Universidad de las Ciencias Informáticas

FACULTAD 3

Título: Módulo Control de vehículo y Seguridad activa y pasiva del Sistema Control de Flota y Mantenimiento para la Dirección de Transporte de la Universidad de las Ciencias Informáticas.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores: Rayner Guerra Aizpurua
Ernesto González Perdomo.

Tutor: Ing. Dasiel Otero Dartayet.

Junio de 2013

PENSAMIENTO

“Si no se tiene avidez por el conocimiento, no se conocerá el éxito”.

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al Departamento de Soluciones Empresariales del Centro de Informatización para la Gestión de Entidades de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores:

_____ _____

Rayner Guerra Aizpurua Ernesto González Perdomo

Tutor:

_____

Ing. Dasiel Otero Dartayet

DATOS DE CONTACTO

Ing. Dasiel Otero Dartayet: Ingeniero en Ciencias Informáticas. Graduado en la Universidad de las Ciencias Informáticas en el 2008. Facultad 3.

Correo electrónico: dotero@uci.cu

AGRADECIMIENTOS

Ambos: A los profesores y compañeros que nos apoyaron en este desafío, que ayudaron e hicieron todo lo posible para que llegáramos aquí.

RAYNER:

Termina una etapa de nuestras vidas, pero comienza otra. Una nueva que no existiría sin la anterior, una etapa que no sería posible sin tantos años de sacrificios, sin tantos años de estudios, sin el apoyo y la confianza de muchas personas, que, algunas de ellas no se encuentran aquí en estos momentos, pero su mención en este documento se hace totalmente necesaria.

A mi madre por su incansable apoyo y su total confianza hacia mi persona, confiando siempre en que yo podía y tenía que lograr este importante objetivo.

A mi padre por ser mi motor impulsor, mi sostén, mi mentor. Viejo como siempre dices "Yo me gradúo contigo".

A mi hermana por preocuparse por mis estudios, y por alegrarme los días más grises con ese carácter jovial suyo.

A mi abuelita, que por su delicado estado de salud no le es posible estar conmigo aquí hoy, gracias. Que a pesar de los miles de años que tienes aun no olvidas mi nombre.

A mis vecinos que siempre están pendientes de mi desempeño, y que tantas veces me han ayudado a lo largo de estos 5 años.

A mis compañeros de aula por amenizar mi trayectoria estudiantil, a Lianet, Evelyn, Kirenia, Tony, Lorenzo, a mis eternos compañeros de cuarto Javier Padrón, Rafa, Dyango, Reinier.

A mi compañero de tesis Ernesto….Pollote creo que nos graduamos.

A mis compañeros del politécnico, que se encuentran estudiando conmigo aquí, Popi, Javier, Jova, Abelito, gracias por ser buenos amigos.

También quisiera hacer especiales estos agradecimientos a personas especiales, que de no ser por ellos, estoy seguro que hoy no estaría aquí discutiendo mi tesis de grado.

Agradecimientos

A mi novia Cary, por ser esa persona que me dio ánimos cuando ya no tenía de donde sacarlos, por su constante apoyo, por su total confianza hacia mí en todo momento, tranquilizándome con ese "Todo va a salir bien" palabras que acaricio cada vez que algo no me sale como quisiera.
A mi tutor Dasiel….no tengo palabras para describir mi agradecimiento hacia su persona, ya que fue mi mentor, mi padre, ese que nos sacaba del hueco cuando ya pensaba que de ahí no iba a salir, gracias.
A Roly, ya no me queda aliento para decirte gracias, que sin obtener nada a cambio te partías el lomo con nosotros programando hasta que las cosas no salían totalmente bien, gracias….mi hermano vas a ser tremendo ingeniero.
A todos los profesores del proyecto que se sentaban a nuestro lado, se preocupaban con nosotros, y que corrieron con nosotros cuando no debían hacerlo: Erich, Maylín, Olga, Martha, Virtudes, muchas gracias.
A nuestro tribunal por ser compasivos, profesionales en todo momento, por sus sugerencias y enseñanzas, gracias.
Muchas gracias a todos, el día que vea mi título colgado en la pared, sin dudas los recordaré a todos, muchas gracias.

ERNESTO:
A mi padre que está conmigo siempre y es el ejemplo de persona que quiero seguir. Su amor y comprensión sin límites me llevan hacia adelante.
A mi madre por perseguirme y hacer tanto y tanto por mí. Por quererme, adorarme, comprenderme y ser tan parcial…
A Rey por guiarme en mi vida, por ser el mejor hermano, por ser mi primer profesor, por hacerme ver que el conocimiento real es la única forma de comprenderlo todo.
A ellos tres por igual, por darme la vida, cuidarme y quererme tanto.
A mis abuelos por ser ejemplos de sacrificio constante y por preocuparse tanto de mí y de toda la familia.
A mis tíos y mis primos por prestarme tanta atención y darme tanto cariño.

Agradecimientos

A Juan e Hirina por querernos tanto a mí y a Rey. Su pérdida es tan de ellos como nuestra. Quien encuentra un amigo, encuentra un tesoro. Estos años en la UCI me han servido para conocer demasiados tipos de personas e interactuar con todos. En el docente o en la residencia he compartido 7 años con personas a las que quiero mucho. Ellos han estado cerca desde el inicio y hemos disfrutado buenos momentos, así como nos hemos acompañado en algunos malos. Hoy están conmigo o están lejos, pero siempre los llevo cerca. Pasarán muchos años y cualquiera podrá preguntarme por alguno, que me verá el rostro alegre por saber dónde está.

A Leo y a Janio por sus distintas maneras de ver la vida y llevarme a algún punto donde se unen. Por compartir tanta diversión, problema, viaje, caminata, trabajo, fiestas… por todo.

A Trolly y Mey porque siendo tan feis me han demostrado que solo la belleza interior es la que vale, porque me han permitido ver con el corazón y he tenido los mejores ratos, he llorado, he reído, he molestado, he agradado y soy feliz, muy feliz de tenerles junticos haciendo bello mi mundo.

A Charles y Ernesto por demostrarme que en los momentos definitorios no se permiten los errores y los consejos que me dieron para afrontarlos. Por dar tanto chucho y hacerme más fuerte.

A mi grupo actual, que supo acogerme como otro miembro más y ponerme otros apodos felices. A Javier, Astro, Rafa, Idel, Luisy, Felipe y demás.

A Carlos, Damián, Gustavo, Alexander, Yasmany, Gilbe y demás miembros de los Rangers por los primeros años.

A mis maestros y profesores de tantos años.

A todos, muchas gracias.

RESUMEN

En el Departamento de Soluciones Empresariales perteneciente al Centro de Informatización de la Gestión Entidades de la Universidad de las Ciencias Informáticas se está desarrollando el Sistema de Control de Flota y Mantenimiento para la gestión de los procesos que se realizan para la Dirección de Transporte de la Universidad de las Ciencias Informáticas. En el presente trabajo se realiza el análisis, diseño e implementación de los procesos Control de vehículo y Seguridad activa y pasiva con el objetivo de contribuir a mejorar su gestión y lograr su integración con el resto de los procesos de la Dirección de Transporte de la Universidad de las Ciencias Informáticas. Además, actualmente la información relacionada al control de los vehículos y a la seguridad activa y pasiva es gestionada de forma manual y no se tiene un control adecuado de la información sobre el control y asignación de los vehículos, pues existe información almacenada en archivadores y en hojas de cálculo de Microsoft Excel, lo que hace engorrosa la gestión y el control de estos procesos por parte de la Dirección de Transporte, además de que no exista integración entre los procesos que se ejecutan en esta área.

Con la implementación de los procesos de Control de vehículo y Seguridad activa y pasiva, se contribuyó al ahorro de cuantiosos recursos materiales por parte de los clientes, facilitándoles su trabajo con la realización de una serie de reportes que le posibilitan controlar la información referente a los datos de los vehículos. Se integraron los procesos Control de vehículo, Asignación de vehículos, Planificación del día de la técnica, Control de accidentes y Seguridad activa y pasiva, lo cual permitió tener un control sobre el parque vehicular, control sobre los accidentes y sobre la cantidad de horas empleadas en la ejecución de una actividad de mantenimiento determinada.

PALABRAS CLAVE
“Gestión, Mantenimiento, Vehículo”.

ÍNDICE DE CONTENIDOS

Índice de contenidos

INTRODUCCIÓN

El desarrollo de las Tecnologías de las Informáticas y las Comunicaciones en la actualidad, brinda soluciones de informatización para organizaciones en las que previamente se trabajaba de forma manual y orientándose a la toma de decisiones con el objetivo controlar adecuadamente dichas tareas y procesos.
Entre estas soluciones, se encuentran los Sistemas de Control de Flotas y Mantenimiento de Vehículos, que incluyen una variedad de funciones como mantenimiento, gestión de choferes, gestión de combustible, gestión de piezas de repuesto y gestión de la seguridad personal y vehicular (Bennet, 2010). Estos sistemas tienen procesos delimitados y dedicados al Control Vehicular y a la Seguridad activa y pasiva y registran los detalles de cada vehículo de la flota, su disponibilidad y su asignación a las distintas áreas. Además, aumentan la productividad, contribuyen con el mejoramiento de la gestión de asignación de combustibles o piezas de repuesto, y minimizan los riesgos asociados a la accidentabilidad mediante la prevención, el control y el chequeo del parque automotor. Estos procesos están involucrados en los pasos para delimitar responsabilidades cuando ocurren accidentes, cumpliendo con las normativas legales vigentes (Bennet, 2010).
En el departamento de Soluciones Empresariales del Centro de Informatización de la Gestión Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) se desarrolla el proyecto de Sistema Control de Flota y Mantenimiento para la Base de Transporte de la UCI con procesos como Control de vehículo, en el que se controla la información referente a todos los vehículos, sus mantenimientos tanto preventivos como correctivos, el control de neumáticos, el control de las baterías y el registro del consumo de combustible. El proceso Asignación de vehículos delimita los responsables por áreas; el Control de Accidentes se encarga de manejar este tipo de situaciones, plasmando los problemas identificados en el modelo de abolladuras; en Seguridad Activa y Pasiva se gestiona la organización de la seguridad vehicular y en Planificación del día de la técnica se trabaja con la información referente a la organización de ese evento. Para mejorar la gestión de los procesos anteriores, es necesario poseer una información centralizada e integrada con los demás procesos, con el objetivo de brindar reportes detallados y el manejo adecuado de la información.
La realización del trabajo de forma manual y, a veces, mediante documentación en documentos de Microsoft Excel, está relacionada con errores inherentes al factor humano, trayendo como resultado la obtención imprecisa de la información requerida, datos duplicados entre los documentos Excel y los

papeles, que la información sobre los procesos no esté disponible en tiempo real y que sea engorroso el trabajo de búsqueda entre todos los documentos previamente almacenados en archivadores. Además, la planificación del día de la técnica se hace difícil, porque no se tiene información actualizada de los expedientes de los vehículos y se demora la asignación de estos cuando hay cambios en sus responsables. Los controles al estado de los vehículos se dificultan al tener que revisar gran cantidad de expedientes, que al encontrarse en archivadores y a merced del deterioro por la manipulación y el paso del tiempo, hacen que la obtención de reportes de cualquier índole sea deficitaria. Lo anterior demuestra que no existe una integración real entre estos procesos, haciéndolos lentos en su funcionamiento.

Esta problemática planteada permite definir como **problema a resolver**: ¿Cómo contribuir a mejorar la gestión de los procesos Control de vehículo y Seguridad activa y pasiva del Sistema Control de Flota y Mantenimiento para la Dirección de Transporte de la UCI?

Para la solución del problema se tiene como **objeto de estudio**: Sistemas de control de flota de vehículos.

Su **campo de acción** está dado por: El proceso de Control de vehículo y Seguridad activa y pasiva de la Dirección de Transporte de la UCI.

Se plantea como **objetivo general**: Realizar el Análisis, Diseño e Implementación del módulo Control de vehículo y Seguridad activa y pasiva del Sistema Control de Flota y Mantenimiento para la Dirección de Transporte de la UCI.

Para darle cumplimiento al objetivo general están los **objetivos específicos** siguientes:

- Fundamentar la investigación a través de la creación del Marco Teórico para definir las herramientas, tecnologías y lenguajes a utilizar, así como a los sistemas similares existentes.
- Elaborar el análisis del módulo Control de vehículo y Seguridad activa y pasiva para la Dirección de Transporte de la UCI para identificar los requisitos funcionales y no funcionales del sistema.
- Diseñar el módulo Control de vehículo y Seguridad activa y pasiva para realizar y describir los diagramas de clase del diseño y componentes.
- Realizar la implementación del módulo Control de vehículo y Seguridad activa y pasiva para lograr la versión estable del sistema.
- Efectuar pruebas a la solución para demostrar la validez de la propuesta mediante las métricas escogidas.

Teniéndose como **idea a defender:** La realización del análisis, diseño e implementación del módulo Control de vehículo y Seguridad activa y pasiva para la Dirección de Transporte de la UCI permitirá una adecuada ejecución de los procesos en esa área.

Los siguientes métodos de investigación fueron empleados para la realización de este trabajo para darle cumplimiento a los objetivos específicos:

Métodos Teóricos:

- Histórico-Lógicos: Para realizar el análisis, diseño e implementación del sistema; expresar teóricamente sus elementos fundamentales y para realizar el estado del arte de los software existentes.
- Análisis-Síntesis: Para comprender, resumir y describir los procesos del Sistema de Control de Flotas y Mantenimiento Vehicular.

Métodos empíricos:

- Observación: Para comprender el proceso de Control de vehículo y Seguridad activa y pasiva, estudiando cómo se realizaba previamente este proceso en el área de Transporte de la UCI.
- Entrevista: Para captar las experiencias de los trabajadores que se encargan manualmente de ejecutar el proceso de Control de vehículo y Seguridad activa y pasiva del Sistema de Control de Flotas y Mantenimiento Vehicular.

El presente trabajo consta de tres capítulos y contiene varios anexos con los artefactos generados durante el análisis, diseño, implementación y validación del sistema. A continuación se describe el objetivo principal de cada uno de los capítulos.

Capítulo 1: Fundamentación Teórica.

En este capítulo se realiza el estudio de las soluciones informáticas existentes sobre Sistemas de Control de Flotas, haciendo énfasis en el estudio sobre cómo estos manejan el proceso de Control de vehículo y

Seguridad activa y pasiva. Además, se realiza el estudio de las metodologías de desarrollo, herramientas CASE[1] y lenguajes de modelado para determinar su utilización en el presente trabajo de diploma.

Capítulo 2: Diseño e Implementación.

En este capítulo se presentan los Diagramas de clases del diseño, descripción de las clases del diseño, diagramas de componentes y el modelo de datos, además se realiza un análisis de los patrones de diseño utilizados. Por otra parte, se muestran los estándares de codificación, se realiza una descripción de la implementación por funcionalidades y se muestra la publicación de los servicios entre componentes.

Capítulo 3: Validación y Pruebas.

En este capítulo se exponen las métricas y pruebas utilizadas para la validación del diseño propuesto realizado, así como los resultados de la aplicación de pruebas de caja negra y caja blanca realizadas al sistema.

[1] Acrónimo en inglés de Ingeniería de Software Asistida por Computadoras,

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo comienza mostrando cómo se realiza el proceso de Control de vehículo y Seguridad activa y pasiva para la Dirección de Transporte de la UCI y la importancia que posee este sistema para la institución. Luego, se realiza un análisis de la gestión de control de vehículos en los diferentes sistemas de control de flotas existentes tanto en el extranjero como en Cuba. El capítulo culmina argumentando la selección de cada una de las herramientas, metodologías, lenguajes de desarrollo, *frameworks*, servidores de base de datos y de aplicación web utilizados para la implementación del proceso Control de vehículo y Seguridad activa y pasiva.

1.2 Procesos Control de vehículo y Seguridad activa y pasiva de la Dirección de Transporte de la UCI

En la Dirección de Transporte de la UCI el objetivo de la gestión del proceso Control de vehículo es registrar la información de todos los vehículos de la flota, su disponibilidad y su asignación a las distintas áreas, además de los mantenimientos tanto preventivos como correctivos. En el caso del proceso de Asignación de vehículos, este determina los responsables por áreas. El proceso de Control de Accidentes maneja este tipo de situaciones, además de que minimiza los riesgos asociados a la accidentabilidad mediante la prevención, el control y la inspección del parque automotor; el proceso de Seguridad Activa y Pasiva, gestiona la organización de la seguridad vehicular, y en el proceso de Planificación del Día de la Técnica se trabaja con la información referente a la organización del mismo.

