Implementación de un Sistema de Despacho Aduanero Utilizando "MERODE" Como Método de Análisis

Ricardo Auhing Román[1], Roger Salinas Robalino[2], María V. Macías[3]

[1] Ingeniero en Computación especialización Sistemas de Información
[2] Ingeniero en Computación especialización Sistemas Tecnologicos
[3] Directora de Tesis. Ingeniera en Computación, Escuela Superior Politécnica del Litoral, 2000.

RESUMEN

Este proyecto surgió ante la necesidad de saber en cuánto puede afectar una metodología de análisis nueva, como lo es Model Driven Existence Dependency Relation Object Oriented (MERODE), en la calidad de un software resultante. Dado que MERODE ha tenido mucho éxito en terminos de flexibilidad en la industria del software europea, se decidió aplicarlo y ver su eficiencia para la industria local.

Los lenguajes de modelamiento tales como UML[1] ofrecen un grupo de modelos básicos para describir un sistema de software con diversas maneras de ver y en diversos niveles de la abstracción pero las herramientas que apoyan el uso de modelos de UML no pueden garantizar la consistencia entre modelos de vistas múltiples, debido a la carencia de una definición formal de la semántica de los diagramas de UML.Debido a esta debilidad nace MERODE que se basa en el concepto de un modelo simple para el cual se realizan diferentes vistas comprobando que son consistentes entre si.

Se desarrolló un Sistema de Despacho Aduanero realizando el análisis con MERODE donde se pudo comprobar los beneficios que ofrece esta técnica.

ABSTRACT

The idea of this project is to proof how much could a new methodology, such as Model Driven Existence Dependency Relation Object Oriented (MERODE), affect to the quality of resultant software. As MERODE has been successful, in flexibility terms, in the industry of European software it was decided to apply it and proof its efficiency for the local industry.

Modelling Languages, such as UML[1], offer a group of basic models to describe a software systems by different ways to see and in different levels of abstraction but the tools that support the use of UML models cannot guarantee the consistency between models of multiple views, due to the deficiency of a formal definition of the semantics of the UML diagrams. For this weakness, it was created MERODE that uses a simple model concept for which different views are made verifying that they are consistent to each other.

A Custom Delivery System was developed using MERODE analysis where it was possible verify the benefits that this technique offers.

[1] Rational Software Corporation, The Unified Modeling Language (UML),1997, http://www.rational.com/uml

1. INTRODUCCIÓN

Situación actual del uso de metodologías

Uno de los problemas fundamentales de la ingeniería de software es que cerca del 80% de los recursos disponibles deben ser usados para el mantenimiento del software existente. Parte de ese problema de mantenimiento es que los métodos de desarrollo de software han adoptado un manejo de procesos o un manejo de datos. Durante los últimos 10 años se han desarrollado un número considerable de métodos de diseño y análisis orientado a objetos. Se creía que aplicando la orientación a objetos a través del proceso de desarrollo de software, el re-uso de componentes sería mejorado, reduciendo de esta forma el costo de desarrollo de software. Sin embargo, los métodos actuales de análisis y diseños orientados a objetos sufren de algunas desventajas.

El primer problema es que la mayoría de métodos usan una mezcla de lenguaje natural y notaciones gráficas semi-formales para la especificación del propósito.
El segundo problema es que la orientación a objetos primero apareció en los lenguajes de programación. Los mismos conceptos han sido aplicados a las etapas de diseño y análisis, sin repensar estos conceptos fundamentalmente. Esto dio como resultado, que muchos conceptos y métodos de análisis tengan una tendencia al uso de la tecnología orientada a objetos para la implementación de sistemas. Esta es una contradicción con la esencia del análisis.

Situación actual de los Sistemas de Despacho Aduanero

Es de conocimiento general que los trámites aduaneros constan de muchos detalles que deben ser manejados por los diferentes agentes que participan en los procesos de importación y exportación de las mercaderías. Uno de estos agentes es el despachador aduanero, quien se encarga de hacer los trámites necesarios para “retirar” la carga importada o para “enviar” carga al exterior. Ya sea para importar o exportar mercadería, el despachador debe llevar un control del estado de los trámites y por supuesto garantizar al cliente la correcta finalización del proceso.

