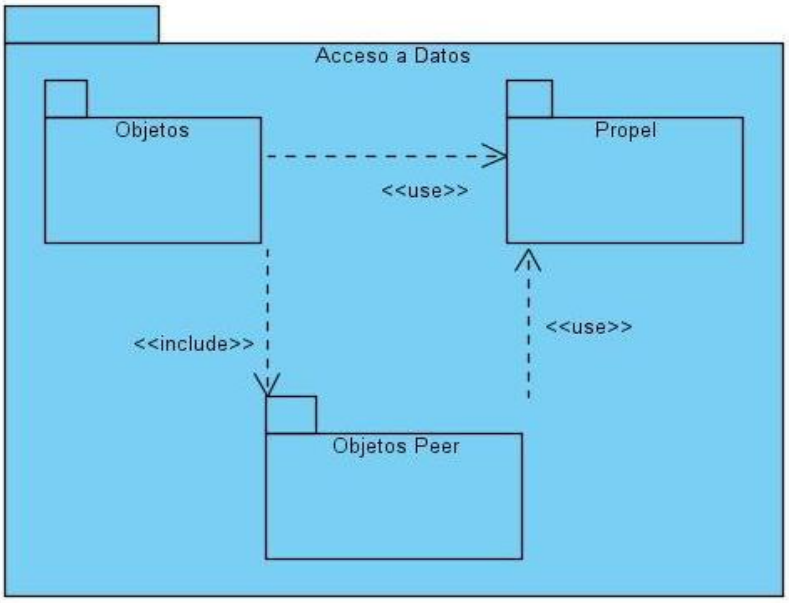

Figura 8: Diagrama de paquetes Acceso a Datos.

Por cada tabla de la base de datos, existen 2 clases que pertenecen al paquete Objetos, por ejemplo: la clase TcLineaEnfrentamiento hereda de BaseTcLineaEnfrentamiento, de esta forma se logran los objetivos (inserción y actualización) sólo modificando la clase hija, ya que la clase base tiene las funcionalidades para el acceso a la base de datos con el uso del paquete Propel e incluyendo del paquete Objetos Peer su clase correspondiente "Base<clase>Peer", en este ejemplo es BaseTcLineaEnfrentamientoPeer. Las clases del paquete Objetos Peer contienen los métodos y atributos para efectuar consultas a la base de datos, aquí también por cada una de las tablas en la base de datos se presentan dos clases siguiendo el mismo ejemplo se tendría que TcLineaEnfrentamientoPeer hereda de BaseTcLineaEnfrentamientoPeer, así solo se modificará la clase hija y se podrán utilizar todos los métodos y atributos de la clase padre para lograr las consultas (Céspedes Basteiro and Rodríguez Pérez, 2009).

El paquete "Interfaces JS" contiene los scripts Ext-base.js y Ext-all.js los cuales representan el núcleo de las librerías ExtJS utilizadas, y los otros constituyen las interfaces del subsistema control de MTI, estas contienen códigos Java Script que son ensamblados por el navegador una vez que la página cliente se está interpretando.

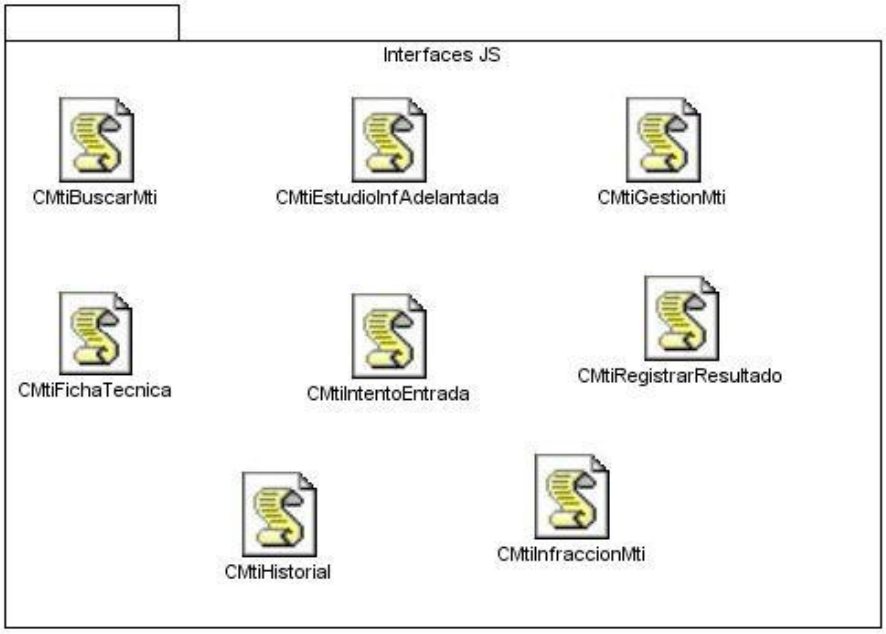

Figura 9: Diagrama de paquetes Interfaces JS.

2.4 Diagramas de Secuencia orientado a actividades

Los diagramas de secuencia orientados a actividades plantean dos tipos de soluciones para los dos casos que representan, el Diagrama Visual y el Diagrama del negocio. Estos diagramas tienen el mismo objetivo con la diferencia que del negocio se obtiene más provecho de los datos almacenados y requiere de mucha más profundidad en el alcance de los métodos mientras que en el visual se tratan las funcionalidades necesarias para incorporarle a la vista del negocio elementos que permitan la selección de los datos para la ejecución de los diagramas del sistema y no requieren de la petición a través de datos del controlador frontal para su funcionamiento.

Diagrama de Secuencia orientado a actividades Visual

Este diagrama es más rápido y corto en su ejecución ya que necesita responder a los diagramas del negocio. Trabaja fundamentalmente en la capa visual del Symfony.

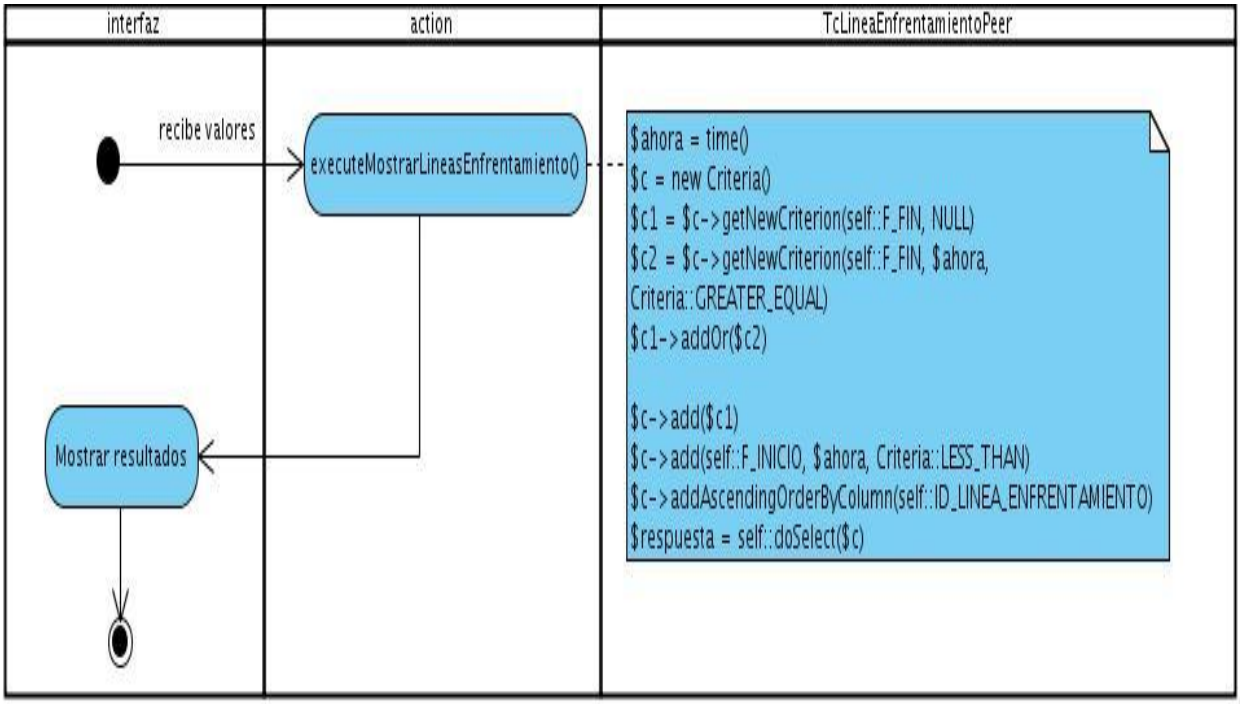

Figura 10: Diagrama de Secuencia orientado a actividades Visual Mostrar Líneas de Enfrentamiento.

