**Universidad de las Ciencias Informáticas**

**"Facultad 3"**


# Título: "Análisis y Diseño de los módulos Inventario y Administración del proyecto ERP Cubano".

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

**Autores:** Dina Yaksilik Torres Saquipova.
Carlos de la Rosa Hernández.
**Tutor:** Ing. Carlos Felipe Pérez León.
**Tutor Asesor:** MSc. José Raúl Rodríguez Galera.
**Tutor Consultante:** Ing. Susel Vázquez Acuña.

Ciudad de La Habana
Junio, 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

| | | |
|---|---|---|
| _____ | _____ | _____ |
| Firma del Autor | Firma del Autor | Firma del Tutor |

DATOS DE CONTACTO

Ing. Carlos Felipe Pérez León.

Ingeniero en Ciencias Informáticas. Instructor Recién Graduado.

MSc. José Raúl Rodríguez Galera.

Licenciado en Educación. Pedagogo. Máster en Género, Educación Sexual y Salud Reproductiva. Profesor de Metodología de la Investigación Científica en pregrado y postgrado. Cuenta con 24 años de experiencia en la docencia. 8 años de experiencia en tutorías, oponencias y tribunales de tesis de pregrado. Cotutor de dos tesis de maestría en Farmacología en la UH y 16 de pregrado para Ingenieros en Ciencias Informáticas en la UCI. Cuenta en su haber con investigaciones institucionales y nacionales, participación en eventos nacionales, internacionales y un mundial, 10 publicaciones en los últimos 5 años. Exvicedecano de Investigaciones del ISCM de La Habana.

Ing. Susel Vázquez Acuña.

Ingeniera en Ciencias Informáticas.

AGRADECIMIENTOS

*A nuestro Comandante en Jefe Fidel Castro Ruz, que aportó, apoyó y siguió constantemente la brillante idea de fundar la Universidad de las Ciencias Informáticas.*

*A la universidad, por haberme dado todas las alegrías, penas, amigos y experiencias inolvidables que agradezco y llevaré conmigo para siempre.*

*A mis grandes amigos del preuniversitario, Luis Enrique, Ariel Yaser, Jorge Daniar y Raúl René, además del resto del grupo, por regalarme los momentos más inocentes, puros y gratos de mi vida, por hacerme llorar añorando que regresen atrás esos tiempos y por estar presentes todo el tiempo.*

*A todos los que una vez comenzaron conmigo la universidad, miedosos, primerizos al fin, pero que lograron formar parte de mi corazón, los estudiantes del 3110, que luego pasó a ser 3206 y continuó evolucionando y afianzando más mis ideas de que ese fue el mejor grupo que tuve en la universidad.*

*A mis nuevos amigos de la universidad, que no quiero que se sientan dolidos, pero me duele más dejar de mencionar algunos que se me puedan pasar, por eso el agradecimiento va para todos, por hacer mi estancia en la escuela mucho más agradable, amena y alegre.*

*A Carlos, mi compañero de tesis, por ser uno de los hombres más nobles y discretos que conozco, por haberme dado el placer de trabajar con él y comprobar, una vez más, que el conocimiento no se mide por lo mucho que se hable, sino por lo que se logra hacer con él.*

*A mis tutores Carlos Felipe, José Raúl y Susel, por haberme ayudado a realizar este trabajo.*

*A mi familia, los seres por los que estoy eternamente agradecida y a los que le debo la vida:*

*A mis padres, Aliya y Pablo, por ser las personas más buenas para mí, por quererme, enseñarme, educarme y confiar en mí en todo momento, por ser los seres incondicionales que me dieron vida y que ahora esperan lo mejor de mí, por ser mis amigos, las personas que más quiero y admiro.*

*A mi hermanita Dinara, que siempre será "mi chiquitica", por llegar al mundo para completar la familia, por servirme de inspiración para ser mejor cada día con tal de que siga mi ejemplo y acepte mi apoyo cuando lo necesite, por ser esa eterna niña que tanto quiero y a la que le deseo el mejor de los futuros.*

*A mi esposo Fernando, por ser la persona más paciente que he tenido a mi lado, por apoyarme y sensibilizarse con todo lo que me concierne, por todo su amor y entrega, por su confianza y admiración por mí.*

*Dina.*

*A mis padres Yolanda y Carlos, gracias por todo el apoyo, confianza y cariño que siempre me han dado. Ustedes saben que este trabajo es de ustedes.*

*A mis abuelos Otilio, Bena, Domingo y Luisa, gracias por todo lo que hicieron por mí, ustedes saben que gran parte de este esfuerzo se los debo a ustedes.*

*A mis hermanos Carel y Lisi, gracias por ser los mejores hermanos y amigos. Ustedes saben que siempre que me necesiten estaré presente como ustedes siempre lo estuvieron para mí.*

*A mis tíos Alberto y Mario, por preocuparse siempre por mi.*

*A mis primas Anyela, Jenny y Ailyn.*

*A mi novia Susy, gracias por estar siempre cuando te necesité, por todo tu apoyo, tu confianza y por el amor que has depositado en mí. Tú sabes que te quiero mucho.*

*A mis amigos de la infancia Franklin, Raymer, Jesús y Livier gracias por los momentos que pasamos juntos y las cosas que compartimos.*

*A mis amigos de Ceballos 2: Maykel y Orley.*

*Al grupo 3107, que después fue el 3201 y 3301 por estar siempre unidos y poder contar con todos ustedes.*

*A mis amigos de la UCI, Roldán, Oliver, Enrique, Yoan, Armando, Richard, Jorrín, Verdecia, Pedro, por todos los recuerdos tan buenos que me llevo de ustedes y de los momentos que pasamos en la Universidad.*

*A mi compañera de tesis, Dina, por todo este tiempo que estuvimos juntos, gracias por tu amistad y tus buenos consejos.*

*A mi tutor, Carlos Felipe, por ayudarnos y por el tiempo que nos dedicó.*

*A todas las personas que me han ayudado de alguna manera a terminar mis estudios.*

*Carlos.*

DEDICATORIA

*A mis padres, mi hermana y mi esposo por ser mi familia, por quererme tanto, por ser el motivo de mi felicidad y mi motivo de inspiración en este trabajo.*

*Dina.*

*Lo único que trasciende la existencia del ser humano… es su obra.*
*Roberto González O.*

*A mis padres, abuelos y hermanos por todo el cariño que recibí de su parte y por poder contar con ustedes.*

*A Susy, por ser parte de mí.*

*A Livier, mi "hermanito".*

*Carlos.*

*Libre, y para mi sagrado, es el derecho de pensar... La educación es fundamental para la felicidad social; es el principio en el que descansan la libertad y el engrandecimiento de los pueblos.*
*Benito Juárez*

RESUMEN

Debido a la necesidad que presenta el país de automatizar e integrar sus procesos, se crea un proyecto en la Universidad de las Ciencias Informáticas, con el objetivo de desarrollar un sistema de Planificación de Recursos Empresariales, que permita gestionar la información y automatizar las operaciones de las empresas cubanas dedicadas a la producción de bienes o servicios, tales como ventas, compras, contabilidad, administración de inventarios, que por su gran complejidad, constituyen módulos del sistema.

El propósito fundamental de este trabajo es realizar el modelado del sistema y del diseño de los módulos Inventario y Administración.

El documento consta de cuatro capítulos. El primero abarca un estudio del estado del arte de todo el contenido teórico que se debe conocer para su realización. El capítulo dos presenta el modelo del negocio y del sistema. El capítulo tres presenta el diseño del sistema y el último capítulo evalúa la calidad del trabajo.

Su importancia y novedad radica en que se realiza el análisis y diseño de dos módulos de un software completamente nuevo en Cuba, un ERP Cubano, posibilitando pasar a su implementación. Este sistema deberá lograr optimizar los procesos empresariales y el manejo de la información, disminuir los costos totales de operación, compartir información entre todos los componentes de la empresa y disminuir el margen de contaminación y repetición de la información.

PALABRAS CLAVE

Requerimientos, diseño, patrones, métricas.

**Contenido**

# Contenido

# Contenido

## INTRODUCCIÓN

En el mundo de hoy la mayoría de las operaciones y actividades giran en torno a una computadora. La informática soporta importantes y continuos cambios que repercuten en nuestra sociedad y en el desarrollo de nuestro mundo empresarial.

Las Tecnologías de la Información y las Comunicaciones (TIC), producto de sus avances y como integrantes de esta ciencia, han posibilitado el incremento rápido del desarrollo de la economía y la sociedad. Estos avances vienen dados gracias a la ambición de conocimiento del ser humano, a su naturaleza creadora, y son aplicados en gran medida en el desarrollo de la economía y la sociedad mediante la producción de software. Esta actividad demanda cada vez más un mayor estudio y una mejor planificación porque cada vez se solicitan programas más complejos y con mayor calidad.

Los sistemas de software se especializan en negocios determinados. Recientemente se está estilando desarrollar un sistema que constituye un paquete de software empresarial que integra todas las áreas de la organización que se vinculan al logro de su objetivo: la generación de productos y servicios, es el sistema de Planeación de Recursos Empresariales (*Enterprise Resource Planning*, ERP). Este sistema garantiza la centralización de la información de una empresa.

Implementar un sistema ERP es un proceso largo, costoso, complejo y que requiere de gran cantidad de desarrolladores. Es por esto que es más factible dividir su desarrollo en módulos que representan las distintas áreas de la empresa, de forma que se viabilice su proceso de desarrollo.

Uno de estos módulos abarca la Gestión de Inventarios. Esta permite controlar los medios que posee una organización. Se encarga de evaluar los procedimientos de entrada y salida de sus bienes, es decir, controla su existencia con el fin de hacer más rentable su posesión y garantizar en cierto grado el éxito de la organización.

Otro módulo presente en estos sistemas de gestión de información, comprende los trámites necesarios para acceder y administrar los permisos del sistema.

Cuba cuenta con algunos sistemas para la gestión de inventarios y almacenes especializados en el funcionamiento de entidades determinadas. Sin embargo, no existe un proceso estándar, por lo que se pretende implementar un sistema ERP cubano que pueda ser utilizado por cualquier empresa y que permita centralizar toda la información en la misma.

Además, se ha trazado como meta informatizar la sociedad para valerse de las ventajas que se derivan del uso de la tecnología y favorecer el desarrollo económico y social. Además de la evolución hacia el uso de las TIC, se necesitan realizar transformaciones referidas a la gestión empresarial, administrativa y de la calidad.

La Universidad de las Ciencias Informáticas (UCI) fue creada en el 2002 con el propósito de preparar recursos humanos que contribuirán con la informatización del país. Es una universidad productiva donde los estudiantes y sus conocimientos van encaminados a la producción de software, por lo que se le dio a ella la tarea de desarrollar el ERP Cubano. Esta tarea conlleva a un proceso en el cual es necesario optar por técnicas, métodos, herramientas, que permitan desarrollarlo con la mayor calidad y organización posible.

Una etapa muy importante y decisiva de este proceso es llevar a cabo una buena Ingeniería de Requisitos buscando obtener un mejor entendimiento entre las partes involucradas y lograr una correcta y clara definición de los requerimientos, de forma que se pueda realizar un mejor análisis y diseño.

Captar las necesidades del cliente es una problemática bastante generalizada durante el desarrollo de un software, tanto en el mundo como en nuestro país y en la UCI. Para desarrollar un software es necesario lograr un entendimiento común entre desarrolladores y clientes, e interpretar claramente lo que el cliente necesita. De esta manera se facilita el proceso de captar correctamente los requerimientos, y por tanto realizar un análisis y diseño lo más factibles posible, pues de otra manera, en la mayoría de los casos se producen fallos o errores que pueden prolongar el costo y tiempo de desarrollo del software.

Dada esta situación, se define como **problema:** ¿Cómo transformar las necesidades de los clientes a un lenguaje entendible por los desarrolladores, que permita la posterior implementación?

Para darle solución al problema planteado se define como **objetivo**: Realizar el levantamiento de Requisitos y el modelo de Análisis y Diseño de los módulos Inventario y Administración del proyecto de software ERP cubano.

El **objeto de estudio** para darle cumplimiento al objetivo es: el Proceso de desarrollo de Software, del cual se deriva como **campo de acción:** el Levantamiento de Requisitos y Modelado de Análisis y Diseño.

# Introducción

La investigación se basa en la **hipótesis** de que**:** Si se logra realizar el levantamiento de los requisitos y el modelado de análisis y diseño de los módulos Administración e Inventario, entonces se podrán transformar las necesidades del cliente a un lenguaje entendible por los desarrolladores.

Las **tareas** a realizar para realizar el trabajo de investigación son:

- Realización de un estudio del estado del arte enfocado en la ingeniería de requisitos y el diseño, así como de las herramientas, técnicas y patrones, que ayuden a comprender y poner en práctica lo estudiado para la realización de este trabajo.
- Modelación del negocio, sistema y diseño, identificando claramente los procesos que definen las operaciones propias de un almacén y la gestión de inventarios, definiendo los requisitos y aplicando patrones que hagan más sencilla la solución propuesta.
- Evaluación de la calidad de la solución propuesta mediante la aplicación de métricas para evaluar los requerimientos capturados, así como los casos de uso y el diseño propuestos.

Los siguientes **métodos teóricos** sustentan la investigación:

*Histórico-Lógico:* Su empleo permitió el desarrollo evolutivo y coherente en el estudio de la ingeniería de requisitos y diseño, patrones de casos de uso y de diseño, herramientas Case y sistemas ERP para el desarrollo de los artefactos que proponen los flujos estudiados.

*Analítico-Sintético:* Permitió integrar y descomponer el conocimiento, descubriendo las relaciones para la utilización de artefactos propuestos por constituir este método una unidad dialéctica, determinando los aspectos esenciales y el arribo a conclusiones prácticas y teóricas.

*Inductivo-deductivo:* Permitió pasar de lo particular a lo general y viceversa favoreciendo objetivamente el enlace que se establece en la realidad entre lo singular y lo general ya que ambas se complementan mutuamente en el proceso de desarrollo científico.

*Modelación:* Pues se crean abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados.

**Técnicas empleadas:**

Se emplean técnicas para comprender los procesos de negocio, para comprender sus necesidades a automatizar. Estas son la arqueología o revisión de documentos, la tormenta o lluvia de ideas.

Los **aportes prácticos** esperados del trabajo son:

- La obtención del modelo del negocio del módulo Inventario de forma que posibilite el entendimiento de sus procesos.
- Una especificación clara de los requisitos de los módulos Inventario y Administración que permita modelar el sistema.
- El modelado del sistema determinado por los requisitos captados para ambos módulos.
- El modelado del diseño que posibilite a los programadores integrarse rápidamente a la implementación del sistema.

El documento se divide en 3 capítulos en los que se puede encontrar:

*Capítulo 1:* Este capítulo presenta un estudio del arte sobre la ingeniería de requisitos, definiendo sus etapas básicas y las actividades que se realizan en cada una de ellas, así como del diseño, abordando su definición, principios, fundamentos para alcanzar una buena calidad y principales patrones. Se definen y clasifican los requerimientos. Se realiza una caracterización de la metodología de desarrollo empleada, así como de la herramienta utilizada para desarrollar los artefactos. Se presenta además una breve descripción de los roles analista de sistemas y diseñador como desarrolladores fundamentales en las etapa de captura de requisitos y diseño del sistema.

*Capítulo 2:* En este capítulo se realiza una caracterización del negocio propuesto. Se exponen los artefactos generados producto del análisis de los principales procesos de la gestión de inventarios en los almacenes y su descripción. Se emplea y explica la estrategia de captura de requisitos y modelado del sistema, representando los artefactos que se generan para ello según la metodología usada.

*Capítulo 3:* En este capítulo se realiza el modelado del diseño, donde se exponen los artefactos que lo componen, como subsistemas, diagramas de clases, arquitectura. Se aplican, además, los patrones de diseño y las métricas que se utilizan para evaluar la calidad de los requerimientos capturados, los casos de uso definidos y el diseño realizado.

Este documento posee, además, Conclusiones por cada capítulo, Conclusiones generales, Recomendaciones, Referencias bibliográficas, Bibliografía y Anexos.

## CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

### 1.1 Introducción.

En el capítulo se aborda un estudio del estado del arte de la Ingeniería de Requisitos, de sus técnicas, actividades y problemas para capturar los requerimientos de un software, así como del Diseño, sus conceptos, principios, patrones. Se realiza además un estudio de las herramientas, lenguaje de modelado y metodología de desarrollo de software a utilizar durante el desarrollo práctico del trabajo.

### 1.2 ¿Qué es un ERP?

Los sistemas de Planificación de Recursos Empresariales (*Enterprise Resource Planning*, ERP) son sistemas de gestión de información gerenciales que integran, automatizan y manejan muchas de las acciones asociadas con las operaciones de producción y distribución de una compañía dedicada a la producción de bienes o servicios.

Los ERP pueden intervenir en el control de muchas actividades de negocios, tales como ventas, compras, entregas, pagos, producción, contabilidad, administración de inventarios y de recursos humanos. Funcionan en todo tipo de empresas. Para ello, integran todos los departamentos funcionales de la empresa, herramientas de mercadotecnia y administración estratégica en un solo sistema (Wailgum, 2008).

En este trabajo investigativo se desean desarrollar dos módulos:

***Inventario:*** Controla los medios que posee una organización. Se encarga de inspeccionar las existencias de productos, dándole entrada y salida del almacén a los mismos, regulando el flujo de mercancía en el almacén, con el fin de hacer más rentable su posesión y garantizar en cierto grado el éxito de la organización.

***Administración:*** Comprende las actividades necesarias para acceder y administrar los permisos del sistema, de los usuarios y las terminales de usuarios.

Los **objetivos** principales de los sistemas ERP son (Adpime, 2008):

- Optimizar los procesos empresariales.
- Lograr un acceso a toda la información de forma confiable, precisa y oportuna (integridad de datos).
- Dar la posibilidad de compartir información entre todos los componentes de la empresa.
- Eliminar datos y operaciones innecesarias de reingeniería.

### *1.2.1 Características de los sistemas ERP.*

Un ERP se distingue de otro software empresarial porque es un sistema integral, con modularidad y adaptable (Wailgum, 2008):

- Integrales: Todos los departamentos o áreas de una empresa se relacionan entre sí, permitiendo de esta forma controlar los distintos procesos de la entidad, donde la salida de un proceso puede ser la entrada de otro.
- Modulares: Se dividen en módulos, que son las áreas que se integran como un todo para el manejo óptimo de la información. Estos módulos se instalan según las necesidades del cliente.
- Adaptables: Son diseñados para ser configurables según lo que cada empresa necesite.
- Base de datos centralizada.
- Sus componentes interactúan entre sí consolidando todas las operaciones.
- En un sistema ERP los datos se ingresan sólo una vez y deben ser consistentes, completos y comunes.
- Las empresas que lo implanten suelen tener que modificar alguno de sus procesos para alinearlos con los del sistema ERP. Este proceso se conoce como Reingeniería de Procesos, aunque no siempre es necesario.

### *1.2.2 Ventajas.*

- Los módulos de la empresa sufren mejoras en la comunicación entre ellos.
- Es un software que se puede personalizar, configurar y adaptar.
- Producto del gran análisis que emprende desarrollar un ERP se puede visualizar y aplicar cambios en los procesos de negocio de la empresa para optimizarlos (Adpime, 2008).
- Maneja mejor los datos y la información de la empresa, controlándolos y centralizándolos, y permitiendo compartirla entre los diferentes módulos, de manera que evita que los datos se corrompan, se dañen, se dupliquen (González Rodríguez, 2003).

### *1.2.3 Desventajas.*

Los sistemas ERP son complejos y difíciles de desarrollar e implementar, principalmente para las grandes compañías y trasnacionales. Conocer todos los departamentos, con todos sus procesos, y hacer una ingeniería con esa información, es algo bastante difícil. Se necesita de personal capacitado y entregado, de un gran esfuerzo, de mucho tiempo y capital monetario, pues los costos de implementación, prueba, implantación y capacitación son elevados. Teniendo incluso todas estas

necesidades cubiertas, no se puede garantizar que se puedan modelar correctamente todos los procesos del negocio de la empresa.

Si alguna área es ineficiente, afectará al resto de las áreas y por tanto, a la empresa.

Tampoco se puede garantizar que se haga una implantación del software exitosa. Lo que se puede hacer es trabajar lo mejor posible y llevar a cabo una correcta metodología de desarrollo (González Rodríguez, 2003).

### *1.2.4 Importancia.*

Aplicando un ERP en una empresa se brinda apoyo a los clientes del negocio, se ofrecen respuestas rápidas a los problemas y se optimiza el manejo de información que permita la toma oportuna de decisiones y disminución de los costos totales de operación. Esto se obtiene gracias a que es un sistema integrado. De esta forma se evita tener varios programas que controlen todos los procesos de una empresa, situación en la cual es más difícil lograr que la información no se duplique, que no aumente el margen de contaminación en la información y que no se cree un escenario favorable para malversaciones.

Con un sistema ERP la información no se manipula y se encuentra protegida.

### *1.2.5 Ejemplos.*

**Openbravo** es un sistema de gestión empresarial integrado (ERP) en software libre y basado íntegramente en web. Se encuentra disponible en español, inglés, italiano, portugués, ruso y ucraniano. Es una aplicación que puede ser usada desde cualquier navegador web (Openbravo, 2008).

**OpenXpertya** es un ERP de código abierto en español, especialmente adaptado para la legislación y el mercado español e hispanoamericano. Incluye los módulos necesarios para su consideración como un ERP, es una aplicación ERP de Software Libre. OpenXpertya se encuentra en fase plenamente funcional. Su principal ventaja monetaria es que no existen costos de licencia para el producto en sí mismo. El mayor diferenciador de todos modos es que se puede, además, obtener el código fuente (Openxpertya, 2008).

**Abanq** es software libre de tipo ERP (Enterprise Resource Planning) orientado a la administración, gestión comercial, finanzas y en general a cualquier tipo de aplicación donde se manejen grandes bases de datos y procesos administrativos. Su aplicación abarca desde la gestión financiera y comercial en empresas hasta la adaptación a procesos complejos de producción. Fue creado por

InfoSiAl y galardonado con el primer premio del concurso de creación de empresas de base tecnológica convocado por INNOVARED. Es el primer software ERP de código libre profesional, encontrándose ya en algunas de las más importantes distribuciones de Linux (Abanq, 2008).

**GestiCam** (Gestión Informática Integrada para Autónomos y Micropymes) es un software de Gestión en Código Libre y con una integración dentro del S.O. MoLinux que distribuye la Junta de Comunidades de Castilla-La Mancha (JCCM). Los módulos con los que cuenta son: Gestión de Compras, Gestión de Ventas, Gestión de Empresas, Gestión Financiera, Gestión de Inventario, Gestión de Productos, Gestión de Producción, Administración del Sistema, Copias, Restauraciones, etc., Informes. Posee una interfaz adaptable al usuario. Es multilingüe, multi-empresa, multi-departamentos, multiusuario, multi-moneda.

**Adempiere:** El proyecto ADempiere fue creado en Septiembre de 2006 después de las diferencias que se tuvieron entre los desarrolladores de Compiere™ y la comunidad que se formó alrededor del proyecto (Adempiere, 2008).

ADempiere cubre las siguientes áreas de negocio:

- Administración Planeación de Recursos (ERP)
- Administración de la Cadena de Suministro (SCM)
- Administración de la Relación con los Clientes (CRM)
- Análisis del Desempeño Financiero
- Solución Integrada de Punto de Venta (TPV - POS)
- Tienda Web Integrada

### 1.3 Ingeniería del Software.

Sommerville define al software en su libro (Sommerville, 2004), como programa de computadora y documentación asociada como requerimientos, modelo de diseño y manuales de usuario que puede ser generado para un cliente en particular o para todo un mercado.

El software según Pressman: “es la máquina que conduce a la toma de decisiones comerciales” (Pressman, 2001). Los desarrolladores capacitados, preparados y responsables de construir el software se conocen como ingenieros del software.

El término Ingeniería es definido por el Diccionario de la Real Academia Española de la Lengua (DRAE) como: “conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía”.

El proceso de desarrollo del software ha estado evolucionando constantemente. En la década de 1980 este proceso era menos complejo y se basaba en el ensayo-error. A medida que se ha desarrollado la industria informática, los sistemas de software han necesitado ampliarse y ganar en complejidad y calidad. Esto ha contribuido al cambio de la perspectiva de desarrollo de software y el surgimiento de nuevas técnicas, métodos, tendencias, disciplinas, dando lugar a la aparición de la Ingeniería de Software (IS), disciplina que emerge para aplicar estos nuevos conocimientos en el desarrollo de software y ganar en calidad y eficiencia, así como disminuir los costos de mantenimiento.

Varios autores han definido la IS de diversas maneras.
Boehm en su definición destaca que la IS es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora, resaltando la necesidad de generar documentación asociada al software, ya sea durante su desarrollo, durante su extensión hacia los usuarios o clientes, o para su posterior mantenimiento. (Boehm, 1976)

Por otra parte la IEEE Standards Collection de manera muy general afirma que la IS es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, funcionamiento y mantenimiento del software. (IEEE, 1993)

Esta disciplina ha favorecido el crecimiento económico y productivo en las últimas décadas. Los servicios sociales también han sido beneficiados, producto de la disminución de los costos y el aumento de la calidad de los servicios.

Para lograr esto se unifica el esfuerzo de varias personas, a veces grandes cantidades, que juegan diversos roles en el proceso de desarrollo de software: arquitectos, programadores, analistas, diseñadores.

### *1.3.1 Rol Analista del Sistema.*

La metodología Rational Unified Process (RUP) define varios roles que son adoptados por los desarrolladores del proyecto, de forma que las actividades queden más organizadas y tengan sus responsables, con el fin de llevarlo a cabo lo más organizado posible. Uno de estos roles lo constituye

el analista, que es quien analiza los sistemas existentes con el fin de automatizarlos. Es la persona que:

- Logra la comunicación y el entendimiento mutuo con el cliente.
- Estudia la situación o problema existente y lo describe.
- Identifica qué se desea automatizar.
- Identifica mejoras.

Un analista debe tener y cumplir con ciertas competencias que lo distinguen y hacen un tanto difícil su elección. Debe ser una persona inteligente, detallista, diplomática, analítica, perceptiva, innovadora, negociadora, conversadora, escuchadora, conocedora del tema.
"El objetivo del analista es el reconocimiento de los elementos básicos del problema tal y como los percibe el cliente/usuario." (Pressman, 2001)

El analista de sistemas lidera y coordina la elicitación de requisitos y el modelado de los casos de uso delimitando las funcionalidades del sistema. (Rational, 2003)

### *1.3.2 Rol Diseñador del Sistema.*

El diseñador es responsable de diseñar una parte del sistema, que incluye requisitos, arquitectura y proceso de desarrollo del proyecto. Para ello, diseña los subsistemas, paquetes y clases que deberá tener el sistema, que permiten la realización de los casos de uso.

El diseñador identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos del diseño. Se encarga de realizar un diseño consistente con la arquitectura del software que se tenga, y se detalla hasta el punto en que la programación puede proceder. (Rational, 2003)

Para realizar estas actividades, es necesario que el diseñador tenga conocimientos sobre requerimientos de un sistema, arquitectura, técnicas de diseño, incluyendo análisis orientado a objeto y UML, y las tecnologías con las que se implementará el software.

### *1.3.3 Ingeniería de Requisitos.*

Actualmente existe un alto porcentaje de proyectos de software que no terminan exitosamente, que no cumplen sus objetivos. Una de las principales causas radica en la definición de los requerimientos. Esta actividad no consumada con la seriedad y calidad requerida, provoca que muchos proyectos de software fracasen.

Durante el proceso de desarrollo de software se realiza una tarea fundamental: definir lo que se desea producir, que consiste en comprender y describir claramente el comportamiento del sistema y de los problemas que se necesita resolver. En general, en esto se basa la Ingeniería de Requisitos (IR), en definir las necesidades del sistema.

Esta disciplina no posee una técnica estandarizada que garantice la calidad de los resultados, sino que estudia métodos, técnicas y herramientas que garanticen una correcta y adecuada definición de los requisitos, dependiendo de cada equipo de desarrollo y sus clientes, lo que estimula el éxito de un proceso de desarrollo de software en gran medida. Un factor muy importante para llevarla a cabo es lograr un entendimiento común entre desarrolladores y clientes. De esta forma se desarrolla mejor la actividad y se identifican con más claridad los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

Christel y Kang definen la IR como el proceso sistemático de desarrollar requisitos mediante un proceso iterativo y cooperativo de analizar el problema, documentar las observaciones resultantes en varios formatos de representación y comprobar la precisión del conocimiento obtenido. (Christel, y otros, 1992)

Boehm, por su parte plantea que la IR “es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizará el sistema". (Boehm, 1976)

“La Ingeniería de Requisitos es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un coste reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo que satisfaga las necesidades del usuario”. (Pressman, 2001)

El levantamiento de requisitos es un proceso iterativo en el que se especifican y validan las necesidades de los usuarios finales y clientes, convertidas en funcionalidades que el software debe cumplir y en características que este debe poseer.

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. (Jacobson, y otros, 2000) Con la IR se logra una mejor descripción de lo que el sistema debe hacer, dado por una mejor comunicación y entendimiento entre clientes y desarrolladores, situación que logra en gran medida disminuir los costos y retrasos de un proyecto por errores que se hubiesen

detectado más tarde. Esto provoca que mejore la calidad del software, y por consiguiente la satisfacción del equipo de desarrolladores y los clientes.

La IR permite gestionar las necesidades de forma organizada y bien definida. Constituye además un punto de partida para actividades mayormente de estimación que se realicen en el proyecto.

#### *1.3.3.1 ¿Qué es un requerimiento?*

La IEEE define un requerimiento como: "Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. (2) Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal. (3) Una representación documentada de una condición o capacidad como en (1) o (2)". (IEEE, 1993)

Los requerimientos representan las necesidades de los usuarios y los objetivos del sistema. Con frecuencia, durante el ciclo de vida del proyecto, estos varían por diferentes razones. Algunas pueden ser por no captar correctamente la información, por cambios que surgen en las empresas o en las perspectivas del cliente, por cambios en el mercado. Es necesario destacar que los cambios en los requisitos implican cambios en el tiempo de desarrollo y en todas las actividades que dependen de esta, incluso pueden llegar a perjudicar la calidad del software.

#### *1.3.3.2 Características de los requerimientos.*

Los requerimientos deben ser concisos y completos. Deben especificar claramente todo lo que necesitan para llevarse a cabo, ya sean entradas, salidas, válidas o no, todas las posibles respuestas y situaciones. No deben ser ambiguos, solo deben tener una interpretación. Deben ser verificables, es decir, se debe poder chequear que cada requisito es cumplido por el software. Además, los requisitos deben ser lo más adaptables y modificables posible. (Torres, 2008)

Los requerimientos deben ser capaces de satisfacer todos y cada uno de los objetivos del sistema. Deben estar clasificados según su importancia.

#### *1.3.3.3 Clasificación de los Requerimientos.*

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales.

Los requerimientos funcionales (RF) definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Además, son independientes de las tecnologías usadas por el producto. (Robertson, y otros, 2006)

"Los requerimientos no funcionales (RNF) especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad". (Jacobson, y otros, 2000)

Los RNF "son propiedades o cualidades que el producto debe tener". (Young, 2004)

Estas propiedades pueden determinar su calidad o la satisfacción del cliente, pues son los que garantizan que el sistema sea agradable al usuario, confiable y más rápido.