Existen sistemas que ayudan al despachador con el manejo de trámites aduaneros pero no satisfacen las necesidades de los clientes, por lo que se propone desarrollar un sistema generalizado y flexible para que pueda ser usado por diferentes usuarios, y además que éste sea consecuente con las disposiciones de la ley orgánica de aduanas ecuatoriana, satisfaciendo las necesidades de los clientes.

La complejidad principal del sistema de despacho aduanero es que este debe ser muy flexible, porque las leyes de la aduana cambian con una frecuencia muy alta. Como resultado de esto, el diseño del sistema debe ser construido para garantizar esta propiedad.

El modelo del sistema de despacho aduanero se lo realizó con un método nuevo de nombre MERODE (Model Driven Existence Dependency Relation Object Oriented Development) desarrollado en la Universidad Católica de Lovaina de Bélgica por el Professor Ph.D. Guido Dedene y Professor Ph.D. Monique Snoeck. Este método ha sido usado con mucho éxito en Bélgica y en Holanda; y ha demostrado con casos de estudio

que una de sus principales ventajas es brindar flexibilidad a los sistemas desarrollados; por lo que en teoría sería ideal para el desarrollo del sistema de despacho aduanero.

2. Comparación de MERODE con otras metodologías

Arquitectura de desarrollo de sistemas de capas

El desarrollo de modelo-conducido sigue la idea que la complejidad de los sistemas de información es conducida por la complejidad del mundo real. Dando atención especial al modelar el mundo real antes que la especificación de la funcionalidad deseada, los desarrolladores de los sistemas de información pueden manejar mejor la complejidad de los sistemas desarrollados. El acercamiento de modelo-conducido es también la respuesta a la carencia de la modularidad en la mayoría de los métodos de análisis y de diseño orientado a objetos. Aunque los conceptos básicos de la orientación a objetos mejoran la capacidad de mantenimiento de los sistemas significativamente, hay un sitio para una mejora importante organizando objetos en capas y particiones.

Algunas metodologías proporcionan técnicas tales como áreas de temas definidos arbitrariamente para reducir la complejidad. Sin embargo, es necesario mantener especificaciones (y la puesta en práctica) manejables y conservables en el reparto sistemático de los objetos del análisis. Para alcanzar un reparto sistemático, deben ser consideradas la naturaleza particular y las características de las especificaciones. Las especificaciones del primer tipo constituyen un modelo de la empresa (o el modelo del dominio del negocio) que contenga el conocimiento relevante del dominio para saber como funciona un negocio. La separación entre un modelo de la empresa y un modelo de la funcionalidad conduce a una partición natural de sistemas en tres subsistemas débilmente acoplados, cada uno que corresponde a un solo aspecto del problema, y desarrollándose independientemente del resto de aspectos. Los tres aspectos independientes son:

- La lógica del negocio (el modelo de la empresa);
- La organización del trabajo (los servicios de entrada);
- las necesidades de información (los servicios de salida).

Cuando los usuarios formulan requisitos del sistema, tienden a mezclar especificaciones del conocimiento del dominio con requisitos de la funcionalidad. Para la metodología JSD[2], modelar el mundo real o el "dominio" permite construir una base sólida para un sistema de software. La complejidad de la mayoría de los sistemas es causada por la complejidad de la realidad con la que ellos tienen que tratar. Tiene así sentido construir un modelo del mundo real antes del desarrollo de un sistema de información. Por otra parte, no tiene ningún sentido construir un modelo del mundo real sin tener cierta funcionalidad en mente. Aquí, ambos modelos son una parte integral de un modelo del análisis con una misma importancia.