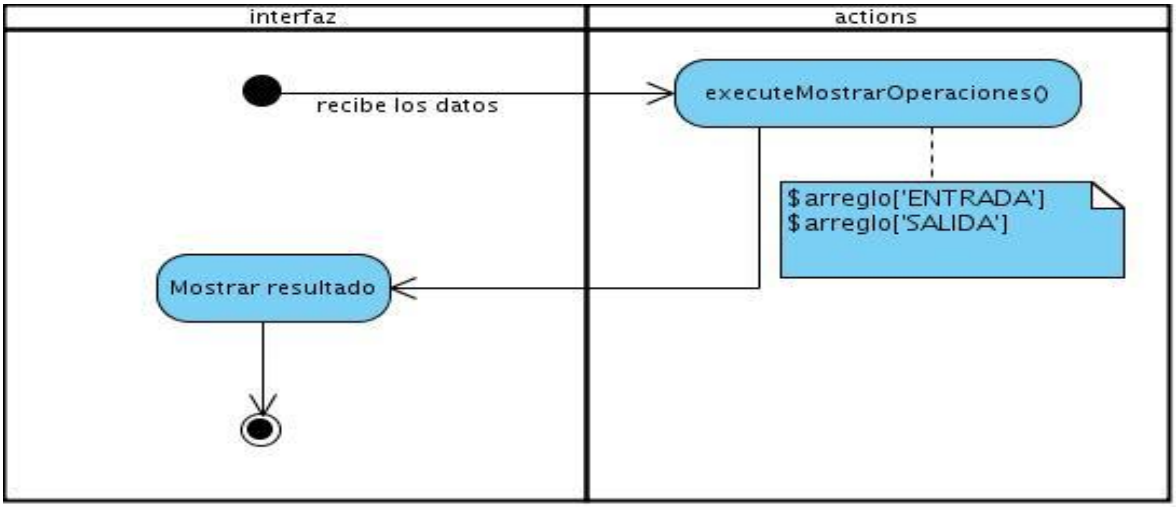

Figura 11: Diagrama de Secuencia orientado a actividades Visual Mostrar Operaciones.

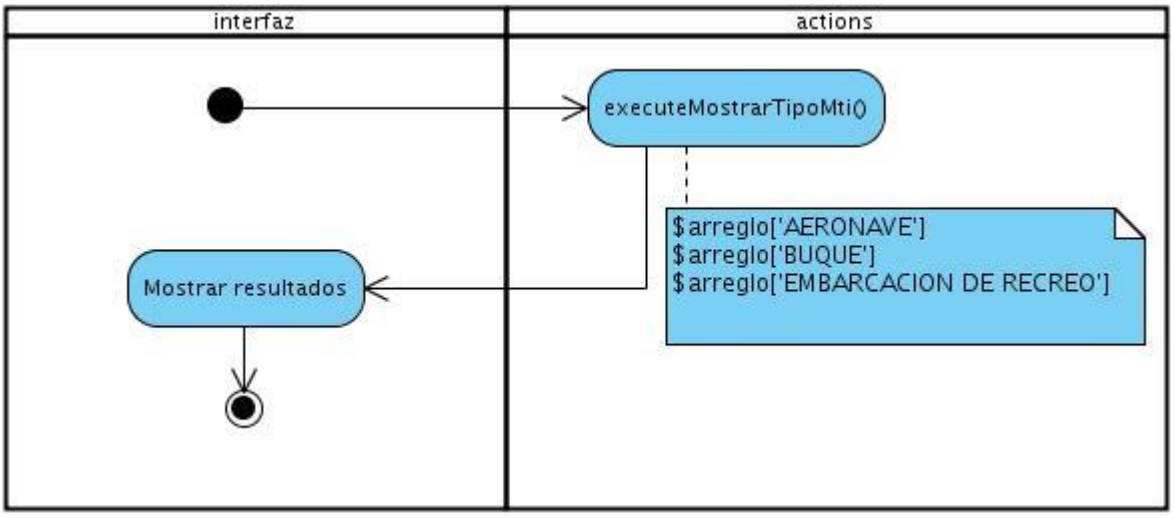

Figura 12: Diagrama de Secuencia orientado a actividades Visual Mostrar Tipo MTI.

Diagrama de Secuencia orientado a actividades Negocio

Este diagrama representa el diseño de los requisitos funcionales capturados en fases anteriores y evidencia paso a paso la interacción entre las tres capas de la arquitectura y los pasos que debe seguir el programador para darle respuesta a los requerimientos. También presenta un nivel superior de interacción con los datos del sistema.

Buscar MTI

Este requisito usa un componente implementado en el núcleo del sistema, el cual se encarga de realizar la búsqueda de los MTI por tipos.