- *Requisitos de Software:* Se refieren al software que se debe tener, al sistema operativo con el que el sistema podrá trabajar.
- *Requisitos de Hardware:* Se refieren a los elementos de hardware que se necesitan para que el sistema o software cumpla sus funcionalidades.
- *Restricciones en el diseño y la implementación:* Se refieren a restricciones ordenadas, que se deben cumplir estrictamente. Especifican o restringen la construcción del sistema.
- *Requisitos de apariencia o interfaz externa:* Se refieren a la apariencia que debe presentar el producto, a cómo debe ser la interfaz externa del producto y no específicamente al diseño de la interfaz detalladamente.
- *Requisitos de Seguridad:* Se refieren al manejo de la seguridad. Si no son bien gestionados, puede conducir a riesgos grandes. Pueden responder a la confidencialidad: que la información esté protegida de accesos no autorizados, integridad: que la información esté segura de cambios corruptos manejados sin autorización, disponibilidad: que la información esté disponible y según el nivel de acceso que tenga cada usuario.
- *Requisitos de Usabilidad:* Se refieren a los niveles de usabilidad que existirán para acceder a información y demás en el sistema. Determinan las características generales de la capa de presentación del sistema en cuanto a las características de diseño gráfico de la misma, además de las facilidades para que el uso del sistema por parte del usuario final.

- *Requisitos de Soporte:* Se refieren a las actividades que se desarrollan después de terminado el software para ayudar, preparar y asistir a los clientes, y para mejorar gradualmente el sistema.

- *Requisitos de Rendimiento:* Están relacionados con tiempos de respuesta estimados, requeridos y esperados para la ejecución en línea de procesos del sistema, teniendo como base la plataforma tecnológica y escenarios específicos a los que en teoría el sistema estará expuesto y frente a los que deberá responder. (Informática, 2006)

#### *1.3.3.4 Actividades de la Ingeniería de Requisitos.*

Diferentes autores definen diferentes actividades para la realización de la ingeniería de requisitos. Estas al final se centran en los siguientes procesos:

- *Elicitación:* Consiste en extraer de varias fuentes y formas, información que necesita el equipo de desarrollo de software para entender el problema en cuestión y saber qué es lo que el sistema necesita. Permite conocer los procesos que se desarrollan en el negocio y comenzar la interacción con los clientes. Este proceso no es sencillo, pues el equipo de trabajo comienza a interactuar en un entorno y terminología desconocidos para él.

- *Análisis de Requisitos y Negociación:* Cuando se tienen los requisitos, se agrupan y organizan por categorías y subconjuntos, se analizan, se examinan buscando problemas en su consistencia, completitud, ambigüedad, compatibilidad con los demás requisitos, y se clasifican según las necesidades del cliente. Muchas veces los clientes proponen requisitos contradictorios. El desarrollador, en este caso el analista, debe llevar a cabo entonces un proceso de negociación con el cliente, donde se clasifiquen los requisitos, se vean los conflictos presentes según su prioridad, los riesgos que puede presentar cada requisito y se realicen estimaciones de esfuerzo para valorar el impacto de cada uno de los requisitos en el plazo de entrega y coste del proyecto.

- *Especificación de Requisitos:* Se describen las restricciones, funcionalidades y características que deberá tener el sistema a desarrollar. “La Especificación del Sistema es el producto final sobre los requisitos del sistema obtenido por el ingeniero. Sirve como fundamento para la ingeniería del hardware, ingeniería del software, la ingeniería de bases de datos y la ingeniería humana. Describe la función y características de un sistema de computación y las restricciones que gobiernan su desarrollo. La especificación delimita cada elemento del sistema. La

Especificación del Sistema describe la información (datos y control) que entra y sale del sistema". (Pressman, 2001). La Especificación de los Requisitos es el documento que permite negociar el acuerdo de lo que el sistema debe hacer y cumplir, y de esta forma evitar retrasos y mayores costos por errores a solucionar. Es el punto de partida de la estimación de costo, tiempo y esfuerzo del proyecto e indudablemente, influye en la calidad del producto final.

- *Validación de Requisitos:* Se evalúa la calidad del trabajo realizado hasta el momento. Examina las especificaciones con el objetivo de asegurarse de que los requisitos no son ambiguos, inconsistentes y que los errores detectados anteriormente fueron corregidos. Se basa también en demostrar que los requisitos definidos para el sistema constituyen lo que el cliente desea. La validación puede guiarse para descubrir errores. Algo que resulta muy útil es chequear cada requisito con un cuestionario. Esto le da más seguridad al equipo a la hora de revisar cada requisito. La revisión técnica formal es un mecanismo de validación. Se reúne un equipo de clientes, usuarios, desarrolladores, y revisan la documentación en busca de errores, ambigüedades, inconsistencias, redacciones inconclusas.

- *Gestión de Requisitos:* Es un conjunto de actividades paralelas a las anteriores, que se llevan a cabo durante todo el período de desarrollo que le proporcionan al equipo de trabajo la posibilidad de identificar, controlar y seguir los cambios en los requisitos. (Pressman, 2001)

#### *1.3.3.5 Problemas que se presentan cuando se identifican los requisitos:*

Los desarrolladores, durante el levantamiento de los requisitos, se pueden encontrar con algunas afectaciones que impiden capturarlos con mayor claridad, lo que repercute luego en la calidad del producto final, y a veces hasta puede provocar la no terminación del mismo. Algunos de estos problemas son: (Gómez Vela)

- Muchos de los detalles aportados con el fin de clarificar pueden confundir a los analistas y los objetivos del sistema.
- Los desarrolladores pueden no escuchar correctamente a los clientes, y esto puede traer consigo falta de información o incomprensiones luego.
- La comunicación y el entendimiento mutuo entre desarrolladores y clientes puede afectarse por relaciones sociales, diferencias de caracteres y demás.
- Los clientes a menudo omiten detalles pensando que no son importantes.
- Los requisitos no son estables, pueden cambiar.
- Los requisitos no siempre son obvios.

- El lenguaje de los clientes puede resultar ambiguo si no tienen amplio conocimiento del tema.
- Cuando los proyectos son grandes, resultan gran cantidad de requisitos y se hacen más difíciles de manejar.
- Los clientes no están seguros de lo que necesitan, no tienen un amplio conocimiento informático, o no dominan bien el problema.

#### *1.3.3.6 Técnicas para el levantamiento de requisitos.*

A través de los años se ha optado por desarrollar algunas técnicas con el objetivo de ayudar a mejorar el levantamiento de información. Estas son manejadas por un equipo de clientes y desarrolladores que trabajan juntos para comprender el problema, proponer soluciones, negociar diferentes perspectivas o puntos de vista y especificar un conjunto básico de requisitos de la solución.

***Entrevista:*** Es usada con frecuencia para acercar al cliente y al desarrollador, pues logra una muy buena comunicación entre ambas partes. Es muy aceptada y ampliamente extendida. Permite obtener información sobre el problema en cuestión y comprender los objetivos para su solución. Para realizarla es necesario seleccionar correctamente a los entrevistados, definir previamente las preguntas, pues la forma de redacción puede influir en las respuestas y por último, analizar los resultados. (Koch, y otros, 2002)

***Cuestionarios:*** Consisten en una serie de preguntas a responder que pueden ser abiertas o cerradas. Las abiertas son más reveladoras, pues permiten que los encuestados respondan con su propia terminología y no los limitan en sus respuestas. Este tipo de preguntas son útiles cuando no se conoce mucho del tema, y por ende no es posible predeterminar posibles respuestas, o cuando se desea profundizar en criterios personales, opiniones, ideas y, por tanto, necesitan de mayor esfuerzo para su elaboración, respuesta y análisis. Las cerradas poseen las posibles respuestas a elegir, requieren menos esfuerzo por parte del encuestado, son más fáciles de preparar para su análisis, requieren de menos tiempo de realización, pero limitan muchas veces al encuestado a la hora de responder, pues a veces su respuesta no se ajusta exactamente con las posibles variantes que se le presentan. (Dávila, 2001)

***Tormentas de idea (Brainstorming):*** Es una técnica de reuniones que se usa para generar ideas. Su objetivo principal es concebir la mayor cantidad de requisitos para el sistema. Reúne integrantes de varias disciplinas, mientras mayor experiencia y preparación tengan, mayor posibilidad se presenta de generar buenas ideas. En este tipo de reunión no se debe socavar ideas que parezcan locas, ni evaluar o criticar las ideas de los demás, porque puede producir un ambiente hostil que perjudique el

dinamismo de la reunión, además de que esa idea puede constituir un gran aporte luego de ser madurada y perfeccionada o relacionada con otras. (Zapata, y otros, 2004)

***Desarrollo conjunto de aplicaciones (Joint Application Development, JAD):*** Es una práctica de grupo donde participan analistas, clientes, usuarios, administradores. Se basa en cuatro principios: dinámica en grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación (Ford, y otros, 1994). Esta técnica se realiza durante varios días y favorece la dinámica de grupo, el trabajo en equipo, la mantención de un proceso organizado. En cada encuentro se analizan los problemas y se exploran soluciones que se presentan de manera que se concrete y documente todo lo acordado y concluido. De esta forma se analizan opiniones de varias personas a la vez, a diferencia de la entrevista, donde se ve todo por separado. Esta técnica constituye una alternativa a la entrevista individual que ahorra tiempo al evitar que se analice la opinión de cada cliente por separado. Además, la documentación es revisada por todo el grupo presente (Robertson, y otros, 2006)

***Mapas conceptuales:*** Es una técnica muy eficiente para obtener una idea del negocio, las relaciones que presenta y el vocabulario, incluso puede ayudar a aportar términos al glosario de términos. Consiste en la representación de conceptos sobre el problema donde se muestran sus asociaciones y atributos mediante un grafo, donde los conceptos son los vértices, y las relaciones son las aristas. Esta técnica es muy usada en la IR, pues es fácil de entender por parte del usuario o cliente, aunque puede llegar a ser ambigua si no se usa correctamente (Fowler, 1996).

***Arqueología de documentos:*** Se inspecciona la documentación de la empresa con el fin de familiarizarse más con el negocio y poder identificar posibles requisitos.

#### *1.3.3.7 Patrones de Casos de Uso.*

Los patrones de casos de uso son parejas de problema/solución que permiten resolver problemas que se le pueden presentar a los desarrolladores a la hora de modelar el sistema de forma rápida y ágil. Algunos de estos son (Övergaard, et al., 2004):

- CRUD (Crear, Leer, Modificar, Eliminar):
  - Completo: Se utiliza para gestionar información en los casos en los que se quiere crear, visualizar, modificar y eliminar información. Este patrón permite reducir el número de casos de uso y el tamaño del modelo, lo que lo hará más entendible.

- Parcial: Modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es mas significativa, larga o más compleja que las otras.

- Concordancia (Commonality): Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

- Extensión concreta: Patrón que consiste en dos casos de uso y una relación de extensión. El caso de uso extendido es concreto, es decir, puede ser instanciado por su cuenta como por el caso de uso base. Este patrón es aplicable cuando un flujo puede extender el flujo de otro caso de uso, lo que significa que puede ocurrir el proceso del caso de uso base, o puede ocurrir el del caso de uso base con su caso de uso extendido. (Övergaard, y otros, 2004)

- Múltiples actores:

  - Roles diferentes: Captura la concordancia entre actores manteniendo roles separados. Trabaja con un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso.

  - Roles comunes: Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso.

Estos patrones brindan la posibilidad de realizar un mejor y más entendible modelado del sistema. Esto es de gran importancia, pues este modelo es una entrada fundamental para realizar el diseño del sistema.

### *1.3.4 Diseño de Sistema.*

El diseño es una representación significativa de la ingeniería de algo que se va a construir. En el contexto de la ingeniería del software se centra en cuatro áreas importantes de interés: datos, arquitectura, interfaces y componentes (Pressman, 2001)

Cuando se diseñan sistemas informáticos es indispensable hacerlo de forma correcta. El diseño propuesto tiene que cumplir a cabalidad con los requerimientos del sistema. Es la única forma de

convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado (Pressman, 2001); debe ser capaz de facilitar las mejoras del software, tiene que ser entendible por otros profesionales de la especialidad, servir como guía para los demás pasos de la ingeniería de software, permitir la comprobación del sistema fácilmente.

Los principales **objetivos** del Diseño, según Jacobson son (Jacobson, y otros, 2000):

- Comprender a profundidad los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos y tecnologías de interfaz de usuarios.
- Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Crear una abstracción sin costuras de la implementación del sistema, en el sentido de que la implementación es un refinamiento del diseño.

El modelo de diseño, principal resultado en este flujo de trabajo, “se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica” (Jacobson, y otros, 2000), es decir, contribuye a crear un plano del modelo de implementación, ya que describe la realización física de los casos de uso, considerando el impacto que tienen para el sistema los requisitos y las restricciones relacionadas con el entorno de la programación.

Este modelo se puede representar por subsistemas de diseño, que no son más que porciones más manejables del diseño que constan de realizaciones de casos de uso, clases del diseño, interfaces, además de otros subsistemas, y que deben ser cohesivos y débilmente acoplados.

El diseño de las especificaciones del producto en relación con las funciones esperadas del mismo por el usuario, determinará las características y limitaciones, modo de utilización, exigencias de fiabilidad, seguridad, mantenimiento, vulnerabilidad, ergonomía, transporte.

La simplicidad, reducción del número de componentes, facilidad de manejo y manipulación, contribuyen a evitar errores y defectos y, por regla general, reducen el coste del producto. (Berlinches Cerezo, 2004)

#### *1.3.4.1 Conceptos del Diseño.*

Los conceptos del diseño permiten conseguir un modelo correcto de diseño para asegurar la buena calidad del sistema a diseñar. (Carlos, s/a)

Los conceptos básicos del diseño son de mucha importancia para cualquier diseñador, debido a que proporcionan la base para aplicar los métodos de diseño. Al hacer un programa se puede lograr que funcione, pero es necesario lograr que lo haga correctamente. Eso lo permiten los conceptos de software. Pressman en una de sus publicaciones trata una serie de conceptos importantes (Pressman, 2001):

***Abstracción***

Se pueden exponer distintos niveles de abstracción, en dependencia del problema que se esté resolviendo. El nivel más alto es empleando el lenguaje del entorno del problema. En los inferiores a este se toma una orientación procedimental, hasta llegar al nivel más bajo, donde se establece la solución para implementarse.

Cada paso en el proceso de desarrollo de software es un refinamiento en el nivel de abstracción. Así desde el Negocio, los Requisitos, el Análisis y Diseño y demás disciplinas de RUP se van reduciendo los niveles de abstracción hasta llegar al más bajo, cuando se genera el código fuente.

Existen tres tipos fundamentales de abstracciones que se crean a la hora de adentrarse en los distintos niveles de abstracción, ellos son: abstracción procedimental, abstracción de datos y abstracción de control.

***Refinamiento***

El desarrollo de un programa se realiza refinando sucesivamente los niveles de detalle procedimentales. El refinamiento es un proceso de elaboración, se comienza a un alto nivel de abstracción y se va trabajando sobre las sentencias originales, llegando a un alto nivel de detalle a medida que se van realizando los distintos refinamientos. Todos los pasos del refinamiento implican decisiones en el diseño.

***Modularidad***

Cuando un equipo de desarrollo está encargado de construir un sistema grande, comúnmente este se divide en distintas partes, llamadas módulos, que al integrarlos cumplen con los requerimientos que plantea el cliente. La modularidad es la encargada de dividir el problema en pequeñas partes, pero hay que prestar atención al dividir en módulos, debido a que un número mayor de módulo implicaría un menor tamaño de los mismos, pero un mayor costo de desarrollo. Existen cinco criterios que permiten definir un sistema modular efectivo:

- Capacidad de descomposición modular.
- Capacidad de empleo de componentes modulares.
- Capacidad de compresión modular.
- Continuidad modular.
- Protección modular.

***Arquitectura de Software***

Es la estructura del software, la estructura jerárquica de los componentes, módulos, la manera en que ellos interactúan y las estructuras de datos que usan los distintos componentes. Entre los objetivos del diseño está generar distintas vistas arquitectónicas del sistema, que son importantes para la comprensión del mismo por los demás integrantes del equipo de desarrollo.

***Jerarquía de Control***

O denominada también como estructura de programa, representa la organización de los componentes e implica una jerarquía de control. No se puede aplicar a todos los estilos arquitectónicos. Para arquitecturas de llamada y de retorno el diagrama más común es el de forma de árbol. Para la elaboración de un software orientado a objetos no se aplican esas medidas.

***División Estructural***

Cuando el estilo arquitectónico es jerárquico, la estructura se puede dividir tanto horizontal como vertical. La división horizontal proporciona diferentes ventajas: software más fácil de probar, conduce a un software más fácil de mantener, propaga menos efectos secundarios, software más fácil de ampliar. En la vertical o factorización, como también se le suele llamar, los módulos de nivel superior llevan a cabo las funciones de control, y los inferiores son los que realizan todas las tareas de entradas, proceso y salida.

***Ocultación de Información***

Plantea que los módulos se caracterizan por las decisiones de diseño, donde cada módulo oculta al otro, los módulos tienen que quedar diseñados de tal manera que la información que esté dentro de un módulo sea inaccesible para aquellos que no la usan. Significa que se puede conseguir una modularidad efectiva teniendo los módulos de manera independiente, pero comunicándose entre sí y solo intercambiando la información necesaria para el correcto funcionamiento del sistema.

Para realizar un diseño modular efectivo es necesario garantizar la independencia funcional, que procede de trabajar la modularidad, la abstracción y el ocultamiento de información. Es necesario, además, que se ponga en práctica la alta cohesión (que los módulos tengan funcionalidades únicas,) el bajo acoplamiento (mínima interacción con el resto) y que las interfaces de usuario sean lo más sencillas posibles.

### *1.3.4.2 Principios del Diseño.*

Varios autores plantean una serie de principios básicos que se tienen que tener en cuenta a la hora de diseñar para garantizar el correcto funcionamiento del software.

Según Alan Davis son los siguientes (Davis, 1990) :

- En el proceso deben tomarse enfoques alternativos.
- Deberá rastrearse hasta el análisis.
- Se debe reutilizar.
- Tratar de imitar el dominio del problema.
- Uniformidad e integración.
- Deberá estructurase para admitir cambios.
- Debe prever la adaptación a circunstancias inusuales.
- No codificar.
- Evaluarse en función de calidad mientras está creciendo.
- Minimizar errores conceptuales.

Pressman plantea los mismos principios de diseño, lo que ratifica la concordancia entre ambos autores, y por tanto, su validez.

Cuando estos principios son aplicados correctamente el diseño creado muestra los factores de calidad internos (son importantes para los ingenieros y personal encargado de realizar el proyecto) y externos (son aquellas propiedades del software que se observan fácilmente: velocidad, fiabilidad, grado de corrección, usabilidad).

#### *1.3.4.3 Patrones de Diseño.*

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores. Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia, que han demostrado que funcionan (Gracia, 2005) y que facilitan la comunicación entre diseñadores y el aprendizaje al programador inexperto, pues permiten establecer parejas problema-solución que agilizan la implementación.

Los patrones de diseño son una realidad que ayudan a no cometer los mismos errores reiteradas veces y a realizar un diseño flexible, modulado y reutilizable.

Los patrones de diseño tienen como características (Pressman, 2001):

- *Son soluciones concretas:* Proponen soluciones a problemas concretos, no son teorías genéricas.
- *Son soluciones técnicas:* Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- *Se utilizan en situaciones frecuentes:* Ya que se basan en la experiencia acumulada la resolver problemas reiterativos.
- *Favorecen la reutilización de código:* Ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.

Cuando se tiene en cuenta un patrón, este no se refleja en el código. Sin embargo la tendencia actual es incluir en el nombre de las clases el nombre del patrón, para así facilitar la comunicación y el entendimiento entre los desarrolladores, a pesar de que su uso no determine su reutilización, pues las clases son concretas para problemas concretos, y no se pueden aplicar fácilmente a otro problema.

#### *1.3.4.3.1 Patrones GRASP.*

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Larman, 2004). GRASP es un acrónimo de General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para sugerir la importancia de usar estos principios para diseñar con éxito el software orientado a objetos.

Existen nueve patrones GRASP. Cada uno de ellos indica una solución a una problemática. En la siguiente tabla se muestran estos patrones, con los problemas que resuelven y las soluciones que se le dan.

Tabla 1: Patrones GRASP (Larman, 2004)

| Nombre | Solución | Problema que resuelve |
|---|---|---|
| Experto | Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. | ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos? |
| Creador | Asignarle a una clase la responsabilidad de crear una instancia de otra clase en casos diferentes. | ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? |
| Bajo acoplamiento | Asignar una responsabilidad para mantener bajo acoplamiento. | ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? |
| Alta cohesión | Asignar una responsabilidad de modo que la cohesión siga siendo alta. | ¿Cómo mantener la complejidad dentro de límites manejables? |
| Controlador | Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una de las clases. | ¿Quién debería encargarse de atender un evento del sistema? |
| Polimorfismo | Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán mediante operaciones | ¿Cómo manejar las alternativas basadas en el tipo? ¿De qué manera crear componentes de software conectables? |

| | | |
|---|---|---|
| | polimórficas a los tipos en que el comportamiento presenta variantes. | |
| Fabricación Pura | Asignar un conjunto altamente cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema. | ¿A quién asignar la responsabilidad cuando se está desesperado y no se quiere violar los patrones Alta Cohesión y Bajo Acoplamiento? |
| Indirección | Se asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, y estos no terminen directamente acoplados. | ¿A quién se asignarán las responsabilidades a fin de evitar el acoplamiento directo? ¿De qué manera se desacoplarán los objetos de modo que se obtengan un Bajo Acoplamiento y se observe un alto potencial de reutilización? |
| No Hables con Extraños | Se asigna la responsabilidad a un objeto directo del cliente para que colabore con un objeto indirecto, de modo que el cliente no necesite saber nada del objeto indirecto | ¿A quién asignar las responsabilidades para evitar conocer las estructuras de los objetos indirectos? |

Es importante entender y ser capaces de aplicar estos principios durante la creación de los diagramas de interacción porque un desarrollador de software con poca experiencia en la tecnología de objetos necesita dominar estos principios tan rápido como sea posible; constituyen la base de cómo se diseñará el sistema (Larman, 2004).

#### *1.3.4.3.2 Patrones GOF.*

Estos patrones fueron creados por Gamma, Helm, Johnson y Vlissides (Gang of Four). Se clasifican según su propósito en:

***Patrones de Creación*** (Gamma, y otros, 1995)

Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla.

Según donde se tome dicha decisión se pueden clasificar los patrones de creación en: patrones de creación de clases (la decisión se toma en los constructores de las clases y usan la herencia para determinar la creación de las instancias).

Los patrones de creación son:

- *Abstract Factory (Fábrica Abstracta):* Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- *Builder (constructor virtual):* Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto. "Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones". (Gracia, 2005)
- *Factory Method (Método de fabricación):* Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que se crea.
- *Prototype (Prototipo):* Crea nuevos objetos clonándolos de una instancia ya existente.

- *Singleton (Instancia única):* Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

***Patrones Estructurales*** (Gamma, y otros, 1995)

Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y éstas están determinadas por las interfaces que soportan los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos.

Los patrones estructurales son:

- *Adapter (Adaptador):* Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- *Bridge (Puente):* Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente. (Gracia, 2005)

- *Composite (Objeto compuesto):* Permite tratar objetos compuestos como si de un objeto simple se tratase.
- *Decorator (Envoltorio):* Añade funcionalidad a una clase dinámicamente.
- Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- *Flyweight (Peso ligero):* Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- *Proxy:* Mantiene un representante de un objeto.

***Patrones de Comportamiento*** (Gamma, y otros, 1995)

Plantea la interacción y cooperación entre las clases. Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal.

Los patrones de comportamiento son:

- *Chain of Responsibility (Cadena de responsabilidad):* Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
- *Command (Orden):* Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- *Interpreter (Intérprete):* Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
- *Iterator (Iterador):* Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.
- *Mediator (Mediador):* Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- *Memento (Recuerdo):* Permite volver a estados anteriores del sistema.
- *Observer (Observador):* Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.
- *State (Estado):* Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.

- *Template Method (Método plantilla):* Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.
- *Visitor (Visitante):* Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.
- *Strategy (Estrategia):* Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.

#### *1.3.4.4 Calidad del Diseño*

La calidad del producto dependerá del nivel de calidad obtenido en el diseño del mismo. La calidad en el diseño es el grado en que las especificaciones de diseño (especificaciones acerca de los distintos componentes del producto) responden a las expectativas del consumidor. (Berlinches Cerezo, 2004)

Pressman (Pressman, 2001) sugiere algunas características que sirven como guía para la evaluación de un buen diseño, entre las que están:

- El diseño deberá ser una guía legible y comprensible para aquellos que generan código y para aquellos que comprueban y consecuentemente, dan soporte al software.
- El diseño deberá proporcionar una imagen completa del software, enfrentándose a los dominios de comportamiento, funcionales y de datos desde una perspectiva de implementación.
- Presentar una organización jerárquica que haga un uso inteligente del control entre los componentes del software.
- Ser modular, es decir, se debe hacer una partición lógica del software en elementos que realicen funciones y subfunciones específicas.
- Contener abstracciones de datos y procedimientos.
- Producir módulos que presenten características de funcionamiento independiente.
- Conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.

#### *1.3.4.5 Métricas para evaluar el Diseño de Software.*

Se define métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. (Aguilar, y otros, 2008)

Definir una métrica específica que proporcione una medida completa de la complejidad del software es bastante difícil. Se han propuesto numerosas métricas, pero todas tienen diferentes puntos de vista, no siempre determinan el mismo valor cuando se mide la complejidad del software. No obstante, su uso es esencial, pues hasta cierto nivel, garantizan un producto con mayor calidad.

#### *1.3.4.5.1 Métricas para Sistemas Orientados a Objetos.*

Varios autores proponen métricas para evaluar diseño. A continuación se mencionan algunas de ellas (Olmedilla Arregui, 2005):

***Métricas propuestas por Lorenz y Kidd.***

Estos autores las proponen basadas en las clases. Las separan en cuatro categorías: tamaño, herencia, valores internos y valores externos. También proponen otras orientadas al tamaño y la complejidad de las clases.

Las métricas orientadas al tamaño se centran en contar los atributos y operaciones de cada clase y los valores para el sistema como un todo son:

- Número de Métodos de Instancia Públicos (PIM).
- Número de Métodos de Instancia (NIM).
- Número de Variables de Instancia (NIV).
- Tamaño de clase (TC).
- Tamaño medio de operación (TMO).
- Número de escenarios (NE).
- Número de clases claves (NCC).
- Número de subsistemas (NSUB).

Las que se basan en la herencia para ver en la forma que las operaciones se reutilizan en la jerarquía de clases son:

- Número de operaciones redefinidas para una subclase (NOR).
- Número de operaciones añadidas por una subclase (NOA).
- Índice de especialización (IES).
- Número de Métodos Heredados (NMI).

***Métricas propuestas por Chidamber y Kemerer (CK).***

Es uno de los conjuntos de métricas más difundidos y conocidas como las CK, también se les llama MOOSE. Adoptan tres criterios a la hora de definirlas: capacidad de satisfacer propiedades analíticas, aspecto intuitivo a los profesionales y facilidad para su recogida automática. Definen los siguientes conceptos:

- Métodos ponderados por clase (MPC).
- Profundidad del árbol de herencia (PAH).
- Número de hijos (NDH).
- Acoplamiento entre clases objeto (AEC).
- Respuesta para una clase (RPC).
- Carencia de cohesión en los métodos (CCM).

***Métricas propuestas por Li y Henry***.

Se consideran como una extensión del MOOSE con otras métricas de tamaño, acoplamiento y diseño, también proponen un sistema de predicción de reutilización y mantenibilidad. Ellos modificaron y ampliaron las CK, además de añadir algunas nuevas. Tienen un objetivo claro; la medición y mejora de la mantenibilidad del software OO.

Las métricas tomadas de CK son:

- Métodos ponderados por clase (MPC).
- Árbol de profundidad de herencia (PAH).
- Número de descendientes (NDH).
- Respuesta para una clase (RPC).
- Carencia de cohesión en los métodos (CCM).

Definen otras cinco métricas para el acoplamiento y tamaño:

- Acoplamiento por paso de mensaje (APM).
- Acoplamiento por abstracción de datos (AAD).
- Número de métodos locales (NML).
- Tamaño 1.

- Tamaño 2.

Estas métricas permiten evaluar el diseño, así como saber con anterioridad a qué clase, subsistema y módulo se le debe prestar más atención y designar más recursos para su implementación.

Es importante destacar que todo el proceso estudiado anteriormente se aplica apoyándose en una metodología de desarrollo de software, que es la que guía el desarrollo evolutivo de la creación de un software.

**1.4 Metodologías de Desarrollo de Software.**

Piattini define la metodología como un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software (Piattini, 1996).

Las metodologías de desarrollo de software consisten en fases o etapas descompuestas en subfases, módulos, etapas, pasos; estrategia que beneficia a los desarrolladores durante la elección de las técnicas a utilizar en cada estado del proyecto, facilitando la planificación, gestión, control y evaluación de los proyectos.

El autor expresa que "una metodología representa el camino para desarrollar software de una manera sistemática". (Piattini, 1996)

Debido a la importancia de las metodologías como guías en el desarrollo de software, se han desarrollado varias de estas, dedicadas a diferentes tipos de proyectos. Por esto, es necesario realizar un estudio de las mismas para poder determinar cual es la más ajustada al proyecto en cuestión.

***Extreme Programming (XP).***

La Programación Extrema forma parte de las metodologías ligeras empleadas en proyectos cortos y con poco personal de desarrollo, donde siempre está presente el cliente o usuario final. Se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado (Fernández Escribano, 2002). XP es recomendada para aumentar la velocidad de desarrollo de un producto, el cual fundamenta su desarrollo en los casos de prueba y en las historias de usuario (Beck, y otros, 2004). Para mas información ver tesis (Piñeiro Pérez, y otros, 2007).

***Scrum.***

La mayoría de las prácticas de Scrum son compatibles con XP. Scrum trabaja con iteraciones de 30 días llamadas sprint y reuniones cotidianas de 15 minutos. Su práctica es trabajar con un solo representante, el dueño del producto final, aunque últimamente se estila crear un grupo de clientes finales para darle agilidad al proceso (Larman, 2003). Para mas información ver tesis (Fernández Carballo., y otros, 2007).

***Rational Unified Process (RUP).***

Es una metodología que ayuda a desarrollar y desplegar software fragmentando su desarrollo en fases y ciclos, de forma iterativa e incremental, guiado por casos de uso y basado en la arquitectura. Es una colección de buenas prácticas experimentadas en la ingeniería de software que provee a sus usuarios de una fundamentación arquitectónica fuerte que permite construir el software con las necesidades requeridas.

Puede especializarse para gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial del Proceso Unificado, sus desarrollos fueron paralelos (Jacobson, y otros, 2000).

RUP presenta algunas características importantes:

- Dirigido por casos de uso.
- Iterativo e incremental.
- Centrado en la arquitectura.
- Adaptable a proyectos de largo plazo.
- Genera muchos artefactos, convirtiéndose en una metodología muy usada para lograr una certificación de desarrollo de software.
- Destaca la importancia de lograr una buena captura de requisitos.
- Maneja como actividades paralelas la gestión de la calidad, la gestión de riesgos.
- Trabaja con la herramienta Rational Rose y se basa en UML como lenguaje orientado a objetos para visualizar, especificar, construir y documentar los artefactos.