[2] J.A. Zachman, 1999, A framework for information systems architecture

La arquitectura de MERODE conduce a los aumentos significativos de la productividad durante el mantenimiento del sistema.
Como las funciones son módulos independientes, pueden ser "enchufadas" y "desenchufadas" fácilmente y las clases de objetos de la empresa se pueden agregar al modelo de la empresa sin efectos sobre funciones existentes y sobre otras clases de objetos de la empresa. Además, es fácil determinar el grupo de funciones que es afectado por la modificacion de los objetos de la empresa dado que las funciones se basan siempre en una parte específica del modelo de la misma.

Comparación con el Método de Jacobson

A primera vista, MERODE se vio parecido al acercamiento manejado a casos de uso[3] de Ivar Jacobson. En el método de Jacobson, los requisitos funcionales son la base para el desarrollo del modelo de los requisitos. Los requisitos funcionales son especificados por los casos de uso.

El modelo de Casos de uso puede ser expresado en términos de un modelo de objetos del dominio, que es una clase de modelo de la empresa. En cambio, el modelo de análisis MERODE utiliza tres clases de objetos: Los objetos entidad, los objetos de la interface y objetos de control. En general, los objetos del dominio serán transformados en objetos de la entidad y la funcionalidad de un caso del uso será repartida y asignada a los objetos de los tres tipos.

Al comparar el método de Jacobson con MERODE, uno podría decir que los tipos de objetos del dominio de MERODE son tipos del objeto de la entidad de Jacobson y que las funciones de MERODE son equivalentes a los casos de uso.

Sin embargo, MERODE se enfoca más que el método de Jacobson al dominio que modela. Además, se requiere que las funciones sean independientes una de otra: La ejecución de una función nunca debería de depender de la ejecución o disponibilidad de otras funciones. Toda comunicación debe pasar a través del modelo de la empresa. Esto no es requerido para los casos de uso. Otra diferencia importante es que en un modelo de la empresa de MERODE se incluye la definición de eventos, atributos y operación de objeto; el acercamiento conducido a los casos de uso pospone la definición de operaciones hasta la fase de diseño del sistema.

Interacción de Objetos en MERODE

El paradigma orientado a objetos proviene de lenguajes de programación. La orientación a objetos también se ha aplicado a las etapas de diseño y de análisis de los ciclos de vida del desarrollo de software.

En estas etapas, la mayoría de los métodos utilizan conceptos básicos de la orientación a objetos sin la debida adaptación. Casi todos los métodos orientados a objetos actuales de análisis y de diseño utilizan el mensaje que pasa como paradigma de la comunicación. Con este formalismo, los objetos pueden invocar directamente operaciones de otros objetos, esto seguramente es un mecanismo de comunicación eficiente y permite que los modeladores controlen fuertemente el ordenamiento y la sincronización.

[3] Use Cases: the Pros and cons, 2004, The Dangers of Misusing Use Cases, http://www.ksc.com/article7.htm

Sin embargo, usar el paso de mensajes para los propósitos de la especificación implica casi el uso del paso de mensaje para el diseño y la implementación. Además, debido al uso de diversos paradigmas para la especificación y la implementación, la trazabilidad de requisitos no puede ser asegurada. De hecho, la situación actual viola el acuerdo general que un modelo conceptual (en este caso un modelo de análisis orientado a objetos) debería ser un modelo del dominio del problema y no ser influenciado solamente por el dominio de la solución. Igualmente, las técnicas del análisis no se deben influenciar por opciones posibles de la puesta en práctica. Este tipo de interacción de objetos se puede modelar por medio del álgebra de proceso. Las ventajas de usar la comunicación por medio de eventos comunes son numerosas:

-En el contexto del modelamiento del dominio o de la empresa, se conduce a la identificación explícita de los eventos del negocio; esto puede ser representado por una técnica de modelamiento sencilla:

-La tabla de objetos-eventos;

-La noción de los eventos del negocio se integra con la noción de la tarea;

-La noción del evento combina bien con las máquinas de estado finito, donde los eventos son los disparadores para las transiciones de un estado a otro;

-El comportamiento y la comunicación entre objetos son fáciles de concluir;

-La noción de los eventos del negocio con las nociones del evento del usuario y los servicios repetidos usados en bibliotecas del GUI-objeto;

-Permite una puesta en práctica conducido a eventos, que es fácil de realizar con herramientas de implementación no orientadas a objetos;

-No imposibilita una puesta en práctica orientada a objetos con el paso de mensajes como mecanismo de la interacción.

