01170
14

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA
DIVISION DE ESTUDIOS DE POSGRADO

EL PROCESO UNIFICADO APLICADO A UN SISTEMA PARA LAS UNIONES DE CREDITO DEL SECTOR SOCIAL

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERIA

(E L E C T R I C A)

P R E S E N T A :

MARICELA RODRIGUEZ MONDRAGON

DIRECTOR DE TESIS:
M.C. MA. GUADALUPE ELENA IBARGUENGOITIA GONZALEZ

CIUDAD UNIVERSITARIA, JUNIO 2001

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

Quiero manifestar mi agradecimiento a todas las personas que hicieron posible la conclusión de este trabajo sin importar la forma en que colaboraron.

El reconocimiento mayor es para mi directora de tesis la M.en C. Guadalupe Ibargüengoitia González, así como a la Dra. Hanna Oktaba quienes han compartido y trasmitido su capacidad académica, su apoyo y colaboración. Gracias.

Así mismo, deseo manifestar mi profundo agradecimiento al M.I. Eugenio López Ortega que hizo posible la elaboración de una parte importante de este tema ya que me permitió participar en el proyecto del Instituto de Ingeniería de la Universidad Nacional Autónoma de México. Gracias.

Mención aparte merece el Instituto de Ingeniería y la División de Estudios de Posgrado de la Universidad Nacional Autónoma de México por su disposición a facilitarme todas las herramientas necesarias para la elaboración de esta tesis.

Me limito a mencionar a aquellos que me enseñaron y formaron académicamente como son mis profesores y compañeros ya que dándose cuenta o no enriquecieron mi estancia en la Universidad.

También cerca en lo personal, aunque fuera de la Universidad, a mi familia quienes acrecentaron lo que siempre habían venido dispensándome, su amor y respeto. Los quiero mucho. Gracias.

Finalmente a Héctor por su gran cariño y comprensión.

En general, gracias a la Universidad Nacional Autónoma de México porque dentro de sus aulas se forjaron los cimientos que me sostendrán durante mi vida como profesionista.

Introducción

Objetivo

El objetivo principal de esta tesis es proporcionar una forma sencilla y rápida de utilizar la Metodología del Proceso Unificado desde la definición de los requerimientos hasta la implementación, ejemplificándolo con un sistema real que se desarrolló en el Instituto de Ingeniería de la Universidad Nacional Autónoma de México dirigido por el Maestro Eugenio López Ortega (Investigador).

Metodología

Para poder entender el Proceso Unificado, se revisó el libro The Unified Software Development Process donde explica el Proceso Unificado .

Posteriormente se hizo una recopilación de toda la información que se tiene de los usuarios del sistema (Sistema Integral de Uniones de Crédito para el Sector Social SIIUCSS) para poder identificar las necesidades y plasmarla en los requerimientos.

Por el tamaño del sistema analizado únicamente se decidió presentar en este trabajo una iteración para ejemplificar el proceso.

Contenido

En el primer capítulo se define qué es el Proceso Unificado. Así mismo define las fases y los procesos de trabajo.

En el Segundo capítulo se definen las razones por las que es difícil entender al cliente, porqué son importantes los requerimientos y a partir de qué se deben definir, se presenta una breve explicación del contexto del sistema que se va a analizar, el papel principal de las uniones de crédito, qué son, cuál es su problemática principal y a partir de ésta la definición de los requerimientos, se construye un glosario de términos, se representan los requerimientos como casos de uso, se identifican los actores principales del sistema tanto los internos como los externos, se crea un prototipo de la interfaz de usuario por caso de uso así como la relación entre caso de uso, interfaz y actor.

En el capítulo tres se lleva a cabo el análisis de los casos de uso identificados en el capítulo dos, primeramente se agrupan los casos de uso en paquetes de acuerdo a su contenido y dependencias, posteriormente se analizan las clases tomando como base el glosario de términos (realizado en el capítulo dos) y éstas se clasifican de acuerdo a la función de cada una; se analiza caso de uso por caso de uso representándolos en diagramas de colaboración para ver cómo las clases tienen relación en cada caso de uso analizado y finalmente se definen los atributos, responsabilidades y asociaciones por cada clase.

En el capítulo cuatro se realiza el diseño a partir del análisis, utilizando el lenguaje de los desarrolladores, es decir, se diseña la arquitectura del sistema, identificando los nodos y la configuración de la red, por ejemplo, características del equipo de cómputo, el lenguaje de programación, el sistema operativo, el manejador de base de datos y la capacidad de las computadoras. Se define la distribución física de los subsistemas a partir de los paquetes (identificados en el análisis) y su relación entre ellos y finalmente se identifican las interfaces de los subsistemas y la navegabilidad del sistema.

En el capítulo cinco se presenta el resultado de implementar el sistema en términos de componentes, es decir, identificar los componentes de tipo archivo (a partir de las clases) y los de tipo ejecutable (a partir de los subsistemas), su distribución física en los nodos; verificar que el sistema se vaya construyendo poco a poco y se realicen pruebas por cada constructor y finalmente implementar las clases como componentes de tipo archivos que pueden contener código.

Finalmente se presentan las conclusiones de este trabajo así como las recomendaciones

Capítulo 1. Proceso Unificado

1.1. Introducción

En la ingeniería de software el objetivo es construir un producto software o mejorar uno existente, proporcionar normas para su desarrollo eficiente de calidad, reducir el riesgo y hacer el proyecto más predecible.

En la industria del software se necesita de un proceso que guíe a todos los participantes como son los desarrolladores, clientes y usuarios y que a su vez sea capaz de evolucionar, pero debe limitar su alcance en determinado tiempo por las tecnologías, herramientas, personas y organización.

El problema mayor de los desarrolladores es coordinar las múltiples actividades de un proyecto de software, por eso necesitan de un proceso que:
1. Proporcione una guía para ordenar las actividades del grupo de trabajo.
2. Dirija las tareas de cada desarrollador y del grupo.
3. Especifique los artefactos que deben desarrollarse.
4. Ofrezca criterios para el control, la medición de los productos y las actividades del proyecto.

Por eso un proceso bien definido y bien gestionado es la diferencia entre proyectos productivos y proyectos que fracasan.

Uno de los problemas principales en el desarrollo de software, es que no se entiende realmente lo que desea el cliente, (porque no se definen correctamente los tiempos de entrega del producto) o porque los analistas no están los suficientemente preparados y no es fácil entender y definir las necesidades del cliente. Así uno de los problemas principales es la definición de los requerimientos ya que éstos van a ser la base para desarrollar e implementar el sistema, por eso es muy importante trabajar en conjunto los usuarios, los analistas y los desarrolladores ya que de esta forma se van a definir apropiadamente y van a cumplir con todas las necesidades del cliente.

Por todos los problemas que aquejan a la ingeniería de software, Ivar Jacobson, James Rumbaugh y Grady Booch investigaron cuáles son los problemas más frecuentes y cómo resolverlos; se dieron cuenta que el problema mayor es utilizar la metodología y herramienta apropiada para llevar a cabo un buen análisis y diseño aunados a la definición de las necesidades del cliente. A partir de esto, definieron su propia metodología tomando en cuenta sus tres metodologías, por eso le dieron el nombre de Proceso Unificado ya que es la unión de ellas.

El Proceso Unificado es un Proceso de Desarrollo de Software que define todas las actividades que se necesitan para convertir los requerimientos del usuario en un producto software. Un proceso de desarrollo de software también define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. El Proceso Unificado sustituye a los métodos de análisis y diseño orientado a objetos que aparecieron a finales de los años ochentas y principios de los noventas, unificando los métodos de Booch, Rumbaugh (OMT), Jacobson y abarcando también ideas de otros métodos. Esta basado en modelos por eso utiliza el lenguaje unificado de modelado

(UML), el cual fue creado en paralelo por los mismos investigadores (Grady Booch, Ivar Jacobson y Jim Rumbaugh) para poder especificar, visualizar y documentar a través de diagramas, por eso UML es una parte esencial del Proceso.

1.2. Qué es el Proceso Unificado

El Procesos Unificado es un proceso de desarrollo de software. Un Proceso de Desarrollo de Software es el conjunto de actividades necesarias para transformar los requerimientos de un usuario en un sistema software, sin embargo, es más que un proceso, puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. Está basado en componentes interconectados a través de interfaces bien definidas y utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los modelos de un sistema software.

El Proceso Unificado se basa en tres ideas claves que lo hacen único: casos de uso, centrado en la arquitectura, e iterativo e incremental.

1.2.1. El Proceso Unificado está dirigido por casos de uso

Un Caso de Uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante, representa los requerimientos funcionales y todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad total del sistema, una especificación funcional contesta a la pregunta ¿qué debe hacer el sistema?, los casos de uso le añaden tres palabras muy importantes a esa pregunta ¿.. para cada usuario?; estas palabras ayudan a pensar en términos de importancia para el usuario y no en términos de funciones.

Los casos de uso se han adoptado para la captura de requerimientos de sistemas software, pero en particular dirigen el proceso de desarrollo en su totalidad, desde la captura de requerimientos, pasando por el análisis, diseño e implementación hasta la prueba, enlazando todos estos flujos de trabajo. Aunque los casos de uso guían el proceso de desarrollo, no se desarrollan aisladamente, se desarrollan a la vez que la arquitectura.

La captura de requerimientos tiene dos objetivos:

1. Encontrar los verdaderos requerimientos.- Aquellos que cuando se implementan añadirán el valor esperado a los usuarios.
2. Representarlos de un modo adecuado para los usuarios, clientes y desarrolladores.- La descripción obtenida de los requerimientos debe ser comprensible por usuarios y clientes.

¿Por qué casos de uso?

1. Porque proporcionan un medio sistemático de capturar los requerimientos funcionales centrándose en el valor agregado para el usuario.
2. Porque dirigen todo el proceso de desarrollo debido a que la mayoría de las actividades como el análisis, diseño y prueba se llevan a cabo partiendo de los casos de uso.
3. Porque implican a los usuarios, clientes y desarrolladores para definir qué debe hacer el sistema para los usuarios, así como a delimitarlo.
4. Porque sus descripciones ayudan a encontrar las clases.
5. Porque ayudan a desarrollar interfaces de usuario.
6. Porque ayudan a los jefes de proyecto a planificar, asignar y controlar tareas que los desarrolladores realizan.
7. Porque ayudan a idear la arquitectura mediante la selección correcta de casos de uso.
8. Porque se utilizan como punto de partida para escribir el manual de usuario.

Captura de casos de uso
La mayoría de los sistemas tienen muchos tipos de usuarios, cada tipo de usuario es representado por un actor y los actores utilizan el sistema para interactuar con los casos de uso y éstos a su vez forman un modelo de casos de uso que muestra un conjunto de casos de uso y actores asociados entre ellos.

No todos los actores representan a personas, pueden ser otros sistemas o hardware externo que interactuará con el sistema; se puede dar el caso de que pueda actuar uno o varios actores, dependiendo de la interacción con el sistema. Estos se comunican con el sistema mediante el envío y recepción de mensajes hacia y desde el sistema según se lleve a cabo los casos de uso. A medida que se define lo que hacen los actores y lo que hacen los casos de uso, hay una separación entre las responsabilidades de los actores y del sistema; esta separación ayuda a delimitar el alcance del sistema.

Los casos de uso están diseñados para cumplir los deseos del usuario cuando utiliza el sistema, por eso, un caso de uso especifica una secuencia de acciones, incluyendo variantes que el sistema puede llevar a cabo y que produce un resultado observable para un actor en concreto.

Para poder identificar los casos de uso, se examina cómo los usuarios necesitan utilizar el sistema para realizar su trabajo y cada uno de esos modos de utilizar el sistema es un caso de uso candidato; éstos se ampliarán, se cambiarán, dividirán en casos de uso más pequeños o se integrarán en casos de uso más completos hasta integrar todos los requerimientos funcionales, de tal forma que los clientes y usuarios lo comprendan y lo utilicen para comunicar sus necesidades de un modo consistente y no redundante y los desarrolladores puedan dividir el trabajo entre ellos y posteriormente utilizarlo como entrada para el análisis, diseño, implementación y prueba del sistema.

Los casos de uso también se utilizan como contenedores de los requerimientos no funcionales, tales como requerimientos de disponibilidad, exactitud, seguridad que son específicos de un casos de uso, por eso, los requerimientos funcionales se representan a través de un caso de uso y los requerimientos no funcionales pueden asociarse a los casos de uso.

Análisis, diseño e implementación para realizar los casos de uso
Durante el análisis y diseño se transforma el modelo de casos de uso mediante la realización de los casos de uso, de tal forma que el sistema ofrezca un rendimiento adecuado y pueda evolucionar en el futuro.

Creación del modelo del análisis a partir de los casos de uso
El modelo del análisis crece a medida que se analizan los casos de uso, primero se identifican todos los casos de uso para una iteración, después se lee la descripción de cada caso uso y se proponen los clasificadores y asociaciones necesarios para llevar a cabo el caso de uso, posteriormente se construye una estructura de clases del análisis y las relaciones entre ellas (esto se hace para todos los casos de uso de una iteración); una vez que ya se saben las responsabilidades que deberían cumplir, se deben identificar cómo interactúan entre ellas para realizar un caso de uso usando los diagramas de colaboración para mostrar como interactúan los objetos secuencialmente o en paralelo, numerando los mensajes que envían unos con otros.

Los responsables de analizar y realizar los casos de uso son responsables de especificar los roles de las clases, deben agrupar todos los roles en un conjunto completo de responsabilidades para la clase y después integrar un conjunto consistente de responsabilidades. También deben verificar que las clases realicen los casos de uso con la calidad adecuada. Si se cambia una clase se debe verificar que la clase siga cumpliendo sus roles en la realización de los casos de uso y si se cambia un rol se debe comunicar el cambio de la clase, por tanto, los roles ayudan a mantener la integridad del análisis.

Creación del modelo del diseño a partir del modelo del análisis
El modelo del diseño se crea tomando el modelo del análisis como entrada principal; funciona como esquema para el modelo de implementación y define las clases, subsistemas e interfaces, sus relaciones entre ellas y las colaboraciones que llevan a cabo los casos de uso.

Los subsistemas agrupan clases
Las clases se agrupan en subsistemas y un subsistema es un agrupamiento de clases o de otros subsistemas. Poseen un conjunto de interfaces. Pueden diseñarse de forma descendente cuando se identifican los subsistemas y las interfaces antes de que se hayan identificado las clases o ascendentemente cuando los subsistemas se basan en las clases que ya se han identificado.

Creación del modelo de implementación a partir del modelo de diseño
El modelo de implementación está formado por componentes que incluyen todo lo necesario para obtener un sistema ejecutable como: componentes ejecutables, de fichero, de tablas, etc. Los componentes son reemplazables, es decir se pueden intercambiar uno por otro, siempre que el nuevo proporcione y requiera las mismas interfaces. Si se implementan en un lenguaje de programación orientado a objetos, la implementación de las clases es directa. Cada clase de diseño corresponde con una clase en la implementación y cada componente fichero puede implementar varias de esas clases, dependiendo de las convenciones del lenguaje de programación.

Prueba de los casos de uso
Durante la prueba se verifica que el sistema implemente correctamente su especificación; se desarrolla un modelo de prueba compuesto por casos de prueba y procedimientos de prueba.

Un caso de prueba es un conjunto de entradas de prueba, condiciones de ejecución, y resultados esperados desarrollados para un objetivo concreto, tal como probar un camino concreto a través de un caso de uso o verificar que cumple un requisito específico. Un procedimiento de prueba es una especificación de cómo llevar a cabo la preparación, ejecución y evaluación de los resultados de un caso de prueba particular.

La forma práctica de probar las funciones de un sistema es la prueba de que el sistema puede utilizarse de manera que tenga sentido para los usuarios, lo que hay que hacer es introducir manualmente los datos necesarios para ejecutar las pruebas.

Las pruebas de los casos de uso pueden llevarse a cabo desde la perspectiva de un actor que considera al sistema como una caja negra, o bien desde una perspectiva de diseño, en la cual el caso de prueba se construye para verificar que las instancias de las clases participantes en la realización del caso de uso hacen lo que deberían hacer. Las pruebas de caja negra pueden identificarse, especificarse y planificarse tan pronto como los requerimientos sean algo estables.

1.2.2. El Proceso Unificado está centrado en la arquitectura

Los casos de uso solos no son suficientes, se necesitan algo más para conseguir un sistema. Eso es la arquitectura ya que muestra una clara perspectiva del sistema completo necesaria para controlar el desarrollo y para describir los elementos más importantes del modelo como los subsistemas, dependencias, interfaces, colaboraciones, nodos y clases activas que describen los cimientos necesarios del sistema para comprenderlo y desarrollarlo.

La arquitectura se describe mediante diferentes vistas (una vista del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y del modelo del despliegue) del sistema en construcción; incluye los aspectos estáticos y dinámicos más significativos del sistema. Por eso la arquitectura es una vista del diseño completo con las características más importantes para que las comprendan todos los involucrados en el sistema.

La arquitectura del software abarca decisiones muy importantes sobre:
1. La organización del sistema.
2. Los elementos estructurales que compondrán el sistema y sus interfaces.
3. La composición de los elementos estructurales y del comportamiento en subsistemas más grandes y
4. El estilo de la arquitectura (los elementos, sus interfaces, sus colaboraciones y su composición).

Sin embargo, la arquitectura está afectada no sólo por la estructura y el comportamiento, sino por el uso, la funcionalidad, el rendimiento, la flexibilidad, la reutilización, la facilidad de comprensión, las restricciones y compromisos económicos, tecnológicos y de estética.