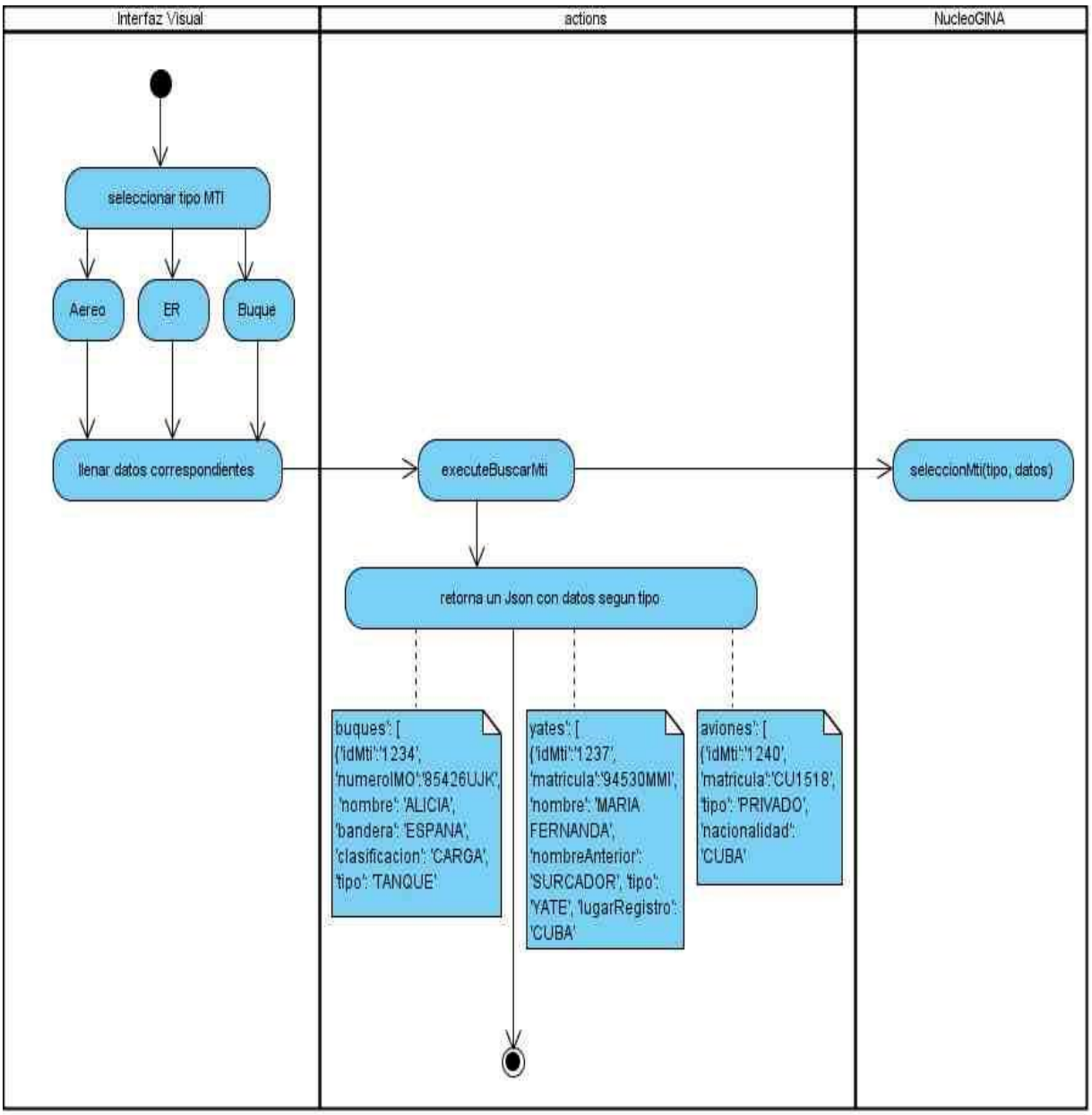

Figura 13: Diagrama de Secuencia orientado a actividades Buscar MTI.

Buscar Historial

Este requisito usa el componente dar datos MTI implementado en el núcleo del sistema GINA.

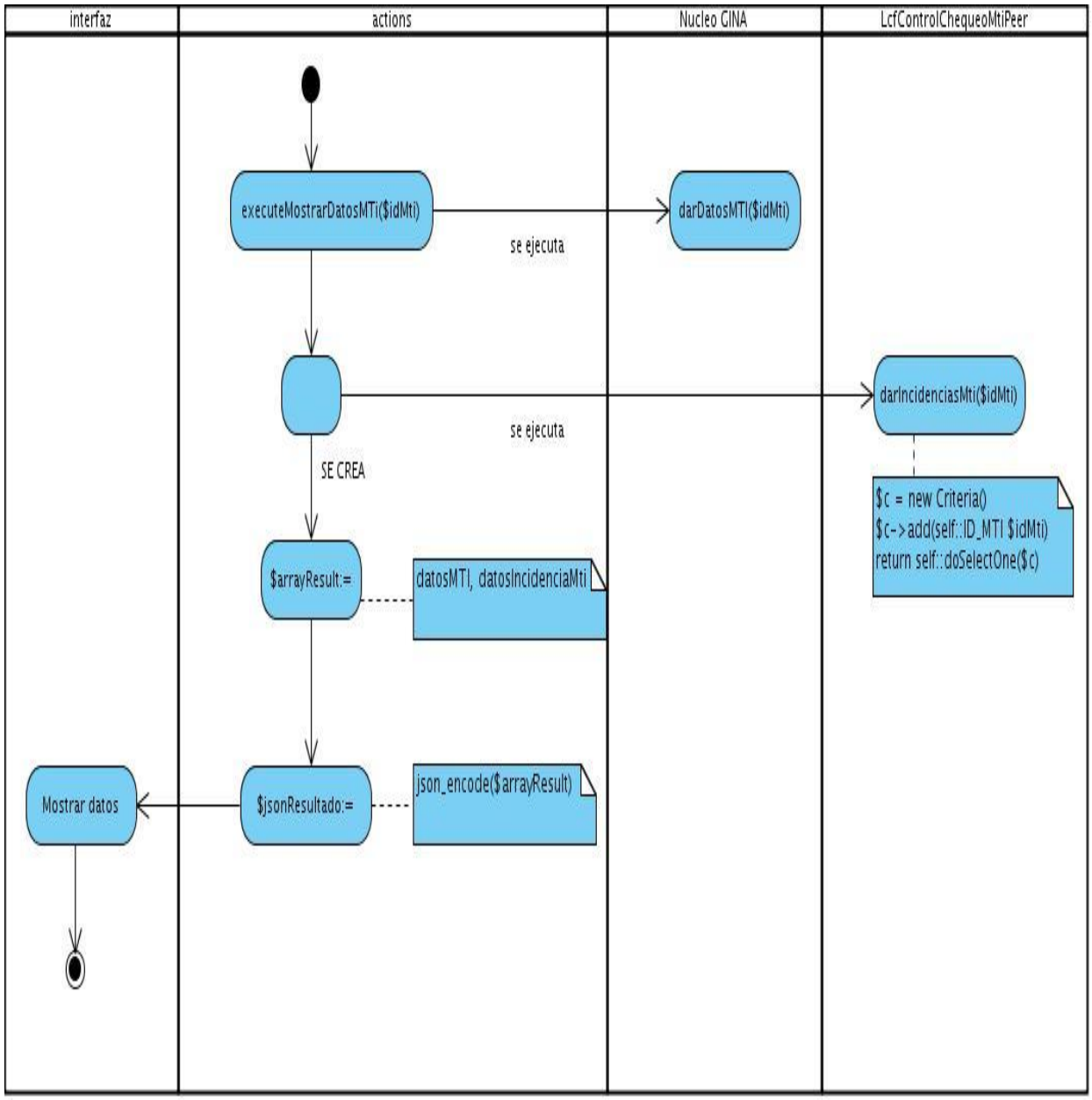

Figura 14: Diagrama de Secuencia orientado a actividades Buscar Historial.