1.3 Gestión del control de vehículos en otros sistemas de control de flotas

1.3.1 Sistemas internacionales

1.3.1.1 SoftFlot

Ofrece, dentro de su portafolio de productos de Aplicaciones Empresariales, una herramienta para la administración de su parque vehicular denominado Autocontrol **(Comsun Car System, 2009).**

Autocontrol es un sistema que controla los vehículos de la organización, automatizando las principales funciones de esta actividad. Cuenta con los módulos de solicitud, asignación, registro de las condiciones

físicas de entrada y salida del vehículo, seguimiento y trámites administrativos, hasta el servicio de mantenimiento que cada vehículo requiere incluyendo la generación de órdenes de servicio.
Este sistema está difundido por países de América Latina como Chile, México y Venezuela. Entre las funcionalidades que contiene este software se encuentran:

- Control de vehículo.
- Control de recursos humanos.
- Asignación de choferes.
- Almacén de recursos materiales.
- Costos y presupuestos.
- Control de combustibles
- Control de accidentes.
- Asignación, registro y gestión del calendario de tareas de mantenimiento a los vehículos.
- Reporte de fallas.

Ventajas:

Es de sencilla instalación y gestión de los procesos con interfaces amigables para el usuario. Está compuesto por módulos de Control de vehículo, Seguimiento Técnico de los vehículos y Sistemas de Seguridad Automotor. Posee compatibilidad con sistemas ERP desarrollados por la misma empresa que fabrica este software. Permite el control online y en tiempo real de la gestión ilimitada de vehículos, además de poseer un sistema de exportación de información a hojas de cálculo de Microsoft Excel.

Desventajas:

Está diseñado para trabajar sobre plataformas de software privativas pertenecientes a Microsoft y Oracle **(Comsun Car System, 2009).** Teniendo esto como dificultad la necesidad de pagar por las licencias de utilización y explotación de dichas plataformas.

1.3.1.2 SIMPYC

Mides SA es una empresa dedicada al Mantenimiento y la Reparación de Estaciones de Servicio. Dispone de diversas delegaciones distribuidas por la península Ibérica. Debido a que cada día las exigencias del mercado son más altas y al rápido crecimiento que está experimentando dicha área, Mides SA ha decidido

emprender una política de informatización, decantándose por un paquete informático personalizado (SIMPYC, 2011).

Ventajas:

- Gestiona un sistema de facturación con capacidad de crear reportes sobre las funcionalidades que tiene implementadas.
- Gestiona un sistema de control de vehículos.

Desventajas:

- Este sistema solo gestiona informaciones sobre la planificación de rutas de vehículos, dándole poca importancia a los mantenimientos y a la gestión de los accidentes vehiculares.
- Además, está hecho para plataformas de software privativo.

1.3.1.3 GIM

GIM (Gestión Integrada de Mantenimiento) es un sistema de gestión dedicado al mantenimiento de flotas. Realiza el seguimiento de los distintos tipos de mantenimiento así como de avisos y notas generales facilitadas por los operarios. Permite además, generar libros Microsoft Excel totalmente personalizados, permitiendo confeccionar rápidamente y sin límite alguno cualquier tipo de informe deseado (listados, cuadros de análisis, estadísticas y gráficos) en la plataforma más utilizada y conocida del mercado (GIM, 2009). Posee una interfaz de usuario intuitiva, a pesar de su amplia funcionalidad.

Ventajas:

- Posee un sistema de control de vehículos y de gestión de mantenimiento que tiene implementada una amplia gama de funcionalidades.
- Los requerimientos para su implantación son modestos y adaptables a compañías tanto de grande, como de mediano formato.
- Una de sus principales fortalezas es la integración con la Suite Ofimática de Microsoft Office (GIM, 2009), lo que va en oposición de las políticas de nuestro país de migración a software libre.

Desventajas:

- No contempla gestión de los accidentes.
- No incluye la gestión de las inspecciones técnicas con el objetivo de controlar el estado de las partes de los vehículos, lo que impide controlar con mayor eficiencia el estado de los mismos y prever en que momento es más óptimo la realización de los mantenimientos preventivos planificados.

1.3.2 Sistemas nacionales

1.3.2.1 Siscompa.net

Este software garantiza el control y la gestión de la flota automotor de transporte de carga, contribuyendo al ahorro de recursos materiales, combustibles y tiempo. Posee soporte para gestionar hojas de rutas y piezas de repuesto.

Está compuesto por cinco módulos técnicos que intercambian información, además de un módulo adicional para la gestión del sistema y la integración con otros sistemas de gestión de recursos empresariales:

- Módulo de Control de Tráfico.
- Módulo de Control de Técnica.
- Módulo de Control de Seguridad Automotor.
- Módulo de Control de Portadores Energéticos.
- Módulo de Dirección.
- Módulo adicional de Administración.

Ventajas:

El hecho de ser creado en nuestro país, por lo que está programado en función de satisfacer necesidades puntuales comunes en nuestro país, por lo que ha sido desplegado en empresas cubanas.

Desventajas:

Está orientado a Sistemas Operativos de Microsoft a partir de Windows 2000, por lo que no cumple con las políticas de migración a plataformas de software libre del país. (Transoft, 2010).

1.3.2.2 SGestMan

Este es un sistema informático para la organización y control de la actividad de mantenimiento en cualquier organización empresarial, tanto para instalaciones de bienes de producción como de servicios. Su estructura informática basada en una base de datos con filosofía Cliente/Servidor, garantiza una óptima funcionalidad, en redes informáticas, y un adecuado almacenamiento y uso de la información que en ella se registra. Está integrado por módulos, que se encuentran relacionados entre sí, permitiendo una adecuada distribución de la información con que debe contar cualquier organización de mantenimiento (SGestMan, 2010)

Está integrado por los módulos:

- Patrimonio.
- Recursos Humanos.
- Preventivo.
- Solicitudes.
- Órdenes de Servicio.
- Contratos:.
- Informativo:.

Ventajas:

- Apoya los procesos de aseguramiento de la calidad de productos y servicios.
- Gestión de los Recursos Humanos y materiales utilizados en las acciones de Mantenimiento.
- Control de las herramientas, repuestos e insumos, así como el registro de la información de cada uno de los equipos instalados.

Desventajas:

- Este sistema informático no está diseñado teniendo en cuenta las características propias que tiene cada organización, por lo que en ocasiones resulta un poco incomoda su puesta en práctica para controlar los mantenimientos preventivos planificados.
- No gestiona información sobre el mantenimiento vehicular y por tanto no incluye ninguna funcionalidad que verifique y controle el estado de las partes de las unidades.
- No gestiona información alguna sobre los accidentes y no está vinculado con otros procesos que se realizan en las empresas provocando atraso en la información que se gestiona.

1.3.2.3 OffiMant

Es un sistema informático, que facilita una adecuada gestión de las informaciones del mantenimiento a equipos e instalaciones. Se distingue por ser de fácil uso, alta productividad, multiusuario, versatilidad en informes, integración y enlaces y flexibilidad al cambio (Offimant, 2011).

El sistema está integrado por tres de módulos como son:

- Módulo de Administración: Registra la licencia de usuarios que permitirá posteriormente trabajar en el Módulo de Mantenimiento. Registra la base de datos y logra la seguridad de toda la información.
- Módulo de Mantenimiento: Permite definir y mantener toda la información relacionada con los activos, establecer y planificar tareas, generar solicitudes y órdenes de trabajo.

- Módulo de Solicitud de Trabajo: Emite solicitudes desde diferentes estaciones de trabajo al departamento de mantenimiento.

Este sistema realiza funciones como creación de carpeta técnica de los equipos, planificación de tareas, control de órdenes de trabajo, fiscalización de presupuestos de gastos, gestión de productos en almacén y gestión de solicitudes de trabajos.

Ventajas:

Su utilización adecuada puede introducir ahorros en gastos por conceptos de mano de obra, equipos fuera de servicio, reducción de inventarios y toma de decisiones.

Desventajas:

- No proporciona información alguna sobre el mantenimiento vehicular.
- No gestiona tampoco ninguna funcionalidad controlando los recursos humanos que son fundamentales para una correcta gestión del mantenimiento, por lo tanto este sistema no cumple con ninguna funcionalidad que sea de interés para el sistema que se pretende desarrollar.
- No posee una funcionalidad que controle la gestión de accidentes.
- No gestiona los resultados de las inspecciones técnicas, luego de haberse realizado estas.

1.3.2.4 Sistema de Gestión de Mantenimiento Vehicular para el CPNB de Venezuela

Es un sistema creado por el Departamento de Soluciones Empresariales del CEIGE en la UCI, para la gestión de la flota vehicular del Cuerpo de la Policía Nacional Bolivariana de la República Bolivariana de Venezuela y gestiona los siguientes procesos:

- Estructura y Composición: Crea, actualiza o elimina las estructuras que componen la organización, posibilitando una definición jerárquica de sus elementos y permite establecer las estructuras en cada una de las unidades a través de las áreas.
- Clasificadores: Gestiona los tipos de accesorios, mantenimientos, causas de fallas, repuestos, herramientas, tipos de unidades, documentos técnicos y unidades de medidas.
- Configuración: Define los diferentes grupos de unidades de acuerdo a su marca, modelo, régimen de mantenimiento, entre otras características. También precisa la frecuencia de mantenimiento por la cual se van a regir todas los vehículos para la realización de los mantenimientos preventivos planificados que le corresponden y los clientes.
- Persona: Se encarga de lo referente a los Recursos Humanos.

- Vehículo: Gestiona lo referente a los vehículos y sus expedientes.
- Taller: Tramita las órdenes de trabajo y las diferentes inspecciones técnicas que se le realizan a los vehículos.

Ventajas:

Es un sistema hecho en el propio departamento en el que se realiza esta propuesta, por lo que se van reutilizar varios de sus componentes y aplicar técnicas, tecnologías, lenguajes y herramientas que se han utilizado previamente con resultados satisfactorios.

Desventajas:

No posee las mismas especificaciones de requisitos del negocio que esta propuesta, porque varias funcionalidades cambian y se adaptan al país y a la Base de Transporte de la UCI como lo son el Control del Día de la Técnica, la Seguridad activa y pasiva, la Asignación de vehículos y algunos aspectos del Control de vehículo. En el caso del módulo de Control de accidentes cambian el sentido del negocio pues las especificaciones legales en los accidentes varían entre Venezuela y Cuba, pues ya no es necesaria la documentación referente a los fiscales y los juicios en estas situaciones, pero se debe ajustar el negocio a las normativas legales vigentes en Cuba y, en sentido general, respecto al negocio, los datos a guardar cambian casi por completo.

1.3.3 Análisis general de los sistemas

A partir del análisis de la información obtenida sobre el proceso de Control de vehículo y Seguridad activa y pasiva de la Dirección de Transporte de la UCI, se realizó un análisis con los sistemas previamente mencionados, con el fin de conocer como estos gestionan el control de los vehículos y su seguridad y determinar la factibilidad de su utilización. El análisis arrojó como resultado que ninguno puede ser utilizado por la Base de Transporte de la UCI por los siguientes inconvenientes encontrados:

- Casi todos los sistemas antes mencionados son software basados en plataformas privativas, además de que se ejecutan y utilizan plataformas privativas, lo cual impide su utilización en la UCI por su política de uso de tecnologías libres.
- Los sistemas nacionales son aplicaciones de escritorio que limitan la accesibilidad del sistema solo a donde están instalados los recursos para utilizarla y la seguridad se distribuye entre todos los usuarios y el administrador del sistema.

- En el caso de los sistemas internacionales, es imposible utilizarlos en otro hardware o transferirlos sin pagar derechos a sus autores o creadores.
- Su código fuente no puede ser modificado, por lo que luego de la implantación del software, no se pueden añadir nuevas funcionalidades que necesite el cliente.

1.4 Modelo de desarrollo

Para guiar el desarrollo del Sistema Control de Flota y Mantenimiento se emplea el modelo de desarrollo orientado en componentes creado por el CEIGE, caracterizado por ser un modelo estandarizado que especifica las actividades de cada una de las distintas fases del ciclo de vida de los proyectos del centro por las que se debe transitar, unido al conjunto de artefactos a generar en cada una de estas fases sin tener en cuenta las herramientas o métodos utilizados. Está basado en buenas prácticas y principios de varias metodologías, ya sean ágiles o pesadas, haciendo énfasis en las características más convenientes de cada una de ellas. Está orientado a las necesidades y artefactos que son generados durante el desarrollo de cualquier software, define todos los roles involucrados y sus responsabilidades, las actividades que realizan, junto con el flujo de éstas y los artefactos que se deben generar (CEIGE, 2012). Además de ser el modelo establecido por el centro para el desarrollo de todos sus productos, se tienen en cuenta algunas de sus características como son: orientado a componentes, posibilitando la independencia de funciones del sistema a la hora de mantener o modificar el sistema funcional. Además involucra a los clientes y funcionales en el proyecto, permitiendo que ellos también tengan parte de la responsabilidad para el éxito del mismo, dándole de esta forma una mayor claridad a la fase de ingeniería de requisitos (CEIGE, 2012).

La utilización de este modelo se debe a que es el establecido por el centro para el desarrollo de todos sus productos además de presentar como características:

- **Centrado en la arquitectura:** A través de la arquitectura se orientan las prioridades del desarrollo, se resuelven las necesidades tecnológicas y de soporte, se determina la línea base, los elementos de software estructurales a partir de los existentes en la arquitectura de negocio, se interviene en la gestión de cambios y se diseña la evolución e integración del producto.
- **Orientado a componentes:** Según el nivel de significado arquitectónico de los componentes, son orientadas las iteraciones.

- **Iterativo e incremental:** El equipo de arquitectura, los clientes y la alta gerencia, planifican y coordinan las iteraciones, estas constituyen el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo la evolución incremental del producto.
- **Ágil y adaptable al cambio:** Los clientes y funcionales son involucrados en el proyecto, por lo que poseen parte de la responsabilidad del éxito del mismo. Semanalmente se concilian, discuten y aprueban los cambios. El desarrollo de las partes formaliza solo las características principales de la solución, priorizándose de esta manera los talleres y las comunicaciones.
- **Son utilizados solamente los artefactos necesarios para documentar el producto.**
- **Se modela el negocio por procesos y no por casos de uso.**

1.5 Arquitectura de software.

1.5.1 Arquitectura basada en componentes

La Arquitectura de Software se define como la representación de alto nivel de la estructura de un sistema o aplicación, que describe las partes que la integran, las iteraciones entre ellas, los patrones que supervisan su composición y las restricciones a la hora de aplicar esos patrones. De esta forma, aparecen las basadas en componentes. Este tipo es completamente modular y favorece la reutilización de todos sus elementos, incluyendo los que definen las distintas relaciones entre ellos (Fuentes, y otros, 2009). Para el desarrollo del Sistema Control de Flota y Mantenimiento fue definida, por el CEIGE, la utilización de la Arquitectura basada en componentes. Esta se enfoca en la descomposición del software en componentes funcionales, lo cual provee un mayor nivel de abstracción y permite la reutilización de componentes preexistentes.

1.5.2 Patrones Arquitectónicos

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes (POSA 01, 2000). Se pueden ver como la descripción de un problema en particular y recurrente de diseño, que aparece en contextos de diseño arquitectónicos específicos, y representa un esquema genérico demostrado con éxito para su solución (POSA 01, 2000).

1.5.2.1 Modelo-Vista-Controlador

El patrón **Modelo-Vista-Controlador** divide una aplicación en tres componentes: el modelo, la vista y el controlador. El modelo se encarga de administrar el comportamiento y los datos del dominio de la aplicación, responder a los requerimientos de información sobre su estado y las instrucciones de cambiar el mismo. Mantiene el conocimiento del sistema y no depende de ninguna vista o controlador. La vista maneja la visualización de la información, mientras que el controlador analiza los mensajes de eventos que recibe el sistema, modifica u obtiene datos del modelo en respuesta a las peticiones del usuario (POSA 01, 2000). Este patrón arquitectónico será empleado para el desarrollo del Sistema Control de Flota y Mantenimiento debido a que su utilización fue definida por el CEIGE, además de estar implementado dentro del marco de trabajo Sauxe, sobre el cual será desarrollado el sistema.

1.6 Tecnologías

1.6.1 AJAX

El término AJAX es un acrónimo de Asynchronous JavaScript[2] + XML[3]. En sí misma, no es una tecnología, porque en realidad se compone en tecnologías independientes que se unen y complementan en formas nuevas y asombrosas (Garrett, 2005). Estas son XHTML[4], CSS[5], DOM[6], XML, XSLT[7], JSON[8], XMLhttpRequest[9] y JavaScript (Garrett, 2005). En la implementación del Sistema de Gestión de Mantenimiento Vehicular será utilizada para la comunicación entre la capa de la vista y la controladora, por las ventajas que ofrece:

[2] Lenguaje de programación orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

[3] Lenguaje de Marcas Extensible, del inglés Extensible Markup Language.