¿Por qué es necesaria la arquitectura?
La arquitectura se necesita para:
. Comprender el sistema,
. Organizar el desarrollo,
. Fomentar la reutilización y
. Hacer evolucionar el sistema.

Para que una organización desarrolle un sistema debe ser comprendido por todos los que vayan a intervenir en él, por eso el primer requisito en una descripción de la arquitectura; se debe capacitar a todos los que intervengan para comprender qué se está haciendo con suficiente detalle y así facilitar su participación, también se debe comprender porque los sistemas:
- Abarcan un comportamiento complejo,
- Operan en entornos complejos,
- Son tecnológicamente complejos,
- A menudo combinan computación distribuida, productos y plataformas comerciales y reutilizan componentes y marcos de trabajo,
- Deben satisfacer demandas individuales y de la organización y
- En algunos casos, son tan grandes que se tiene que dividir el trabajo en varios proyectos añadiendo dificultades a la hora de coordinarlos.

Cuanto mayor sea la organización del proyecto, mayor será la sobrecarga de comunicación entre los desarrolladores para intentar coordinar sus esfuerzos; esta sobrecarga se incrementa cuando el proyecto está disperso geográficamente. Dividiendo el sistema en subsistemas, con las interfaces claramente definidas y con un responsables para cada subsistema se puede reducir la carga de comunicación entre los grupos de trabajo de los diferentes subsistemas, tanto si están en el mismo edificio como en diferentes continentes. Una buena arquitectura es la que define explícitamente estas interfaces, haciendo que sea posible la reducción en la comunicación.

Cualquier sistema debe ser capaz de evolucionar sin problemas, debe permitir implementar nuevas funcionalidades (casos de uso) sin impactar el diseño y la implementación existentes por eso debe ser flexible o tolerante a los cambios.

Casos de Uso y Arquitectura
Los casos de uso y la arquitectura tienen mucha relación ya que un producto tiene una función y una forma, la función corresponde a los casos de uso y la forma a la arquitectura, por eso debe existir relación entre ambos, por un lado los casos de uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los casos de uso ahora y en el futuro. En realidad, tanto la arquitectura como los casos de uso deben evolucionar en paralelo.

Para que se le pueda dar forma(que es la arquitectura) al sistema se deben identificar los casos de uso claves, que pueden ser entre el 5 y 10 por ciento de todos los casos de uso, pero deben ser los que constituyan las funciones fundamentales del sistema.

- Primero se debe crear un esquema de la arquitectura, empezando por la que no está especificada en los casos de uso por ejemplo la plataforma.
- Segundo, se debe trabajar con los casos de uso que representan las funciones claves del sistema en desarrollo y
- Tercero, conforme se vaya avanzando se deben ir refinando los casos de uso y mejorando la arquitectura.

Estos pasos se deben repetir hasta que la arquitectura sea estable.

Si los casos de uso son correctos, los usuarios pueden emplearlos para llevar a cabo sus objetivos. Para conseguirlo, se debe construir una arquitectura que permita implementar los casos de uso de una forma económica ahora y en el futuro.

Existen diferentes tipos de requerimientos y productos que influyen en la arquitectura, aparte de los casos de uso.
Por ejemplo:
- Qué productos software del sistema se quieren desarrollar,
- Qué productos de la capa intermedia se quieren utilizar,
- Qué sistemas heredados se quieren utilizar en el sistema,
- Qué estándares y políticas corporativas se deben adaptar.
- Requerimientos no funcionales (no específicos de los casos de uso) y
- Las necesidades de cómo distribuir el sistema.

Los pasos hacia una arquitectura
Para definir la arquitectura, primero se empieza determinando un diseño de alto nivel a modo de una arquitectura en capas, después se forma la arquitectura en un par de construcciones dentro de la primera iteración. En la primera construcción se trabajan con las partes generales en cuanto al dominio y que no son específicas del sistema que se piensa desarrollar, (es decir, se selecciona el software del sistema (capa de software del sistema), capa de en medio, los sistemas heredados, los estándares y las políticas de uso). Se decide qué nodos contenderá el modelo de desarrollo y cómo deben interactuar entre ellos. También se decide cómo manejar los requerimientos generales no funcionales, así como su disponibilidad.

Con la primera pasada es suficiente para tener una visión general del funcionamiento de la aplicación. En la segunda construcción se trabaja con los aspectos de la arquitectura específicos de la aplicación (capa específica de la aplicación), se escoge un conjunto de casos de uso relevantes en cuanto a la arquitectura, se capturan los requerimientos, se analizan, se diseñan, se implementan y se prueban. El resultado serán nuevos subsistemas implementados como componentes del desarrollo que soportan los casos de uso seleccionados.

¿Qué es primero, la arquitectura o los casos de uso?.
La mejor forma de resolver este problema es a través de una iteración, primero se construye una arquitectura tentativa básica a partir de una buena comprensión del área del dominio, pero sin considerar los casos de uso detallados, posteriormente se escoge un par de casos de uso y se amplia la arquitectura adaptándola para que soporte esos casos de uso, finalmente se escogen algunos casos de uso más y se construye una arquitectura mejor y así sucesivamente. Entonces los casos de uso ayudan a mejorar gradualmente la arquitectura según se va iterando para completar el sistema. Por eso una buena arquitectura permite obtener los casos de uso correctos.

Descripción de la Arquitectura
La descripción de la arquitectura permite controlar el desarrollo del sistema desde la perspectiva técnica. Se centra en los subsistemas, las clases, en los componentes, en los nodos y en las colaboraciones entre ellos a través de las interfaces.

Esta debe mantenerse actualizada a lo largo de la vida del sistema para reflejar cambios y las adiciones que son relevantes para la arquitectura. Estos cambios deben incluir:
- La identificación de nuevas clases abstractas,
- La adición de una nueva funcionalidad a los subsistemas existentes,
- La actualización a nuevas versiones de los componentes reutilizables y
- La reordenación de la estructura de procesos.

La descripción de la arquitectura incluye casos de uso, subsistemas, interfaces, algunas clases, componentes, nodos, colaboraciones, requerimientos significativos para ella pero que no están descritos por medio de los casos de uso (requerimientos no funcionales relativos a la seguridad y restricciones a la distribución y concurrencia). También debería incluir una breve descripción de la plataforma, los sistemas heredados y el software comercial que utilizará.

Pero no incluye información que sólo sea necesaria para validar o verificar la arquitectura, por tanto, no tiene casos o procedimientos de prueba porque no desempeña ningún papel en la descripción de la arquitectura.

Como ya se mencionó anteriormente la descripción de la arquitectura tiene cinco secciones una para cada modelo.
- La vista de la arquitectura del modelo de casos de uso presenta los actores y casos de uso más importantes.
- La vista de la arquitectura del análisis con las clases, sus responsabilidades y relaciones.
- La vista de la arquitectura del diseño presenta los subsistemas, interfaces y clases más importantes pero fundamentalmente las clases activas.
- La vista de la arquitectura del modelo del despliegue define la estructura física del sistema por medio de nodos interconectados.
- La vista de la arquitectura del modelo de implementación, es una correspondencia directa de los modelos del diseño y de despliegue.

1.2.3. El Procesos Unificado es Iterativo e Incremental.

Este paso proporciona la estrategia para desarrollar un sistema software en pasos pequeños, ya que permite:
- Planificar un poco,
- Especificar, diseñar e implementar un poco e
- Integrar, probar y ejecutar un poco en cada iteración.

Es práctico dividir el trabajo en partes más pequeñas o miniproyectos donde cada miniproyecto es una iteración que debe estar controlada, es decir, debe seleccionarse y ejecutarse de una forma planificada. La selección de lo que se implementará en una iteración se basa en dos factores: Primero, se deben agrupar los casos de uso que ayuden a mejorar la utilidad del producto desarrollado y en segundo lugar, tratar los riesgos más importantes.

Al ser miniproyectos, comienzan con los casos de uso y continúan con el análisis, diseño, implementación y prueba y finaliza con una preparación para la entrega. En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes y verifican que los componentes satisfagan los casos de uso. Si una iteración cumple con sus objetivos, el desarrollo continúa con la siguiente iteración, hasta que se hayan realizado todas las iteraciones que se habían planificado se tiene un producto desarrollado para distribuir a los clientes y usuarios, pero cuando no cumple, los desarrolladores deben revisar sus decisiones previas y probar con un nuevo enfoque.

Por eso las tres ideas clave tienen igual importancia, la arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo

de cada iteración. La eliminación de una de las tres reduciría drásticamente el valor del proceso unificado.

¿Porqué un desarrollo iterativo e incremental?

- Para obtener un mejor software,
- Para tomar las riendas de los riesgos críticos y significativos desde el principio,
- Para poner en marcha una arquitectura que guíe el desarrollo de software,
- Para proporcionar un marco de trabajo que gestione de mejor forma los cambios en los requerimientos,
- Para construir el sistema a lo largo del tiempo y
- Para proporcionar un proceso de desarrollo que permita al personal trabajar de manera más eficaz.

Obtención de una Arquitectura robusta

La consecución de una arquitectura robusta es en si misma el resultado de las iteraciones. En la fase de inicio se encuentra una arquitectura esencial que satisface los requerimientos clave, evita los riesgos críticos y resuelve los problemas de desarrollo principales. En la fase de elaboración se establece la línea base de la arquitectura que guiará el resto del desarrollo, pero en esta fase aún se puede cambiar.

Cada iteración debe progresar mediante una serie de construcciones hasta alcanzar el resultado esperado. El tener un sistema con funcionamiento parcial en una fase inicial permite que los usuarios y otros interesados hagan sugerencias sobre él o señalen requerimientos que se pueden escapar. El plan aún no es inamovible, de forma que se pueden incorporar revisiones con más facilidad. Por tanto el ciclo de vida iterativo permite a los clientes observar fácilmente la necesidad de requerimientos adicionales o modificados dentro del ciclo de desarrollo, esto va a permitir trabajar más fácilmente con ellos.

También se pueden resolver los problemas y los aspectos no cubiertos por las primeras construcciones e incluir cambios para corregirlos casi a la vez, pero si la iteración sólo tuvo éxito parcial, puede ampliarse para cubrir aspectos no resueltos o para realizar las correcciones necesarias para la siguiente iteración. En un caso extremo, cuando la evaluación sea totalmente negativa, pueden cancelar el proyecto entero.

La aproximación iterativa está dirigida por los riesgos

La ingeniería de software necesita de un proceso que "cree una aproximación al proceso de desarrollo de software dirigida por los riesgos en lugar de una proceso dirigido por los documentos o dirigido por el código".

Un riesgo es la probabilidad de que un proyecto experimente sucesos no deseables, como retrasos en las fechas, excesos de costo o la cancelación directa. Se identifican, se priorizan y se llevan a cabo las iteraciones sobre la base de los riesgos y su orden de importancia. Esto se lleva a cabo cuando se evalúan tecnologías nuevas y cuando se trabaja para cumplir con las necesidades de los usuarios. También cuando se va a establecer una arquitectura robusta. Otros riesgos importantes son velocidad, capacidad, precisión, disponibilidad, integridad de las interfaces de usuarios, adaptabilidad y portabilidad.

Este es el motivo por el cual se deben llevar a cabo iteraciones que examinen los riesgos, incluso en las fases de inicio y elaboración y hasta la codificación y la prueba.

Como ya se dijo, una iteración es un miniproyecto que obtiene una versión interna del proyecto a través de un flujo de trabajo integrado por trabajadores que utilizan y producen artefactos. En el proceso unificado se distingue entre flujo de trabajos fundamentales (requerimientos, análisis, diseño, implementación y prueba) que ayudan a describir el flujo de trabajo de la iteración; y el flujo de trabajo de iteración que pasa por los cinco flujos de trabajo fundamentales; se crea superponiendo uno encima de otro, planificando y analizando el que se termina.

El objetivo de una iteración es determinar si aparecen nuevos requerimientos o han cambiado los existentes afectando a las iteraciones siguientes.

Planificación de las iteraciones
Se deben ordenar las iteraciones para obtener un camino directo en el cual las primeras iteraciones proporcionen la base para las siguientes, ya que las primeras dan como resultado un mayor conocimiento de los requerimientos, los problemas, los riesgos y el dominio de la solución; mientras que las siguientes dan incrementos aditivos que finalmente conforman el producto para el cliente.

Las iteraciones sobre el ciclo de vida
Una iteración no es una entidad completamente independiente, es una etapa dentro de un proyecto y se ve fuertemente condicionada por ello. Se dice que es un miniproyecto porque se parece al ciclo de vida ya que el ciclo de vida iterativo produce resultados tangibles en forma de versiones preliminares y cada una de ellas aporta un incremento y demuestra la reducción de los riesgos con los que se relaciona. Además estas versiones se pueden presentar a los clientes y usuarios para que puedan retroalimentar la validación del trabajo, en resumen se puede decir que un ciclo de vida se compone de secuencia de iteraciones .

Las iteraciones se organizan dentro de las cuatro fases, cada una de ellas con necesidades concretas de personal, financiación y planificación y con sus propios criterios de comienzo y fin. Pero al comienzo de cada fase, la dirección puede decidir cómo llevarla a cabo, los resultados que deben entregarse y los riesgos que deben reducirse.

Cada una de las cuatro fases termina con un ciclo principal:
- Inicio: objetivos del ciclo de vida.
- Elaboración: arquitectura del ciclo de vida.
- Construcción: funcionalidad operativa inicial.
- Transición: versión del producto.

El objetivo de cada ciclo principal es garantizar que los diferentes modelos de flujo de trabajo evolucionen de manera equilibrada durante el ciclo de vida del producto.

Los objetivos de la fase de inicio son: establecer lo que debería hacer el producto; la reducción de los peores riesgos y la preparación del análisis del negocio; en pocas palabras, establecer los objetivos del ciclo de vida del proyecto.

Los objetivos de la fase de elaboración son: obtener la línea base de la arquitectura, capturar la mayoría de los requerimientos y reducir los siguientes peores riesgos, es decir, establecer la arquitectura del ciclo de vida. También en esta fase se debe estimar los costos y las fechas de planificar la fase de construcción con algún detalle.

Los objetivos de la fase de construcción son: desarrollar el sistema entero y la garantía de que el producto pueda comenzar su transición a los clientes, es decir, que tenga una funcionalidad operativa inicial.

El objetivo de la fase de transición son: garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios. Durante esta fase se enseña a los usuarios a utilizar el software.

Como se puede observar cada iteración produce resultados, por tanto, al final de cada iteración se tiene un nuevo incremento en el modelo de: casos de uso, análisis, diseño, despliegue, implementación y pruebas. Este incremento se integrará con el resultado de la iteración anterior, obteniendo una nueva versión del modelo.

También un proyecto de desarrollo de software puede dividirse en dos etapas: las fases de inicio y elaboración y las fases de construcción y de transición. Durante las fases de inicio y elaboración, la mayoría del esfuerzo se dedica a la captura de requerimientos y a un análisis y diseño preliminares. Durante la construcción se pasa al diseño detallado, la implementación y la prueba.

1.3. La Vida del Proceso Unificado

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema, cada ciclo produce una nueva versión y cada versión es un producto preparado para su entrega. El producto terminado debe ajustarse a todas las necesidades de la gente que trabajará con él, por eso, debe incluir todos los elementos que permitan al usuario utilizar y modificar el sistema como son:

1. Un modelo de casos de uso y su relación con los usuarios.
2. Un modelo de análisis con los casos de uso refinados con mas detalle.
3. Un modelo de diseño que defina la estructura estática del sistema en la forma de subsistemas, clases e interfaces y los casos de uso reflejados como colaboraciones entre los subsistemas, clases e interfaces.
4. Un modelo de implementación que incluya componentes que representen el código fuente y las clases con los componentes.
5. Un modelo de despliegue que defina los nodos físicos y la correspondencia de los componentes con los nodos.
6. Un modelo de prueba que describa los casos de prueba que verifiquen los casos de uso.
7. Y una representación de la arquitectura.

También debe contar con un modelo del domino del negocio que describa el contexto del negocio en el que se incluye al sistema.

1.3.1. Fases dentro de un ciclo

Cada ciclo se desarrolla a lo largo del tiempo, éste se divide en cuatro fases, inicio, elaboración, construcción y transición a través de una secuencia del modelo. Dentro de cada fase se puede descomponer el trabajo en iteraciones con sus incrementos.

Cada fase termina con un ciclo y cada ciclo se determina por la disponibilidad de un conjunto de artefactos. Los ciclos permiten controlar el progreso del trabajo cuando pasa por las cuatro fases, al final, se obtiene un conjunto de datos que son útiles en la estimación del tiempo y los recursos humanos para futuros productos.

La figura 1.1 muestra en la columna izquierda los flujos de trabajo, requerimientos, análisis, diseño, implementación y prueba. Las curvas son una aproximación de hasta dónde se llevan a cabo los flujos de trabajo en cada fase. Donde cada fase se divide normalmente en iteraciones o mini-proyectos. Una iteración típica pasa por los cinco flujos de trabajo.

Durante la fase de inicio, se desarrolla una descripción del producto final a partir de un texto y se presenta el análisis del negocio para el producto. En esta fase se debe contar:

- Con un modelo de casos de uso que contenga los casos de uso más críticos,
- Una esbozo de la arquitectura provisional que muestre los subsistemas más importantes,
- Se identifican y priorizan los riesgos más importantes,
- Se planifica en detalle la fase de elaboración y
- Se estima el proyecto de manera aproximada.

Durante la fase de elaboración se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La arquitectura se expresa en forma de vistas de

todos los modelos del sistema (vistas del modelo de casos de uso, del análisis, del diseño, de implementación y de prueba) los cuales representan al sistema entero.

Al final de la fase de elaboración, ya se pueden planificar las actividades y estimar los recursos necesarios para terminar el proyecto.