3. Concepto de MERODE

Aunque los conceptos básicos de la orientación a objetos (como encapsulación, polimorfismo, herencia) incrementan significativamente el mantenimiento del sistema, el efecto de modificaciones dentro del diseño no debe ser desestimado. Desde que el número de objetos en un sistema aumenta, la maniobralidad se dificulta rápidamente. Esto también atenta con un mantenimiento fácil. Como resultado, un sistema orientado a objetos que sea grande es muy difícil de mantenerlo. El principal objetivo en el desarrollo de MERODE[4,5,6,7,8] es dar una solución para los problemas anteriormente mencionados: La falta de facilidades para garantizar la calidad de especificaciones, la implementación con tendencia a muchas técnicas de análisis orientado a objetos, y la falta de un mayor nivel de modularidad en el desarrollo orientado a objetos. Esto ha resultado en tres conceptos que son las tres características sobresalientes del MERODE:

---

[4] Management Information Systems Research Group, 2000, Manual del Uso de MERMAID. Snoeck Monique, 1995, On a process Algebra Approach to the construction and analysis of MERODE based conceptual models

[5] Monique Snoeck, Guido Dedene, Object – Oriented Enterprise Modelling with MERODE

[6] Management Information Systems Research Group, http://www.econ.kuleuven.ac.be/tew/academic/infosys/research/merode.htm

[7] Richard Paige, Jonathan Ostroff, 2002, The single Model Principle, http://www.jot.fm

[8] Monique Snoeck, Cindy Michiels, Guido Dedene, Consistency by Construction: the case of MERODE.

-Un desarrollo orientado al modelo.
-Un concepto de interacción de objetos más abstracto
-Una definición formal de todas las técnicas que nos aseguren un nivel más alto en el control de calidad.

Las fases más importantes del proceso de desarrollo de MERODE son: La fase de especificación y la fase de implementación. La especificación del sistema está dividida de acuerdo a la estructura de los requerimientos del usuario.

-Fase del modelamiento de la empresa.
-Modelamiento de Tipo de objeto de la empresa y tipo de eventos del negocio.
-Identificación del tipo de objeto de la empresa.
-Identificación del tipo de eventos del negocio.
-Construcción del gráfico de dependencia de existencia.
-Construcción de la tabla objeto-evento.
-Especificación de los constraints.
-Especificación de los vectores del estado y métodos objeto-evento.
-Definición de los vectores de estado.
-Definición de los métodos objetos-eventos.
-Definición de constraints de datos.

El proceso entero es dirigido principalmente por un sistema de reglas rigurosas de la verificación que determinan la calidad de las especificaciones.

Tipos de objetos de la empresa y tipos de eventos del negocio

Debido al concepto de la comunicación por medio de eventos comunes, un modelo de la empresa no está solo integrado por objetos de la empresa, sino también de los eventos del negocio.
El primer paso en el modelamiento de la empresa es la identificación de los tipos de objetos de la empresa y los tipos de eventos del negocio que ocurren en la realidad descrita y que, al mismo tiempo, son relevantes para que el sistema sea construido.

Para ser incluido en la lista de elementos relevantes, los tipos de objetos tanto como los tipos de eventos deberán satisfacer algunos requisitos importantes.

Para ser un candidato válido de tipo de objeto, un elemento debe satisfacer las características básicas: Un objeto tiene una identidad, tiene un estado descrito por medio de atributos y demuestra un cierto comportamiento. En el contexto de modelamiento de la empresa, el aspecto del comportamiento se puede formular en términos de la participación del evento. Además, los objetos de la empresa tienen que corresponder a los conceptos del mundo real, que conduce a los criterios adicionales. Consecuentemente, los tipos del objeto de la empresa deberían satisfacer el siguiente grupo de requisitos:

-Un objeto se puede describir por un número de características. Por ejemplo, un cliente puede ser descrito por medio de un nombre, por la fecha de nacimiento, una dirección. Una fecha no es un objeto sino un elemento que describe por ejemplo, fecha de nacimiento describe al cliente.

