Ministerio de Educación Superior
Universidad Oscar Lucero Moya de Holguín
Facultad de Informática-Matemática.

SOFTWARE PARA DETERMINAR LA CAPACIDAD DE CARGA Y RÉGIMEN DE MARCHA DEL CONJUNTO TRACTIVO

Trabajo de diploma para optar por el titulo de:
Ingeniero Informático.

Autor: Leonar Alberto López Escobar.

Tutores: MsC. Ing. Vladimir Álvarez Sánchez.
Ing. Arquímedes R. Leyva Téllez.

Consultante: Lic. Isidro M. Corría Rodríguez.

Holguín, 2010

"Cuando se nos otorga la enseñanza se debe percibir como un valioso regalo y no como una dura tarea, aquí está la diferencia de lo trascendente."

Albert Einstein

Agradecimientos

Les agradezco a todos los que me ofrecieron ayuda incondicionalmente, especialmente a Vladimir, Pedrito, JK, Scull, Becerra, Brea, Robert, Ari.

A mi familia por el apoyo y la confianza que depositaron en mí.

Dedicatoria

A la memoria de mi mamá.

A mi papá, mi hermano y familia más allegada.

Resumen

La organización TRANSMINAZ tiene como objeto empresarial el de encargarse de toda la actividad de transportación en el MINAZ y durante el período de zafra es la responsable de todo el traslado de la cosecha cañera hacia la industria. Ellos tienen como principio la explotación de su parque al máximo de eficiencia. Esta organización no dispone de una herramienta que le permita determinar sobre bases científicas la mejor conformación del “Conjunto Tractivo” de modo que pueda maximizar la eficiencia de la actividad de transporte de la cosecha cañera. Con tal propósito el MSc. Ing. Vladimir Álvarez Sánchez desarrolla una teoría científica mediante la cual se puede determinar un valor racional de la capacidad de carga de los remolques cañeros y el régimen de marcha del “Conjunto Tractivo”, de modo que durante su explotación los gastos de la actividad de transporte de la cosecha cañera tiendan a un mínimo.

El presente trabajo de tesis propone el desarrollo de una herramienta informática, (sobre la base de la teoría científica antes mencionada) que permite la implementación de un sistema, capaz de gestionar eficientemente la mejor variante de conformación y explotación de un “Conjunto Tractivo”, con un alto grado de confiabilidad, integridad y eficiencia.

El documento hace un recorrido por todo el proceso ingenieril en torno al proyecto y toca puntos cardinales, como el estudio teórico, las fases detalladas de la metodología de desarrollo empleada, expresada a través de sus artefactos con lenguaje UML y el estudio de sostenibilidad realizado.

Abstract

The organization TRANSMINAZ has as its principal purpose to provide support all of the activity associated with transportation in MINAZ and during the sugar cane harvesting period is responsible for transporting the harvest from the cane field to the industry. One of the main objectives is the maximum utilization of their park making it more efficient.

This organization does not have a tool that permits the scientific determination of the best conformation of Traction Group so that it may maximize the efficiency of transporting the cane harvest. To resolve this problem Vladimir Álvarez Sánchez Msc. Ing. developed a scientific intervening theory which can determine a rational value of the cargo capacity of the sugar cane trailers and parade's regimen of the Traction Group, in a way that the expenses associated with the transportation the cane harvest is as low as possible.

The present work proposes the development of a computer software system, (based on the scientific theory mentioned above,) that allows the implementation of a system, capable of obtaining efficiently the best variant of conformation and exploitation of the Traction Group with a height, reliability grade, integrity and efficiency.

The document takes a journey throughout the projects engineering process touching cardinal points, like the theoretic study, a detailed analysis of the software development methodology used, expressed through the use of the UML language and a study of its sustainability was also carried out.

Índice

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN.

En el V Congreso del Partido Comunista de Cuba (PCC) en su resolución económica se plantea: *"La eficiencia es por tanto, el objetivo central de la política económica pues constituye una de las mayores potencialidades con que cuenta el país. Hacer un mejor uso de los recursos, elevar la productividad del trabajo, alcanzar resultados óptimos con menos costos, tendrán un efecto positivo en nuestro balance financiero, facilitando la participación en el comercio internacional y el acceso a los mercados de capital e inversiones"* [RESOLUCION].

En las Bases Generales del Perfeccionamiento Empresarial [GOFICIAL, 1998] se plantea como objetivo central incrementar al máximo la eficiencia y competitividad de la empresa estatal y que deberá conducir a que la innovación y la actividad de gestión tecnológica a él asociadas, se conviertan en elementos esenciales para la dirección de las empresas.

En el MINAZ la organización empresarial "TRANSMINAZ", tiene como objeto social encargarse de toda la transportación de la cosecha cañera, para ellos es prioridad que los "Conjuntos Tractivos" se exploten al máximo de eficiencia. Aunque la teoría planteada por Álvarez Sánchez da solución a esta necesidad, no se dispone de una herramienta informática que permita implementar de modo práctico la misma, es decir, no se dispone de un software con el cual calcular el valor óptimo de capacidad de carga de los remolques cañeros y régimen de marcha del Conjunto Tractivo.

Se denomina "Conjunto Tractivo" a la combinación de un camión o tractor con uno o varios remolques (la primera se denomina rodo-tren y la segunda tractor-tren). El tractor se puede evaluar como un caso particular de un camión cuya capacidad de carga es cero.

El modelo matemático formulado por el MSc. Ing. Vladimir Álvarez Sánchez, es un modelo de programación no lineal, de varias variables (la capacidad de carga de los remolques cañeros (CC) y las velocidades del viaje de ida (Vi) y de vuelta (Vv), que definen el régimen de marcha del "Conjunto Tractivo"), en él se plantean la dependencia de los gastos de la explotación de la actividad de transporte de la cosecha cañera de las tres variables antes descritas. Y al

determinar el mínimo de la función objetivo, el valor que ellas adquieren describe la capacidad de carga y el régimen de marcha que permite que se obtenga los gastos mínimos de la actividad de transporte de la cosecha cañera.

Actualmente para determinar estos valores se procesan todos los datos con la herramienta Solver del paquete de Microsoft Office (Excel) lo cual resulta un proceso muy engorroso, debido a que todos los datos característicos de cada elemento que conforma un caso de estudio deben ser introducidos en todas las variantes de combinaciones posibles por cada caso a evaluar.

Haciendo un análisis de lo antes expuesto surge el siguiente **problema científico:**

¿Cómo lograr una aplicación que permita de modo eficaz y sencillo la selección de los valores óptimos del régimen de marcha y capacidad de carga del "Conjunto Tractivo"?

Como **objeto de investigación** en el cual se enmarca la situación actual se tiene: Teorías científicas existentes en cuanto a la determinación de la eficiencia del transporte en la cosecha cañera.

El **campo de acción** en el cual se centra la investigación es: Informatización de la teoría científica desarrollada para analizar la eficiencia de la actividad de transporte de la cosecha cañera.

El **objetivo principal** que se persigue es desarrollar una herramienta informática que permita obtener valores óptimos de capacidad de carga de los remolques cañeros y del régimen de marcha del "Conjunto Tractivo", para que se obtengan los mínimos gastos por concepto de eficiencia de la actividad de transporte de la cosecha cañera.

Para guiar la investigación se plantean las siguientes **preguntas científicas:**

1. ¿Cuáles son los fundamentos teóricos y el desarrollo obtenido en cuanto a la determinación de la eficiencia del transporte en la cosecha cañera?
2. ¿Cuáles son las tendencias y tecnologías existentes para el desarrollo del software que permita determinar el mínimo valor de la función objetivo del modelo matemático?

3. ¿Cómo realizar el análisis y diseño del software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo?
4. ¿Será factible y sostenible el producto informático dado como solución propuesta?
5. ¿Cómo implementar el software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo?

Las **tareas** que dan respuesta a las preguntas científicas y que permitirán el cumplimento del objetivo planteado agrupadas con un orden lógico son:

1. Realizar la fundamentación teórica y diagnosticar el desarrollo obtenido en la determinación de la eficiencia del transporte en la cosecha cañera.
2. Analizar y valorar la tecnología a emplear para desarrollar la aplicación de determinación de la capacidad de carga y régimen de marcha del conjunto tractivo.
3. Hacer el análisis y diseño de la herramienta informática propuesta como solución.
4. Valorar la factibilidad y sostenibilidad del producto informático según las dimensiones administrativa, socio-humanista, ambiental y tecnológico.
5. Implementar el software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo.

Para realizar las tareas se utilizaran los siguientes **métodos de Investigación:**

Métodos teóricos:

- **Análisis y síntesis:** para procesar la información teórica y empírica que caracterizan el problema, el objeto y campo de estudio, así como en la elaboración de la presente introducción.
- **Histórico – lógico:** para hacer una exposición de las formas de selección del valor de la capacidad de carga de los remolques cañeros en Cuba y en otras partes del mundo, así como en el estudio de los modelos matemáticos aplicados con este fin.
- **Modelación:** para modelar todo el proceso de ingeniería de la aplicación a desarrollar.

Métodos Empíricos

- **Entrevista:** para obtener información detallada así como un mejor entendimiento sobre cómo es el flujo del proceso de determinación del régimen de marcha y capacidad de carga del conjunto tractivo
- **Revisión de documentos:** para la recopilación de información relacionadas con el problema y las posibles tecnologías a emplear.

La investigación que se presenta consta de dos capítulos, el **Capítulo I** titulado "Fundamentación teórica" en el cual se exponen los aspectos relacionados con el estado de desarrollo en la determinación de la eficiencia de la actividad de transporte en la cosecha cañera; se mencionan algoritmos y métodos para optimizar problemas generales de la programación seleccionándose el de exploración como método a implementar en la solución propuesta; las tecnologías existentes para el desarrollo de aplicaciones. Se hace un análisis de las metodologías para el desarrollo de software. A modo de conclusión del capítulo se eligen las tecnologías apropiadas, así como la metodología para el desarrollo de la solución propuesta. El **Capítulo II** se titula "Descripción de la solución propuesta", en el mismo se detalla cada paso de la ingeniería del desarrollo del software propuesto guiado por ICONIX como metodología de desarrollo y se hace la valoración de sostenibilidad de la solución propuesta (producto informático).

CAPÍTULO I FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace la fundamentación teórica del objeto de investigación de la presente, una descripción de las principales tecnologías actuales de persistencia. Además se mencionan los conceptos básicos de base dato y sus características. También se exponen los distintos Sistemas Gestores de Bases de Datos con sus particularidades. Se hace alusión a las metodologías de desarrollo de software, así como sus ventajas y desventajas. Para concluir el capítulo el autor explica las principales razones por las cuales hace la selección de las herramientas y tecnologías.

1.2 Teorías Científicas sobre determinación de la capacidad de carga.

En estudios precedentes en cuanto al tema de la eficiencia en la actividad de transporte se exponen criterios y formulaciones matemáticas de los factores que influyen en el mismo.

Cardone aborda la organización racional de los medios de transporte cañero en un pelotón cañero, sobre la base de determinar el número de vehículos de carga necesarios, planteando que su número depende de: cantidad de caña por el tiempo del ciclo de transportación entre el producto de la capacidad de carga de los vehículo por el tiempo del turno de trabajo [CARDONE, 1985].

En otros estudios desarrollados por los autores Rodríguez y Márquez, se plantea en sus conclusiones dos aspectos importantes de los que depende la eficiencia de la actividad de transporte, el primero es el valor de la variable capacidad de carga y el segundo es la masa de los remolques [RODRIGUEZ, 1985].

Dominico, Carballo y Martín evaluaron la productividad (t-km), la cual depende directamente de la capacidad de carga real, de la velocidad técnica media y del tiempo del ciclo de viaje [DOMINICO, 1990].

Pedro Rodríguez desarrolla el método de Isocosto empleando la ecuación de los costos de la tonelada kilómetro, en la que muestra que estos son

inversamente proporcional a la capacidad de carga de los remolques [RODRIGUEZ, 1991].

Jróbostov expresa que, para asegurar altos índices de trabajo de un Conjunto Tractivo se debe asegurar la mayor utilización del esfuerzo de tracción, lo cual se determina por la razón entre el esfuerzo de tiro necesario para mover un tren de remolques cañeros y la fuerza de tiro del tractor. El esfuerzo de tiro a su vez depende directamente de la masa total de los remolques cargados, (es decir de su capacidad de carga y de su masa) por el número de ellos que se acoplan, por la resistencia que ofrece el terreno a su desplazamiento [JROBOSTOV, 1997].

Los casos estudiados antes mencionados evidencian la dependencia de la eficiencia de la actividad de transporte de la capacidad de carga, de la masa de los remolques y de las velocidades del viaje.

Iglesias determinó la ecuación que caracteriza la masa que debe poseer un remolque en función de su capacidad de carga, con lo cual simplifica matemáticamente la determinación del valor de la capacidad de carga de un remolque una vez conocido el esfuerzo de tiro del tractor y los valores de resistencia del camino, porque ahora existe una sola variable independiente, que es la capacidad de carga [IGLESIAS79].

Ninguna de las formulaciones estudiadas es capaz de conjugar el análisis de los gastos de la actividad de transporte, la capacidad de carga y el régimen de marcha del conjunto tractivo en un solo modelo matemático, esto se logra por primera vez con la fundamentación planteada en [ÁLVAREZ10] por el MSc. Ing. Vladimir Álvarez Sánchez., donde desarrolla un modelo de múltiples variables (CC, Vi, Vv) de programación no lineal, con restricciones.

1.3 Algoritmos para la programación

Del modelo matemático expuesto por MSc. Ing. Vladimir Álvarez Sánchez se desea conocer el régimen de marcha (velocidad de ida y velocidad de vuelta), así como la capacidad de carga del remolque y la cantidad de remolques que debe tener el conjunto tractivo; todos estos elementos conforman las variables

del modelo de las cuales se desea obtener el valor idóneo que conduzca a la eficiencia del conjunto tractivo [ÁLVAREZ10].

Existen varios algoritmos de optimización para la programación en general de modelos matemáticos propuestos entre los cuales se encuentran el Método de Programación Aproximada (MAP, por sus siglas en inglés), el Gradiente Reducido (GR) con condiciones lineales, el Gradiente Reducido Generalizado (GRG), el Gradiente Reducido Generalizado 2 (GRG2), el Sistema Modular de Programación No-lineal con condiciones lineales (MINOS, por sus siglas en inglés), el de Programación Aproximada en base al Gradiente Conjugado (CGAP, por sus siglas en ingles) entre otros [ESCUDERO, 1978].

Todos estos algoritmos presentan características tales como:

1. No precisan que para su convergencia la función objetivo a optimizar sea convexa, aunque si no lo fuese, el óptimo local encontrado tampoco garantiza que sea el óptimo global.
2. Combinan los métodos de programación lineal (PL) con los de programación sin restricciones (PSR).
3. Utilizan el gradiente reducido y la matriz hessiana reducida de la función objetivo para determinar la dirección de las variables.
4. Tienen su validez utilizándolos en problemas reales.

Además de estos algoritmos existen métodos como el de exploración el cual consiste en iterar las variables del modelo, evaluar la función objetivo para cada iteración y comparar el resultado, este ciclo se repite mientras tanto no se encuentre el mínimo o el máximo de la función objetivo, según el tipo de problema que sea, garantizando la obtención de un óptimo global.

Este método es factible utilizarlo en la optimización de problemas pero tiene aparejado un alto costo computacional como inconveniente, cuando el número de variables de la función objetivo es muy elevado las soluciones a evaluar son muchas, debido a que las soluciones crecen exponencialmente según la cantidad de variables y el rango de valores que puedan tomar las variables.

Teniendo en cuenta que el número de variables que intervienen en el modelo matemático propuesto por Álvarez es pequeño (solo de 4 variables), se

selecciona el método de exploración para implementarlo en el software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo.

1.4 Tecnologías de persistencia

El autor se refiere con tecnologías de persistencia a las librerías, APIs (Applications Programming Interface: Interfaz de Programación de Aplicaciones) y frameworks que se emplean para el proceso de almacenamiento de los objetos o datos.

Un API representa un medio de comunicación entre componentes de software o entre diferentes tecnologías [SUNAPI10]. Es el conjunto de invocaciones a determinadas librerías que brindan sus servicios, es una forma de lograr una abstracción en la programación entre los niveles y/o las capas tanto inferiores como superiores del software. El objetivo principal del API es ofrecer un conjunto de funcionalidades para uso general. Permitiéndole a los desarrolladores que se beneficien de las ventajas del API, para no tener que implementar toda la aplicación desde cero.

Los frameworks en el campo del desarrollo de software son una estructura conceptual y tecnológica de soporte definidos que permiten el desarrollo de otros proyectos así como su organización, típicamente incluyen el soporte de programas, bibliotecas y lenguajes interpretados para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Es un esqueleto sobre el cual varios objetos son integrados para lograr una solución, debido a esto, los frameworks permiten la incorporación de una o varias API.

Ejemplos de estas tecnologías son Hibernate, Toplink, JDBC (Java Database Conectivity: conectividad a bases de datos en java), JDO (Java Data Objects), JPA (Java Persistence API). En los siguientes sub-epígrafes se aborda con un poco de detalle acerca de las mismas.

1.4.1 Hibernate

Hibernate es una tecnología ORM (Object/relational Mapping) para la plataforma java que facilita el mapeo entre una base de datos relacional y el modelo de objetos de una aplicación, a través de archivos XML que permiten

hacer esta relación [GLÖGLS07]. Hibernate se empezó a desarrollar hace algo más de 4 años por Gavin King, siendo hoy día Gavin y Christian Bauer los principales gestores de su desarrollo. Es software libre, distribuido bajo los términos de la licencia GNU LGPL.

Presenta la filosofía de mapear objetos Java "normales", también conocidos en la comunidad como "POJOs" (Plain Old Java Objects), no contempla la posibilidad de automatizar directamente la persistencia de Entity Beans tipo BMP (Bean Managed Persistence) (generar automáticamente este tipo de objetos), aunque aún así es posible combinar Hibernate con este tipo de beans utilizando los conocidos patrones para la delegación de persistencia en POJOs.

El diseño de Hibernate presenta características tales como: se pueden utilizar los objetos Java definidos por el usuario tal y como son, es decir, no utiliza técnicas como generación de código a partir de los descriptores de modelos de datos o manipulación de bytecodes en tiempo de compilación (técnica conocida por su amplio uso en JDO), ni obliga a implementar interfaces determinados, ni heredar de una superclase. Utiliza en vez de ello el mecanismo de reflexión de Java, característica que le permite un modelado iterativo, fluido y natural basado en UML, un factor fundamental para lograr un trabajo ágil y productivo. Además abre las puertas a utilizar herencia en el modelo de datos (esta opción estaría limitada si una herramienta obliga a que los objetos de datos hereden de una superclase por no soportar Java herencia múltiple).

1.4.2 Toplink

TopLink probablemente sea el marco de trabajo ORM más maduro. Primero fue desarrollado en Smalltalk en 1994. Desde 1997, su foco principal ha sido Java. Hoy día TopLink se puede utilizar tanto fuera como dentro de un servidor de aplicaciones J2EE. Aunque ahora es propiedad y está soportado por Oracle, funciona con cualquiera de las principales bases de datos (no sólo con Oracle). Funciona con todos los servidores de aplicaciones J2EE (o incluso fuera de ellos).

TopLink brinda muchas funcionalidades ORM. Como todas las buenas herramientas ORM, TopLink permite persistir POJOs, que tienen una mínima

dependencia de los APIs de TopLink. También proporciona varios mapeos avanzados, incluyendo soporte para tipos de árboles y OBjetos Binarios Largos (BLOB, por sus siglas en inglés). Además provee un sofisticado soporte para búsqueda optimista, lo que puede producir grandes beneficios si se utiliza apropiadamente [ORACLE10].

Es importante resaltar que TopLink no es puramente una herramienta ORM ya que también ofrece una solución de caché integrado (incluyendo coordinación del caché entre clusters), un soporte de consultas ricas, capacidades O-X, varias características de rendimiento (como la "indirección") que optimizan las interacciones con la base de datos, y el control de transacciones.

1.4.3 JDBC

JDBC es un API incluido dentro del lenguaje Java que permite la ejecución de instrucciones SQL (Structured Query Language: Lenguaje Estructurado de consultas) para el acceso a BD relacionales (crear, manipular, examinar y gestionar bases de datos) [SUNJDBC10]. Consiste en un conjunto de clases e interfaces escritos en Java que ofrecen un completo API para la programación de BD. La primera aparición de JDBC (JDBC 1.0) se encuentra dentro del paquete *java.sql* que fue incorporado en la versión del JDK 1.1.x (Java Development Kit) correspondiente a la versión 1.1 del lenguaje Java, JDBC 2.0 sigue estando en el mismo paquete pero en las versiones JDK 1.2 y JDK 1.3 que se corresponden con la versión 2 del lenguaje Java, o también denominada plataforma Java 2 (Java 2 Platform).