RUP presenta fases e hitos que completan el período de desarrollo de un software.

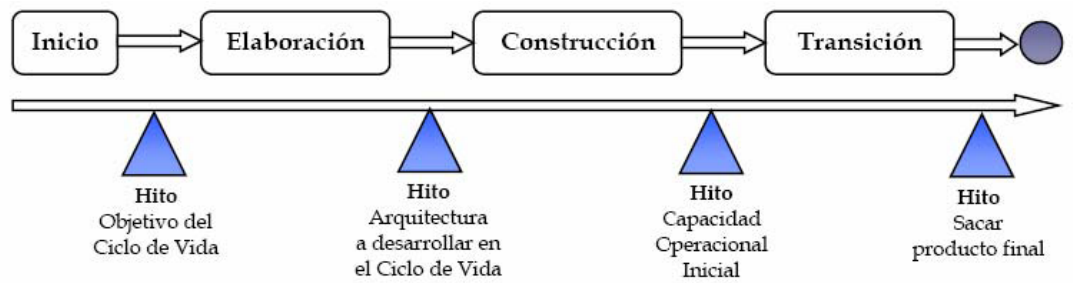



**Figura 1. Fases e Hitos que propone RUP.**

Los flujos de trabajo que propone RUP son los que se presentan a continuación. En este trabajo se desarrollan los tres primeros:

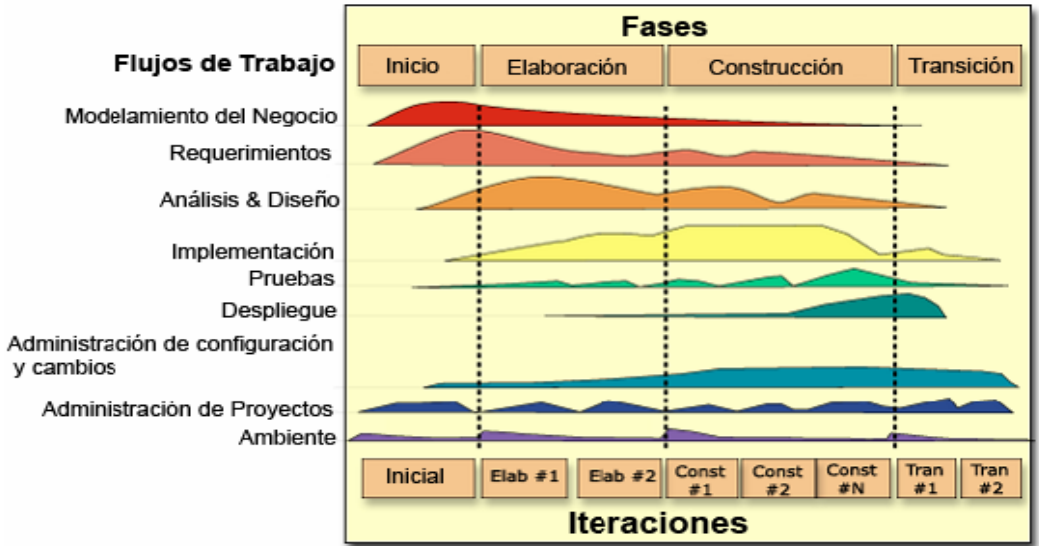



**Figura 1.2. Flujos de trabajo que propone RUP.**

### *1.4.1 Justificación de la metodología seleccionada.*

Se realizó un estudio de las metodologías más usadas para el desarrollo de software por el equipo de desarrollo del proyecto. De acuerdo con las características y complejidad de este proyecto de software, se decidió trabajar con RUP.

Esta metodología hace énfasis en una buena captura de los requisitos, lo que a su vez representa una debilidad en otras metodologías, fundamentalmente las ágiles. Además, el producto que se desea desarrollar es muy grande, por lo que las metodologías ágiles no son las más indicadas, pues se enfocan más en captar con rapidez las necesidades iniciales para proceder de inmediato con la

implementación, lo cual no es factible para un software de la envergadura de un ERP, compuesto por tantos módulos, y esto provocaría fallas más adelante por un inadecuado levantamiento de requisitos.

Otra razón por la que se selecciona RUP como metodología a usar, es que el proyecto estará compuesto por gran cantidad de personal, que variará con el paso del tiempo, producto de la dinámica de la universidad, y será necesario generar la mayor cantidad de documentación posible para poder capacitar al nuevo personal, además de contribuir con la reusabilidad entre los módulos lo mayor posible.

**1.5 Lenguaje de modelado.**

La comunidad del software precisa de una forma de comunicar sus modelos, no solo entre los miembros de un proyecto, sino a todas las personas involucradas en el, necesita un lenguaje para proporcionar un marco en el que desarrolladores individuales puedan pensar y analizar (Jacobson, y otros, 2000). Es por esto que surgen numerosos lenguajes de modelado. Algunos de ellos son:

***Process Modeling Language (PML):*** Es un lenguaje de modelado orientado a objetos diseñado para describir el comportamiento del sistema físico mediante estructuras de representación modulares (clases de modelado). Las clases PML representan conceptos físicos que son familiares al modelador. El conocimiento físico declarado por las clases se utiliza para analizar los modelos estructurados, obteniéndose de manera automatizada la representación matemática de las dinámicas de interés. (Ramos González)

***Modélica:*** Surge a mediados de la década del noventa debido a uno de los grandes problemas con todas las herramientas de modelado que iban surgiendo, la falta de transportabilidad de los modelos desarrollados que eran dependientes del entorno utilizado. Fue definido por un grupo de muchos expertos de diferentes dominios de la ingeniería, así como de la gran mayoría de desarrolladores de los lenguajes de modelado orientado a objetos que habían ido surgiendo, que tenían por objetivo crear un lenguaje de modelado capaz de expresar la conducta de modelos de un amplio abanico de campos de la ingeniería y sin limitar a los modelos a una herramienta comercial en particular. Se puede considerar a Modélica no solo como un lenguaje de modelado sino como una especificación de dominio público que permite el intercambio de modelos. Es un lenguaje de modelado que no tiene propietario y su nombre es una marca registrada de la “Modelica Association” que es la responsable de la publicación de la especificación del lenguaje Modelica (Dormido, 2006).

***Lenguaje de Modelado para Realidad Virtual (VRML):*** Se utiliza para describir simulaciones interactivas de participantes múltiples, esto es, mundos virtuales enlazados de manera global vía Internet e hiperenlazados con el World Wide Web. Puede utilizarse para especificar todos los aspectos del despliegue del mundo virtual, su interacción y trabajo con redes internas. La intención de sus diseñadores es hacer del VRML el lenguaje estándar para la simulación interactiva, dentro del World Wide Web (Peru).

***Lenguaje Unificado de Modelación (UML):*** Es un lenguaje gráfico para visualizar, especificar, construir, documentar y comunicar los artefactos de un sistema de software (Jacobson, y otros, 2000). Es utilizado para modelar la información del sistema basado en concepto de objetos. Especifica varios diagramas que permiten crear artefactos durante el proceso de desarrollo del software.
UML es un lenguaje flexible de modelado que permite definir modelos de análisis y modelos del diseño (Larman, 2004).

Beneficios de UML:

- Soporta tecnología orientada a objetos.
- Soporta RUP.
- Los errores son tratables en todas las etapas del desarrollo del software.
- Especifica, mediante los diagramas, cómo se estructura y se comporta el sistema.
- Permite obtener un “plano del sistema”.
- Reduce los costos y el tiempo de desarrollo.
- Representa una colección de las mejores prácticas de ingeniería que tienen una probación exitosa en la modelación de sistemas largos y complejos.

### *1.5.1 Justificación de la selección del lenguaje de modelado.*

Este lenguaje es el más utilizado a nivel mundial para el modelar los artefactos creados durante el proceso de desarrollo de software. Ya que fue desarrollado conjuntamente con la metodología RUP, responde a todas sus necesidades combinándose para formar una perfecta elección de un equipo de desarrollo que decida usar RUP.

Además, es importante destacar que los demás lenguajes de modelado no responden a las necesidades del proyecto, es por ello que para la realización de este trabajo se adopta UML como lenguaje de modelado.

### 1.6 Herramientas Case

Hoy en día, la Ingeniería de Software cuenta con una serie de herramientas automatizadas destinadas a diferentes propósitos. La utilización de herramientas CASE (Computer Aided Software Engineering), ingeniería de software asistida por computadora, se ha difundido ampliamente en el ámbito del desarrollo de software.

La ingeniería de sistemas asistida por ordenador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de las CASE para automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas (Kendall, 1997).

Acerca de las herramientas automatizadas para la ingeniería de software se puede decir que:

- Permiten un mayor control de proyectos complejos.
- Permiten reducir costos y retrasos en la liberación de un proyecto.
- Permiten una mayor comunicación en equipos de trabajo.
- Ayudan a determinar la complejidad del proyecto y esfuerzos necesarios.

Estas herramientas permiten a los desarrolladores modelar y documentar sus artefactos, cubriendo el ciclo de vida del proceso de desarrollo de software. A pesar de ello, presentan diferencias que contribuyen a seleccionar una herramienta según las características del proyecto:

***Rational Rose:*** Esta herramienta que domina en el mercado mundial, fue desarrollada por los creadores de UML y emplea este lenguaje para modelar. Es una de las más potentes desarrolladas hasta el momento, pero consume muchos recursos, lo que hace que si no se presenta la tecnología adecuada, resulte incómodo trabajar con ella.

***Visual Paradigm:*** Esta herramienta es multiplataforma y está diseñada para desarrollar software con programación orientada a objetos. Es una herramienta de desarrollo que se integra al IDE de Eclipse. Su licencia cuesta más de 2300 dólares y además, es importante destacar que posee librerías privadas, lo que le imposibilita a los programadores utilizar sus propias librerías para generar código de acceso a datos.

***Enterprise Architect (EA):*** Es una herramienta CASE para el diseño y construcción de sistemas de software.

Beneficios de EA (Sparxsystems, 2008):

- Soporta la especificación de UML 2.0, que describe un lenguaje visual por el cual se pueden definir mapas o modelos de un proyecto, y los diagramas de UML 2.1.
- Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento.
- También provee soporte para pruebas, mantenimiento y control de cambio.
- Sincroniza código y elementos del modelo.
- Diseña y genera elementos de base de datos.
- Sus modelos se pueden exportar en formato RTF para una personalización y presentación final.
- Sus plantillas soportan las características de los elementos de los modelos de EA y datos extendidos (pruebas, riesgos, recursos, cambios), además de soportar formatos como encabezados, índices, etc.
- Permite exportar un modelo completo o una rama simple del modelo a páginas web en HTML.
- Realiza ingeniería directa e inversa de código fuente en Action Script, C++, C#, Delphi, Java, Python, PHP, VB.NET, Visual Basic.
- Tiene una alta capacidad de sincronización de código.
- Posee soporte para Corba también disponible como plug-in libre.
- Posee plug-ins para vincular EA a Visual Studio.NET y Eclipse.
- Posee una interfaz de usuario intuitiva.
- Realiza ingeniería inversa para sistemas de Base de Datos muy populares como Oracle, SQL Server, MySQL, Access, PostgreSQL, etc.
- Permite generar tablas del Modelo de Base de Datos, columnas, claves, llaves foráneas, etc.
- Permite generar los scripts DLL para crear las estructuras de Base de Datos.
- Posibilita el desarrollo distribuido, es decir, es multiusuario.
- Soporta diferentes repositorios basados en DBMS, incluyendo Oracle, SQL Server, My SQL, PostGreSQL.
- Soporta importar/exportar archivos XMI para manejar la distribución y actualización de frameworks y otros paquetes basados en la estructura del modelo.

- Soporta control de versiones de repositorios.
- Soporta importar archivos .JAR en java y ensamblados .NET.
- Posee un perfil incorporado para XSD para simplificar el desarrollo de esquemas XML usando UML.
- Perfil UML para XSD disponible para descarga libre.
- Generar esquemas complejos de XML de modelos UML
- Ingeniería Reversa de esquema XML a Modelos UML.
- Ingeniería Directa de esquema XML desde Modelos UML
- Posee gran velocidad, estabilidad y buen rendimiento.

### *1.6.1 Herramienta utilizada en la propuesta de solución*

La herramienta a utilizar en la propuesta de solución, es el Enterprise Architect. Es muy fácil de usar, ligera, y brinda la posibilidad de exportar documentos. Su licencia cuesta menos de 400 dólares, lo que la hace más barata que la del Visual Paradigm, y posee una característica fundamental en el desarrollo de este proyecto: es multiusuario, lo que garantiza que varios desarrolladores puedan trabajar a la vez sobre la herramienta.

## 1.7 Conclusiones parciales.

Se realizó un estudio del estado del arte de los sistemas de software ERP, logrando sintetizar sus objetivos, características, ventajas y desventajas de su desarrollo, situación que permite comprender su importancia y la necesidad de desarrollar un sistema ERP para el país, con sus propias características, de forma que se optimicen los procesos en las empresas cubanas.

Se realizó un estudio que permitió fundamentar el uso de:

- El Proceso Unificado de Racional como metodología de desarrollo de software con UML como lenguaje de modelado.
- La herramienta Enterprise Architect para apoyar a los desarrolladores a la hora de modelar el sistema.
- Las técnicas para capturar los requisitos del sistema, de forma que se logre tener un conocimiento suficiente como para poder seleccionar cuales deben realizarse para este trabajo de investigación.

- Los patrones de casos de uso y de diseño, como mecanismos de ayuda para los desarrolladores, ya que proporcionan la solución a un problema que se puede presentar, y que posee una solución ya probada.

CAPÍTULO 2: MODELADO DE NEGOCIO Y SISTEMA.

## 2.1 Introducción.

En este capítulo se presenta parte de la propuesta de solución de la investigación. Apoyándose en la metodología usada, se desarrollan varias actividades y artefactos que permiten darle cumplimiento al problema planteado, y de esta forma, dominar el negocio como comienzo de un proceso que continuará con la especificación de los requisitos y con el diseño del sistema.

## 2.2 Modelado del Negocio.

El modelo de negocio representa y describe detalladamente los procesos del negocio, de forma que constituye el artefacto esencial para comprenderlos. “Está soportado por dos tipos de modelos UML: modelo de casos de uso y modelo de objetos” (Jacobson, y otros, 2000). Por tanto, en este trabajo se desarrollan estos artefactos, basados en las técnicas descritas anteriormente.

### *2.2.1 Técnicas usadas para el caso propuesto.*

Desarrollar una buena ingeniería de requisitos y captarlos adecuadamente implica disminuir los costos y retrasos del proyecto, mejorar la calidad del software, la comunicación entre equipos y aumentar la satisfacción de los usuarios finales.

Para lograr la comprensión entre ambas partes, clientes y desarrolladores, y lograr dominar el negocio en cuestión, se desarrollaron técnicas para capturar los procesos que se deben modelar luego. “Una buena comprensión de los procesos del negocio es importante para construir los sistemas correctamente” (Rational, 2003).

Para este trabajo se usaron la arqueología de documentos, el desarrollo conjunto de aplicaciones y las tormentas de ideas.

### *2.2.2 Elicitación y análisis y negociación de requisitos.*

Primeramente se llevó a cabo un estudio de la bibliografía que se obtuvo del funcionamiento de las empresas clientes (arqueología de documentos). Esta etapa es muy importante porque en ella los analistas comienzan a conocer la terminología de los clientes, se forman una idea del negocio y comienzan a detectar donde se debe trabajar con los clientes para lograr la comprensión del negocio.

Luego se desarrolló la técnica JAD varias veces, empleada por un amplio grupo de desarrolladores y clientes, donde se analizó el negocio y se documentó todo lo mejor posible.

Al concluir esta etapa, se procedió a las tormentas de ideas entre desarrolladores, a analizar toda la información que se tenía, y generar entrevistas con los clientes, con el fin de captar la información lo más exacta, clara y abarcadora posible, de forma tal que se lograra un refinamiento de los requisitos capturados hasta el momento.

### *2.2.3 Procesos de Negocio.*

Llevar a cabo un control de la gestión de inventario es una tarea fundamental para cualquier empresa, ya que esto influye en el grado de organización y aprovechamiento que se tenga del almacén y de los recursos de la empresa.

Se identificaron algunos procesos básicos dentro de la gestión de inventario, que se muestran a continuación. Es necesario destacar que para el módulo de Administración no se presenta negocio, pues como se explicó anteriormente, su objetivo es gestionar el acceso al sistema, por lo tanto, este módulo se analiza en el modelado del sistema, más adelante.

*Recepcionar productos:* Proceso mediante el cual el almacenero recepciona y cuenta la mercancía que le entra. Luego elabora el documento Listado para Recepción Ciega (LRC) (Ver Anexo 1), se compara con la Factura (F) (Ver Anexo 2) que posee el departamento de economía y se elabora el Informe de Recepción (IR) (Ver Anexo 3).

*Devolver productos:* Proceso que inicia el responsable de sección cuando detecta productos defectuosos o faltantes tras una solicitud en el almacén. Se devuelven los productos y se emite el Vale de Entrega o Devolución (VED) (Ver Anexo 4).

*Realizar reclamación:* Este proceso se inicia cuando el responsable de sección detecta que recibió productos con defectos o faltantes de productos. Entonces realiza una reclamación emitiendo el Informe de Reclamación (IRecl.) (Ver Anexo 5).

*Realizar despiece:* Proceso que inicia el responsable de sección solicitando el despiece de algún producto, es decir, solicita un producto para fragmentarlo en partes. Se les da salida a los productos a despiezar y entrada a los productos resultantes, consultándose las Fichas Técnicas (FT) (Ver Anexo 6)

de despiece de los productos a despiezar y generándose un Reporte de Elaboración (RE) (Ver Anexo 7).

*Solicitar productos:* Proceso en el que el responsable de sección solicita productos al almacén mediante la Solicitud de Entrega (SE) (Ver Anexo 8) y estos son despachados generando un VED.

*Mover productos entre secciones:* Proceso que se inicia cuando se realiza una solicitud de productos, y el almacén no puede satisfacer la demanda. En este caso el almacenero realiza un movimiento de productos de otra sección, y los despacha.

### *2.2.4 Actores del Negocio.*

Un actor expresa un rol, no una persona. Puede ser un individuo, sistema, entidad, que interactúa con el negocio y que se beneficia de esto. (Rational, 2003)

| Actor | Descripción |
|---|---|
| Responsable de sección | Es la persona con la autoridad para solicitar productos, autorizar despieces y recepciones, realizar reclamaciones y devoluciones de productos. |

### *2.2.5 Trabajadores del Negocio.*

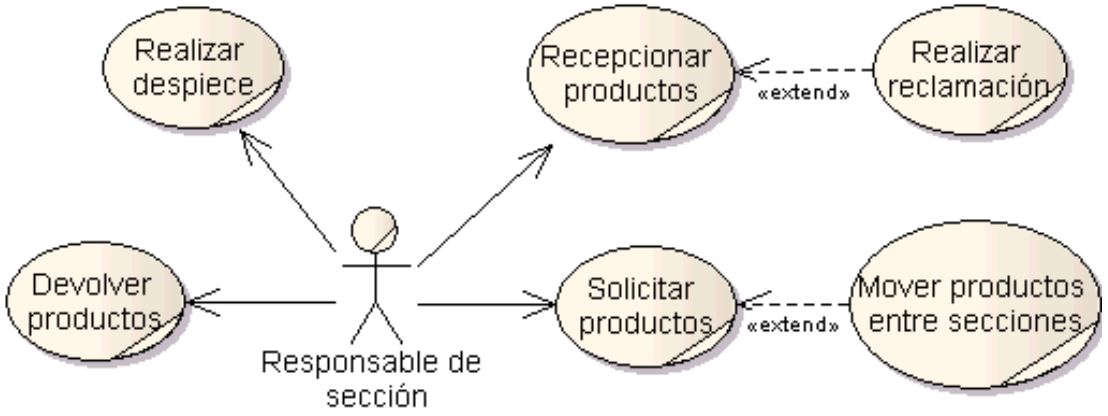
El trabajador del negocio es una abstracción de un humano o un sistema de software que representa un rol que realiza las actividades de los casos de uso. (Rational, 2003)

| Trabajador | Descripción |
|---|---|
| Normador | Persona que define las FT y solicita los productos al almacén para realizar los despieces. |
| Personal de economía | Persona que lleva a cabo la contrapartida contable de las operaciones que se efectúan en el almacén, es decir, emite Comprobantes Contables (CC). |
| Almacenero | Persona que le da entrada y salida a los productos del almacén y maneja varios informes que avalan los mismos. |
| Proveedor | Entidad externa que provee al almacén de productos pedidos |

| | |
|---|---|
| | previamente. |

### *2.2.6 Diagrama de casos de uso del negocio.*

El Diagrama de Casos de Uso del Negocio describe parte del modelo de CUN y muestra los casos de uso y los actores, con sus relaciones. Es decir, brinda información clara sobre los actores y procesos del negocio.




### *2.2.7 Descripción de los casos de uso del negocio.*

| **Caso de Uso:** | **Realizar despiece** |
|---|---|
| **Actores:** | Responsable de sección |
| **Trabajadores:** | Normador, Almacenero, Personal de economía. |
| **Resumen:** | El CU se inicia cuando el Responsable de Sección solicita productos para la realización de un despiece. Tras consumarse el despiece se ubican en el almacén y se actualiza la existencia del producto. |
| **Precondiciones:** | Si el área proveedora no pertenece a la empresa se requiere de una autorización. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| 1. El Responsable de sección solicita el | 2. El Normador solicita la extracción de los productos del almacén con una SE.<br>3. El almacenero despacha los productos y elabora un VED, |

| | |
|---|---|
| despiece de uno o varios productos. | enviando copias a economía.<br>4. El despiece se realiza en un área específica para ello.<br>5. El Normador consulta la FT para ver las normas establecidas y refleja los resultados del despiece en el RE. |
| 5. El Responsable de sección chequea el RE y lo firma. | 6. El Normador entrega los productos resultantes del despiece y una copia del RE al almacén.<br>7. El Almacenero actualiza la Tarjeta de Estiba (TE) (Ver Anexo 9) y envía copia de RE a economía.<br>8. El Personal de economía emite el CC. |
| **Flujo Alternativo:** | |
| | 5.1. Si no existe la FT de despiece del producto, el Normador crea una FT y decide las normas del despiece del producto. Luego el flujo de actividades sigue igual. |
| **Poscondiciones** | Se realiza un despiece, que puede satisfacer solicitudes de productos que se obtienen como resultado del producto despiezado. |
| **Ver Diagrama de actividades en Anexo 10.** | |

| | |
|---|---|
| **Caso de Uso:** | **Devolver productos** |
| **Actores:** | Responsable de sección |
| **Trabajadores:** | Almacenero, Personal de economía, Proveedor |
| **Resumen:** | El caso de uso se inicia cuando el Responsable de sección chequea los productos que recibió, y por alguna razón los quiere devolver. Entonces se realiza la devolución de los productos. |
| **Precondiciones:** | Debe contar con los documentos que justifican una reclamación: VED**,** F, IR.<br>Debe estar comprendido en el período permitido para realizar la devolución.<br>Debe haber problemas con los productos entrantes, como faltantes o defectos. |
| **Flujo Normal de Eventos** | |

| Acción del Actor | Respuesta del Negocio |
|---|---|
| 1. El Responsable de Sección entrega los productos y el IR de la mercancía. | 2. El Proveedor recibe el informe, chequea que la devolución es en el período de tiempo permitido y autoriza la devolución.<br>3. El Almacenero recibe los productos, elabora el VED y envía copias a economía.<br>4. Actualiza las TE.<br>5. El Personal económico emite un CC. |
| **Flujo Alternativo:** | |
| | 2.1. En caso de estar fuera de los 60 días permitidos para la devolución, se cancela la operación. |
| **Poscondiciones** | Se actualizan las existencias de los productos. |
| **Ver Diagrama de actividades en Anexo 11.** | |

| | |
|---|---|
| **Caso de Uso:** | **Recepcionar productos** |
| **Actores:** | Responsable de sección. |
| **Trabajadores:** | Almacenero, Personal de economía, Proveedor. |
| **Resumen:** | El caso de uso se inicia cuando el Responsable de sección autoriza la recepción de productos. Se realiza el proceso de recepción por parte del almacenero. En caso de haber problemas se realiza una reclamación (**Ver CU: Realizar Reclamación**). |
| **Precondiciones:** | Debe haber una solicitud de productos previa.<br>Los productos deben estar activados en economía.<br>La recepción debe efectuarse sin conocimiento de las cantidades de productos que aparecen en la factura. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Negocio** |

| | |
|---|---|
| 1. El Responsable de sección autoriza la recepción de productos. | 2. El Proveedor se presenta en el departamento de economía con la F.<br>3. El Personal de economía recibe la F y emite el documento LRC, el cual se entrega al área de recepción para que proceda al conteo físico de la mercancía.<br>4. El Almacenero recibe los productos y el LRC, donde detalla la cantidad de productos que recibió y la envía a economía.<br>5. El Personal de economía aprueba la recepción.<br>6. El Almacenero recibe los productos, elabora el IR y actualiza las TE.<br>7. El Almacenero envía copias a economía.<br>8. El Personal económico emite un CC. |
| **Flujo Alternativo:** | |
| | 5.1. Si los documentos no coinciden se realiza una reclamación (**Ver CU**: Realizar Reclamación). |
| **Poscondiciones** | Se actualizan las existencias de los productos. |
| **Ver Diagrama de actividades en Anexo 12.** | |

| | |
|---|---|
| **Caso de Uso:** | **Realizar reclamación** |
| **Actores:** | Responsable de sección |
| **Trabajadores:** | Proveedor, Almacenero, Personal de economía |
| **Resumen:** | El caso de uso se inicia cuando el Responsable de sección chequea los productos que recibió y nota problemas con estos. Entonces realiza la reclamación de los productos. |
| **Precondiciones:** | Debe contar con los documentos que justifican una reclamación: Informe de Reclamación y de Recepción.<br>Debe estar comprendido en el período permitido para realizar la reclamación.<br>Debe haber problemas con los productos entrantes, como faltantes o defectos. |
| **Flujo Normal de Eventos** | |

| Acción del Actor | Respuesta del Negocio |
|---|---|
| 1. El Responsable de sección emite un IRecl y lo presenta junto con el IR de la mercancía a reclamar. | 2. El Proveedor recibe los informes y chequea fechas para ver si tiene derecho a reclamar.<br>3. El Proveedor confirma el problema y los envía al Almacenero.<br>4. El Almacenero compara los informes, anexa al IR una copia del IRecl y envía copia a Economía.<br>5. El Personal de economía emite CC. |
| **Flujo Alternativo:** | |
| | 2.1. En caso de presentar la reclamación con más de 90 días después de haber hecho la recepción de los productos, el proveedor detecta que no es válida y se cancela la operación. |
| **Poscondiciones** | Se realiza una reclamación. |
| **Ver Diagrama de actividades en Anexo 13.** | |

| **Caso de Uso:** | **Solicitar Productos** |
|---|---|
| **Actores:** | Responsable de sección |
| **Trabajadores:** | Personal de economía, Almacenero |
| **Resumen:** | El Responsable de una sección solicita productos al almacén mediante una SE. El Almacenero chequea si cuenta con los productos para satisfacer el pedido, y los despacha, sino, realiza un movimiento entre secciones y despacha los productos. |
| **Precondiciones:** | Se debe presentar el SE |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| 1. El Responsable de sección solicita productos al almacén con una SE. | 2. El Almacenero recibe el SE y verifica que puede satisfacer la solicitud.<br>3. El Almacenero elabora un VED con copias.<br>4. El Almacenero entrega productos al solicitante |

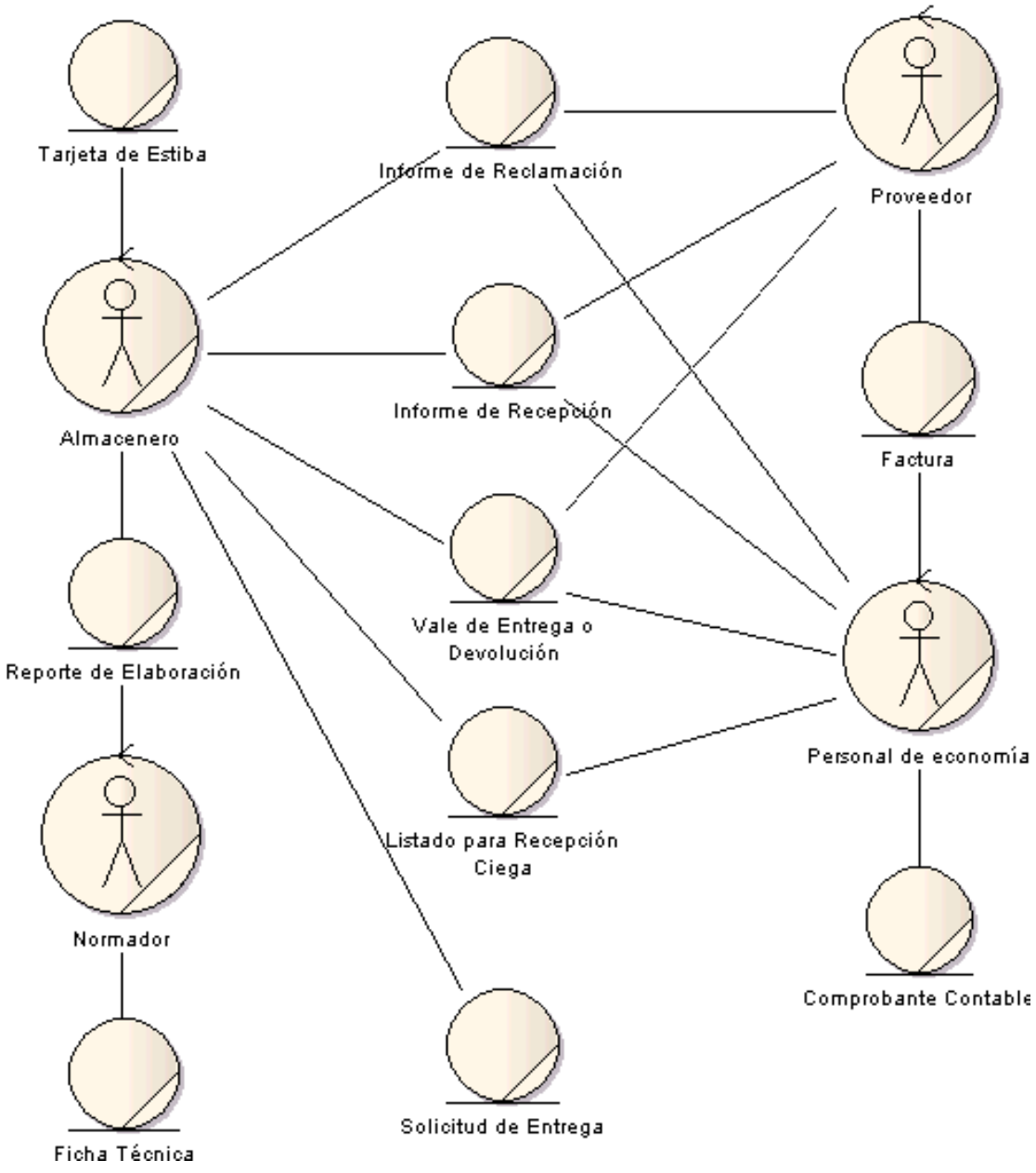
| | |
|---|---|
| | con una copia.<br>5. El Almacenero actualiza las TE. |
| 6. Recibe productos. | 6. El Almacenero envía otra copia a economía.<br>7. El Personal económico emite un CC. |
| **Flujo Alternativo:** | |
| | 2.1. El Almacenero detecta que no puede satisfacer la solicitud, entonces se realiza un movimiento entre secciones. (**Ver CUN Mover productos entre secciones**). |
| **Pos condiciones** | Se realiza una solicitud de productos |
| **Ver Diagrama de actividades en Anexo 14.** | |

| **Caso de Uso:** | **Mover productos entre secciones** |
|---|---|
| **Actores:** | Responsable de sección |
| **Trabajadores:** | Personal de economía, Almacenero |
| **Resumen:** | El Responsable de una sección solicita productos al almacén mediante una SE. El Almacenero verifica que no puede satisfacer el pedido, y realiza un movimiento de productos entre secciones. |
| **Precondiciones:** | Se debe presentar el SE. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Negocio** |
| 1. El Responsable de sección solicita productos al almacén con una SE. | 2. El Almacenero recibe el SE y verifica que no puede satisfacer la solicitud.<br>3. El Almacenero envía la solicitud a otra área que pueda satisfacer el pedido.<br>4. El Almacenero le despacha los productos y elabora un VED con copias.<br>5. El Almacenero entrega una copia al Responsable de sección.<br>6. El Almacenero actualiza las TE. |
| 7. El Responsable de | 8. El Almacenero envía otra copia del VED a |

| | |
|---|---|
| sección recibe los productos. | economía.<br>9. El Personal económico emite un CC. |
| **Poscondiciones** | Se realiza una solicitud de productos |
| **Ver Diagrama de actividades en Anexo 15.** | |

### *2.2.8 Modelo de objetos del negocio.*

Este modelo muestra los trabajadores del negocio con las entidades con las que se relaciona.