Durante la fase de construcción se crea el producto. En esta fase la línea base de la arquitectura crece hasta convertirse en el sistema completo para ser entregado a los usuarios. Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión. Sin embargo, puede que no esté completamente libre de defectos, ya que estos se descubrirán y solucionarán durante la fase de transición.

La fase de transición cubre el periodo durante el cual, el producto se convierte en versión beta, donde un número reducido de usuarios con experiencia lo prueba e informan defectos y deficiencias. Se corrigen los problemas e incorporan algunas de las mejoras sugeridas.

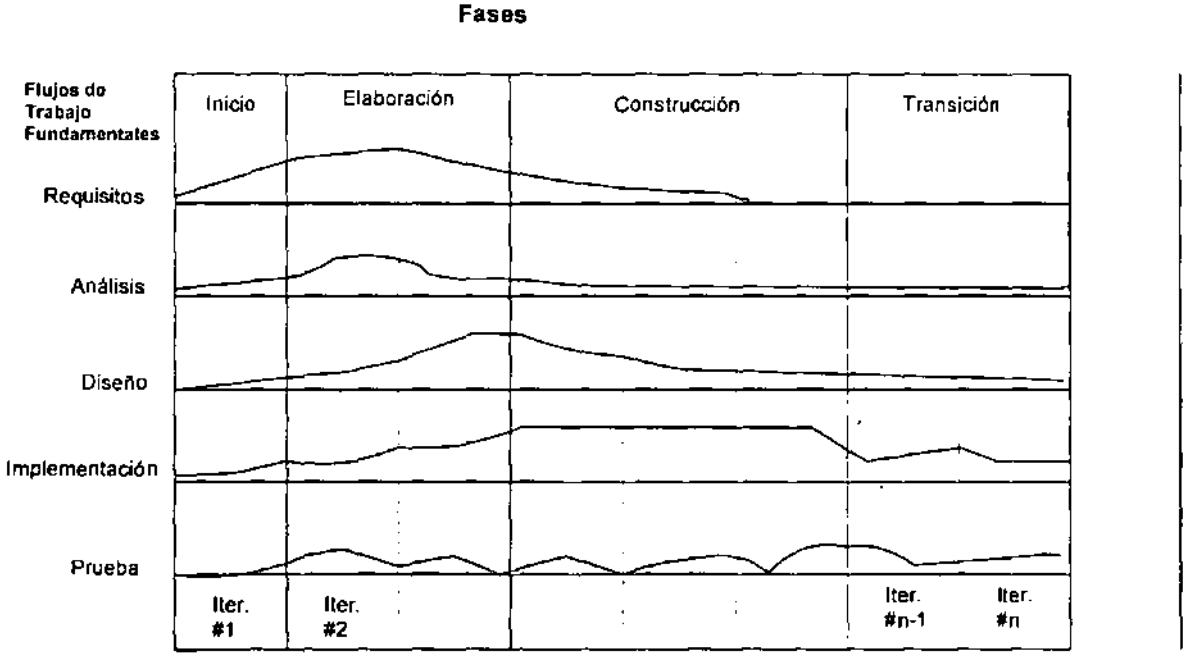


Tabla 1.1 El Proceso Unificado

Flujos de Trabajo Fundamentales

Mediante la descripción separada de los flujos de trabajo (requerimientos, análisis, diseño, implementación y pruebas) uno detrás de otro, se piensa que el proceso de desarrollo de software pasa por una secuencia de flujos de trabajo sólo una vez. Pero no es así se recorren los cinco flujos de trabajo secuencialmente, una vez por cada iteración, no una vez para el proyecto completo, por ejemplo si se tienen cinco iteraciones sobre cuatro fases, se llevan a cabo cinco flujos de trabajo, es decir, se llevan a cabo los flujos de trabajo en cada iteración mientras sea necesario para cada iteración en concreto.

Flujo de Trabajo de los Requerimientos

El objetivo principal del flujo de trabajo de los requerimientos es guiar el desarrollo hacia el sistema correcto mediante una descripción de los requerimientos del sistema, para que se pueda llegar a un acuerdo entre el cliente y los desarrolladores sobre qué puede y qué no hacer el sistema .

Una forma de conseguirlo es usar el lenguaje del cliente, ya que éste va a permitir que el cliente sea capaz de leer y comprender la captura de los requerimientos. Para identificar los requerimientos del sistema se basa en los casos de uso, éstos capturan los requerimientos funcionales y no funcionales que son específicos de cada caso de uso.

Como cada usuario desea que el sistema haga algo para él, es decir, que lleve a cabo ciertos casos de uso, entonces un caso de uso es un modo de utilizar el sistema, en consecuencia si los analistas pueden describir todos los casos de uso que necesita el usuario, entonces se sabe que debe hacer el sistema.

Para capturar los casos de uso que realmente se quieren para el sistema, se requiere conocer en profundidad las necesidades del usuario y del cliente, por eso se tiene que comprender el contexto del sistema, entrevistar a los usuarios, discutir propuestas, etc.

Pero también debido a que los requerimientos cambian constantemente, se necesita una forma de actualizarlos y es a través de las iteraciones, ya que cada iteración va a reflejar algún cambio en el conjunto de requerimientos, y el número de cambios disminuirá a medida que se adentre en la fase de construcción y que se estabilicen los requerimientos.

Flujo de Trabajo del Análisis

El modelo del análisis es un modelo de objetos conceptual que analiza los requerimientos mediante su refinamiento y estructuración. Por eso incluye, paquetes del análisis, paquetes de servicio y sus dependencias y contenidos. Los paquetes del análisis pueden aislar los cambios en un proceso del negocio, el comportamiento de un actor o en un conjunto de casos de uso relacionados. Los paquetes de servicio aíslan los cambios en determinados servicios ofrecidos por el sistema.

Las clases del análisis, sus responsabilidades, atributos, relaciones y requerimientos especiales, aíslan los cambios al comportamiento y a la información que representan. Un cambio en la interfaz de usuario o en una interfaz de comunicación normalmente se ubica en una o más clases de interfaz; un cambio en la información duradera y a menudo persistente, se ubica en una o más clases de entidad; un cambio en el control, coordinación, secuencia, transacciones, se ubica en una o más clases de control.

La realización de casos de uso del análisis que describen cómo se refinan los casos de uso en términos de colaboraciones. La vista de la arquitectura aislará los cambios de la arquitectura. El modelo del análisis se considera la entrada fundamental para las actividades del diseño, por eso se debe conservar en todo lo posible la estructura definida en este modelo.

Flujo de Trabajo del Diseño

El resultado principal del diseño es el modelo de diseño que se esfuerza en conservar la estructura del sistema impuesta por el modelo del análisis y que sirve como esquema para la implementación.

Este modelo incluye los subsistemas de diseño y subsistemas de servicio y sus dependencias, interfaces y contenidos. Los subsistemas del diseño de las dos capas superiores (la capa específica y la capa general de la aplicación) se obtienen a partir de los paquetes del análisis. Algunas de las dependencias entre subsistemas del diseño se obtienen a partir de las dependencias entre paquetes del análisis y algunas interfaces se obtienen a partir de las clases del análisis.

A partir de las clases del análisis se obtienen las clases del diseño incluyendo las clases activas, sus operaciones, atributos y requerimientos de implementación.

La realización de casos de uso del diseño que describen cómo se diseñan los casos de uso en términos de colaboraciones dentro del modelo del diseño.

De la vista arquitectónica del modelo de diseño se obtiene el modelo del despliegue que describe todas la configuraciones de red sobre las cuáles debería implantarse el sistema, incluye: nodos, sus características y conexiones y una correspondencia de las clases activas sobre los nodos.

Flujo de trabajo de la Implementación
Los modelos de diseño y del despliegue se consideran la entrada principal para las actividades de implementación y prueba.

Los subsistemas de diseño y de servicio se implementan mediante subsistemas de implementación que contienen los verdaderos componentes: ficheros de código fuente, scripts, binarios, ejecutables y similares. Estos subsistemas de implementación poseerán un mapeo uno a uno hacia los subsistemas de diseño

Las clases de diseño se implementarán mediante componentes de fichero que contendrán el código fuente. Es frecuente implementar varias clases del diseño dentro de un mismo componente de fichero, aunque esto va a depender del lenguaje de programación que se esté utilizando. Por otro lado, las clases activas del diseño que representen procesos pesados se utilizarán como entrada cuando se identifiquen los componentes ejecutables.

La realización de casos de uso del diseño se utilizarán al planificar y llevar a cabo el esfuerzo de implementación en pasos pequeños y manejables que den como resultado construcciones que implementarán un conjunto de realizaciones de casos de uso o partes de ellas.

Flujo de Trabajo de la Prueba
El resultado principal del modelo de prueba es describir cómo ha sido probado el sistema. Incluye casos de prueba, que especifican qué probar en el sistema, procedimientos de prueba que especifican cómo realizar los casos de prueba y componentes de prueba que automatizan los procedimientos de prueba.

1.4. Un proceso Integrado

El Proceso Unificado está basado en componentes. Utiliza el nuevo estándar de modelado visual, el Lenguaje Unificado de Modelado (UML) y se sostiene sobre tres ideas básicas, casos de uso, arquitectura y desarrollo iterativo e incremental. Integra ciclos, fases, flujos de trabajo, gestión del riesgo, control de calidad, gestión de proyecto y control de la configuración.

El resultado de un proyecto software es un producto que toma forma durante su desarrollo porque intervienen muchas personas durante todo el ciclo de vida, por tanto, el proceso que guía este desarrollo debe funcionar correctamente para las personas que lo utilizan, por eso debe estar muy bien organizado para que no les afecte.

Como se ya comentó, las personas son las que realizan las actividades del desarrollo de software, por eso es necesario un proceso uniforme que esté soportado por herramientas y un Lenguaje Unificado de Modelado para hacer que las personas sean más eficaces. Esto permitirá construir un software de mejor calidad porque va a permitir especificar los requerimientos que mejor se ajusten a las necesidades de los usuarios y va a elegir una arquitectura que permita construir los sistemas de forma económica y puntual.

Construir un sistema es un proceso de construcción de modelos para describir todas las perspectivas del sistema. Un modelo es una abstracción semánticamente cerrada del sistema, es una vista en el sentido de que un usuario de un modelo no necesita más información para

interpretarlo. Este debe describir las iteraciones entre el sistema y los que le rodean, es decir, debe incluir elementos que describan partes relevantes de su entorno, como los actores. Por ejemplo el modelo de casos de uso comprende a los casos de uso y los actores, el modelo de diseño describe los subsistemas y las clases del sistema y cómo interactúan para llevar a cabo los casos de uso.

Un sistema contiene todas las relaciones y restricciones entre elementos incluidos en diferentes modelos. Por tanto, un sistema no es sólo la colección de sus modelos sino que también contiene las relaciones entre ellos. Por ejemplo, cada caso de uso en el modelo de casos de uso tiene una relación con una colaboración en el modelo del análisis.

1.4.1. El proceso dirige los proyectos

Un proceso de desarrollo de software es una definición del conjunto de todas las actividades que se necesitan para convertir los requerimientos de un usuario en un conjunto de artefactos que conforman un producto software.

Un proceso es una definición de un conjunto de actividades, no de su ejecución. Cubre todos los ciclos de desarrollo y los posteriores.

Un proceso se describe en términos de flujos de trabajo, donde un flujo de trabajo es un conjunto de actividades. Primero se identifican los distintos tipos de trabajadores que participan en el proceso, después, los artefactos que se necesitan crear durante el proceso para cada tipo de trabajador. Una vez que se han identificado se puede describir cómo fluye el proceso a través de los diferentes trabajadores y cómo ellos crean, producen y utilizan los artefactos de los demás.

A partir de este punto se pueden identificar fácilmente las actividades que estos trabajadores deben realizar y si algún trabajador en concreto necesita participar más de una vez en el flujo de trabajo. En otras palabras, se describe el proceso entero en partes llamadas flujos de trabajo donde participan los trabajadores y artefactos que se generan en el flujo de trabajo.

Por consiguiente, un proceso de desarrollo de software del mundo real debe ser adaptable y configurable para cumplir con las necesidades reales de un proyecto y/o organización concreta. El proceso Unificado se diseñó para poder ser adaptado, es un proceso genérico, es decir, un marco de trabajo de proceso.

El Proceso Unificado puede especializarse para cumplir diferentes necesidades de aplicación o de organización. Al mismo tiempo, es deseable que el proceso sea, al menos, completamente consistente dentro de una organización.

1.4.2. Las herramientas son esenciales en el proceso

El proceso se ve influido fuertemente por las herramientas. Las herramientas son buenas para automatizar procesos repetitivos, mantener las cosas estructuradas, gestionar grandes cantidades de información y guiar un desarrollo concreto.

Las herramientas se desarrollan para automatizar actividades de manera completa o parcial, para incrementar la productividad y la calidad y para reducir el tiempo de desarrollo. A medida que se introduce soporte a través de herramientas, se obtiene un proceso más formal. Se pueden incluir nuevas actividades que sería poco práctico realizar sin herramientas. Podemos trabajar de una manera más precisa durante el ciclo de vida entero: se puede utilizar un lenguaje de modelado formal como UML para asegurar que cada modelo es consistente internamente y en relación con otros modelos. Se puede utilizar un modelo y a partir de él generar partes de otro modelo.

Por un lado, el proceso dirige el desarrollo de las herramientas, y las herramientas dirigen el desarrollo del proceso, en otras palabras, el proceso debe aprender de las herramientas y las herramientas deben soportar un proceso bien pensado. El desarrollo con éxito de la automatización

de un proceso no puede hacerse sin el desarrollo paralelo de un marco de trabajo de proceso en el cual vayan a funcionar las herramientas.

Capítulo 2. Captura y Modelado de Requerimientos como Casos de Uso

2.1. Introducción

Generalmente las personas que se dedican a desarrollar software para alguien o para ellos mismos dicen que "los usuarios saben que es lo que quieren, pero no saben expresarlo de una forma precisa"; esto se debe a que la mayoría de los sistemas cuentan con diferentes tipos de usuarios, pero ellos realmente no conocen la operación, ni cómo pueden hacer más eficiente el sistema.

Cuando se encuentran con este tipo de situaciones solicitan la ayuda de un analista para que él sea capaz de interpretar las necesidades de cada usuario, pero cuando ese analista no tiene los conocimientos suficientes para definir correctamente los requerimientos, éste los registra en documentos muy grandes lo cual no le va a permitir que sean correctos, consistentes y completos por el volumen de información y por lo tanto no va a lograr que esa información se pueda transformar en el diseño e implementación de las especificaciones. Por eso es importante que en la especificación de los requerimientos participen los usuarios, intermediarios y desarrolladores ya que todos pueden aportar muy buenas ideas que van a ayudar a la especificación que va a permitir que se comprenda realmente las necesidades de cada usuario. Un punto importante es que no se pierda la misión para la cual se va a construir el sistema.

Por eso, el mayor esfuerzo de los requerimientos es desarrollar el modelo del sistema a través de los Casos de uso, ya que éstos ofrecen un camino sistemático para capturar los requerimientos para cada usuario individual o para cada sistema externo, con lo que los analistas se ven forzados a pensar como los usuarios del sistema.

Para realizar el modelado de los requerimientos con casos de uso se producen tres productos:
- Diagramas de casos de uso,
- Prototipo de interfaz de usuario y
- Estructurar los casos de uso.

En este capítulo se identifican las necesidades del sistema, se definen los requerimientos y éstos se detallan mediante casos de uso y su interacción con los usuarios (actores); A su vez se realiza un esbozo de las interfaces de usuario que podría tener el sistema.

2.2. Descripción de la Arquitectura

El propósito principal del flujo de trabajo es dirigir el desarrollo hacia un sistema que cumpla con todas las necesidades del usuario, por eso es importante describir la especificación de los requerimientos y las condiciones o capacidades con las que debe estar conformado (que puede

hacer y que no) entre los usuarios y los desarrolladores, posteriormente, se puede empezar a encontrar algunos casos de uso (actores y casos de uso), después diseñar las interfaces de usuario para esos casos de uso con el propósito de revisar que se necesita para realizar nuevos casos de uso para agregarlos. Se puede dar el caso de que se rompa la secuencia, cuando esto sucede, una actividad se puede retomar muchas veces y cada una de éstas puede acarrear la ejecución de una parte de la actividad.

Para preparar la primera versión del modelo de casos de uso primero se buscan los actores y las actividades de los casos de uso; posteriormente se identifican los casos de uso prioritarios y en paralelo se sugiere la interfaz de usuario por cada actor basado en los casos de uso. Con esto se reestructura el modelo de casos de uso para comprenderlos lo mejor posible.

El resultado de esta primera iteración consiste de una primera versión del modelo de casos de uso y algunos prototipos de la interfaz de usuario asociados.

Actividades
1. Visión general de la organización (2.2.1.)
2. Definición de los requerimientos (2.2.2.)
3. Modelo de casos de uso (2.2.3.)
4. Detallar casos de uso (2.2.4.)
5. Prototipo de la interfaz de usuario (2.2.5.)
6. Estructurar el modelo de casos de uso (2.2.6.)

2.2.1. Visión general de la organización

Es muy importante que se lleven a cabo reuniones entre analistas, desarrolladores, clientes y usuarios para comprender las necesidades del cliente y usuario por parte de los analistas y desarrolladores, estos a su vez deben conocer el funcionamiento y la organización de la empresa para la que van a desarrollar el producto.

Ejemplo
Por eso en este trabajo a continuación se da una breve explicación del papel principal de una unión de crédito en el sector social, identificación de las necesidades del sistema, que son las Uniones de crédito y conformación de dos uniones de crédito que se tomaron como base para identificar su problemática y así definir los requerimientos con los que va a estar conformado el sistema.