JDBC es un especificación formada por una colección de interfaces y clases abstractas, que deben implementar todos los fabricantes de drivers que quieran realizar una implementación de su driver Java y compatible con JDBC (JDBC-compliant driver). Debido a que JDBC está escrito completamente en Java también posee la ventaja de ser independiente de la plataforma [Fisher03]. No será necesario escribir un programa para cada tipo de BD, una misma aplicación escrita utilizando JDBC podrá manejar BD Oracle, Postgres, Derby o SQL Server. Además podrá ejecutarse en cualquier sistema que posea una Máquina Virtual de Java, es decir, serán aplicaciones completamente independientes de la plataforma.

1.4.4 JDO

JDO (Java Data Objects) es una definición basada en interfaces de persistencia de objetos para el lenguaje Java que describe el almacenamiento, consultas e instanciaciones de objetos. Además soporta transparencia en la persistencia de objetos [CASTILLO04].

JDO es un estándar para persistir objetos java, no es una tecnología ORM en sí, en vez de esto, define una API que puede utilizarse para manipular la persistencia de objetos sin tener en cuenta el mecanismo de persistencia, pudiendo este ser cualquiera, desde un sistema basado en ficheros hasta una BD orientada a objetos.

Debido a que en la actualidad la mayoría de las implementaciones emplean BD relacionales, esto significa que estas implementaciones proveen ORM; JDO provee un mecanismo de persistencia totalmente transparente al desarrollador con el cual se puede diseñar cualquier cosa, incluyendo clases persistentes con un modelo de objetos, mediante las técnicas de diseño estándares de la programación orientada a objetos como herencia y composición.

Para implementar la persistencia con esta tecnología no se necesita tener un extra de conocimiento de código java. En un fichero XML se declaran las clases persistentes y de forma opcional sus atributos. Con la herramienta especial *enhacer* se procesan las clases javas después de la compilación para adicionar los métodos y los atributos que JDO requiere para proveer la persistencia.

1.4.5 JPA

JPA al igual que Hibernate, Toplink, proporciona un modelo de persistencia basado en POJO's para mapear bases de datos relacionales en Java. Puede utilizarse directamente en aplicaciones web y aplicaciones clientes. Fue desarrollado por el grupo de expertos de EJB 3.0 como parte de JSR-220 (Java Specification Requerimient, especificación implementada por un proveedor de servidor de aplicaciones), aunque su uso no se limita a los componentes software EJB.

En su definición, se han combinado ideas y conceptos de los principales frameworks de persistencia como Hibernate, Toplink y JDO, y de las versiones

anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA.

El mapeo objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones [ORACLE10] en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML. También pueden definirse transacciones como anotaciones JPA.

JPA consta de tres áreas:

- El API de persistencia en java (JPA).
- El lenguaje de consultas.
- El mapeo de los metadatos objeto/relacional.

En esta tecnología una entidad es un objeto de dominio de persistencia la cual representa una tabla en el modelo de datos relacional, cada instancia de esta entidad corresponde a un registro en esa tabla [SCHINCARIOL06]. El estado de persistencia de una entidad se representa a través de campos persistentes o propiedades persistentes. Estos campos o propiedades usan anotaciones para el mapeo de estos objetos en el modelo de base de datos.

El estado persistente de una entidad puede ser accesible a través de variables de instancia a la entidad o bien a través de las propiedades de estilo de JavaBean.

1.5 Bases de Datos

Las Bases de Datos (BD) son un conjunto de datos estructurados relacionados entre si llamadas comúnmente entidades.

Las entidades no son más que aquello que tenga existencia propia e independiente con una serie de características (atributos) de las cuales se quiere almacenar información, las mismas se clasifican en fuertes y débiles. Las débiles son entidades que no pueden existir si no existe otra entidad y las fuertes son las que no tienen dependencia de alguna otra [CHURCHER07].

Las BD se pueden clasificar fundamentalmente de dos formas: según la variabilidad de los datos y en cuanto a los modelos de BD.

1.5.1 Clasificación según la variabilidad de los datos

Teniendo en cuenta la variabilidad de los datos se pueden clasificar de la siguiente manera:

Bases de datos estáticas:

Son bases de datos de sólo lectura, empleadas principalmente para almacenar datos históricos, que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, esencialmente para realizar proyecciones y tomar decisiones.

Bases de datos dinámicas:

Son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta.

1.5.2 Clasificación según el modelo de los datos

Existen varios tipos de modelos de bases de datos los cuales se clasifican como sigue:

- Relacional: Consiste en una base de datos con una colección de tablas a cada una de las cuales se le asigna un nombre único y una fila de la tabla representa una relación entre un conjunto de valores (un registro).
- Orientado a objetos: Almacena en la base de datos los objetos completos (estado y comportamiento). Incorpora todos los conceptos importantes del paradigma de objetos como el *encapsulamiento*, la *herencia* y el *polimorfismo*.
- Objeto-relacional: Esta integra las características de las *orientadas a objetos* y las *relacionales*.
- Jerárquico: Su estructura básica es el árbol. Presenta un nodo padre y una serie de nodos hijos, la conexión se hace eligiendo quien va a ser el padre y quien va a ser el hijo.
- Red: Es un modelo ligeramente distinto del jerárquico; la diferencia es la modificación del concepto de *nodo* que permite que un mismo nodo tenga varios padres. Se conforma por una colección o set de registros, los cuales se conectan entre sí por medio de los enlaces en la red.

1.6 Sistemas Gestores de Bases de Datos (SGBD)

Los SGBD son herramientas que permiten definir, construir y manipular bases de datos, sus principales funcionalidades son las realizaciones de consultas, actualizaciones, inserciones, borrados y modificaciones de los datos.

Los SGBD se clasifican según la representación de los datos en la BD y le corresponden a la clasificación de la misma. En la actualidad los tipos de SGBD más empleados son el relacional, el orientado a objeto y el objeto-relacional.

Un SGBD es una interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Este software ofrece facilidades para el manejo de grandes cantidades de datos.

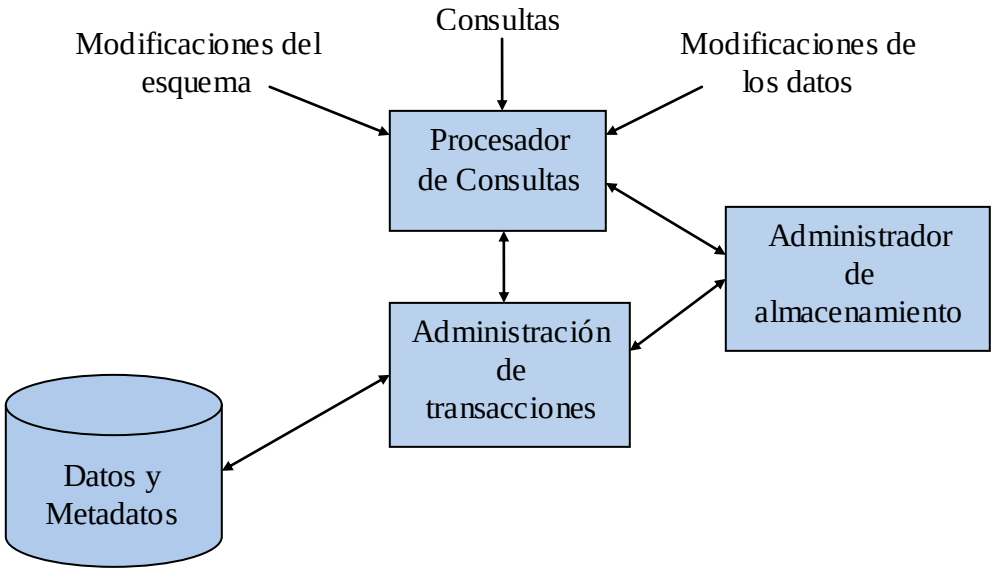


Figura 1. 1 Arquitectura de los SGBD

Algunos ejemplos de SGBD son MySQL, Oracle, PostgreSQL, MS SQL Server, DB2, Derby entre otros. A continuación se describen por sub-epígrafes los principales SGBD que se utilizan en la actualidad.

1.6.1 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Se ofrece bajo la GNU GPL para cualquier uso compatible con este tipo de licencia, pero por otro lado las empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI.

Al contrario que proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado.

MySQL, es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

Existen varias APIs que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby,Gambas, REALbasic (Mac), FreeBASIC, y Tcl; cada uno de estos utiliza una API específica. También existe un interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse

con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP.

MySQL es muy utilizado en aplicaciones web como, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

Es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

1.6.2 PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada al software libre, publicado bajo la licencia BSD. Como en muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group) [POSTGRESQL05].

PostgreSQL ha tenido una larga evolución, comenzando con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con el mismo, Michael volvió a la Universidad para trabajar en un nuevo proyecto sobre la experiencia de Ingres, el cual se llamó "post-ingres" o simplemente POSTGRES.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman

una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En POSTGRES la base de datos «comprendía» las relaciones y podía obtener información de tablas relacionadas utilizando reglas.

Para el año 1996 se unieron al proyecto personas ajenas a la Universisdad como Marc Fournier, Bruce Momjian y Vadim B. Mikheev quienes comenzaron a trabajar para estabilizar el código de Postgres95. En el año 1996 decidieron cambiar el nombre de Postgres95 de tal modo que refleje la característica del lenguaje SQL y lo terminaron llamando PostgreSQL.

Con el pasar del tiempo muchos desarrolladores entusiastas de los motores de base de datos se unieron al proyecto y entre todos comenzaron a incorporar muchas características al motor. Una de sus principales características es la alta concurrencia, mediante un sistema denominado MVC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

1.6.3 Derby

Apache Derby tiene su origen en la empresa Cloudscape Inc, en Oakland, California fundada en 1996 para desarrollar una tecnología de base de datos para Java [ZIKOPOULOS05]. La primera versión del motor de base de datos, que en aquel tiempo se llamó JBMS, tuvo lugar en 1997. Posteriormente el producto fue renombrado como Cloudscape y aparecieron versiones nuevas cada seis meses. En 1999 Informix Software, Inc., adquirió a Cloudscape, Inc y dos años más tarde IBM adquirió los activos de Informix Software, incluyendo Cloudscape. El motor de base de datos fue renombrado a IBM Cloudscape y

continuaron apareciendo versiones, enfocadas principalmente a usos embebidos en productos Java de IBM y middleware.

En agosto de 2004 IBM cedió el código a la Apache Software Fundation para Derby, un proyecto patrocinado por el proyecto Apache DB. El proyecto Derby continuó desarrollándose como sub-proyecto base de datos de alto nivel en Apache. La Sun se unió al proyecto Derby con el objetivo de utilizarlo como componente en sus propios productos, y con el lanzamiento de Java 6 en diciembre de 2006, comenzó a empaquetar Derby en el JDK llamado Java DB.

Las características principales de Derby son:

- APIs para JDBC y SQL: soporta todas las características de SQL92 y la mayoría de SQL99. La sintaxis SQL usada proviene de IBM DB2.
- Su código mide alrededor de 2000KB comprimido.
- Soporta procedures, compresión, roles, permisos y cifrado completo. Además posee SQL Schemas para separar la información en una única base de datos y control completo de usuarios.
- Trae soporte multilenguaje y localizaciones específicas.
- A partir de la versión 10.4 trae un sistema simple de replicación maestro esclavo.
- Transacciones y recuperación ante errores ACID.
- Posee tres productos asociados a la marca:
    - Derby Embedded Database Engine: El motor propiamente dicho.
    - Derby Network Server: Permite convertir Derby en una base de datos que sigue el modelo cliente-servidor tradicional.
    - Database Utilities: Un paquete de utilidades.

1.7 Metodologías de desarrollo de software.

Los usuarios exigen calidad frente a los requisitos y los desarrolladores deben contar con técnicas y herramientas que le permitan lograr la satisfacción de los usuarios.

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Esto lo hacen desarrollando un proceso detallado de un conjunto de procedimientos y

técnicas, apoyados en un soporte documental y herramientas con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. Usualmente consiste en una serie de fases fragmentadas en sub-fases (módulos, etapas, pasos, etc.). Esta fragmentación del proceso de desarrollo guía a los desarrolladores en la selección de las técnicas que debe elegir para cada etapa del proyecto facilitando la planificación, gestión, control y evaluación de los proyectos.

Para llevar a cabo el desarrollo de estos procesos las metodologías disponen de un lenguaje de modelación, el cual no es más que un lenguaje gráfico para visualizar, especificar, construir y documentar el sistema. El lenguaje más conocido y utilizado en la actualidad es UML (Unified Modeling Language: Lenguaje Unificado de Modelado), este ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema; y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Por tanto, resulta indispensable el uso de una metodología para el desarrollo de aplicaciones logrando un sistema íntegro, que cumpla con los requisitos de los usuarios. Pero usualmente los desarrolladores de software entran en la divergencia de cuál metodología utilizar para el desarrollo de su aplicación.

En los siguientes sub-epígrafes se detallan algunas de las metodologías más empleadas hoy día.

1.7.1 Metodologías tradicionales.

Entre las tecnologías tradicionales se encuentran RUP y MSF que son las más conocidas por los desarrolladores de software. El enfoque principal de estas metodologías se basa en hacer una documentación integral del proyecto y el cumplimiento de un plan de trabajo, todo esto se realiza en la fase inicial del desarrollo del proyecto. A continuación se especifican con más detalles estas metodologías.

1.7.1.1 Rational Unified Process (RUP).

RUP es un proceso de desarrollo de software y junto con UML, constituye una metodología para el análisis, implementación y documentación de sistemas. Proporciona funcionalidades para asignar tareas y responsabilidades dentro de una organización de desarrollo.

Se basa en seis principios fundamentales:

1. Adaptar el proceso
2. Equilibrar prioridades
3. Demostrar valor iterativamente
4. Colaboración entre equipos
5. Elevar el nivel de abstracción
6. Enfocarse en la calidad

Su objetivo fundamental es garantizar la producción del software con alta calidad para satisfacer las necesidades de los usuarios.

El ciclo de vida del desarrollo de una aplicación con esta metodología tiene cuatro etapas:

- Inicio: para determinar la visión del proyecto.
- Elaboración: para especificar la arquitectura óptima.
- Construcción: para obtener la capacidad operacional inicial.
- Transmisión: para poner en circulación el proyecto.

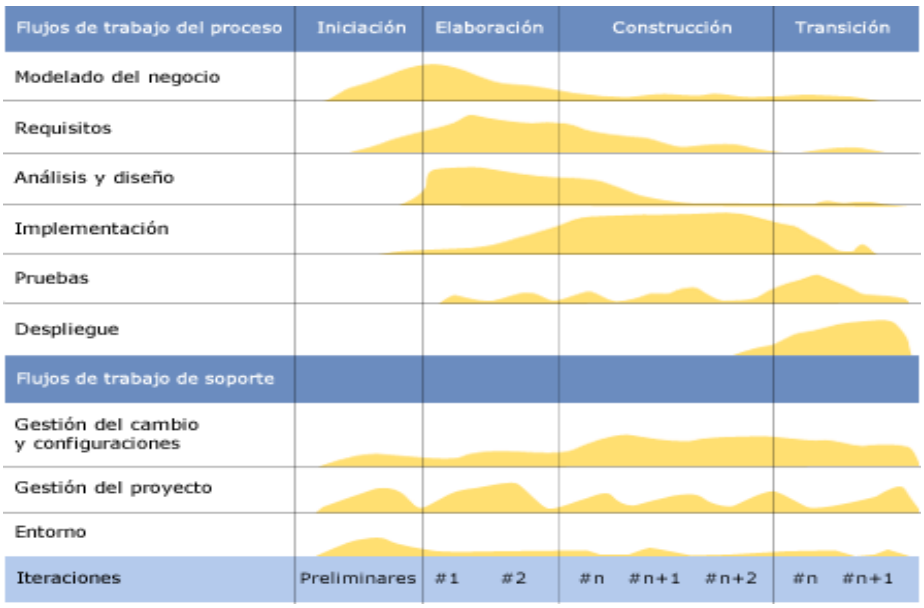


Figura 1. 2 Etapas de la metodología RUP

1.7.1.2 Microsoft Solution Framework (MSF)

MSF es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, para controlar la planificación, el desarrollo y la gestión de proyectos. Se enfoca en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF presenta varios modelos encargados de planificar las diversas partes implicadas en el desarrollo de cualquier proyecto:

- Modelo de Arquitectura de Proyecto
- Modelo de Equipo
- Modelo de Proceso
- Modelo de Gestión del Riesgo
- Modelo de Diseño del Proceso
- Modelo de Aplicación

El ciclo de vida de un proyecto empleando esta metodología consta de las fases:

- Visión y Alcances.
- Planificación.
- Desarrollo.
- Estabilización.
- Implantación.

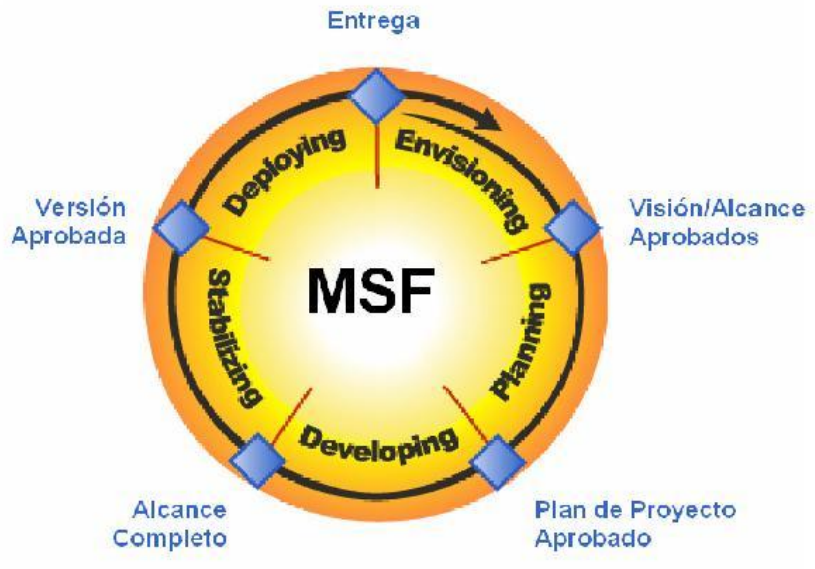


Figura 1. 3 Etapas de la metodología MSF

1.7.2 Metodologías ágiles.

Las metodologías ágiles surgen a partir de una reunión celebrada en Utah-EEUU. En este debate participaron un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Luego de esta reunión se creó The Agile Alliance 3, una organización, con el objetivo de promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. Esto se concretó con el *"Manifiesto Ágil"*, un documento que resume la filosofía "ágil" [CANÓS].

Este manifiesto plantea:

- Valorar a las personas y la interacción entre ellas como equipo por encima del proceso y herramientas que se utilicen.
- Tiene vas valor un producto software que funcione que realizar una buena documentación.
- Es más importante colaborar con el cliente antes que el trámite de contratación.
- Tener en cuenta los cambios que puedan surgir en el desarrollo del software antes que seguir un plan estricto.

Entre las metodologías ágiles se encuentra Scrum, Crystal Methodologies, DSDM (Dynamic System Development Method), ASD (Adaptive Software Development), FDD (Feature-Driven Development), XP (Extreme Programming), ICONIX.

En los siguientes sub-epígrafes se describen XP e ICONIX pues son dos de las metodologías ágiles más usadas en la actualidad por los desarrolladores.

1.7.2.1 Programación Extrema (XP por sus siglas en inglés).

XP es una metodología ágil que hace énfasis en la satisfacción del cliente. Se centra en el trabajo en equipo, como vía más factible para lograr el desarrollo del producto software. Todos los miembros de un equipo son considerados por igual (administradores, desarrolladores, clientes, etc.). Además presenta la filosofía de que los equipos se auto organicen las tareas para darle solución al problema tan eficazmente como sea posible.

Las características fundamentales del método son:

- Desarrollo iterativo e incremental
- Pruebas unitarias continuas
- Programación por parejas.
- Frecuente interacción.
- Corrección.
- Refactorización del código.
- Propiedad del código compartido.
- Simplicidad en el código.

XP mejora un proyecto software en cinco maneras principalmente:

- Comunicación (communication)
- Simplicidad (simplicity)
- Retroalimentación (Feedback).
- Respeto (respect)
- Coraje (courage)

Todas estas características se reflejan en el siguiente diagrama:

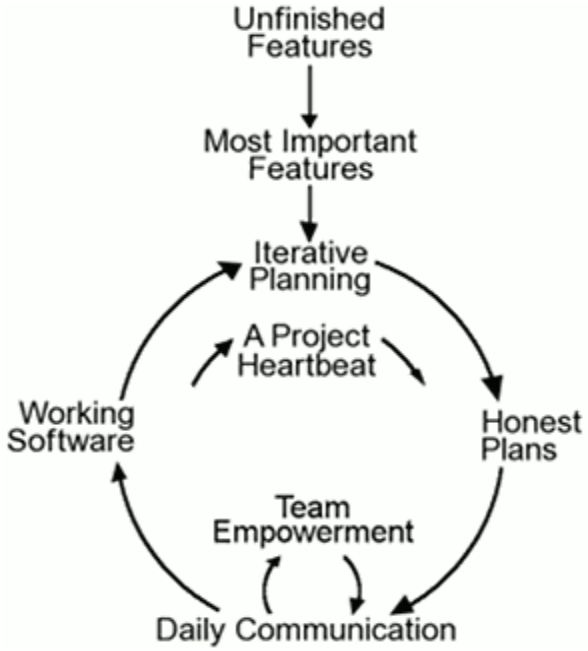


Figura 1. 4 Esquema de la metodología XP

1.7.2.2 ICONIX

ICONIX es una metodología ágil considerada como un proceso de desarrollo de software. Se enmarca entre la complejidad de RUP y la simplicidad de XP, pero no excluye el análisis y diseño como XP. Expone las actividades a realizar

en cada etapa del desarrollo y describe los pasos a seguir para el cumplimiento de las mismas [ROSENBERG].