### *2.2.9 Reglas del negocio.*

Las reglas del negocio son “condiciones que deben ser satisfechas” (Rational, 2003). Los procesos estudiados presentan condiciones que son necesarias para su buen funcionamiento. Estas son:

1. Los productos deben ser movidos con los documentos oficiales establecidos por la Resolución No. 11-2007 del Ministerio de Finanzas y Precios.
2. Los documentos no deben tener tachaduras o errores.
3. No se puede consumir productos que hayan sido recibidos, sin haber presentado los documentos oficiales en las secciones debidas.
4. Se debe tener en cuenta las unidades de medida, destacando que pomos, paquetes, bolsas, fardos, latas, rollos, no son unidades de medidas sino unidades de envases.
5. Los almacenes deben informar las existencias de los productos después de cada entrada o salida.
6. En los almacenes se debe velar por el almacenaje y conservación de los productos.
7. Cuando se le de entrada a productos se debe chequear la misma.
8. El IR debe reflejar los productos recepcionados.
9. Se debe identificar cada producto del almacén con una TE.
10. Las TE tienen que ser firmadas por el que recibe los productos.
11. Las TE deben reflejar la ubicación física del producto.
12. Todos los productos manejados por entrada o salida deben actualizar las TE.
13. Los datos reflejados en las TE deberán coincidir con las existencias reales de los productos.
14. El tiempo disponible para realizar una reclamación será de 90 días.
15. El tiempo disponible para realizar una devolución es de 60 días después de haber hecho una reclamación.
16. Solo se podrán reclamar productos que fueron entrados al almacén.
17. Todos los documentos que avalan las operaciones deben tener la firma de quienes lo autorizan y demás datos para que tengan validez.
18. El VED debe tener la referencia del VED emitido al entregar los productos que se desean devolver.
19. La cantidad de productos a devolver no debe ser mayor que la cantidad de productos reflejada en el VED de salida.
20. Las FT deben tener un identificador para cada producto.

21. No se pueden crear FT a productos que pertenezcan a los Grupos Insumos o Útiles.
22. El LRC se registra sin tener en cuenta la F, solo contando con lo que se recibe en el área de almacenaje.
23. La F se revisa en el departamento económico.
24. Los productos se reciben en el área determinada para ello.
25. Chequear los productos recibidos contra documentos recibidos. No se puede almacenar productos que no estén contemplados en los documentos.
26. Se elabora un IR si hay faltante según dice la F.
27. Se debe tener en el área de almacén un listado actualizado de las personas autorizadas para solicitar productos.
28. Se debe despachar los productos según indique el VED de salida.
29. Toda información de movimientos de productos debe informarse a economía, para que realice los ajustes contables que conlleven.
30. La devolución podrá realizarse por producto sobrante, o por inconformidad con lo recibido.
31. Para darle salida a un producto debe haber existencia de este y estar activo.
32. El personal del almacén no puede tener acceso a la contabilidad.
33. No se puede elaborar un IR sin haber realizado el conteo físico de los productos.
34. El LRC no se puede modificar a menos que se haga el mismo día de la recepción ciega.

## 2.3 Especificación de requisitos.

Durante esta etapa se registran los requisitos capturados hasta el momento, y se especifican como las capacidades operacionales que deberá tener el sistema lo más detalladamente posible.

A continuación se especifican tanto los requisitos funcionales como los no funcionales.

### *2.3.1 Requisitos funcionales.*

Se tomó como alternativa para reflejar los requisitos funcionales, una tabla que contiene el requisito con su nomenclador de la forma RF. #, La descripción del requisito con todos los datos implicados, el caso de uso que le precede y con el que se le da seguimiento al requisito.

| | |
|---|---|
| **RF.1** | Insertar nueva FT de despiece |
| **Descripción** | El sistema debe crear una FT de despiece con los datos siguientes |

| | |
|---|---|
| | obligatorios:<br>• Área de donde proviene<br>• Producto a despiezar<br>• Unidad de medida<br>• Productos resultantes<br>- Grupo<br>- Familia<br>- Producto<br>- Unidad de medida<br>- Puntos<br>1.1. El sistema debe verificar que el producto a despiezar no es del Grupo Insumo ni Útiles. |
| **Precedencia** | CUN Realizar despiece |
| **Seguimiento** | CU Gestionar Ficha Técnica del despiece |

| | |
|---|---|
| **RF.2** | Eliminar FT de despiece |
| **Descripción** | El sistema debe eliminar una FT de despiece seleccionada.<br>2.1. El sistema debe buscar una FT de despiece |
| **Precedencia** | CUN Realizar despiece |
| **Seguimiento** | CU Gestionar Ficha Técnica del despiece |

| | |
|---|---|
| **RF.3** | Mostrar FT de despiece |
| **Descripción** | El sistema debe permitir buscar una FT de despiece y mostrarla. |
| **Precedencia** | CUN Realizar despiece |
| **Seguimiento** | CU Gestionar Ficha Técnica del despiece |

| | |
|---|---|
| **RF.4** | Insertar datos de un nuevo RE |
| **Descripción** | El sistema debe crear un RE introduciendo los siguientes datos:<br>• Área.<br>• Fecha.<br>• Producto despiezado.<br>• Unidad de medida. |

| | |
|---|---|
| | • Cantidad despiezada.<br>- Grupo<br>- Familia<br>- Producto<br>- Unidad de medida<br>- Cantidad.<br>- Puntos. |
| **Precedencia** | CUN Realizar despiece |
| **Seguimiento** | CU Gestionar despiece |

| | |
|---|---|
| **RF.5** | Eliminar datos de un RE |
| **Descripción** | El sistema debe permitir eliminar los datos de un RE.<br>5.1. El sistema debe buscar un RE |
| **Precedencia** | CUN Realizar despiece |
| **Seguimiento** | CU Gestionar despiece |

| | |
|---|---|
| **RF.6** | Mostrar datos de un RE |
| **Descripción** | El sistema debe permitir buscar y mostrar los datos de un RE |
| **Precedencia** | CUN Realizar despiece |
| **Seguimiento** | CU Gestionar despiece |

| | |
|---|---|
| **RF.7** | Rebajar existencia de productos |
| **Descripción** | El sistema permite eliminar un producto de la base de datos.<br>7.1. El sistema debe buscar los productos a eliminar |
| **Precedencia** | CUN Solicitar productos. |
| **Seguimiento** | CU Gestionar existencias de productos |

| | |
|---|---|
| **RF.8** | Dar entrada a nuevos productos |
| **Descripción** | El sistema debe registrar los datos siguientes:<br>✓ Nombre y código de la entidad receptora.<br>✓ Nombre y código del proveedor. |

| | |
|---|---|
| | ✓ Del Transportador:<br>• Nombre.<br>• Carnet de identidad.<br>• Chapa.<br>✓ Del producto:<br>• Código, descripción, unidad de medida, cantidad, precio unitario total, importe y saldo en existencia según almacén de cada producto.<br>✓ Importe Total.<br>✓ Número y nombre de Factura.<br>✓ Recepciona.<br>✓ Jefe del Almacén.<br>✓ Contabiliza.<br>✓ Fecha de emisión del modelo.<br>8.1. El sistema debe generar un número consecutivo del modelo al guardar. |
| **Precedencia** | CUN Recepcionar productos |
| **Seguimiento** | CU Gestionar existencias de productos |

| | |
|---|---|
| **RF.9** | Mostrar los datos de un Informe de Recepción |
| **Descripción** | El sistema permite buscar y mostrar los datos de un IR. |
| **Precedencia** | CUN Realizar reclamación y CUN Devolver productos |
| **Seguimiento** | CU Gestionar existencias de productos |

| | |
|---|---|
| **RF.10** | Insertar los datos de un Informe de Reclamación |
| **Descripción** | El sistema registra el informe con los siguientes datos:<br>✓ Del comprador:<br>• Nombre, código y dirección.<br>✓ Del proveedor:<br>• Nombre, código, dirección, cuenta bancaria y sucursal del Banco.<br>✓ Del transportador: |

| | |
|---|---|
| | • Nombre, dirección, carné de identidad y chapa del vehículo utilizado.<br>✓ De los productos:<br>• Código, descripción, unidad de medida, cantidad, precio unitario total e importe de cada producto a reclamar.<br>✓ Importe Total.<br>✓ Número de la factura(o Informe de Recepción).<br>✓ Receptor.<br>✓ Transportador.<br>✓ Proveedor.<br>✓ Contabiliza.<br>✓ Fecha de emisión del modelo.<br>10.1. El sistema debe verificar que la diferencia de días entre la fecha actual y la fecha de recepción es menor que 90. |
| **Precedencia** | CUN Realizar reclamación |
| **Seguimiento** | CU Gestionar reclamación. |

| | |
|---|---|
| **RF.11** | Mostrar los datos de un Informe de Reclamación |
| **Descripción** | El sistema permite buscar y mostrar un IRecl. |
| **Precedencia** | CUN Realizar reclamación |
| **Seguimiento** | CU Gestionar reclamación. |

| | |
|---|---|
| **RF.12** | Eliminar los datos de un Informe de Reclamación |
| **Descripción** | El sistema permite eliminar los datos de un IRecl.<br>12.1. El sistema debe buscar el IRecl. |
| **Precedencia** | CUN Realizar reclamación |
| **Seguimiento** | CU Gestionar reclamación. |

| | |
|---|---|
| **RF.13** | Insertar los datos de un nuevo Vale de Entrega o Devolución |
| **Descripción** | El sistema debe registrar los datos del VED entrando los siguientes datos:<br>✓ Nombre y código de la entidad. |

| | |
|---|---|
| | ✓ Nombre y código del área.<br>✓ Centro de costo, código y Orden de Trabajo.<br>✓ De los productos:<br>• Código, descripción, unidad de medida, cantidad, precio unitario total, importe y saldo en existencia de cada producto.<br>✓ Importe Total.<br>✓ Entrega o recibe.<br>✓ Recibe o entrega.<br>✓ Contabiliza.<br>✓ Fecha de emisión del modelo.<br>13.1. El sistema debe verificar que la diferencia de días entre la fecha actual y la fecha de recepción es menor que 90.<br>13.2. El sistema debe verificar que tiene referencia del VED de entrega del producto a devolver. |
| **Precedencia** | CUN Devolver productos |
| **Seguimiento** | CU Gestionar entregas o devoluciones |

| | |
|---|---|
| **RF.14** | Mostrar los datos de un VED |
| **Descripción** | El sistema permite buscar y mostrar los datos de un VED seleccionado |
| **Precedencia** | CUN Devolver productos |
| **Seguimiento** | CU Gestionar entregas o devoluciones |

| | |
|---|---|
| **RF.15** | Eliminar los datos de un VED |
| **Descripción** | El sistema debe permitir eliminar los datos de un VED.<br>15.1. El sistema debe buscar el VED. |
| **Precedencia** | CUN Devolver productos |
| **Seguimiento** | CU Gestionar entregas o devoluciones |

| | |
|---|---|
| **RF.16** | Insertar datos del Listado para Recepción Ciega |
| **Descripción** | El sistema debe permitir crear el LRC entrando los siguientes datos: |

| | |
|---|---|
| | ✓ Área de destino.<br>✓ Código y descripción del proveedor.<br>✓ Del producto:<br>• Producto, código, descripción, unidad de medida, precio de costo, precio de venta, fecha de vencimiento, cantidad recepcionada.<br>✓ Recibe.<br>✓ Entrega.<br>✓ Fecha.<br>16.1. El sistema debe generar automáticamente el número consecutivo del LRC al guardar. |
| **Precedencia** | CUN Recepcionar productos |
| **Seguimiento** | CU Gestionar recepción ciega |

| | |
|---|---|
| **RF.17** | Mostrar los datos del Listado para Recepción Ciega seleccionado |
| **Descripción** | El sistema debe permitir buscar el LRC y mostrarlo |
| **Precedencia** | CUN Recepcionar productos |
| **Seguimiento** | CU Gestionar recepción ciega |

| | |
|---|---|
| **RF.18** | Eliminar datos del Listado para Recepción Ciega seleccionado |
| **Descripción** | El sistema debe permitir eliminar el LRC.<br>18.1. El sistema debe buscar el LRC.<br>18.2. El sistema debe eliminar solo si la fecha de la recepción ciega coincide con la del momento en que se desea eliminar. |
| **Precedencia** | CUN Recepcionar productos |
| **Seguimiento** | CU Gestionar recepción ciega |

| | |
|---|---|
| **RF.19** | Insertar datos de una nueva Solicitud de Entrega |
| **Descripción** | El sistema debe permitir crear una SE entrando los siguientes datos:<br>✓ Nombre y código de la entidad.<br>✓ Nombre y código del área.<br>✓ Centro de costo y su código. |

| | |
|---|---|
| | ✓ Número de la Orden de Trabajo.<br>✓ De los productos:<br>• Código, descripción, unidad de medida y cantidad.<br>✓ Solicita.<br>✓ Fecha de la Solicitud.<br>✓ Autoriza y fecha de autorizo.<br>✓ Recibe y fecha de recibo.<br>19.1. El sistema debe generar un número consecutivo del modelo al guardar. |
| **Precedencia** | CUN Solicitar productos |
| **Seguimiento** | CU Gestionar solicitud de productos |

| | |
|---|---|
| **RF.20** | Mostrar datos de una SE |
| **Descripción** | El sistema debe permitir buscar y mostrar una SE |
| **Precedencia** | CUN Solicitar productos |
| **Seguimiento** | CU Gestionar solicitud de productos |

| | |
|---|---|
| **RF.21** | Eliminar SE |
| **Descripción** | El sistema debe permitir eliminar una SE.<br>22.1. El sistema debe buscar una SE. |
| **Precedencia** | CUN Solicitar productos |
| **Seguimiento** | CU Gestionar solicitud de productos |

| | |
|---|---|
| **RF.22** | Buscar productos |
| **Descripción** | El sistema debe permitir buscar existencia de un producto. |
| **Precedencia** | CUN Solicitar productos |
| **Seguimiento** | CU Gestionar existencias de productos |

| | |
|---|---|
| **RF.23** | Permitir movimiento de productos entre secciones |
| **Descripción** | El sistema debe permitir el movimiento de productos entre secciones, mostrando los siguientes datos: |

| | |
|---|---|
| | Invariables:<br>✓ Nombre y código de la entidad.<br>✓ Nombre y código del área.<br>✓ Centro de costo y su código.<br>✓ Número de la Orden de Trabajo.<br>✓ De los productos:<br>• Código, descripción, unidad de medida y cantidad.<br>✓ Solicita.<br>✓ Fecha de la Solicitud.<br>✓ Autoriza y fecha de autorizo.<br>✓ Recibe y fecha de recibo. |
| **Precedencia** | CUN Mover productos entre secciones. |
| **Seguimiento** | CU Transferir productos entre secciones |

Se determinó crear un Módulo Administración, pues para realizar cualquier operación con el sistema, este debe tener creados usuarios y políticas de acceso. A continuación se describen los requisitos:

| | |
|---|---|
| **RF.24** | Gestionar usuarios |
| **Descripción** | El sistema debe permitir administrar los accesos al sistema.<br>24.1. Crear un nuevo usuario.<br>24.2. Modificar información de un usuario.<br>24.3. Eliminar un usuario. |
| **Seguimiento** | CU Gestionar Usuarios |

| | |
|---|---|
| **RF.25** | Permitir autenticar un usuario. |
| **Descripción** | El sistema debe permitir que los usuarios se autentiquen. |
| **Seguimiento** | CU Autenticar |

| | |
|---|---|
| **RF.26** | Gestionar roles de usuarios |
| **Descripción** | 26.1. Crear roles de usuarios.<br>26.2. Modificar un rol de usuario.<br>26.3. Eliminar roles de usuarios. |
| **Seguimiento** | CU Gestionar roles de usuarios |

| | |
|---|---|
| **RF.27** | Gestionar estaciones de trabajo |
| **Descripción** | 27.1. Registrar estaciones de trabajo.<br>27.2. Eliminar estaciones de trabajo. |
| **Seguimiento** | CU Gestionar estaciones de trabajo |

### *2.3.2 Requisitos no funcionales.*

Se deben tener en cuenta los requisitos no funcionales planteados por la arquitectura del sistema. Para su buen funcionamiento el sistema debe contar con un servidor de base de datos y un servidor de aplicaciones. A continuación se proponen los requisitos específicos para cada uno de ellos.

Servidor de Software

- Tener periféricos Mouse y Teclado.
- 1Mb de cache L2, 1 GB de memoria RAM.
- 2 GB de espacio libre en disco.
- Tarjeta de red.
- Uninterruptible Power Supply (UPS).
- Sistema Operativo: Windows 2000 o superior, Linux, Unix.
- Máquina Virtual de Java: Java Development Kit (JDK) versión 1.6

Servidor de Base de Datos

- Tener periféricos Mouse y Teclado.
- 1Mb de cache L2, 1 GB de memoria RAM.
- 2 GB de espacio libre en disco.
- Tarjeta de red.
- Uninterruptible Power Supply (UPS).
- Sistema Operativo: Windows 2000 o superior, Linux, Unix.
- Gestor de Base de Datos: PostgreSQL 8.2

Estaciones de Trabajo

- Tener periféricos Mouse y Teclado.
- 256 Mb de memoria RAM.
- Tarjeta de red.

- Uninterruptible Power Supply (UPS).
- Sistema Operativo: Windows 2000 o superior, Linux, Unix.
- Máquina Virtual de Java: Java Development Kit (JDK) versión 1.6
- Las estaciones que estén designadas para generar reportes deben tener una impresora.

Redes:

- La red existente en las instalaciones debe de soportar la transacción de paquetes de información de al menos unas 10 máquinas a la vez.

- Para hacer más fiable la aplicación debe de estar protegida contra fallos de corriente y de conectividad, para lo que se deberá parametrizar los tiempos para realizar copias de seguridad e implementar las transacciones de paquetes en la red con el protocolo TCP/ IP de manera que permita la recuperación de los datos.

Seguridad:

- Se definirán niveles de acceso donde los usuarios tendrán asignados los permisos que le correspondan.
- El servidor de Bases de Datos funcionara las 24 horas.
- Se usarán en la Base de Datos Triggers para mayor seguridad a la hora de tratar con la información.

Soporte:

- El sistema contará con un manual de usuario, para facilitar su uso.
- Se contará con personal preparado para dar soporte por 3 meses en la entidad donde se instale el sistema.

## 2.4 Modelado del sistema.

En esta etapa se desarrolla el modelo de casos de uso basado en los requisitos previamente capturados. Este artefacto posibilita llegar a un acuerdo entre desarrolladores y clientes. Además, es la entrada fundamental para el análisis, diseño y pruebas.

### *2.4.1 Actores del sistema.*

En el modelado del sistema los actores suelen ser los trabajadores del negocio. Representan a los que interactúan con el sistema.

| Actor | Descripción |
|---|---|
| Normador | Es quien gestiona los RE y las FT. |
| Responsable de sección | Elabora el Informe de Reclamación. |
| Almacenero | Es quien elabora el Informe de Recepción, el VED. Actualiza las existencias de los productos del almacén. |
| Administrador | Persona encargada de administrar el acceso al sistema. |
| Usuario | Rol general que representa a todos los anteriores y que se autentica. |

### *2.4.2 Patrones de casos de uso.*

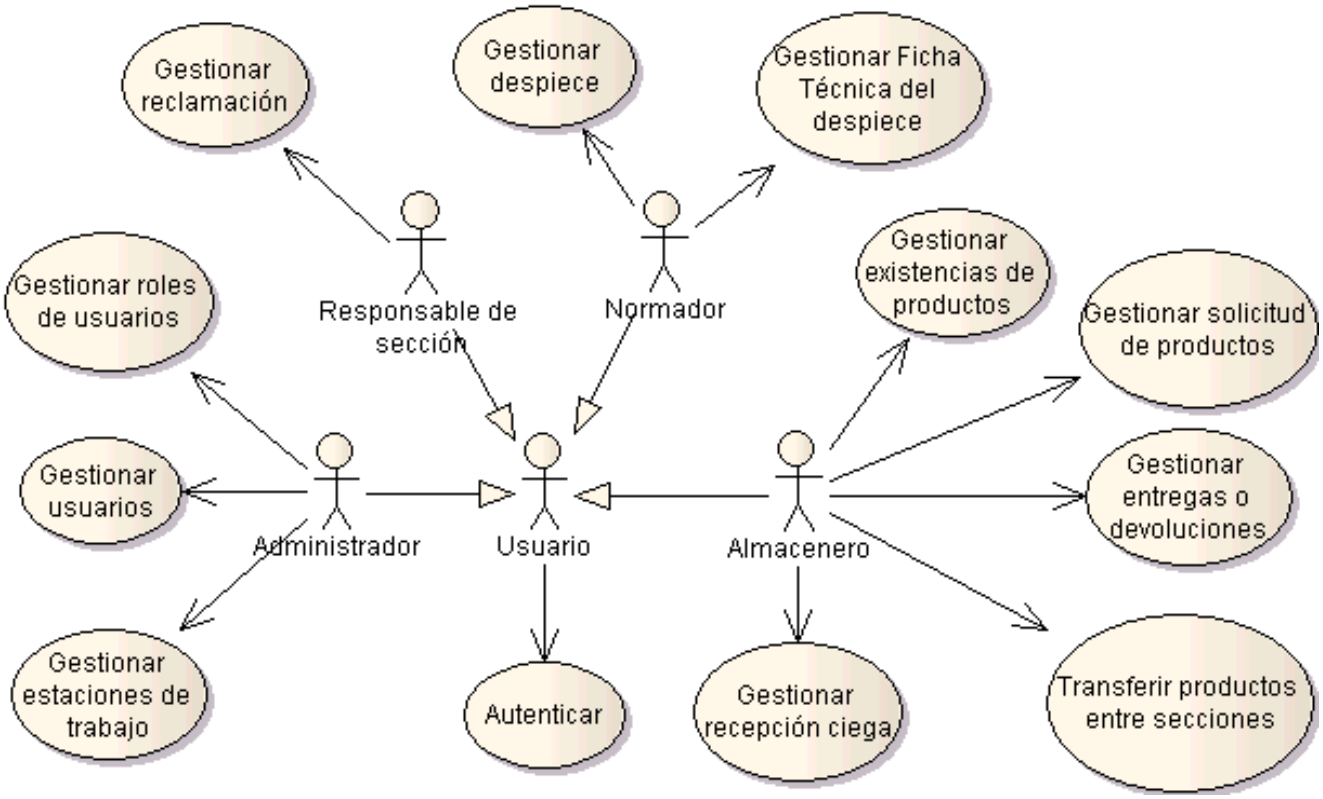
Tras el proceso de análisis de los requisitos y la definición de los casos de uso que se tendrán en el sistema, que responden y abarcan esos requisitos, se representan los actores y casos de uso mediante el Diagrama de Casos de Uso (se muestra en el siguiente epígrafe). En este diagrama se utilizan varios patrones de casos de uso. Estos son:

CRUD Completo: Ya que se presentan en casos de uso de gestionar información, los procesos crear, modificar, eliminar.

Múltiples actores, específicamente el patrón Rol común: Pues se destacan actividades comunes para varios actores, por lo que se define un actor del cual heredan los actores con estas funcionalidades comunes.

### *2.4.3 Diagrama de casos de uso del sistema.*

Este modelo contiene actores, casos de uso y sus relaciones. Describe lo que hace el sistema para cada actor.




### *2.4.4 Descripción de los casos de uso del sistema.*

| **Caso de Uso:** | **Gestionar Ficha Técnica de despiece** |
|---|---|
| **Actores:** | Normador |
| **Resumen:** | Este caso de uso permite al actor crear, visualizar o eliminar FT de despiece. |
| **Precondiciones:** | El usuario debe tener los permisos para realizar la operación. Deben estar definidos los datos del producto a despiezar y el área de la que proviene. |
| **Referencias:** | RF.1, RF.2, RF.3 |
| **Prioridad:** | Secundaria |
| **Flujo Normal de Eventos: Ver Anexo 16** | |

| **Caso de Uso:** | **Gestionar despiece** |
|---|---|
| **Actores:** | Normador |
| **Resumen:** | Este caso de uso permite registrar los productos obtenidos tras el despiece y generar el RE, así como modificar la información o |

| | |
|---|---|
| | eliminarla. |
| **Precondiciones:** | El usuario debe tener los permisos para realizar la operación.<br>Debe estar definida la FT del producto y el área.<br>Debe haber una solicitud previa del producto que se elabora. |
| **Referencias:** | RF.4, RF.5, RF.6 |
| **Prioridad:** | Secundaria |
| **Flujo Normal de Eventos: Ver Anexo 17** | |

| **Caso de Uso:** | **Gestionar reclamación.** |
|---|---|
| **Actores:** | Responsable de sección |
| **Resumen:** | El caso de uso permite registrar, modificar y eliminar una reclamación. |
| **Precondiciones:** | Los productos y proveedores deben estar predefinidos.<br>El IR debe estar registrado.<br>La fecha de la reclamación no debe pasarse de los 90 días después de la recepción.<br>El usuario debe tener los permisos necesarios para efectuar la operación. |
| **Referencias:** | RF.10, RF.11, RF.12 |
| **Prioridad:** | Secundaria |
| **Flujo Normal de Eventos: Ver Anexo 18** | |

| **Caso de Uso:** | **Gestionar entregas o devoluciones** |
|---|---|
| **Actores:** | Almacenero |
| **Resumen:** | Este caso de uso permite registrar en el sistema las entregas y devoluciones de productos, así como modificarlas o eliminarlas. |
| **Precondiciones:** | El usuario debe tener los permisos necesarios para efectuar la operación.<br>El producto debe existir y estar activo. |
| **Referencias:** | RF.13, RF.14, RF.15 |
| **Prioridad:** | Secundaria |
| **Flujo Normal de Eventos: Ver Anexo 19** | |

| **Caso de Uso:** | **Gestionar solicitud de productos** |
|---|---|
| **Actores:** | Almacenero |
| **Resumen:** | El caso de uso permite registrar, modificar y eliminar las solicitudes de entregas de materiales. |
| **Precondiciones:** | El usuario debe tener los permisos necesarios para efectuar la operación. |
| **Referencias:** | RF.19, RF.20, RF.21, RF.22 |
| **Prioridad:** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 20** | |

| **Caso de Uso:** | **Transferir productos entre áreas** |
|---|---|
| **Actores:** | Almacenero |
| **Resumen:** | El caso de uso permite |
| **Precondiciones:** | El usuario debe tener los permisos necesarios para efectuar la operación.<br>El producto debe existir y estar activo. |
| **Referencias:** | RF.23 |
| **Prioridad:** | Secundaria |
| **Flujo Normal de Eventos: Ver Anexo 21** | |

| **Caso de Uso:** | **Gestionar recepción ciega** |
|---|---|
| **Actores:** | Almacenero |
| **Resumen:** | El caso de uso permite insertar, modificar y eliminar un LRC. |
| **Precondiciones:** | El usuario debe tener los permisos necesarios para efectuar la operación. |
| **Referencias:** | RF.16, RF.17, RF.18 |
| **Prioridad:** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 22** | |

| **Caso de Uso:** | **Gestionar existencias de productos** |
|---|---|
| **Actores:** | Almacenero |

| | |
|---|---|
| **Resumen:** | El caso de uso permite darle entrada o salida a algún producto, así como buscarlo. |
| **Precondiciones:** | El usuario debe tener los permisos necesarios para efectuar la operación.<br>Los productos, las monedas, las áreas y las unidades de medida deben estar nomenclados.<br>Los proveedores deben estar registrados. |
| **Referencias:** | RF.7, RF.8, RF.9 |
| **Prioridad:** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 23** | |

| | |
|---|---|
| **Caso de Uso:** | **Autenticar** |
| **Actores:** | Usuario |
| **Resumen:** | Este caso de uso permite identificar el usuario que trata de entrar al sistema y darle entrada en dependencia del rol que tiene. |
| **Precondiciones:** | El sistema debe estar instalado y ejecutado correctamente. |
| **Referencias** | RF.25 |
| **Prioridad** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 24** | |

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Usuarios** |
| **Actores:** | Administrador del Sistema |
| **Resumen:** | Este caso de uso comprende la funcionalidad de crear usuarios, cambiar su rol de usuario, cambiar la contraseña. |
| **Precondiciones:** | El administrador es el único usuario con permisos para iniciar este caso de uso. |
| **Referencias** | RF.24 |
| **Prioridad** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 25** | |

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Roles de Usuarios** |
| **Actores:** | Administrador |

| | |
|---|---|
| **Resumen:** | Este caso de uso tiene la funcionalidad de crear roles de usuarios con las opciones de accesibilidad para cada uno |
| **Precondiciones:** | El sistema debe ser instalado y ejecutado correctamente.<br>Se debe cargar del sistema las opciones a las que se podrá tener acceso en el sistema.<br>El administrador es el único que puede iniciar el caso de uso. |
| **Referencias** | RF.26 |
| **Prioridad** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 26** | |

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Estaciones de Trabajo** |
| **Actores:** | Administrador |
| **Resumen:** | Este caso de uso comprende la funcionalidad de crear estaciones de trabajo desde la que se ejecutara la aplicación. |
| **Precondiciones:** | El sistema debe ser instalado y ejecutado correctamente.<br>Solo el administrador puede iniciar el caso de uso. |
| **Referencias** | RF.27 |
| **Prioridad** | Primaria |
| **Flujo Normal de Eventos: Ver Anexo 27** | |

Es importante señalar que la gestión de los requisitos se tuvo en cuenta durante toda la etapa de levantamiento de requisitos, pues cuando se presentaron cambios, se realizaron tormentas de ideas para administrarlos, manejarlos y adecuarlos lo mejor posible, tratando de afectar lo menos posible el desarrollo del trabajo.