[4] Lenguaje Extensible de Marcado de Hipertexto, del inglés Extensible Hypertext Markup Language.

[5] Hojas de Estilo en Cascada, del inglés Cascading Style Sheets.

[6] Modelo de Objetos del Documento, del inglés Document Object Model.

[7] Lenguaje de Hojas Extensibles de Transformación, del inglés eXtensible StyLesheeT Language.

[8] Notación de Objetos de JavaScript, del inglés JavaScript Object Notation

[9] Lenguaje de Marcas Extensible/Protocolo de Transferencia de Hipertextos, del inglés Extensible Markup Language/ Hypertext Transfer Protocol

- Mejorar la interacción del usuario con la aplicación pues evita las recargas constantes de la página, realizando el intercambio con el servidor en un segundo plano.
- Sustituir las peticiones http por javaScript que son realizadas al elemento encargado de AJAX, por lo que, las que no necesiten la intervención del servidor obtienen una respuesta inmediata, mientras las otras se realizan de forma asíncrona a través de AJAX.
- Puede ser utilizada en cualquier plataforma o navegador (Garrett, 2005).

1.6.2 Frameworks

1.6.2.1 EXTJS 3.4

Ext JS es un *framework* que soporta JavaScript para construir aplicaciones complejas e interactivas en internet, además de ser la base para Ext. Designer, el cual es una aplicación de escritorio que permite construir aplicaciones web (Sencha Inc, 2013). En la implementación del Sistema Control de Flota y Mantenimiento será empleado para la construcción de todas las interfaces de la aplicación debido a las ventajas que ofrece:

- Crear aplicaciones web mediante javaScript, utilizando componentes predefinidos.
- Ser utilizado sobre cualquier navegador.
- Proveer un balance entre Cliente-Servidor, lo que distribuye la carga, permitiendo que el servidor gestione más clientes al mismo tiempo.
- Poseer un API[10] fácil de usar.
- Contar con licencias comerciales y de código abierto.
- Desarrollar interfaces parecidas a las aplicaciones de escritorio, con la adición de modernos diseños para interfaces de usuario (Sencha Inc, 2013).
- Obtener información del servidor sin estar sujeto a la acción de un usuario.

1.6.2.2 Zend Framework 1.5.0

[10] Interfaz de Programación de Aplicaciones del inglés Application Programming Interface

Zend Framework es un marco de trabajo de código abierto para PHP, desarrollado por Zend; en su más bajo nivel es una librería de componentes escritos en PHP5, para facilitar el desarrollo de sitios web (Zend Technologies Ltd., 2013). Será utilizado dado las ventajas que ofrece:

- Es de código abierto por lo cual no hay que preocuparse por cuestiones de derecho de autor o patentes.
- Se encuentra bajo una licencia de tipo BSD11, lo cual permite su distribución, así como las aplicaciones que se desarrollen con él.
- Posee una clara y amplia documentación.
- Está basado en PHP5 por lo cual es completamente orientado a objetos.
- Implementa el patrón modelo-vista-controlador.
- Sus componentes tienen un bajo acoplamiento por lo que se pueden usar de forma independiente.
- Cuenta con soporte para localización de aplicaciones.
- Contiene adaptadores para gran cantidad de base de datos diferentes.

1.6.2.3 Doctrine Framework 0.11.0

Doctrine es un marco de trabajo ORM[12] para PHP 5.2 inspirado en Hibernate que permite trabajar con un esquema de base de datos como si fuera un conjunto de objetos, no de tablas y registros. Brinda una capa de abstracción de la base de datos muy completa (Doctrine, 2013). Se empleará debido a sus ventajas:

- La separación total del sistema de base de datos, por lo que si en un futuro se decidiera cambiar el motor de base de datos, esto no afectaría al sistema.
- La utilización de los métodos de un objeto de datos desde distintas partes de la aplicación.
- Posee un sistema para evitar tipos de ataque como pueden ser las inyecciones SQL18.

[11] Distribución de Software Berkeley del inglés Berkeley Software Distribution.

[12] Mapeo Objeto-Relacional del inglés *Object-Relational Mapping*.

- Ordena de forma correcta la capa de datos, por lo que facilita el mantenimiento del código.
- Encapsula la lógica de los datos permitiendo hacer cambios que afecten toda la aplicación, únicamente modificando una función.

1.6.2.4 Marco de Trabajo Sauxe

El desarrollo del Sistema Control de Flota y Mantenimiento será realizado utilizando el marco de trabajo Sauxe, desarrollado por el Departamento de Tecnología del CEIGE. Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Está desarrollado sobre lenguaje de programación PHP y en la capa de presentación utiliza Java script y HTML. En la capa de presentación utiliza ExtJs por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. En la capa de negocio utiliza Zend Framework y en la de acceso a los datos Doctrine Framework. (Morera, junio del 2011)

1.6.2.5 Acaxia

Para garantizar la seguridad del Sistema Control de Flota y Mantenimiento será empleado Acaxia, el cual fue desarrollado sobre tecnologías libres como el lenguaje PHP, el gestor de base de datos Postgres y el servidor web Apache. Acaxia gestiona las conexiones a la base de datos, las funcionalidades asociadas y las acciones que realizan. Con esta información se le asignan los permisos a los roles creados en el sistema, o sea, a partir del rol que tenga asignado un usuario, será el nivel de acceso que tendrá este con respecto a las acciones y funcionalidades que podrá realizar. Permite la administración dinámica de perfiles de usuario y consultar todas las acciones realizadas en los sistemas, con toda la información asociada, dígase tiempo, valores e interacción. (Centro de Informatización de la Gestión de Entidades, 2011).

1.6.3 Modelado

1.6.3.1 BPMN[13]

Esta es una notación gráfica que describe la lógica de los pasos de un proceso de negocio que ha sido diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades (BPMN, 2013). Este lenguaje tiene las siguientes características:

- Es independiente de cualquier metodología de modelado de procesos.
- Modela los procesos de una manera unificada y estandarizada, permitiendo un entendimiento entre todas las personas de una organización.
- Crear un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.
- Es fácil de entender.

1.6.3.2 UML [14]

Este es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema orientado a objetos. (UML, 2013) Será utilizado por las siguientes ventajas:

- Es una consolidación de muchas notaciones y conceptos más usados orientados a objetos.
- Utiliza un conjunto de símbolos para representar gráficamente los diversos componentes en relación con el sistema.
- Puede ser utilizado para el modelado de negocio de procesamiento y modelado de requisitos.
- Es independiente del lenguaje de programación.
- Realiza la documentación de todos los artefactos que son generados durante el proceso de desarrollo.
- Se aprende con facilidad.
- Es un lenguaje consolidado.

1.6.4 Lenguajes de programación

[13] Notación de modelado de procesos de negocio, del inglés Business Process Modeling Notation

[14] Lenguaje Unificado de Modelado, del inglés Unified Modeling Language.

1.6.4.1 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlo, lo que brinda la ventaja de probarse directamente en cualquier navegador sin necesidad de procesos intermedios. Este lenguaje permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. (Morera, junio del 2011).

1.6.4.2 PHP

PHP es un lenguaje de programación multiplataforma para la creación rápida de contenidos dinámicos de sitios web. Su nombre surge de la abreviación del concepto PHP Hypertext Preprocessor (PHP, 2013). Su empleo para el desarrollo del Sistema Control de Flota y Mantenimiento se debe a que el marco de trabajo Sauxe, sobre el cual se implementará el sistema, fue realizado con este lenguaje de programación, debido a sus ventajas:

- Posee gran cantidad de documentación y librerías.
- Provee diferentes niveles de seguridad, que pueden ser configurados en el archivo .ini.
- Es de código abierto por lo cual no hay que pagar para la obtención de actualizaciones, ni por el uso de este.
- La interacción dinámica entre la página web y el usuario.
- La creación de aplicaciones para servidores e independientes del navegador.
- Utilizando el mismo código fuente, funciona sobre múltiples plataformas.
- Se pueden leer y escribir archivos, crear imágenes, conectarse a servidores remotos y realizar consultas a base de datos.

1.7 Herramientas

1.7.1 Visual Paradigm

Es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Pressman, 2006). Esta brinda una serie de ventajas que permiten utilizarla:

- Es multiplataforma.

- El diseño es centrado en casos de uso y enfocado al negocio.
- Uso de un lenguaje estándar común que facilita la comunicación para todo el equipo de desarrollo.
- Posee la capacidad de realizar ingeniería directa e inversa.
- Cuenta con una licencia gratuita y comercial.
- Las imágenes y reportes generados son de muy buena calidad.
- Contiene soporte para varios idiomas.
- Es fácil de instalar y actualizar.
- La exportación e importación de ficheros XML.

1.7.2 Servidor de aplicaciones

1.7.2.1 Apache 2.2

El servidor Apache HTTP es un servidor web de código abierto, sólido y para uso comercial. Está desarrollado por la Apache Software Foundation (The Apache Software Fundation, 2013). Este servidor de aplicaciones web ofrece una serie de ventajas:

- Es de código abierto y altamente configurable.
- Soporta lenguajes como Perl, Python, TCL y PHP.
- Posee soporte para SSL[15] y TLS[16].
- La autenticación de base de datos está basada en SGBD[17].

1.7.3 Servidor de base de datos

1.7.3.1 PostgreSQL 8.3

Es un potente motor de base de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de base de datos comerciales (PostgreSQL 8.3, 2013). Posee una serie de ventajas que permiten utilizarlo:

[15] Capa de Conexión Segura, del inglés Socket Secure Layer.

[16] Seguridad en la Capa de Transporte, del inglés Transport Layer Security.

[17] Sistemas de Gestión de Bases de Datos del inglés Database Management System.

- Está distribuido bajo licencia BSD.
- Su código fuente está disponible libremente.
- Utiliza un modelo cliente/servidor.
- Funciona correctamente con grandes cantidades de datos y una alta concurrencia de usuarios accediendo al mismo tiempo.
- Posee acceso encriptado vía SSL.
- Tiene múltiples métodos de autenticación.
- Es multiplataforma.
- Cuenta con una amplia y completa documentación.
- Realiza copias de seguridad.
- Contiene numerosos tipos de datos y permite definir nuevos.

1.7.4 Navegador

1.7.4.1 Firefox 4.0

Es un navegador de Internet con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios (mozilla.org, 2013). Se empleará por sus ventajas:

- Es de código libre y multiplataforma.
- Guarda la contraseña para entrar a un sitio determinado y restaura las ventanas y pestañas que se estuvieran usando antes de cerrar el navegador.
- Seguridad avanzada en la navegación pues permite la identificación de algún sitio web de forma instantánea, posibilita navegar de forma privada y sin registro de lo accedido en internet.
- Establece conexiones seguras a sitios web.
- Está asociado a una de las mayores comunidades de software libre del mundo, lo que permite la implementación rápida de muchas soluciones y que las vulnerabilidades se resuelvan con prontitud.

1.7.4.2 Eclipse-PHP-Galileo

Eclipse es un Entorno de Desarrollo Integrado de código abierto basado en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios (Eclipse Galileo 3.5, 2013). Será empleado por las ventajas que ofrece:

- Es multiplataforma.

- El desarrollo de cualquier lenguaje de programación a través de plugins.
- Es de código abierto y libre.
- Altamente configurable.

1.8 Pruebas de software

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una visión final de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente, el software debe ser probado para descubrir y corregir, el máximo de errores posibles antes de su entrega al cliente. Glen Myers en su libro estableció varias normas que pueden servir acertadamente como objetivos de las pruebas, estas son (Pressman, 2005):

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y de alguna manera, indican la calidad del software como un todo. (Pressman, 2005) Para realizarle las pruebas a los módulos Control de vehículos y Seguridad activa y pasiva, con el objetivo de descubrir y corregir errores existentes en estos, se emplean las pruebas de caja blanca y caja negra.

1.8.1 Pruebas de caja blanca

La prueba de caja blanca o prueba de caja de cristal, es un método de diseño de casos de prueba que utiliza la estructura de control del diseño procedimental para obtener los casos de prueba. (Pressman, 2005) La prueba de caja blanca que se aplicará a la solución desarrollada será la Prueba del Camino Básico, que permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa. (Pressman, 2005)

1.8.2 Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (Pressman, 2005) Los métodos de caja negra que se utilizaran para asegurar la calidad de la aplicación desarrollada serán:

1.8.3 Partición equivalente

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. (Pressman, 2005)

1.8.4 Análisis de valores límite

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límite (AVL) como técnica de prueba. El análisis de valores límite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de las clases. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida. (Pressman, 2005)

1.9 Conclusiones Parciales

Al culminar el estudio realizado en este capítulo, se puede concluir que existen inconvenientes para la utilización de los sistemas existentes tanto nacionales como extranjeros en la informatización del proceso de Control de vehículo y Seguridad activa y pasiva en la Dirección de Transporte de la UCI, por lo que se hace necesaria la creación de un nuevo sistema informático que sea capaz de satisfacer las necesidades que presenta esta área. Además, se definieron en este capítulo el modelo de desarrollo y la arquitectura de software y las tecnologías, herramientas, pruebas de software y lenguajes de programación, modelo y diseño que se van a utilizar en el desarrollo del módulo de Control de vehículo y Seguridad activa y pasiva para la Dirección de Transporte de la UCI.

CAPÍTULO 2: MODELO DE NEGOCIO Y REQUISITOS

2.1 Introducción

En el presente capítulo se describen los procesos de Control de vehículo, Seguridad activa y pasiva, Asignación de vehículos, Control de accidentes y Planificación del Día de la Técnica de la Base de Transporte de la UCI. Se elabora un mapa de procesos en el cual son expuestos los procesos claves y artefactos relacionados con la gestión de estos procesos. Se identifican y registran en el Modelo Conceptual los conceptos fundamentales que se manejan en la Base de Transporte de la UCI en cuanto al Control de vehículo y Seguridad activa y pasiva. Los requisitos funcionales identificados a partir de la aplicación de técnicas, son especificados y validados.

2.2 Modelado de Procesos de Negocio

El Modelado de procesos de negocios representa los procesos de negocio de una empresa para que puedan ser analizados (Nogueras, 2011). Este análisis valida todas las tareas y ciclos mientras éstos son simulados, lo que concreta las tareas que van a llevarse a cabo mediante la reutilización de procesos probados como más eficientes y detectando tareas no realizables. Además se reutilizan los procesos más productivos, que conlleva al ahorro de costes antes de la implementación y que trae mejoras de calidad en los procesos (Nogueras, 2011).

2.1.1.1 Mapa de procesos

Los mapas de procesos de negocios muestran los procesos que se desarrollan dentro de una organización, así como las relaciones que puedan existir entre ellos, lo que permite la visualizar y gestionar la organización de los procesos y la apreciación de las interrelaciones entre los procesos y actividades para perfeccionar los resultados que los clientes desean, por lo que en su elaboración se debe ser claro y preciso con la expresión gráfica a transmitir.

En la confección de los mapas de procesos de negocio del Sistema Control de Flota y Mantenimiento para la Dirección de Transporte de la UCI fue empleada la plantilla conformada por el CEIGE, en la cual se representan los procesos identificados en el Dirección de Transporte de la UCI agrupados por niveles, colocando en un nivel 0 los procesos claves de esta organización y un nivel 1 sus procesos derivados. Dentro de los procesos del nivel 0 se encuentran: Disponibilidad del parque automotor, Control de reparaciones y mantenimiento y Control y análisis de índices de combustible, en el documento CIG-CFM-

N-MTTO-i1101 se especifica información más detallada sobre ellos. Para los procesos identificados en el nivel 1 se crea un documento que contiene: nombre, breve descripción, nivel y proceso padre al que pertenecen (consultar documento entregable CIG-CFM-N-MTTO-i1102). El contenido de este trabajo se enmarca en los procesos de Control de vehículo, Asignación de vehículos, Planificación del día de la técnica, Control de accidentes y Seguridad activa y pasiva pertenecientes al nivel 1 del proceso Disponibilidad del Parque Automotor. Se identifican para cada uno de los procesos del nivel 1 los artefactos de entrada y salida involucrados en el proceso y el formato en que se manejan actualmente en la Dirección de Transporte de la UCI. La matriz de procesos contiene la relación que existe entre los procesos, sistemas o involucrados en estos procesos, exponiendo los artefactos de salida de cada proceso que constituyen entradas para otros y viceversa (consultar Tabla 1 y documento entregable CIG-CFM-N-MTTO-i1102).