-Un objeto tiene una identidad, no necesariamente explícita y visible al usuario. (En otras palabras, el concepto de identidad no es el mismo que el concepto de clave (primaria) en el modelo relacional).

-Un objeto corresponde a un concepto del mundo real: Tiene contrapartes del mundo real que pueden o no ser físicas. Por ejemplo, un "cliente" es un objeto en el mundo real.

-Un objeto existe por cierto período de tiempo. Por ejemplo, un préstamo tiene cierta duración: Comienza cuando se pide prestado el libro y termina cuando se devuelve el libro.

-Un objeto está siempre implicado en por lo menos dos eventos: Su creación y su conclusión. El CLIENTE es un tipo de objeto; la "fecha" y la "edad" no son, porque no hay tipos de eventos tales como create_date, end_of_date, create_age o end_of_age.

Para el negocio, los requisitos para los tipos de eventos son:

-Un evento del negocio de cierto tipo ocurre o es reconocido en un punto en el tiempo; su duración posible puede ser abstraído. Un buen ejemplo es el acontecimiento "paga". Cuando incluimos "paga" en la lista de los tipos relevantes del acontecimiento, refiere al momento en el cual la actividad de pagar termina. Por ejemplo, consideremos un pago en efectivo. Tal actividad durará cierto rato: El cliente tiene que tomar su cartera, dinero de la cuenta, pero la duración real de todo esto no tiene ningún interés para el modelo.

-Un acontecimiento del negocio empareja algo que sucede en el mundo real dentro del universo del discurso; no es un acontecimiento del sistema de información.

-Otra consideración es que solamente los elementos que son necesarios para entregar la funcionalidad que tenemos en mente deben ser conservados. El modelo de la empresa determina totalmente qué funcionalidad puede ser entregada. En este sentido es importante incluir bastantes tipos del objeto y tipos del acontecimiento. Pero la retención de muchos elementos aumentará el costo del desarrollo y por lo tanto no es deseable.

En el modelo de la empresa, los tipos del objeto de la empresa y los tipos del acontecimiento del negocio se relacionan el uno al otro y se definen más detalladamente. El modelo de la empresa de MERODE consiste en tres sub-modelos:

Un esquema del gráfico de la dependencia de la existencia (EDG) o de la relación del objeto, una tabla del objeto-evento (OET) y un modelo del comportamiento.

En MERODE se requiere que todas las relaciones expresen dependencia de la existencia. Un objeto (llamado ' objeto subordinado o esclavo ') es dependiente de la existencia en otro objeto (llamado el ' objeto principal o maestro') si la vida del primer objeto siempre se contiene completamente en la vida del último.

El ciclo vital de los objetos se puede representar por medio de los autómatas finitos. En un autómata finito, los estados se representan como círculos y las transiciones como flechas etiquetadas con los nombres de los tipos de eventos. El estado inicial es indicado por un círculo que lleva una flecha entrante y el estado final como círculo con la línea doble.

Como la tabla del objeto-evento también contiene la indicación del tipo de implicación (cree, modifíquese o finalice), los JSD-diagramas de dibujo3 o los autómatas finitos serán solamente necesarios para la especificación de los constraints adicionales que todavía no son especificados por la tabla objeto-evento.