### 2.5 Conclusiones Parciales.

Durante la realización del capítulo se logró:

- Modelar los procesos del negocio, permitiendo tener una mejor comprensión de los procesos de la gestión de inventarios.
- Capturar los requisitos funcionales y no funcionales del sistema, aplicando técnicas como la arqueología de documentos, el desarrollo conjunto de aplicaciones y las tormentas de ideas.

- Aplicar patrones de casos de uso de forma que se pudiera modelar más claro el sistema.
- Generar un artefacto muy importante para poder realizar el diseño: el modelo del sistema.

CAPÍTULO 3: MODELO DE DISEÑO Y VALIDACIÓN.

## 3.1 Introducción.

En este capítulo se realiza el modelado del diseño del sistema, donde se explican los subsistemas, las clases, los patrones utilizados, la arquitectura definida por los arquitectos del proyecto y la realización de los casos de uso desarrollados para los Módulos Inventario y Administración del ERP Cubano.
El diseño se ha realizado para ser implementado en la plataforma Java, utilizando la máquina virtual Java Development Kit (JDK) en su versión 6.0 y, por lo tanto, utilizando el lenguaje de programación Java. La fundamentación de esta selección se encuentra en la tesis (Perera, 2007). Se utiliza la herramienta Case el Enterprise Architect 7.0, que usa la especificación UML 2.1, para modelar.

Además, se realiza una evaluación de la calidad del trabajo realizado, aplicando métricas a los requisitos capturados, los casos de uso definidos y al diseño propuesto.

## 3.2 Flujo de análisis y diseño.

El modelo del análisis es un artefacto que usualmente se genera para entender mejor los requisitos y realizar mejor el diseño. Es típicamente un artefacto temporal, que brinda una aproximación al diseño y se realiza normalmente cuando no se tiene una idea clara de los procesos y requerimientos. (Jacobson, y otros, 2000) Es decir, se realiza el análisis para lograr un mejor acercamiento a lo que se desea construir, para sentar las bases del diseño.

Según el estudio realizado, no se presenta una gran complejidad modular en el sistema en desarrollo, y los procesos y requerimientos están bien definidos, por lo que se decide no realizar el modelo de análisis, que de igual modo, brinda elementos técnicos, pero no abarca el lenguaje de programación, la plataforma de desarrollo y demás elementos necesarios que sí se tienen en cuenta en el diseño, permitiendo gestionar mejor el tiempo y el avance del proceso de desarrollo.

### *3.2.1 Modelo de Diseño.*

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en como los requisitos y restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Describe la jerarquía de subsistemas de diseño que contienen clases del diseño, realizaciones de casos de uso-diseño e interfaces. Sirve de abstracción de la implementación del sistema y es usado como una entrada fundamental de las actividades de implementación. (Jacobson, y otros, 2000)

#### *3.2.1.1 Arquitectura definida para el Sistema.*

En el Análisis y Diseño, para lograr un correcto modelo de diseño y demás artefactos generados en ese flujo de trabajo, se relacionan fundamentalmente dos trabajadores: el diseñador del sistema y el arquitecto, ya que se generan tres de las cinco vistas arquitectónicas del sistema y, por tanto, no pueden verse separados el diseño y la arquitectura de un software.

La arquitectura de software es quien le da forma al software para que soporte todos los requisitos. Establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema.

Se necesita una arquitectura robusta, que guíe el proceso de desarrollo, y que defina de manera abstracta los componentes que llevan a cabo alguna tarea, sus interfaces y la comunicación ente ellos.

La arquitectura seleccionada fue la arquitectura de tres capas véase (Perera, 2007). Esta se ha vuelto muy común a la hora de construir software de gestión debido a las facilidades que brinda, pues permite la reutilización y la independencia entre las capas, se pueden realizar cambios en capas sin tener que modificar las otras, facilita la estandarización, la utilización de los recursos y la administración.

En este caso se crean tres capas:

- ✓ *Capa de Presentación:* Tiene que ver con las interfaces y la interacción con los usuarios. Es la encargada de contener los controles de usuarios que son usados para la realización de los distintos casos de uso y de mantener todas las ventanas y menús que permiten el intercambio de información con la aplicación.

- ✓ *Capa de Negocio:* Ubica los subsistemas y clases encargadas de realizar la mayoría de las funcionalidades que pide el cliente, así como los aspectos del software que automatizan o apoyan los procesos de negocio que llevan a cabo los usuarios, como tareas, reglas y restricciones.

- ✓ *Capa de Acceso a Datos:* Contiene las clases que interactúan con la base de datos y que permiten realizar las distintas operaciones sobre ella, como son: procedimientos almacenados, consultas.

La arquitectura seleccionada cuenta además con dos paquetes: Core y Commons.

En el paquete Core es donde se encuentran un conjunto de clases y componentes comunes para todas las capas de la aplicación, estructuradas por paquetes y que implementan métodos de

validación, conversiones entre distintos tipos de datos, trabajo con cadenas y distintos componentes generales de pruebas. Además, en este paquete se encuentran todas las configuraciones de los frameworks utilizados en sistema.

En el paquete Commons se encuentran las clases del Objeto de mapeo, los ficheros XML del mapeo y los ficheros para las consultas con la que interactúan las clases del sistema para la solución de cada caso de uso.

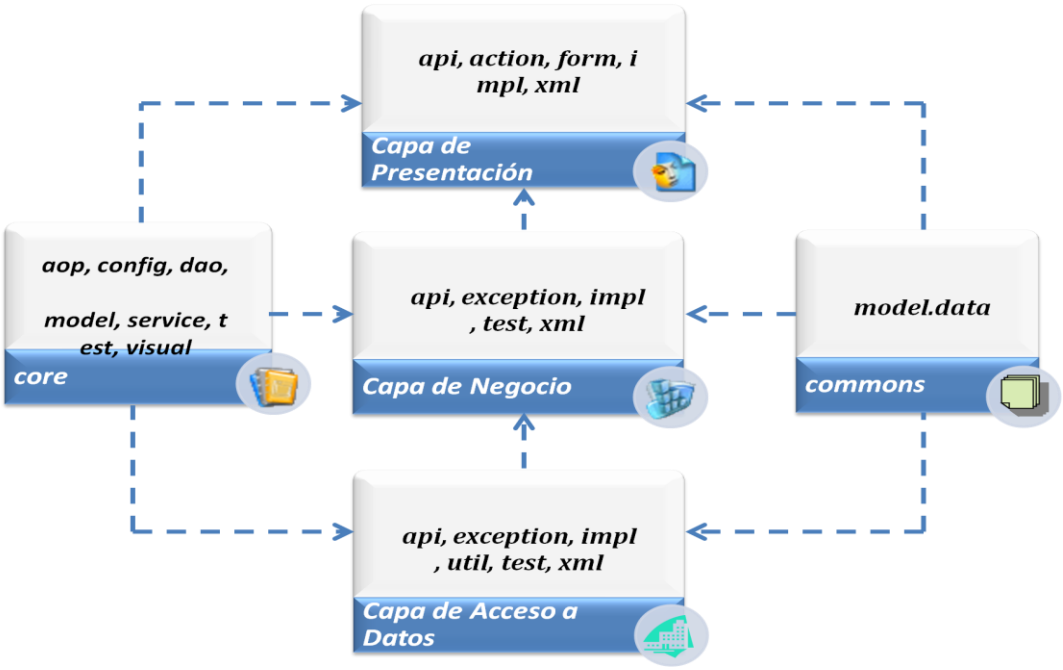



Figura 3.1: Estructura de la Arquitectura.

### *3.2.1.2 Descripción de los Módulos principales.*

En el diseño de sistema la modularidad es una de sus principales características.

El sistema ERP Cubano está compuesto por varios módulos que interactúan entre ellos para llegar al propósito final, brindar el mejor servicio al cliente. Los módulos desarrollados en este trabajo de investigación son: Administración e Inventario, producto a que cada uno posee tareas bien específicas y concretas.

*Módulo Administración:*

Su función es brindar la mayor seguridad posible al sistema. Es el encargado de administrar el acceso al sistema. Permite el acceso de acuerdo al rol asignado al usuario.

*Módulo Inventario:*

Su función es regular el flujo de mercancía. Es el que permite tener el control de toda la mercancía que posee la empresa. Evalúa los procedimientos de entrada y salida de sus bienes (controla su existencia) con el fin de hacer más rentable su posesión y garantizar en cierto grado el éxito de la organización.

Se puede apreciar que estos módulos se relacionan mutuamente, debido a que las funcionalidades de uno están incluidas dentro del otro y para que uno realice sus funciones correctamente, es necesaria la realización correcta de las tareas de otro módulo.

Los componentes o clases usadas para la realización de distintas funcionalidades de los módulos se implementaron en subsistemas independientes interrelacionados. Esta división permite la implementación paralela (al mismo tiempo) después de haber sido definida la arquitectura base y los componentes comunes para las distintas funcionalidades, para luego ser integrados haciendo de esta manera más rápido el trabajo.

#### *3.2.1.3 Justificación de los Patrones de Diseño utilizados.*

En el Modelo de Diseño se propone usar patrones pertenecientes a los GOF y los patrones de asignación de responsabilidades (GRASP), para que el diseño tenga un Bajo Acoplamiento, Alta Cohesión y que cada clase experta en algún tipo de información sea la encargada de realizar las operaciones con la misma, logrando con esto un mejor funcionamiento del sistema.

Además, se propone el uso de distintos frameworks que vienen destinados para facilitar el trabajo en los sistemas de gran envergadura como el presente. Para mas información véase (Perera, 2007):

*Swing*: Para la Capa de Presentación.

*Hibernate:* Para manejar la persistencia de los datos en la Capa de Acceso a Datos.

*Spring*: Para la Capa de Negocio. Es importante destacar que este framework promueve las buenas prácticas de diseño y programación ya que maneja patrones de diseño como Factory, Abstract Factory, Builder, Decorator y Service Locator pertenecientes a los patrones GOF y que son muy utilizados. Esto

permite a los desarrolladores abstraerse a la hora de aplicar un grupo de patrones que ya se garantizan con el uso de este framework.

#### *3.2.1.4 Subsistemas de Diseño.*

El Modelo de Diseño está compuesto por subsistemas de diseño que interactúan entre sí para realizar los casos de uso.

Subsistemas:

- ✓ Subsistema Commons (Comunes).
- ✓ Subsistema Core (Núcleo).
- ✓ Subsistema Inventario.
- ✓ Subsistema Administración.

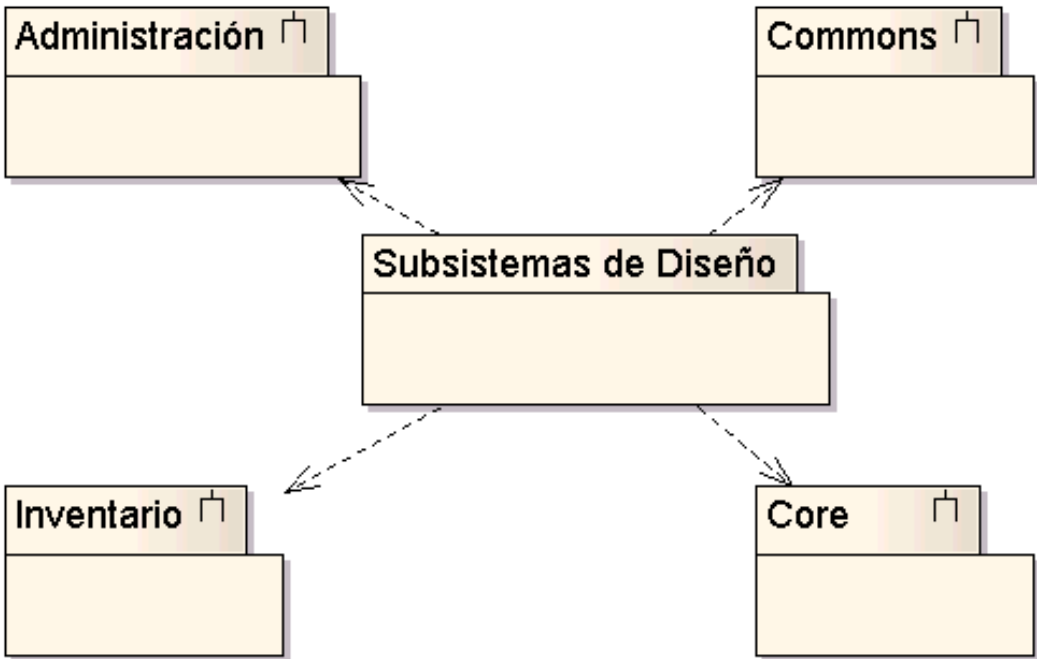



**Figura 3.2. Modelo de Diseño del Sistema.**

A continuación se explica la funcionalidad de los principales subsistemas, se muestra el diagrama de las clases que lo conforman y se describen las clases más importantes.

*Subsistema Commons:*

Es un subsistema compuesto por todas las clases que serán utilizadas por otros subsistemas. Es decir, es donde se encuentran las clases persistentes mapeadas de la base de datos, por la capa de Acceso a Datos, con las que interactúa cada una de las capas en la solución de cada CU.

*Subsistema Core:*
En este subsistema se encuentra el núcleo del sistema, es decir, es donde se encuentran las configuraciones de cada una de las capas de la arquitectura. Aquí se encuentran todos los archivos de configuración de las tres capas de la arquitectura y la configuración de los frameworks utilizados para el desarrollo del sistema.

*Subsistema Inventario:*
El subsistema Inventario es un subsistema muy importante, debido a que es donde se encuentran los casos de uso de más peso e importancia del sistema. En Inventario es donde se gestiona todo lo que tiene que ver con el flujo de información por parte del sistema. Aquí es donde se encuentran los casos que responden a los procesos de inventario analizados.

*Subsistema Administración:*
Este subsistema es el que se relaciona con la seguridad y la administración del sistema. Está compuesto por todos los casos de usos y las clases que permiten administrar el acceso al sistema.

#### *3.2.1.5 Diseño de clases.*

El diseño de clases asegura que las clases proporcionen el comportamiento que requieren las realizaciones de los casos de uso, así como la consistencia de la arquitectura de software, y la información suficiente para evitar ambigüedades en las clases implementadas.

##### *3.2.1.5.1 Realización de los Casos de Uso.*

La realización de los casos de uso está definida por varias clases. Estas, a su vez, están compuestas por varios atributos y métodos que permiten darle solución a las funcionalidades de cada caso de uso. Estas clases están distribuidas por paquetes dentro de las capas que componen la arquitectura.
La asignación de los nombres de las clases más importantes que interactúan en los casos de uso es la siguiente, donde para cada caso de uso el Nombre sería el nombre del caso de uso:

Nombre**Form**: Son las clases formularios o clases de interfaz de usuario. Estas clases son las que permiten el intercambio de información con el usuario.

Nombre**ActionImpl**: Son las clases que permiten capturar los datos insertados por el usuario en el formulario y realizar las acciones o eventos del formulario.

Nombre**ServiceImpl**: Son las clases donde se realiza todo lo relacionado con darle solución a las funcionalidades del caso de uso.

**BaseServiceImpl:** Es la clase donde se realizan todas las funcionalidades comunes que tienen los casos de uso.

**BaseDAOHibernate:** Es la clase donde se realiza todo lo necesario para el intercambio de información de la base de datos con la aplicación.

A continuación se presenta los diagramas de clases, que son las relaciones entre las clases y sus objetos (Jacobson, y otros, 2000), y los diagramas de secuencia, que son los que muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas (Jacobson, y otros, 2000), para los casos de usos mas importantes de cada uno de los módulos.

### *3.2.1.5.2 Modulo Administración.*

**CU Gestionar Usuarios:**

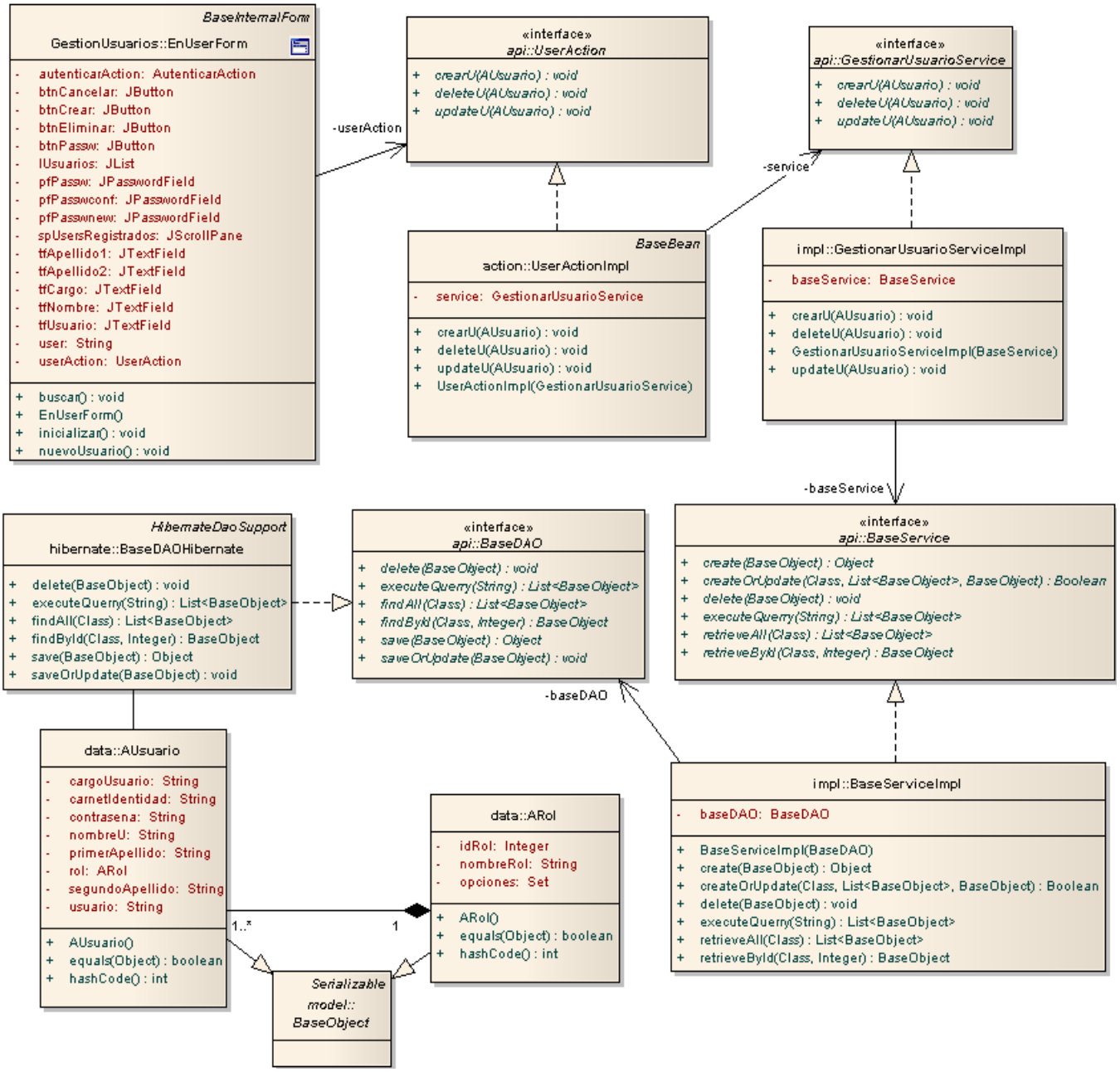



Figura 3.3: Diagrama de clases del caso de uso CU Gestionar Usuarios.

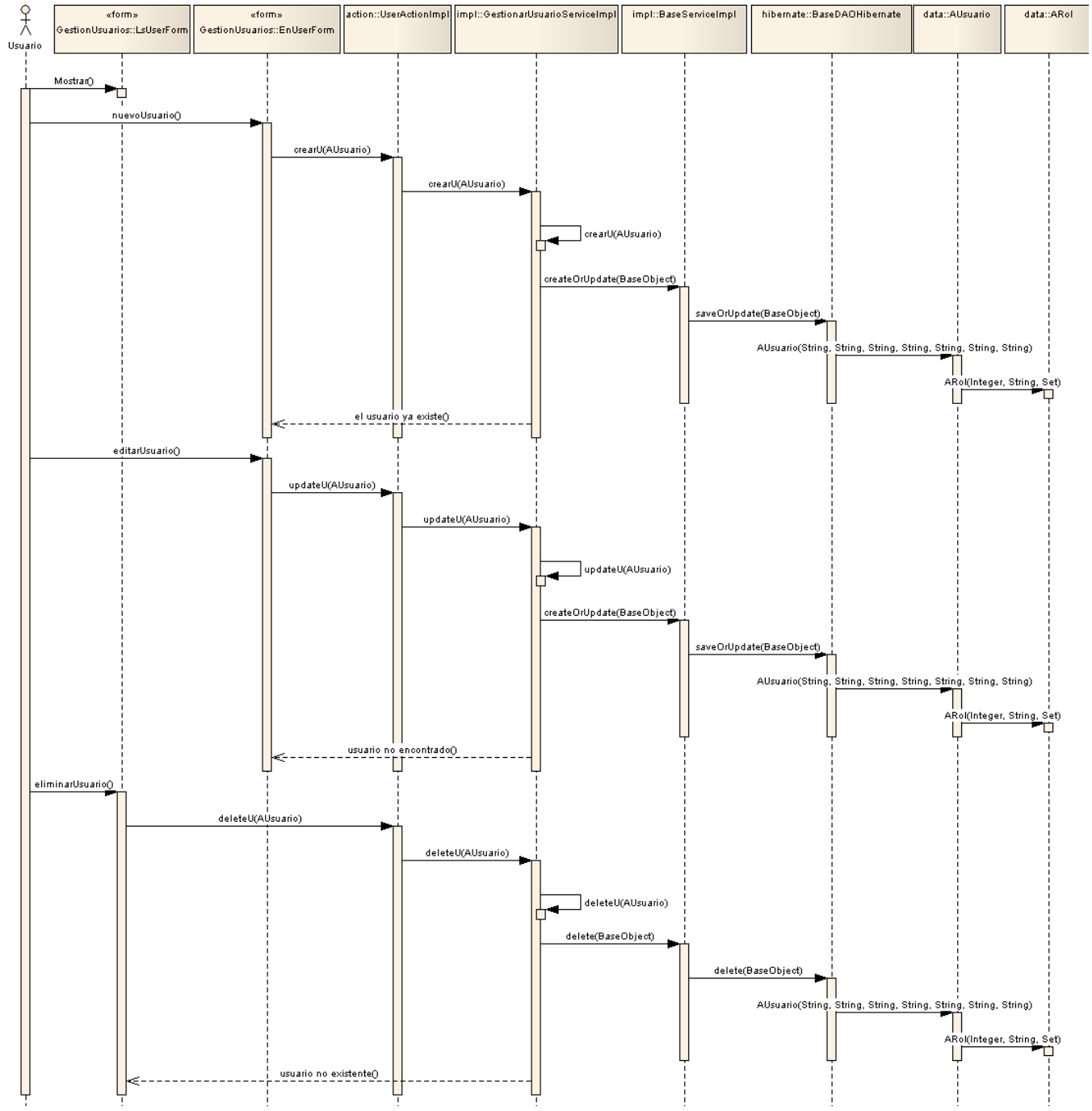



Figura 3.4: Diagrama de secuencia del caso de uso CU Gestionar Usuarios.

**CU Gestionar Roles:**

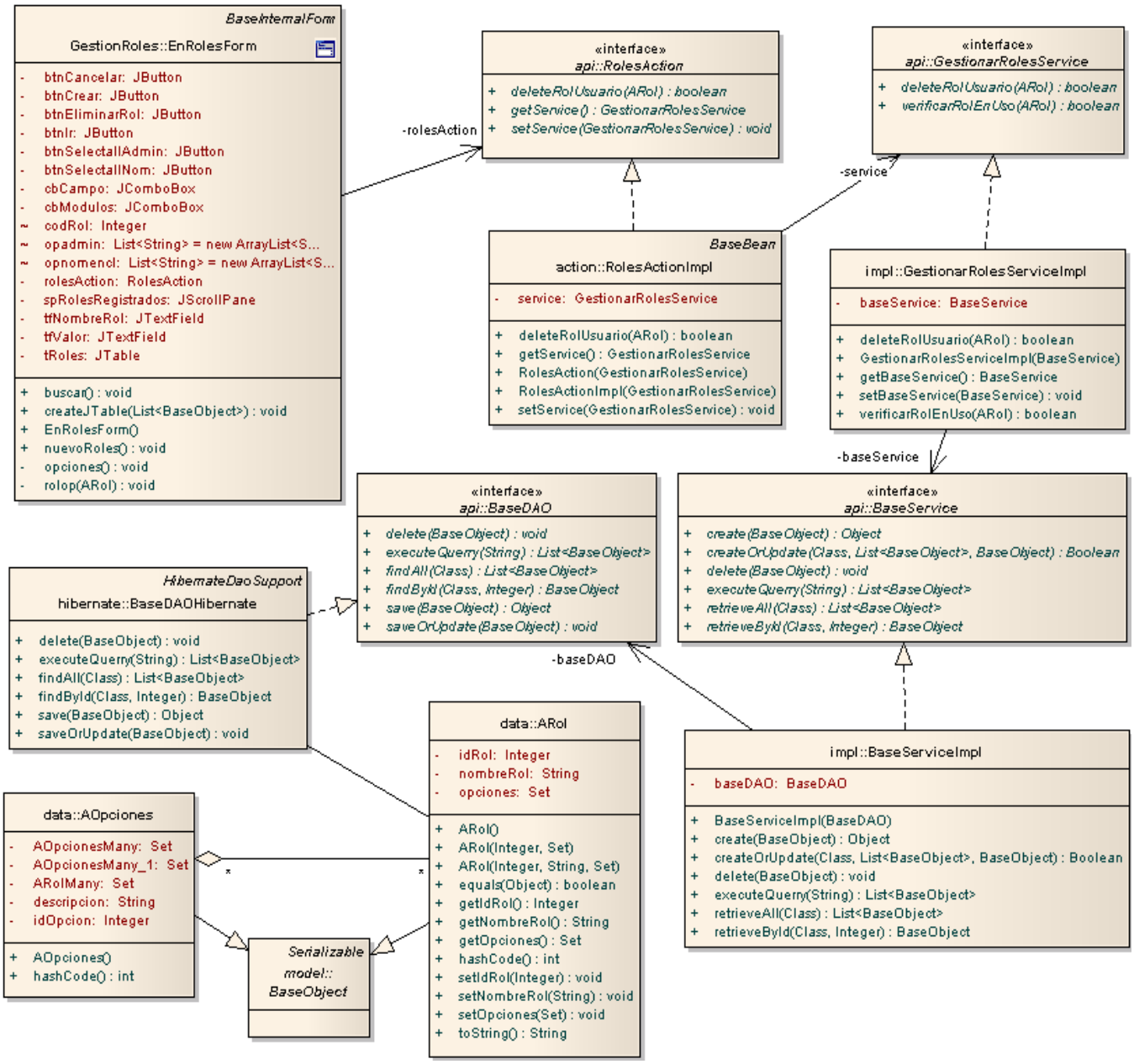



Figura 3.5: Diagrama de clases del caso de uso CU Gestionar Roles.

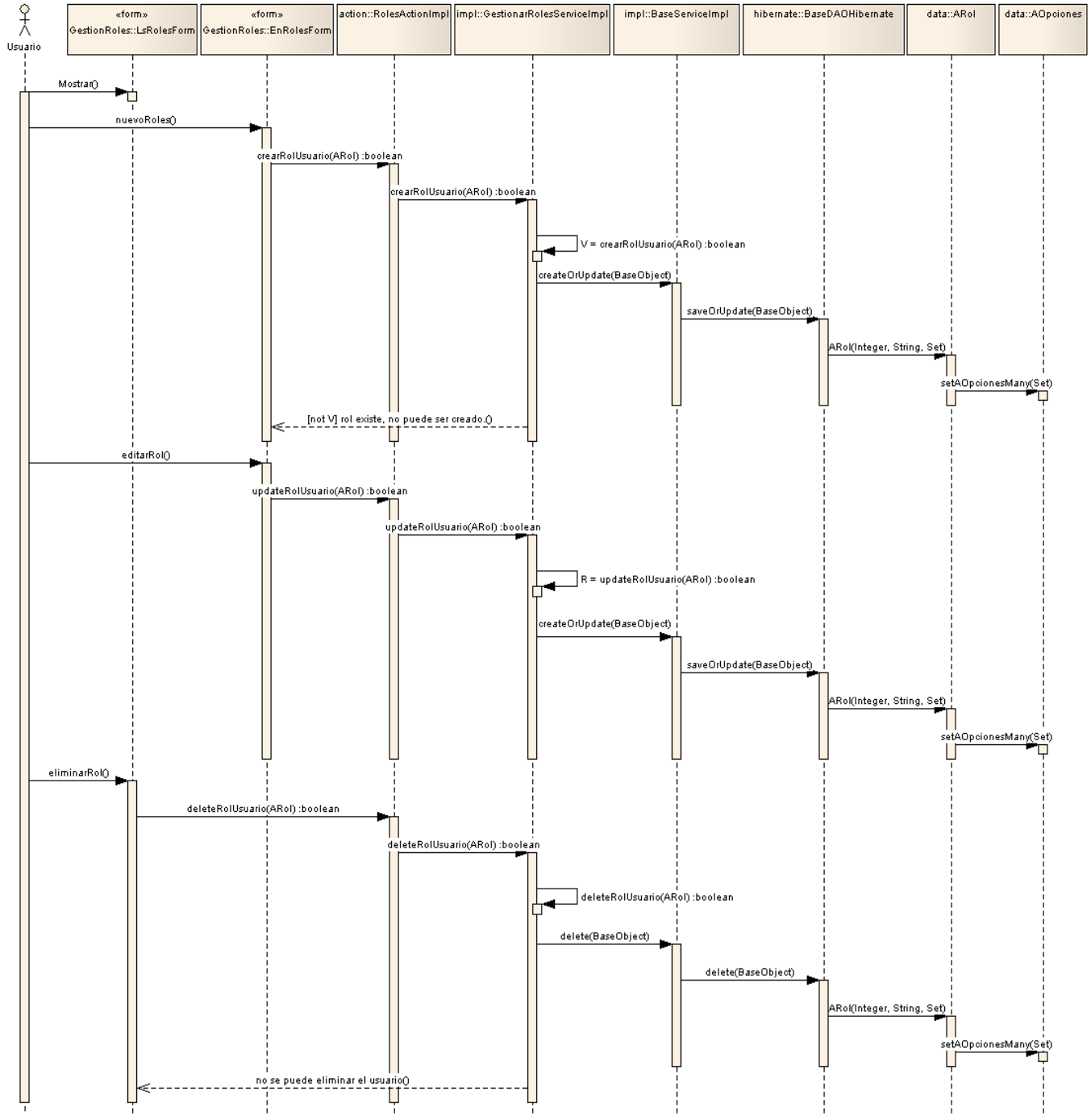



Figura 3.6: Diagrama de secuencia del caso de uso CU Gestionar Roles.