Papel principal de las Uniones de Crédito
El papel principal de una Unión de Crédito para el sector social es otorgar créditos y canalizar recursos provenientes de otras instituciones a grandes y pequeños productores agrícolas y a empresas dedicadas principalmente a la comercialización de productos o insumos relacionados directamente con la producción agrícola.

Una Unión de Crédito está formada por un conjunto de socios los cuales poseen una cierta cantidad de acciones que los hacen partícipes del capital total de dicha unión. Ese capital se utiliza para otorgar préstamos que por ley se dan únicamente a socios de la unión en función del porcentaje de capital aportado.

La administración de la unión está a cargo del Gerente general, quien es apoyado por personas encargadas de realizar las operaciones administrativas; éstos son supervisados por el Consejo de administración que decide, cuáles de las solicitudes de crédito son viables y, en consecuencia, cuáles de los créditos solicitados son aprobados.

Los recursos con los que dispone una unión para otorgar préstamos son: los propios, los provenientes del FIRA (Fondos Instituidos Relacionados con la Agricultura), de PRONASOL y de

algunos organismos internacionales y gubernamentales como BANRURAL, además de una pequeña participación de la banca privada.

El otorgamiento de un préstamo generalmente se lleva acabo a través de proyectos que surgen a iniciativa del socio; se estudia y formaliza a través de la estructura administrativa de la unión. Una vez que el proyecto ha sido formulado se pasa al comité de crédito para su aprobación. Existen créditos menores llamados quirografarios, los cuales tienen un monto muy pequeño y no requieren de proyecto ni de aprobación de crédito, únicamente requieren de la autorización del gerente de la unión.

La necesidad de contar con un sistema de cómputo para realizar con mayor facilidad y eficacia las tareas administrativas, surge por las dificultades que se tienen al poseer una gran cantidad de información y por lo que la velocidad de respuesta es lenta al otorgar el servicio a los socios en ciertos periodos –a inicio o término de los periodos agrícolas- cuando acuden con mayor frecuencia a la unión a solicitar créditos y a realizar pagos de los mismos. También se ha observado que en la mayoría de las uniones de este tipo, no existe un registro de las ganancias o las pérdidas que pudiese haber al finalizar el año, y en muchas ocasiones no se tiene la seguridad de la cantidad exacta que hay que pagar a las fuentes de financiamiento.

Además, se ha planteado la necesidad de acercar más los servicios de la unión a los socios mediante la apertura de sucursales para comunicarse de una manera más eficiente, rápida e inmediata con la oficina matriz; por lo que un sistema de cómputo puede ayudar.

Otro de los planteamientos, ha sido la necesidad de individualizar los créditos de los socios morales, ya que la mayoría de los socios morales de la unión son asociaciones formadas por pequeños productores. Por eso se requiere tener un mayor control sobre el destinatario final, ya que en muchas ocasiones un socio moral solicita y recibe créditos de socios físicos de las asociaciones. Esta situación puede perjudicar a toda la asociación y, con ello, a los demás miembros de la misma ya que si un socio suspende sus pagos, la asociación a la que pertenece va a pasar a la cartera vencida y se le castiga con la imposibilidad de solicitar y recibir otro crédito hasta que haya pagado toda la deuda. Por esta razón se pretende individualizar a todos los miembros de las asociaciones, de manera que sólo ellos sean los responsables directos de la operación de sus créditos.

Identificación de las necesidades del sistema

De acuerdo a la importancia de las Uniones de Crédito en el Sector Financiero Mexicano, La Asociación Mexicana de Uniones de Crédito del Sector Social, A.C., (AMUCSS), se vio en la necesidad de solicitar al Programa Universitario de Alimentos (PUAL) de la Universidad Nacional Autónoma de México (UNAM) y este a su vez al Instituto de Ingeniería la elaboración de un sistema de cómputo que permitiera automatizar las funciones operativas y administrativas de las Uniones de Crédito asociadas para realizar eficientemente sus tareas diarias. Las Uniones de Crédito juegan un papel muy importante en la economía del país y su razón de ser es ayudar al sector rural a través del otorgamiento de créditos, para eso se eligieron dos uniones (una de Oaxaca y la otra de Jalisco) de la muestra total de Uniones de Crédito, las cuales tienen diferente giro económico; la de Oaxaca se dedica principalmente a la producción y venta de café y la de Jalisco a la de maíz.

Para definir los requerimientos, primeramente se conoció el funcionamiento de AMUCSS: Qué es, cuáles son sus objetivos principales, sus características, cómo esta formada, qué tipos de créditos maneja, qué tipos de acciones tienen, sus fuentes de recursos económicos y los servicios principales que presta. Posteriormente se estudió el funcionamiento de dos uniones de crédito a través de un diagnóstico integral elaborado por ellos que ayudó a comprender su estructura organizativa, los sistemas administrativos y operativos, la situación laboral, la normatividad, las opiniones que tienen los socios sobre la administración de la unión e indicando las actividades que realiza cada empleado de una manera general. Una vez leído, se decidió visitar a las dos uniones

de crédito para aclarar las dudas que se tuvieron, pero principalmente para identificar la problemática que existe y así obtener los procesos administrativos de las funciones que realiza cada empleado de la unión.

Qué son las Uniones de Crédito y por quién están Reguladas

Unión de Crédito del Sector Social

A nivel jurídico son organismos auxiliares de crédito, bajo la modalidad de sociedades anónimas de capital variable. Dentro del sistema financiero mexicano actúan como intermediarios entre la banca y sus solicitantes de financiamiento.

A diferencia del resto de las Uniones de Crédito en México, las del Sector Social nacieron de las luchas de los pequeños productores por apropiarse de sus procesos productivos. Por lo mismo sus motivaciones y funcionamientos se encuentran emparentados con el movimiento cooperativista internacional.

Las uniones de crédito están regidas por la Comisión Nacional Bancaria y por la Ley General de Organismos y Actividades Auxiliares de Crédito.

El objetivo de la Comisión Nacional Bancaria y de la Ley General de Organismos y Actividades Auxiliares de Crédito es: Regular el servicio de Banca y Crédito, la organización y funcionamiento de las instituciones de crédito, las actividades y operaciones que las mismas podrán realizar, su sano y equilibrado desarrollo, la protección de los intereses del público y los términos en que el estado ejercerá la rectoría financiera del Sistema Bancario Mexicano.

Los artículos que regula La Ley General de Organismos y Actividades Auxiliares de Crédito a las Uniones de Crédito son. Art. 39,40,41,43 y 44.

Asociación Mexicana de Uniones de Crédito del Sector Social, A.C. (AMUCSS)

La Asociación Mexicana de Uniones de Crédito del Sector Social, A.C. (AMUCSS) es una organización social conformada como red plural que promueve la creación de organismos financieros locales de ahorro y crédito para el desarrollo de las comunidades rurales regionales y también busca que las uniones de crédito que pertenecen a ella sea una garantía de calidad y de confianza.

Sus objetivos principales:

1. *Desarrollar nuevos créditos que requieran las unidades de producción agropecuarias.*
2. *Aumentar las oportunidades económicas con la movilización del ahorro e*
3. *Implantar una nueva cultura financiera en la población rural.*

Su principal característica es la diversidad económica, social, cultural y geográfica de las uniones de crédito que van desde las regiones indígenas con economías de subsistencia combinadas con agricultura de mercado, hasta regiones con agricultura comercial.

Las uniones de crédito asociadas a ella son resultado de un movimiento popular de autoayuda impuesto por productores agrícolas de escasos recursos, sus principales socios son: ejidatarios, comuneros, pequeños propietarios, colonos y sus diversa figuras asociativas. Sus unidades de producción varían de 5 a 20 hectáreas, sus actividades económicas son; la producción de granos (maíz, frijol, trigo, soya y sorgo), producción y comercialización de café, frutos tropicales, ganadería, pequeño comercio, artesanías, servicios y pequeña industria rural.

32 uniones de crédito campesinas están asociadas a AMUCSS con una cobertura territorial de 20 estados y 255 municipios, su integración social está conformada por:

- *Socios Morales 1,329*

- *Socios Físicos 15,756*
- *Cartera Vigente $ 191'931,000*
- *Cartera Vencida $ 180'735,000*
- *Liquidados $ 561'480,000*
- *Capitales Sociales $ 72'014,337*
- *Tipos de Crédito*
  - *Avío.- destinado a los cultivos de trigo, sorgo, maíz, frijol, cítricos, café y ganado bovino.*
  - *Refaccionario.- destinado a la adquisición de maquinaria y equipo.*

Sus Fuentes de Recursos Económicos
Es un organismo social autónomo. Para desarrollar sus actividades obtienen recursos de cuotas de socios, se determinan anualmente y cubren gastos administrativos de la asociación; donativos y aportaciones públicas y privadas nacionales y extranjeras; venta de servicios a los socios, a otras uniones de crédito, otros organismos campesinos, instituciones educativas y financieras.

Servicios principales que presta:
- *Comunicación financiera.- Intercambio de información y consulta de banco de datos nacionales e internacionales.*
- *Asesoría especializada y capacitación.- En aspectos financieros, contables, administrativos, normativos, fuentes de fondeo, organizativos, etc.*
- *Diagnóstico integral con enfoque de calidad total.- Aplicación de un taller de calidad total para encaminar procesos de mejora continua en operación técnica, organizativa y administrativa.*
- *Apoyo a la formación de uniones de créditos y desarrollo de esquemas financieros de base.- Apoya a organizaciones campesinas en la formación de uniones de crédito, impulsa la creación de cajas de ahorro y préstamos.*
- *Informática.- Se está desarrollando un sistema de información específico para uniones de crédito.*

Conformación de las Uniones de Jalisco y Oaxaca

Unión de Crédito Regional Amequense, S.A. de C.V. (UCRA)
La Unión de Crédito Regional Amequense inició operaciones a partir del 1°. De octubre de 1994, esta ubicada en el estado de Jalisco, está distribuida en 18 municipios (el 15.3% del estado), abarcando 81 localidades y 513 socios (31 personas físicas y 464 personas morales), coexisten productores de rendimientos mayores y pequeños productores de rendimientos inferiores, sobre todo en el cultivo de maíz, sin embargo, su mayor presencia se da en Ameca, San Martín Hidalgo y Tlajomulco.

UCRA se caracteriza por contar con una infraestructura de servicios comerciales y financieros muy desarrollados por ser altamente demandante de insumos, por contar con carreteras pavimentadas y caminos de acceso permanentes y por poseer condiciones agro ecológicas adecuadas para este cultivo –aunque también el sistema de producción de caña de azúcar es predominante-, lo que hace que sea un mercado muy atractivo para la acción de multitud de agentes financieros tanto formales como informales, disputando el mercado de la distribución de insumos –fertilizantes, semillas, insecticidas, etc.- y el acopio y comercialización de maíz y otros granos.

La unión de crédito se soporta en la actividad agrícola y de servicios orientados a este mismo giro (comercialización de insumos agropecuarios).
Del 100% de los socios, el 2% se dedica a agro servicios, el 1% a servicios, el 3% mixtos, comercio 6% y agrícolas el 94.8%.

Normatividad
No existen los reglamentos y manuales para la operación del crédito, cada año realizan el Plan de Operaciones que da origen a la elaboración del presupuesto de ingresos y egresos.

Capital Social $ 2'600,000.
Créditos que otorgan: Quirografarios, Capital de trabajo, Avíos agrícolas, comerciales y refaccionarios.

Unión de Crédito estatal de Productores de Café del Estado de Oaxaca, S.A. de C.V. (UCEPCO)
Está integrada por 39 organizaciones que agrupan 23,000 pequeños productores cafetaleros distribuidos en 29 municipios; estos municipios albergan a 41,459 unidades de producción con actividades agropecuarias, la base de la organización son productores de café minifundistas que representan 51% de las unidades de producción y el 74% dedicadas al cultivo de aromáticos; todos los productores tienen acceso a programas de apoyo gubernamentales, como Procampo, así como estímulos a la agricultura, cuentan con un organismo económico dedicado a la creación de empresas maquiladoras en el medio rural. Cuenta con seis empleados, está conformada por tres organismos estatales, CEPCO (Comercializadora Estatal de Productores de Café del Estado de Oaxaca), CAEO (Comercializadora Agropecuaria del Estado de Oaxaca), UCEPCO (Unión de Crédito Estatal de Productores de Café del Estado de Oaxaca) y OR's (Organismos Regionales) son de tipo organizativo y económico y todos de tipo estructural y operativo, los cuales brindan una serie de apoyos principalmente de gestoría y de representación.

Su objetivo principal es mejorar las condiciones de sus miembros, por lo que sus actividades no se reducen al campo económico, sino que actúa igualmente en los campos social y político, además cuenta con instrumentos financieros como son las cajas solidarias y el manejo de inversiones en moneda nacional y dólares.

La unión cuenta con socios morales y físicos, ha concentrado sus actividades crediticias en el acopio y comercialización de café (80%). Una vez terminado el proceso, CAEO se encarga de liquidar los créditos de cada OR's con el producto y venta del café. Los OR's son los únicos socios de la unión, los créditos individuales deben salir a través de los OR's, distribuyéndose posteriormente a los usuarios finales.

2.2.2. Definición de los Requerimientos

Como sabemos cada sistema es único. Existen variaciones en los tipos de sistemas, en los clientes, en el tipo de información que maneja y en el tipo de tecnología. Hay diferentes formas de definir los requerimientos, puede ser porque ya existe un sistema pero no coincide con las necesidades del cliente, o se puede trabajar con un sistema que existe pero únicamente necesita modificaciones o también se puede utilizar el modelo de otro sistema tomándolo como patrón para el desarrollo de uno nuevo.

También existen situaciones en las cuales el cliente no sabe cómo deberá funcionar el sistema pero tiene nociones muy vagas al respecto; cuando esto sucede el analista debe hacer todo tipo de combinaciones para que sea el punto de partida para la definición, éste debe ser capaz de definir los requerimientos de acuerdo a esas nociones o puntos de vista, ya que cada uno posee diferentes tipos de riesgo pero el analista debe lograr definirlo para reducir esos riesgos.

Se deben considerar los siguientes puntos para disminuir los riesgos:
1. La lista de requerimientos posibles,
2. Comprender el contexto del sistema,
3. Los requerimientos funcionales y
4. Los requerimientos no funcionales.

Lista de requerimientos posibles

Durante la vida del sistema es importante que participen los clientes, usuarios y analistas ya que en conjunto se van a obtener muy buenas ideas las cuales pueden ser candidatas para implementar los requerimientos y estos a su vez como casos de uso; cada idea propuesta debe tener un pequeño nombre con una breve explicación de lo que se trata y debe tener un valor o rango de acuerdo a su importancia.

Se recomienda incluir:

- Status.- Propuesta aprobada, incorporada o validada.
- Estimación de los costos por implementación.- En términos de tipo de recursos y horas hombre.
- Prioridad.- Niveles de riesgo en la implementación (críticos, significantes u ordinarios).

Los puntos anteriores se toman en cuenta para estimar el tamaño del proyecto y decidir cómo se va a dividir.

La prioridad y los niveles de riesgo asociados con las características son usados para decidir en qué parte de la iteración se va a implementar.

Ejemplo
Para el ejemplo que se está desarrollando primeramente se identificó la problemática de la unión de crédito regional amequense y de la unión de crédito estatal de productores de café para poder definir los posibles requerimientos del cliente,.

Problemática Identificada en ambas Uniones de Crédito

1. *Por falta de un sistema de computación no se cuenta con información financiera del acreditado, lo que lógicamente limita la capacidad de análisis e incrementa el grado de riesgo en la operación crediticia.*
2. *No se realiza el análisis de riesgo porque no se cuentan con evidencias de referencias comerciales de los acreditados.*
3. *No se cuenta con soportes de la solvencia moral.*
4. *No se tiene un control estricto en la solicitud de garantías para el otorgamiento del crédito.*
5. *Tienen algunas deficiencias en el resguardo de documentación crediticia-operativa, arqueo, depuración documental y análisis de riesgo.*
6. *No se lleva un control de los créditos individualizados.*
7. *Existen problemas entre las áreas operativas ya que algunas cifras no cuadran con contabilidad (cuentas de orden del balance).*
8. *No cuenta con programas para el control de las operaciones de registro, control y seguimiento de la cartera en medios electrónicos (software) han desarrollado sus procedimientos a través de cálculo diseñados en Excel (versión 5.1) y desde el punto de vista técnico no reúnen las condiciones óptimas de memoria en sus sistemas.*
9. *No se práctica sistemáticamente la conciliación documental y numérica entre las áreas de contabilidad y crédito.*
10. *No existe en el libro de accionistas el control y seguimiento de las acciones; se lleva en una hoja de cálculo excel.*
11. *Existe duplicidad de funciones.*
12. *No cuenta con un registro histórico por insuficiencia en la memoria del equipo de cómputo.*
13. *El equipo de cómputo esta en nivel de obsolescencia por lo que tendrá que ser renovado.*
14. *No existen los reglamentos y manuales para la operación del crédito, los registros y controles que deben realizarse.*
15. *No cuenta con algunos servicios que son primordiales para los socios como son: Créditos para la producción, asesoría para invertir mejor sus recursos y supervisión para mejorar sus labores y abatir sus costos. Créditos Refaccionarios para la adquisición de maquinaria y ganado,*

asistencia técnica a la producción, existencia de un área de ahorro e inversión, la existencia de sucursales para socios con créditos para la producción primaria.