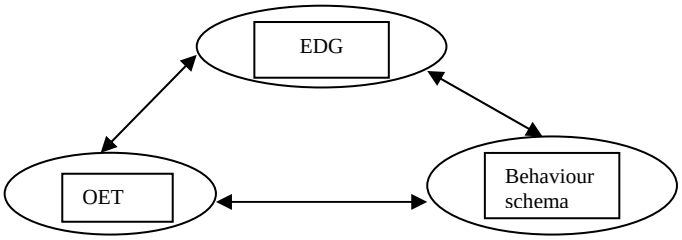

Figura 2.1 Comprobación de consistencia entre los 3 esquemas de modelamiento de la empresa

Usando el gráfico de dependencia de la existencia como la clave para el control de la consistencia, es posible lograr una mejor calidad en esquemas de empresa que cuando solamente se controla la concordancia sintáctica entre los sub-esquemas.(Figura 2.1)

La semántica del EDG, de OET y del FSM y la consistencia de la visión se han definido por medio de álgebra de proceso. Existe tambien un sistema de reglas de comprobación de consistencia que prevé un cierto chequeo básico de la integridad. La Fig.2.2 da una descripción de las opiniones y de las reglas.

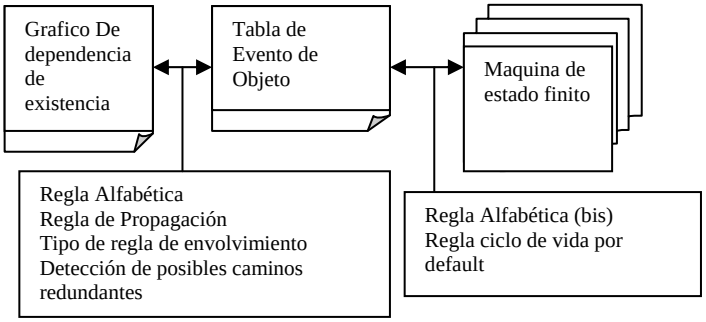

Figura 2.2 Vista y consistencia que comprueba reglas en MERODE

Requerimientos funcionales del Sistema

Se deberá desarrollar un sistema del tipo aplicación ejecutable que sea capaz de solicitar a un Agente de Despacho, conocedor del área de declaración aduanera, la información pertinente a cualquier trámite de importación o exportación de mercadería, sujeto a cualquiera de los regímenes ya establecidos por la Corporación Aduanera Ecuatoriana (C.A.E.), acogiéndose a los estándares impuestos por la Institución mencionada. Posteriormente, esa información deberá ser guardada adecuadamente en una base de datos de manera que en cualquier momento pueda ser recuperada enteramente para consultas, y bajo ciertas condiciones, editada o cambiada. Finalmente, en base a la información solicitada previamente, el sistema debe estar en capacidad de generar los documentos concernientes al trámite en cuestión a fin de completar la declaración aduanera. Para ello, la información solicitada debe ser concisa y no redundante, para que no necesite ninguna información adicional para generar los mencionados documentos. Esto no siempre se da, ya que por lo menos en documentos como

el DUI o el DAU, se necesita información de sus respectivos refrendos, que ineludiblemente se ingresa luego de que son generados.

Adicionalmente, el sistema debe permitir la actualización de información compartida por todos los usuarios. Tal es el caso de la información de los clientes, por ejemplo. Se debe brindar la posibilidad de añadir y modificar los datos de un cliente y esta nueva información estar disponible para que los digitadores la usen en sus trámites.

A continuación se presenta el modelo de análisis del Sistema de Despacho Aduanero hecho con el método de Análisis MERODE.