		Entradas				
Salidas		Control de vehículo	Asignación de vehículos	Planificación del día de la técnica	Control de accidentes	Seguridad activa y pasiva
	Control de vehículo		Expediente del vehículo			Expediente del vehículo
	Asignación de vehículos	Modelo de inspección técnica				
	Planificación del día de la técnica	Expediente del vehículo				Modelo de inspección técnica
	Control de accidentes					
	Seguridad activa y pasiva					

Tabla 1 Matriz de relación de procesos de negocio de nivel 1.

2.2.2 Descripción de procesos de Negocio

2.2.2.1 Descripción del proceso Control de vehículo.

El proceso de Control de vehículo tiene como objetivo principal controlar la información referente a los vehículos del área de Transporte de la UCI, manejando todo lo referente a la confección de los expedientes de cada uno de los vehículos de esta área. Este proceso comienza con el registro de los vehículos por tipo. Luego se realiza la conciliación entre el registro de vehículos del MININT. Se verifica la información que tiene el registro de vehículos del MININT con la información contenida en el registro de vehículos del área de Transporte de la UCI. Luego se realiza la Asignación de vehículos (subproceso), por lo que se procede a ejecutar el flujo básico de este subproceso, asignando de esta forma un vehículo a un área determinada.

A continuación se presentan los artefactos de entrada y de salida del proceso Control de vehículo:

Proceso de Control de vehículo	
Artefactos de Entrada:	Artefactos de Salida:
Circulación del vehículo	Registro de vehículos área de Transporte UCI.
Registro de vehículos del MININT	Expediente del vehículo.
Factura	

Tabla 2 Artefactos de entrada y salida del proceso Control de vehículo.

Para obtener mayor información sobre la descripción de este proceso, consultar documento entregable CIG-CFM-N-MTTO-i1501.

2.2.2.2 Descripción del proceso Asignación de vehículos.

El proceso de Asignación de vehículos tiene como objetivo principal asignar un vehículo a un área de la UCI. Este proceso comienza por definir las áreas a las que se les van a dar servicios de mantenimiento. Luego se asignan el vehículo a un área y se emite el acta de entrega por cada vehículo asignado. Posteriormente se definen los responsables del vehículo y se registran estos responsables en el documento Relación de responsables y choferes. Finalmente se accede a la Asignación de combustible (subproceso).

A continuación se presentan los artefactos de entrada y de salida del proceso Asignación de vehículos:

Proceso de Asignación de vehículos	
Artefactos de Entrada:	Artefactos de Salida:
Ninguno.	Acta de entrega
	Listado de áreas (UCI)
	Relación de responsables y choferes

Tabla 3 Artefactos de entrada y salida del proceso Asignación de vehículos.

Para obtener mayor información sobre la descripción de este proceso, consultar documento entregable CIG-CFM-N-MTTO-i1509.

2.2.2.3 Descripción del proceso Planificación del día de la técnica.

Este proceso tiene como objetivo principal la planificación del día de la técnica, comenzando por elaborar el plan del día de la técnica por cada tipo de vehículo. Luego se envía una notificación al responsable del vehículo con hora, lugar y fecha. Posteriormente se realiza la revisión al vehículo chequeando cada una de las partes del vehículo de acuerdo al Modelo de inspección, se llena este modelo con los datos de la inspección y luego se archiva.

A continuación se presentan los artefactos de entrada y de salida del proceso Planificación del día de la técnica:

Proceso de Planificación del día de la técnica	
Artefactos de Entrada:	Artefactos de Salida:
Ninguno.	Plan del día de la técnica por tipo de vehículo.
	Email.
	Modelo de inspección.
	Expediente del vehículo.

Tabla 4 Artefactos de entrada y salida del proceso Asignación de vehículos.

Para obtener mayor información sobre la descripción de este proceso, consultar documento entregable CIG-CFM-N-MTTO-i1512.

2.2.2.4 Descripción del proceso Control de accidentes.

Este proceso tiene como objetivo principal controlar los accidentes de los vehículos. Este proceso comienza con la elaboración del documento con la relatoría de los hechos y el croquis del accidente creado por el Técnico General de seguridad automotor. Luego se realiza la denuncia de los hechos a la policía y se elabora otro documento con los costos provocados por los daños del vehículo. Posteriormente

se realiza archiva toda la información del accidente en el expediente de accidentes y finalmente se realiza el informe de accidentabilidad por parte del Técnico General.

A continuación se presentan los artefactos de entrada y de salida del proceso Control de accidentes:

Proceso de Control de accidentes	
Artefactos de Entrada:	Artefactos de Salida:
Ninguno.	Control de golpes y abolladuras
	Relatoría de los hechos
	Denuncia
	Documento con el costo de los daños
	Expediente de accidentes
	Informe de accidentabilidad

Tabla 5 Artefactos de entrada y salida del proceso Control de accidentes.

Para obtener mayor información sobre la descripción de este proceso, consultar documento entregable CIG-CFM-N-MTTO-i1511.

2.2.2.5 Descripción del proceso Seguridad activa y pasiva.

Este proceso tiene como objetivo principal controlar la seguridad activa y pasiva de los vehículos. Este proceso comienza con la elaboración de la Planificación del día de la técnica (subproceso) y luego revisar la seguridad activa y pasiva del vehículo en función del modelo correspondiente. Luego es llenado el control de golpes y abolladuras del vehículo por el Técnico General, quien luego archiva el documento en el Expediente del vehículo.

A continuación se presentan los artefactos de entrada y de salida del proceso Seguridad activa y pasiva:

Proceso de Control de accidentes	
Artefactos de Entrada:	Artefactos de Salida:
Modelo de inspección.	Control de golpes y abolladuras (doc.)
	Expediente del vehículo (doc.)

Tabla 6 Artefactos de entrada y salida del proceso Seguridad activa y pasiva.

Para obtener mayor información sobre la descripción de este proceso, consultar documento entregable CIG-CFM-N-MTTO-i1513.

2.2.3 Patrones de Control de Flujo

2.2.3.3 Secuencia

Secuencia es el patrón básico de todo flujo de trabajo. Se requiere cuando hay una dependencia entre dos actividades, de tal forma que una actividad no pueda iniciarse antes de que otra haya terminado.

Un diagrama de procesos de negocio ilustra este patrón como una serie de actividades conectadas por flujos de secuencia. La secuencia indica que una actividad será habilitada, solo hasta que la actividad anterior sea ejecutada. Para ilustrar mejor esta idea se hará uso del concepto de "token", el cual viaja de un punto de inicio a un punto final u objetivo del proceso según el direccionamiento de las flechas. En este patrón, el token pasará a la siguiente actividad solo cuando la actividad precedente haya sido ejecutada (Workflow Management Coalition, 2013). Por ejemplo en el proceso de Asignación de vehículos no se puede emitir el acta de entre hasta tanto no se haya asignado el vehículo a algún área. Ver imagen 1.

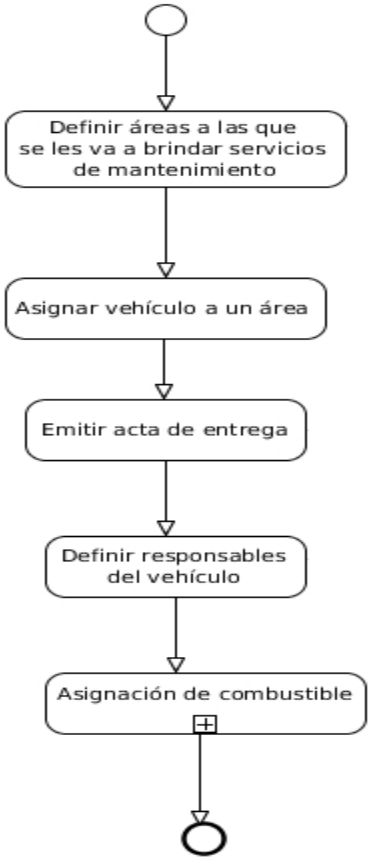

Imagen 1: Proceso de Asignación de vehículos.

2.2.3.4 Selección exclusiva

Ocurre cuando en un punto del flujo de trabajo se escoge solo una de varias ramas del proceso. Generalmente esta decisión se toma basándose en datos de control del flujo de proceso. Si uno de los dos caminos debe ser escogido, una compuerta exclusiva (elemento de divergencia, que en este caso es la coincidencia entre el registro de vehículos del MININT) puede ser usada. Las transiciones de salida de la compuerta exclusiva tendrán asociadas una regla de negocio booleana que será evaluada para determinar cuál secuencia del flujo deberá ser seleccionada para continuar al siguiente paso. Cuando un *token* llega a una compuerta exclusiva, las condiciones serán evaluadas para definir el camino a seguir (es decir, aquel que sea válido) y el *token* continuará hasta la próxima actividad. Por cada *token* que entre a una compuerta exclusiva, solo saldrá un token (Workflow Management Coalition, 2013). Ver imagen 2.

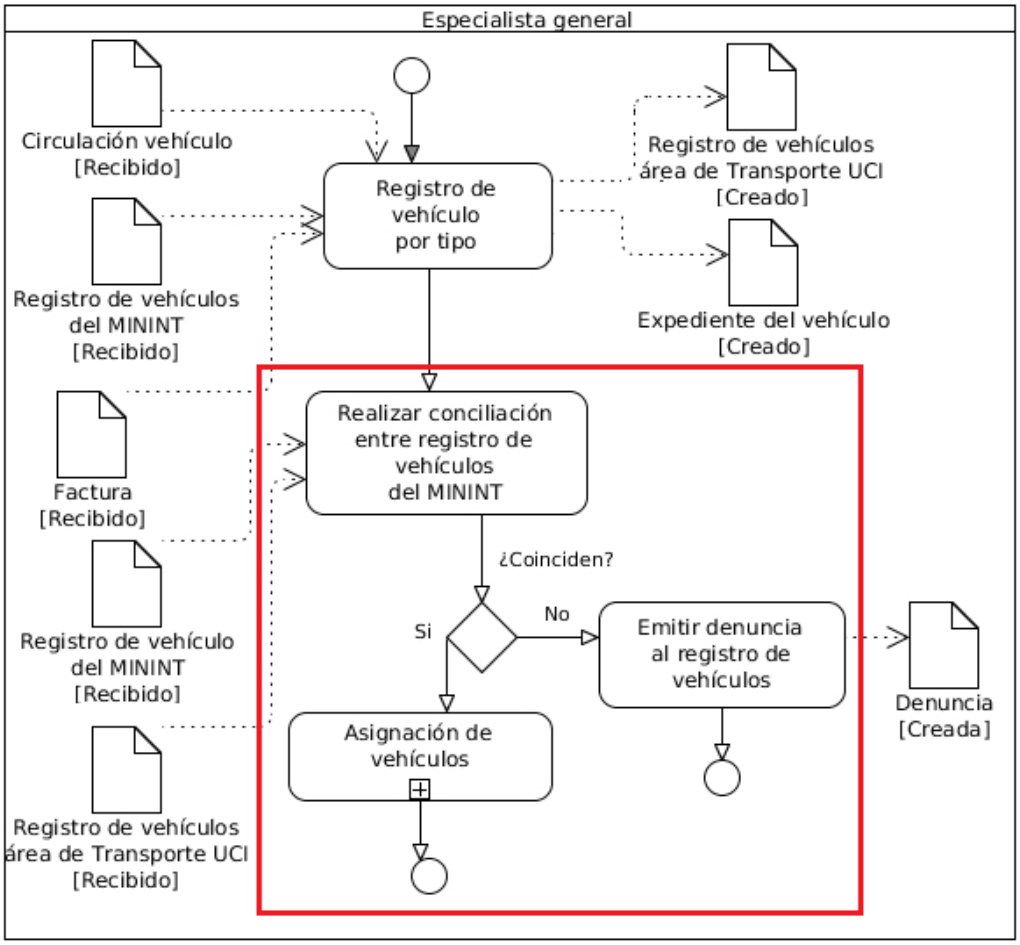

Imagen 2: Ejemplo de Selección Exclusiva

2.2.4 Validación del Negocio

La validación del modelado de procesos de negocio se realizó aplicando la técnica Revisión Técnica Formal, donde los clientes revisaron cada documento de descripción de procesos de negocio realizado con el objetivo de validar su correcta elaboración y que el flujo de actividades de cada proceso estuviera en correspondencia con la información brindada.

El centro de CALISOFT de la UCI validó el modelado de negocio realizado, mediante dos iteraciones, revisando elementos como ortografía, formato y aspectos técnicos, fueron liberados los artefactos Descripción de procesos de negocio y Mapa de procesos de negocio. (Ver **Anexo 5**).

2.2.5 Modelo Conceptual

El modelo conceptual es un diagrama conformado con los conceptos que son significativos para el área que se analice, así como las relaciones existentes entre ellos. Permite identificar, organizar y realizar razonamientos sobre los componentes y comportamientos del sistema, siendo la guía para el proceso de diseño del software, pudiéndose usar además como referencia para evaluar un diseño particular y razonar sobre la solución realizada. El modelo conceptual del proceso Control de vehículos fue confeccionado con todas las clases conceptuales identificadas en este proceso junto con los atributos y relaciones existentes entre las mismas (consultar Imagen 3).

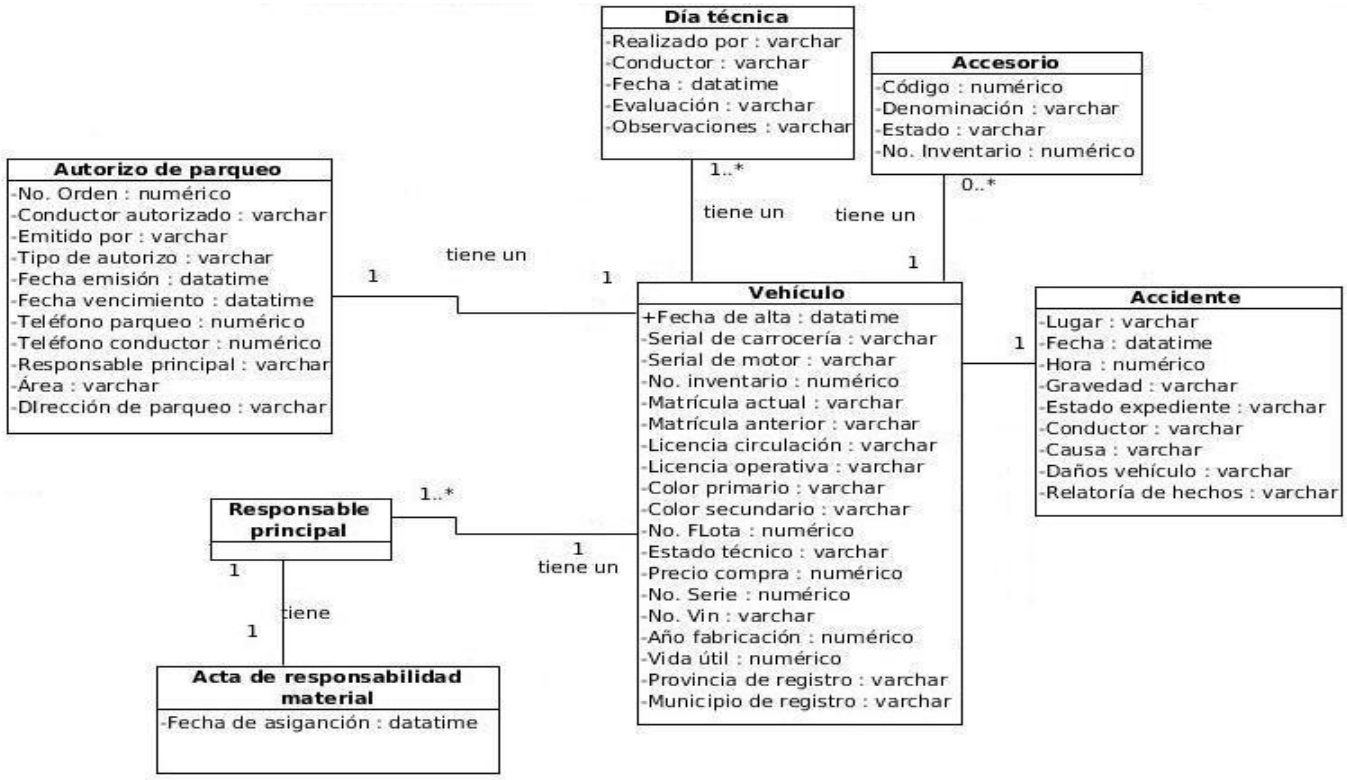

Imagen 3 Modelo conceptual del proceso Control de vehículo.