16. *La relación de accionistas emitida por la Comisión Nacional Bancaria y de Valores no reúne los requerimientos de presentación establecidos por este organismo.*
17. *No se tiene evidencia del acreditado en el Registro Federal de Contribuyentes.*
18. *En algunos contratos de créditos se presentaron diferencias entre el monto convenido y la suma de los pagarés relacionados con el contrato.*
19. *El otorgamiento de créditos se da actualmente a través de operaciones "amarradas".*
20. *No se práctica la clasificación de asociados, la calificación de cartera y estimación de reservas.*

De acuerdo a la problemática anterior, ésta se clasificó en varios rubros de tal forma que sirviera para definir de una forma más sencilla las necesidades del usuario.

1. *Información Compartida. Uno de los problemas que se tiene es que la información se maneja de forma separada en cada uno de los departamentos ya que en la mayoría de los casos la información generada en un departamento es la misma o tiene relación directa con la que utilizan otros departamentos; por eso existe duplicidad de información almacenada de diferente forma.*
2. *Presentan dificultades para coordinarse entre los departamentos, principalmente porque no existe una forma común de almacenar y tener acceso a la información.*
   Uno de los requerimientos que debe cumplir el sistema es permitir el manejo común de información, de manera que todos los departamentos involucrados con determinados datos tengan acceso directo sin que haya redundancia y utilizando los mismos formatos.
3. *Automatización de Procesos. Actualmente las actividades que se realizan en la unión se hacen de manera prácticamente manual, es decir no utilizan ninguna herramienta desarrollada especialmente para dichas actividades. Aunque utilizan hojas de cálculo para auxiliarse en los trabajos cotidianos y, en algunos casos se han creado macros y plantillas para adaptarlas a las necesidades de los usuarios, su uso resulta bastante complejo y depende completamente de la disposición y conocimiento de la persona que las crea y utiliza.*
4. *Es importante resaltar que los empleados de la Unión dedican gran parte de su tiempo a actividades administrativas las cuales en general son mecánicas y tediosas. Además, existen periodos en los cuales la carga de trabajo aumenta (por ejemplo en los días de pago o en épocas de siembra) y algunos empleados tienen que realizar muchas veces las mismas actividades y en esas ocasiones el tiempo es un factor importante a considerar puesto que se generan largas filas de socios esperando ser atendidos.*
5. *Debido a que los empleados casi siempre están ocupados con ese tipo de tareas, dedican poco o nada de tiempo para elaborar nuevos proyectos, supervisar los créditos que se encuentran en operación o realizar planes orientados a mejorar el funcionamiento de la unión.*
   Por eso, se ha identificado la necesidad de automatizar, los procesos administrativos que realizan actualmente de manera que su trabajo sea mucho más sencillo y más rápido.
6. *Coordinación entre departamentos. Existen actividades en las cuales se involucra a más de un departamento, tales actividades no sólo implican el manejo de la misma información, sino la realización de tareas conjuntas; por ejemplo cuando se otorga un crédito que requiere la realización de un proyecto. En tal situación un socio llega a solicitar un crédito, el departamento de crédito es el encargado de atenderlo y formular su solicitud, después solicita al departamento de cartera el techo crediticio del socio y el comportamiento del mismo en otros créditos, posteriormente canaliza la solicitud al departamento especial el cual, junto con el departamento de crédito y el socio se encargan de elaborar un proyecto de crédito; una vez elaborado el departamento de crédito se encarga de programar el otorgamiento de recursos al socio, los pagos y los intereses, posteriormente canaliza esta información al departamento de cartera y al departamento de contabilidad.*

7. *Esta forma de trabajar también implica que la información se maneje por separado, existiendo duplicidad en las tareas o que en algunos departamentos tengan que realizar lo que otros realizan o debieran realizar.*
   Por tal motivo se hace indispensable contar con una herramienta que permita no sólo tener acceso a la misma información, sino también tener una forma de manejar procesos que involucran la participación de distintos departamentos de manera simultánea o en secuencia. Esto implica que se deberá tener la posibilidad de coordinar procesos y compartir información.
8. *Apertura de Sucursales. Una de las preocupaciones de la unión es la necesidad de acercar geográficamente a sus socios los servicios que ofrece debido a que muchos de los socios se encuentren lejos y es difícil transportase al lugar en donde se encuentra actualmente la oficina de la unión.*
9. *También se han observado las posibilidades de un crecimiento futuro y con ello la necesidad de crear sucursales que dependan y estén coordinadas con una oficina matriz.*
   Por este motivo se hace necesario que el sistema tenga la posibilidad de coordinar y comunicar de manera prácticamente inmediata a las sucursales con la oficina matriz. Esto implica que se considere a las sucursales como un conjunto de departamentos los cuales podrán compartir información entre ellos y entre los departamentos de la oficina matriz, además de que deberán tener la posibilidad de realizar procesos de manera conjunta.
10. *Registro Histórico de Actividades. Otro de los problemas que se tienen es que no se cuenta con un registro histórico de los créditos operados por la unión debido a la dificultad que implica integrar toda la información referente a un crédito a los diferentes formatos que existen para el manejo de información y a la gran cantidad de recursos que se requieren para almacenarla en la forma en que se tiene capturada.*
    Estos registros pueden resultar de gran utilidad para poder evaluar el comportamiento que han tenido los socios en todos sus créditos pasados, ya que hasta el momento esta evaluación del socio se hace sólo con datos muy recientes y con el conocimiento personal de los socios. Esto permitirá el análisis de riesgo de los créditos y con ello tener mayor certeza de las condiciones en las cuales está operando la unión. Se considera que el sistema debe tener la capacidad de llevar un registro histórico de todos los movimientos realizados en la operación de los créditos, llevar un registro de qué y quién realiza las operaciones, principalmente las relacionadas con el manejo de recursos y la autorización de créditos y aquellas actividades que involucren un riesgo para la unión.

Comprender el Contexto del Sistema

Según los autores[1] es muy difícil capturar los requerimientos del cliente, por eso deben participar todas las personas involucradas para poder comprender el contexto del sistema; para entenderlo recomiendan dos formas: Modelando el dominio y Modelando el negocio.

- Modelar el Dominio se encarga de identificar y comprender las cosas que existen o eventos que suceden en el entorno en el que trabaja el sistema; éstas se obtienen mediante una especificación de requerimientos o a través de entrevistas con los expertos del dominio. Su objetivo principal es identificar las clases más importantes y definir un glosario de términos.
- El Modelo del Negocio es una técnica que se encarga de comprender los procesos de negocio de la organización y los plasma en términos de casos de uso y actores y a su vez, su relación entre ellos.

Modelar el Dominio

La modelación del dominio captura los objetos más importantes del contexto del sistema. El dominio de objetos o clases representa las cosas que existen o eventos que suceden en el ambiente en el que trabaja el sistema; muchos de estos objetos del dominio o clases son

[1] Jacobson, Booch y Rumbaugh

importantes porque permiten encontrar fácilmente la especificación de los requerimientos o sirven para realizar entrevistas con los expertos.

Existen tres formas para llegar a las clases del dominio:
1. Objetos de Trabajo.- Representan cosas que son manipuladas en un trabajo como ordenes, contratos, cuentas, etc.
2. Objetos y conceptos del mundo real.- Que necesita un sistema para que funciones, por ejemplo para atacar a los enemigos misiles, aeronaves, etc.
3. Eventos.- Que han ocurrido, como el arribo de una aeronave, la marcha de la aeronave, etc.

Desarrollar el Modelo del Dominio

El propósito de modelar el dominio es comprender y describir las clases más importantes en el contexto del dominio; generalmente requiere entre 10 y 50 clases. Las clases que no se consideran pero que son candidatas se guardan como definiciones en un glosario de términos, por eso es importante considerar la terminología de los involucrados y participantes en el sistema para que todos lo entiendan.

Si en algunos casos es pequeño el dominio, no es necesario desarrollar un modelo del objeto para el dominio; en su lugar, un glosario de términos es suficiente. El glosario y el modelo del dominio ayudan a los usuarios, clientes y desarrolladores para tener un vocabulario común.

En algunos casos es muy importante tener en mente el propósito de modelar el dominio para contribuir a comprender el contexto del sistema y así mismo a comprender los requerimientos que originan el contexto. En otras palabras, modelar el dominio contribuye a comprender el problema que el sistema va a resolver.

Las clases del dominio y el glosario de términos se utilizan:
1. Cuando se describen los casos de uso y el diseño de la interfaz de usuario.
2. Para asegurar las clases internas durante el desarrollo del análisis.

Según los autores un glosario es usado para definir los términos más importantes y comunes que usan los analistas y los desarrolladores cuando describen el sistema; un glosario es útil cuando consigue un consenso entre los desarrolladores respecto a la definición de varios conceptos y notaciones para reducir el riesgo de conceptos falsos.

Para realizar el glosario de términos se toman en cuenta objetos físicos como facturas, contratos, recibos, entre otros; éstos se revisan con el cliente.

Ejemplo de Glosario de términos

Solicitante	*Persona que desea ser socio de la Unión.*
Gerente	*Persona encargada de la Unión de Crédito.*
Comité de Admisión	*Grupo de personas que trabajan en la Unión y se encargan de autorizar los Créditos y el ingreso de personas que desean ser socios.*
Solicitud de Ingreso a la Unión	*Formato llenado por un empleado de la unión que contiene datos generales de la persona que desea ser socio.*
Ex Socios	*Personas que fueron socios de la unión pero que ya no lo son.*
Socios Vigentes	*Son personas físicas, morales empresa y morales organización, que son parte fundamental de la unión y que tienen derecho*

	a préstamos.
Solicitud de crédito	*Documento que describe una solicitud de un préstamo de dinero del socio a la Unión*
Encargado de Crédito	*Persona responsable de revisar si se le puede otorgar un crédito a los socios de la Unión y de calcularlo.*
Encargado de Cartera	*Persona responsable del comportamiento de los créditos.*
Encargado de Caja	*Persona encargada de cobrar y dar el dinero de los créditos.*
Acciones	*Documentos que avalan a las personas como socios de la Unión y especifican el capital invertido por los socios.*
Garantías	*Documentos de los socios que amparan un bien hipotecario que sirve para amparar un crédito.*
Contrato Fuente	*Documento que ampara un crédito de la Unión con la o las Fuentes Financieras; especificando las líneas su monto, las ministraciones, las recuperaciones y las tasas de interés.*
Paquete Tecnológico	*Documento otorgado por la Fuente Financiera donde especifica las labores con sus procesos respectivos indicando la cantidad y el costo por cada proceso para poder determinar el costo total por hectárea de determinado cultivo.*
Actividades	*Giro al que se dedican los socios y los empleados de la unión.*
Autorizaciones	*Empleados de la unión que están autorizados a otorgar un crédito o un ingreso a la unión de una persona como socio.*
Ciclos	*Período en que se puede sembrar o recoger la cosecha.*
Tipos de Crédito	*Tipos de préstamos que otorga la unión.*
Comisiones del contrato de la fuente	*Tipos de comisión que se les cobra a la unión por parte de las fuentes financieras y a su vez, la unión a los socios al otorgarles un crédito.*
Estados de la República Mexicana	*Divisiones geográficas de la república mexicana.*
Estado Ministraciones	*Comportamiento del dinero que se les presta a los socios.*
Unión de Crédito	*Asociación que se dedica a otorgar créditos a sus socios, principalmente a los campesinos que se dedican al cultivo.*
Carpeta de Solicitudes de Crédito	*Conjunto de solicitudes de crédito pendientes para su autorización.*
Dinero	*Especie monetaria*
Falta de dinero	*Necesidad que pueden tener los socios como la unión.*

Tipo de Socios	*Especifica la figura fiscal (Persona Física, Moral Empresa y Moral Organización)*
Requerimientos para ser socio	*Lista de documentación que debe presentar la persona que desea ser socio de la unión.*
Recuperaciones	*Número de pagos que va a realizar el socio a la unión.*
Ministraciones	*Forma en que le va a otorgar el dinero del crédito la unión al socio.*
Estado de Cuenta	*Documento que refleja el comportamiento de un crédito.*
Carpeta de Solicitudes de Ingreso	*Conjunto de solicitudes de ingreso pendientes para su autorización.*
Cartera	*Comportamiento de los créditos en operación de los socios.*
Usuarios del sistema	*Empleados de la unión que operan el sistema.*
Fuentes	*Nombre de las entidades financieras que les otorga créditos a la unión.*
Líneas	*Tipos de cultivos que realizan los socios.*
Labores	*Tipo de actividad que realiza el campesino para sembrar.*
Procesos Labor	*Conjunto de procesos que se llevan a cabo para realizar una labor.*
Motivos Baja	*Razones por las que un socio puede renunciar a la unión.*
Puestos	*Nombre de la actividad que realiza un socio.*
Regiones	*Orientación geográfica norte, sur, este, oeste*
Tasas	*Interés que se va a cobrar a los socios por un crédito que se les otorgue.*

Una vez identificado el glosario se realizó una lista de categorías sin importar el orden de la importancia para poder identificar más claramente las clases.

Categoría	*Concepto*
Objetos Físicos o tangibles	*Tierras para el cultivo, garantías hipotecarias*
Especificaciones, diseño o descripciones de cosas	*Tipos de Crédito, Tipos de Líneas, Carpeta de solicitudes de Ingreso, Carpeta de solicitudes de crédito.*
Lugares	*Unión de Crédito*
Transacciones	*Altas, bajas, consultas, cambios, dar y recibir dinero, intereses, estados de cuenta.*
Línea o renglón de elemento de transacciones	*Socios, Solicitud de crédito, Solicitud de Ingreso a la Unión, Solicitud de crédito, Acciones, Contrato Fuente, Ciclos, Paquete Tecnológico, Actividades, Créditos, Comisiones, Autorizaciones, Estados de la República Mexicana, Fuentes, Líneas, Estado Ministraciones, Puestos, Motivos Baja, Regiones, Tasas.*
Papel de las personas	*Gerente, Comité de Admisión, Socios,*

	Encargado de Cartera, Encargado de Caja, Encargado de Crédito.
Conceptos de nombres abstractos	*Falta de recursos financieros.*
Organizaciones	*Fuentes Financieras*
Eventos	*Altas, bajas, consultas, cambios, dar y recibir dinero, cálculo de intereses.*
Procesos	*Otorgar créditos*
Reglas y políticas	*Ser socio de la unión*
Catálogos	*Acciones, Ciclos, Actividades, Créditos, Comisiones, Autorizaciones, Estados de la República Mexicana, Fuentes, Líneas, Estado Ministraciones, Puestos, Motivos Baja, Regiones, Tasas, recuperaciones.*
Registros de finanzas, de trabajo de contratos de asuntos legales	*Contrato de la fuente, capital de la unión.*
Instrumentos y servicios financieros	*Valor de las tasas*
Manuales, libros	*Políticas de la Unión.*

De acuerdo a la lista anterior se identificaron las posibles clases

Acciones	*Créditos*
Ciclos	*Solicitud de crédito*
Actividades	*Solicitud de Ingreso*
Comisiones	*Paquete tecnológico*
Autorizaciones	*Tasas*
Estados de la Rep Méxicana	*Regiones*
Fuentes Financieras	*Socios vigentes*
Líneas de crédito	*Exsocios*
Tipos de crédito	*Labores*
Contrato con la fuente	*Procesos*

Modelo del Negocio

Modelar el negocio es una técnica que ayuda a comprender los procesos de trabajo de una organización; la regla es identificar los casos de uso a través de los procesos, los actores y las entidades de la organización que puedan ser soportadas por el sistema así como conocer su contexto.

Un modelo de objetos describe cómo cada caso de uso lo realizan los empleados que usan una serie de entidades(por ejemplo una factura) y unidades de trabajo(conjunto de entidades); cada caso de uso se detalla con un diagrama de interacción y un diagrama de actividades.

Las entidades y las unidades de trabajo se usan para representar algunos tipos de conceptos como las clases del domino, ordenes, facturas y cuentas. Cada trabajador, entidad de trabajo y unidad de trabajo pueden participar en la realización de más de un caso de uso.

Para desarrollar un modelo del negocio:

1. Primero se deben identificar los casos de uso del negocio donde especifique los actores y casos de uso generales e importantes del negocio.

2. Posteriormente se desarrolla un modelo de objetos compuesto de empleados, entidades y unidades y la relación entre ellos para realizar los casos de uso.

Encontrar Casos de Uso para un Modelo del negocio
Usando un modelo del negocio como suministro, el analista primero identifica un actor por cada trabajador, posteriormente todos los roles de cada actor identificado y finalmente un caso de uso por cada rol; por eso un actor puede tener varios casos de uso ya que cada caso de uso representa las actividades que realiza cada actor.

Ejemplo
Según los puntos anteriores y la lista de categorías (realizada en el glosario de términos) primero se identificaron los actores principales como son: Comité de admisión, administrador del sistema, encargado de crédito, encargado de cartera, contador general, cajero y encargado de la tenencia accionaria y los casos de uso generales a partir de sus actividades.

Por cada actor se identificó un caso de uso general

Tenencia Accionaria	*Caso de Uso: Control de la tenencia accionaria. Llevar un control de los socios de la unión que considere los datos generales, tipo de socio, capital invertido en la unión y antigüedad por socio.*
Comité de Admisión	*Caso de Uso: Evaluar Solicitudes de Crédito y de Ingreso. El comité de admisión debe evaluar las solicitudes de créditos de los socios así como las solicitudes de ingreso de personas que desean ser socias de la unión.*
Contador General	*Caso de Uso Emitir los Cheques de las Ministraciones. El Contador General debe verificar que la unión tenga fondos para emitir los cheques de las ministraciones vencidas de los socios.*

Crédito

Caso de Uso Otorgar Créditos a los socios de la unión de crédito. El encargado del departamento de crédito debe llenar solicitudes de crédito de los socios, verificar que el socio cumpla con todos los requerimientos para otorgarle el crédito y posteriormente pasar la solicitud al comité para que la evalúe.