Las características principales de ICONIX son:

- Iterativo e Incremental: muchas iteraciones ocurren entre el desarrollo del modelo de dominio y la identificación de los casos de uso de una aplicación, el modelo estático se refina incrementándose por los modelos dinámicos.
- Trazabilidad: cada paso es referenciado por algún requerimiento. Se define trazabilidad a la capacidad de seguir una relación entre los diferentes artefactos.
- Dinámica del UML: ofrece un uso dinámico del UML como los diagramas de caso de uso, diagramas robustez y diagramas secuencia.

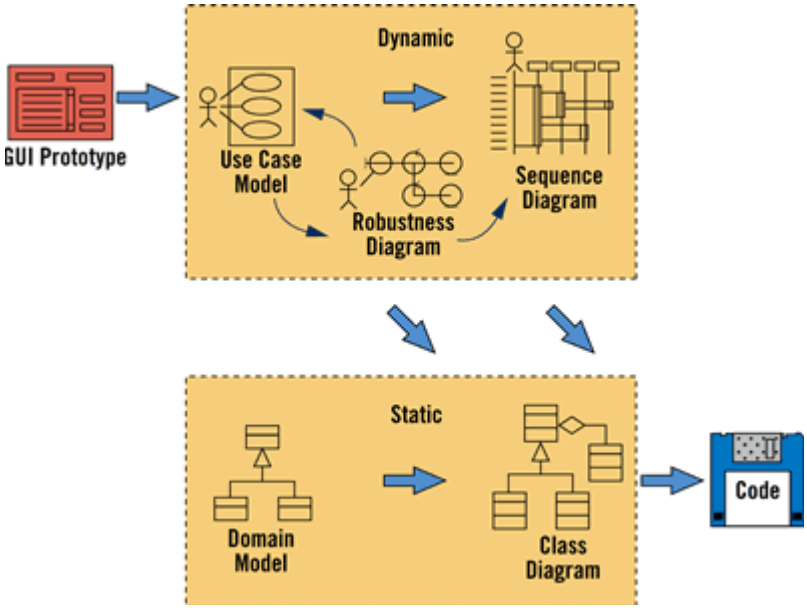


Figura 1. 5 Esquema de la metodología ICONIX

1.8 ¿Por qué se selecciona JPA, Derby, ICONIX?

JPA es seleccionado como la tecnología para persistir los objetos a través del framework EclipseLink JPA 2.0 ya que esta tecnología soporta o provee cada uno de los mejores rasgos de los frameworks de persistencia que existen en la actualidad como se muestra en la tabla 1.1.

Crear entidades con JPA es tan simple como crear clases serializables. JPA soporta una amplia colección de datos, uso concurrente, consistencia de datos

y consultas compatibles con JDBC. Como un software Objeto-Relacional JPA permite el uso de conceptos avanzados de la programación orientada a objetos tales como la herencia, polimorfismo entre otros. JPA se enfoca en base de datos relacionales y es extremadamente fácil de usar.

Soportes:	Hibernate	Toplink	JDBC	JDO	JPA
Objeto Java	Sí	Sí	No	Sí	**Sí**
Conceptos Avanzados OO	Sí	Sí	No	Sí	**Sí**
Integridad transaccional	Sí	Sí	Sí	Sí	**Sí**
Concurrencia		Sí	Sí	Sí	**Sí**
Datos largos	Sí	Sí	Sí	Sí	**Sí**
Esquemas existentes	Sí	Sí	Sí	Sí	**Sí**
Relacional y No-Relacional	Sí	Sí	No	Sí	**Sí**
Consultas	Sí	Sí	Sí	Sí	**Sí**
Estándares estrictos /Portabilidad	No	No	No	Sí	**Sí**
Simplicidad	Sí	Sí	Sí	Sí	**Sí**

Tabla 1. 1 Comparación de las tecnologías de persistencia

El software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo es una aplicación pequeña que no requiere una base de datos que soporte varios usuarios simultáneos. Para este tipo de aplicaciones Apache Derby es una perfecta solución, especialmente porque no requiere de un Administrador de Base de Datos (DBA, por sus siglas en inglés).

Los aspectos que marcan las diferencias de Derby respecto a otros SGBD son:

- *Fácil de administrar:* cuando se embebe en una aplicación cliente, el sistema Derby no requiere de intervención administrativa.
- *Es embebido:* las aplicaciones pueden embeber el Sistema de Administración de Base de Datos (DBMS) en el proceso de aplicación, eliminando la necesidad de administrar por separado un proceso o un servicio de una base de datos.
- Puede ejecutarse como un proceso separado, usando el framework de servidor de red que se desee.
- *Es una librería de clases de puro Java:* esto es importante para los desarrolladores que utilizan Java que tratan de mantener las ventajas de la tecnología Java, tales como la independencia de la plataforma y es de fácil configuración.
- *No necesita una JVM propietaria:* Escrita enteramente en el lenguaje Java, se ejecuta sobre cualquier JVM certificada.
- El motor es ligero.
- *Store Procedure:* Provee la facilidad de usar procedimientos almacenados y funciones en Java que pueden ejecutarse en cualquier nivel de una aplicación. Derby no tiene un lenguaje de procedimientos almacenados propietario, para ello utiliza JDBC.

Una vez analizados estas características y teniendo en cuenta que el software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo no requiere que se despliegue en una arquitectura cliente-servidor, se escoge Derby como base de datos para la persistencia y recuperación de los datos que son utilizados para determinar los valores óptimos de que se desean conocer.

Los desarrollos de software van cambiando debido a las innovaciones tecnológicas, estrategias de mercado y otros sucesos de la industria de la informática, esto trae como consecuencia que los desarrolladores evolucionen para obtener aplicaciones en menor tiempo y menos costosas. ICONIX es una metodología simple en comparación con las tradicionales, las cuales se centran más en abarcar todo el ciclo de vida del proyecto y la planificación del mismo.

Entre las metodologías agiles XP sería una buena solución a utilizar, pero esta presenta la política de trabajar en equipo y la programación en parejas lo cual hace que se descarte esta metodología, debido a que la aplicación se desarrolla por un solo integrante.

Teniendo en cuenta estos aspectos se escoge la metodología ágil ICONIX para guiar el desarrollo del software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo. Como herramienta case para desarrollar la ingeniería se toma el EA (Enterprise Architecture) ya esta herramienta que posee un plugin desarrollado especializado sobre ICONIX.

1.9 Conclusiones

En el presente capítulo se hizo la fundamentación del objeto de estudio planteado, se mencionaron algoritmos y métodos para la programación de problemas generales seleccionándose el método de exploración para la implementación de la software propuesto, se realizó un análisis sobre las tecnologías de persistencia más difundidas y usadas hoy día por los desarrolladores de software, entre las que se destacan Hibernate, Toplink, JDBC, JDO y JPA, percibiendo la potencialidad que estas brindan para el desarrollo de aplicaciones que acceden a bases de datos relacionales y se hizo una descripción sobre los sistemas gestores de base de datos MySQL, PostgreSQL y Derby.

Después de un exhaustivo análisis se determinó que Derby y JPA eran las herramientas apropiadas para el desarrollo del software.

También se valoraron las metodologías para el desarrollo de aplicaciones tanto las tradicionales como las ágiles arribando a la conclusión de que la metodología ICONIX es la más ideal para guiar la construcción de la solución propuesta.

CAPITULO II DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En este capítulo se realiza una descripción de los procesos de análisis, diseño e implementación del software para la determinación del régimen de marcha y capacidad de carga del conjunto tractivo. ICONIX como metodología para el desarrollo de aplicaciones propone la realización de una serie de pasos para concretar la fabricación del producto informático, este proceso es una guía que nos describe como ir desde la captura de los requerimientos hasta la codificación del software.

También se hace la valoración de sostenibilidad del software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo en las cuatro dimensiones que se evalúa la misma.

2.2 Análisis de los requerimientos

Esta etapa inicial tiene como objetivo principal realizar la revisión de requerimientos del sistema. Incluye dos fases entre las que se encuentra el proceso de captura de requisitos, la cual se lleva a cabo mediante la comunicación con los clientes y los usuarios. Debido a que los requisitos son las condiciones o términos que la aplicación debe satisfacer, así como las especificidades de sus acciones. La segunda fase es el análisis de requisito para determinar si estos son confusos, parciales, ambiguos, o contradictorios. Para una mejor observación y entendimiento es recomendable obtener una lista de requerimientos funcionales, enmarcados en el contexto de pequeños párrafos donde se describen cada uno de ellos.

Requerimientos funcionales

La solución propuesta de la aplicación debe permitir:

1. Que el especialista inserte los parámetros característicos del CAI, del cual va a realizar la evaluación del caso de estudio.
2. La actualización de los equipos de tiro existentes actualmente.
3. La inserción de equipos de tiro que aún no se hayan comercializados.
4. La eliminación de los equipos de tiro que ya no estén en explotación.
5. La actualización de los caminos existentes actualmente.

6. La inserción de caminos que aún no se hayan valorado.
7. La eliminación de los caminos que ya no sean transitados.
8. La actualización de las cosechadoras existentes actualmente.
9. La inserción de cosechadoras que aún no se hayan comercializados.
10. La eliminación de las cosechadoras que ya no estén en explotación.
11. La determinación del valor óptimo de la capacidad de carga de un remolque cañero y el comportamiento del régimen de velocidades del conjunto tractivo.
12. Procesar estadísticamente los indicadores de eficiencia y valores de salida que se evalúan en cada uno de los casos de estudio objeto de análisis.
13. Guardar la información del caso de estudio analizado.
14. Cargar la información de un caso de estudio previamente analizado.
15. Generar un reporte final con la información tabulada y analítica de las valoraciones realizadas sobre los casos que se han estudiado o evaluado con los siguientes aspectos.

2.3 Modelo del dominio

El modelo del dominio hace un compendio de las palabras y conceptos relacionados con el problema que el sistema diseñado trata de resolver. Cuando se crea el modelo del dominio, se realiza una representación de los objetos y las acciones que debe cumplir el sistema [ROSENBERG07]. Para la realización del mismo se utiliza la denominada "técnica de inspección gramatical". Esta técnica da una pasada rápida sobre el material relevante que se posee (requerimientos y/o glosario de términos), resaltando los sustantivos y verbos que se utilizan en el proceso de determinación del régimen de marcha y capacidad de carga del conjunto tractivo.

A continuación se describen las definiciones que se consideran más relevantes para el desarrollo de la solución propuesta.

Especialista (transportista, inversionista o fabricantes): Son las personas que necesitan determinar el valor idóneo de la capacidad de carga y la configuración geométrica de los remolques.

Caso de estudio: Es la evaluación de una combinación de datos del equipo de tiro, la cosechadora, el camino y los parámetros del CAI para determinar la capacidad de carga del remolque.

Equipo de Tiro: Son los camiones o tractores que se emplean en la actividad de transporte de la cosecha cañera.

Cosechadora: Maquinaria que se utiliza para el corte de la caña.

Camino: Son los diferentes vías por las cuales se trasladan los equipos de tiro para trasladar la caña desde el campo de caña hasta el central.

Parámetros: Datos específicos de los CAI que se emplean en para la determinación de la capacidad de carga de los remolques.

Reporte: Información que se genera a partir del caso de estudio analizado.

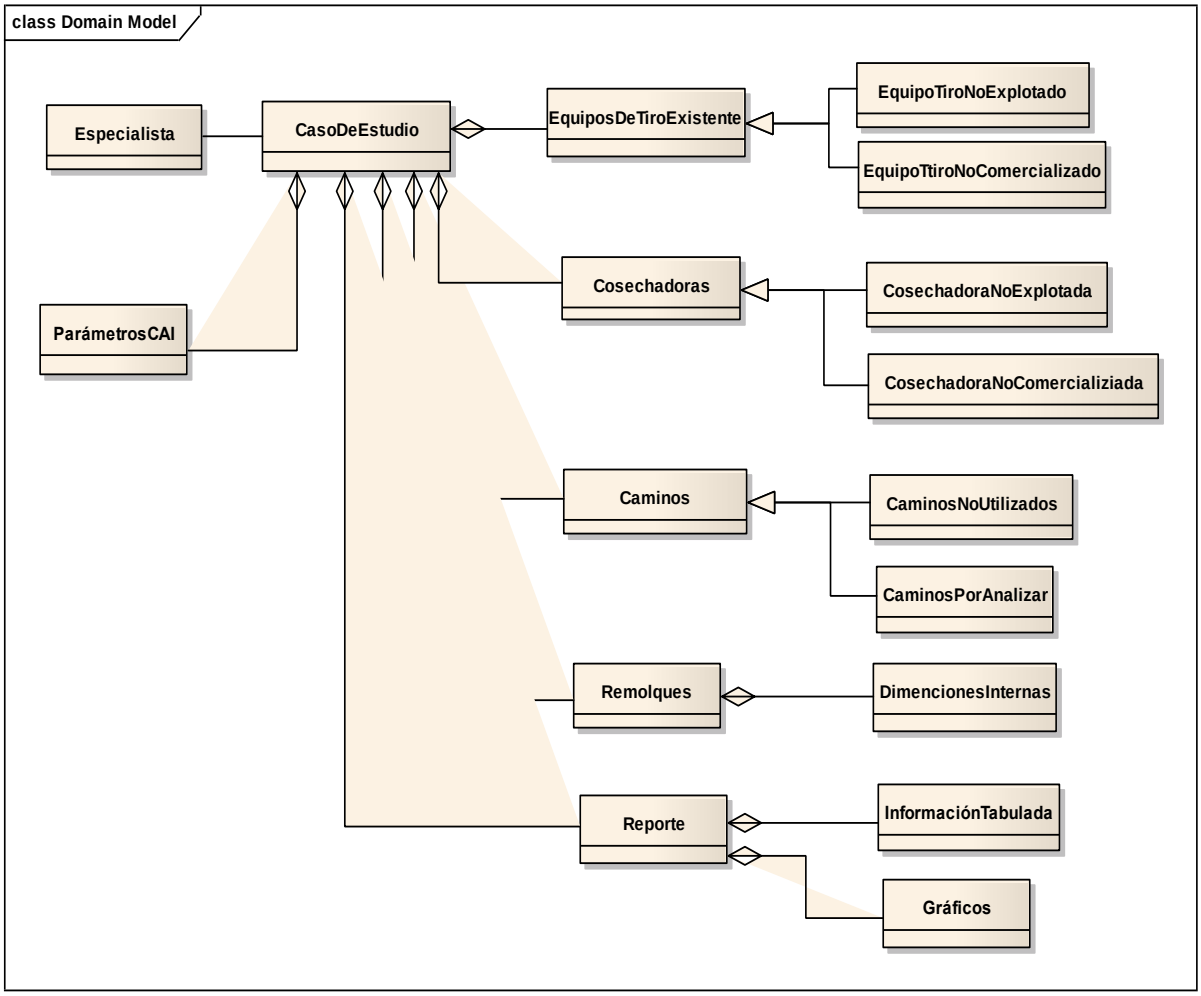


Figura 2. 1 Modelo del dominio

2.4 Descripción de los casos de uso

El paso siguiente que propone el proceso de ICONIX es la descripción de los casos de uso el cual se descompone por tres partes.

- La identificación de los casos de uso mediante el diagrama.
- La organización de los casos de uso por paquetes.
- La descripción textual de los casos de uso.

A continuación se muestra un diagrama general de los casos de uso presentes en el software para la determinación del régimen de marcha y carga del conjunto tractivo.

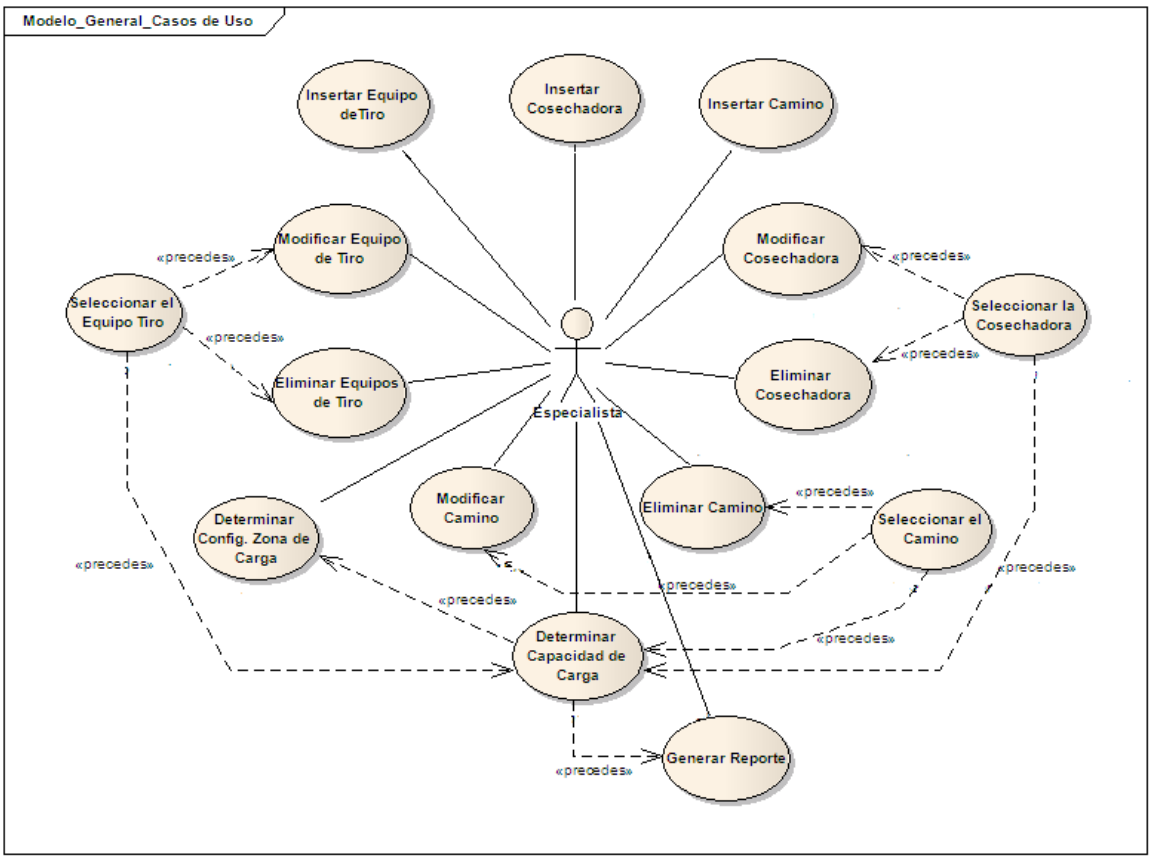


Figura 2. 2 Diagrama general de los casos de uso para la aplicación

Para facilitar el entendimiento el autor separa los casos de uso por paquetes resultando como se muestra en las siguientes figuras.