Realizar Estudio de Información Adelantada

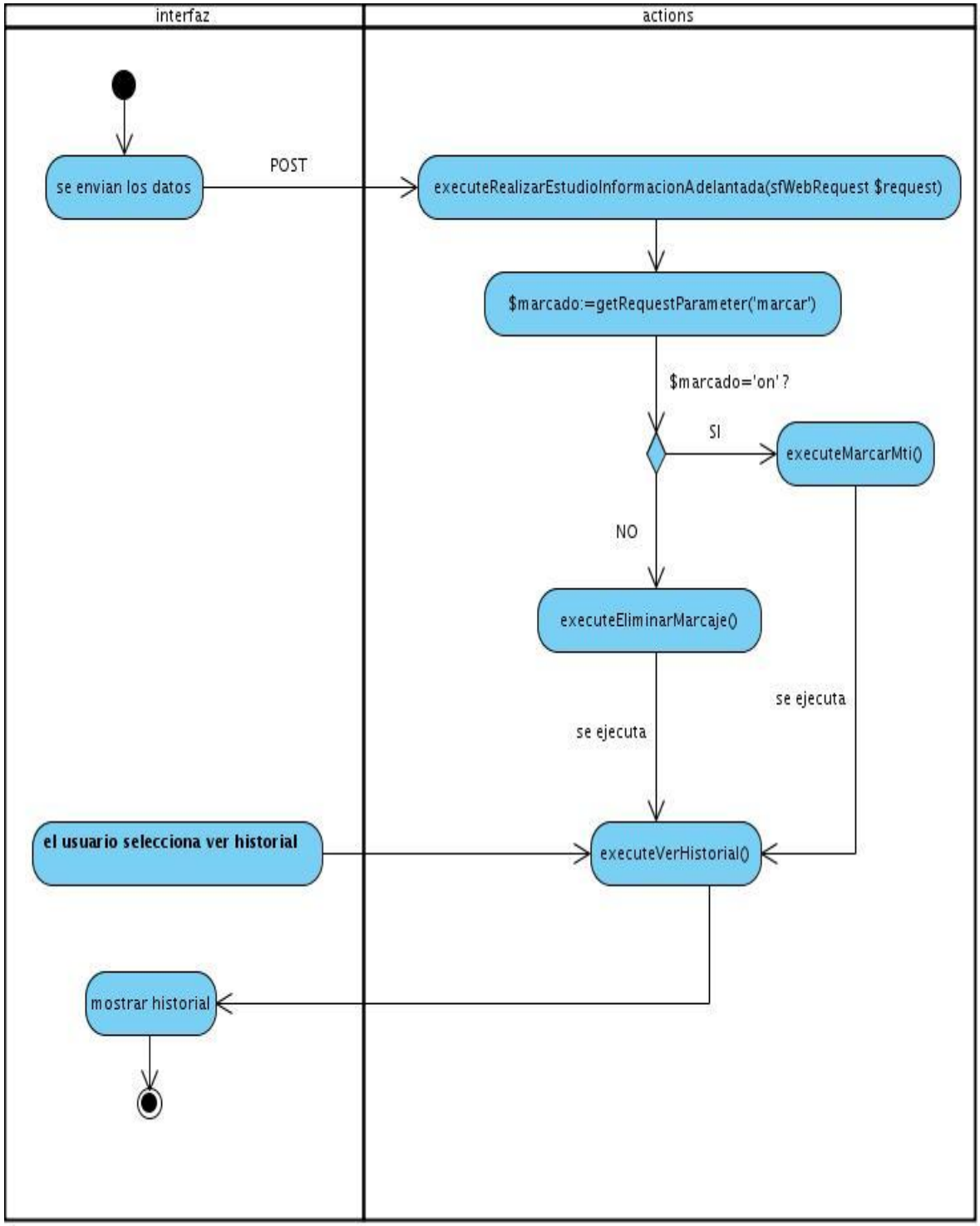

Figura 15: Diagrama de Secuencia orientado a actividades Realizar Estudio de Información Adelantada.

Registrar intento de entrada al país

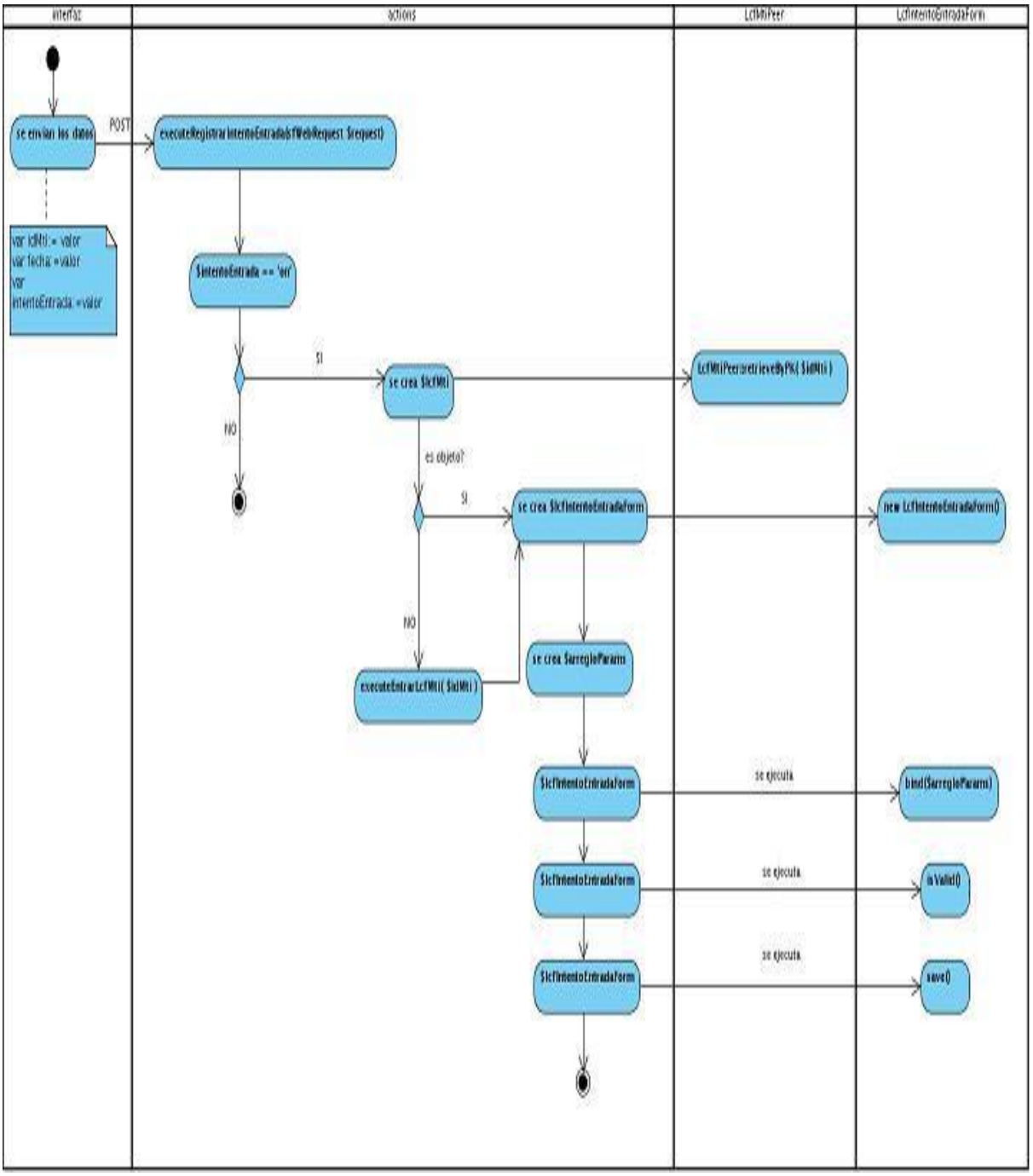

Figura 16: Diagrama de Secuencia orientado a actividades Registrar intento de entrada al país.

Todos los diagramas responden a la descripción realizada en el requisito funcional y da cumplimiento a las peticiones. Los restantes se encuentran anexados en el Modelo de diseño realizado perteneciente a la documentación del proyecto LCF.

2.5 Diseño de la Base de Datos

En el proceso y construcción de todo sistema informático, el diseño de la Base de Datos ocupa un lugar importante, definiéndose como un proceso relativamente independiente dentro del diseño del sistema y compuesto por una serie de etapas.