2.3 Requerimientos de software

Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. (Sommerville, 2005) y que también puede tomarse como una condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. Estos pueden dividirse en dos categorías: requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales son los que definen las funciones que el sistema será capaz de realizar, además, pueden ser una descripción de lo que un sistema debe hacer, describiendo las transformaciones que el sistema realiza sobre las entradas para producir las salidas. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones. (La Ingeniería de Requerimientos y su importancia en el desarrollo de proyectos de software, 2005). Por otra parte, un requisito no funcional especifica algo sobre el propio sistema, y cómo debe realizar sus funciones. La captura de requisitos es la

actividad, mediante la cual, el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema.

2.3.1 Técnicas para captura de requisitos

Entre las técnicas aplicadas para la captura de los requerimientos se encuentran:

Entrevista

Se planificaron encuentros con los proveedores de requisitos del área de transporte de la UCI con el objetivo de identificar los requisitos funcionales de la propuesta de solución, a través de entrevistas se fueron obteniendo las necesidades de informatización del cliente en cada uno de los procesos identificados.

JAD[18]

El equipo de desarrollo junto con los proveedores de requisitos, fueron revisando los prototipos de interfaz de usuario realizados, con el objetivo de verificar que los requisitos identificados satisficieran las necesidades de los proveedores de requisitos.

Lluvia de ideas

Se realizaron talleres con el equipo de desarrollo (analistas, arquitecto de BD y de sistema) en los cuales fueron analizados cada uno de los requisitos identificados, fluyendo variadas ideas acerca de la realización de los mismos, lo cual arrojó como resultado que se obtuviera una visión más amplia de lo que se quería implementar, favoreciéndose de esta forma el avance de futuras etapas en el desarrollo de la propuesta de solución.

2.3.1.1 Requisitos funcionales

Los requisitos funcionales identificados de los procesos Control de vehículo y Seguridad activa y pasiva de la Base de Transporte de la UCI son:

[18] JAD (Joint Application Design, por sus siglas en inglés), técnica basada en reuniones participativas entre clientes, directivas y desarrolladores enfocándose más en el negocio que en el aspecto técnico. (Yatco, Mei C., 1999)

- RF 1. Asociar conductores.
- RF 2. Quitar conductores.
- RF 3. Asociar responsable principal.
- RF 4. Quitar responsable principal.
- RF 5. Agrupación de requisitos: Gestión de autorizo de parqueo:
  - RF 5.1 Buscar autorizo de parqueo.
  - RF 5.2 Listar autorizo de parqueo.
  - RF 5.3 Adicionar autorizo de parqueo.
  - RF 5.4 Modificar autorizo de parqueo.
  - RF 5.5 Eliminar autorizo de parqueo.
  - RF 5.6 Imprimir autorizo de parqueo.
- RF 6. Agrupación de requisitos: Gestión del día de la técnica.
  - RF 6.1 Buscar día técnica.
  - RF 6.2 Listar día de la técnica.
  - RF 6.3 Adicionar día de la técnica.
  - RF 6.4 Modificar día de la técnica.
- RF 7. Asignar vehículo.
- RF 8. Agrupación de requisitos: Gestionar accesorios a los vehículos.
  - RF 8.1 Listar accesorios de un vehículo.
  - RF 8.2 Adicionar accesorios de un vehículo.
  - RF 8.3 Eliminar accesorios de un vehículo.
- RF 9. Agrupación de requisitos: Gestionar accidente.
  - RF 9.1 Buscar accidentes.
  - RF 9.2 Listar accidentes.
  - RF 9.3 Adicionar accidente.
  - RF 9.4 Modificar accidente.
  - RF 9.5 Cancelar accidente.
  - RF 9.6 Imprimir accidente.
- RF 10. Agrupación de requisitos: Gestionar vehículos.

- RF 10.1 Buscar vehículo.
- RF 10.2 Búsqueda avanzada de vehículo.
- RF 10.3 Listar vehículos.
- RF 10.4 Adicionar vehículo.
- RF 10.5 Modificar vehículo.
- RF 10.6 Imprimir expediente del vehículo.

- RF 11. Registro accidentes período.
- RF 12. Coeficiente de aprovechamiento de la capacidad.
- RF 13. Coeficiente de tráfico.
- RF 14. Historial de accidentes del vehículo.
- RF 15. Historial de asignaciones del vehículo.
- RF 16. Informe de explotación.
- RF 17. Vehículos operativos.
- RF 18. Vehículos operativos por área.
- RF 19. Registro de accidentes por área.
- RF 20. Registro de vehículos período.
- RF 21. Vehículos en mantenimiento.
- RF 22. Vehículos inspeccionados en un período.
- RF 23. Vehículos paralizados.
- RF 24. Vehículos paralizados por área.

2.3.2 Especificación de requisitos

La especificación de los requisitos identificados se realizó siguiendo los pasos descritos en la plantilla conformada por el CEIGE.

Precondiciones	El usuario ha sido validado. Se ha registrado en el sistema el grupo al cual va a pertenecer el vehículo.
Flujo de eventos	
Flujo básico	
Se introducen los datos generales del vehículo: Grupo de vehículo (seleccionar)	

	Serial carrocería Serial de motor No. Inventario Fecha de alta Matrícula anterior Licencia circulación Licencia operativa Matrícula actual Color primario Color secundario No. Flota Estado técnico Precio venta Precio compra Vida útil No. Serie No. Vin Año de fabricación Provincia Municipio Tipo de combustible Tipo de actividad Capacidad Unidad
	En la pestaña Inicializar valores se introducen los siguientes datos: Se inicializa el valor del medidor de la unidad para los tipos de mantenimiento definidos en el grupo: Valor del medidor
	Se inicializan los siguientes datos de cada unos de los tipos de mantenimiento definidos en el grupo en la pestaña Inicializar valores: Fecha de último servicio Valor de último servicio
	Se introducen los datos del seguro en la pestaña Seguro: Fecha inicio del seguro Fecha de vencimiento del seguro No. de póliza Nombre de la aseguradora
	Se introducen los datos de los accesorios en la pestaña Accesorios:

	Se registran los accesorios del vehículo ver requisito funcional Gestionar accesorios del vehículo..
	En la pestaña Propiedades se introducen los valores de las propiedades definidas en el grupo (ver validación 2).
	Se introducen las observaciones del vehículo: Observaciones
	El sistema valida (ver validación 1) los datos introducidos.
	Si los datos son correctos el sistema los registra.
	El sistema confirma el registro de los datos.
	Concluye el requisito.
Post-condiciones	
	Se registró en el sistema una nueva unidad.
Flujos alternativos	
Flujo alternativo 4.a La unidad no está asegurada.	
	Volver al paso 5 del flujo básico.
Post-condiciones	
1	NA
Flujo alternativo 5.a La unidad no tiene accesorios.	
	Volver al paso 6 del flujo básico.
Post-condiciones	
	NA
Flujo alternativo 9.a Información errónea.	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 8 del flujo básico.
Post-condiciones	
	NA
Flujo alternativo 9.b Información incompleta.	
	El sistema señala los datos vacíos y permite corregirlos.
	El usuario corrige los datos.
	Volver al paso 8 del flujo básico.
Post-condiciones	
	NA
Flujo alternativo *.a El usuario cancela la acción.	
	Concluye el requisito.
Post-condiciones	
	NA

Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CIG-CFM-N-MTTO-002.	
	Las propiedades se definen en el grupo de vehículos, el valor de esta se puede definir en el grupo de unidades o en el vehículo.	
Relaciones	**Requisitos Incluidos**	Paso 8: Listar tipos de mantenimientos del grupo, en la agrupación Gestionar tipos de mantenimientos al grupo. Paso 9: Listar propiedades a las unidades, en la agrupación Gestionar propiedades al grupo. Paso 4: Gestionar accesorios de los vehículos.
	Extensiones	NA
Conceptos	**Vehículo**	**Visibles en la interfaz:** Serial carrocería Serial de motor No. Inventario Fecha de alta Matrícula anterior Licencia circulación Licencia operativa Matrícula actual Color primario Color secundario No. Flota Estado técnico Precio venta Precio compra Vida útil No. Serie No. Vin Año de fabricación Provincia Municipio Tipo de combustible Tipo de actividad Capacidad Unidad de medida

	Grupo de vehículo	**Visibles en la interfaz:** Nombre Marca Modelo Tipo de vehículo
	Tipos de mantenimiento	**Visibles en la interfaz:** Tipo de mantenimiento Fecha de último servicio Valor de último servicio
	Documentos técnicos	**Visibles en la interfaz:** Nombre Código Descripción
	Propiedades	**Visibles en la interfaz:** Propiedad Valor
	Seguro	**Visibles de la interfaz:** Asegurada No. de póliza Fecha inicio Fecha de fin Nombre de la aseguradora
	Accesorios	**Visibles en la interfaz:** Denominación Código Cantidad Estado
Requisitos especiales	NA	
Asuntos pendientes	Aquí solo varía la pestaña accesorios y datos generales, el resto se queda igual.	

Tabla 7 Especificación del requisito Adicionar vehículo.

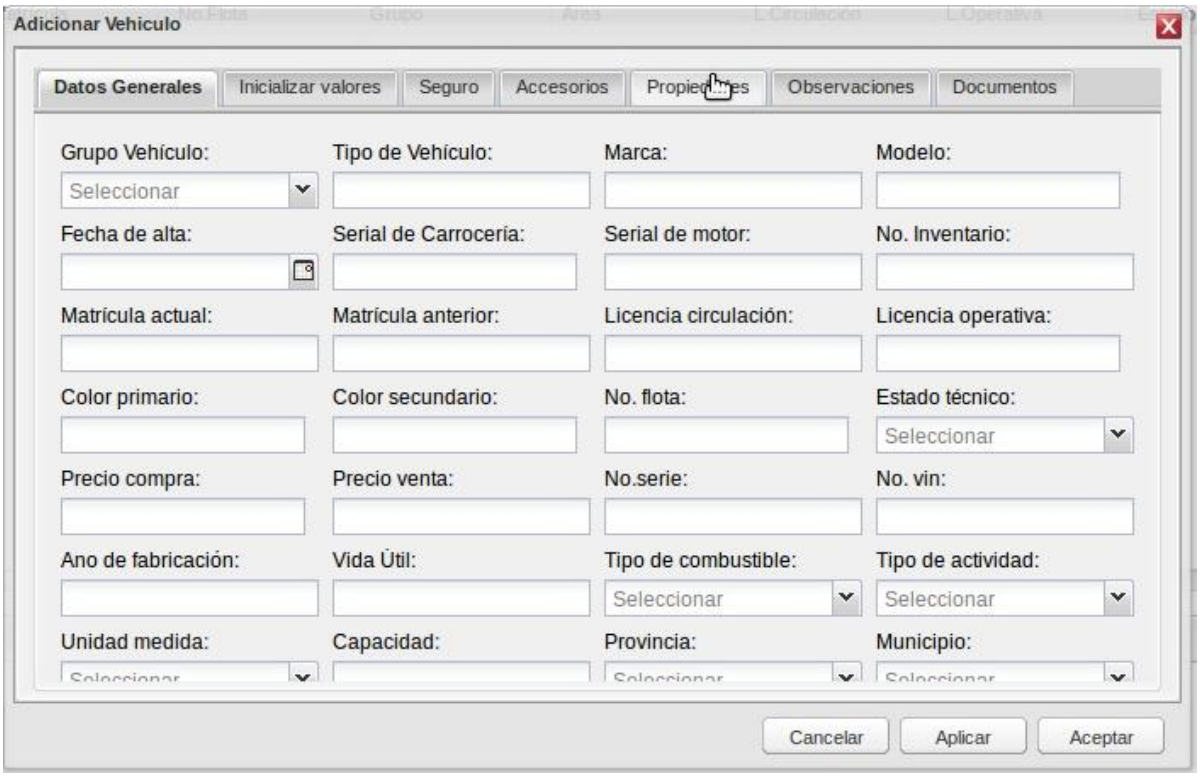

Imagen 4 Prototipo de interfaz elemental de usuario

2.3.3 Administración de requisitos

La administración de los requisitos es un enfoque sistemático para obtener, organizar y documentar los mismos, así como mantener un acuerdo entre el cliente y el equipo del proyecto en los posibles cambios. La clave para una efectiva administración de requisitos es: mantener un enunciado claro, junto con los atributos para cada tipo de requisitos y su seguimiento con otros o con elementos del proyecto. (Sandoval Carvajal, y otros, 2006).

La administración de los requisitos identificados como parte de la propuesta solución se realizó utilizando la herramienta Visual Paradigm, definiendo los siguientes elementos de trazabilidad:

- Requisitos.
- Modelo conceptual (entidades del negocio).
- Componentes.
- Diseños de casos de prueba.

Mediante un diagrama de requisitos se definió la relación entre los elementos de trazabilidad propuestos, y se realizaron las siguientes matrices de trazabilidad:

- requisitos-modelo conceptual-entidades del negocio.
- requisitos-componente.
- requisitos-diseños de casos de prueba.

Para definir la relación entre los elementos de trazabilidad mencionados anteriormente se realizó el diagrama de requisitos. Consultar el **Anexo 2**.

2.3.4 Priorización de Requisitos

En el desarrollo de software existe generalmente diversidad de opiniones e incompatibilidades entre las prioridades de los participantes (desarrolladores, clientes y usuarios finales). En estos casos se deben desarrollar soluciones aceptables para los principales involucrados, lo que implica un proceso de negociación de los requerimientos en conflicto. La clave para la negociación es poder determinar el conjunto de requerimientos prioritarios.

La priorización de los requisitos se realizó utilizando la plantilla evaluación de requisitos propuesta por el CEIGE (consultar documento entregable CIG-CFM-MTTO-i6101.xls). Los criterios definidos para determinar la complejidad de los requisitos y su priorización son los siguientes:

- **Criterios de Complejidad:** Para la determinación de la complejidad de los requisitos se analizan individualmente los criterios siguientes, llegando al resultado de si el requisito es de complejidad Alta, Media o Baja. La clasificación de la complejidad permite estimar el esfuerzo de implementación del requisito y contribuye a la decisión sobre la inclusión en las etapas de desarrollo del software.
- **Diferentes comportamientos:** Un mismo requisito se comporta de manera diferente ante determinadas situaciones.
- **Formas de inicialización:** Un mismo requisito puede ser inicializado de diferentes formas.
- **Consultas a fuentes de almacenamientos:** Los requisitos pueden presentar diversidad en la cantidad y complejidad de la interacción con la fuente de datos.
    - Base de Datos.
    - Ficheros.
    - Otros.

- **Restricciones de validación:** Complejidad de todas las validaciones que lleve un requisito, tanto las validaciones en el lado del cliente, como en el servidor.
- **Grado de reutilización:** Complejidad de un requisito, para poder ser reutilizado por otros.
- **Lógica de negocio:** Los requisitos pueden presentar diferentes niveles de complejidad para la implementación de la lógica de negocio que contienen.

La aplicación de los indicadores antes definidos arrojó los siguientes resultados:

Resumen	
Cant. de requisitos con complejidad Alta	6
Cant. de requisitos con complejidad Media	13
Cant. de requisitos con complejidad Baja	4

Tabla 8 Resumen de la aplicación de los indicadores.

2.3.5 Validación de Requisitos

La validación de los requisitos se realiza con la finalidad de comprobar que los requerimientos identificados sean correctamente descritos, además de brindar información precisa, consistente, realista en todo momento, y que sean verificables. A su vez debe definir lo que el usuario desea del producto final, que los errores que hayan sido detectados sean corregidos y el resultado del trabajo cumpla con los estándares establecidos para el proceso, el proyecto y el producto, por lo que para la validación de los requisitos fueron aplicadas las siguientes técnicas:

Revisión Técnica Formal

La revisión de las especificaciones de los requerimientos de la propuesta de solución fue realizada con los proveedores de requisitos y los miembros del equipo de desarrollo (analista principal y jefe de proyecto) verificando que no existieran errores en el contenido o malas interpretaciones, información errónea o incompleta, dando como resultado que fueran aprobados los requisitos funcionales, teniendo como constancia el acta de aceptación (Consultar Anexo 6).

Prototipos

A partir de las especificaciones de requisitos fueron conformados prototipos de interfaz de usuario. Validando de esta manera que los requerimientos estaban en concordancia con las necesidades del

cliente. El empleo de esta técnica ofreció como resultados que el cliente tuviera una idea de la estructura de la interfaz de usuario y se favoreciera la comunicación con el mismo, ya que este tenía una visión inicial del módulo a través del cual se gestionarían los procesos de Control de vehículo y Seguridad activa y pasiva.