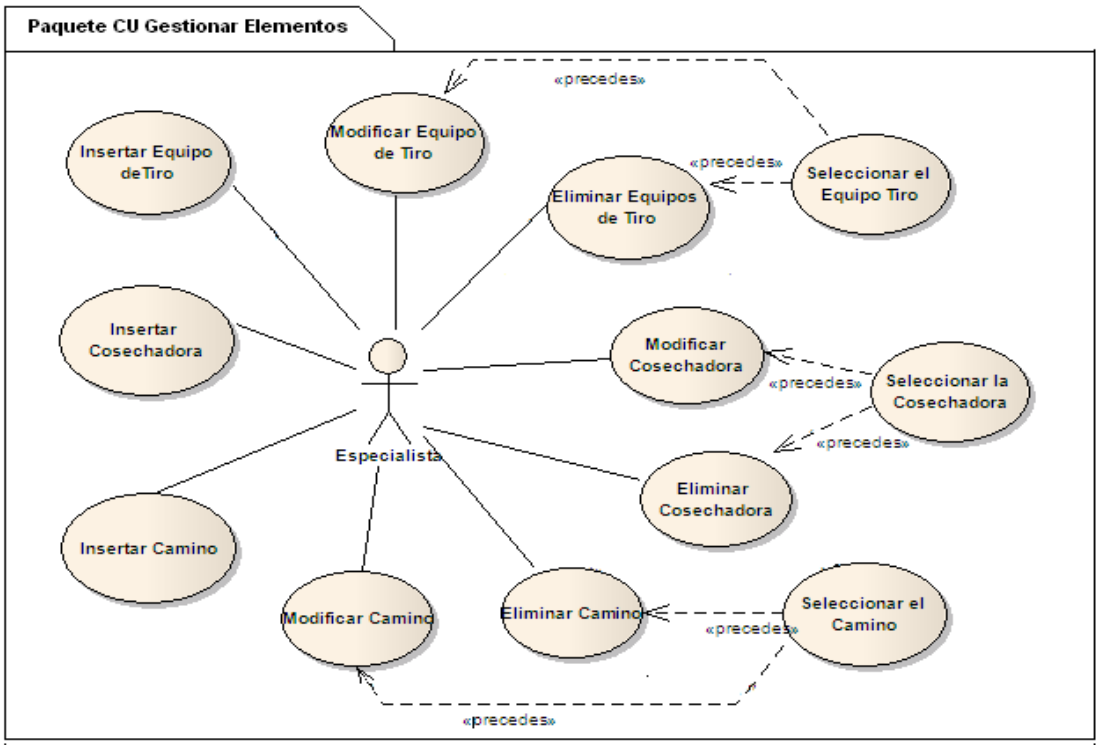


Figura 2. 3 Diagrama de los casos de uso del paquete Gestionar elementos de la aplicación

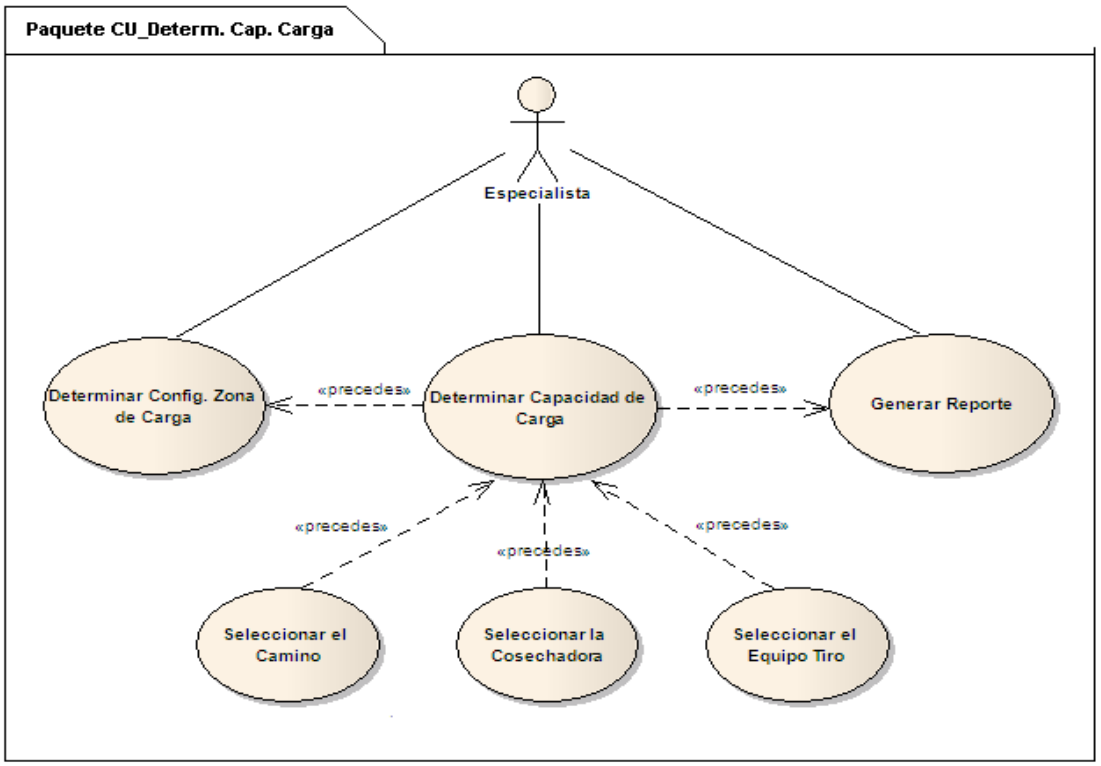


Figura 2. 4 Diagrama de los casos de uso del paquete Determinar Capacidad de Carga de la aplicación

2.4.1 Plan de ejecución.

Una vez definidos los casos de uso y agrupados por paquete se elabora un plan de realización, dándole un orden ascendente según la complejidad para lograr la implementación de cada requerimiento de la aplicación. A continuación se muestra una tabla con el plan a realizar para el software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo.

Prioridad	Caso de Uso
1	Insertar Equipo de Tiro
	Insertar Cosechadora
	Insertar Camino
2	Eliminar Equipo de Tiro
	Eliminar Cosechadora
	Eliminar Camino
3	Modificar Equipo de Tiro
	Modificar Cosechadora
	Modificar Camino
4	Determinar la Capacidad de Carga.
5	Guardar información del caso de estudio
6	Cargar caso de estudio.
7	Generar Reporte

Tabla 2. 1 Plan de Ejecución de los casos de uso

2.4.2 Descripción textual de los casos de uso.

En este sub-epígrafe se hace una breve descripción sobre los casos de uso y su relación con los actores del sistema, este paso se desarrolla mediante una tabla en la cual se presenta el nombre del caso de uso y su respectiva descripción. La descripción se divide en dos aspectos, primero el flujo básico de las operaciones que debe realizar el proceso y segundo el flujo alternativo que no es más que una captura de posibles errores o sucesos que pueden ocurrir durante la operación.

En la siguiente tabla se procede a la descripción de algunos de los casos de uso implicados en el software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo.

Caso de Uso	Descripción Textual
Insertar Equipo Tiro.	Flujo básico: El caso de uso comienza cuando el especialista necesita insertar un nuevo equipo de tiro con sus características en la Base de Datos, el software realiza la operación; efectuando la conexión a la BD embebida, verificando que los datos que se van a insertar son válidos, comprobando que no exista otro equipo de tiro con el mismo identificador en la BD y finalmente insertando el objeto, el caso de uso termina cuando el objeto es insertado satisfactoriamente. Flujo alterno: Si el software no pudiese realizar la conexión, muestra al desarrollador la razón por la cual no se realizó la misma y termina al caso de uso; si los datos a insertar no son validos le sugiere al usuario que inserte los datos correctamente; si existe otro equipo de tiro con el mismo identificador en la BD le advierte al usuario que no puede ser insertado el nuevo y finaliza el caso de uso.
Modificar Equipo Tiro	Flujo básico: El caso de uso comienza cuando el especialista necesita modificar algún dato de los equipos de tiro existentes en la BD, el software realiza esta operación; verificando que los nuevos datos que se van a insertar son validos. El caso de uso finaliza cuando la aplicación modifica los datos. Flujo alterno: Si los nuevos datos no son validos el software le sugiere al usuario que inserte los datos correctamente y termina el caso de uso.
Eliminar Equipo Tiro	Flujo básico: El caso de uso comienza cuando el especialista desea eliminar uno (o varios) equipo (s) de tiro que ya no se

	este (n) explotando en la actividad de transporte de la cosecha, el software realiza la eliminación del (los) equipo (s) de tiro seleccionado (s). El caso de uso finaliza cuando el software borra el objeto. Flujo alterno: Si el software no pudiese eliminar un equipo de tiro, se muestra al desarrollador la causa y termina el caso de uso.
Determinar Capacidad de Carga.	Flujo básico: El caso de uso comienza cuando el especialista realiza un caso de estudio para determinar el valor óptimo de la capacidad de carga del remolque, el software realiza la operación. El caso de uso finaliza cuando el software determina el valor y muestra los resultados. Flujo alterno: Si el software no pudiese determinar la capacidad de carga, se muestra al especialista la causa y termina el caso de uso.

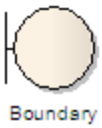
Tabla 2. 2 Descripción textual de los casos de uso

La descripción textual de los demás casos de uso que conforman la aplicación se pueden observar en el anexo 1.

2.5 Análisis de robustez.

Este paso de la metodología de desarrollo esta estrechamente aparejado con los casos de uso, el mismo brinda una idea preliminar de cómo diseñar un software que efectúe un caso de uso dado. Otro de los propósitos de esta actividad es descubrir en caso de que falte alguno, todos los objetos necesitados para la realización del producto informático. Este proceso se lleva a cabo mediante la construcción de un diagrama de robustez, para cada caso de uso. En las figuras 2.4 y 2.5 se presentan dos de los diagramas de robustez para el software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo.

Los principales artefactos que se utilizan en el diagrama de robustez son:



Los objetos del tipo "boundary" conocidos como "interfaz" representan mediante lo cual el actor va a interactuar con el sistema.

Los objetos del tipo "entity" generalmente representan objetos del modelo del dominio.

Los objetos del tipo "control" también llamados "controladores" representan las acciones que realiza el sistema.

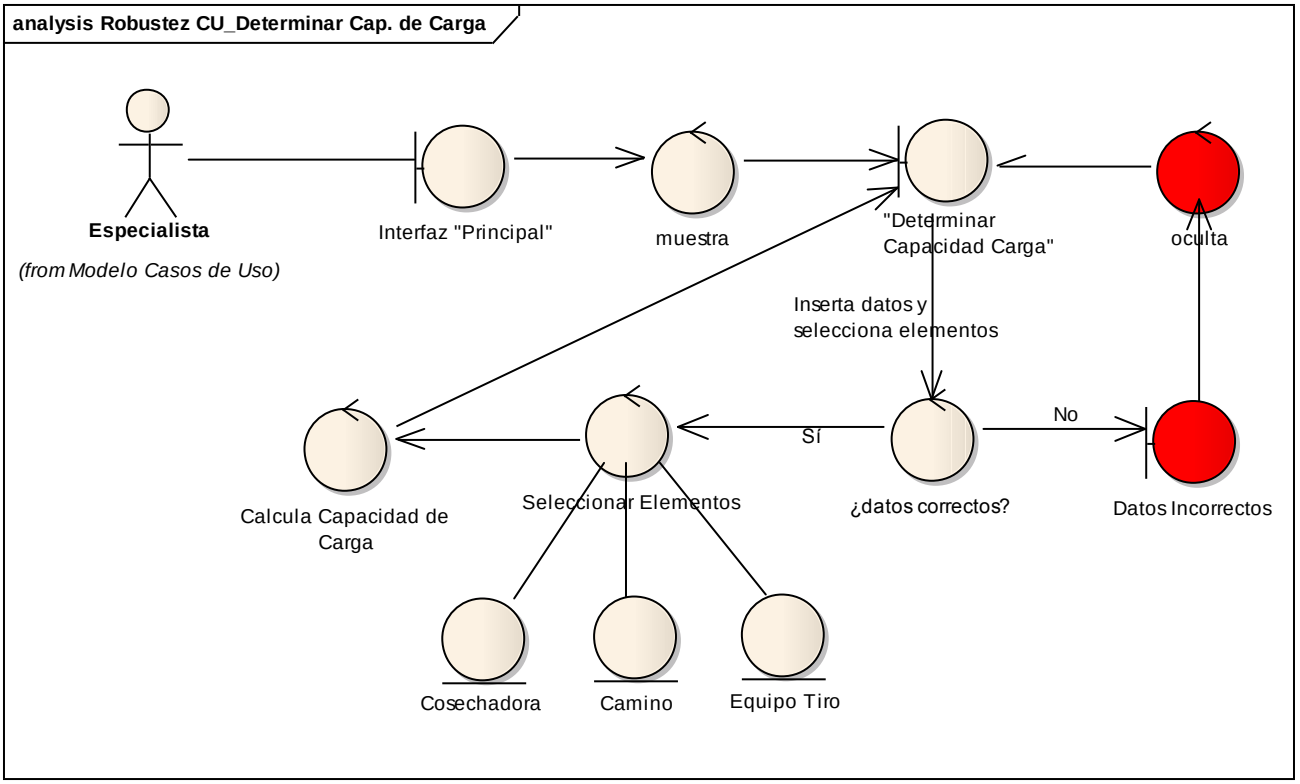


Figura 2. 5 Diagrama de robustez del caso de uso Determinar Capacidad de Carga

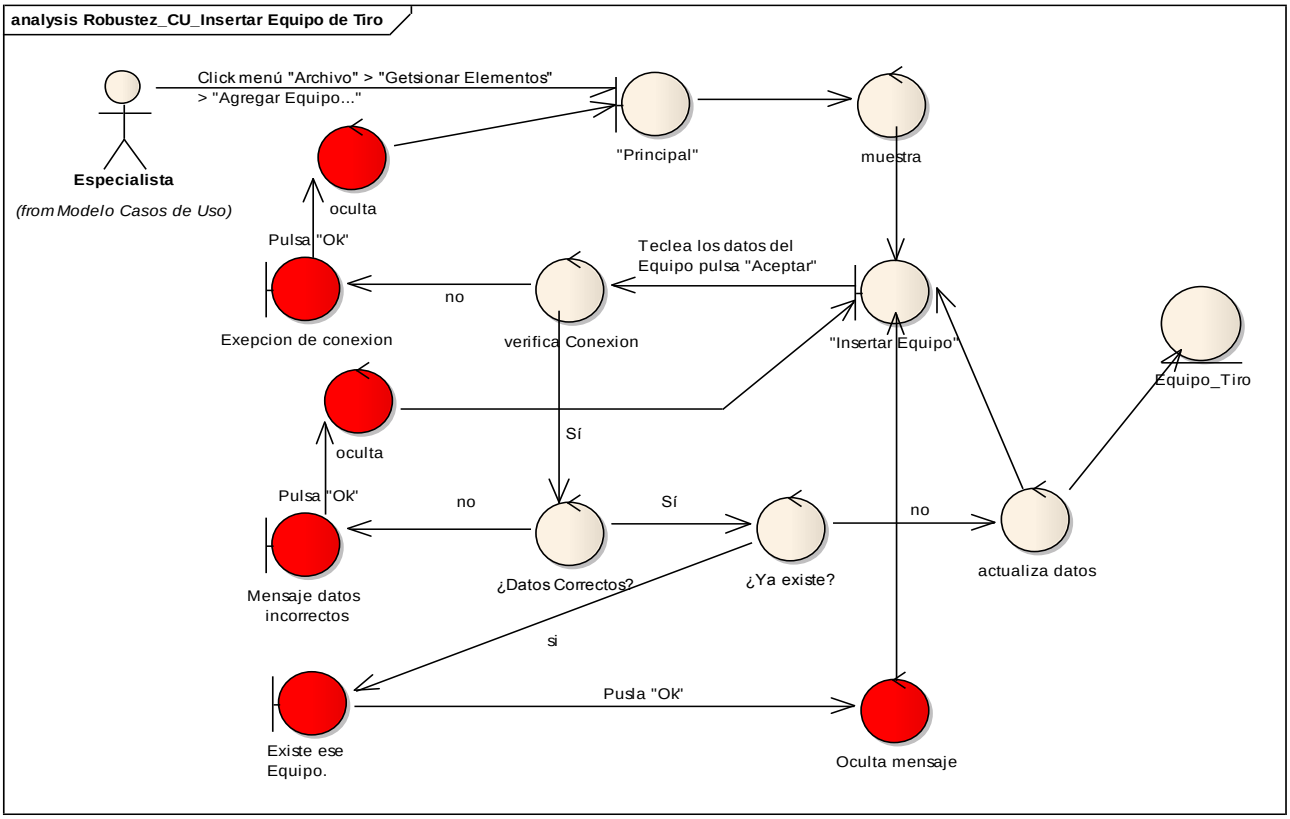


Figura 2. 6 Diagrama de robustez del caso de uso Insertar Equipo de Tiro

Los demás diagramas de robustez que conforman la aplicación se exponen en los anexos 2 al 8.

2.6 Arquitectura del software.

La arquitectura del software es la descripción de la aplicación que se pretende desarrollar en términos de su estructura. Es cimentar la arquitectura para satisfacer los requerimientos del negocio y el nivel de servicio de la aplicación que se va a desarrollar. En ella se recogen los aspectos tecnológicos, así como la topología del sistema [ROSENBERG].

Para la construcción de la arquitectura se analizan los requerimientos funcionales y los no-funcionales, así como las características específicas en cuanto al número de personas que van a interactuar con la aplicación.

2.6.1 Requerimientos no-funcionales.

Los requerimientos no-funcionales juegan gran un papel en esta etapa del desarrollo debido a que son la base para definir la arquitectura técnica que se ajusta al software que se desarrolla.

Estos requerimientos son propiedades o cualidades que el producto debe poseer, pues describen atributos del sistema o del ambiente del mismo, además explican las características que de una u otra forma puedan limitar al software. Se precisan teniendo en cuenta la aceptación de los usuarios finales, así como el buen funcionamiento, la flexibilidad y escalabilidad que proporciona el mismo.

Los requerimientos no-funcionales determinados para el software de determinación del régimen de marcha y capacidad de carga del conjunto tractivo propuestos en la presente investigación son:

Apariencia o Interfaz externa:

Ambiente orientado al entorno de trabajo del cliente, con el objetivo de que se sienta cómodo e identificado con la aplicación a través de simbologías acorde a la actividad además de la personalización de las acciones a desempeñar por los especialistas.

Usabilidad:

Por ser una aplicación de escritorio (desktop) portable, el especialista podrá realizar los análisis de los casos de estudio en cualquier maquina computadora.

Rendimiento:

El software debe tener una alta velocidad de procesamiento de los datos para mostrar de manera inmediata los resultados obtenidos y debe garantizar la precisión requerida en los cálculos internos debido a que se trata de optimizar la eficiencia en la actividad de transporte de la cosecha cañera.

Portabilidad:

Las herramientas utilizadas para el desarrollo del software propuesto son multiplataforma, lo cual implica que este también lo sea.

Facilidad de Mantenimiento

El software debe procurar facilidad de mantenimiento una vez implantado para posibilitar una mejoría continua del mismo, lanzando nuevas versiones.

Ayuda y documentación en línea

Debe contar con un Manual de Usuario y un sistema de ayuda que le ofrezca al usuario la suficiente orientación sobre cómo interactuar con el software.

Software

En este caso solo se necesita la Máquina Virtual de Java (JRE v1.6 o superior).

Hardware

Para ejecutar el software en las máquinas deben tener Microprocesador a 1.7 GHz de velocidad de procesamiento (o superior), y mínimo 256 MB de memoria RAM.

2.6.2 Modelo de despliegue.

El modelo de despliegue es un elemento importante de la arquitectura técnica el cual se representa generalmente por un diagrama de despliegue, en él se describen las capas que conforman la aplicación, siendo un modelo de objetos que describe la distribución física de la aplicación, además permite distribuir las funcionalidades entre las capas..

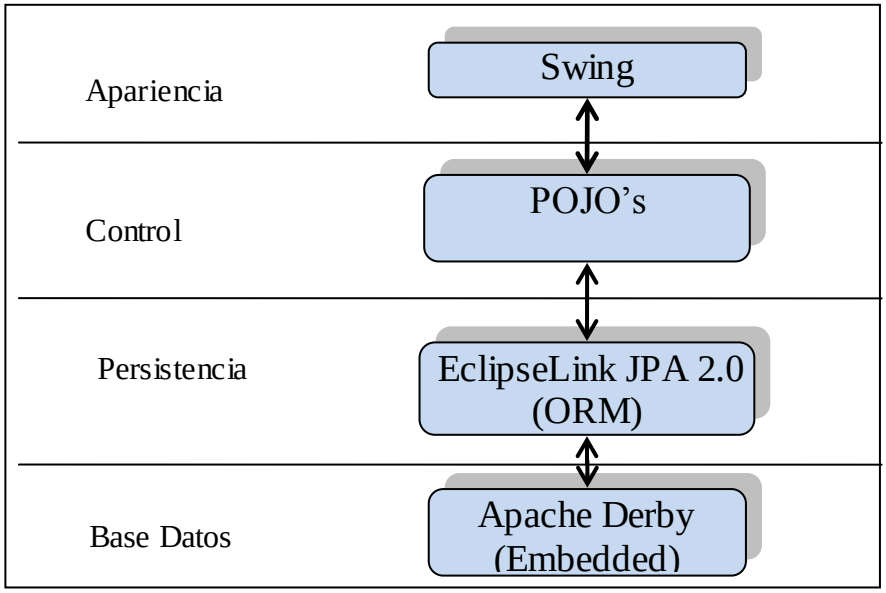


Figura 2. 7 Arquitectura del software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo

En el desarrollo del software propuesto se tuvo en cuenta los principios de una arquitectura multicapa, lo cual aumenta la mantenibilidad y flexibilidad de la aplicación, pues a la hora de corregir, perfeccionar o adaptar el software resulta sencillo detectar en que lugar han de realizarse las modificaciones necesarias.

En la capa de presentación se tienen las vistas las cuales son el medio de interacción con el usuario. En la capa de objetos se encuentran los POJOs los cuales son manejados posteriormente como entidades por los controladores. Para el manejo del modelo de datos se empleó JPA a través del framework EclipseLink, el cual aumenta la reusabilidad del código y establece un único punto de entrada para el acceso a las clases persistentes. La gestión de los datos almacenados en la BD se realiza a través de los controladores de entidades, lo cual abstrae un poco al programador de utilizar sentencias SQL y gestionar las conexiones a las BD. Para separar la lógica del negocio de la presentación se empleó el patrón de diseño MVC (Model View Controller).

2.7 Diseño detallado.

Con la construcción de los diagramas de robustez del diseño preliminar se hacen suposiciones sobre como interactuarían las clases entre ellas, ahora en la etapa del diseño detallado se precisan estas afirmaciones teniendo en cuenta la arquitectura técnica definida [ROSENBERG07].