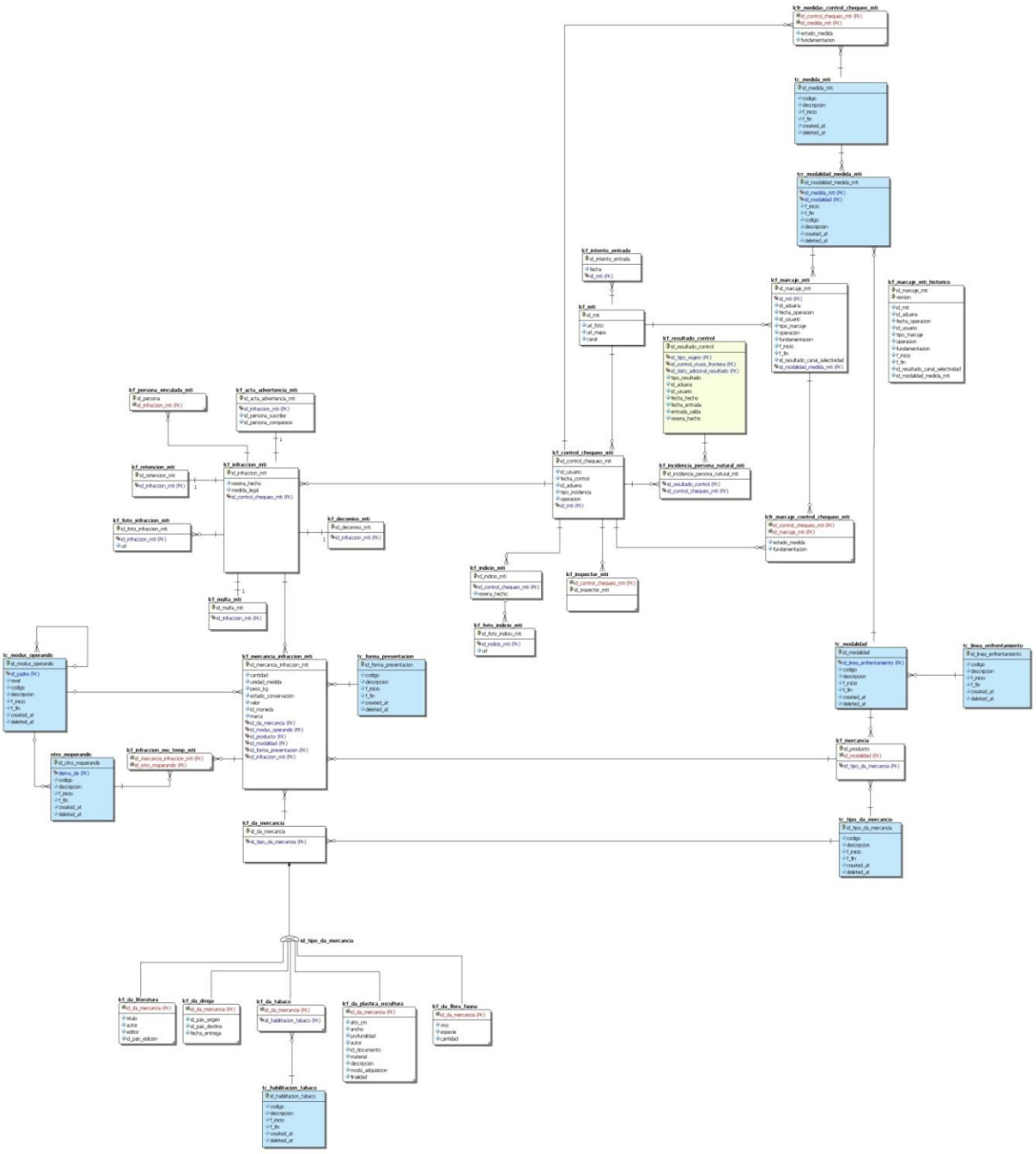

Figura 17: Modelo de datos del subsistema Control MTI.

Este modelo está basado en una percepción del mundo real que consta de un conjunto de objetos básicos llamados entidades con sus atributos y de las interrelaciones que existen entre estos objetos. Es usado para describir la representación lógica y física de la información persistente manejada por el sistema.

2.6 Diagrama de despliegue

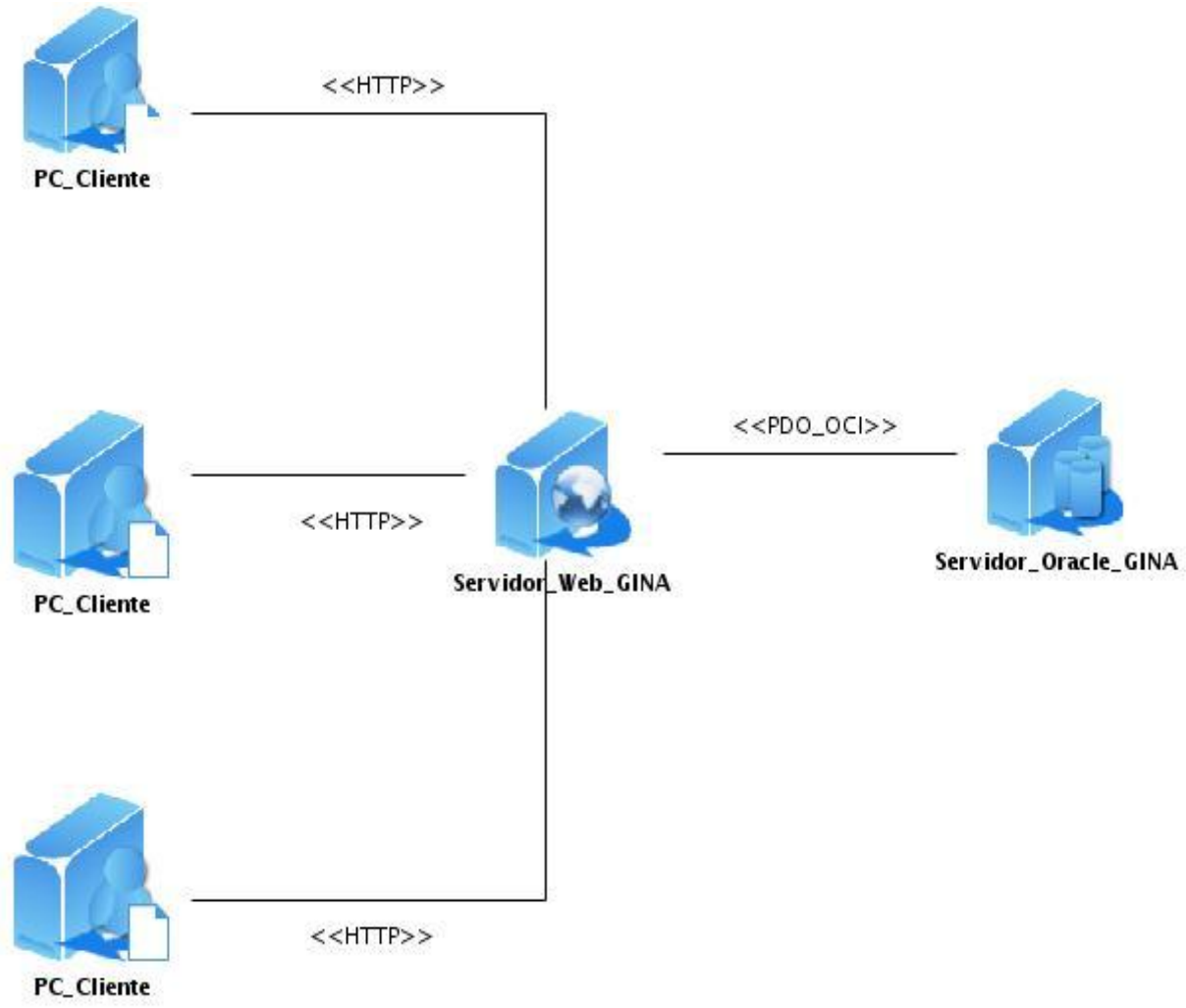

Figura 18: Diagrama de despliegue del subsistema Control MTI

2.7 Conclusiones parciales

Con el desarrollo de este capítulo se logró modelar el sistema de forma que soporte los requisitos funcionales y no funcionales, se obtuvieron los artefactos del diseño del subsistema, se realizaron las clases del diseño con estereotipos Web, diagramas de secuencia así como el diseño de la Base de Datos el objetivo de lograr un almacenamiento adecuado de la información que se maneja en el subsistema. Estos artefactos servirán de entrada para el flujo de implementación, facilitando un mayor entendimiento de la aplicación a los desarrolladores.

CAPÍTULO 3. VALIDACIÓN DEL DISEÑO DEL SUBSISTEMA

3.1 Introducción

Desde los albores de la Informática se han tratado de medir de una manera u otra, distintos atributos del software. Estos atributos pueden ser el tamaño, la complejidad, la frecuencia esperada de aparición de errores, cobertura de pruebas, o incluso atributos del proceso software como pueden ser la productividad. Dichos atributos se pueden medir directa o indirectamente, esto es, a través de la medición previa de otros atributos con los que se sabe que aquellos tienen una relación matemática. Los objetivos perseguidos por dichas mediciones pueden ser la búsqueda de un método de estimación de esfuerzo de desarrollo, el cálculo de la cobertura de pruebas en el aseguramiento de la calidad o como en este caso, la calidad en el diseño software (Olmedilla Arregui, 2005).