**CU Autenticar:**

Ver diagrama de clases en Anexo 28 y diagrama de secuencia en Anexo 39.

**CU Gestionar Terminal:**

Ver diagrama de clases en Anexo 29 y diagrama de secuencia en Anexo 40.

### *3.2.1.5.3 Modulo Inventario:*

**CU Gestionar Solicitud de Productos:**

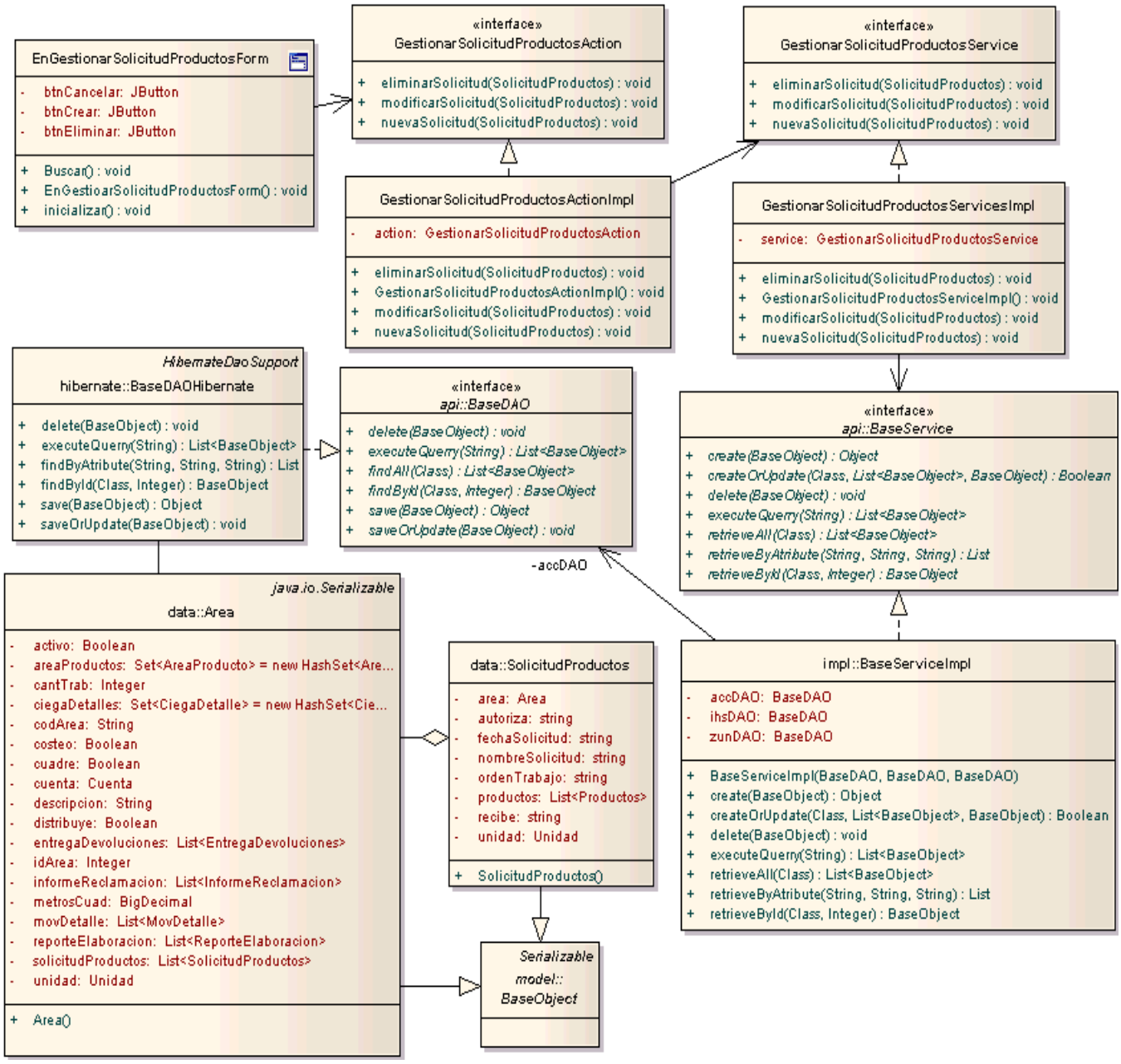



Figura 3.7: Diagrama de clases del caso de uso CU Gestionar Solicitud de Productos.

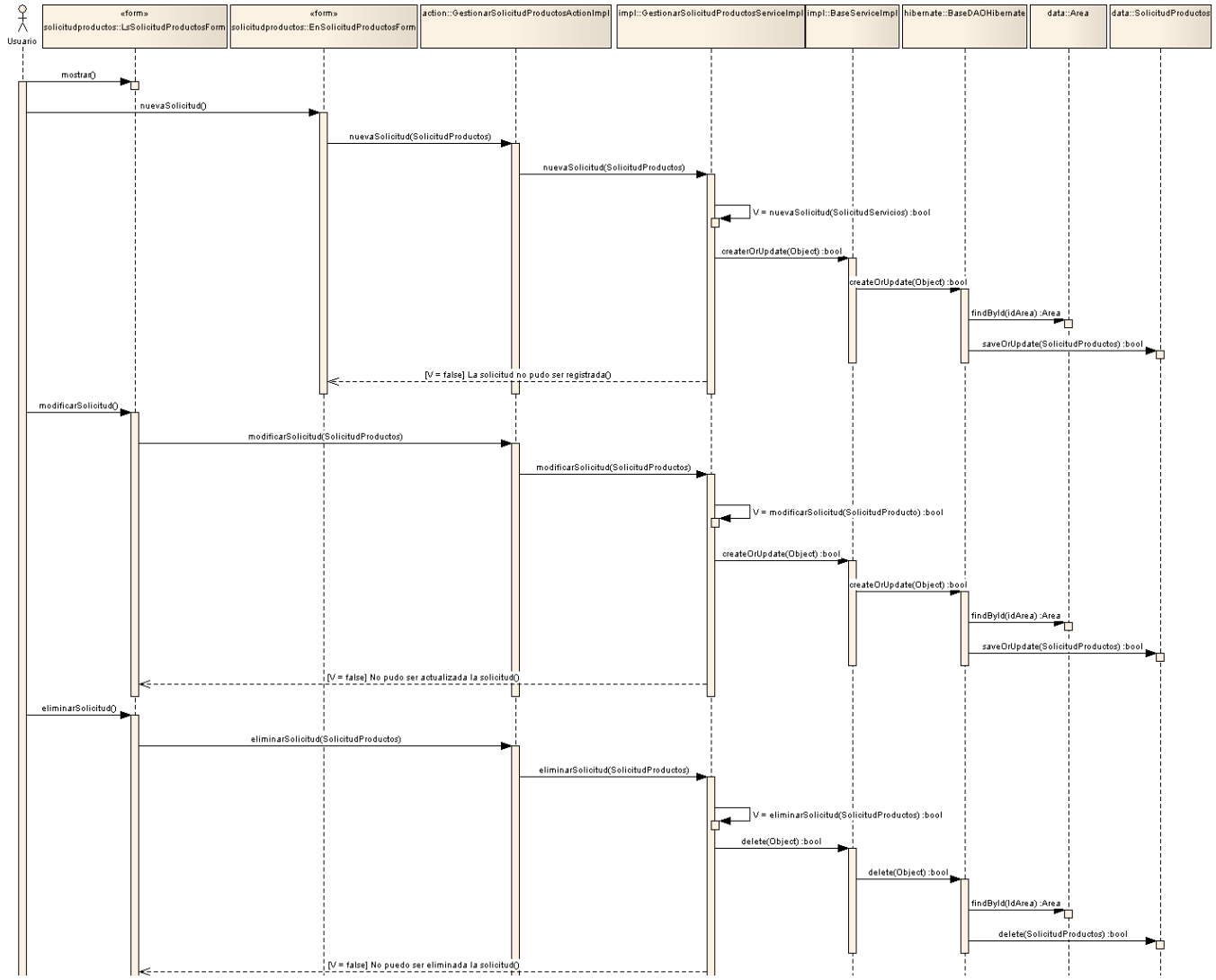



Figura 3.8: Diagrama de secuencia del caso de uso CU Gestionar Solicitud de Productos.

**CU Gestionar Recepción Ciega:**

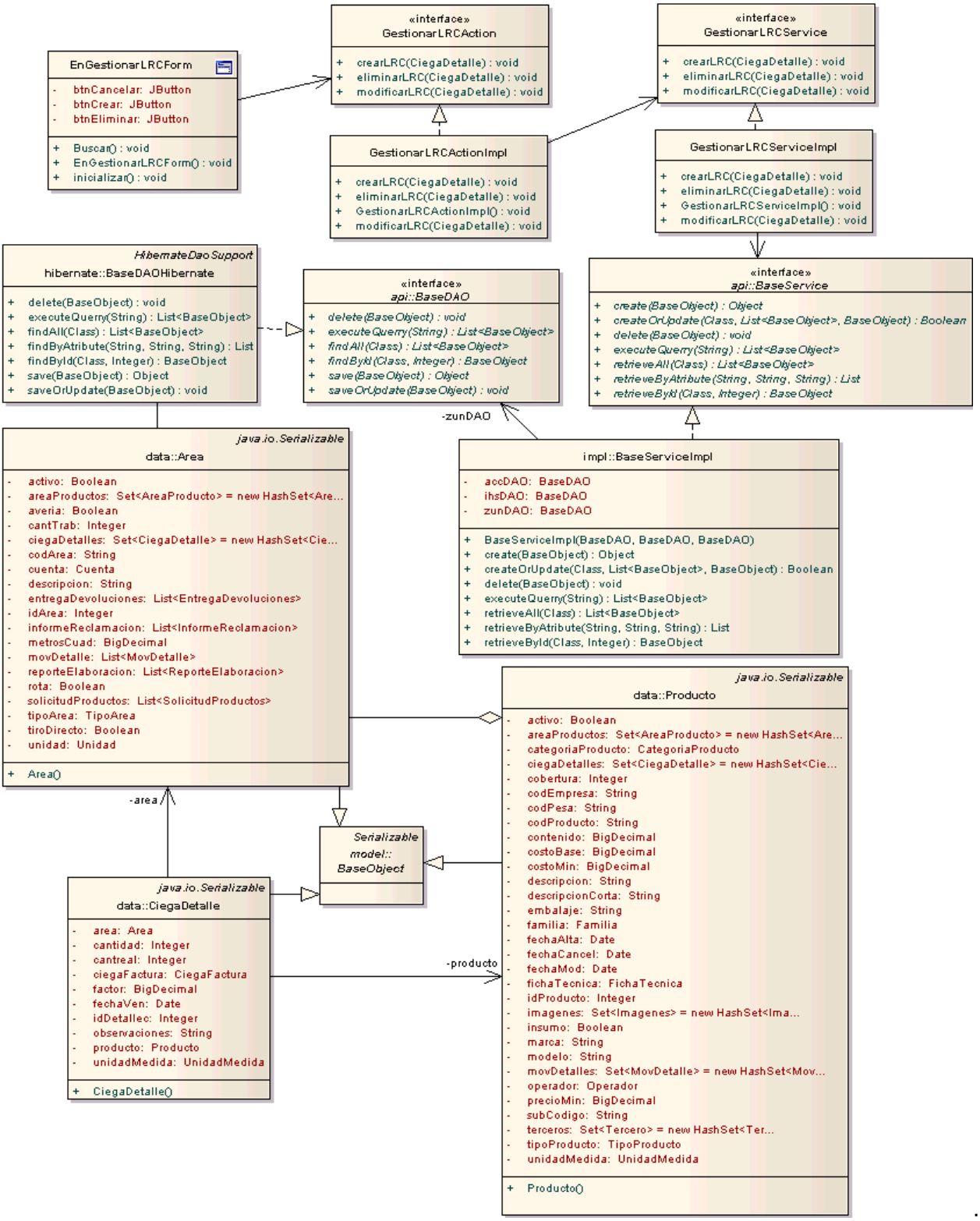



Figura 3.9: Diagrama de clases del caso de uso CU Gestionar Recepción Ciega.

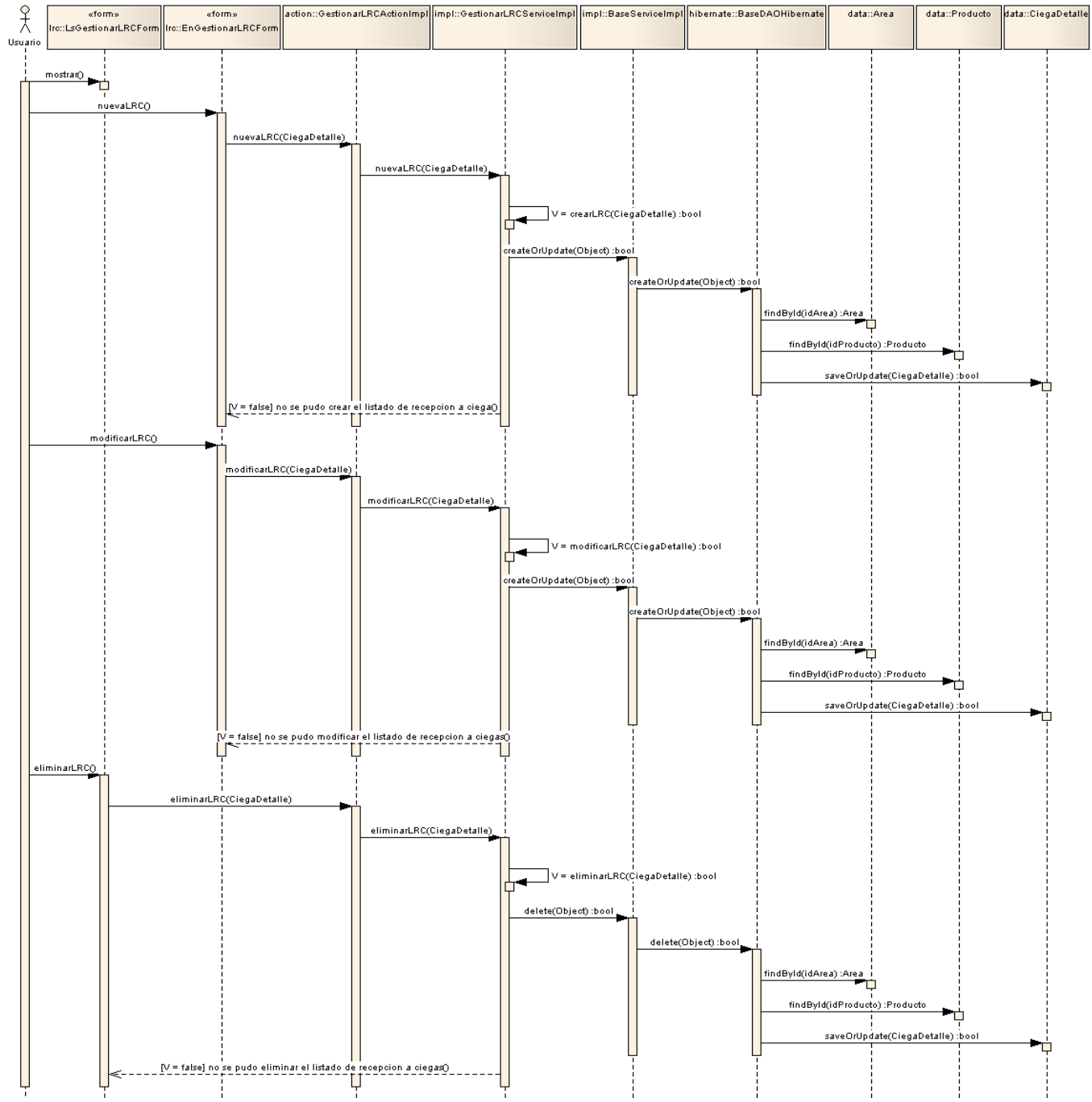



Figura 3.10: Diagrama de secuencia del caso de uso CU Gestionar Recepción Ciega.

**CU Gestionar Existencia de Productos:**

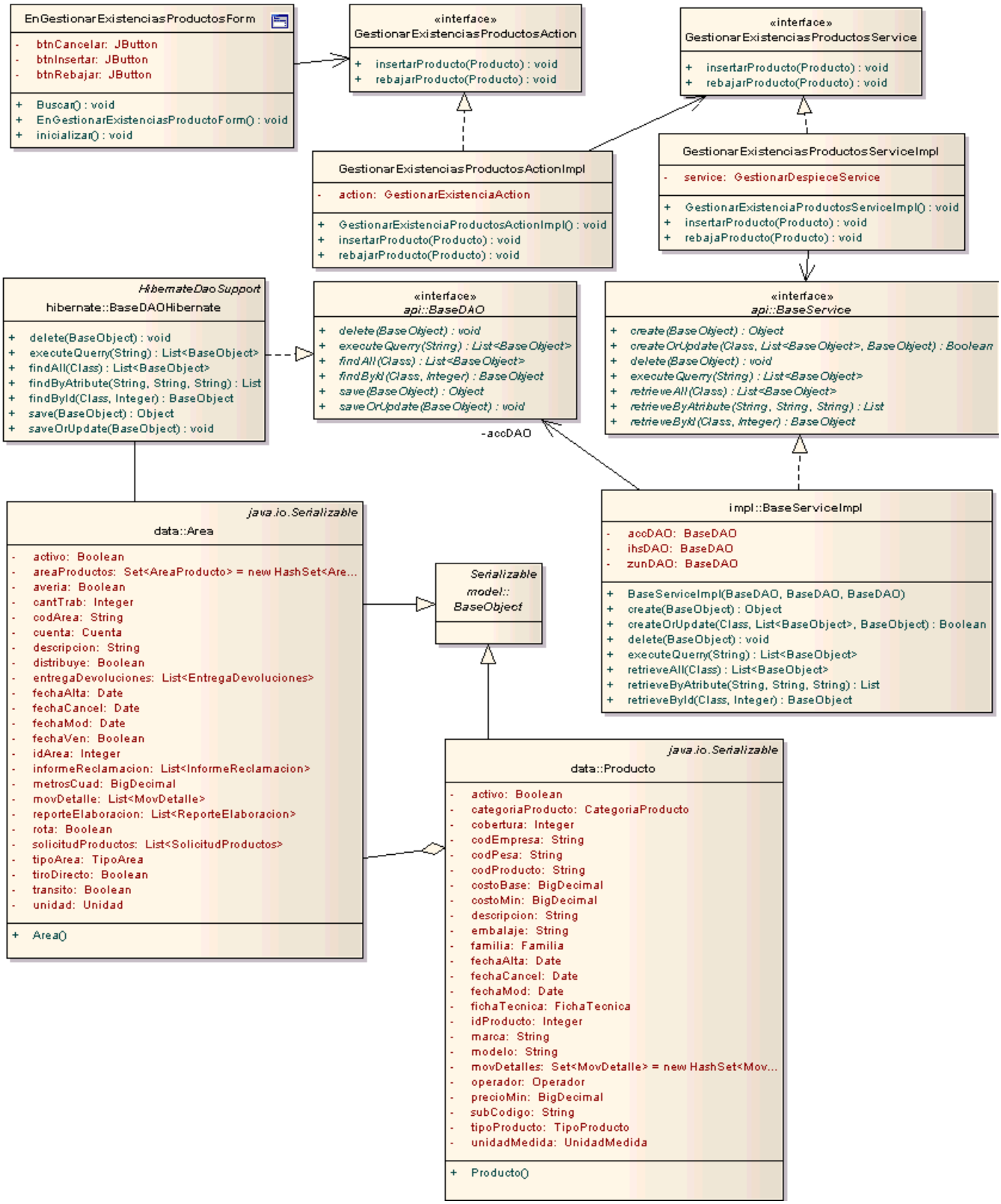



Figura 3.11: Diagrama de clases del caso de uso CU Gestionar Existencia de Productos.

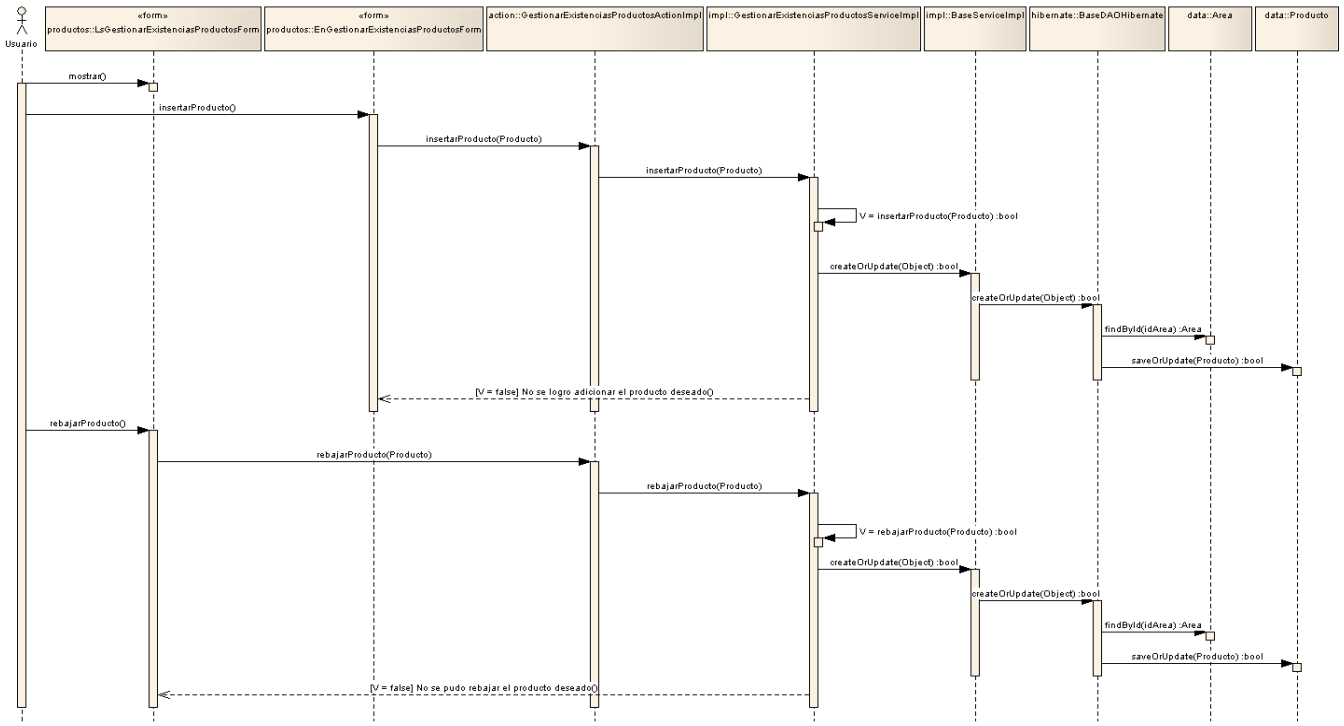



Figura 3.12: Diagrama de secuencia del caso de uso CU Gestionar Existencia de Productos.

**CU Gestionar Despiece:**

Ver diagrama de clases en Anexo 30 y diagrama de secuencia en Anexo 35.

**CU Gestionar Entrega o Devolución:**

Ver diagrama de clases en Anexo 32 y diagrama de secuencia en Anexo 36.

**CU Gestionar Ficha Técnica de Despiece:**

Ver diagrama de clases en Anexo 31 y diagrama de secuencia en Anexo 37.

**CU Gestionar Reclamación:**

Ver diagrama de clases en Anexo 33 y diagrama de secuencia en Anexo 38.

**CU Transferencia Productos:**

Ver diagrama de clases en Anexo 34 y diagrama de secuencia en Anexo 41.

## 3.3 Validación de la solución propuesta.

En este epígrafe se valida el trabajo realizado durante la elaboración del documento mediante métricas. Estas tienen por objetivo medir la calidad de los productos intermedios generados en un proyecto de software.

La palabra calidad designa el conjunto de atributos o propiedades de un objeto que permiten emitir un juicio de valor acerca de él.

El modelo de métricas define cuatro atributos genéricos de propiedades de calidad: consistencia, correctitud, completitud y complejidad, que son propiedades que caracterizan los productos obtenidos en cada fase del ciclo de vida de un proyecto, y que tienen un significado concreto de acuerdo al tipo de artefacto software y al nivel de abstracción que este describe. Un atributo se analiza en términos de un conjunto de factores cada uno de los cuales tendrá asociada una métrica.

### *3.3.1 Métricas para evaluar los requisitos.*

El personal de calidad realizó revisiones técnicas formales durante el levantamiento de los requisitos y aplico métricas para evaluar la calidad de los mismos.

Se utilizó la siguiente métrica donde:

**NR:** Requisitos que hay en una especificación.
**NF:** Es el número de Requisitos Funcionales.
**NNF:** Es el número de Requisitos No Funcionales.
**NR = NF + NNF**
NF = 47
NNF = 29
NR = 47 + 29 = 76

**Q1:** Consistencia de la interpretación de los revisores
**Nu1:** Es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

**Q1 = Nu1 / NR**
Q1 = 75/ 76
Q1 = 0.99

Después de haber aplicado esta métrica se demostró, gracias a que la mayoría de las interpretaciones de los revisores coincidían, que las especificaciones de los requisitos presentan un alto grado de claridad, ya que los requisitos responden a una única interpretación, pues el valor obtenido Q1 es bastante cercano a 1.

### *3.3.2 Métricas para evaluar los casos de uso.*

Una forma de evaluar la calidad de los casos de uso es mediante métricas. A continuación se explican los atributos, factores y métricas que se tuvieron en cuenta para la validación del Diagrama de Casos de Uso.

- ✓ Completitud Nivel Conceptual: Grado en que se ha logrado definir de forma clara y concisa todos los casos de uso del negocio.
- ✓ Nivel Especificación: Grado en que se ha logrado detallar todos los casos de uso relevantes.
- ✓ Consistencia Nivel Conceptual: Grado en que los casos de uso del negocio representan en forma única y no contradictoria los requerimientos funcionales.
- ✓ Nivel Especificación: Grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema.
- ✓ Correctitud Nivel Conceptual: Grado en que los casos de uso del negocio son entendidos y aceptados por el usuario.
- ✓ Nivel Especificación: Grado en que las interacciones actor / sistema soportan adecuadamente el proceso del negocio.
- ✓ Complejidad: Grado de claridad en la presentación de los elementos que describen el contexto y funcionalidad del sistema.

| Atributo | Factor | Métrica |
|---|---|---|
| Completitud | Factor 3. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información? | Métrica 3: el número de roles relevantes omitidos es 0.<br><br>Se presenta un 100%. |
| | Factor 6. ¿Se presenta una descripción resumida | Métrica 6: el número de casos de uso que no tiene descripción resumida es 0. |

| | | |
|---|---|---|
| | (descripción de alto nivel) de todos los casos de uso del negocio? | Se presenta un 100%. |
| | Factor 7. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso? | Métrica 7: el número de requisitos omitidos por caso de uso es 0.<br>Se presenta un 100%.<br><br>Métrica 8: el número de casos de uso que tienen requisitos omitidos es 0.<br>Se presenta un 100%. |
| | Factor 13. ¿Todos los casos de uso del negocio han sido clasificados de acuerdo a su relevancia (primario / secundario / opcional)? | Métrica 15: el número de casos de uso que no han sido clasificados es 0.<br><br>Se presenta un 100%. |
| | | Se presenta un 100%. |
| Consistencia | Factor 14. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario? | Métrica 16: el número de casos de uso que tienen un nombre incorrecto es 0.<br><br>Se presenta un 100%. |
| | Factor 18. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos? | Métrica 21: Grado de adecuación de la descripción del flujo de eventos para un caso de uso<br>La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción.<br>Se presenta un 100%. |
| | Factor 19. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del | Métrica 23: el número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema es 0.<br><br>Se presenta un 100%. |

| | | |
|---|---|---|
| | sistema claramente identificable? | |
| | Factor 21. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados? | Métrica 25: el número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos es 0.<br>Se presenta un 100%. |
| | | Se presenta un 100%. |
| Correctitud | Factor 23. ¿Representa el caso de uso requisitos comprensibles por el usuario? | Métrica 28: el número de casos de uso en que los requisitos representados no son comprensibles por el usuario es 0.<br>Se presenta un 100%. |
| | Factor 25. ¿Las interacciones definidas describen la funcionalidad requerida del sistema? | Métrica 31: el número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema es 1.<br>Este caso de uso representa el 9,09% del total.<br>El rango permitido es hasta el 10%, así que se considera aceptable.<br>Acción sugerida: se debe modificar el caso de uso de forma que se adecue a las necesidades reales del sistema y sus clientes.<br>Se presenta un 90,91%. |
| | Factor 26. ¿Las interacciones definidas introducen mejoras al proceso actual? | Métrica 32: el número de casos de uso que deben ser modificados para mejorar el proceso actual es 1<br>Este caso de uso representa el 9,09% del total.<br>El rango permitido es hasta el 20%, así que se considera aceptable.<br>Acción sugerida: se debe analizar la situación descrita en la interacción y estudiar la manera de mejorar el proceso con el uso de la tecnología informática.<br>Se presenta un 90,91%. |
| | | Se presenta un 93,94%. |
| Complejidad | Factor 29. ¿Los elementos dentro | Métrica 36: el número de elementos del |

| | | |
|---|---|---|
| | del diagrama están adecuadamente ubicados de manera que facilitan su interpretación? | diagrama que requieren reubicación es 0. Se presenta un 100%. |
| | | Se presenta un 100%. |

Tabla 4.1. Resultado de las métricas aplicadas a los casos de uso.

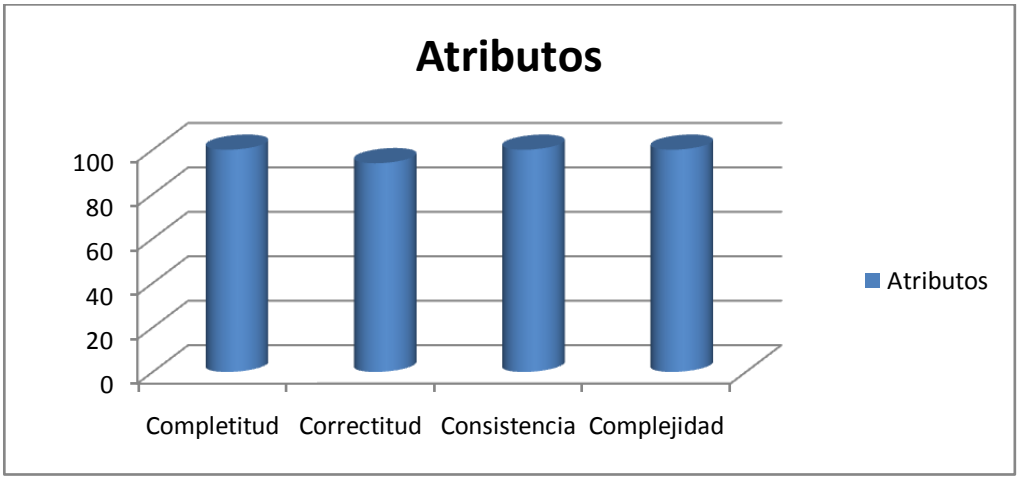



Figura 4.1. Resultados gráficos de la aplicación de las métricas para evaluar la calidad de los casos de uso.