2.7.1 Diagrama de Secuencia

Según la perspectiva de ICONIX, los diagramas de secuencia representan el producto de trabajo de un mayor modelo. Se dibuja un diagrama de secuencia que abarque el camino básico y todos los caminos alternativos dentro de cada uno de los casos de uso. Los resultados forman el centro de su modelo dinámico que se define en gran detalle.

Existen fundamentalmente cuatro tipos de elementos en un diagrama de secuencia: el texto para el camino de acción de los casos de uso, los objetos, los mensajes y los métodos (funcionalidades).

A continuación se muestran dos de los diagramas de secuencia para los casos de uso del software para le determinación del régimen de marcha y capacidad del conjunto tractivo, los restantes se encuentran en los anexos 9 al 16.

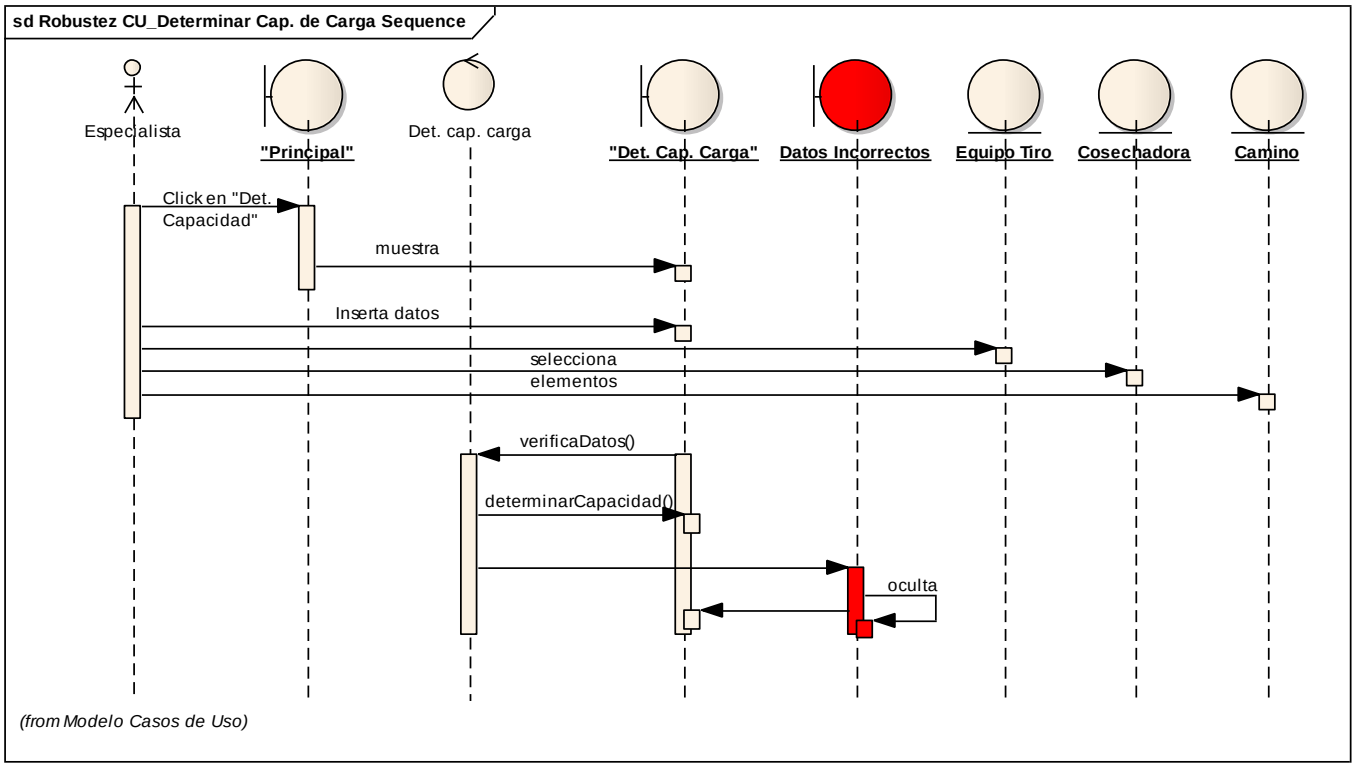


Figura 2. 8 Diagrama de secuencia del caso de uso Determinar capacidad de carga

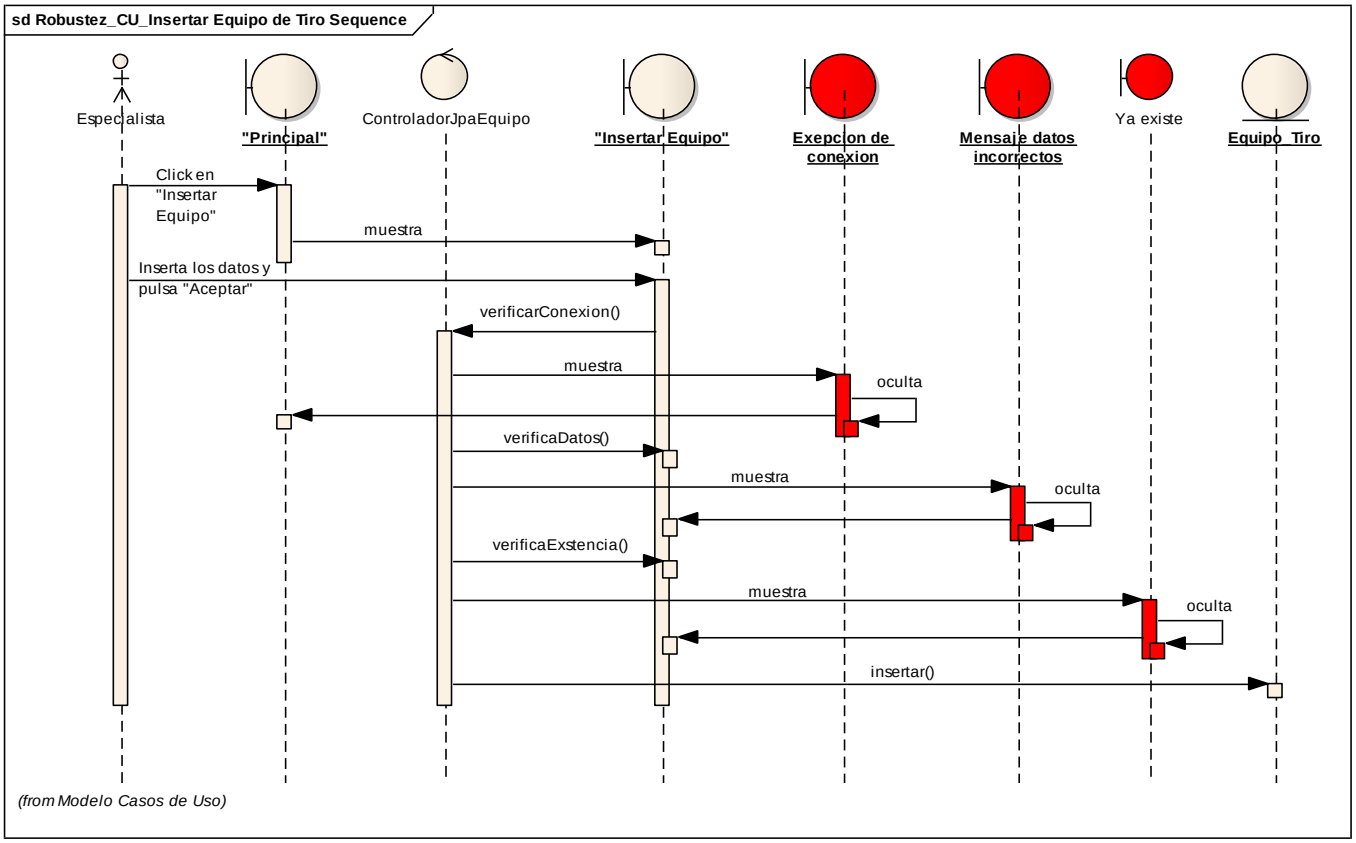


Figura 2. 9 Diagrama de secuencia del caso de uso Insertar Equipo de Tiro

2.7.2 Diagrama de Clases

En este sub-epígrafe se hace alusión del diagrama de clases para el software que se desarrolla, denominado modelo estático, con sus respectivas relaciones entre clases, lo cual permite apreciar los posibles errores que se pudieron haber cometido en la fase de construcción de los diagramas de secuencia. En esta etapa de construcción es donde se refinan (en caso de que existan) los errores cometidos, pero esto no significa que estas sean las clases definitorias de la aplicación, debido a que ICONIX es un proceso iterativo e incremental.

En el caso del software para la determinación del régimen de marcha y capacidad de carga del conjunto tractivo se implementa una clase de servicios (controlador) para cada una de las entidades, esto se debe a que cada una posee características propias (atributos) por lo que para realizar algún cambio sobre estas solo habría que referirse a las propiedades especificas del objeto a modificar, además facilita considerablemente el mantenimiento del software que es un aspecto muy importante a tener en cuenta.

En este documento no se hace constancia visual de este diagrama debido a que se implementan muchas clases y no se distinguen bien todas estas clases en una figura, pero sí se cuenta con el mismo en el proyecto realizado con la herramienta case EA (formato digital).

2.7.3 Diagrama de clases persistentes.

A continuación se muestra en la figura 2.10 el diagrama de clases persistentes correspondiente al software que se implementa, este se hace tomando como referencia el diagrama de de clases expuesto anteriormente.

Este modelo representa las entidades (objetos) que van a ser persistidas en la base de datos, sobre las cuales se realizarán operaciones, ya sea la inserción, eliminación, el proceso de búsqueda, el cambio de los atributos o campos que representan las propiedades de los objetos.

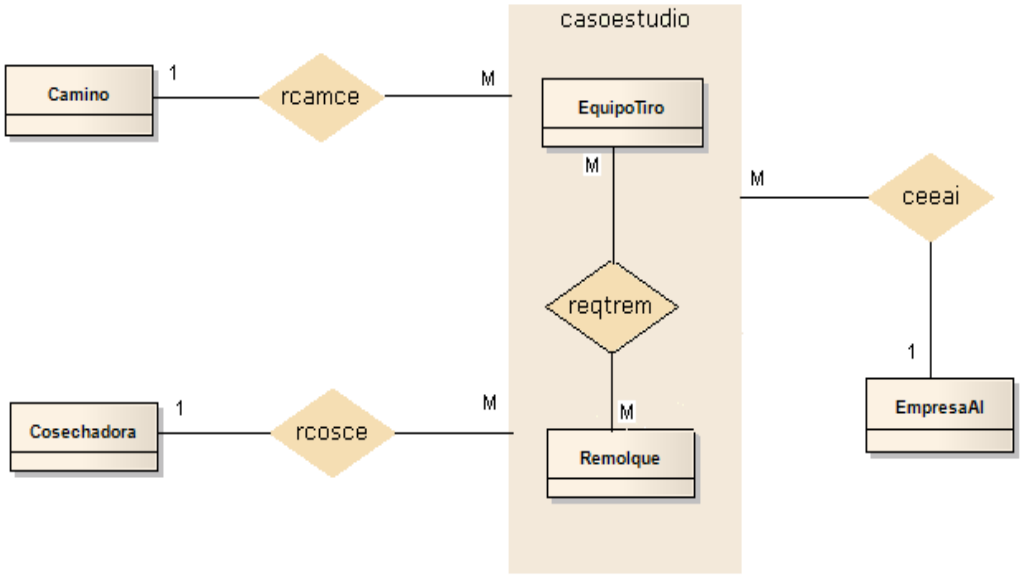


Figura 2. 10 Diagrama de clases persistentes para el software de determinación de capacidad de carga y régimen de marcha

2.8 Valoración de sostenibilidad.

La confección de un Producto Informático (PI) bien sea una aplicación Web o Desktop trae consigo resultados positivos y/o negativos en el entorno del PI, es por eso que se hace necesario proceder a la valoración de sostenibilidad del mismo. Este trabajo adoptará el procedimiento que plantea que la misma debe realizarse a través de cuatro dimensiones de gestión de sostenibilidad: administrativa, socio-humanista, ambiental y tecnológica.

La valoración de sostenibilidad del PI es el proceso de evaluación de los impactos antes mencionados, previsibles desde el diseño del proyecto, que favorece su autorregulación para la satisfacción de la necesidad que resuelve, con un uso racional de recursos y la toma de decisiones adecuadas a las condiciones del contexto y el cliente [CONCEPCIÓN09].

En la presente investigación se abordarán exhaustivamente las dimensiones atendiendo los aspectos de cada una para determinar si el PI es sostenible y perdurable en el tiempo.

2.8.1 Dimensión administrativa

Esta dimensión incluye aspectos como: ahorro, gastos, calidad de la producción y los servicios, administración de recursos, toma de decisiones administrativas.

Para la determinación de los costos se emplea el Modelo Constructivo de Costos (COCOMO II, por sus siglas en inglés), el cual permite estimar el costo asociado al desarrollo de un software.

Puntos de función desajustados

Con el objetivo de calcular los puntos de función desajustados (UFP, por sus siglas en inglés), a continuación se presentan las características del software para la determinación del régimen de marcha y capacidad de carga del conjunto tractivo representadas mediante diferentes tablas, estas aspectos a tener en cuenta son las entradas externas, las salidas externas, los ficheros lógicos internos, ficheros lógicos externos y las consulta externas.

En el caso del software propuesto no se valoran los ficheros lógicos externos y las consulta externas porque no se utiliza otra base de datos ajena a la que se implementa para el mismo.

Entradas Externas (EI): son todas aquellas entradas que le son proporcionadas al sistema.

Nombre	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación
Insertar Equipo de Tiro	1	12	Bajo
Modificar Equipo de Tiro	1	11	Medio
Eliminar Equipo de Tiro	1	12	Medio
Insertar Cosechadora	1	4	Bajo
Modificar Cosechadora	1	3	Bajo
Eliminar Cosechadora	1	4	Bajo
Insertar Camino	1	6	Bajo
Modificar Camino	1	5	Bajo
Eliminar Camino	1	6	Bajo

Guardar Caso de estudio	5	7	Alto
Insertar datos Empresa	1	16	Medio

Tabla 2. 3 Entradas externas del software

Salidas externas (EO): Salidas asociadas al sistema que tiene elementos de filtraje de información.

Nombre	Cantidad de Ficheros	Cantidad de elementos de datos	Clasificación
Mostrar Equipos de Tiro	1	12	**Bajo**
Mostrar Cosechadoras	1	4	**Bajo**
Mostrar Caminos	1	6	**Bajo**
Mostrar Casos de estudio	5	7	**Alto**

Tabla 2. 4 Salidas externas del software

Ficheros lógicos internos (ILF): son los grupos de datos relacionados lógicamente que permiten el almacenamiento de la información perteneciente al sistema.

Nombre	Cantidad de Ficheros	Cantidad de elementos de datos	Clasificación
tb_ Equipotiro	1	12	**Bajo**
tb_Cosechadoras	1	4	**Bajo**
tb_Caminos	1	6	**Bajo**
tb_Casoestudio	5	7	**Alto**
Tb_Empagroindust	1	5	**Bajo**

Tabla 2. 5 Ficheros Lógicos Internos del software

Para cada una de las características del sistema mostradas en las tablas anteriores, se cuenta la cantidad de funcionalidades clasificadas en cada Nivel

de complejidad (Bajo, Medio, Alto) y se multiplica por el peso asociado en la tabla. Luego todos estos productos se suman y se obtiene la cantidad de puntos de función desajustados como se observa en el anexo 17.

Cálculo de las instrucciones fuentes

Para el cálculo de las instrucciones fuentes (SLOC), COCOMO II se basa en la cantidad de instrucciones por punto de función que genera el lenguaje de programación empleado. Utilizando la siguiente fórmula se obtiene:

SLOC = UFP * ratio

Características de las Instrucciones fuentes	
Características	**Valor**
Puntos de función desajustados	**99**
Lenguaje java	**100 %**
Ratio del lenguaje	**Java 53**
Instrucciones * lenguaje	**Java 5247**
Instrucciones fuentes (SLOC, en miles líneas de código)	**5,24**
Reducción de un 40 % (Reutilización de código)	**3,14**

Tabla 2. 6 Calculo de las instrucciones fuentes

Cálculo de esfuerzo, tiempo, cantidad de hombres y costo

Los multiplicadores de esfuerzo (ME), que se muestran en la anexo 18 representan las características del proyecto y expresan su impacto en el desarrollo total del producto de software. Para determinar el esfuerzo asociado al desarrollo del sistema, denominado PM, se utilizan los multiplicadores de esfuerzo, los factores de escala, así como los valores constantes A, B, C, D. Los factores de escala y los valores constantes se observan en los anexos 19 y 20 respectivamente.

Cálculo del esfuerzo

El esfuerzo de desarrollo está representado por las siglas PM y se expresa en hombres por mes. La fórmula para obtenerlo se muestra a continuación:

$PM = A*Size^{E} \prod_{i=1}^{n} EMi$ Donde $E = B + 0.01 * \sum_{j=1}^{5} SFj$

E = 1,0668

PM ≈ 19 hombres por mes.

Cálculo del tiempo de desarrollo (TDEV)

El tiempo de desarrollo se obtuvo a partir de la fórmula:

$TDEV = C * PM^{F}$ Donde $F = D+022 * (E-B)$

TDEV ≈ **9** meses.

Cálculo de cantidad de hombres estimado (CH)

La cantidad de hombres es el resultado de la división del esfuerzo y el tiempo estimado de desarrollo:

$CH = PM / TDEV$

CH ≈ **2** hombres.

Costo total (CT)

El costo total es el resultado de multiplicar el costo de hombres por mes (CHM) y el tiempo de desarrollo. En este caso se va a asumir el salario promedio de un trabajador para determinar el costo de hombres por mes, por tanto:

SP = **$250** Salario Promedio

CHM = cantidad de hombres * SP

CHM = **$ 500** Costo de hombres por mes para **2** trabajadores

Costo total:

$CT = CHM * TDEV$

CT ≈ $ 4823

En el caso de la confección de esta aplicación no se incurrirá en un alto costo por mano de obra pues el desarrollador del PI es un estudiante universitario que opta por el titulo de Ing. Informático.

No se añadirán gastos por concepto de equipamiento pues el producto es una aplicación desktop y podrá ser ejecutado sobre cualquier máquina computadora existente en la entidad.

No generará ingresos directamente a la entidad a fin ya que su funcionalidad principal será como herramienta de apoyo para los especialistas.

No se incurrirá en gastos por concepto de pago de licencia porque el producto se desarrolla con herramientas tecnológicas de software libre y las cuales son libres de pago.

2.8.2 Dimensión socio-humanista

En esta dimensión se analiza el PI según aspectos como: formación ético humanista, modo de vida, desarrollo de un grupo social, satisfacción de las necesidades sociales, la ciencia y la tecnología como procesos sociales.

Con respecto al impacto socio-humanista del PI, se puede asegurar que una vez implantado, agilizará considerablemente el proceso de determinación del régimen de marcha y capacidad del conjunto tractivo, por lo cual no será extenso el intercambio máquina computadora-usuario y no aumentará los daños a la vista y a la columna vertebral provocados por un prolongado tiempo de trabajo sin las condiciones mínimas necesarias.

La implantación del nuevo PI no trae consigo la generación o disminución de empleo en la entidad, su influencia está orientada, por el contrario, a facilitar el trabajo de los especialistas que ya son empleados por la entidad.

A partir de lo expuesto anteriormente, se arriba a que el software es sostenible en cuanto a la dimensión socio-humanista.

2.8.3 Dimensión ambiental

Esta dimensión analiza al PI según: las condiciones favorables o no y si minimiza daños e impactos a las personas o cosas.

El diseño de las interfaces del PI será lo más amigable posible y de fácil uso por parte de los usuarios a fin. Se utilizarán colores tenues adecuados y un tamaño de letra que no dañen ni canse la vista para lograr una mayor identificación del usuario con la aplicación.

La solución propuesta en el presente trabajo no tendrá un impacto directo sobre el medio ambiente, pues disminuirá el consumo de productos de ofimáticos (papeles, lápices, entre otros) que provienen de la naturaleza.

Las tecnologías que se utilizarán en su desarrollo son bajo condiciones Open Source (OS por sus siglas en inglés, código abierto), esto contribuye a que se puedan reutilizar códigos existentes en el mundo y que sirvan para disminuir el tiempo de desarrollo. Así como el mismo código a generar podrá ser reutilizado con vista a la implementación de otros módulos o nuevos requerimientos del PI.

La implantación del PI aumentará el uso de la computadora pero por breve tiempo de intercambio con la misma pudiéndole causar enfermedades a los usuarios, por lo que se recomienda una postura correcta de los usuarios en las sillas, así como tener el monitor a la altura de los ojos, utilizar protector de pantalla y una iluminación adecuada en el local.

Atendiendo a lo antes expuesto se concluye que el PI es sostenible en la dimensión Ambiental.

2.8.4 Dimensión tecnológica

En esta dimensión se analiza el PI según: uso de la tecnología adecuada y si el producto es asimilable por el usuario.

Para la utilización de esta aplicación no será necesaria la capacitación de los usuarios que interactuarán con él, debido a que será una herramienta fácil de usar y su contexto será completamente entendible para los especialistas a fin, atendiendo los requerimientos del cliente, aunque se adjuntará al producto su respectivo manual de ayuda.