El término “Métricas del Software” comprende muchas actividades, todas ellas relacionadas de alguna manera con la idea de mejorar la calidad del software. Cada una de esas actividades ha evolucionado por sus propios medios dentro de un dominio más amplio que las abarca a todas: la Ingeniería del Software (Lorenz and Kidd, 1994).

Hay varias razones que justifican el uso de las métricas en el proceso de desarrollo de software. Por un lado se dice que cuando se puede medir aquello de lo cual se está hablando y se puede expresar en números, se sabe realmente acerca de ello; pero cuando no puede medirse, y no puede expresarse en números, el conocimiento que se tiene de ello es escaso e insatisfactorio.

El contar con datos estadísticos de métricas de software, da un panorama de situaciones reales que ayudan aplicar y dar seguimiento a las diferentes formas de evaluar y determinar métricas de calidad para un mejor desempeño en la calidad de software.

Este capítulo se basa en el uso de métricas que ayuden a la detección de errores y a la mejora de la calidad en el diseño en fases tempranas o sea antes que dicho diseño se haya implementado en código. Interesa saber si el diseño del subsistema Control de Medios de Transporte Internacional para el sistema de Gestión Integral de Aduanas se ajusta al nivel de calidad fijado por la Aduana.

3.2 Métricas para el Modelo de Diseño

Se conoce que las medidas y las métricas son componentes claves de cualquier disciplina de la ingeniería. Las métricas para diseño proporcionan al diseñador una mejor visión interna y ayudan a que el diseño evolucione a un nivel superior de calidad.

Para la aplicación de estas métricas se definen cinco características fundamentales:

Localización: es una característica del software que indica la forma que se concentra la información dentro de un programa. Dado que el software convencional hace hincapié en las funciones como mecanismos de localización, las métricas de software se han centrado en la estructura interna o complejidad de las funciones (longitud del módulo, cohesión, o complejidad ciclomática) o bien en la forma en que las funciones se conectan entre sí (acoplamiento de módulos).

Encapsulamiento: se define el encapsulamiento como “el empaquetamiento (o enlazado) de una colección de elementos. El encapsulamiento influye en las métricas cambiando el objetivo de la medida, que pasa de ser un único módulo a ser un paquete de datos (atributos) y de módulos de procesamiento (operaciones). Además, el encapsulamiento impulsa a la medida hasta un nivel de abstracción más elevado.

Ocultamiento de la información: suprime los detalles operativos de un componente de un programa. Tan sólo se proporciona la información necesaria para acceder a ese componente o a aquellos otros componentes que deseen acceder a él.

Herencia: es un mecanismo que hace posible que los compromisos de un objeto se difundan a otros objetos. Dado que la herencia es una característica fundamental de muchos sistemas, hay muchas métricas que se centran en ella. Entre los ejemplos que se tratarán más adelante: se cuentan el número de descendientes (número de instancias inmediatas de una clase), número de predecesores (número de generalizaciones inmediatas), y grado de anidamiento de la jerarquía de clases (profundidad de una clase dentro de una jerarquía de herencia) y otros.

Abstracción: es un mecanismo que permite al diseñador centrarse en los detalles esenciales de algún componente de un programa (tanto si es un dato como si es un proceso) sin preocuparse por los detalles de nivel inferior. Cuando los niveles de abstracción van elevándose, se ignoran más y más detalles, por lo tanto, se proporciona

una visión más general de un concepto u objeto. A medida que se pasa a niveles más reducidos de abstracción, se muestran más detalles, lo cual proporciona una visión más específica de un concepto.

3.2.1 Métricas orientadas a Clases

Una clase es la unidad principal de todo sistema lo cual implica que las medidas y métricas para una clase individual, la jerarquía de clases, y las colaboraciones de clases resultarán sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño.

Los umbrales que se usaron para evaluar estas métricas en el diseño propuesto fueron planteados por los especialistas del proyecto de Planificación de Recursos Empresariales (ERP) por sus siglas en inglés, perteneciente al CEIGE. Son necesarios los valores de los umbrales ya que estos han sido para los parámetros de calidad, una polémica a nivel mundial en el diseño de sistemas.

3.2.1.1 Métricas propuestas por Lorenz y Kidd

En el libro de métricas realizado por Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones a lo largo y ancho de la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código, y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización.

Tamaño de clase (TC)

Para realizar la medición del tamaño general de una clase, Lorenz y Kidd proponen una serie de medidas a emplear. Para evaluar el diseño propuesto solo se utilizó el número total de operaciones que están encapsuladas dentro de la clase.

Esta métrica plantea que si existen valores grandes de TC estos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor

medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

Para evaluar esta métrica se tuvieron en cuenta la cantidad de procedimientos por clase y se obtuvo el siguiente resultado: para un total de 22 clases que definen operaciones, existe un promedio de operaciones de un 2.18 como se muestra en la tabla 4.

Total de clases	22
Promedio de procedimientos	2.18

Tabla 4: Promedio de procedimientos por clase.

Teniendo en cuenta la tabla anterior se aplicaron los umbrales de métricas para dar una categoría a las clases según la cantidad de procedimientos presentes en las mismas.

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

Tabla 5: Umbrales para Responsabilidad

Complejidad Implementación	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	>2*Prom.

Tabla 6: Umbrales para la Complejidad de la implementación

Reutilización	Baja	>2*Prom
	Media	Entre Prom. y 2*Prom
	Alta	<= Prom

Tabla 7: Umbrales para Reutilización

La aplicación de la métrica TC al diseño propuesto para el subsistema arrojó los resultados que se observan a continuación:

De un total de 22 clases que definen operaciones existe un 9% de clases con alta responsabilidad, un 5% de clases con un nivel medio de responsabilidad y un 86% de clases con responsabilidad baja.

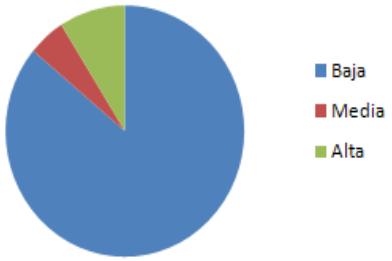

Figura 19: Por ciento de Responsabilidad.

De total de clases que definen operaciones el 86% tiene baja complejidad, existe un 5% de clases de media complejidad, así como un 9% de clases con alta complejidad.

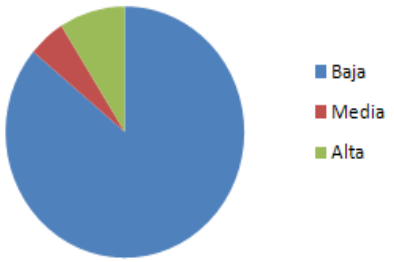

Figura 20: Por ciento de Complejidad.

Del total de las 22 clases que definen operaciones los resultados arrojados por los umbrales fueron: un 9% de clases con baja reutilización, un 5% de las mismas presentan un porciento de reutilización medio, así como el 86% restante que representa las clases de alta reutilización.

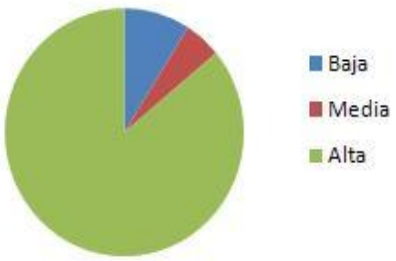

Figura 21: Por ciento de Reutilización.

Los valores obtenidos al concluir la evaluación del diseño del subsistema aplicando la métrica TC demuestran que existen bajos niveles de responsabilidad y de complejidad así como un alto nivel en el indicador de reutilización, lo cual garantiza la calidad del diseño realizado.