Cartera

Caso de Uso Controlar los Créditos vigentes y vencidos de la unión de crédito. El encargado de cartera debe estar al tanto del comportamiento de los créditos, como cálculo de intereses, cortes, emisión de estados de cuenta, emitir recibos de vencimiento de créditos a los socios y envío de avisos.

Cajero

Caso de Uso Recibir las Recuperaciones de los créditos, emitir comprobantes de pago, dar las ministraciones y emitir recibos de que ya se dieron las ministraciones.

Administrador del sistema

Caso de Uso Administrar el sistema . El encargado debe dar de alta a los usuarios del sistema, debe otorgar permisos para su manejo y debe realizar respaldos de la información. .

Requerimientos Funcionales

De acuerdo a la identificación de los actores y casos de uso generales, se analizó cada uno para definir las actividades detalladas que realizan por cada caso de uso general y determinar varios casos de uso específicos con los que se va a trabajar durante todo el flujo de trabajo del proyecto.

Casos de Uso específicos

1. *Controlar el Ingreso de personas que desean ser socias de la Unión.- Llenar una solicitud de ingreso, registrar los requerimientos que debe cumplir para ser socio, pagar las acciones, permitir modificarla, evaluarla y formalizarla para ser socio de la unión, consultar solicitudes de ingreso, consultar solicitudes pendientes por autorizar, registrar tipo de socio, actividad, capital que aporta a la unión, si es socio moral cuántos socios indirectos dependen de él e imprimir la solicitud.*
2. *Integrar la Carpeta de Solicitudes de Ingreso.- El sistema debe ordenar las solicitudes por fecha, por crédito, por socio y por monto para que las evalúe el comité. debe imprimirlas por cualquiera de las opciones anteriores.*
3. *Controlar la venta y compra de acciones así como el comportamiento de cada acción.- Llevar un registro del comportamiento de las acciones por serie, por socio y por unión.*

4. *Llevar un registro de los socios vigentes.- Debe permitir la consulta global e individual por cada socio vigente presentando sus datos generales, su actividad, capital aportado a la unión, su antigüedad, comportamiento de sus créditos, tipo de socio, sus garantías y la impresión de los datos anteriores.*
5. *Llevar un Histórico de los ExSocios de la Unión.- Debe existir un registro de las personas que ya no son socias indicando las razones por las que renunció, permitir la consulta, imprimir y eliminar exsocios.*
6. *Debe permitir registrar personas que ya son socios pero que no se han registrado en el sistema.*
7. *Debe automatizar los créditos que otorgan a los Socios.- Llenar una solicitud de crédito por socio directo y socio indirecto, registrar las garantías, requerimientos y avales que amparen la solicitud de crédito, registrar las líneas que van a conformar el crédito tanto del contrato con la fuente como del paquete tecnológico, verificar que el socio o socios se les pueda otorgar el crédito; se deben modificar, evaluar, formalizar, eliminar e imprimir las solicitudes. Se deben registrar a la o las personas que autorizan el crédito. Si el crédito no es autorizado indicar las razones en la solicitud.*
8. *Integrar la carpeta de solicitudes de crédito.- El sistema debe ordenar las solicitudes por fecha, por crédito, por socio y por monto para que las evalúe el comité. Debe imprimirlas por cualquiera de las opciones anteriores.*
9. *Automatizar el comportamiento de los crédito por socio directo e indirecto.- Se debe llevar un control de los créditos por socio directo e indirecto, así como un socio directo con sus respectivos socios indirectos, debe permitir la consulta e impresión por socio y por crédito.*
10. *Registrar la entrega y recepción del crédito.- El sistema debe registrar la autorización o bloqueo de las ministraciones y su entrega; debe calcular los intereses, realizar los cortes mensuales, debe emitir estados de cuenta por socio directo e indirecto, por crédito, debe registrar las tasas mensuales por crédito para calcular los intereses, debe registrar la recepción de las recuperaciones. Debe imprimir toda la información anterior.*
11. *Registrar los créditos que le otorgan las fuentes financieras a la unión de crédito.- Debe registrar los contratos de los créditos que le otorgan la fuentes financieras a la unión de crédito como tipo de crédito, monto, número de contrato, núm. de cta. de cheques a donde se va a depositar el crédito, las líneas, su monto, su fecha de apertura y vencimiento, las comisiones que va a cobrar por el crédito que otorgó, el porcentaje de participación por cada fuente en el crédito, las tasas de interés ordinarias y moratorias que va a aplicar a la unión y a los socios, en cuántas ministraciones va a dar el dinero, el monto de cada ministración y la fecha de entrega de cada una. Cuándo va a recuperar ese dinero, si va a ser en una o varias recuperaciones. Consulta de contratos globales, individuales, por tipo de crédito y por líneas. Unión del contrato con un paquete tecnológico, eliminar contratos que no estén relacionados a un paquete o a un crédito. E imprimir todo lo anterior.*
12. *Unir un Contrato con un Paquete tecnológico.- Se debe relacionar una contrato con un paquete para poder otorgar el crédito al socio e indicar en qué va a poder usar ese recurso, debe coincidir tipo de crédito, líneas y ciclo.*
13. *Registrar los Paquetes Tecnológicos.- Debe registrar el tipo de crédito, el ciclo, calcular los costos por línea, por labor y por procesos por cada línea para que de un total por hectárea y así registrar las ministraciones y recuperaciones por línea. Debe permitir crear un paquete de uno que ya existe, modificarlo, eliminarlo (siempre y cuando no este unido con un contrato) e imprimirlo.*
14. *El sistema debe funcionar en sucursales y en la matriz.- El sistema debe correr en línea en diferentes terminales y sucursales sin ningún problema.*
15. *El sistema debe automatizar la contabilidad de la unión.- Debe afectar las cuentas contables para realizar automáticamente los estados financieros.*
16. *Debe permitir el manejo común de información entre usuarios y departamentos.*
17. *Debe controlar el acceso al sistema.- El sistema debe controlar el acceso al sistema y a los módulos a través de una contraseña del usuario.*
18. *Debe permitir el respaldo de información*

Requerimientos No Funcionales

Los requerimientos no funcionales especifican las propiedades del sistema, así como las restricciones en el ambiente, en la implementación, en la ejecución, en la plataforma, etc. Estos requerimientos imponen condiciones a los requerimientos funcionales como la rapidez en el tiempo de respuesta y uso de la memoria, se deben considerar en la descripción de los casos de uso.

Ejemplo

El problema que se encuentra en la identificación de requerimientos es el equipo de cómputo que tienen actualmente ya que es muy lento, el procesador de la mayoría de sus PC's es 486 y el espacio en disco es muy pequeño.

2.2.3. Modelo de Casos de Uso

El modelo de casos de uso permite que los desarrolladores y los clientes lleguen a un acuerdo sobre las condiciones y posibilidades que debe cumplir el sistema, por eso el modelo es una guía para el análisis, diseño y pruebas. Va a estar formado de actores, casos de uso y su relación entre ellos.

Si el modelo es muy grande y difícil de entender por el número de actores y casos de uso, UML permite presentar el modelo en diagramas a través de paquetes para minimizar el tamaño.

En un modelo de casos de uso se deben llevar a cabo las siguientes actividades, pero éstas no tienen porque ser ejecutadas en un orden en particular ya que generalmente se hacen simultáneamente.

1. Encontrar actores y casos de uso,
2. Priorizar casos de uso,
3. Detallar casos de uso,
4. Prototipo de la interfaz de usuario y
5. Estructurar el modelo de casos de uso.

Encontrar Actores y Casos de Uso

Para identificar los casos de uso y los actores se debe:

1. Definir el perfil de quién y qué actores interactuarán con el sistema y qué funcionalidades (casos de uso) van a desarrollar.
2. Delimitar el sistema a su entorno
3. Capturar y definir un glosario de términos común que sea esencial para crear y detallar las descripciones de la funcionalidad del sistema.

Identificar los actores y los casos de uso es la actividad más importante para obtener los requerimientos correctos.

Esta actividad consiste de cuatro puntos, los cuales no tienen porque ser ejecutados en un orden en particular, ya que generalmente se hacen simultáneamente.

1. Encontrar actores
2. Encontrar casos de uso
3. Describir brevemente cada caso de uso
4. Describir el modelo de casos de uso

Encontrar Actores

Cada tipo de usuario y cada sistema externo esta representado en uno o más actores; los actores representan las partes externas del sistema que colaboran con él, por eso, primero se identifican todos los actores y posteriormente el medio ambiente externo. Frecuentemente los actores corresponden a los empleados de una compañía, cada rol del empleado define qué función o proceso realiza en la empresa; estos roles se pueden usar para derivar un caso de uso por cada rol que le corresponde al actor, por eso un actor juega un rol por cada caso de uso.

La tarea de encontrar actores va a depender del punto de partida; lo ideal es contar con un modelo del negocio, ya que este modelo va a permitir que sea más sencillo, porque por cada trabajador del negocio se va a asignar un actor que va a utilizar la información del sistema. Pero si no se cuenta con un modelo el analista con el cliente identifica los usuarios y trata de organizarlos en categorías que representen los actores de sistemas externos y los actores para el sistema que le den mantenimiento y operación necesaria. Para identificarlos se deben considerar dos puntos importantes:

- Primero, identificar al menos un usuario que pueda ser el actor candidato, este puede ayudar a identificar los actores relevantes y evitar actores fantasmas por nuestra imaginación.
- Segundo, debe haber un mínimo de coincidencias entre los roles de los diferentes actores que juegan con relación al sistema, ya que no se quiere más de dos actores para el mismo rol. El analista de sistemas nombra a los actores y brevemente describe los roles de cada uno y quiénes van a usar el sistema.

El resultado de este punto, es una versión del modelo de casos de uso con un conjunto de actores cada uno con una breve descripción. Estos actores ahora pueden usarse como punto de partida para encontrar los casos de uso.

Ejemplo
Para ejemplificar lo anterior se tomo como base el modelo del negocio ya que ahí se identificaron los actores principales de la organización y que además tienen relación con los casos de uso generales e importantes.

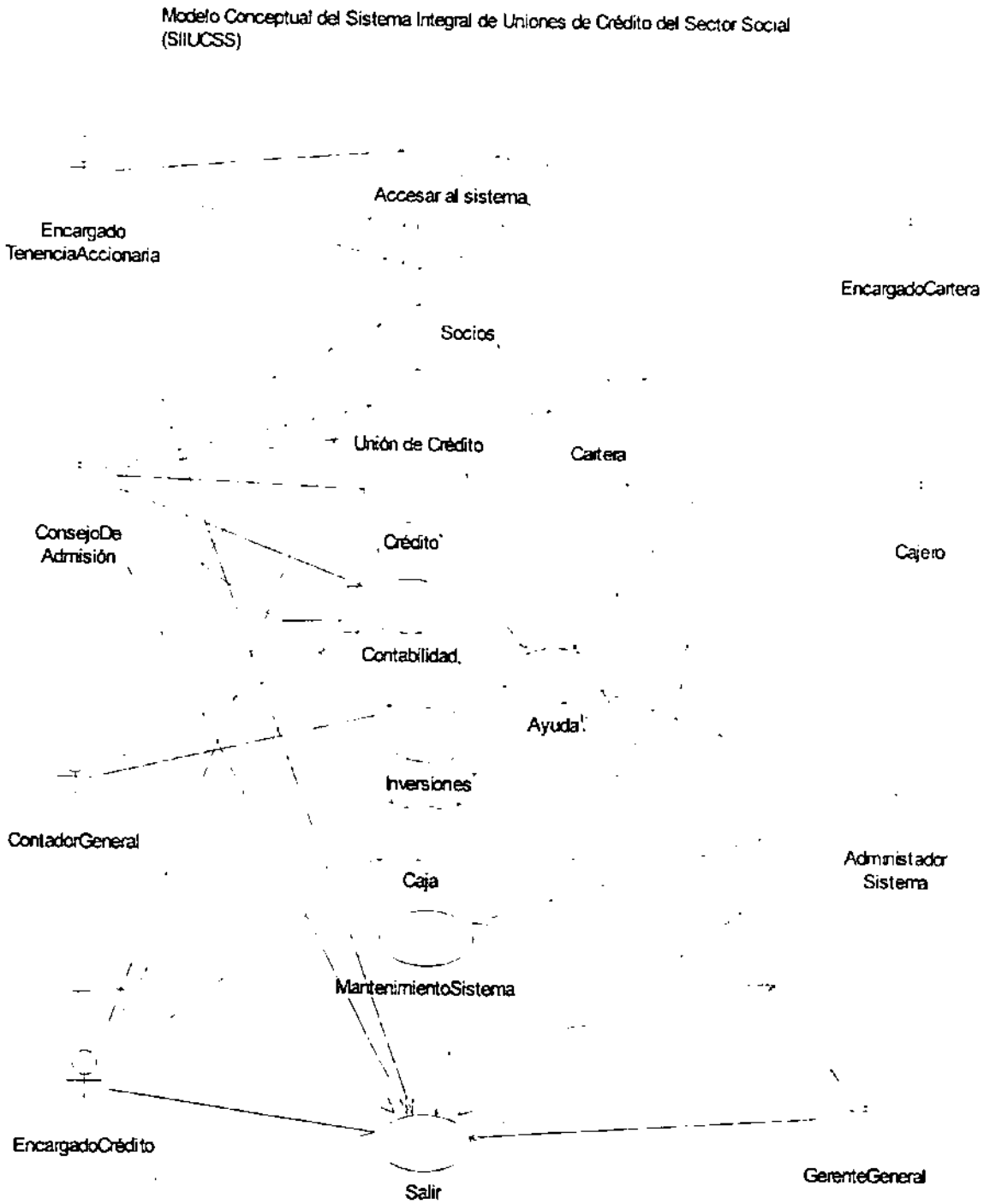
ACTORES	**ACTIVIDADES**
Consejo de Administración	*1. Tienen que ver con la operación de la Unión de Crédito* *2. Las relaciones con las fuentes financieras, los planes de operación, los servicios complementarios, la recuperación del crédito, tratamiento de la cartera vencida.* *3. Asuntos administrativos.* *4. Autorizan créditos.* *5. Autorizan ingresos a la unión de crédito.*
Gerente General	*1. Cuenta con poderes amplios sobre la Unión.* *2. Tiene facultades para autorizar préstamos hasta por $ 10,000.* *3. Elabora programas de crédito* *4. Es el responsable de la unión de crédito.* *5. Tiene acceso a todo el sistema.*
Encargado de Crédito	*1. Es responsble de llenar las solicitudes de crédito.* *2. Revisar los requistos que debe cumplir para poder otorgar un crédito.* *3. Revisar las garantías y los avales.* *4. Registrar los socios indirectos que van a conformar el crédito.* *5. Integrar la carpeta de solicitudes para pasarsela al comité para su evaluación.*

	6. *Revisar que los socios que solicitan el crédito no tengan algún problema económico con la unión.* 7. *Si la solicitud es aceptada, formalizarla con el socio para que pase a ser un crédito vigente.* 8. *Registrar los contratos con la fuente financiera.* 9. *Registrar los paquetes tecnológicos.* 10. *Unir los contratos con los paquetes.*
Encargado de Cartera	1. *Se encarga del registro, control y seguimiento de la cartera.* 2. *Lleva el resguardo de garantías.* 3. *Se encarga de informar a los socios las fechas del vencimiento de sus créditos.* 4. *Realiza el cálculo de intereses para los efectos de cobranza y se los reporta al departamento de crédito y a contabilidad.* 5. *Realiza los cortes mensuales.* 6. *Consulta de saldos* 7. *Emite estados de cuenta* 8. *Integra la carpeta de ministraciones pendientes por autorizar.* 9. *Verifica si se pueden otorgar las ministraciones de los créditos vigentes.* 10. *Da aviso al contador de las ministraciones que se van a dar a los socios para que emita los cheques.*
Contador General	1. *Realización de conciliaciones bancarias, controla entradas y salidas de efectivo, control del capital social, realiza aplicaciones contables de todas las pólizas tanto de recursos internos como externos; supervisa y revisa las obligaciones fiscales y laborales, calcula pagos trimestrales y anuales del ISR e IVA, elabora los estados financieros, realiza proyecciones de gastos presupuestados y reales y cada 3 meses, las relaciones de responsabilidades para determinar el capital de cada socio y los créditos que recibe.* 2. *Así mismo, concilia con cartera y crédito los movimientos de la tenencia accionaria y los de cartera de créditos.* 3. *Emite los cheques para los socios.*
Encargado de la Tenencia Accionaria	1. *Es responsable de llenar las solicitudes de ingreso.* 2. *Revisar los requisitos que debe cumplir para poder ser socio de la unión.* 3. *Integrar la carpeta de solicitudes para su evaluación.* 4. *Pasar la carpeta al comité.* 5. *Registrar la compra y venta de acciones.* 6. *Registrar la evaluación de la solicitud.* 7. *Formalizar la solicitud de ingreso.* 8. *Hacer modificaciones a la solicitud.* 9. *Realizar consultas de socios vigentes, de solicitudes pendientes, de Exsocios, del comportamiento de las acciones y de los folios.* 10. *Registrar la salida de socios y las razones por las que desea dejar la unión de crédito.*
Cajero	1. *Recibe los pagos de la compra y venta de acciones.*

	2. *Reçibe las recuperaciones de los créditos.* 3. *Da los cheques a los socios.* 4. *Emite comprobantes de pagos.*
Administrador del Sistema	1. *Se encarga de administrar el sistema.* 1. *Otorgar permisos a los usuarios.* 2. *Asignar claves.* 3. *Hacer modificaciones a los permisos y a las claves.* 4. *Respaldar información.* 5. *Revisar que funcione correctamente el sistema*

Tabla 2.1 Actores principales de la organización

Una vez identificados los actores y los requerimientos se definió el modelo conceptual del sistema integral de uniones de crédito del sector social (SIIUCSS).