La entidad cuenta con la estructura que le permitirá la implantación y correcta explotación del PI. Esta herramienta será portátil lo cual permitirá que se pueda ejecutar en cualquier maquina computadora con la simple restricción de que tenga instalada la Máquina Virtual de Java (JRE v1.6 o superior).

El producto brindará facilidades para la correcta explotación por parte del usuario, con independencia del productor, pues el usuario tendrá los privilegios para actualizar la información según el rol que tenga en la aplicación.

Teniendo en cuenta los aspectos analizados en cuanto a la dimensión tecnológica se concluye que el PI es sostenible en la misma.

2.9 Implementación

El objetivo principal de esta etapa es implementar todo el código fuente para construir el software propuesto, para lo cual se procede a integrar los requerimientos solicitados en el análisis y plasmado en el diseño, para validar que el producto obtenido satisface los requerimientos definidos previamente.

Para el software propuesto se implementaron clases entidades, clases controladoras, clases de cálculos, clases de estructuras y clases de apariencia, relacionándose entre si para gestionar los datos de la base de datos.

2.9.1 Prueba

La metodología ICONIX plantea que el proceso de prueba se debería comenzar mucho antes que el de codificación, incluso recomienda que se tenga en cuenta en la etapa de análisis identificando los casos de prueba a partir de los diagramas de robustez, de esta forma es posible eliminar una gran cantidad de errores (inclusive antes de que existan) . Es por ello que probar un sistema debe verse como una parte importante dentro del ciclo iterativo e incremental de desarrollo, que verifica que el producto cumple con el propósito específico para el que fue creado [ROSENBERG07]. Teniendo en cuenta lo antes expresado se desarrollaron comprobaciones que permitieron probar de manera sistémica que el comportamiento del software descrito en cada caso de uso está implementado correctamente, básicamente las funciones identificadas durante el análisis de robustez. Estas pruebas se realizaron considerando en todo momento que los requerimientos estuvieran íntimamente relacionados con las mismas, se trató que al menos hubiera dos casos prueba para verificar cada requerimiento.

Además se hizo una clase de prueba con datos reales para comprobar si los resultados arrojados por el software eran los esperados.

Los problemas detectados como resultado de las pruebas realizadas se tomaron en cuenta y a la vez fueron corregidos en iteraciones posteriores.

2.10 Conclusiones

En este capítulo se hizo un análisis detallado de la solución propuesta, así como el diseño de la misma guiado por la metodología de desarrollo ICONIX.

Se valoró la sostenibilidad del software para determinar el régimen de marcha y capacidad de carga del conjunto tractivo en las cuatro dimensiones: administrativa, socio-humanista, ambiental y tecnológica, llegando a la conclusión que la solución empleada es factible y el sistema es sostenible. El producto será perdurable en el tiempo ofreciendo una herramienta informática que les permita gestionar de una forma eficaz los datos a procesar, arrojando un resultado que oriente a los especialistas sobre la conformación idónea del conjunto tractivo.

CONCLUSIONES

Con el desarrollo del sistema propuesto, dedicado a lograr una aplicación que permita de modo eficaz y sencillo la selección de los valores óptimos del régimen de marcha y capacidad de carga del "Conjunto Tractivo", se cumple con el objetivo de esta investigación. Muestra de esto son las siguientes conclusiones.

- Elegir el método de exploración resultó el adecuado para optimizar el modelo matemático planteado.
- La elección de las tecnologías y la arquitectura de software empleadas en el software propuesto resultaron una solución efectiva.
- El uso de una metodología ágil como ICONIX para guiar el análisis, diseño y desarrollo de la aplicación resultó eficiente.
- El estudio de valoración de sostenibilidad del producto informático arrojó que el desarrollo del software es factible y que según su impacto social, económico, tecnológico y ambiental el mismo es perdurable en el tiempo.
- Como resultado se obtiene un producto informático que satisface los requerimientos del cliente.

RECOMENDACIONES

Con los resultados obtenidos hasta el momento se procede a brindar una serie de recomendaciones, que proporcionarán ampliación, modificación, mejora y construcción en una futura versión de la propuesta presentada en este trabajo, como:

- Se recomienda agregarle otras funcionalidades al software como:
  - Determinar los valores óptimos de la configuración de la zona de carga del remolque cañero.
  - Graficar los resultados obtenidos en cuanto a los indicadores de eficiencia.
- Mejorar la apariencia del software respecto al diseño con iconos sugerentes al tema.

BIBLIOGRAFÍA

01. **[ACTION07]** Action, E.I., *Spring and EJB 3: All-Star Team or Neighbors with Good Fences.* 2007.
02. **[ÁLVAREZ, 1993]** Álvarez Sánchez, V., *Informe de Pruebas del Remolque RC-03.*, H.d.d. Julio., Editor. 1993.
03. **[ÁLVAREZ10]** Sánchez, V. Á., *Determinación de la capacidad de carga de los remolques cañeros y regímenes de marcha óptimos en el transporte de la cosecha cañera.* Tesis presentada en opción al grado de Doctor en Ciencias Técnicas, 2010, Universidad de Holguín: Holguín.
04. **[APACHE08]** Apache, *OpenJPA User's Guide*. 2008. 338.
05. **[BUYLLA81]** Álvarez Buylla, M., *Modelo de Simulación para el Análisis del tiro Partido en Cosecha de Caña de Azúcar.* Revista Centro Azúcar, 1981.
06. **[CANÓS]** Canós, J. H., P.L.y.M.C.P., *Métodologías Ágiles en el Desarrollo de Software.* p. 8.
07. **[CARDONE, 1985]** Cardone, M.I., *La Organización Racional del Transporte Cañero.*, in *Revista ATAC.* 1985: La Habana.
08. **[CASTILLO04]** Castillo, J.M., *Persistencia de objetos. JDO, solución java.*, in *Facultad Informática Murcia.* 2004, Universidad de Murcia: Ciudad de Murcia.
09. **[CELKO99]** Celko, J., *Joe Celko's Data and Databases: Concepts in Practice.* 1999: Morgan Kaufmann Publishers.
10. **[CHURCHER07]** Churcher, C., *Beginning Database Design From Novice to Professional.* 2007. 267.
11. **[CHURCHER08]** Churcher, C., *Beginning SQL Queries From Novice to Professional:Apress.* 2008. 240.
12. **[CONCEPCIÓN09]** Concepción, R., *Procedimiento para la valoración de sostenibilidad de un producto informático.* 2009, Universidad de Holguín, Cuba.
13. **[COOPER00]** Cooper, J.W., *Java™ Design Patterns: A Tutorial.* 2000: Addison Wesley. 352.

14. **[CORNELL08]** Cornell, C.S.H.G., *Core Java™ Volume II–Advanced Features, Eighth Edition.* Eighth Edition ed. Vol. Volume II. 2008: Prentice Hall. 1056.
15. **[CRUME06]** Crume, K.M.a.C.Z.w.J.L.W.a.J., *Beginning Java EE 5 From Novice to Professional.* 2006. 673.
16. **[DARWIN04]** Darwin, I.F., *Java Cookbook, 2nd Edition.* 2nd Edition ed ed. 2004: O'Reilly. 862.
17. **[DEBUPANDA07]** Debu Panda, R.R., Derek Lane, *EJB 3 in Action.* 2007.
18. **[DEITEL04]** Deitel ,H. M. - Deitel & Associates, I., P. J. Deitel - Deitel & Associates, Inc., *Java™ How to Program, Sixth Edition.* Sixth Edition. ed. 2004. 1568.
19. **[DERBY09]** Derby, A., *DerbyAdmin 1.0.* 2009.
20. **[DMICHIEL06]** Linda DeMichiel, M.K., *Java™ Persistence API.* 2006. 56.
21. **[DMICHIEL07]** DeMichiel, L., *Java™ Persistence 2.0.* 2007. 60.
22. **[DONALD03]** Donald K. Burleson, J.C., John Paul Cook, Peter Gulutzan, Trudy Pelzer., *Advanced SQL Database Programmers Handbook.* 2003.
23. **[Dominico, 1990]** Dominico, D.L.C.R., *Evaluación de Diferentes Variantes en la Transportación de la Caña de Azúcar.*, in *Revista ATAC.* 1990: La Habana.
24. **[ESCUDERO, 1978]** Escudero, L.F., *Programación no-lineal con restricciones. Algoritmos Aplicables (I).* 1978.
25. **[FRIESEN07]** Friesen, J., *Beginning Java™ SE 6 Platform: From Novice to Professional.* 2007. 511.
26. **[FISHER03]** Maydene Fisher, J.E., Jonathan Bruce, *JDBC™ API Tutorial and Reference, Third Edition.* Third Edition ed. 2003: Addison Wesley.
27. **[FOUNDATIONM08]** Foundation, T.A.S., *Derby Reference Manual.* 2008.
28. **[FOUNDATIONDG08]** Foundation, T.A.S., *Derby Developer's Guide.* 2008.
29. **[FOUNDATIONTUG08]** Foundation, T.A.S., *Derby Tools and Utilities Guide.* 2008.
30. **[GOFICIAL, 1998]** Oficial, G., *Bases del Perfeccionamiento Empresarial.* 1998: La Habana.
31. **[GLÖGLS07]** Glögls, M., *Persistencia de Objetos Java: El Camino hacia Hibernate.* 2007. 47.

32. **[HERBERT03]** Herbert Schildt, J.H., *The Art of Java*. 2003: McGraw-Hill. 385.
33. **[HILLEGAS06]** Hillegas, R., *Java in the Database.* 2006.
34. **[HORTON05]** Horton, I., *Ivor Horton's Beginning Java™ 2, JDK™ 5 Edition.* 2005: Wiley Publishing, Inc. 1501.
35. **[IGLESIAS]** Iglesias C, C., *Fundamentos para la modelación de la masa y costo de los remolques cañeros en función de su capacidad de carga*.
36. **[INYOUNG06]** Inyoung Cho, T.C., Charles Ditzel,Tim Boudreau., *Twelve Reasons To Use NetBeans™ Software—Episode 2, in 2006 Java One Conference*. 2006: Sun Microsystems, Inc.
37. **[JROBOSTOV, 1977]** Jróbostov, S.N., *Explotación del Parque de Tractores y Máquinas.* 1977, Editorial Mir.: Moscú.
38. **[LANS06]** Lans, R.F.v.d., *Introduction to SQL: Mastering the Relational Database Language, Fourth Edition/20th Anniversary Edition*. 2006: Addison Wesley Professional. 1056.
39. **[MCCUNE06]** McCune, B., *Exploring the Java Persistence API.* 2006. 49.
40. **[MATT07]** Matt, D.R.S., *Use Case Driven Object Modeling with UML: Theory and Practice*. 2007. 400.
41. **[MULLER99]** Muller, R.J., *Database Design for Smarties: Using UML for Data Modeling.* 1999: Morgan Kaufmann Publishers. .
42. **[NACIONAL, 1992]** Nacional, D.d.T.A.M., *Algunas Consideraciones Sobre el Transporte Automotor Utilizados en Cuba.* 1992.
43. **[PATEL99]** Patel, P., *Java Database Programming with JDBC.* . 1999: The Coriolis Group. 209.
44. **[ORACLE10]** Oracle. *JPA* [cited (21/3/2010).]; Available from: http://www.oracle.com/technology/products/ias/toplink/jpa/index.html.
45. **[RESOLUCION]** Anónimo, *Resolución Económica V Congreso del PCC. Periódico Granma*.20. [POSTGRESQL05] A., E.Q., INTRODUCCION A POSTGRESQL. 2005.
46. **[RODRIGUEZ, 1985]** RODRIGUEZ, A.M.M.E., *Estudio de las Tecnologías de Transportación de la Caña con Tractores y Carretas.*, in Revista ATAC. 1985: La Habana.

47. **[RODRIGUEZ, 1991]** Rodríguez, P.A.D., *Selección de la Mejor Alternativa de Transportación.* Ministerio del Azúcar., 1991.
48. **[ROSENBERG]** Rosenberg D., M.S.A.M.C.-C., *Agile Development with ICONIX Process—People, Process, and Pragmatism.*
49. **[ROSENBERG07]** Rosenberg, D. and M. Stephens, *Use Case Driven Object Modeling with UML. Theory and Practice*. 2007: Apress.
50. **[SCHINCARIOL06]** Schincariol, M.K.M., *Pro EJB 3 Java Persistence API*. 2006. 470.
51. **[SUNAPI10]** SunMicrosystems, I. *API*. [cited (25/02/2010).]; Available from: http://java.sun.com/javaee/5/docs/api/javax/persistence/package-summary.html.
52. **[SUNJDBC10]** SunMicrosystems, I. *JDBC*. [cited 23/03/2010]; Available from: http://developers.sun.com/product/jdbc/drivers2.
53. **[VATKINA06]** Vatkina, M., M.M., Pramod Gopinath, *Java™ Persistence API in 60 Minutes.* 2006. 68.
54. **[ZIKOPOULOS05]** Zikopoulos P. C., G.B., Dan Scott, *Apache Derby—Off to the Races: Includes Details of IBM® Cloudscape.* 2005: IBM Press. 600.

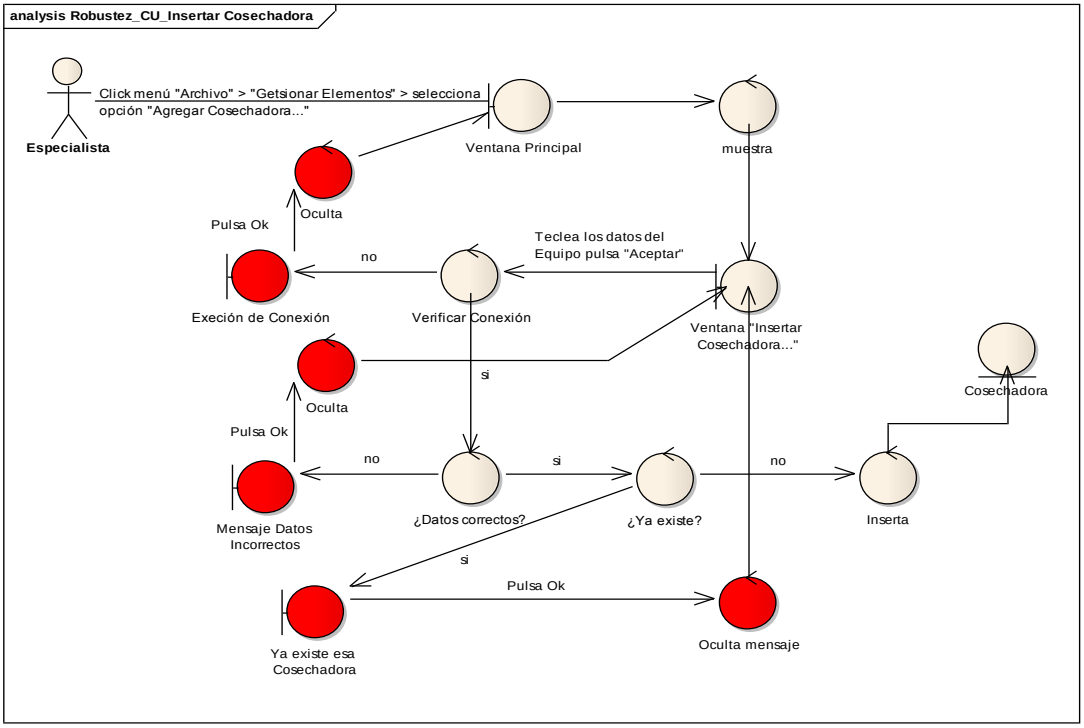
ANEXOS

Caso de Uso	Descripción Textual
Insertar Cosechadora.	Flujo básico: El caso de uso comienza cuando el especialista necesita insertar una nueva cosechadora con sus características en la Base de Datos, el software realiza la operación; efectuando la conexión a la BD embebida, verificando que los datos que se van a insertar son válidos, comprobando que no exista otra cosechadora con el mismo identificador en la BD y finalmente insertando el objeto, el caso de uso termina cuando el objeto es insertado satisfactoriamente. Flujo alterno: Si el software no pudiese realizar la conexión, muestra al desarrollador la razón por la cual no se realizó la misma y termina al caso de uso; si los datos a insertar no son validos le sugiere al usuario que inserte los datos correctamente; si existe otra cosechadora con el mismo identificador en la BD le advierte al usuario que no puede ser insertado la nueva y finaliza el caso de uso.
Modificar Cosechadora	Flujo básico: El caso de uso comienza cuando el especialista necesita modificar algún dato de las cosechadoras existentes en la BD, el software realiza esta operación; verificando que los nuevos datos que se van a insertar son validos. El caso de uso finaliza cuando la aplicación modifica los datos. Flujo alterno: Si los nuevos datos no son validos el software le sugiere al usuario que inserte los datos correctamente y termina el caso de uso.
Eliminar Cosechadora	Flujo básico: El caso de uso comienza cuando el especialista desea eliminar una (o varias) cosechadora (s) que ya no se esté (n) explotando en la actividad de transporte de la cosecha, el software realiza la eliminación del (los) cosechadora (s) seleccionada (s). El caso de uso finaliza

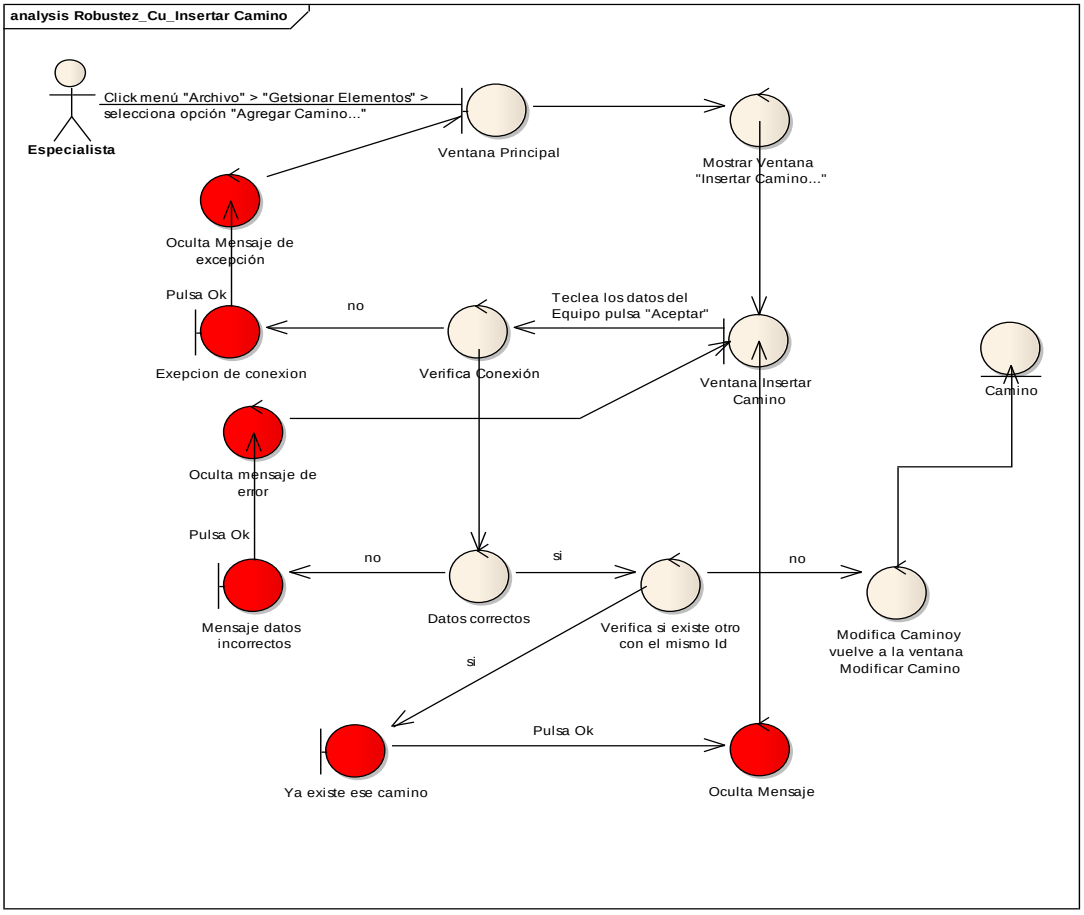
	cuando el software elimina el objeto. Flujo alterno: Si el software no pudiese eliminar un equipo de tiro, se muestra al desarrollador la causa y termina el caso de uso.
Insertar Camino.	Flujo básico: El caso de uso comienza cuando el especialista necesita insertar un nuevo camino con sus características en la Base de Datos, el software realiza la operación; efectuando la conexión a la BD embebida, verificando que los datos que se van a insertar son válidos, comprobando que no exista otro equipo de tiro con el mismo identificador en la BD y finalmente insertando el objeto, el caso de uso termina cuando el objeto es insertado satisfactoriamente. Flujo alterno: Si el software no pudiese realizar la conexión, muestra al desarrollador la razón por la cual no se realizó la misma y termina al caso de uso; si los datos a insertar no son validos le sugiere al usuario que inserte los datos correctamente; si existe otro camino con el mismo identificador en la BD le advierte al usuario que no puede ser insertado el nuevo y finaliza el caso de uso.
Modificar Camino	Flujo básico: El caso de uso comienza cuando el especialista necesita modificar algún dato de los caminos existentes en la BD, el software realiza esta operación; verificando que los nuevos datos que se van a insertar son validos. El caso de uso finaliza cuando la aplicación modifica los datos. Flujo alterno: Si los nuevos datos no son validos el software le sugiere al usuario que inserte los datos correctamente y termina el caso de uso.
Eliminar Camino	Flujo básico: El caso de uso comienza cuando el especialista desea eliminar uno (o varios) camino (s) que ya no se esté (n) explotando en la actividad de transporte de la cosecha, el software realiza la eliminación del (los) equipo (s) de tiro seleccionado (s). El caso de uso finaliza cuando el software borra el objeto.