3.2.1.2 Métricas propuestas por Chidamber y Kemener

Uno de los conjuntos de métricas de software a los que se hace más ampliamente referencia es el propuesto por Chidamber y Kemener. Estas métricas de diseño propuestas se basan en clases, a las cuales suele aludirse con el nombre de conjunto de métricas CK para sistemas.

Este conjunto de métricas están compuestas a su vez por 6 especificaciones:

- Métodos ponderados por clase
- Profundidad árbol de herencia
- Número de descendientes
- Acoplamiento entre clases
- Respuesta para una clase
- Carencia de cohesión en los métodos

Estas métricas permiten conocer hasta qué punto están bien definidas las clases y el subsistema, lo cual tiene un impacto directo en la mantenibilidad del mismo, tanto por la comprensión de lo desarrollado como por la dificultad de modificarlo con éxito. Estas están enfocadas fundamentalmente a la herencia y se aplicaron dos durante la evaluación del diseño.

Árbol de Profundidad de Herencia (APH)

Esta métrica se define como la longitud máxima desde el nodo hasta la raíz del árbol. A medida que crece el APH, es más probable que las clases de niveles inferiores hereden muchos métodos. Esto da lugar a posibles dificultades cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda (con un valor grande de APH) lleva también a una mayor complejidad de diseño. Por el lado positivo, los valores grandes de APH implican que se pueden reutilizar muchos métodos.

Para evaluar esta métrica se tuvieron en cuenta la cantidad de descendientes por clase como se muestra en la tabla 10:

Clase	Clase Padre	Niveles del árbol de herencia.
LcfDaLiteratura	LcfDaMercancía	1
LcfDaDroga	LcfDaMercancía	1

LcfDaPlásticaEscultura	LcfDaMercancía	1
LcfDaTabaco	LcfDaMercancía	1
LcfDaFloraFauna	LcfDaMercancía	1

Tabla 8: Niveles de herencia por clase

Para un total de 5 clases que son herederas, se obtuvo que todas heredan de la misma clase LcfDaMercancía, lo que resulta en el nivel 1 de la herencia, por todo esto se concluye que el APH se encuentra en un nivel bajo de profundidad de herencia lo que a su vez garantiza el bajo acoplamiento entre las clases.

Número de Descendientes (NDD)

Se denominan descendientes a las subclases que son inmediatamente subordinadas a otras. A medida que crece el número de descendientes, se incrementa la reutilización, pero también es cierto, que a medida que crece NDD, la abstracción representada por la clase predecesora puede verse diluida. Lo cual trae consigo la posibilidad de que algunos de los descendientes no sean realmente miembros propios de la clase predecesora. A medida que NDD va creciendo, la cantidad de pruebas crecerá también.

Clase	Clase Padre	Número de descendientes
LcfDaLiteratura	LcfDaMercancía	0
LcfDaDroga	LcfDaMercancía	0
LcfDaPlásticaEscultura	LcfDaMercancía	0
LcfDaTabaco	LcfDaMercancía	0
LcfDaFloraFauna	LcfDaMercancía	0

Tabla 9: Número de descendientes por clase

A continuación se plantean los umbrales definidos para esta métrica:

	Categoría	Criterio
Reutilización	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	>2*Prom.

Tabla 10: Umbrales para Reutilización

Abstracción	Indefinida	>5
	Afectada	Entre 2 y 5
	Definida	<=2

Tabla 11: Umbrales para Abstracción

	Categoría	Criterio
Cohesión	Baja	>5
	Media	Entre 2 y 5
	Alta	<=2

Tabla 12: Umbrales para Cohesión

Cantidad de pruebas	Baja	<=2

	Media	Entre 2 y 5
	Alta	>5

Tabla 13: Umbrales para Cohesión

Al aplicar estos umbrales a los resultados del análisis de los datos de la tabla 10 se obtuvieron los siguientes resultados:

De las 5 clases que heredan, ninguna de ellas tiene descendientes, lo cual trae consigo que haya una baja reutilización, en estas clases una muy definida abstracción de clases, la cohesión de la jerarquía de clases es alta y la cantidad de pruebas es baja.

En el modelo de datos elaborado la herencia que existe mantiene bajos niveles, lo cual garantiza el bajo acoplamiento entre clases ya que la colaboración se mantiene en un mínimo aceptable estableciendo solo las relaciones necesarias entre clases. Esto también facilita que no haya carencia de cohesión en los métodos.

3.2.2 Métricas orientadas a Operaciones

Dado que la clase es la unidad dominante en los sistemas, se han planteado menos métricas para las operaciones de clases. Algunos especialistas describen esto cuando plantean que los resultados de los últimos estudios indican que los métodos tienden a ser pequeños, tanto en términos del número de sentencias como en términos de su complejidad lógica, lo cual sugiere que la estructura de conectividad de un sistema pueda resultar más importante que el contenido de los módulos individuales.

Métricas propuestas por Lorenz y Kidd

Lorenz y Kidd proponen en su libro algunas métricas sencillas para el conocimiento cualitativo de la efectividad de las funcionalidades del subsistema partiendo del diseño realizado. A continuación se aplican dos de ellas.

Número Medio de Parámetros por Operación (NPavg)

Esta métrica plantea que en cuanto sea más grande el número de parámetros de la operación, será más compleja la colaboración entre objetos. En general, el NPavg debería de mantenerse tan bajo como sea posible.

En el diseño del subsistema se diseñaron las funcionalidades con la cantidad de atributos mínima y suficiente lo cual permite obtener los resultados requeridos por el requisito y evita que aumente la complejidad entre objetos.

Tamaño medio de operación (TOavg)

Esta métrica plantea que el número de mensajes enviados por la operación proporciona una alternativa para el tamaño de la operación. A medida que asciende el número de mensajes enviados por una única operación, es posible que las responsabilidades no hayan sido bien estipuladas dentro de la clase.

En las funcionalidades diseñadas para el subsistema son muy pocas las que emiten mensajes, lo cual evidencia una correcta distribución de responsabilidades dentro de las clases.

3.3 Conclusiones parciales

En este capítulo se aplicaron métricas que permiten la validación del diseño realizado, lo cual da solución al objetivo planteado. El diseño propuesto no presenta una alta complejidad lo cual permite que las pruebas no sean complejas y que pueda ser integrado al sistema GINA sin que exista ningún problema, además presenta un bajo acoplamiento pues no se usó la herencia en el diseño de las clases. Se puede concluir que el diseño obtenido presenta una calidad adecuada, lo cual facilitará la implementación en etapas posteriores.

CONCLUSIONES

Una vez concluido este trabajo se puede apreciar que a partir de la fundamentación teórica

- Se enriqueció el conocimiento sobre todo lo que respecta y se relaciona con el control de medios de transporte internacional a nivel mundial y nacional, así como los principales sistemas que existen en este ámbito y se relacionan con esta temática.
- Se estudiaron los artefactos el análisis existentes a partir de lo cual se procedió al desarrollo del diseño del subsistema Control MTI
- Se realizó la validación del mismo partiendo de métricas predefinidas por especialistas reconocidos internacionalmente lo cual arrojó un resultado de calidad aceptable para la solución propuesta.

RECOMENDACIONES

Considerando la experiencia alcanzada durante la confección de este trabajo se recomienda:

- Implementar el subsistema Control de MTI para el sistema GINA partiendo del diseño desarrollado durante este trabajo.
- Continuar con la investigación de nuevas tecnologías informáticas para garantizar mejoras en futuras versiones del subsistema Control MTI perteneciente al sistema GINA.