Generación de casos de prueba

Fueron definidos y diseñados casos de pruebas para cada requerimiento especificado, con el objetivo de verificar el estricto cumplimiento de los mismos. La utilización de esta técnica ofreció los siguientes resultados: fueron identificados los posibles escenarios de los requisitos, así como los juegos de datos de los campos determinados en estos, con el objetivo de validar el flujo básico de cada uno de los requisitos identificados.

Liberación por el Departamento de calidad de CEIGE

Los requisitos fueron validados en 3 iteraciones, emitiendo un acta de liberación (consultar documento entregable Acta de Liberación de ER Mantenimiento)

2.4 Requisitos No Funcionales

Los requisitos no funcionales de un software detallan las propiedades o características que hagan al sistema más atractivo y usable, generalmente se distribuyen en diferentes categorías como el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, fiabilidad y la seguridad (Chaves, 2005). Los requisitos no funcionales del sistema Control de Flota y Mantenimiento de la Dirección de Transporte de la UCI fueron identificados por el Analista principal del proyecto, algunos de estos requisitos son:

- El idioma de todas las interfaces de la aplicación será el español.
- El sistema será consistente en el uso de abreviaturas, usará la misma abreviación siempre para la misma palabra y nunca en un elemento de selección o menú.
- Los flujos de navegación para la gestión de cualquier concepto del negocio no excederán las 3 interfaces.
- No se utilizarán textos extensos para las etiquetas de la interfaz de usuario, en su lugar se usarán iconos para actividades típicas (Adicionar, Modificar, Eliminar). Los íconos previstos para dichas actividades quedarán plasmados en el Manual de pautas de diseño.

- El sistema expondrá el menú general desde cualquiera de sus páginas.
- El sistema mostrará las opciones desactivadas siempre que no se hayan cumplido las condiciones previas para su activación, evitando así errores del usuario en la gestión.
- El sistema contará con un Manual de usuario.
- El sistema estará disponible durante 24 horas, los 7 días de la semana, los 365 días del año.
- El sistema tendrá un respaldo de la información del centro de datos, permitiendo la recuperación ante la pérdida parcial o total de la información.
- El sistema manejará la seguridad de acceso y administración de usuarios mediante el otorgamiento de privilegios y roles, asignación de perfiles.
- Se concederá acceso al sistema a partir de un nombre de usuario y una contraseña.
- El sistema concederá acceso a cada usuario autenticado sólo a las funciones que le estén permitidas de acuerdo a la configuración del sistema. Los menús serán generados a partir del propio perfil del usuario.

Consultar documento entregable Especificación de requisitos de software (CIG-CFM-N-MTTO-i2301) para obtener información acerca de los restantes requisitos no funcionales.

2.5 Conclusiones parciales

Con el modelado del negocio de los procesos Control de vehículo y Seguridad activa y pasiva fueron generados una serie de artefactos que crearon las condiciones para la captura de los requerimientos funcionales de estos procesos, los cuales fueron identificados, especificados y validados, garantizando con esto último, que los errores existentes que no hubieran sido corregidos pudieran afectar futuras etapas del desarrollo del Sistema de Control de Flota y Mantenimiento, dando paso así, a la realización del diseño de la propuesta de solución.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

En el presente capítulo se aborda inicialmente el diseño de la propuesta de solución. Luego son mencionados los mecanismos de diseño utilizados, los diagramas de clases concebidos a partir de lo definido en el análisis y los patrones empleados en la modelación. Mediante métricas es validado el diseño propuesto. Se muestra el modelo de datos, el diagrama de componentes y el modelo de despliegue. Además, se valida mediante pruebas el software realizado, haciendo uso específico de las pruebas de caja blanca y caja negra.

3.2 Diseño

En el diseño se modela el sistema y es donde convergen los requisitos del cliente, las necesidades de negocio y las consideraciones técnicas, uniéndose en la formulación de un producto o sistema y creándose un modelo de software que, a diferencia del modelo de análisis, proporciona detalles acerca de las estructuras de datos, la arquitectura, las interfaces y los componentes del software que son necesarios para implementar el sistema (Pressman, 2006).

3.2.1 Mecanismos de diseño

Los mecanismos de diseño son utilizados con el objetivo de simplificar los diagramas de clases. Cada diseñador acorde a sus necesidades puntuales establece sus propios mecanismos de diseño, dirigidos a cumplir con el objetivo antes descrito pero siempre teniendo en cuenta los patrones y estilos seleccionados (Cabrera Pereira., y otros, 2009).

Mecanismo de diseño para las clases controladoras.

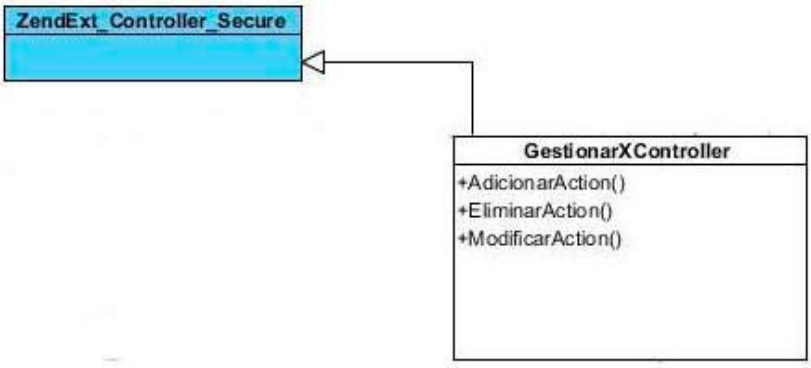

Imagen 5: Mecanismo de diseño para las clases controladoras.

Todas las clases controladoras definidas en el diseño propuesto heredan de la clase ZendExt_Controller_Secure, ya que en ella se incluyen numerosas funcionalidades comunes en todas las controladoras. Mecanismo de diseño para las clases modelos.

Mecanismo de diseño para las clases modelos.

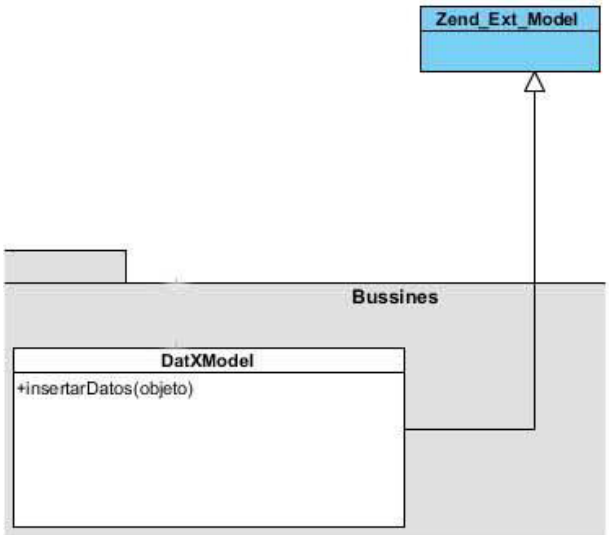

Imagen 6 Mecanismo de diseño para las clases modelos

Todas las clases modelos o model definidas en el diseño heredan de la clase ZendExt_Model, ya que ésta incluye las principales funciones para el manejo de los datos.

Mecanismo de diseño para las clases del dominio.

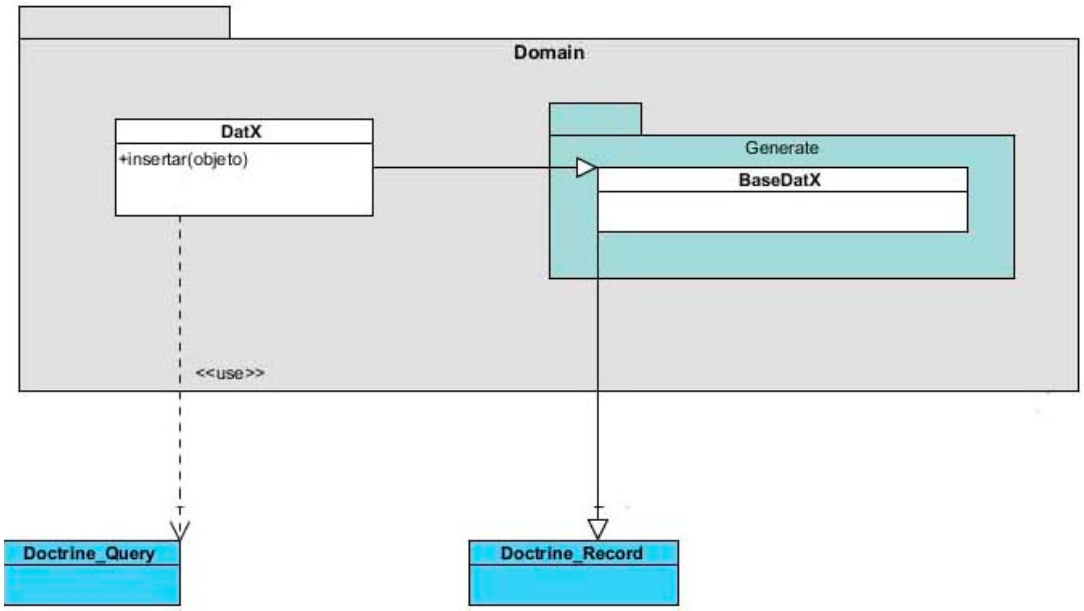

Imagen 7 Mecanismo de diseño para las clases del dominio

La clase DatX representa a las clases del dominio (*Domain*), que usan a la Clase Doctrine_Query del marco de trabajo Doctrine, para el acceso a la base de datos a través del lenguaje DQL; estas clases además heredan de las clases cuyo prefijo es Base con los atributos mapeados de las tablas de la base de datos; en este caso BaseDatX. Todas las clases con el prefijo Base como BaseDatX heredan de Doctrine_Record que permite agrupar en registros u objetos mapeados los datos de las tablas. En el documento entregable Modelo de diseño (CIG-CFM-N-MTTO-i3801) se encuentran el resto de los mecanismos de diseños definidos.

3.2.2 Diagramas de clases del diseño

El diagrama de clases del diseño describe gráficamente el comportamiento del sistema. Además de describir las especificaciones de las clases de software y de las interfaces en una aplicación, contiendo información como asociaciones, atributos, métodos, y dependencias. (Visconti, y otros, 2011)

En las imágenes a continuación se muestran las clases contenidas dentro del paquete de dominio con sus respectivos métodos y en la segunda imagen se muestra el diagrama que modela el Requisito Gestionar vehículo.

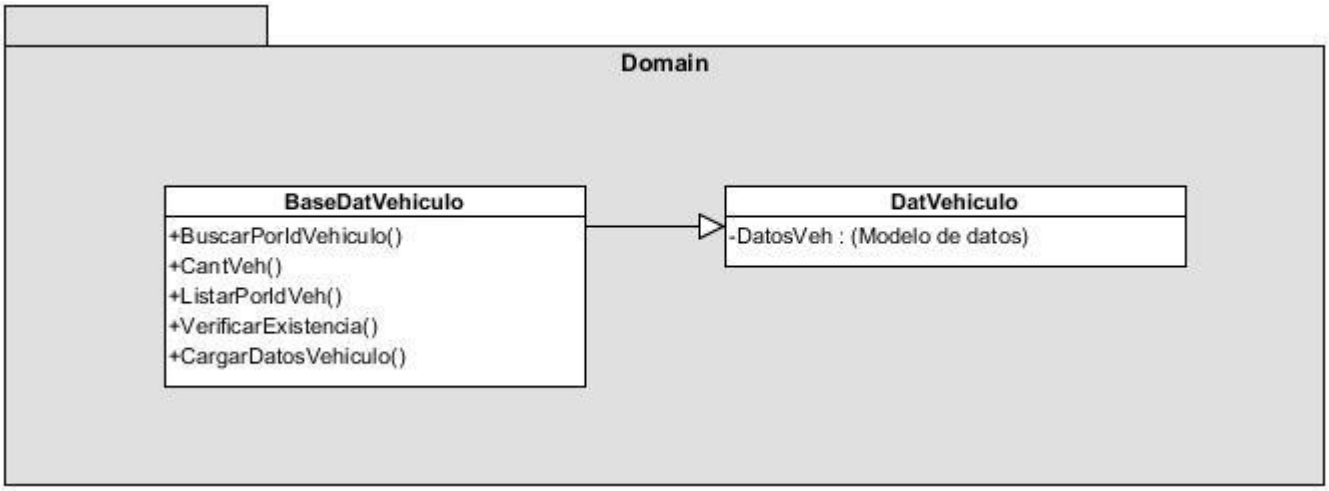

Imagen 7 Mecanismo de diseño para las clases del dominio

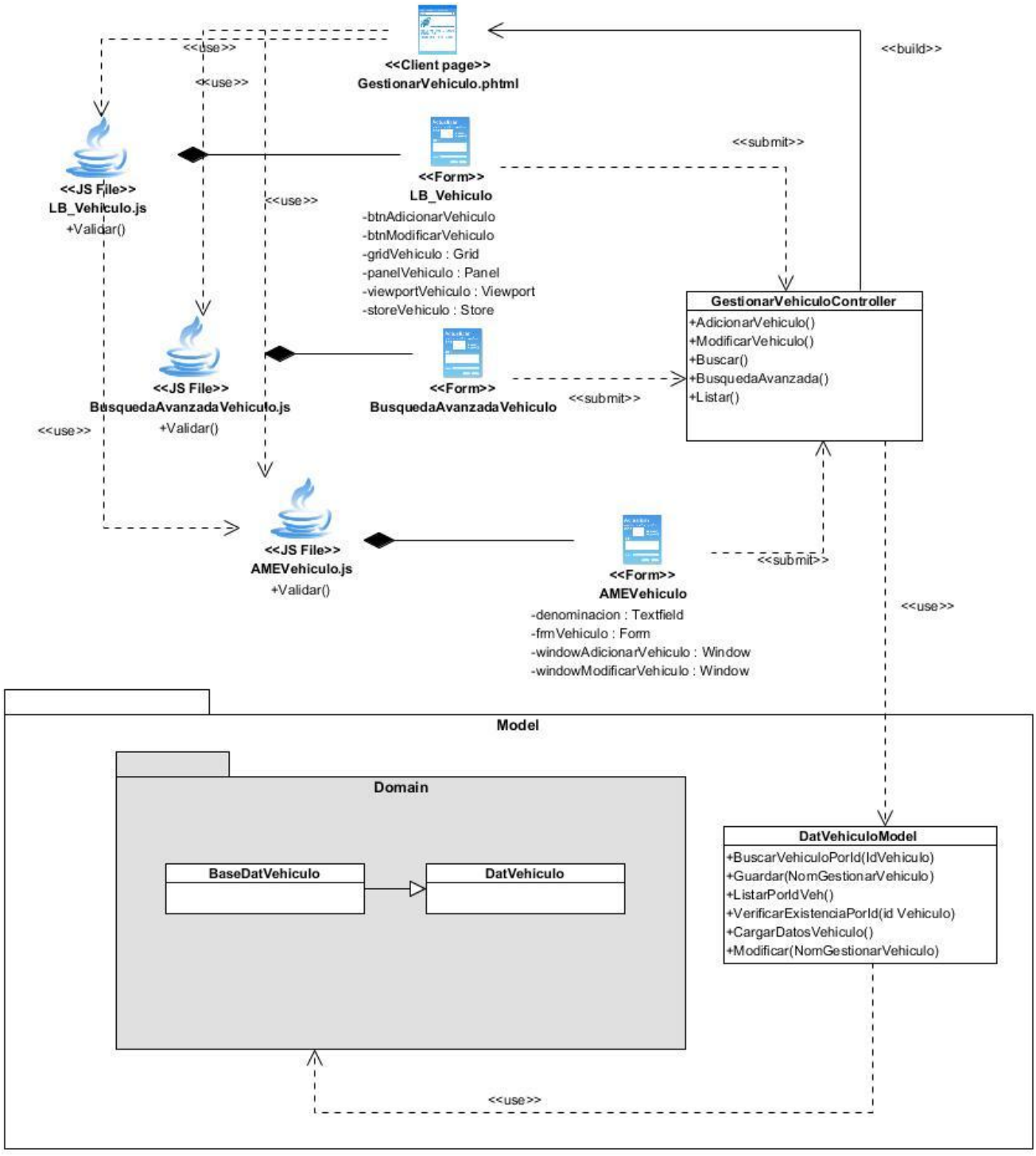
<<Client page>>
GestionarVehiculo.phtml
<<use>>
<<use>>
<<build>>
<<JS File>>
LB_Vehiculo.js
+Validar()
<<Form>>
LB_Vehiculo
-btnAdicionarVehiculo
-btnModificarVehiculo
-gridVehiculo : Grid
-panelVehiculo : Panel
-viewportVehiculo : Viewport
-storeVehiculo : Store
<<submit>>
<<use>>
GestionarVehiculoController
+AdicionarVehiculo()
+ModificarVehiculo()
+Buscar()
+BusquedaAvanzada()
+Listar()
<<JS File>>
BusquedaAvanzadaVehiculo.js
+Validar()
<<Form>>
BusquedaAvanzadaVehiculo
<<submit>>
<<use>>
<<JS File>>
AMEVehiculo.js
+Validar()
<<Form>>
AMEVehiculo
-denominacion : Textfield
-frmVehiculo : Form
-windowAdicionarVehiculo : Window
-windowModificarVehiculo : Window
<<submit>>
<<use>>
Model
Domain
BaseDatVehiculo
DatVehiculo
DatVehiculoModel
+BuscarVehiculoPorId(IdVehiculo)
+Guardar(NomGestionarVehiculo)
+ListarPorIdVeh()
+VerificarExistenciaPorId(id Vehiculo)
+CargarDatosVehiculo()
+Modificar(NomGestionarVehiculo)
<<use>>

Imagen 8 Diagrama de clases de diseño de la agrupación de requisitos Gestionar vehículo.