	Flujo alterno: Si el software no pudiese eliminar un camino, se muestra al desarrollador la causa y termina el caso de uso.
Guardar Caso estudiado	Flujo básico: El caso de uso comienza cuando el especialista desea guardar el caso de estudio analizado con los datos correspondientes a ese caso de estudio, el software realiza la operación. El caso de uso finaliza cuando el software guarda los resultados. Flujo alterno: Si el software no pudiese guardar el caso de estudio, se muestra al especialista la causa y termina el caso de uso.

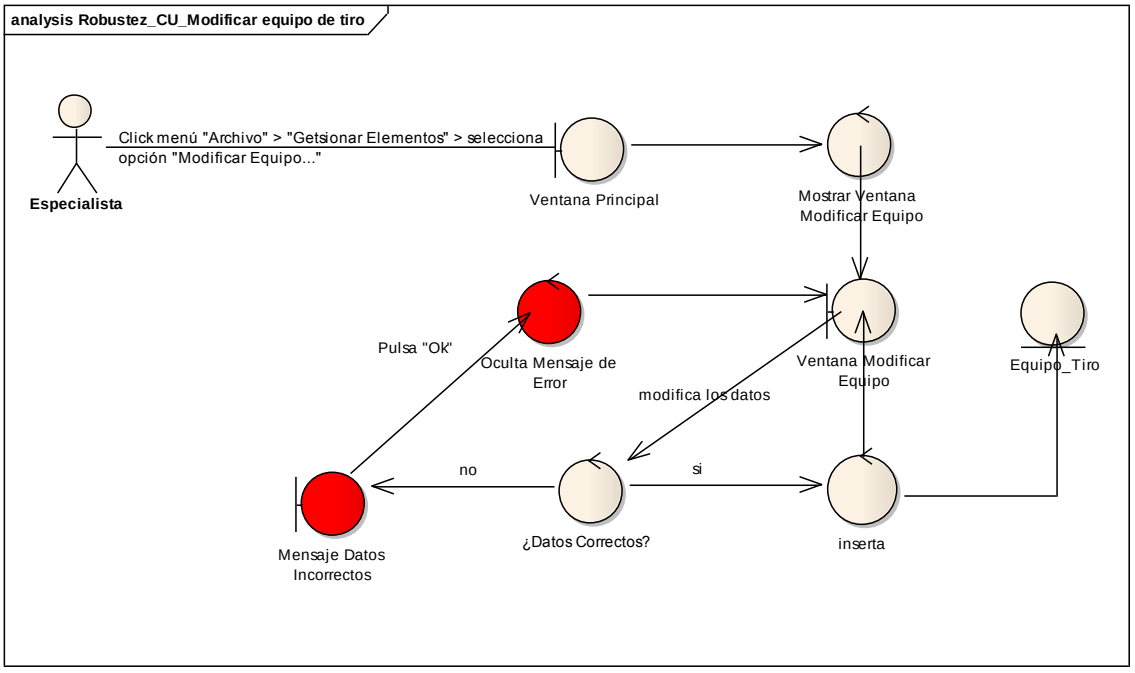
Anexo 1. Descripción de los casos de uso.



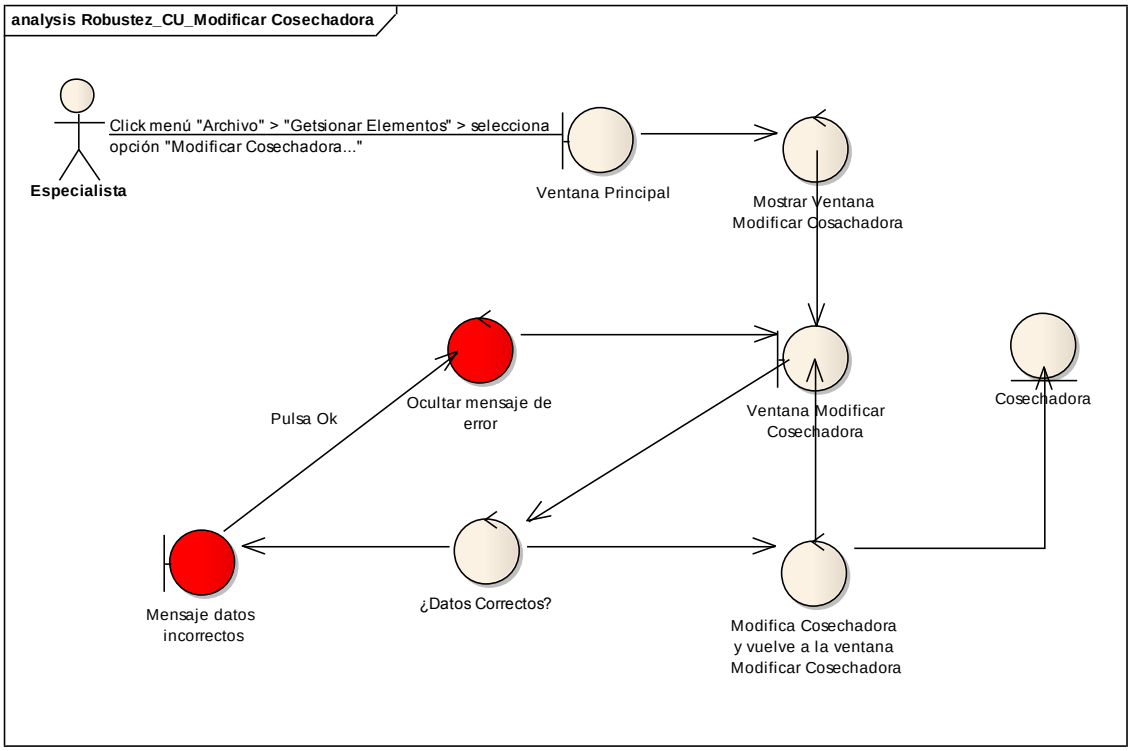
Anexo 2. Diagrama de robustez del caso de uso Insertar Cosechadora.



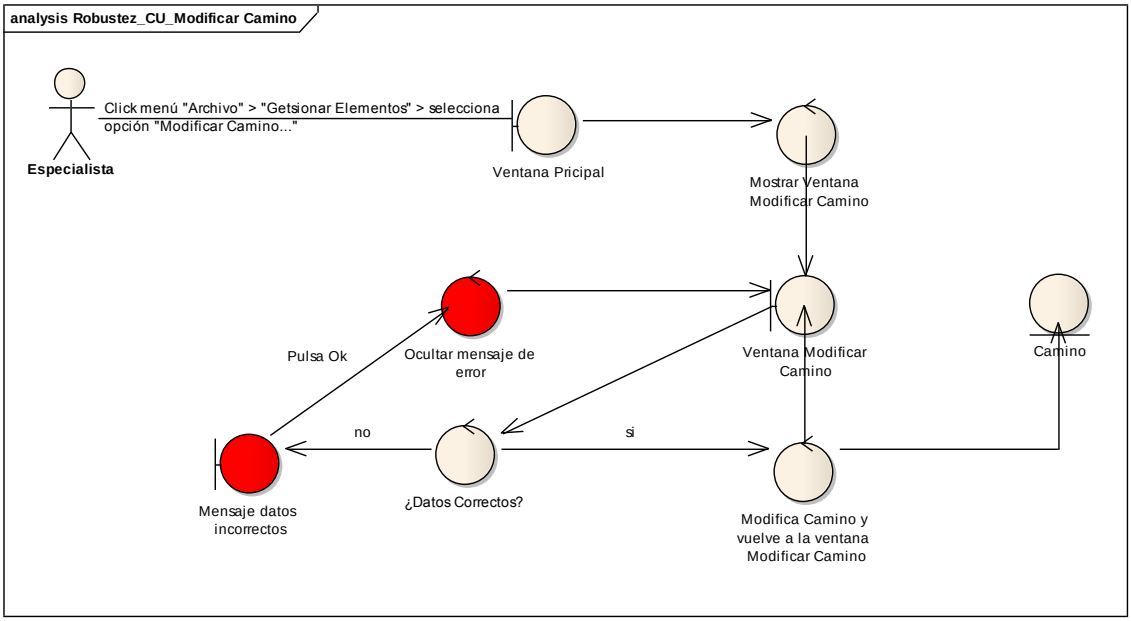
Anexo 3. Diagrama de robustez del caso de uso Insertar Camino.



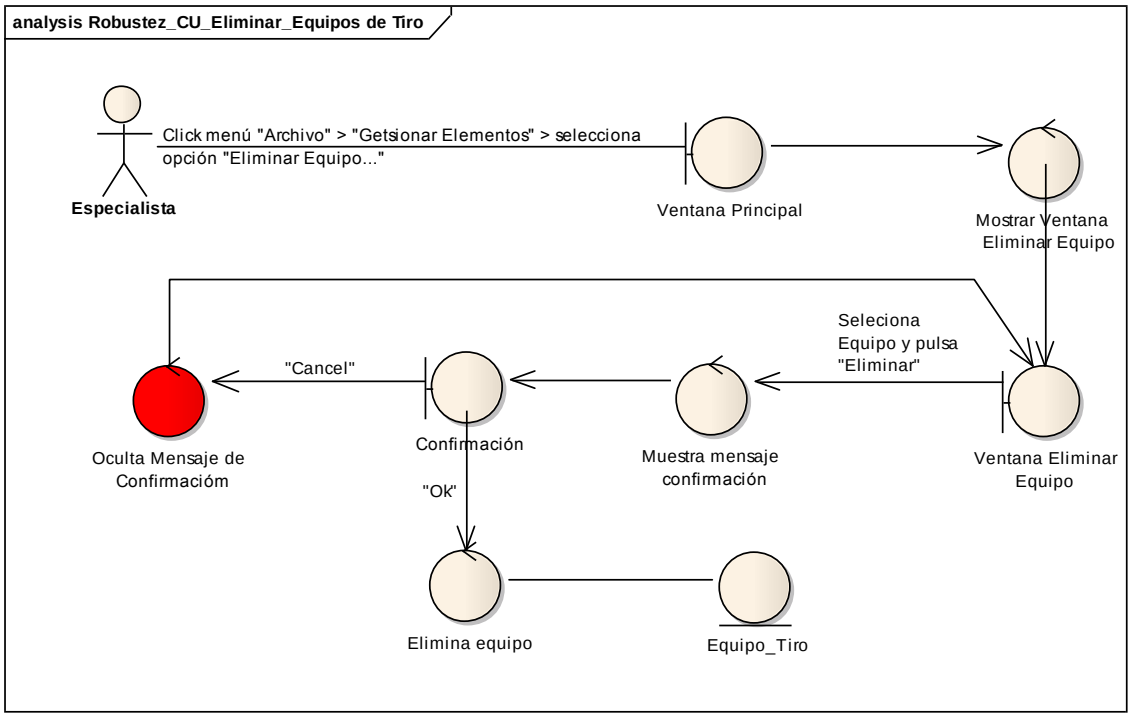
Anexo 4. Diagrama de robustez del caso de uso Modificar Equipo de tiro.



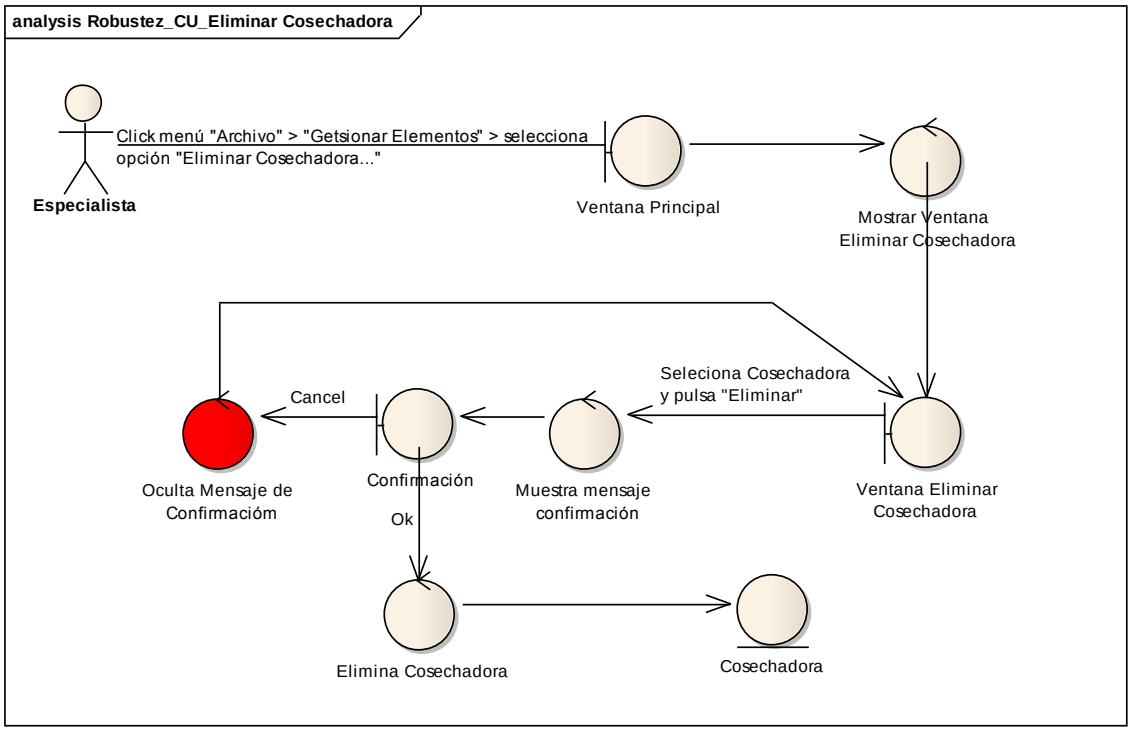
Anexo 5. Diagrama de robustez del caso de uso Modificar Cosechadora.



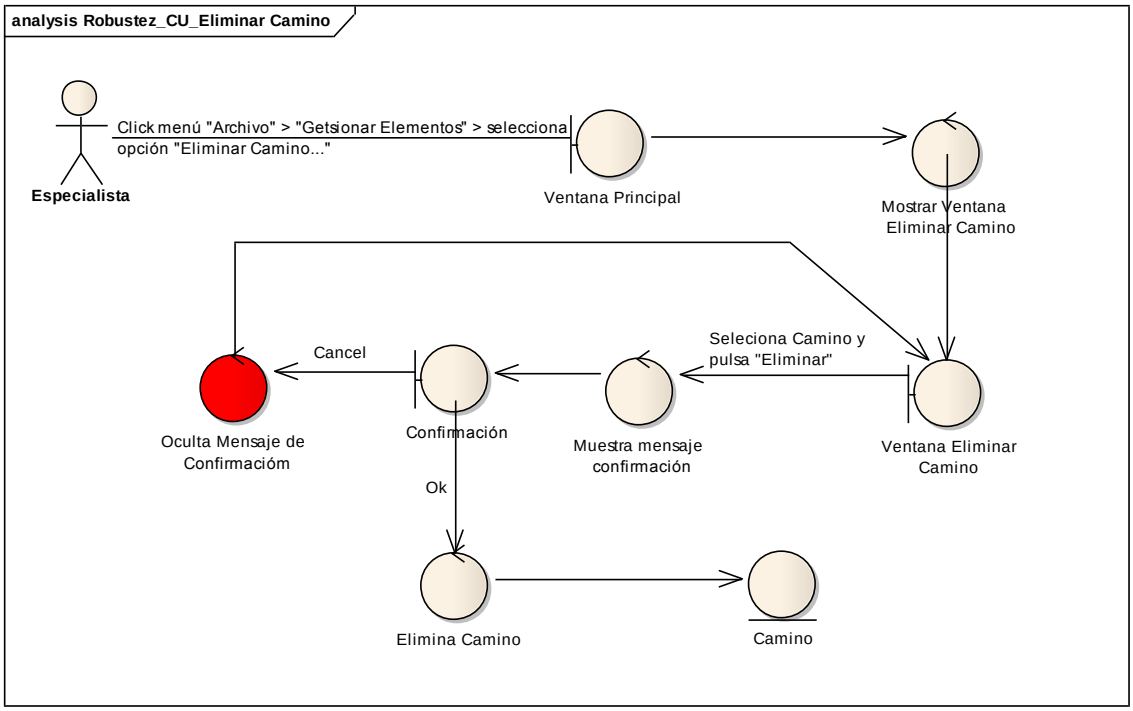
Anexo 6. Diagrama de robustez del caso de uso Modificar Camino.



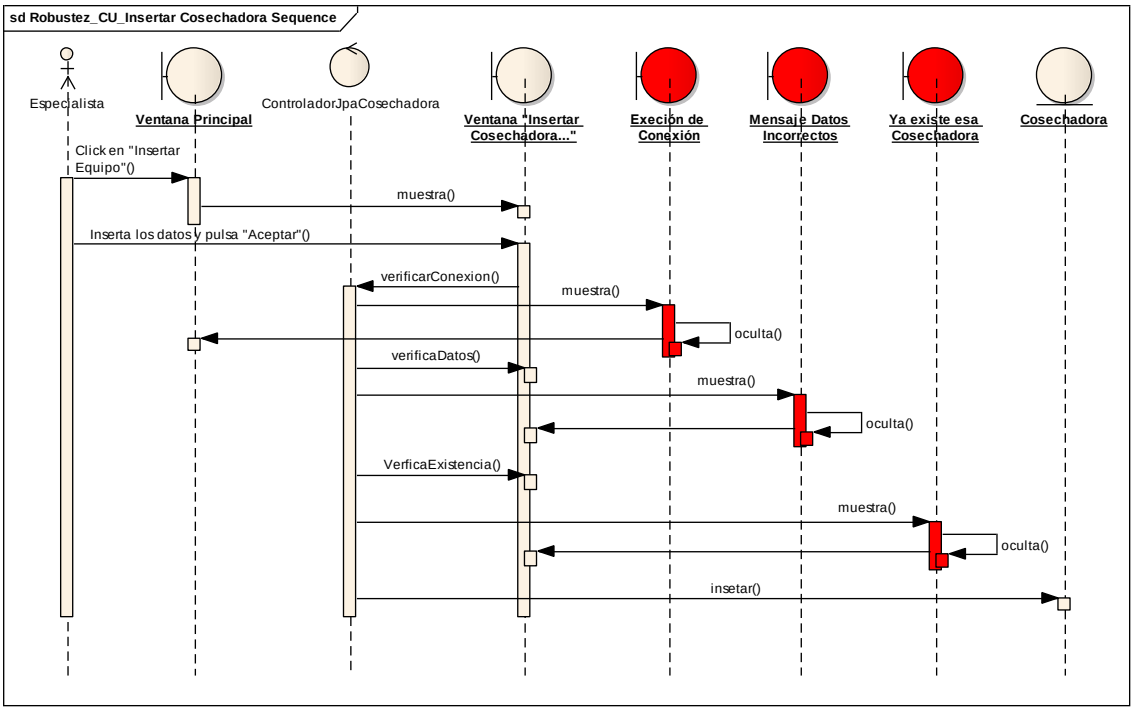
Anexo 7. Diagrama de robustez del caso de uso Eliminar Equipo de tiro.



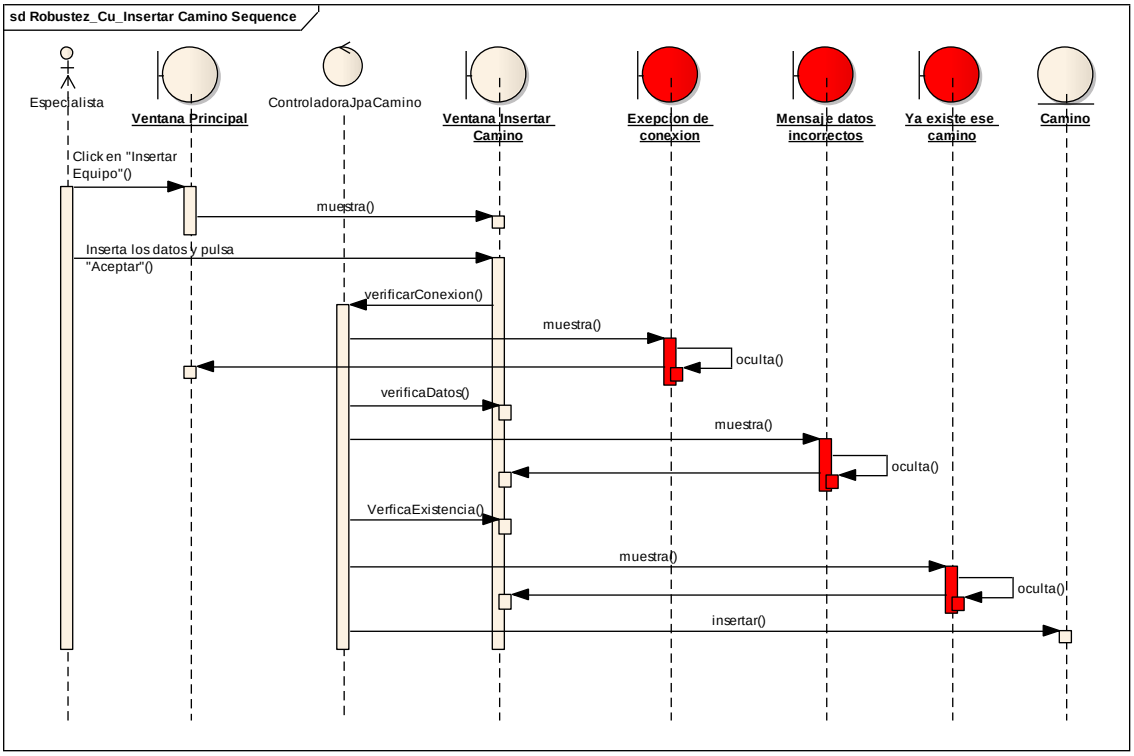
Anexo 8. Diagrama de robustez del caso de uso Eliminar Cosechadora.