### *3.3.3 Métricas para evaluar el diseño.*

#### *3.3.3.1 Acoplamiento*

Este principio responde a uno de los Patrones de Asignación de Responsabilidades, el Bajo Acoplamiento. El diseño propuesto está compuesto por dos módulos, entre los cuales se considera que existe un bajo acoplamiento. Eso lo demuestra la independencia que existe entre ellos, pues son pocas las llamadas entre uno y otro. El módulo de Administración invoca al módulo Inventario una vez que el usuario se autentique y se le asignen los permisos correspondientes a su rol.

En las especificaciones de los casos de usos explicados en el capítulo anterior se puede observar perfectamente la relación existente entre los módulos.

#### *3.3.3.2 Tamaño de Clase.*

La métrica de tamaño de clase pertenece a la familia de métricas propuesta por Lorens y Kidd. Ellos proponen que para medir el tamaño de clases se deben tener los siguientes aspectos en cuenta:

- ✓ Total de operaciones, ya sean las de la clase o las heredadas de las clases padres o interfaces que implementen.
- ✓ Cantidad de atributos, al igual que el anterior, tanto los de ella, como los de los padres.
- ✓ Promedio general de los dos anteriores para el sistema completo.

Si existen valores grandes de Tamaño de Clase estos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación. Por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

Para evaluar las métricas son necesarios los valores de los umbrales. Existen tres umbrales para esta métrica los que responderán a la clasificación de la clase. Estos son:

| Clasificación | Valores de los umbrales |
|---|---|
| Pequeño | <=20 |
| Medio | >20 y <=30 |
| Grande | >30 |

Tabla 4.2: Valores de los umbrales según su clasificación.

**Resultados:** Esta métrica fue aplicada a las diferentes capas de desarrollo del sistema. Se le aplicó a las tres capas definidas: Presentación, Negocio y Acceso a Datos.

En la Capa de Presentación hay un total de 55 clases. De acuerdo a los umbrales que muestra la Tabla 4.2, hay 45 clases pequeñas, lo que representa un 82%, 9 clases medianas para un 16% y 1 clase grande para un 2%.

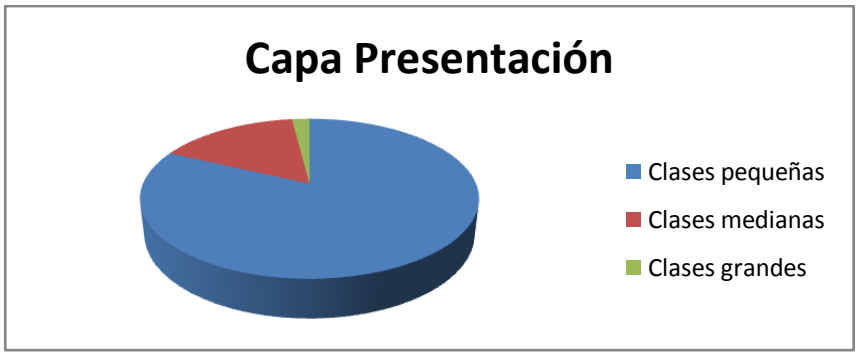



Figura 4.2. Resultados de las métricas aplicadas a la Capa de Presentación.

En la Capa de Negocio hay un total de 32 clases. De acuerdo a los umbrales de la Tabla 4.2 hay un resultado de 30 clases pequeñas, lo que representa un 94% y 2 medianas, para un 6%.


Figura 4.3. Resultados de las métricas aplicadas a la Capa de Negocio.

En la Capa de Acceso a Datos hay un total de 7 clases. De acuerdo a los umbrales de la Tabla 4.2, las 7 clases se encuentran dentro de la clasificación de pequeñas, lo que significa un 100%.

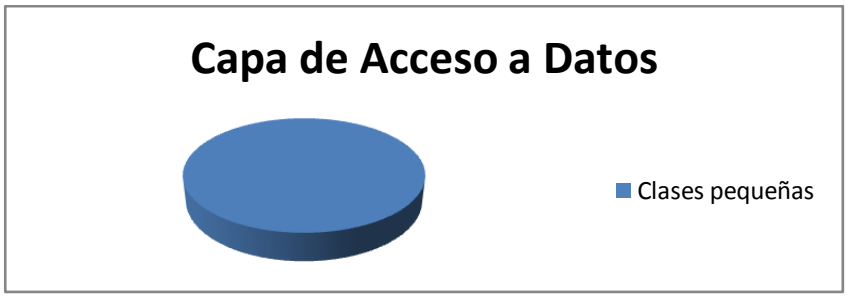
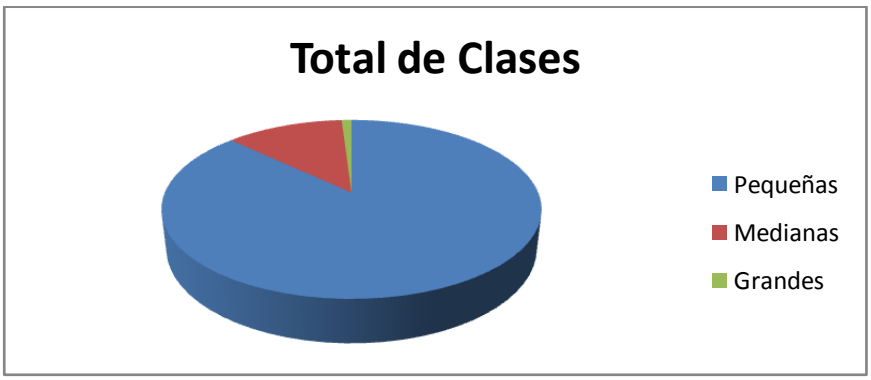



Figura 4.4. Resultados de las métricas aplicadas a la Capa de Acceso a Datos.

En total, hay 94 clases clasificadas entre grandes, medianas y pequeñas, para las que se cumple que grandes hay un 1%, medianas un 12% y pequeñas un 87%.



Estos valores demuestran que los indicadores de calidad reutilización, implementación y responsabilidad no se ven afectados.

#### *3.3.3.3 Árbol de Profundidad de Herencia (APH).*

La métrica Árbol de Profundidad de Herencia fue propuesta por Chidamber y Kemerer (CK). Mide el máximo nivel de jerarquía de herencia realizando una cuenta directa. En el nivel cero de la jerarquía se ubica la clase raíz.

CK propone esta métrica como medida de la complejidad de una clase, del diseño y del potencial de reusabilidad de estos. Esto es debido a que cuanto más profunda se encuentra una clase en la jerarquía, mayor es la probabilidad de heredar un mayor número de métodos.

En este sistema no se presenta un alto nivel de jerarquía de herencia. En la mayoría de los casos es mediante las interfaces, para que varias clases hereden el comportamiento de ellas, y además para que otras puedan heredar de una clase normal y de una interface, quedando solucionado el problema de la herencia múltiple.

Aplicando esta métrica al diseño propuesto se obtiene como resultado que el APH presenta valor 1, pues solo hay presencia de herencia entre las clases y las interfaces de las cuales heredan sus funciones y las redefinen, por lo que existe bajo acoplamiento y los cambios son más fáciles de controlar y gestionar.

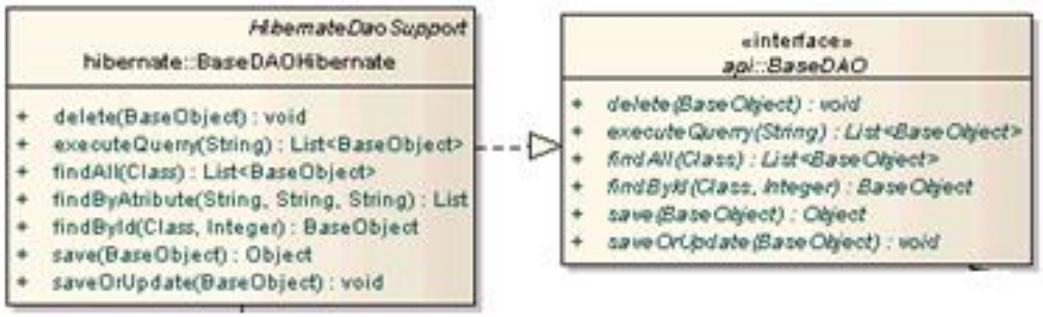



Figura 4.6. Ejemplo de herencia entre una clase interfaz y la clase que la redefine.

### 3.4 Conclusiones.

Se realizó el diseño de clases de los casos del sistema que componen los módulos en los que se divide el mismo, aplicando así uno de los principios del diseño propuesto por Pressman, la modularidad. Se aplicaron patrones de diseño, algunos presentes en los frameworks utilizados en el

desarrollo del sistema, que permiten el desarrollo de forma ágil y eficaz. Se definieron los subsistemas en los que se divide el sistema, de forma que se facilite su desarrollo. Se realizaron diagramas de secuencia, que permiten un entendimiento del flujo de sucesos de los casos de uso.

El empleo de métricas para evaluar la calidad de los requisitos permitió confirmar que el levantamiento de los requisitos fue bastante correcto, arribando a la conclusión de que se realizaron buenas prácticas en esta etapa. El modelo del sistema fue también evaluado con métricas, demostrando que se identificaron y relacionaron correctamente los actores y funcionalidades del sistema y sus descripciones. Se logró confirmar que el diseño realizado es bastante robusto, lo que permite una mayor confiabilidad a la hora de implementar el sistema.

CONCLUSIONES

- Se realizó un estudio del estado del arte de las disciplinas a desarrollar, las herramientas a utilizar para modelar, el lenguaje de modelado y la metodología a tener en cuenta, los patrones de casos de uso y de diseño, las métricas para evaluar la calidad de diseño. Esto permitió la preparación teórica que sustenta el desarrollo del trabajo.

- Se aplicaron algunas técnicas para capturar los requisitos, tanto funcionales como no funcionales, del sistema. Se tuvieron en cuenta además, algunos patrones para modelar el sistema y el diseño, lo que permitió generar artefactos empleando buenas prácticas necesarias para desarrollar con mayor robustez.

- Se evaluó la calidad de los principales artefactos generados, los requisitos, los casos de uso del sistema y el diseño, mediante métricas definidas por autores reconocidos, lo que permitió confirmar la realización de artefactos confiables y con calidad.

RECOMENDACIONES

Se recomienda realizar otras iteraciones para optimizar el proceso de levantamiento de requisitos.

Se recomienda realizar la implementación de la propuesta que se presenta en este trabajo, con el fin de obtener al menos una versión del producto.

BIBLIOGRAFÍA.

**Abanq. 2008.** Abanq. [En línea] 2008. http://abanq.org/.

**Adempiere. 2008.** [En línea] 2008. http://www.adempiere.com.

**Adpime. 2008.** ERP - Sistemas de Gestión PYME. [En línea] 2008. http://www.adpime.com/ERP/Es_ERP_intro.htm.

**Aguilar, Arabel y Lalangui, Galo. 2008.** *Procesos de Ingeniería de Software.* Universidad Tecnica Particular de Loja : Escuela de Ciencias de la Computación, 2008.

**Alexander, Christopher, y otros. 1977.** *A Pattern Language.* New York : Oxford University Press, 1977.

**Beck, Kent y Andres, Cynthia. 2004.** *Extreme Programming Explained: Embrace Change. 2da edición. .* s.l. : Addison-Wesley, 2004.

**Berlinches Cerezo, Andrés. 2004.** *Calidad.* s.l. : Cengage Learning Editores, 2004. ISBN: 8497320832.

**Boehm, Barry W. 1976.** *Software Engineering.* 1976.

**Carlos, Universidad Rey Juan. s/a.** *Patrones de Diseño.* [http://www.escet.urjc.es/~gtazon/IS/ConceptosDiseno.pdf] s/a.

**Christel, Michael G. y Kang, Kyo C. 1992.** *Issues in Requirements Elicitation.* Carnegie-Mellon University, Software Engineering Institute : s.n., 1992.

**Dávila, Nicolás Davyt. 2001.** *Ingeniería de Requerimientos una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto.* 2001.

**Davis, Alan M. 1990.** *Software Requirements: Analysis and Specification.* Universidad de Michigan : Prentice Hall, 1990. ISBN 0138246734.

**Dormido, Sebastián. 2006.** Addlink Software Cientifico. [En línea] Julio de 2006. http://www.addlink.es/productos.asp?pid=673.

**Escalona, María José y Koch, Nora. 2002..** *Ingeniería de Requisitos en Aplicaciones para la Web– Un estudio comparativo.* España. : Universidad de Sevilla Lenguajes y Sistemas Informáticos, 2002.

**Fernández Carballo., Leamny Teresa y Castellanos Rolo., Betxy. 2007.** *Estrategia metodológica para el desarrollo de Software de Gestión a Distancia basado en Programación Extrema.* Habana : s.n., 2007.

**Fernández Escribano, Gerardo. 2002.** *Introducción a Extreme Programming.* 2002.

**Ford, G. y Raghavan, S. 1994.** *Lecture Notes on Requirements Elicitation.* s.l. : S. E. Institute, Carnegie Mellon University, 1994.

**Fowler, M. 1996.** *Analysis Patterns, Reusable Object Models. Reading.* s.l. : Addison-Wesley, 1996.

**Gamma, Erich, y otros. 1995.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison Wesley, 1995. ISBN:0201633612 .

**Gómez Vela, Francisco A.** *Preparación y realización de entrevistas con clientes (Una visión personal).* Sevilla : Universidad de Sevilla, escuela tecnica superior de ingenieria informatica.

**González Rodríguez, Victoria. 2003.** *El impacto de un ERP en la empresa .* 2003.

**Gracia, Joaquin. 2005.** *Patrones de diseño, diseño de software orientado a objetos.* 2005.

**Hernández León, Rolando Alfredo y Coello González, Sayda. 2002.** *El paradigma cuantitativo de la investigación científica.* La Habana : Editorial Universitaria, 2002. ISBN: 959-16-0343-6.

**IEEE. 1993.** *Standards Collection: Software Engineering.* s.l. : IEEE Standard 610.12-1990, 1993.

**Informática, Siglo 21. 2006.** *Documento De Especificación Requerimientos No Funcionales del Proyecto Mejoramiento de procesos, Análisis y Diseño del Sistema de Información para la Vigilancia de eventos en salud pública en la Fase 1: Subsistema básico general y Subsistema de vigilancia.* Bogotá, Colombia : s.n., 2006.

**Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

**Kendall, K. 1997.** *Analisis y diseño de sistemas.* s.l. : Prentice Hall., 1997.

**Koch, Nora y María, José. 2002.** *Ingeniería de Requisitos en Aplicaciones para la Web– Un estudio comparativo.* España : Universidad de Sevilla Lenguajes y Sistemas Informáticos, 2002.

**Larman, Craig. 2003.** *Agile and Iterative Development: A Manager's Guide.* s.l. : Addison-Wesley, 2003. ISBN:0131111558.

**Larman, Graig. 2004.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pentrice Hall, 2004.

**Lewis, G. 1994.** *What is Software Engineering?* 1994.

**Olmedilla Arregui, Juan José. 2005.** *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* Universidad Politécnica de Madrid, Facultad.de Informática : s.n., 2005.

**Openbravo. 2008.** [En línea] 2008. http://www.openbravo.com.

**Openxpertya. 2008.** Openxpertya. [En línea] 2008. http://www.openxpertya.org/index.php?option=com_content&task=view&id=1&Itemid=21.

**Övergaard, Gunnar y Palmkvist, Karin. 2004.** *Use Case Patterns and Blueprints @Team LiB.* Stockholm, Sweden : s.n., 2004.

**Perera, Jose Raul. 2007.** *Arquitectura de software para Sistema de Gestión de inventarios.* s.l. : UCI, 2007.

**Peru, Instituto Nacional de estadistica e informatica de. 1999.** *Herramientas Case.* 1999.

**—.** Realidad Virtual. [En línea] http://www.inei.gob.pe/biblioineipub/bancopub/inf/Lib5047/INDEX.htm.

**Piattini, M. 1996.** *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Madrid. : s.n., 1996.

**Piñeiro Pérez, Yaniet y Plaza Matos, Mairelys. 2007.** *Predictor: Sistema de descarga y procesamiento automatizado de patentes. Rol Analista de sistemas.* Habana : s.n., 2007.

**Pressman, Roger S. 2001.** *Ingeneria de Software, un enfoque practico.* 2001.

**—. 2004.** *Software Engineering: A Practitioner's Approach.* s.l. : McGraw-Hill, 2004. ISBN:0072853182.

**Ramos González, Juan José.** *PML-A modeling Language for Physical Knowledge Representation.* Barcelona : Universidad Autonoma de Barcelona.

**Rational, Software Corporation. 2003.** *Ayuda del Rational Unified Process.* 2003.

**Robertson, Suzanne y Robertson, James. 2006.** *Mastering the Requirements Process Second Edition.* s.l. : Addison Wesley Professional, 2006.

**Schwaber, Ken y Beedle, Mike. 2001.** *Agile Software Development with Scrum.* s.l. : Prentice Hall, 2001. ISBN:0130676349.

**Sommerville, Ian. 2004.** *Software Engineer.* s.l. : Addison-Wesley, 2004.

**Sparxsystems, Empresa. 2008.** [En línea] 2008. [Citado el: 3 de junio de 2008.] http://www.sparxsystems.com.ar/products/ea_features.html.

**Torres, José Luis. 2008.** Promoting Community Wordwide. *Especicifacion de requisitos en Ingenieria de Software.* [En línea] 2008. http://www.uag.mx/ieee/contsep01/requerimientos.htm.

**Wailgum, Thomas. 2008.** ABC: An introduction to ERP. [En línea] 2008. http://www.cio.com/research/erp/edit/erpbasics.html.

**Young, Ralph R. 2004.** *The Requirements Engineering Handbook.* London : s.n., 2004.

**Zapata, Carlos y Arango, J. 2004.** *Alineación entre Metas Organizacionales y Elicitación de Requisitos del Software.* Colombia : Universidad Nacional de Colombia. Red de Revistas Científicas, 2004.

REFERENCIAS BIBLIOGRÁFICAS.

**Abanq. 2008.** Abanq. [En línea] 2008. http://abanq.org/.

**Adempiere. 2008.** [En línea] 2008. http://www.adempiere.com.

**Adpime. 2008.** ERP - Sistemas de Gestión PYME. [En línea] 2008. http://www.adpime.com/ERP/Es_ERP_intro.htm.

**Aguilar, Arabel y Lalangui, Galo. 2008.** *Procesos de Ingeniería de Software.* Universidad Tecnica Particular de Loja : Escuela de Ciencias de la Computación, 2008.

**Beck, Kent y Andres, Cynthia. 2004.** *Extreme Programming Explained: Embrace Change. 2da edición. .* s.l. : Addison-Wesley, 2004.

**Berlinches Cerezo, Andrés. 2004.** *Calidad.* s.l. : Cengage Learning Editores, 2004. ISBN: 8497320832.

**Boehm, Barry W. 1976.** *Software Engineering.* 1976.

**Carlos, Universidad Rey Juan. s/a.** *Patrones de Diseño.* [http://www.escet.urjc.es/~gtazon/IS/ConceptosDiseno.pdf] s/a.

**Christel, Michael G. y Kang, Kyo C. 1992.** *Issues in Requirements Elicitation.* Carnegie-Mellon University, Software Engineering Institute : s.n., 1992.

**Dávila, Nicolás Davyt. 2001.** *Ingeniería de Requerimientos una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto.* 2001.

**Davis, Alan M. 1990.** *Software Requirements: Analysis and Specification.* Universidad de Michigan : Prentice Hall, 1990. ISBN 0138246734.

**Dormido, Sebastián. 2006.** Addlink Software Cientifico. [En línea] Julio de 2006. http://www.addlink.es/productos.asp?pid=673.

**Fernández Carballo., Leamny Teresa y Castellanos Rolo., Betxy. 2007.** *Estrategia metodológica para el desarrollo de Software de Gestión a Distancia basado en Programación Extrema.* Habana : s.n., 2007.

**Fernández Escribano, Gerardo. 2002.** *Introducción a Extreme Programming.* 2002.

**Ford, G. y Raghavan, S. 1994.** *Lecture Notes on Requirements Elicitation.* s.l. : S. E. Institute, Carnegie Mellon University, 1994.

**Fowler, M. 1996.** *Analysis Patterns, Reusable Object Models. Reading.* s.l. : Addison-Wesley, 1996.

**Gamma, Erich, y otros. 1995.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison Wesley, 1995. ISBN:0201633612 .

**Gómez Vela, Francisco A.** *Preparación y realización de entrevistas con clientes (Una visión personal).* Sevilla : Universidad de Sevilla, escuela tecnica superior de ingenieria informatica.

**González Rodríguez, Victoria. 2003.** *El impacto de un ERP en la empresa .* 2003.
**Gracia, Joaquin. 2005.** *Patrones de diseño, diseño de software orientado a objetos.* 2005.
**IEEE. 1993.** *Standards Collection: Software Engineering.* s.l. : IEEE Standard 610.12-1990, 1993.
**Informática, Siglo 21. 2006.** *Documento De Especificación Requerimientos No Funcionales del Proyecto Mejoramiento de procesos, Análisis y Diseño del Sistema de Información para la Vigilancia de eventos en salud pública en la Fase 1: Subsistema básico general y Subsistema de vigilancia.* Bogotá, Colombia : s.n., 2006.
**Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.
**Kendall, K. 1997.** *Analisis y diseño de sistemas.* s.l. : Prentice Hall., 1997.
**Koch, Nora y María, José. 2002.** *Ingeniería de Requisitos en Aplicaciones para la Web– Un estudio comparativo.* España : Universidad de Sevilla Lenguajes y Sistemas Informáticos, 2002.
**Larman, Craig. 2003.** *Agile and Iterative Development: A Manager's Guide.* s.l. : Addison-Wesley, 2003. ISBN:0131111558.
**Larman, Graig. 2004.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pentrice Hall, 2004.
**Olmedilla Arregui, Juan José. 2005.** *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* Universidad Politécnica de Madrid, Facultad.de Informática : s.n., 2005.
**Openbravo. 2008.** [En línea] 2008. http://www.openbravo.com.
**Openxpertya. 2008.** Openxpertya. [En línea] 2008. http://www.openxpertya.org/index.php?option=com_content&task=view&id=1&Itemid=21.
**Övergaard, Gunnar y Palmkvist, Karin. 2004.** *Use Case Patterns and Blueprints @Team LiB.* Stockholm, Sweden : s.n., 2004.
**Perera, Jose Raul. 2007.** *Arquitectura de software para Sistema de Gestión de inventarios.* s.l. : UCI, 2007.
**Peru, Instituto Nacional de estadistica e informatica de.** Realidad Virtual. [En línea] http://www.inei.gob.pe/biblioineipub/bancopub/inf/Lib5047/INDEX.htm.
**Piattini, M. 1996.** *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Madrid. : s.n., 1996.
**Piñeiro Pérez, Yaniet y Plaza Matos, Mairelys. 2007.** *Predictor: Sistema de descarga y procesamiento automatizado de patentes. Rol Analista de sistemas.* Habana : s.n., 2007.
**Pressman, Roger S. 2001.** *Ingeneria de Software, un enfoque practico.* 2001.
**Ramos González, Juan José.** *PML-A modeling Language for Physical Knowledge Representation.* Barcelona : Universidad Autonoma de Barcelona.

**Rational, Software Corporation. 2003.** *Ayuda del Rational Unified Process.* 2003.

**Robertson, Suzanne y Robertson, James. 2006.** *Mastering the Requirements Process Second Edition.* s.l. : Addison Wesley Professional, 2006.

**Sommerville, Ian. 2004.** *Software Engineer.* s.l. : Addison-Wesley, 2004.

**Sparxsystems, Empresa. 2008.** [En línea] 2008. [Citado el: 3 de junio de 2008.] http://www.sparxsystems.com.ar/products/ea_features.html.

**Torres, José Luis. 2008.** Promoting Community Wordwide. *Especicifacion de requisitos en Ingenieria de Software.* [En línea] 2008. http://www.uag.mx/ieee/contsep01/requerimientos.htm.

**Wailgum, Thomas. 2008.** ABC: An introduction to ERP. [En línea] 2008. http://www.cio.com/research/erp/edit/erpbasics.html.

**Young, Ralph R. 2004.** *The Requirements Engineering Handbook.* London : s.n., 2004.

**Zapata, Carlos y Arango, J. 2004.** *Alineación entre Metas Organizacionales y Elicitación de Requisitos del Software.* Colombia : Universidad Nacional de Colombia. Red de Revistas Científicas, 2004.

ANEXOS

**Anexo 1. Listado de Recepción Ciega**

| No.: (1) | Referencia: (2) | Área: (3) | | **Listado para Recepción Ciega** | | | |
|---|---|---|---|---|---|---|---|
| Proveedor: (4) | | | | | | | |
| Código: | Descripción: | | | | | | |
| Productos: | Código: | Descripción: | U/M: | P. Costo: | P. Venta: | F. Venc: | Cant. Real: |
| (5) | (6) | (7) | (8) | (9) | (10) | (11) | (12) |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| Receptor: (13) | | Suministrador: (14) | | Fecha de recepción: (15) | | | |
| Nombre: | | Nombre: | | Día: | Mes: | | Año: |
| Firma: | | Firma: | | | | | |

(1) Número de la Recepción Ciega. Este número es un consecutivo único para la unidad. Se genera automáticamente al salvarse el documento.
(2) Referencia del documento origen.
(3) Área de destino de la mercancía.
(4) Código y descripción del proveedor.
(5) Productos recepcionados.
(6) Código del producto.
(7) Descripción del producto.
(8) Unidad de medida del producto.
(9) Precio de costo del producto.
(10) Precio de venta del producto.
(11) Fecha de vencimiento del producto.
(12) Cantidad recepcionada del producto.
(13) Persona responsable de recibir la mercancía. Incluye nombre y firma.
(14) Persona que hace la entrega. Incluye nombre y firma.
(15) Fecha en que se recepciona la mercancía.

**Anexo 2. Factura.**

| Proveedor: (1) | Código: | Dirección: | | **Factura Modelo SC-2-12** | | | |
|---|---|---|---|---|---|---|---|
| Cuenta bancaria: (2) | | Sucursal: | NIT: () | | | | |
| Comprador: (3) | Código: | Dirección: | | | | | |
| Cuenta bancaria: (4) | Sucursal: | | | Transporta: (6) | | Firma: | |
| Operaciones: (5) | | | | CI: | | Chapa: | |
| | | | | Carta de Porte: | | Casilla: | |
| | | | | Fecha de transportación: | | | |
| | | | | Día: | Mes: | | Año: |
| | | | | | | | |
| Productos: (7) | | | | | | | |
| Producto: | Código: | Descripción: | Cantidad: | U/M: | Precio Unit. Total: | | Importe de c/u: |
| | | | | | | | |
| | | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| Total de la Factura: (8) | | | Número consecutivo del modelo: (9) | | | |
| Entrega: (10) | Recibe: (11) | | Contabiliza: (12) | Fecha de emisión: (13) | | |
| Nombre: | Nombre: | | | Día: | Mes: | Año: |
| Firma: | Firma: | | | | | |

(1) Nombre, dirección, código del proveedor.
(2) Número de la cuenta, de la sucursal y Número de Identificación Tributaria (NIT) del proveedor.
(3) Nombre, dirección y código del comprador.
(4) Número de la cuenta y de la sucursal bancaria del comprador.
(5) Operaciones por la que se emite: corrientes, inversiones, reparaciones corrientes o generales, activos fijos tangibles, etc.
(6) Datos del transportista: nombre, carné de identidad, chapa del vehículo, Carta de Porte número, casilla del ferrocarril y fecha de transportación.
(7) Código, descripción, unidad de medida, cantidad, precio unitario total e importe de cada producto.
(8) Total de la Factura.
(9) Número consecutivo del modelo.
(10) Nombre y firma de la persona que entrega los productos y fecha de la entrega.
(11) Nombre y firma de la persona que recibe los productos y fecha de la recepción.
(12) Firma de la persona que contabiliza la Factura y la anota en el control de Inventario.
(13) Fecha de emisión del modelo.

**Anexo 3. Informe de Recepción.**

| | | | | | | |
|---|---|---|---|---|---|---|
| Entidad receptora: (1) | | Código: | | **Informe de Recepción Modelo SC-2-04** | | |
| Almacén receptor: (2) | | Código: | | | | |
| Nombre del Proveedor: (3) | | Código: | | | | |
| Transportador: (4) | | | Firma: | | | |
| Nombre: | | | CI: | | Chapa: | |
| Productos: (5) | | | | | | |
| Producto: | Código: | Descripción: | U/M: | Precio Unit. Total: | Importe: | Saldo: |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| Importe Total: (6) | | | | | | |
| Nombre de Factura: (7) | Número: | | Número Consecutivo del Modelo: (8) | | | |
| Receptor: (9) | Jefe de Almacén: (10) | | Contabiliza: (11) | Fecha de emisión: (12) | | |
| Nombre: | Nombre: | | Nombre: | Día: | Mes: | Año: |
| Firma: | Firma: | | Firma: | | | |

(1) Nombre y código de la entidad receptora.
(2) Nombre y código del almacén receptor.
(3) Nombre y código del proveedor.
(4) Datos del Transportador: nombre, carné de identidad, chapa y firma.
(5) Código, descripción, unidad de medida, cantidad, precio unitario total, importe y saldo en existencia según almacén de cada producto.

(6) Importe Total del modelo.
(7) Número y nombre de Factura.
(8) Número consecutivo del modelo.
(9) Nombre y firma del empleado que recepciona.
(10) Nombre y firma del Jefe del Almacén.
(11) Nombre y firma del empleado que lo contabiliza.
(12) Fecha de emisión del modelo.

**Anexo 4. Vale de Entrega o Devolución.**

| Entidad: (1) | Código: | Fecha de emisión(5) | | | **Vale de Entrega o Devolución Modelo SC-2-08** |
|---|---|---|---|---|---|
| Almacén: (2) | Código: | | | | |
| Área: (3) | Código: | | | | |
| Centro de costo: (4) | Código: | Orden de Trabajo: | | | |

| Productos: (6) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Producto: | Código: | Descripción: | U/M: | Cantidad despachada o devuelta | Precio Unit. Total: | Importe: | Saldo: |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

| Importe Total: (7) | No. Del VED: (8) | Referencia: (9) |
|---|---|---|
| Receptor: (10) | Entrega: (11) | Contabiliza: (12) |
| Nombre: | Nombre: | Nombre: |
| Firma: | Firma: | Firma: |

(1) Nombre y código de la entidad.
(2) Nombre y código del almacén que entrega o al que se devuelven los productos.
(3) Nombre y código del área.
(4) Centro de costo al que se cargan los productos según se trate de entrega o devolución, código y Orden de Trabajo.
(5) Fecha de emisión del modelo.
(6) Código, descripción, unidad de medida, cantidad despachada o devuelta, precio unitario total, importe y saldo en existencia según almacén de cada producto.
(7) Importe Total del Vale o de la Devolución.
(8) Número consecutivo del Vale de Entrega o del de Devolución.
(9) En caso de ser un VED de salida, referencia del VED de entrega.
(10) Nombre y firma de la persona que recibe o entrega los productos devueltos.
(11)Nombre y firma de la persona que entrega o recibe los productos devueltos por el almacén.
(12) Nombre y firma de la persona que contabiliza la entrega o la devolución.