REFERENCIAS BIBLIOGRÁFICAS

1. ADUANA NACIONAL, B. 2004a. Manual de Control de Manifiestos MODCAR. Bolivia.

2. ADUANA NACIONAL, B. 2004b. Manual de Control de Manifiestos MODCAR. [Accessed octubre 2010].

3. ADUANA NACIONAL, B. 2004c. Manual de Tránsitos MODTRS. [Accessed octubre 2010].

4. ADUANA NACIONAL, B. 2004d. Manual de Usuario SIDUNEA (Sistema Aduanero Automatizado). [Accessed octubre 2010].

5. CATALANI, E. 2010. Available: http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/ [Accessed diciembre 2010].

6. CÉSPEDES BASTEIRO, R. & RODRÍGUEZ PÉREZ, Y. 2009. *Diseño del Módulo Control de Personas del Sistema Único de Aduanas.* UCI.

7. HERNANDEZ RUIZ, G, Y. 2010. *Análisis del subsistema Control de Medios de Transporte Internacional del sistema Gestión Integral Aduanera.* UCI.

8. JACOBSON, I. & RUMBAUGH, J. 2000. El Proceso Unificado de Desarrollo de Software. *In:* WESLEY, A. (ed.).

9. LARMAN, C. 1999. UML y Patrones Introducción al análisis y diseño orientado a objetos. Primera ed. Mexico: Prentice Hall.

10. LORENZ, M. & KIDD, J. 1994. "Object-Oriented Software Metrics". Prentice Hall.

11. MACAS, A. M. & FIERRO, J. 2009. Diseño de software.

12. MARTÍNEZ MANSO, J. 2010. *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* UCI.

13. OLMEDILLA ARREGUI, J. 2005. *Revisión Sistemática de Métricas de Diseño* Universidad Politécnica de Madrid Facultad de Informática

14. PÉREZ MARTINTO, P. 2010. El diseño metodológico de la investigación científica.

15. PHP. 2001. Available: http://www.php.net/ [Accessed octubre 2010].

16. PHPBB. 2000. Available: http://extjses.com/ [Accessed octubre 2010].

17. PRESSMAN, R. 2005. Ingeniería de Software. Un enfoque práctico. Sexta edicion ed. EU.

18. RODRIGUEZ ARCE, J. 2008. La Importancia de las Aduanas en el Comercio Exterior.

19. STANDISH, G. 2008. Available: www.standishgroup.com [Accessed].

20. VISUALPARADIGMGROUP. 1995. *Boost Productivity with Innovative and Intuitive Technologies* [Online]. Available: http://www.visual-paradigm.com/ [Accessed diciembre 2010].

BIBLIOGRAFÍA

1. Rumbaugh, James. El Lenguaje Unificado de Modelado. Manual de Referencia.
2. Rodriguez, Jorge I. Arce. La Importancia de las Aduanas en el Comercio Exterior.
3. ADUANA GENERAL DE LA REPÚBLICA. RESOLUCIÓN No. 187-2008. 2008.
4. ADUANA GENERAL DE LA REPUBLICA. Metodología Interna para el Despacho Aduanero de los Medios de Transporte Internacional. 1994.
5. Rafael Andrés Céspedes Basteiro, Yisel Rodríguez Pérez. Diseño del Módulo Control de Personas del Sistema Único de Aduanas. Ciudad de La Habana : s.n., 2009.
6. Pressman, Roger S. Y,2005. Ingeniería del Software. Un enfoque práctico.
7. Martinto, MSc. Pedro Carlos Pérez. El diseño metodológico de la investigación científica.
8. Larman, Craig. Introducción al análisis y diseño orientado a objeto. UML y Patrones. México : s.n., 1999.
9. Bolivia, Aduana Nacional de. Módulo para el Agente Despachante MODBRK. Bolivia : s.n.
10. Manual de Usuario SIDUNEA (Sistema Aduanero Automatizado). Bolivia : s.n.
11. Manual de Tránsitos MODTRS. Bolivia : s.n.
12. Manual de Control de Manifiestos MODCAR. Bolivia : s.n.
13. Free Download Manager. Visual Paradigm for UML (ME). [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/. .
14. Entorno Virtual de Aprendizaje. Metodología de la Investigación Científica_Tema #3_Conferencia #3. [En línea] 2011. http://eva.uci.cu/course/view.php?id=63.
15. Entorno Virtual de Aprendizaje. Metodología de la Investigación Científica_Tema #3_Conferencia #4. [En línea] 2011. http://eva.uci.cu/course/view.php?id=63.
16. Entorno Virtual de Aprendizaje. Metodología de la Investigación Científica_Tema #3_Conferencia #2. [En línea] 2011. http://eva.uci.cu/course/view.php?id=63.
17. Entorno Virtual de Aprendizaje. Metodología de la Investigación Científica_Tema #3_Conferencia #1. [En línea] 2011. http://eva.uci.cu/course/view.php?id=63.
18. Entorno Virtual de Aprendizaje. Metodología de la Investigación Científica_Tema #3_Conferencia #6. [En línea] 2011. http://eva.uci.cu/course/view.php?id=63.
19. Entorno virtual de Aprendizaje. Metodología de la Investigación Científica_Tema #3_Conferencia #5. [En línea] 2011. http://eva.uci.cu/course/view.php?id=63.

20. MSDN. Microsoft Visual FoxPro 7.0. [En línea] Microsoft Corporation, 2010. http://msdn.microsoft.com/es-es/library/aa695547%28VS.71%29.aspx.
21. Hernández Ruiz, G, Y. 2010. Análisis del subsistema Control de Medios de Transporte Internacional del sistema Gestión Integral Aduanera. UCI.
22. Céspedes Basteiro, R. & Rodríguez Pérez, Y. 2009. Diseño del Módulo Control de Personas del Sistema Único de Aduanas. UCI.
23. ExP. Exportación y comercio exterior. Información para exportar, 2011. http://www.exportapymes.com/article4288-La-Aduana-busca-reemplazar-el-Sistema-Informatico-Maria.html.
24. Catálogo de software. Sistema de Gestión Financiera y Administrativa (SOFIA), 2011. http://www.catalogodesoftware.com/producto-sofia-570.

GLOSARIO DE TÉRMINOS

Aduana

Oficina pública, establecida generalmente en las costas y fronteras, para registrar, en el tráfico internacional, los géneros y mercaderías que se importa o exporta, y cobrar los derechos que adeudan.

AGR

Aduana General de la República de Cuba

CASE

Computer Aided Software Engineering, que 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, su objetivo es acelerar el proceso de automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.

Control aduanero

Conjunto de medidas tomadas con vistas a asegurar la observancia de las leyes y reglamentos que la Aduana está encargada de aplicar.

Despacho

Se refiere al despacho de entrada y salida de buques y aeronaves. Es la operación mediante la cual se tramita ante la Aduana, la documentación que declara la carga, provisiones, pasajeros y tripulantes que transporten los mismos, por el mando del buque o de la aeronave, o su consignatario o representante.

Diagrama

Representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas.

LCF

Lucha Contra el Fraude

MTI

Medio de Transporte Internacional. Vehículo utilizado para el transporte internacional de mercancías y personas. En Cuba por su situación geográfica se refiera a los medios utilizados en la vía aérea y marítima solamente.

Pasajero

Persona que viaja en cualquier medio de transporte sin ser tripulante del mismo.

PIA

Persona de Interés Aduanal.

Software

Se refiere a los programas y datos almacenados en un ordenador.

Tripulantes

Viajero que, a bordo del buque o aeronave, realiza actividades directamente vinculadas con la dirección, administración, mantenimiento y servicios de la misma.

Viajeros

Toda persona que entra o sale del territorio nacional a bordo de buque o aeronave de cualquier tipo considerándose viajero tanto a los pasajeros como a los tripulantes de dichos medios de transporte.

Visual FoxPro

Lenguaje de programación orientado a objetos, ofrece a los desarrolladores un conjunto de herramientas para crear aplicaciones de bases de datos para el escritorio, entornos cliente/servidor o para la Web.