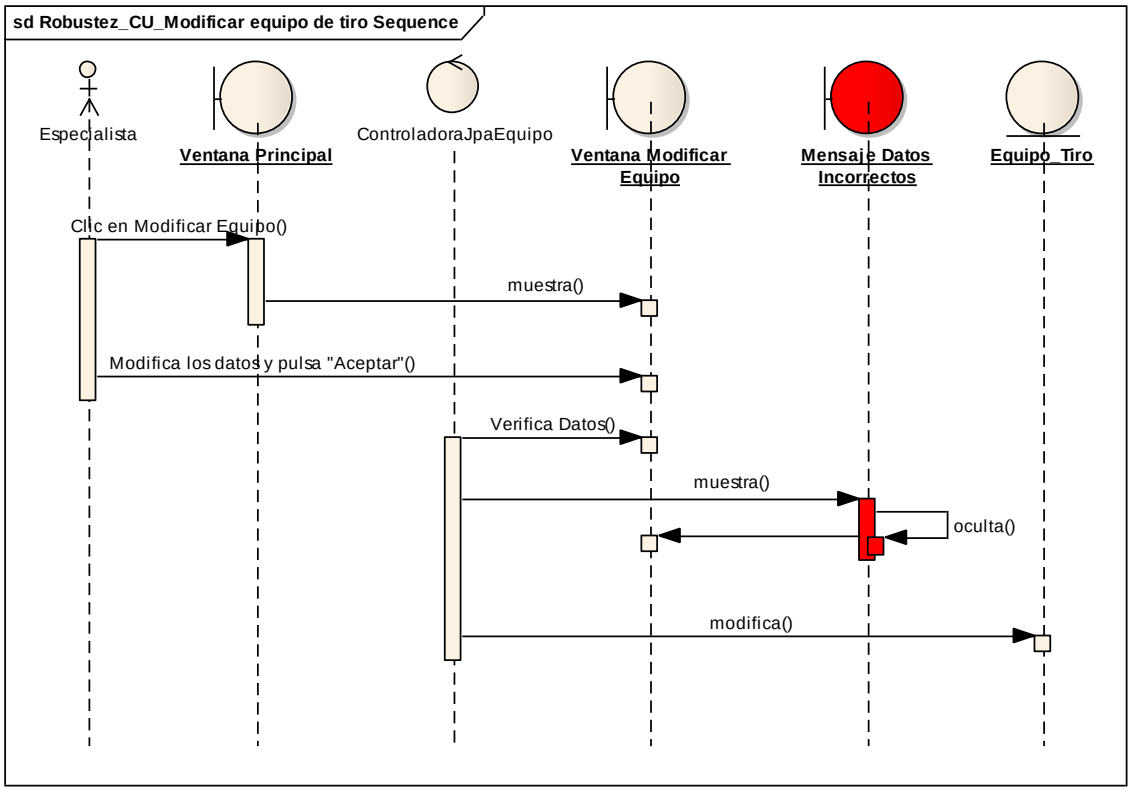
Anexo 8. Diagrama de robustez del caso de uso Eliminar Camino.



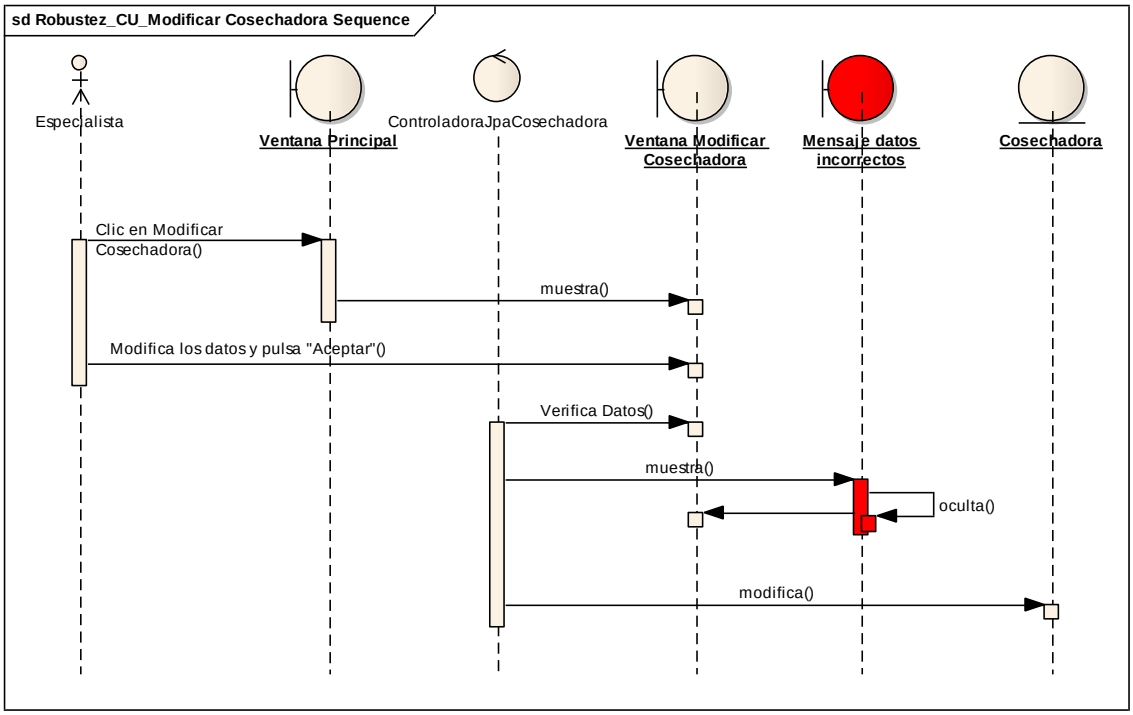
Anexo 9. Diagrama de secuencia del caso de uso Insertar Cosechadora.



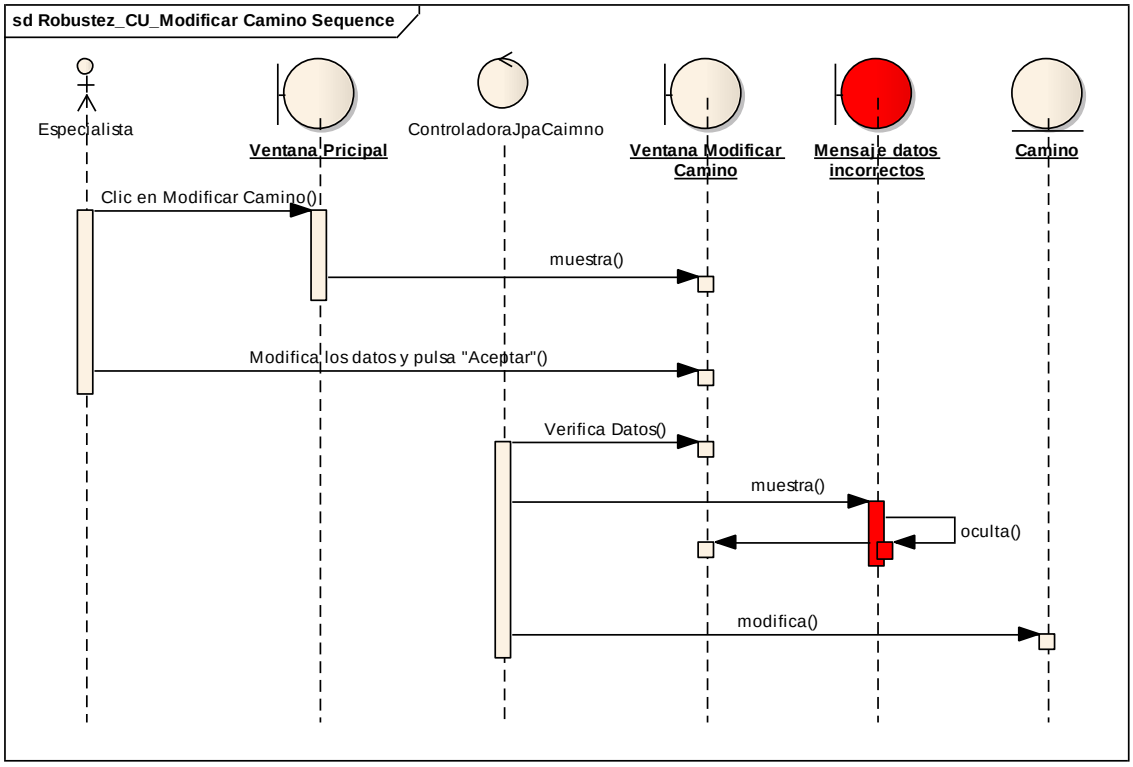
Anexo 10. Diagrama de secuencia del caso de uso Insertar Camino.



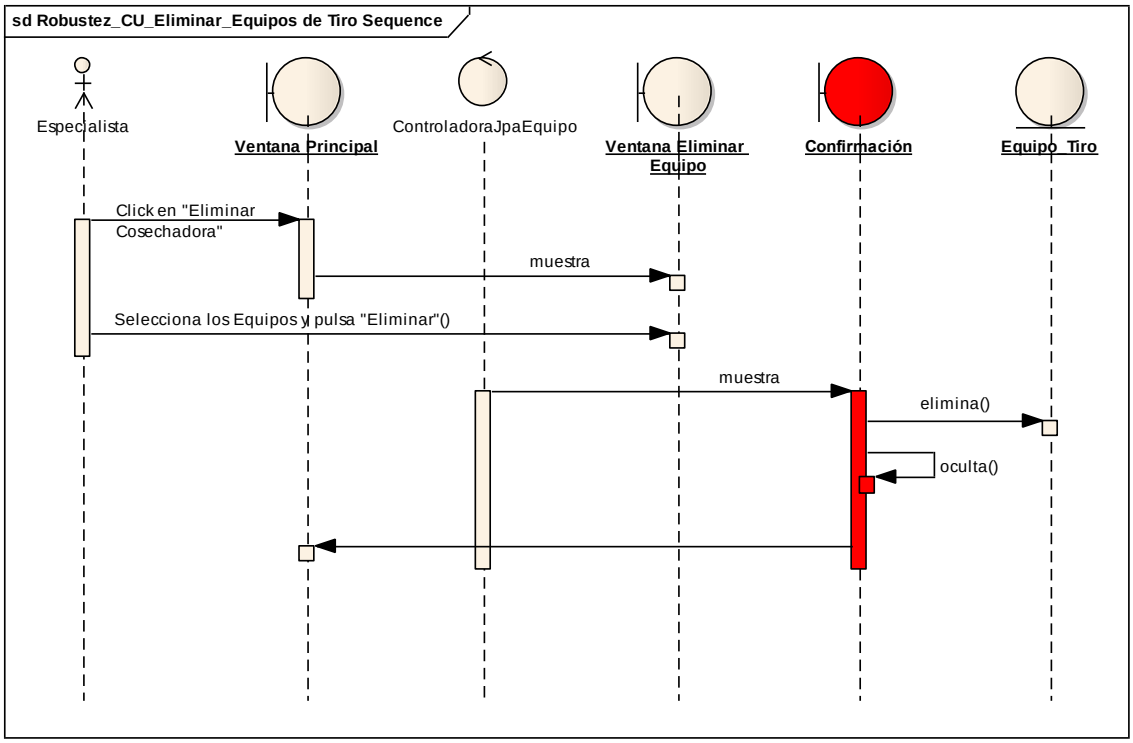
Anexo 11. Diagrama de secuencia del caso de uso Modificar Equipo de tiro.



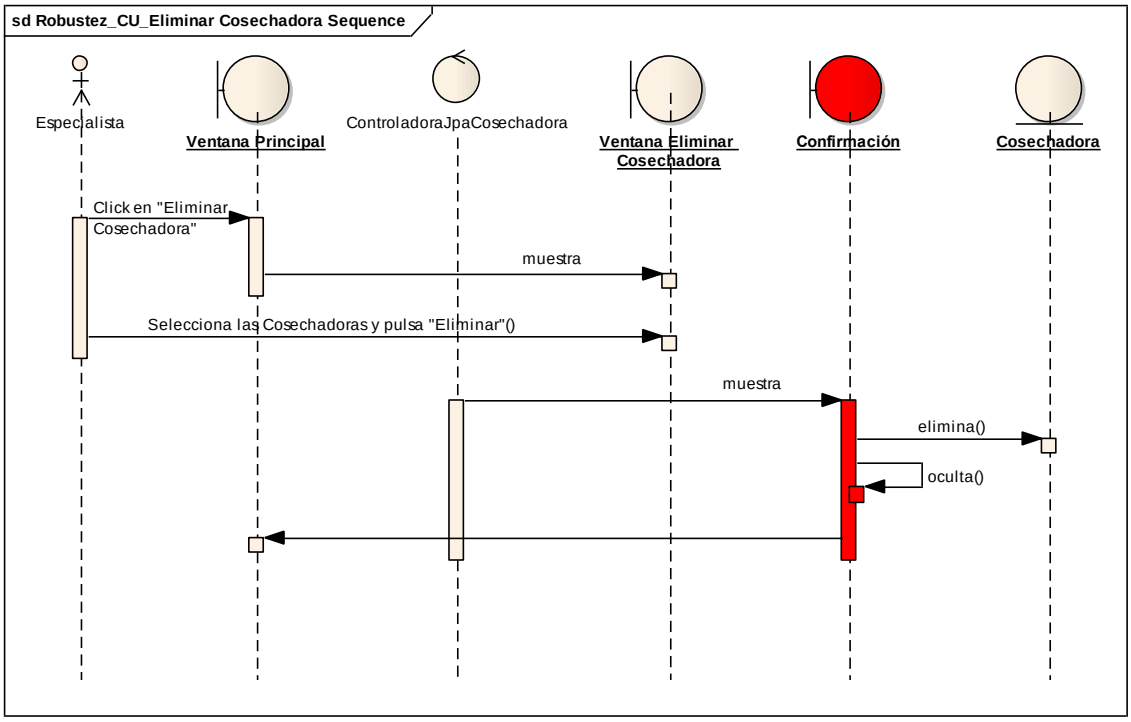
Anexo 12. Diagrama de secuencia del caso de uso Modificar Cosechadora.



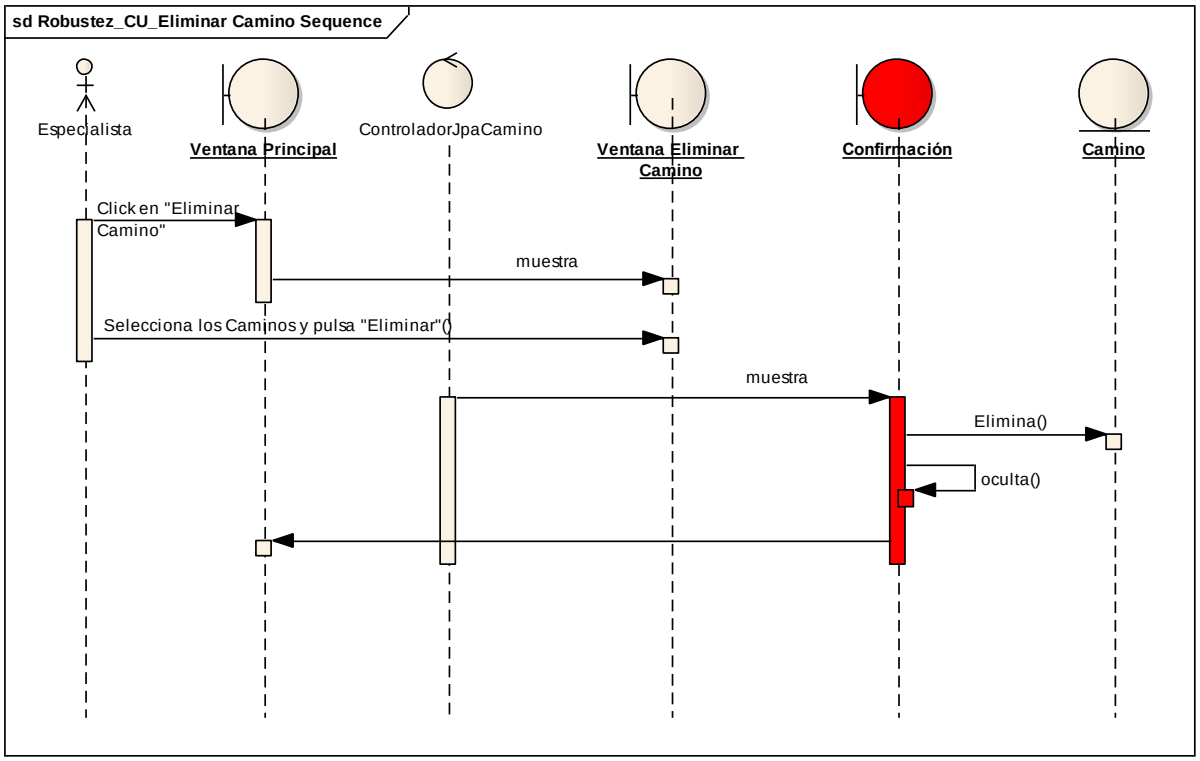
Anexo 13. Diagrama de secuencia del caso de uso Modificar Camino.



Anexo 14. Diagrama de secuencia del caso de uso Eliminar Equipo de tiro.



Anexo 15. Diagrama de secuencia del caso de uso Eliminar Cosechadora.



Anexo 16. Diagrama de secuencia del caso de uso Eliminar Cosechadora.

Elementos	Bajos		Medios		Altos		Subtotal de puntos de función
	No.	Peso	No.	Peso	No.	Peso	
Entradas externas (EI)	**7**	3	**3**	4	**1**	6	**39**
Salidas externas (EO)	**3**	4	**1**	5	**0**	7	**17**
Ficheros lógicos internos(ILF)	**4**	7	**0**	10	**1**	15	**43**
Total							**99**

Anexo 17. Cálculo de los puntos de función desajustados.

Multiplicadores	Descripción	Valor
RELY	Garantía de funcionamiento requerida al software.	**5**
DATA	Tamaño de la base de datos.	**3**
CPLX	Complejidad del producto.	**5**
DOCU	Cantidad de artefactos que deben ser documentados.	**5**
RUSE	Desarrollo para ser reutilizado.	**3**
TIME	Exigencias sobre capacidad de ejecución.	**5**
STOR	Almacenamiento.	**4**
PVOL	Volatilidad de la plataforma.	**5**
ACAP	Capacidad de los analistas.	**4**
PCAP	Capacidad de los programadores.	**5**
PCON	Continuidad del personal.	**4**
APEX	Experiencia previa de los analistas.	**3**
PLEX	Experiencia con la plataforma.	**3**
LTEX	Experiencia previa con el lenguaje y herramientas de desarrollo.	**4**
ITOOL	Uso de herramientas de software.	**4**
SITE	Desarrollo en localidades distribuidas.	**2**
SCED	Exigencias sobre el calendario.	**4**

Anexo 18. Multiplicadores de esfuerzo.

Factores de escala	Descripción	Valor
PREC	Precedencia	**4**
FLEX	Flexibilidad	**4**
RESL	Riesgos	**3**
TEAM	Cohesión del equipo	**0**
PMAT	Madurez de las capacidades	**3**

Anexo 19. Factores de escala.

Constantes	
A	**2,94**
B	**0,91**
C	**3,67**
D	**0,28**

Anexo 20. Valores calibrados.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del trabajo de diploma titulado:
"Software para determinar la capacidad de carga y régimen de marcha del conjunto tractivo" y que el mismo pertenece a la Facultad de Informática y Matemática de la Universidad de Holguín "Oscar Lucero Moya", para que se haga el uso que se estime pertinente con este trabajo.
Para que así conste firman la presente a los ___ días del mes de ___ del año 2010.

_____

Autor: Leonar Alberto López Escobar

_____

Tutor: MSc. Ing. Vladimir Álvarez Sánchez
Prof. de la Fac. Mecánica

_____

Tutor: Ing. Arquimedes R. Leyva Tellez
Prof. del Dpto. Informática