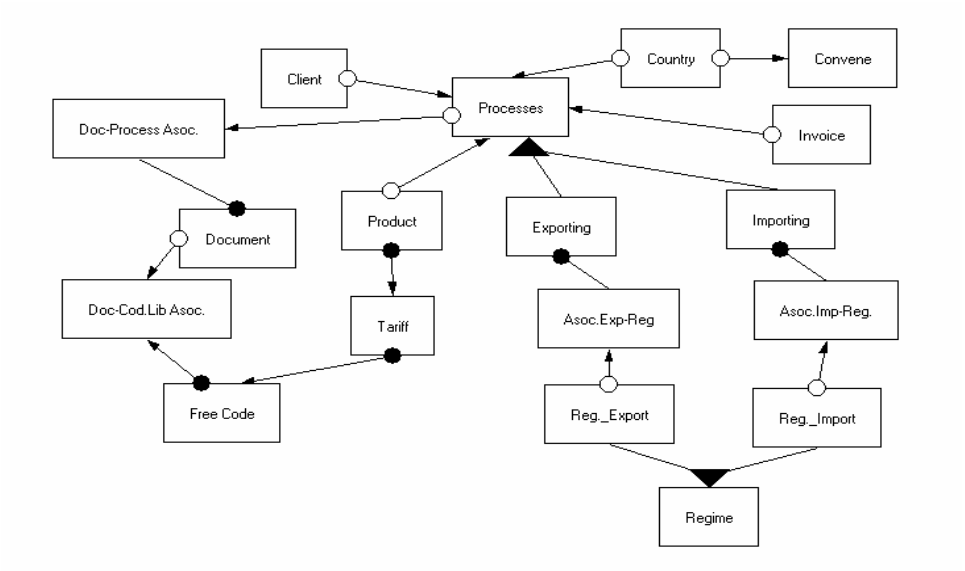

Figura 2.3 Modelo de Objetos del Sistema

4. CONCLUSIONES

Como resultado del desarrollo de esta tesis, tenemos un sistema que se fundamenta básicamente en el dominio de la empresa (despachador aduanero); esto facilita en el mantenimiento y actualización del sistema. Además, tiene una interacción mas amigable que otros sistemas de despacho aduanero por lo que los trámites de importación y exportación se agilitan y se hace más eficiente el proceso que realizan los despachadores aduaneros.

Por medio de este sistema, los despachadores aduaneros podrán realizar el llenado de datos en los formularios de aduana respectivos de una manera más rápida y organizada.

Este sistema puede implementarse en empresas despachadoras de cualquier tamaño, y soportar futuros crecimientos, lo que lo hace escalable.

El modelo de MERODE (EDG) sirvió como punto de arranque para el modelo de la Base de Datos lo cual facilitó mucho su creación. Además, MERODE ayudó a identificar de una manera clara los objetos de la empresa.

Los FSM que se realizaron sirvieron para ver de una forma mas clara cada proceso del sistema.

Como recomendación después del desarrollo de esta tesis es que en la implementación de un sistema se debe usar una herramienta que sea multiplataforma como JAVA para que el sistema pueda ser usado en cualquier lado sin tener ninguna restricción por Sistemas Operativos.

5. BIBLIOGRAFÍA

1. Rational Software Corporation, The Unified Modeling Language (UML),1997, http://www.rational.com/uml
2. J.A. Zachman, 1999, A framework for information systems architecture
3. Use Cases: the Pros and cons, 2004, The Dangers of Misusing Use Cases, http://www.ksc.com/article7.htm
4. Management Information Systems Research Group, 2000, Manual del Uso de MERMAID.
5. Snoeck Monique, 1995, On a process Algebra Approach to the construction and analysis of MERODE based conceptual models
6. Monique Snoeck, Guido Dedene, Object – Oriented Enterprise Modelling with MERODE
7.Management Information Systems Research Group, http://www.econ.kuleuven.ac.be/tew/academic/infosys/research/merode.htm
8. Richard Paige, Jonathan Ostroff, 2002, The single Model Principle, http://www.jot.fm
9. Monique Snoeck, Cindy Michiels, Guido Dedene, Consistency by Construction: the case of MERODE.
CAE, 1998, Ley Organica de Aduanas L.99
CAE, 2000, Manual de Procedimientos Aduaneros
CAE, 2000, Reglamento General a la Ley Orgánica de Aduanas
CAE, 2001, Procedimiento de Control de Operaciones Aduaneras Marítimas
CAE, 2000, Procedimientos para Importar, http://www.aduana.gov.ec
CAE, 2000, Procedimientos para Exportar, http://www.aduana.gov.ec
CAE, 2000, Procedimientos para Importación de Regímenes Especiales, http://www.aduana.gov.ec
CAE, 2000, Proceso Operativo Aduanero, http://www.aduana.gov.ec