**Anexo 5. Informe de Reclamación.**

| Nombre: (1) | Código: | Dirección: | Fecha de emisión(13) | | | **Informe de Reclamación Modelo SC-2-05** |
|---|---|---|---|---|---|---|
| Proveedor: (2) | Código: | Dirección: | Día | Mes | Año | |
| Cuenta bancaria: (3) | | Sucursal: | | | | |
| Nombre: (4) | | Dirección: | CI: | | | Chapa: |

| Productos: (5) | | | | | | |
|---|---|---|---|---|---|---|
| Productos: | Código: | Descripción: | U/M: | Precio Unit. Total: | Importe: | Cant. Real: |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |

| | | |
|---|---|---|
| Importe Total de la Reclamación: (6) | No. de factura: (7) | No. consecutivo del Modelo: (8) |

| | | | |
|---|---|---|---|
| Receptor: (9) | Transportador: (10) | Proveedor: (11) | Contabiliza: (12) |
| Nombre: | Nombre: | Nombre: | Nombre: |
| Firma: | Firma: | Firma: | Firma: |

(1) Nombre, código y dirección del comprador.
(2) Nombre, código y dirección del proveedor.
(3) Códigos de la cuenta bancaria y de la sucursal del Banco en que se opera el proveedor.
(4) Nombre, dirección y carné de identidad del transportador y chapa del vehículo utilizado por éste o casilla del ferrocarril.
(5) Código, descripción, unidad de medida, cantidad, precio unitario total e importe de cada producto objeto de reclamación.
(6) Importe Total de la Reclamación.
(7) Número del documento que ampara los productos objeto de reclamación
(Factura).
(8) Número consecutivo del modelo.
(9) Nombre y firma del receptor.
(10) Nombre y firma del transportador.
(11) Nombre y firma del proveedor, como aceptación de la reclamación.
(12) Nombre y firma de la persona que contabiliza la reclamación.
(13) Fecha de emisión del modelo.

**Anexo 6. Ficha Técnica.**

| **Ficha Técnica** | | | | |
|---|---|---|---|---|
| Área: (1) | | | | |
| Producto a despiezar: (2) | | | U/M: () | |
| Productos resultantes: () | | | | |
| Grupo: | Familia: | Producto: | U/M | Puntos: |
| | | | | |
| | | | | |
| | | | | |

(1) Área.
(2) Producto a despiezar.
(3) Unidad de medida.
(4) Productos resultantes con su grupo, familia, nombre, unidad de medida y puntos cada uno.

**Anexo 7. Reporte de Elaboración.**

| **Reporte de Elaboración** | |
|---|---|
| Área: (1) | Fecha de emisión: (2) |

| Producto despiezado: (3) | | | U/M: (4) | Día: | Mes: | Año: |
|---|---|---|---|---|---|---|
| | | | | | | |
| Productos resultantes: (5) | | | | | | |
| Grupo: | Familia: | Producto: | | U/M: | Puntos: | Cantidad: |
| | | | | | | |
| | | | | | | |
| | | | | | | |

(1) Área.
(2) Fecha.
(3) Producto despiezado.
(4) Unidad de medida.
(5) De los productos que se obtienen se refleja la cantidad despiezada, grupo, familia, producto, unidad de medida, cantidad, puntos.

**Anexo 8. Solicitud de Entrega.**

| Entidad: (1) | | | Código: | | | **Solicitud de Entrega Modelo SC-2-07** | | | |
|---|---|---|---|---|---|---|---|---|---|
| Almacén: (2) | | | Código: | | | | | | |
| Área: (3) | | | Código: | | | | | | |
| Centro de costo: (4) | | | | Código: | | | Orden de Trabajo: (5) | | |
| Productos: (6) | | | | | | | | | |
| Producto: | | Código: | Descripción: | U/M: | | | Cantidad: () | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| Solicitante: (7) | | | Autoriza: (8) | | | Recibe: (9) | | | No. Del Modelo: (10) |
| Nombre: | | | Nombre: | | | Nombre: | | | |
| Firma: | | | Firma: | | | Firma: | | | |
| Fecha: | | | Fecha: | | | Fecha: | | | |
| Día: | Mes: | Año: | Día: | Mes: | Año: | Día: | Mes: | Año: | |
| | | | | | | | | | |

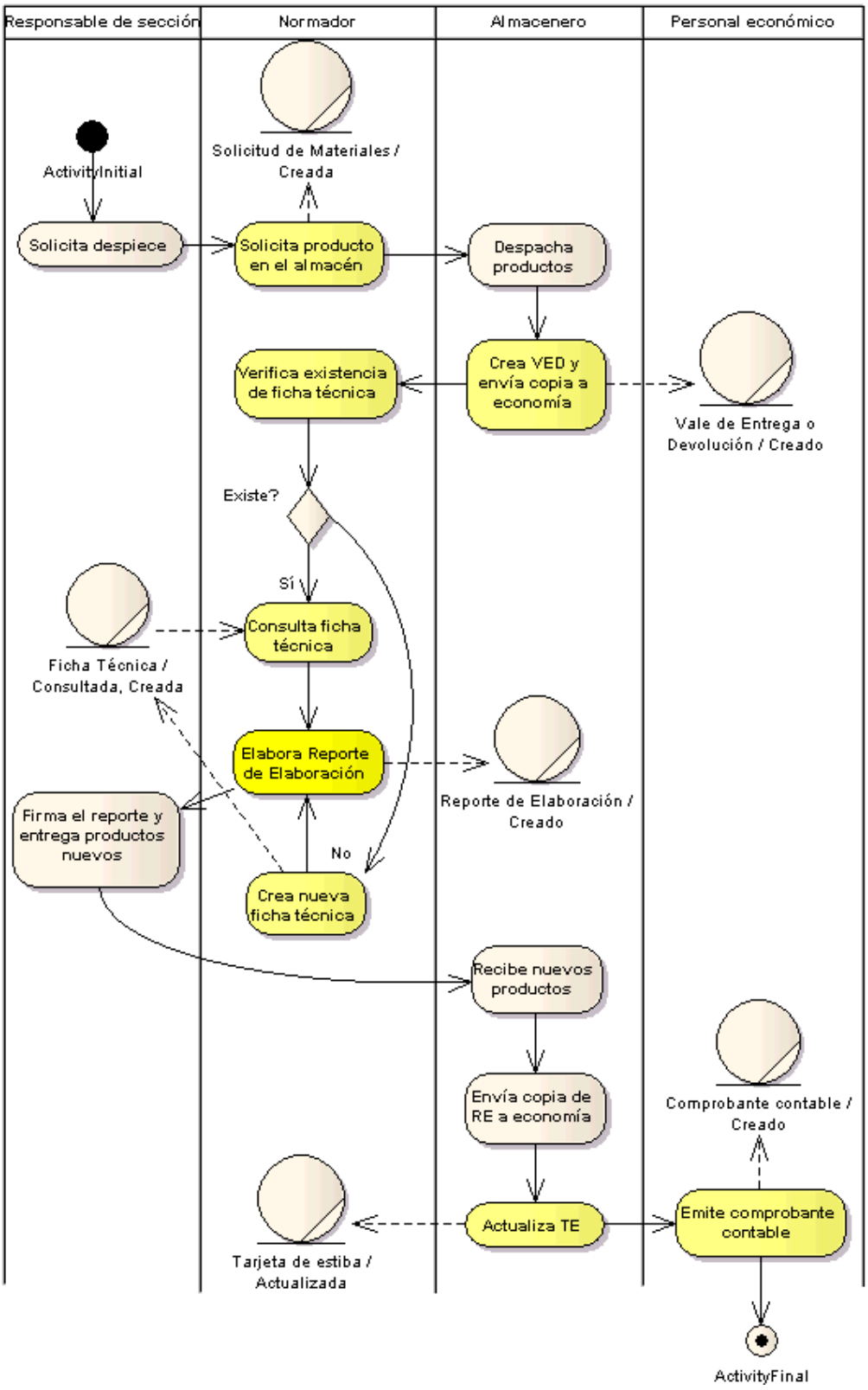
(1) Nombre y código de la entidad.
(2) Nombre y código del almacén al que se efectúa la Solicitud.
(3) Nombre y código del área.
(4) Centro de costo y su código.
(5) Número de la Orden de Trabajo a la que se cargarán los productos.
(6) Código, descripción, unidad de medida y cantidad de cada producto solicitado.
(7) Nombre y firma del solicitante y fecha de la Solicitud.
(8) Nombre y firma del funcionario que autoriza la Solicitud y fecha de la misma.
(9) Nombre y firma del empleado que recibe la Solicitud en el almacén y fecha de la misma.
(10) Número consecutivo del modelo.

**Anexo 9. Tarjeta de Estiba**

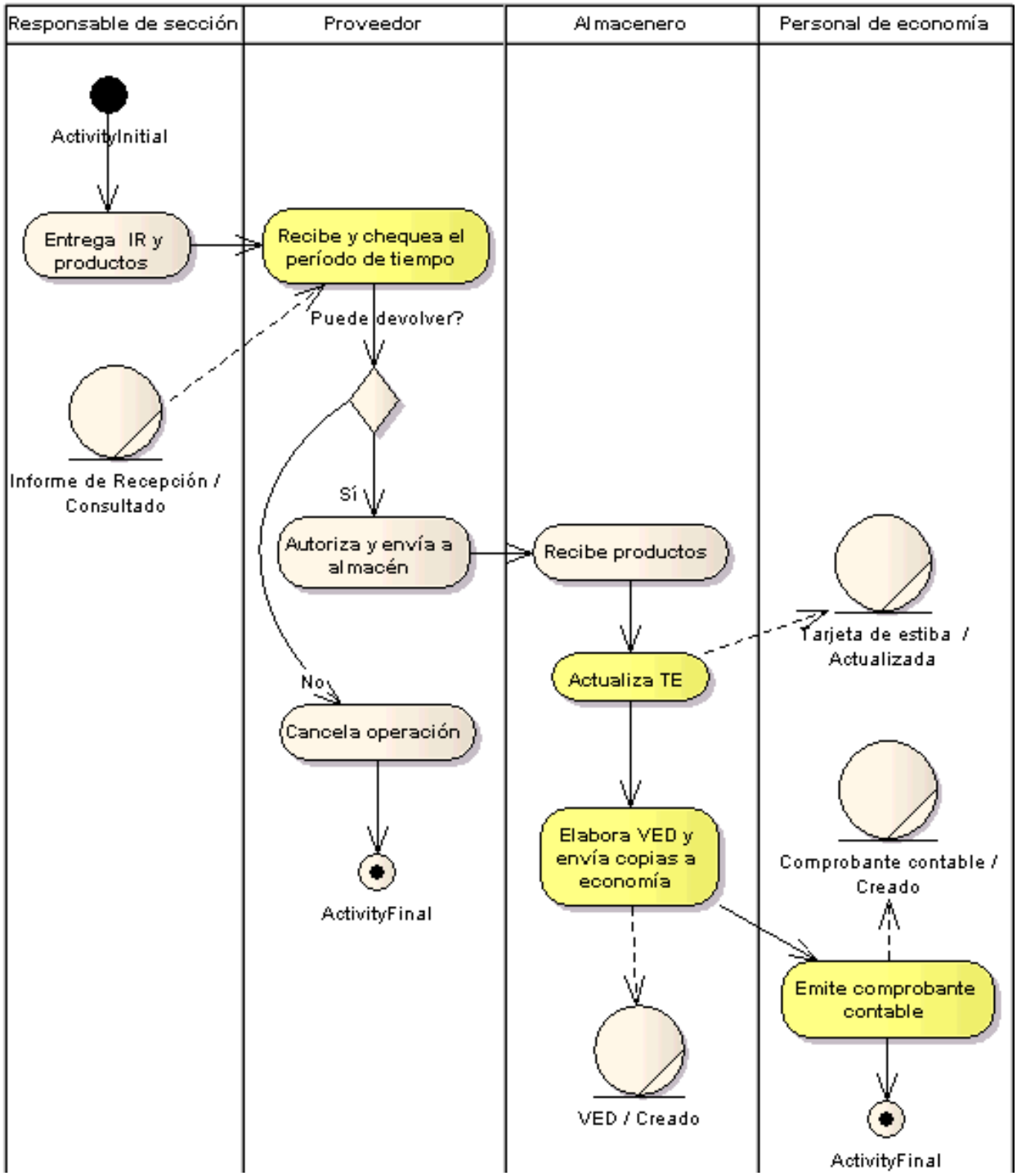
| **Tarjeta De Estiba**<br>**MODELO SC-2-14** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Descripción: (1) | | | | | Código: | | |
| Ubicación: (2) | | | U/M: (3) | | Código de cuenta: (4) | | |
| Sección: | Estante: | Casilla: | | | | | |
| Operaciones: (5) | | | | | | | |
| No.: | Recibidas: | Entregadas: | Existencia: | Día: | Mes: | Año: | Firma: |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

(1) Descripción del producto y código del mismo.
(2) Ubicación: sección, estante y casilla.
(3) Unidad de medida operativa.
(4) Código de la cuenta, subcuenta o análisis en que se contabiliza el producto.
(5) Fecha de cada operación, número del documento que origina el movimiento, unidades recibidas, unidades entregadas, existencia después de cada operación y firma del dependiente del almacén que efectúa la anotación de cada producto.

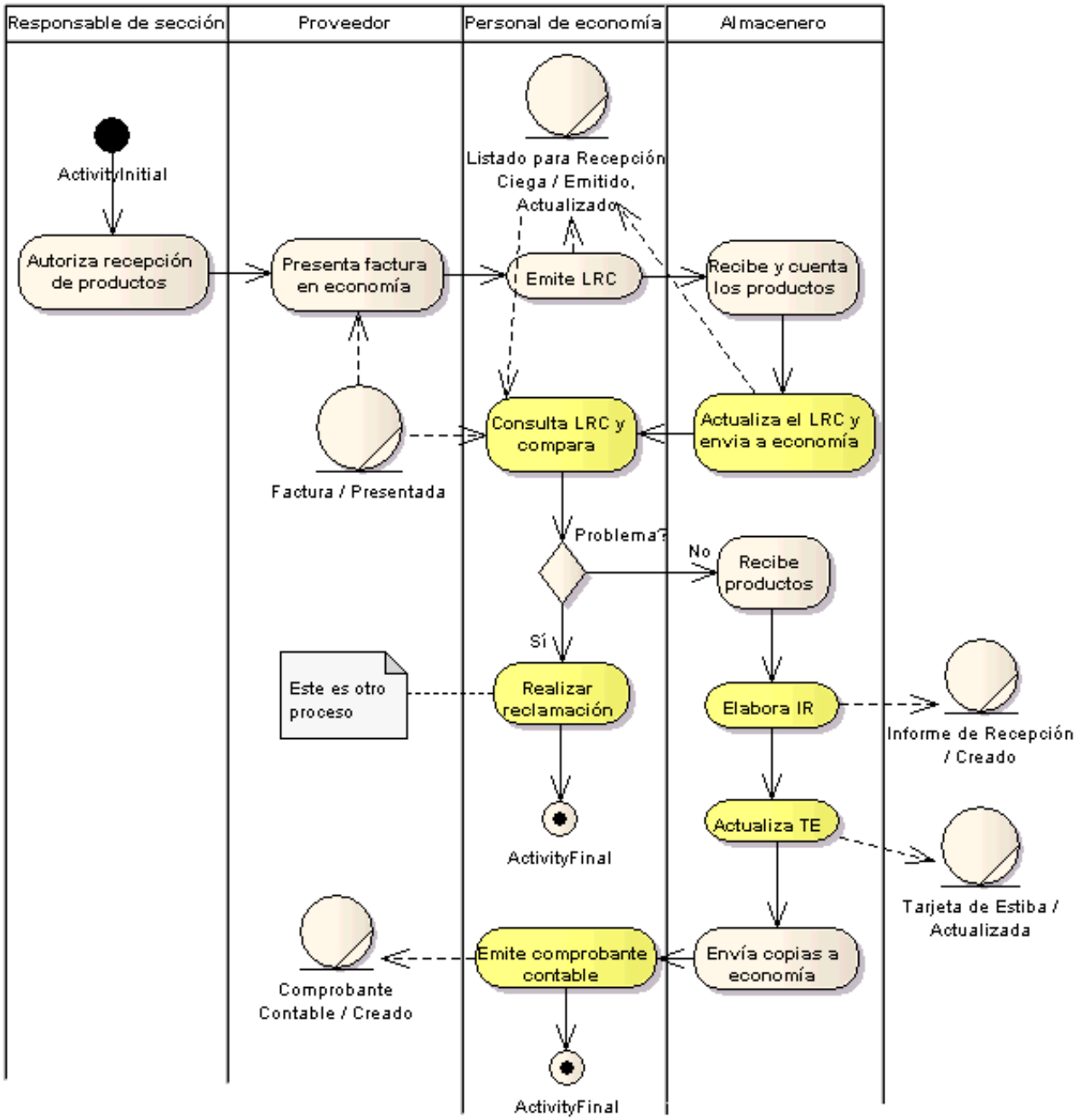
**Anexo 10: Diagrama de Actividad del Caso de Uso de Despiece.**



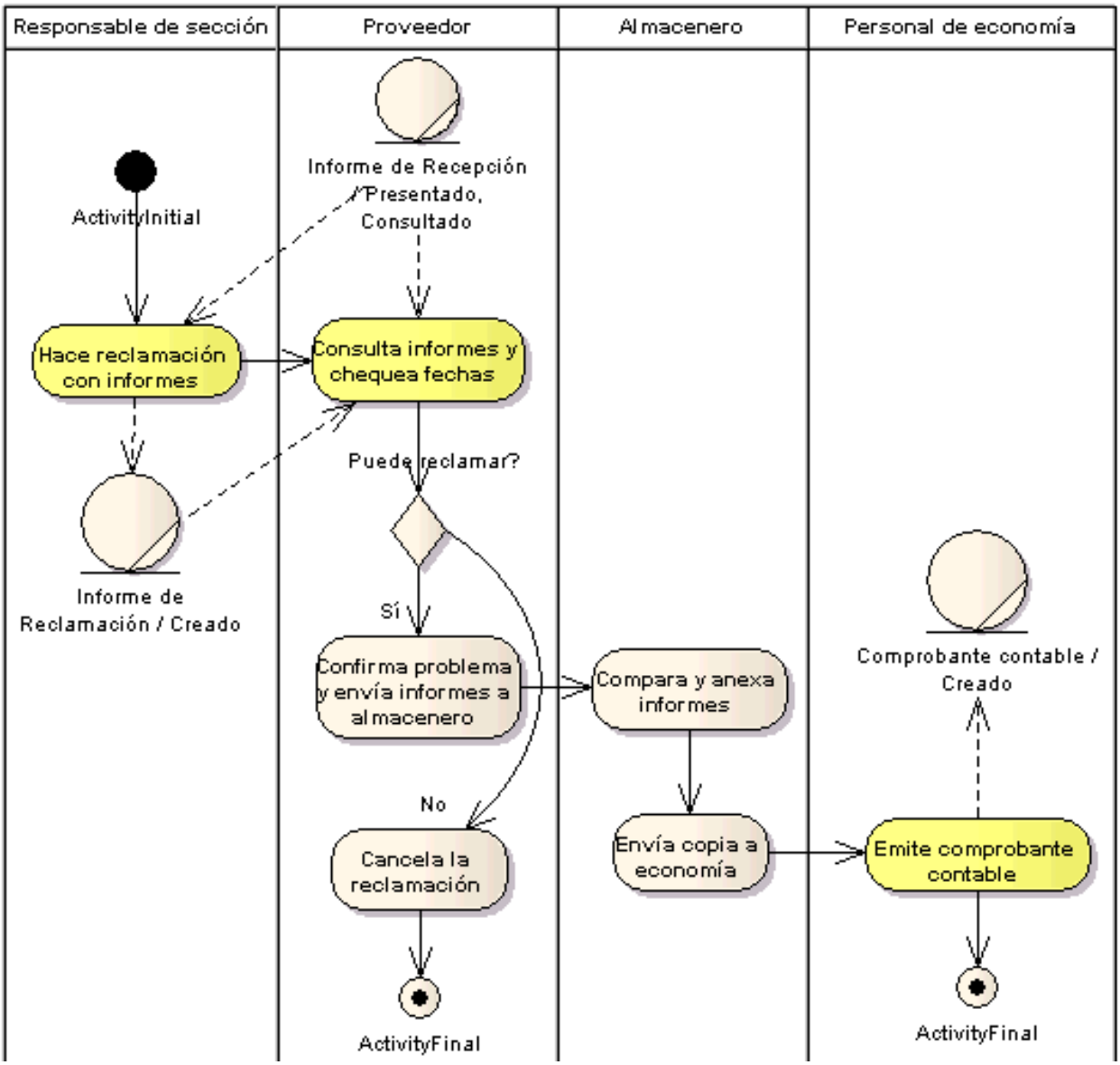
**Anexo 11: Diagrama de Actividades del Caso de Uso Devolver Productos.**



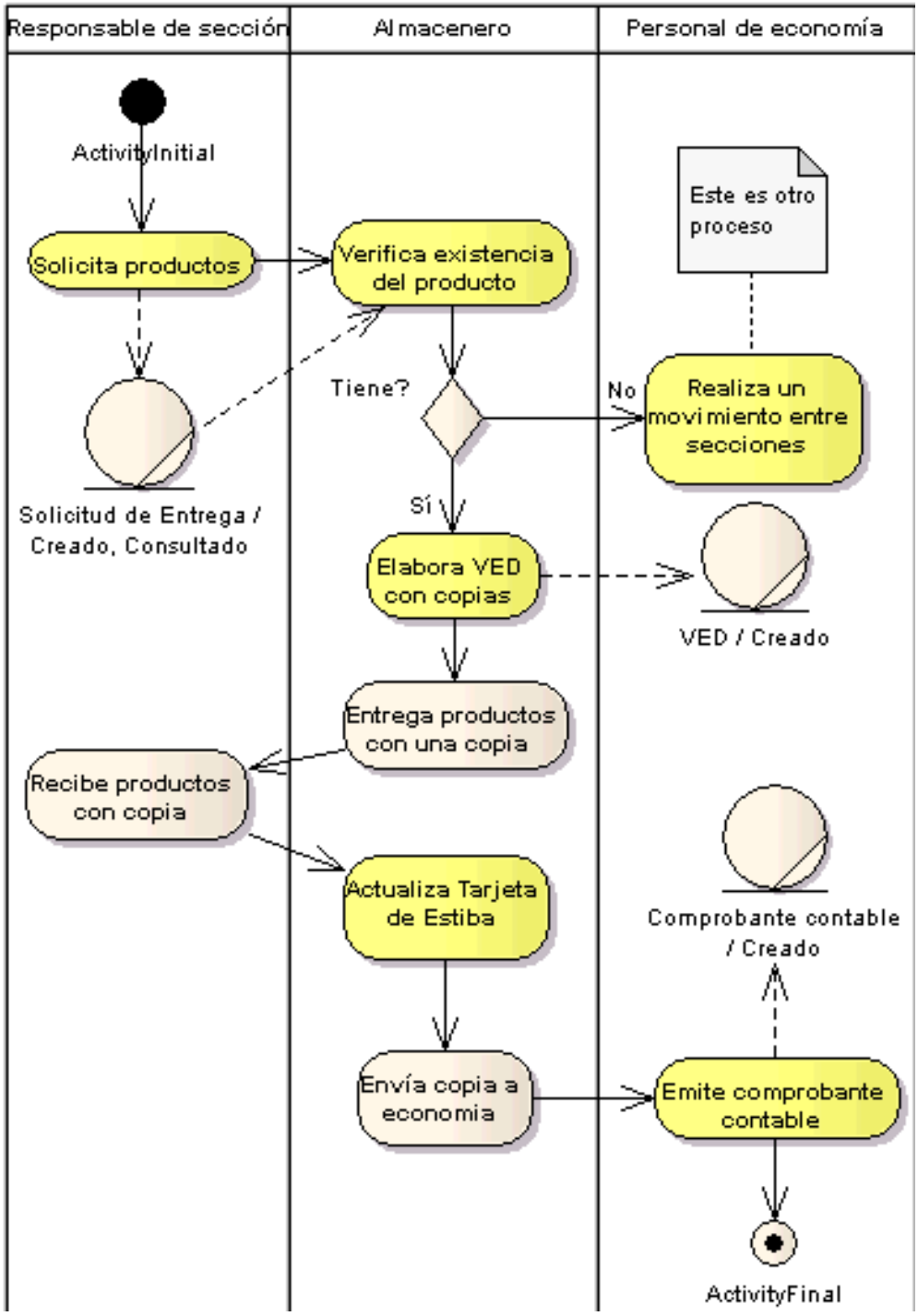
**Anexo 12: Diagrama de Actividades del Caso de Uso Recepcionar Productos.**



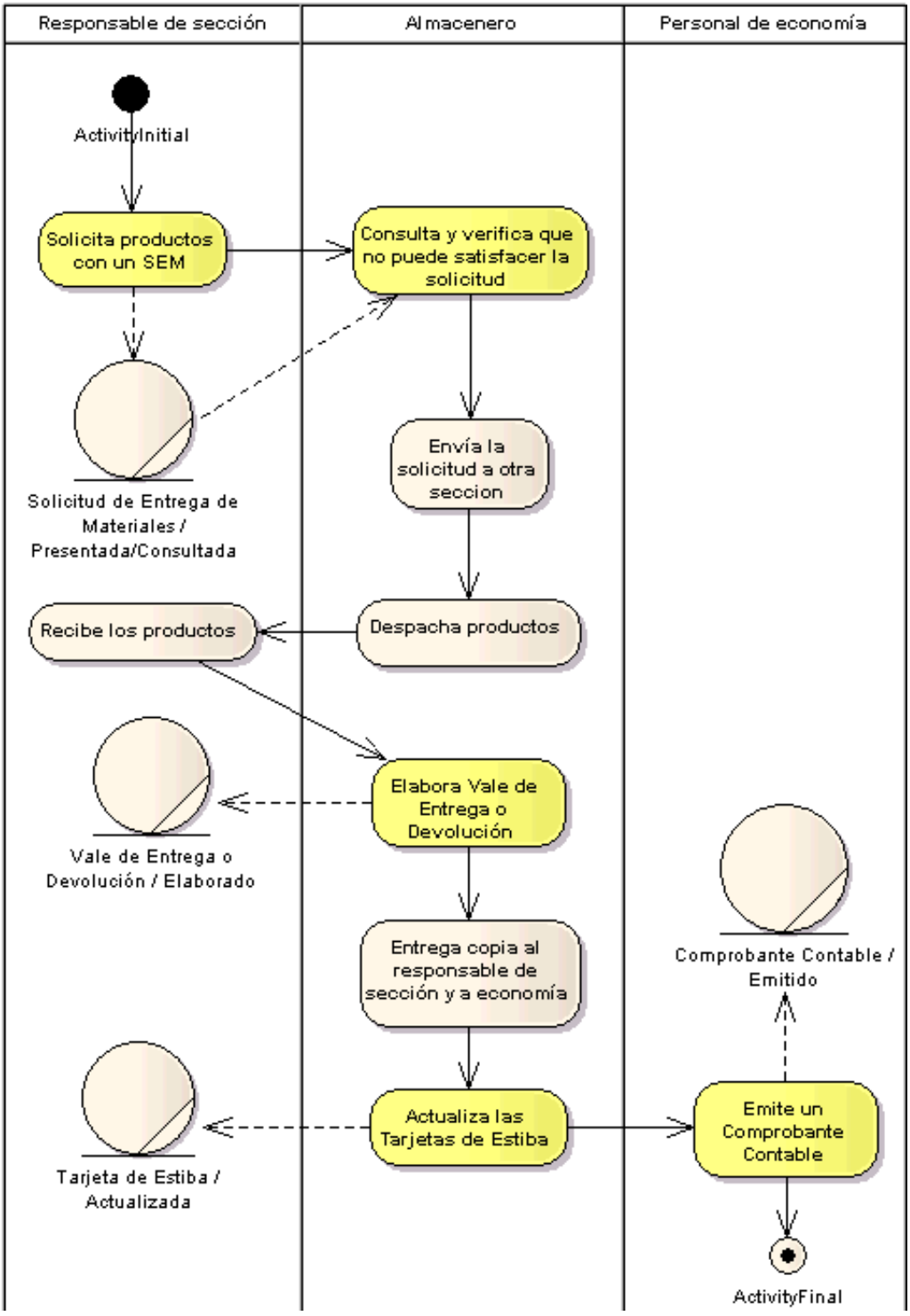
**Anexo 13: Diagrama de Actividades del Caso de Uso Realizar reclamación.**



**Anexo 14: Diagrama de Actividades del Caso de Uso Solicitar Productos.**



**Anexo 15. Diagrama de Actividades del Caso de Uso Mover productos entre secciones.**



GLOSARIO

**Almacén:** Es el lugar o espacio físico en que se depositan las materias primas, el producto semi terminado y/o el producto terminado a la espera de ser transferido al siguiente eslabón de la cadena de suministro. Sirve como centro regulador del flujo de mercancías entre la disponibilidad y la necesidad de fabricantes, comerciantes y consumidores.

**Comprobante Contable:** Es el documento que representa la contrapartida contable de cualquier operación que se realice en el almacén. En él se registra la contabilidad asociada a la operación.

**Despiece:** Operación en la que a partir de un producto se obtienen varios, es decir a partir de dividir un producto obtengo varios sub-productos. Ejemplo: De una pierna de cerdo obtengo carne de primera, carne de segunda, huesos, etc.

**Factura**: Documento que acompaña la venta de productos. En la misma se ingresan todos los datos referentes a la venta y el material que se suministra.

**Ficha Técnica**: Documento único para cada producto que posee la definición del mismo y los ingredientes que lo componen.

**Informe de Recepción (IR):** documento que ampara y formaliza la entrada al almacén de productos suministrados por el proveedor. En él se especifican los datos del proveedor, del almacén que recepciona, de los empleados involucrados en el proceso y la lista de productos suministrados con sus datos específicos.

**Informe de Reclamación (IRecl):** documento confeccionado y enviado al proveedor en caso de que los productos suministrados (o parte de ellos), no satisfagan los requisitos de calidad previamente acordados entre cliente y proveedor, o haya faltante de los mismos.

**Tarjeta Estiba (TE):** control físico de entradas, salidas y existencias que lleva el almacén en el cual se refleja la cantidad de cada producto en el almacén. Se opera manualmente y se ubica en el lugar donde se encuentra almacenado el producto; o en un lugar cercano al mismo, donde sea más factible su cuidado y manipulación. Esta tarjeta tiene impreso igual formato en el anverso y en el reverso, permitiendo su uso por ambas caras.

**Vale de Solicitud de Materiales (VP):** modelo emitido por algún área de la empresa (Ej. taller) solicitando materiales al almacén, contiene datos del área donde se genera, del almacén al cual va dirigido y un listado de productos solicitados.

**Vale de Salida de Materiales (VE):** modelo que ampara las salidas del almacén por solicitudes de materiales, o sea, este es generado por el almacén cuando recibe un Vale de Solicitud de Materiales, debe tener referencia al mismo que lo generó y no debe tener ni más ni menos productos que los que incluía el Vale de Solicitud, además incluirá el resto de los datos del almacén, área al que va dirigida y personal involucrado en el proceso.

**Vale de Entrega o Devolución (VED):** este documento ampara los despachos de productos hechos por el almacén para consumir, o formalizar devoluciones al almacén.

**API (Application Programming Interface - Interfaz de Programación de Aplicaciones):** es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

**Framework:** Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

**Interfaz de Usuario:** Es la parte de una aplicación que se encarga de interactuar con el usuario.