Encontrar Casos de Uso

Un caso de uso especifica una secuencia de acciones que el sistema lleva a cabo al interactuar con sus actores, incluyendo alternativas una instancia de una caso de uso no iteractúa con otra instancia de otro caso de uso, únicamente los tipos de interacciones ocurren entre instancias de actores e instancias de casos de uso. El comportamiento de cada caso de uso se puede interpretar independientemente de otro caso de uso el cual permite que la modulación de los casos de uso sea simple.

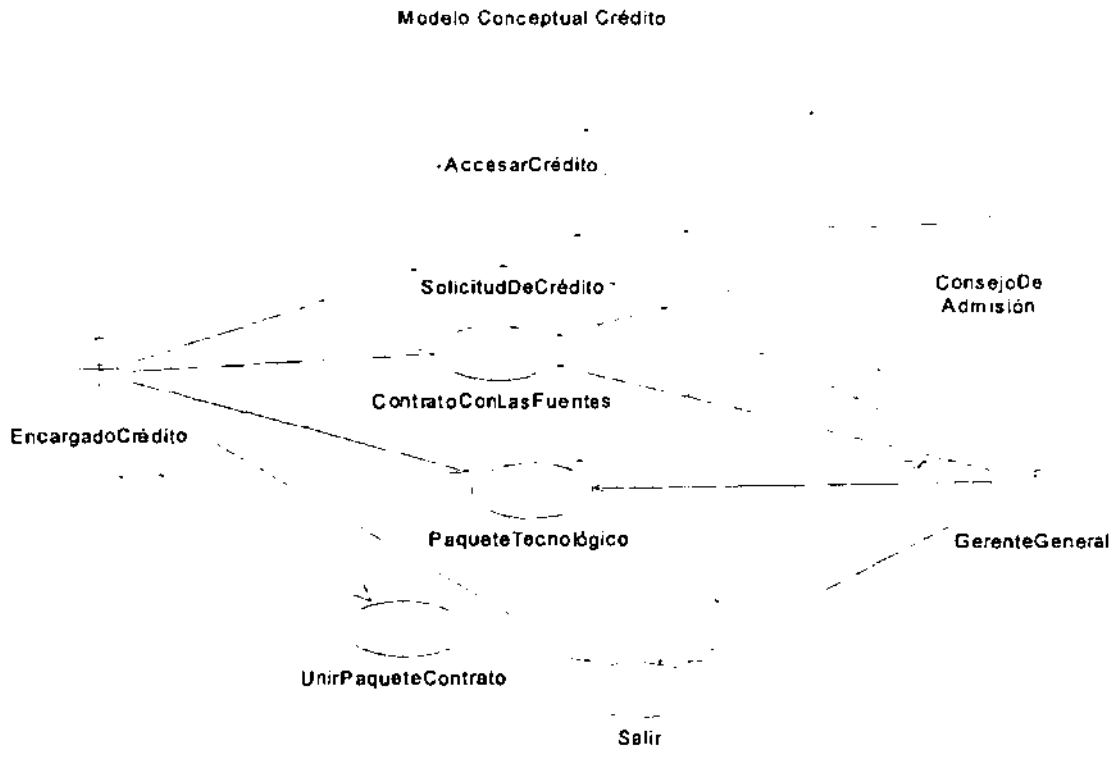
Un caso de uso especifica el comportamiento de cosas dinámicas, se representa en los diagramas de estado, diagramas de actividades, diagramas de colaboración y diagramas de secuencia.

- Un diagrama de estados especifica el ciclo de vida de las instancias del caso de uso en términos de estados y transiciones entre estados. Cada transición es una secuencia de acciones.
- Un diagrama de actividades describe el ciclo de vida más detallado, describe la secuencia de acciones que ocurren en cada transición.
- Un diagrama de secuencias y de colaboración describen las interacciones entre las instancias de los actores y las instancias de los casos de uso.

Se recomienda agregar un nombre por cada caso de uso, frecuentemente debe empezar con un verbo y reflejar cuál es el objetivo de la interacción entre el actor y el sistema.

Ejemplo

De acuerdo al modelo conceptual del sistema integral de uniones de crédito del sector social se eligió uno de sus casos para ejemplificar el flujo de trabajo del proceso unificado a través de una iteración, por eso a continuación se representa el modelo conceptual del caso de uso crédito.



De acuerdo al modelo conceptual anterior se eligió uno de ellos para ejemplificar este punto "contrato con las fuentes" que dio origen a partir del siguiente requerimiento:

Registrar los créditos que le otorgan las fuentes financieras a la unión de crédito.- Debe registrar los contratos de los créditos que le otorgan la fuentes financieras a la unión de crédito como tipo de crédito, monto, número de contrato, a que cuenta se va a depositar el crédito, las líneas, su monto, su fecha de apertura y vencimiento, las comisiones que va a cobrar por el crédito que otorgó, el porcentaje de participación por cada fuente en el crédito, las tasas de interés ordinarias y moratorias que va a aplicar a la unión y a los socios, en cuántas ministraciones va a dar el dinero, el monto de cada ministración y la fecha de entrega de cada una. Cuándo va a recuperar ese dinero, si va a ser en una o varias recuperaciones. Consultar contratos por tipo de crédito y por líneas, modificar, eliminar e imprimir.

Describir brevemente cada Caso de Uso

Los analistas describen cada caso de uso brevemente en una pequeña oración describiendo paso a paso que necesita hacer el sistema cuando interactúa con los actores.

Ejemplo:
De acuerdo a la descripción del caso de uso general del punto anterior, se dividió en varios casos de uso de tal forma que permitiera su comprensión y fuera más fácil trabajar.

Los casos de uso que se identificaron son:

1. *Registrar el contrato con las fuentes.- El sistema despliega la Forma solicitando los datos generales del contrato como son: núm. de contrato, tipo de crédito, monto del contrato, fecha de apertura, cuenta de cheques, si acepta o no pagos. El encargado captura los datos generales.*
2. *Registrar las comisiones .- El sistema despliega una forma solicitando el No. de Contrato, el tipo de comisión y el monto. El encargado captura los datos y el sistema los guarda.*
3. *Registrar las líneas .- El sistema despliega la Forma Líneas solicitando los datos de la líneas: ciclo, unidades, fecha de vencimiento y monto. El encargado los teclea. El sistema los guarda-*
4. *Registrar las fuentes.- El sistema despliega una forma solicitando los datos de la fuente: Fuente, % de Participación, Monto, Observaciones, Tasas Ordinarias y Moratorias. El encargado captura la fuente, el % de participación (el sistema calcula automáticamente el monto) y las observaciones.*
5. *Registrar las tasas ordinarias por fuente.- El sistema despliega una forma solicitando el Tipo de Tasa, el Factor y No. de Puntos. El encargado captura los datos.*
6. *Registrar las tasas moratorias por fuente.- El sistema despliega una forma solicitando el Tipo de Tasa, el Factor y No. de Puntos. El encargado captura los datos.*
7. *Registrar ministraciones por línea.- El sistema despliega la Forma Ministraciones solicitando No. de Ministración, Fecha e Importe. El encargado captura los datos. El sistema los guarda.*
8. *Registrar recuperaciones por línea.- El sistema despliega la Forma Recuperaciones solicitando No. de Recuperación, Fecha e Importe. El encargado captura los datos.*
9. *Eliminar las comisiones del contrato con las fuentes.- El encargado selecciona la Comisión que va a eliminar. El sistema busca la comisión, la elimina y actualiza la Forma Contrato.*
10. *Eliminar las líneas con sus ministraciones y recuperaciones.- El encargado selecciona la línea que va a eliminar. El sistema manda un mensaje de advertencia, primero debe eliminar las ministraciones y recuperaciones. El sistema elimina la línea, las ministraciones y recuperaciones relacionadas a ella.*
11. *Eliminar las fuentes con sus tasas ordinarias y moratorias.- El encargado selecciona la fuente que va a eliminar. El sistema manda un mensaje de advertencia avisando que se van a eliminar las tasas correspondientes a esa fuente. El sistema elimina la fuente y las tasas.*
12. *Consulta de contratos por tipo de crédito.- El sistema despliega una forma con los contratos que tiene la unión con fuentes financieras ordenados por tipo de crédito.*

13. *Consulta de contratos por líneas.- El sistema despliega una forma con los contratos que tiene la unión con fuentes financieras ordenados por línea.*
14. *Consultar un contrato .- El sistema despliega una forma con todos los datos de un contrato, fuentes, comisiones, líneas y tasas.*
15. *Eliminar un contrato.- El encargado selecciona el contrato que desea eliminar. El sistema busca el contrato, lo elimina y actualiza la forma.*
16. *Imprimir un contrato.- El encargado selecciona el contrato que desea imprimir. El sistema actualiza la impresora y lo manda imprimir.*
17. *Imprimir una lista de contratos por tipo de créditos y por líneas.- El encargado selecciona lo que desea imprimir. El sistema actualiza la impresora y lo imprime.*
18. *Modificar datos del contrato.- El encargado selecciona el contrato que desea modificar. El sistema lo despliega editando los campos que se pueden modificar. El encargado lo modifica. El sistema lo guarda.*
19. *Eliminar tasas ordinarias.-. El encargado selecciona la Tasa que va a eliminar. El sistema busca la tasa, la elimina y actualiza la Forma.*
20. *Eliminar tasas moratorias.-. El encargado selecciona la Tasa que va a eliminar. El sistema busca la tasa, la elimina y actualiza la Forma.*
21. *Eliminar Ministraciones.- El encargado selecciona la ministración que desea eliminar. El sistema localiza la ministración, resta el monto de la ministración al monto acumulado, la elimina y actualiza la Forma.*
22. *Eliminar Recuperaciones.- El encargado selecciona la recuperación que desea eliminar. El sistema localiza la recuperación, resta el monto de la recuperación al monto acumulado, la elimina y actualiza la Forma.*

Nota: Al describir y estructurar el caso de uso "registrar un contrato con las fuentes" se observo que es un caso de uso general muy grande, que se subdivide en varios casos de uso por lo que a partir de aquí se decidió tratar de trabajar únicamente con este caso de uso para ejemplificar una iteración en el desarrollo de está investigación.

Es importante explicar cómo los casos de uso y los actores están relacionados unos con otros y cómo ellos en conjunto forman el modelo de casos de uso.

Describir el Modelo de Casos de Uso

Se elaboran diagramas para describir y explicar cómo los casos de uso se relacionan entre si y con los actores. No existe una regla estricta sobre lo que se debe incluir en un diagrama, de hecho se pueden agregar varios diagramas para que ayuden a describir claramente el sistema.

Para asegurar la consistencia cuando se describen varios casos de uso, es práctico desarrollar un glosario de términos; estos términos pueden derivarse en clases en un modelo del dominio o modelo del trabajo. El modelo de casos de uso puede estar organizado en agrupaciones de casos de uso llamado paquetes de casos de uso ya que describe cómo los actores y los casos de uso iteractúan y cómo los casos de uso están relacionados uno con otro.

Cuando ya existe el modelo de casos de uso, gente externa al ambiente del sistema puede revisar el modelo y conducir a una revisión informal para determinar si:

1. Todos los requerimientos funcionales necesarios son capturados como casos de uso.
2. La secuencia de acciones es correcta, completa y comprensible por cada caso de uso.
3. Se identifica algún caso de uso que no proporcione valor.

Ejemplo de glosario de términos

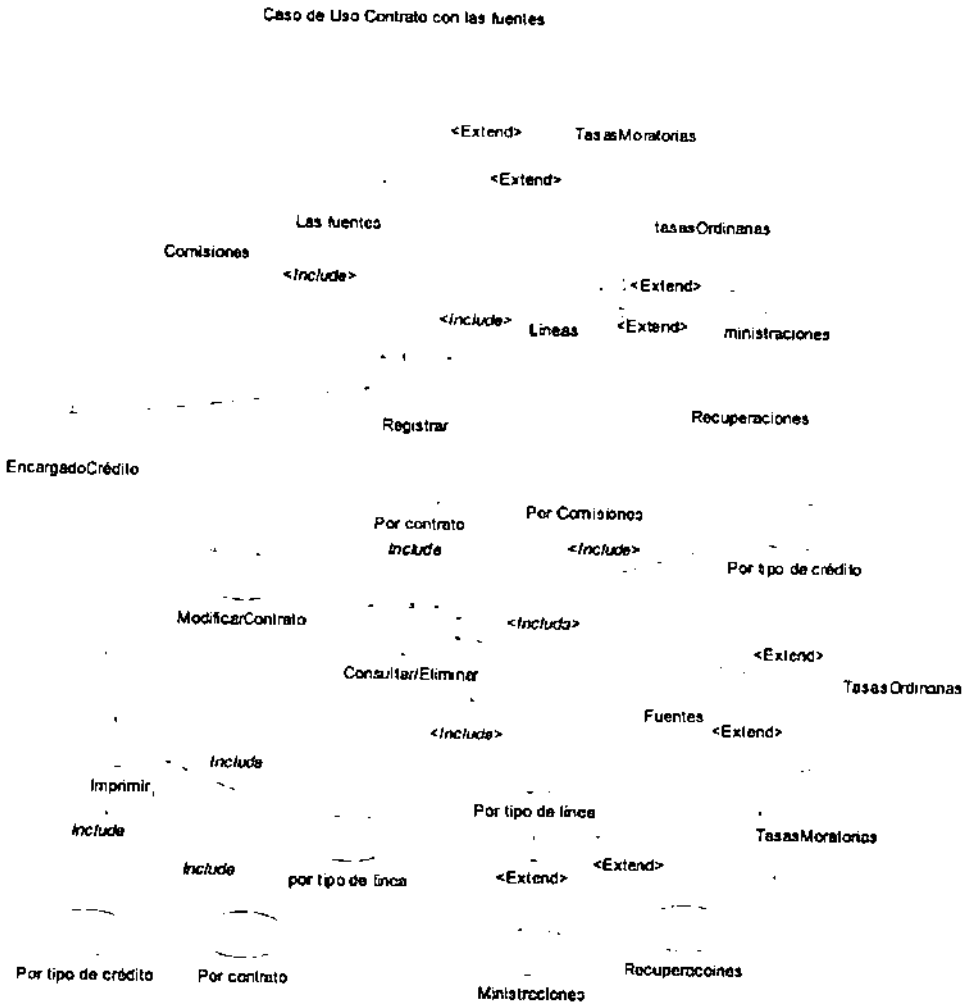
El glosario se definió de acuerdo a los términos que se utilizan o que van implícitos en el contrato con las fuentes financieras con el propósito de entender el modelo de casos de uso.

Glosario de términos del contrato con las fuentes financieraso

Contrato con las fuentes financieras	*Documento que ampara un crédito de la unión con la o las fuentes financieras; especifica las líneas, su monto, las ministraciones, recuperaciones y las tasas de interés que van a aplicar a la unión y ésta a los socios.*
Ciclos	*Período en que se puede sembrar o recoger la cosecha.*
Tipos de Crédito	*Tipos de préstamos que otorga la unión.*
Comisiones	*Tipos de comisión que le cobran a la unión por parte de las fuentes financieras y a su vez la unión a los socios al otorgarles el crédito.*
Recuperaciones	*Número de pagos que va a hacer la unión a la fuente.*
Ministraciones	*Forma en que le va a otorgar el dinero la fuente a la unión.*
Fuentes financieras	*Nombre de los bancos que le otorgan créditos a la unión.*
Tasas	*Interés que va a cobrar la fuente a la unión y ésta a su vez a los socios.*

Tabla 2.2 Glosario de términos

Posteriormente se presenta el modelo de casos de uso registrar contrato con las fuentes con todos los casos de uso identificados.



2.2.4. Priorizar Casos de Uso de acuerdo al Modelo de Casos de Uso

La vista del modelo de casos de uso incluye los casos de uso que describen algo importante y crítico para su funcionalidad o que complican algunos requerimientos importantes que pueden desarrollarse al inició del ciclo de vida del sistema, esta arquitectura también se usa cuando algunos casos de uso son prioritarios para el desarrollo de la iteración.

Identificar las partes funcionales de las descripciones

Para reducir la redundancia, esta parte es extraída y descrita en un caso de uso por separado que pueda ser reusado por un caso de uso original.