3.2.3 Descripción de las clases del diseño

A continuación se describe una muestra de las clases controladoras y de las del modelo.

Nombre: GestaccidenteController	
Tipos de clase: Controladora	
Para cada responsabilidad:	
Nombre	imprimiraccidenteAction()
Descripción	Permite mostrar haciendo uso de la librería TCPDF los datos de una unidad en un documento en formato PDF.
Nombre	listarAccidentesAction()
Descripción	Obtiene los datos de todos los accidentes según los criterios de búsqueda pasados. En caso de no entrarle parámetros obtiene todos los accidentes. Estos datos son cargados en la interfaz de usuario.
Nombre	listarConductorAction()
Descripción	Obtiene los conductores de los vehículos a través de un servicio para cargarlos en la interfaz de usuario.
Nombre	guardarAccidenteAction()
Descripción	Permite adicionar o modificar los datos de un accidente en el sistema.
Nombre	cancelarAccidenteAction()
Descripción	Permite cancelar los datos de un accidente en el sistema.
Nombre	listarGravedadAction()
Descripción	Permite cargar los estados de Gravedad en la interfaz de usuario.

Nombre	listarVehiculosAction()
Descripción	Obtiene los datos de todos los vehículos según los criterios de búsqueda pasados. En caso de no entrarle parámetros, obtiene todos los vehículos. Estos datos son cargados en la interfaz de usuario.
Nombre	listarEstadoexpAction()
Descripción	Permite cargar los estados de los Expedientes de los accidentes en la interfaz de usuario.

Tabla 8 Descripción de la clase GestaccidenteController

Nombre: DatAccidenteModel	
Tipos de clase: Modelo	
Para cada responsabilidad:	
Nombre	Guardar($Accidente)
Descripción	A partir de un arreglo pasado con los datos obtenidos de un accidente, realiza su adición al sistema, enviando estos a la base de datos utilizando la clase DatAccidente.
Nombre	Eliminar($Accidente)
Descripción	A partir de un accidente pasado por parámetro, realiza la eliminación de este en el sistema.
Nombre	listarAccidentes($limit,$start)
Descripción	Permite listar los datos de los accidentes que se encuentran en el sistema.

Nombre	buscarPorID($idaccidente)
Descripción	Permite buscar un accidente en el sistema dado su identificador.

Tabla 9 Descripción de la clase DatAccidenteModel

3.2.4 Diagramas de secuencia

Un diagrama de secuencia es un tipo de diagrama de interacción que muestra cómo se interrelacionan los procesos y en qué orden y muestra las interacciones de objetos dispuestos en una secuencia de tiempo. Representa los objetos y clases que participan en el escenario y la secuencia de mensajes intercambiados entre los objetos necesarios para llevar a cabo la funcionalidad del escenario. (OMG UML, 2008). En el Anexo 6 se muestra detalladamente el diagrama de secuencia del requisito Adicionar accidente.

3.2.5 Patrones de diseño

Para realizar un diseño más eficiente es conveniente utilizar uno o varios patrones de diseño. Los patrones de diseño describen un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software, y entregan una buena solución ya probada. Esto ayuda a diseñar correctamente en menos tiempo, a construir problemas reutilizables y extensibles, facilita la documentación, y facilita la comunicación entre los miembros del equipo de desarrollo (Larman, 1999). En este caso ocupa, estos fueron los patrones de diseño aplicados:

Modelo-Vista-Controlador

Es un patrón de diseño que plantea la separación de diferentes clases en dependencia de la función que realizan con el propósito de que sea posible manejar dinámicamente la forma en que se procesan solicitudes y cómo se mostrarán los resultados al usuario final (POSA 01, 2000). En fin se puede decir que separa la lógica de negocio de la presentación o interfaz.

- **Modelo:** El Modelo se encuentra dividido en los paquetes *bussines*, el cual encierra las clases donde se realiza la lógica del negocio, y *domain*, donde se encuentran las clases del dominio, las que se encargan de leer y/o escribir datos en las tablas de la base de datos.

- **Vista:** Maneja la visualización de la información. Le brinda al usuario la posibilidad de interactuar con el Controlador mediante los mensajes de eventos y le permite visualizar las respuestas a sus peticiones. Se encarga de presentar la interfaz al usuario.
- **Controlador:** Representa el gestor de eventos, el cual está compuesto por las clases controladoras. Este atiende los pedidos que llegan desde la Vista accediendo al paquete *models*.

Dentro de los patrones utilizados en la aplicación, se encuentran los GRASP[19] son patrones generales de software para asignación de responsabilidades, aunque se considera que más que son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. (Larman, 2005), los cuales describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. De ellos fueron utilizados:

- **Experto:** Se evidencia cuando las clases controladoras delegan la responsabilidad de ejecutar determinadas acciones a las clases del modelo cuya información es más adecuada para la resolución del problema en cuestión.
- **Creador:** Es adaptable a las clases del paquete *Domain*, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.
- **Controlador:** Se pone de manifiesto a través del uso de una clase controladora que es la que administra el flujo de acciones invocadas desde la interfaz de usuario y en cada una de ellas asocia la lógica del negocio al modelo responsable. En el modelo, donde está implementada la lógica del negocio, las clases del modelo pueden instanciar otras clases en dependencia de la información que se requiera.
- **Bajo acoplamiento:** En el Modelo de Datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

[19] *object-oriented design **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns*, GRASP por su acrónimo en inglés

- **Alta cohesión:** Su empleo está dado en que fueron asignadas responsabilidades a las clases de forma tal que la cohesión siguiera siendo alta, o sea, cada clase se encargará de realizar solamente las funciones que estén en correspondencia con la responsabilidad que posea.

Durante el diseño de la aplicación se emplearon patrones GOF (Gamma, 1995), específicamente:

- **Cadena de Responsabilidad:** Está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo. De esta manera se distribuyen las responsabilidades entre los diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.
- **Fachada:** La utilización de este patrón está reflejada en la creación y empleo de la clase IOC[20], ya que por la necesidad de integración entre los módulos del Sistema de Gestión de Mantenimiento Vehicular, era necesaria la implementación de una clase fachada donde fueran publicados los servicios necesarios por estos, facilitando su interacción.
- ***Singleton*:** Este patrón fue utilizado en las clases del dominio, donde los métodos contenidos en estas se hicieron de forma estática, con lo cual se garantizaba que pudieran ser accedidos sin crear una instancia de las mismas. Por ejemplo: para acceder al método listarTodos() de la clase del domain DatAccidente, desde la clase DatAccidenteModel, del modelo, solo era necesario poner DatAccidente::listarTodos().

3.2.6 Métricas para evaluar el diseño propuesto

Las métricas de software miden cuantitativamente el grado en que un sistema, componente o proceso posee un atributo dado. Estas métricas tienen como propósito entender y mejorar la calidad del producto,

[20] Inversión de Control del inglés *Inversion of Control*.

evaluar la efectividad del proceso y mejorar la calidad del trabajo llevado a cabo al nivel del proyecto (Pressman, 2005).

Una vez realizado el diseño de los procesos Control de vehículo y Seguridad activa y pasiva, se procedió a su validación y para esto, se utilizaron las métricas orientadas a objetos, específicamente las orientadas a clases, porque las medidas y métricas para una clase individual, la jerarquía de clases y las colaboraciones de estas, permiten medir la calidad del diseño propuesto (Pressman, 2005).

Métricas propuestas por Lorenz y Kidd

En su libro sobre las métricas orientadas a objetos, separan las métricas basadas en clases en cuatro amplias categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño se centran en el recuento de atributos y operaciones para cada clase individual y los valores promedio para el sistema orientado a objetos como un todo (Pressman, 2005).

Las métricas empleadas para evaluar el diseño realizado fueron la métrica **Tamaño de clase (TC)** y la métrica **Relaciones entre clases (RC),** ya que se ajustan para evaluar al diseño realizado, siendo las mismas de fácil utilización.

Estas métricas están diseñadas para evaluar los siguientes atributos de calidad:

- **Responsabilidad.** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación.** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización.** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento.** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento.** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en la planificación del proyecto.
- **Cantidad de pruebas.** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto (componente, módulo, clase, conjunto de clases) diseñado.

La métrica empleada fue **Tamaño de clase (TC)**, para aplicarla se tuvo en cuenta la cantidad de procedimientos que tenían las clases identificadas como parte de la propuesta de solución, luego se realizó el cálculo del promedio de los procedimientos y mediante un criterio se obtuvo la categoría (baja, media, alta) para la Responsabilidad, Complejidad y Reutilización.

Clasificación	Valores de los umbrales
Pequeño	<=Promedio de operaciones(PO)
Medio	>PO y <=2*PO
Grande	>2*PO

Tabla 10 Umbrales para la TC

No	Nombre	Cantidad de atributos	Cantidad de operaciones	Tamaño
1	GestautorizoController	0	9	Pequeño
2	GestvehiculoController	0	32	Grande
3	GestaccidenteController	0	17	Medio
4	DatVehiculoModel	0	15	Medio
5	DatAutorizoModel	0	5	Pequeño
6	DatAccidenteModel	0	8	Pequeño

Tabla 11 Tamaño de las clases según sus atributos y operaciones

Clasificación	Cantidad de clases	Responsabilidad de las clases	Complejidad de implementación de las clases	Reutilización
Pequeño	3	Baja	Baja	Alta
Medio	2	Medio	Medio	Medio
Grande	1	Alta	Alta	Baja

Tabla 12 Cantidad de clases por clasificación

Cantidad de clases	Cantidad de clases pequeñas	Cantidad de clases grandes	Cantidad de clases medias	Promedio de atributos	Promedio de operaciones
6	3	1	2	0	11

Tabla 13 Resultado general de la métrica

Se hace un análisis de los resultados obtenidos en el empleo de esta técnica, se concluye que el diseño realizado es simple y tiene una calidad aceptable, pues la mayoría de las clases que fueron analizadas se encuentran dentro de la categoría de pequeñas demostrándose en los pequeños valores de TC alcanzados que la implementación de forma general es sencilla, se disminuye en gran medida la responsabilidad de las clases lo que significa que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

3.3 Modelo de datos

Un modelo de datos es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. El cual puede ser usado como una herramienta para especificar los tipos de datos y la organización de los mismos. Además para la manipulación de consultas y datos, así mismo es el elemento clave en el diseño de la arquitectura de un manejador de BD (Ullman 2006).

3.4 Implementación

3.4.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, contienen todos los componentes definidos en el subsistema, así como las distintas dependencias existentes entre ellos. Se realizan con el objetivo de obtener una vista general del sistema a partir de las dependencias e integraciones de los componentes.

A continuación se representa el diagrama general de componentes del Sistema Control de Flota y Mantenimiento:

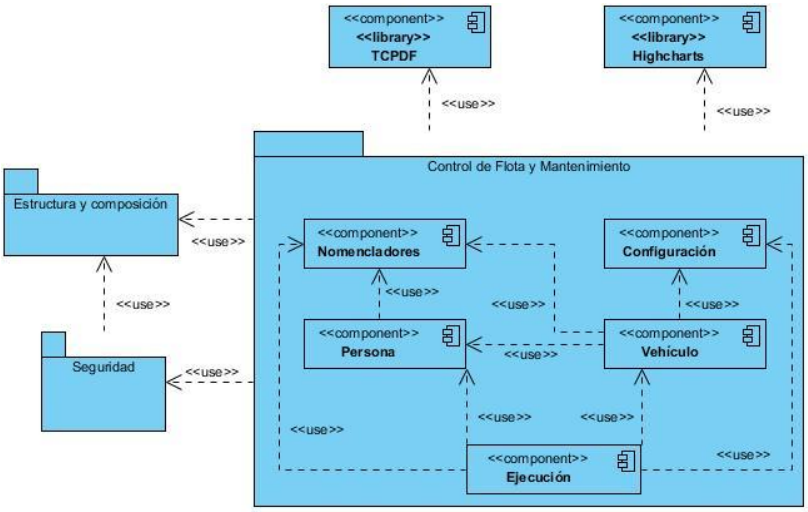

Imagen 9 Diagrama general de componentes.

Las funcionalidades identificadas como parte de la propuesta de solución están incluidas dentro del componente Vehículo.

3.4.2 Modelo de despliegue

Se definirá una arquitectura cliente-servidor detallada de la siguiente manera. Ver figura 10

- **Cliente**: Computador, la aplicación se ejecuta a través de un navegador internet instalado, en este caso se debe usar el Mozilla Firefox sobre cualquier sistema operativo (se sugiere GNU/Linux, en específico la distribución de Ubuntu para estaciones donde se conectarán dispositivos externos: Impresoras).

- **Servidor Web:** Radica la lógica de negocio de la aplicación. Servidor Web Apache2. Utilizando el lenguaje PHP.
- **Servidor de Base de datos:** Servidor de Datos PostgreSQL 8.3
- **Impresora:** Está conectada al computador cliente para imprimir los reportes generados.

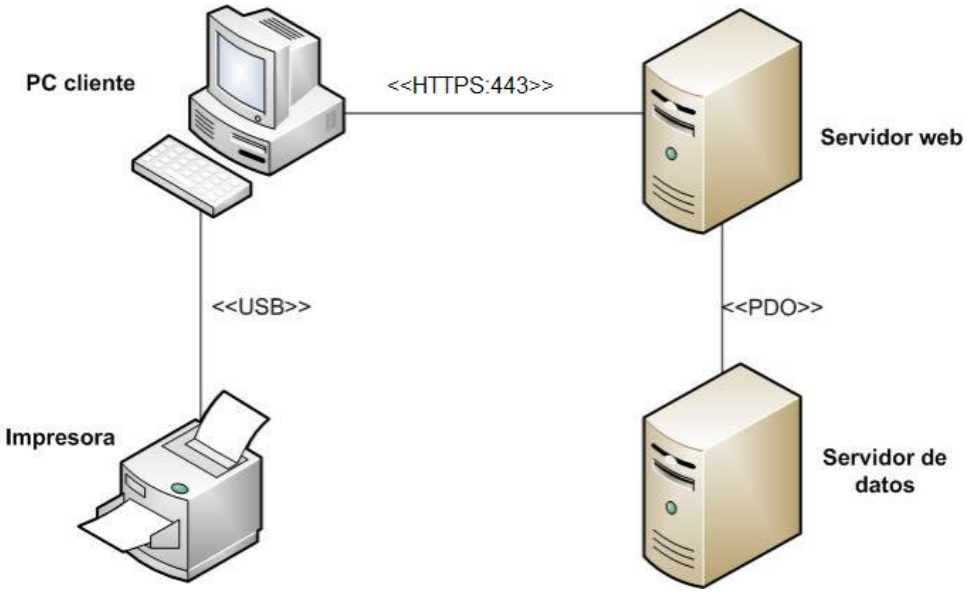

Imagen 10 Modelo de Despliegue

3.4.3 Estándares de codificación

Un estándar de codificación es una regla que se sigue para la escritura del código fuente. De tal manera que a otros programadores se les facilite entender el código (como lo son identificar las variables, las funciones o los métodos). Asegura que los programadores trabajen de forma similar siendo su código entendible por cualquier desarrollador, facilitando también un futuro mantenimiento del sistema.

- **Nomenclatura de las clases según el tipo:**
  - Las clases controladoras que se encuentran dentro de la carpeta "*controller"* comenzarán con la primera letra en mayúsculas y el resto en minúsculas, al final se le añade "Controller". Ej: GestvehiculoController.
  - Las clases modelos que se encuentran dentro de la carpeta "bussines" comenzarán con la primera letra en mayúsculas y el resto en minúsculas, al final se le añade "Model". Ej: DatVehiculoModel.

- Las clases dominios que se encuentran dentro de la carpeta “domain”. Las clases dominios son generadas mediante un mapeo a la base de datos que realiza una herramienta desarrollada por el CEIGE y se llaman igual que la tabla de la base de datos. Ej: DatVehiculo.
- Las clase bases que se encuentran dentro de la carpeta “generated”. Las clases bases son generadas mediante un mapeo a la base de datos que realiza la herramienta Doctrine Generator 3.0 y delante del nombre de la clase se le añade “Base”. Ejemplo: BaseDatVehiculo.
- El nombre de las clases de la vista comenzará con la primera letra en mayúscula y el resto en minúscula, en caso de que este sea compuesto, se utilizará la notación PascalCasing. A este nombre se le agregaran al principio, las letras en mayúscula de las funciones que se realizarán a través de estas clases. Ej: LBVehiculo (listar y buscar), AMVehiculo (adicionar o asociar, y modificar).

- **Nomenclatura de los métodos o funciones:** El nombre de los métodos o funciones de una clase comenzará en minúscula, en caso de que sea compuesto se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo con mayúscula. En caso de que sea una función de una clase controladora se le agregará al final la palabra “Action”. Ejemplo: listarVehiculoAction, adicionarVehiculoAction.
- **Nomenclatura de los atributos:** Los atributos comenzarán con minúsculas, si el nombre se compone de varias palabras excepto la primera las demás comenzarán con mayúsculas (Ejemplo: cantidadMinina).
- **Nomenclatura de las variables:** Las variables serán nombradas comenzando en minúscula, en caso de que el nombre de estas sea compuesto, se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo en mayúscula. En la capa de la Vista, las variables que contengan componentes que formen parte de la interfaz, se les colocará delante un sufijo que indique el tipo de componente que contiene. Ejemplo: btnAdicionar, storeVehiculo.

3.4.4 Publicación de servicios entre componentes

Varios componentes del sistema necesitan datos manejados por otros componentes por lo que estos publican los métodos que necesitan el resto. Estas funcionalidades son implementadas en las clases Services y de forma tal que dichos métodos sean lo más completos posibles. Dichos servicios fueron

implementados de modo que reciban muchos o ningún parámetro, siempre que fuera posible, disminuyendo con esto la cantidad de servicios a publicar.

A continuación se muestran los servicios que son brindados por el componente Vehículo y que son utilizados en el desarrollo del módulo Vehículo:

Servicios	Componentes que lo utilizan	Descripción
Listar_PorTipoCombustible	Vehículo	Muestra un listado de todos los tipos de combustible que se encuentran registrados en el sistema.
Listar_tipovehiculo	Vehículo	Muestra un listado de todos los tipos de vehículos que se encuentran registrados en el sistema.
Listar_GrupoVehiculo	Vehículo	Muestra un listado de todos los grupos de vehículos que se encuentran registrados en el sistema.
Buscar_Grupo	Vehículo	Muestra un grupo de vehículos específico registrado en el sistema.
Buscar_UnidadMedida	Vehículo	Muestra una unidad de medida registrada en el sistema.
Documentos_Grupo	Vehículo	Muestra los documentos técnicos de un grupo de vehículos registrado en el sistema.
Buscar_Trabajador	Vehículo	Muestra los datos de un trabajador específico registrado en el sistema.

Tabla 14 Servicios que brinda Vehículo

3.5 Pruebas de software

3.5.1 Pruebas de Caja Blanca

La prueba de Caja Blanca, o pruebas de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Para la solución

desarrollada la prueba de Caja Blanca aplicada fue la del camino básico. Esta técnica de prueba permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2005).

Para utilizar esta técnica primeramente es necesario realizar el cálculo de la complejidad ciclomática del código fuente que vaya a ser analizado, para lo cual se deben enumerar las sentencias de código del mismo y elaborar el grafo de flujo de la funcionalidad.

A continuación se muestra el grafo de flujo asociado a la funcionalidad:

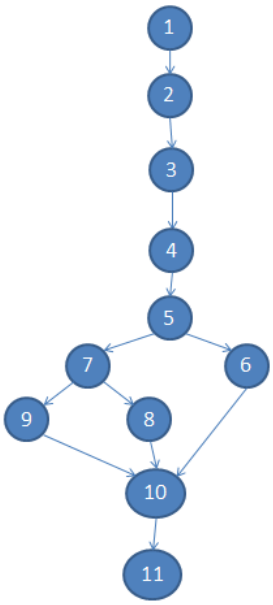

Imagen 11 Flujo asociado del método buscarAutorizoParqueoAction()

Cálculo de la complejidad ciclomática a partir de un segmento de código

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y ofrece un límite superior para el número de pruebas que se deben realizar para asegurar que sea ejecutada cada sentencia al menos una vez (Pressman, 2005). Al calcular la complejidad del método buscarAutorizoParqueoAction() fueron utilizadas

las tres formas posibles con el objetivo de verificar que el cálculo se ha realizado correctamente y los resultados obtenidos en cada una era el mismo. Las fórmulas para realizar dicho cálculo son:

1. V (G) = (A - N) + 2, V (G)= (12 -11)+2, V (G)= 3

Siendo A la cantidad total de aristas del grafo y N la cantidad de nodos.

2. V (G) = P +1, V (G) = 2+1, V (G) = 3

Siendo P la cantidad de nodos predicado (son aquellos de los cuales parten dos o más aristas).

3. V (G) = R, V (G) = 3

Siendo **R** la cantidad de regiones que posee el grafo.

Cada una de las fórmulas **V(G)** representan el valor del cálculo. A partir de los resultados obtenidos en cada uno, se determina que la complejidad ciclomática del código analizado es 3, que a su vez es el número de caminos posibles a circular el flujo y el límite superior de casos de prueba que se le pueden aplicar a dicho código.

Número	Caminos básicos
1	1-2-3-4-5-6-10-11
2	1-2-3-4-5-7-9-10-11
3	1-2-3-4-5-7-8-10-11

Tabla 15 Caminos básicos de flujo.

Para cada uno de los caminos obtenidos se realiza un caso de prueba. Los casos de prueba realizados son los siguientes:

Caso de prueba para el camino básico # 1	
Camino	1-2-3-4-5-6-10-11
Descripción	Los datos de entrada cumplirán con los siguientes requisitos: La variable $matricula está vacía y la variable $noorden contiene datos.
Entrada	$matricula = null, $noorden = 1234
Resultados esperados	Se espera que muestre todos los autorizos de parqueo cuyo número de orden sea 1234.
Resultados obtenidos:	Satisfactorio.

Tabla 16 Caso de prueba para el camino básico 1

Caso de prueba para el camino básico # 2	
Camino	1-2-3-4-5-7-9-10-11
Descripción	Los datos de entrada cumplirán con los siguientes requisitos: La variable $matricula contiene datos y la variable $noorden está vacía.
Entrada	$matricula = “HVA123”, $noorden = null;
Resultados esperados	Se espera que muestre todos los autorizos de parqueo cuya matrícula sea HVA123.
Resultados obtenidos:	Satisfactorio.

Tabla 17 Caso de prueba para el camino básico 2

Caso de prueba para el camino básico # 3	
Camino	1-2-3-4-5-7-8-10-11
Descripción	Los datos de entrada cumplirán con los siguientes requisitos: La variable $matricula está vacía y la variable $noorden está vacía.
Entrada	$matricula = null, $noorden = null;
Resultados esperados	Se espera que muestre todos los autorizos de parqueo.
Resultados obtenidos:	Satisfactorio.

Tabla 18 Caso de prueba para el camino básico 3

Con la aplicación de los casos de prueba expuestos anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, probándose que cada sentencia es ejecutada al menos una vez con los parámetros de entrada y las salidas esperadas. Con las pruebas de caja blanca se examinó el estado de la aplicación en varios puntos de la misma determinando que el estado real coincidía con el esperado.

3.5.2 Pruebas de Caja Negra

Estas pruebas, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, estas pruebas permiten al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Las pruebas de cajas negra no son una alternativa a las técnicas de prueba de caja blanca, se trata de un

enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca (Pressman, 2005).

En este estudio, se realizó la Técnica de la Partición de Equivalencia, que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Esta técnica es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software y descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. (Juristo, 2005)

Para cada uno de los requisitos funcionales fueron definidos casos de pruebas, los cuales son los siguientes:

- Se debe identificar y autenticar ante el sistema, además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema **Mantenimiento/Vehículos/Accidentes**.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar accidente.	El sistema debe adicionar accidentes en el sistema.	EP 1.1: Adicionar accidentes introduciendo datos válidos	– Se introducen los datos del accidente correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.2: Adicionar accidentes introduciendo datos válidos presionando el botón **Aplicar**.	– Se introducen los datos del accidente correctamente. – Se presiona el botón **Aplicar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.3: Adicionar accidentes introduciendo datos inválidos.	– Se introducen los datos inválidos del accidente – Se presiona el botón **Aceptar**. – Se muestra un mensaje de error. – Se presiona el botón **Aceptar**.

		EP 1.4: Adicionar accidentes dejando campos vacíos.	Se introducen los datos accidente dejando algún campo en blanco. Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error. – Se presiona el botón **Aceptar**.
		EP 1.5: Cancelar.	Se introducen o no los datos del accidente. Se presiona el botón **Cancelar**.

Tabla 19 Descripción del caso de prueba para el requisito Adicionar accidente

Para más información sobre el caso de prueba para el requisito Adicionar accidente, consultar el documento CIG-CFM-N-MTTO-i5182.doc. Los casos de prueba elaborados para los restantes requisitos funcionales se encuentran en los documentos entregables referidos a los casos de prueba.

Con la aplicación de las Pruebas de Caja Negra se demostró que cada una de las funciones de la aplicación es totalmente operativa, se obtuvieron los resultados esperados como respuesta del conjunto de condiciones de entrada, comprobándose el correcto funcionamiento de las funcionalidades del módulo Control de vehículo según su especificación.

3.6 Conclusiones parciales

Durante el desarrollo del capítulo fueron expuestos los artefactos generados del diseño del módulo para la gestión de los procesos Control de vehículo y Seguridad activa y pasiva de la Base de Transporte de la UCI, así como las métricas se emplearon para su validación, las que ofrecieron como resultado que el diseño realizado es simple y con una calidad aceptable, permitiendo dar paso a la etapa de implementación. Además se presentan las pruebas estructurales y funcionales que fueron aplicadas, que dieron como resultado que se poseía un correcto funcionamiento, contaba con las condiciones de calidad necesarias y satisfacía las necesidades plasmadas por el cliente.

CONCLUSIONES

El desarrollo de los módulos Control de vehículos, Control de accidentes, Planificación del día de la técnica, Autorizo de parqueo y Seguridad activa y pasiva contribuyó a mejorar la gestión de los procesos de Control de vehículo y Seguridad activa y pasiva de la Dirección de Transporte de la UCI pues:

- Se fundamentó la investigación a través de la creación del Marco Teórico, donde se definieron las herramientas, tecnologías y lenguajes a utilizar.
- Con el modelado del negocio de los procesos Control de vehículo y Seguridad activa y pasiva fueron generados una serie de artefactos que crearon las condiciones para la captura de los requerimientos funcionales de estos procesos.
- El análisis de los artefactos generados en el diseño permitió comprobar que no existía ambigüedad en las especificaciones de los requisitos funcionales. Se describieron los estándares de codificación utilizados en la implementación de la propuesta solución, así como los servicios que se brindan y se consumen por el componente Vehículo.
- Se realizó la implementación del Seguridad activa y pasiva, lográndose una versión estable, y de la mayor parte del módulo Control de vehículo, así como la aplicación de pruebas de software para su validación, dándole solución a la mayor parte de las necesidades de la Dirección de Transporte de la UCI.

RECOMENDACIONES

Se recomienda que en el desarrollo de futuros sistemas de control de flotas sea utilizado el presente trabajo como referencia en cuanto a los procesos relacionados con Control de vehículos y la Seguridad activa y pasiva.

BIBLIOGRAFÍA

Bennett, Sean (February 2, 2010). Heavy Duty Truck Systems (5 ed.). Delmar Cengage Learning. pp. 116–117

Comsun Car System. [En línea] 2009. http://www.comsun1.com/index.html.

SIMPYC: Un planificador de Rutas de Vehículos. PROYECTO EUROPEO ESPRIT. [En línea] http://www.bcncosiver.com/caste/index.htm.

G.I.M. [En línea] 2009- Gestión Integrada de Mantenimiento. Parets del Vallès Barcelona-España: s.n

Transoft, 2009. Siscompa.net, sistema cubano para la gestión de flotas. Ciudad de la Habana, Cuba.

SGestMan 2010. Un Sistema Informático para la Gestión del Mantenimiento. Manual de usuarios. Ciudad de la Habana, Cuba : s.n.,

Offimant, 2011. Silvente Trujillo, Yinelis, Toledo, Raciel y Delgado Hernández, Raimy. INGENIERIA & INFORMATICA OffiMant. Ciudad de la Habana, Cuba : s.n.

Equipo de Producción. Modelo de Desarrollo Orientado a componentes del proyecto ERP-CUBA.

POSA 01, 2000 (Pattern-Oriented Software Architecture, Volume 01), by Douglas Schmidt, Michael Stal, Hans Rohnert and Frank Buschmann ISBN: 0471606952. John Wiley & Sons pp. 17-18, http://Fakademik.del.ac.id/ebooks/SE/Pattern-Oriented%20Software%20Architecture%2C%20Volume%201.pdf

Garrett, Jesse James. [En línea] Ajax: A New Approach to Web Applications, Febrero 18, 2005, http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

Sencha Inc. [En línea] Sencha Ext JS JavaScript Framework for Rich Apps in Every Browser http://www.sencha.com/products/extjs Enero 2013.

Zend Technologies Ltd. [En línea.] Zend Framework Reference. http://framework.zend.com/manual/1.12/en/introduction.html enero de 2013.

Doctrine, Concepto de [En línea] http://www.doctrine-project.org/docs/orm/2.1/en/reference/introduction.html. 2013.

Morera, René Hernandez. junio del 2011. Componente para la configuración visual de los servicios de integración en el marco de trabajo Sauxe. La Habana : s.n

Glosario

Centro de Informatización de la Gestión de Entidades. 2011. La Habana : s.n., 2011.

BPMN Business Process Modelating Notation, 2013 [En línea] http://www.bpmn.org/

UML Unified Modeling Language, 2013 [En línea] http://www.uml.org/.

PHP: Hypertext Preprocessor, 2013 [En línea] http://php.net/manual/es/faq.general.php

Pressman, Roger S. 2006. Ingeniería del Software Un enfoque práctico.

The Apache Software Fundation, 2013 [En línea] http://httpd.apache.org/

PostgreSQL 8.3, 2013. [En línea] http://www.postgresql.org/docs/8.3/static/release-8-3.html

Firefox, 2013 [En línea] http://support.mozilla.org/es/products/firefox/download-and-install

Eclipse Galileo 3.5, 2013 [En línea] http://help.eclipse.org/galileo/index.jsp

Sommerville. 2005. 2005.

Noguera, Manuel 2011 Introducción al Modelado de Procesos de Negocios. [En línea] http://www.ugr.es/~mnoguera/collaborative_systems-business_processes_10-11.pdf

Workflow Management Coalition, 2013 [En línea] http://www.wfmc.org/Download-document/TC00-1003-The-Workflow-Reference-Model.html.

Yatco, Mei C., 1999 "Joint Application Design/Development". University of Missouri-St. Louis.

Sandoval Carvajal, Maria Marta y García Vargas, Maria Adilia, 2006 LA TRAZABILIDAD EN EL PROCESO DE REQUERIMIENTOS DE SOFTWARE. [En línea] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf

Cabrera Pereira., Leisniel Ignacio y Hernández Gómez, Maylin. 2009. Análisis y Diseño del Módulo de Cobros y Pagos del sistema integral de gestión CEDRUX. Ciudad de la Habana : s.n., 2009.

Visconti, Marcelo y Astudillo, Hernán. Fundamento de Ingeniería de Software, 2011. Magma Soft. [En línea]. http://www.magma.com.ni/~jorge/upoli_uml/refs/Introducion_UML.pdf

OMG UML, 2008. OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, p. 485. [En línea]. http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF

Larman, Craig. 1999. *UML y Patrones.* Mexico : s.n., 1999. 970-17-0261-1.

Eric Gamma, Richard Helm, Ralph Johnson y John Vlissides, **Design Patterns.**

Elements of Reusable Software. Addison-Wesley, Reading, MA, 1995 ISBN-13: 978-0201634983

Juristo, Natalia, 2005. Verification and validation. http://eva.uci.cu/file.php/158/