La generalización entre los casos de uso es un tipo de herencia, primero las instancias de los casos de uso generalizados pueden ejecutar todo el comportamiento descrito en la generalización de los casos de uso, en otras palabras, una generalización para un caso de uso A para un caso de

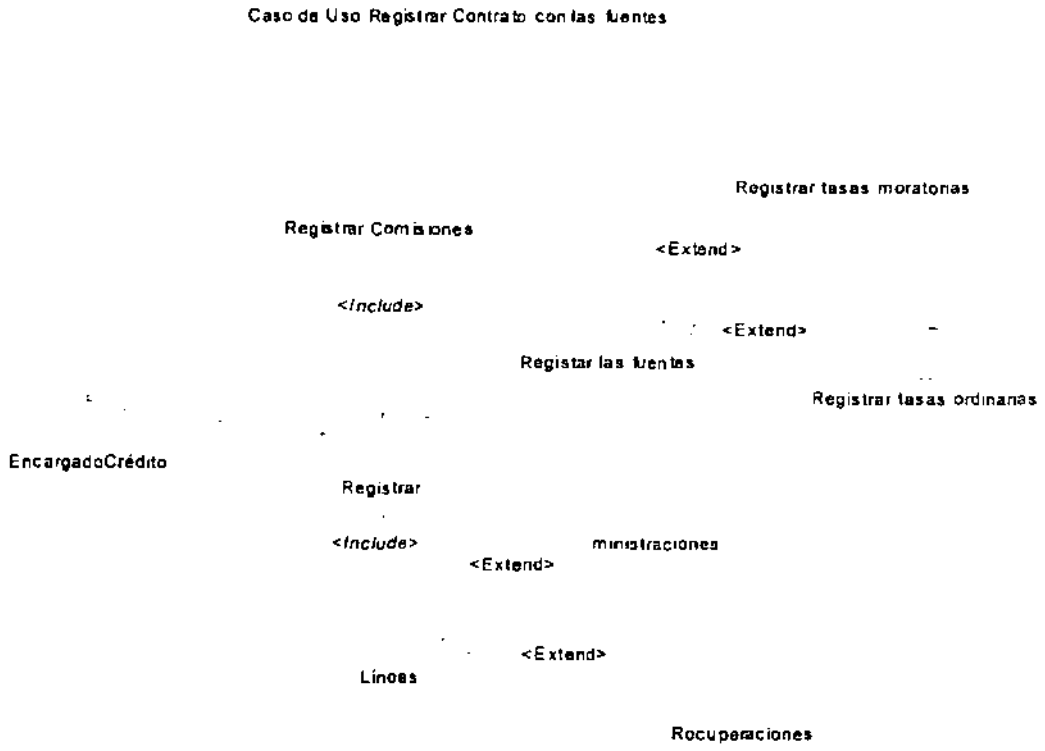
uso B indica que una instancia de un caso de uso A podría incluir el comportamiento especificado por B.

Un caso de uso concreto, inicia por un actor y sus instancias constituyen una secuencia completa de acciones ejecutadas por el sistema. Un caso de uso abstracto podría no ser instanciado por si mismo, pero una instancia de un caso de uso concreto exhibe el comportamiento especifico por el caso de uso abstracto que es de reuso. El propósito de comprenderlo es que los actores perciban cuando interactúan con el sistema.

Identificar las descripciones o funcionalidades adicionales u opcionales
Otra relación entre los casos de uso es la relación extendida. Al ampliar la relación para un caso de uso A a un caso de uso B indica que una instancia de un caso de uso B puede incluir el comportamiento específico por A. El comportamiento específico para algunas extensiones de un pequeño objeto de un caso de uso puede ocurrir con una instancia de un pequeño caso de uso; la relación de la extensión incluye ambas condiciones por la extensión y la referencia a un punto de la extensión en el objeto del caso de uso, esto es una posición en el caso de uso donde la adición puede ser hecha.

Cuando se empieza a modelar un nuevo sistema típicamente se trabaja por otros caminos, empezando por los casos de uso reales e identificando una parte del comportamiento, lo cual separa los casos de uso concretos de los casos de uso abstractos y el comportamiento adicional el cual trata extensiones de otros casos de uso.

Ejemplo
De acuerdo a lo anterior existe un caso de uso real que es registrar un contrato con las fuentes, de éste dependen los casos de uso abstractos o semifabricados que son Registrar los datos de las fuentes, Registrar las líneas del contrato con las fuentes y Registrar las comisiones del contrato con las fuentes. A su vez de los casos de uso abstractos dependen los casos de uso de extensiones que son Registrar las ministraciones, Registrar las recuperaciones y Registrar las tasas ordinarias por fuente.



Una vez identificado el caso real y sus casos de uso abstractos se deben priorizar y detallar cada uno.

2.2.5. Detallar Casos de Uso

El propósito de esta actividad es hacer la priorización de los casos de uso, determinar cuáles iteraciones se deben desarrollar y cuáles pueden desarrollarse posteriormente.

Ejemplo
De acuerdo al modelo de casos de uso, el caso de uso que tiene prioridad es registrar los datos generales del contrato con las fuentes, ya que sin los datos generales del contrato no se van a poder registrar los datos de las fuentes, las tasas, las líneas, las comisiones, las ministraciones y las recuperaciones, es decir los casos de uso semifabricados dependen de los datos generales del contrato.

Posteriormente del caso de uso registrar las líneas del contrato con las fuentes ván a depender los casos de uso registrar las ministraciones y recuperaciones; y del caso de uso registrar los datos de las fuentes van a depender los casos de uso registrar las tasas ordinarias por fuente y registrar las tasas moratorias por fuente.

El caso de uso registrar las comisiones del contrato con las fuentes, registrar las líneas del contrato con las fuentes y registrar las fuentes del contrato dependen directamente del caso de uso registrar los datos generales del contrato con las fuentes.

De acuerdo a lo anterior se les asignó el siguiente orden para su desarrollo:
1. *Registrar los datos generales del contrato con las fuentes.*
2. *Registrar las líneas del contrato con las fuentes.*
3. *Registrar las ministraciones .*

4. *Registrar las recuperaciones.*
5. *Registrar los datos de las fuentes.*
6. *Registrar las tasas ordinarias por fuente.*
7. *Registrar las tasas moratorias por fuente.*
8. *Registrar las comisiones del contrato con las fuentes*

El propósito de detallar cada caso de uso es para describir el flujo de eventos en detalle, incluyendo cómo empiezan los casos de uso, cómo terminan e iteractúan con los actores. Los casos de uso especifican detalladamente la descripción paso a paso cada secuencia de acciones.
Se deben considerar los siguientes pasos:

1. Cómo estructurar la descripción para especificar todas las alternativas o pasos de los casos de uso.
2. Qué incluir en una descripción de caso de uso.
3. Cómo formalizar la descripción del caso de uso cuando es necesario.

Para especificar un caso de uso se necesita trabajar con el usuario real del caso de uso y así discutir la propuesta y preguntar acerca de la descripción. El resultado de esta actividad es una descripción textual de un caso de uso particular en forma de texto y diagramas.

Qué incluye la descripción de los Casos de Uso

Para que los casos de uso sean entendidos por los desarrolladores, clientes y usuarios se recomienda usar un castellano común al describirlos.

1. La descripción de un caso de uso debe definir el estado inicial como una precondición.
2. Cómo y cuándo inician los casos de uso.
3. El orden requerido en la que las acciones se deben ejecutar. Aquí, el orden es definido por una secuencia ordenada.
4. Cómo y cuándo finalizan los casos de uso.
5. La descripción de los casos de uso define los posibles estados finales como postcondiciones.
6. La ruta de ejecución que no es permitida.
7. Una descripción de rutas alternas no está alineada en la descripción básica de la ruta.
8. Una descripción de rutas alternas debe ser extraída de la descripción de rutas básicas.
9. La interacción del sistema con los actores. En otras palabras, describir la secuencia de acciones del caso de uso, cómo estas acciones son invocadas por los actores y cómo se ejecuta el resultado en respuesta a los actores.
10. La interacción del sistema con los actores y que intercambian.
11. El uso de objetos, valores y recursos del sistema.
12. Describir explícitamente qué hace el sistema y qué hacen los actores.

Los atributos de un caso de uso pueden usarse para encontrar más tarde las clases y atributos en el análisis y diseño. También es importante especificar los requerimientos no funcionales que están relacionados a los casos de uso. Estos se pueden documentar en una sección por separado en la descripción del caso de uso.
Si el sistema interactúa con otro sistema, es necesario especificar estas interacciones en las primeras iteraciones.
La descripción de casos de uso termina cuando se estima que se comprendieron correctamente, completamente y consistentemente. Las descripciones son evaluadas por el analista así como por los usuarios y clientes. Únicamente los clientes y usuarios pueden verificar si los casos de uso son correctos.

Nota: El uso de estos diagramas en un contexto del caso de uso puede conducir a un diagrama muy grande y complejo que va a ser muy difícil de comprender, sin embargo, en muchos casos la descripción textual y los diagramas pueden complementarse uno con otro.

Ejemplo
En este punto se van a presentar tres ejemplos de casos de uso que pertenecen al caso de uso real registrar un contrato registrar un contrato con las fuentes.

Caso de Uso Registrar los datos generales del Contrato con las Fuentes

Encargado de Crédito

Registrar un contrato con las fuentes financieras

Precondición: El encargado no puede registrar un contrato si no están especificadas las líneas, las tasas de interés ordinarias y moratorias, las ministraciones y recuperaciones.

Flujo de sucesos

1. El encargado invoca el caso de uso para registrar el contrato. El sistema verifica los datos del contrato; núm. de contrato, tipo de crédito, monto del contrato, fecha de apertura, cuenta de cheques, fuentes, tasas de interés, líneas, ministraciones, recuperaciones y comisiones.
2. El encargado captura los datos generales del contrato. El sistema verifica que se hayan registrado todos los datos.
3. El sistema le asigna un número consecutivo de contrato y guarda el contrato.

Caminos alternativos:

En el paso 1, si ya se registro el contrato, el caso de uso cancelará el registro y enviará un mensaje de aviso al encargado.

En el paso 2, si no se registran todos los datos el caso de uso no registra el contrato y manda un aviso al encargado de lo que hace falta.

Postcondición:

La instancia del caso de uso termina cuando se hayan registrado todos los datos relacionados al contrato o cuando se haya cancelado la invocación del caso de uso.

Caso de Uso Registrar las Líneas del Contrato con las Fuentes

Encargado de Crédito — Registrar las Líneas del Contrato con las Fuentes

Precondición: El encargado no puede registrar las líneas si no se han registrado los datos generales del contrato.

Flujo de sucesos

1. El encargado selecciona la opción Líneas de la FormaContrato.
2. El sistema despliega la forma Líneas y solicita los datos de la línea: nombre de la línea, ciclo, vencimiento y monto.
3. El encargado captura los datos. El sistema verifica que no se repita la línea y que el monto no rebase el capital del contrato.

Caminos alternativos: En el paso 3

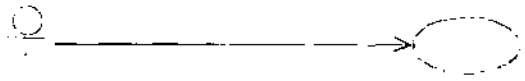
- si falta algún dato el caso de uso envía un mensaje al encargado dando aviso del dato que no se capturo
- si se repite el nombre de la línea debe cancelar la operación y dar aviso al encargado a través de un mensaje.
- si el monto de la línea rebasa el monto del contrato el caso de uso envía un mensaje al encargado y cancela la operación.

Postcondición:

La instancia del caso de uso se puede cancelar cuando el monto de las líneas sea menor o mayor al monto del contrato, por lo tanto, la suma del monto de las líneas debe ser igual al monto del contrato.

La instancia del caso de uso termina cuando se haya capturado correctamente los datos generales de La línea para dar paso a la captura de las ministraciones y recuperaciones por línea.

Caso de Uso Registrar los Datos de las Fuentes



Encargado de Crédito

Registrar los Datos de las fuentes

Precondición: El encargado no puede registrar los datos de las fuentes financieras si no se han registrado los datos generales del contrato.

Flujo de sucesos

1. El encargado selecciona la opción Fuentes de la FormaContrato.
2. El sistema despliega la forma Fuentes y solicita los datos de la fuente: nombre de la fuente, % de participación, monto, observaciones, tasas ordinarias y moratorias.
3. El encargado captura los datos de la fuente. El sistema calcula automáticamente el monto de la participación por fuente de acuerdo al monto total del contrato.

Caminos alternativos:

En el paso 3

- si el % de participación de la fuente es mayor al 100%, el caso de uso enviará un aviso al encargado y cancelará la operación que se está realizando en ese momento.
- si el nombre de la fuente se repite el caso de uso enviará un mensaje al encargado y dará la opción de volver a seleccionar la fuente.

Postocondición:

La instancia del caso de uso se puede cancelar cuando rebase el % de participación, cuando no se hayan dado de alta todas las fuentes que paticipan o cuando se repita el nombre de la fuente. La instancia del caso de uso termina cuando se hayan capturado correctamente todos los datos.

2.2.6. Prototipo de la Interfaz de Usuario

Durante la captura de los requerimientos la interfaz de usuario ayuda a especificar la interacción entre los actores humanos y el sistema para comprender los casos de uso.

El propósito de esta actividad es construir un prototipo de interfaz que permita al usuario ejecutar los casos de uso correctamente; primero se deben discernir los casos de uso de tal forma que se identifique qué se necesita de las interfaces de usuario para habilitar los casos de uso por cada actor. Esto es a través de un diseño lógico de la interfaz de usuario.

Posteriormente se crea el diseño físico de la interfaz de usuario y se desarrollan prototipos que ilustren cómo el usuario puede utilizar el sistema para ejecutar los casos de uso. Mediante la especificación de qué se necesita antes de decidir cómo realizar la interfaz de usuario, se llega a comprender las necesidades antes de intentar realizarlas.

Al finalizar esta actividad se debe contar con una serie de bosquejos y algunos prototipos que vean y perciban la especificación de la interfaz para los actores más importantes.

Crear un diseño lógico de la interfaz de usuario

Cuando los actores interactúan con el sistema, pueden usar y manipular los elementos de la interfaz que representan atributos (generalmente son términos del glosario de términos) de los casos de uso. Los elementos de la interfaz pueden ser iconos, listas y folders, pueden manipularlos por selección o arrastre. Se deben identificar y especificar los elementos por cada actor recorriendo todos los casos de uso a los que puede accesar. Un único elemento de la interfaz puede participar en muchos casos de uso jugando un rol diferente en cada uno. Así los elementos de la interfaz de usuario pueden diseñarse para jugar varios roles.

Las preguntas que se deben contestar por cada actor:

1. ¿Qué elementos de la interfaz son necesarios para permitir los casos de uso?
2. ¿Cómo se pueden relacionar uno con otro?
3. ¿Cómo se pueden usar en diferentes casos de uso?
4. ¿Cómo se pueden manipular?
5. ¿Cuál es su apariencia?

Para determinar qué elementos de la interfaz de usuario necesitan ser accesibles al actor en cada caso de uso se puede preguntar o cuestionar:

1. Qué clases del dominio, entidades de la empresa, o unidades de trabajo son apropiadas como elementos de la interfaz para cada caso de uso.
2. Con qué elementos de la interfaz va a trabajar el actor.
3. Qué acciones puede invocar el actor y qué decisiones puede hacer.
4. Qué información necesita el actor antes de invocar cada acción de los casos de uso.
5. Qué información debe proporcionar el actor al sistema.
6. Qué información debe proporcionar el sistema al actor.
7. Cuál es el promedio o parámetros de entradas y salidas del sistema.

Un camino práctico de trabajar es representar los elementos de la interfaz de usuario como una nota adhesiva y pegarlas en una pizarra y ordenarlas para ilustrar la apariencia de la interfaz. También se debe describir cómo los actores pueden usar estos elementos cuando trabajen con los casos de uso. La ventaja de usar estas notas es que pueden representar la cantidad datos necesarios.

De esta forma se asegura que cada caso de uso sea accesible y que tenga una interfaz de usuario bien integrada, fácil de usar y consistente.

Crear un diseño y un prototipo físico de la interfaz de usuario

- Primero se deben preparar unos esquemas aproximados de la configuración de los elementos de las interfaces de usuario.
- Después realizar un bosquejo de los elementos adicionales necesarios para combinar varios elementos de interfaces de usuario en interfaces de usuario completos por ejemplo, ventanas, herramientas, carpetas y controles.
- Finalmente es importante validar las interfaces de usuario a través de revisiones de prototipos y esquemas ya que pueden prevenir muchos errores que más adelante serán muy caros de corregir.

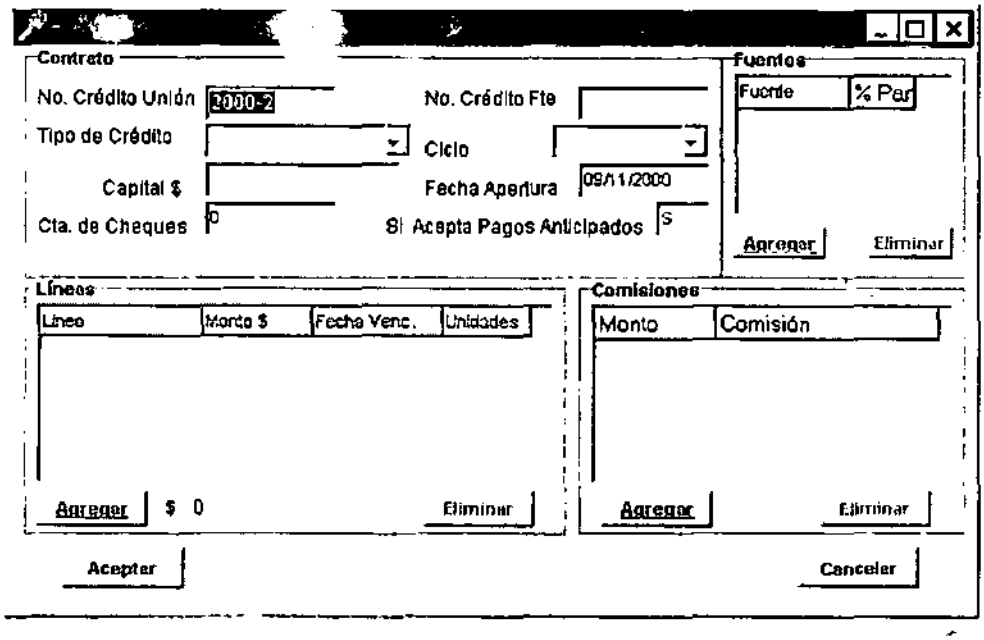
Por eso en cada interfaz se debe verificar que:

1. Permita que los actores naveguen apropiadamente.
2. Que tenga un ambiente agradable, que permita navegar de una forma sencilla y que sea consistente.
3. Que sea estándar en los colores, en el tamaño de los botones y en la colocación de las barras de herramientas.

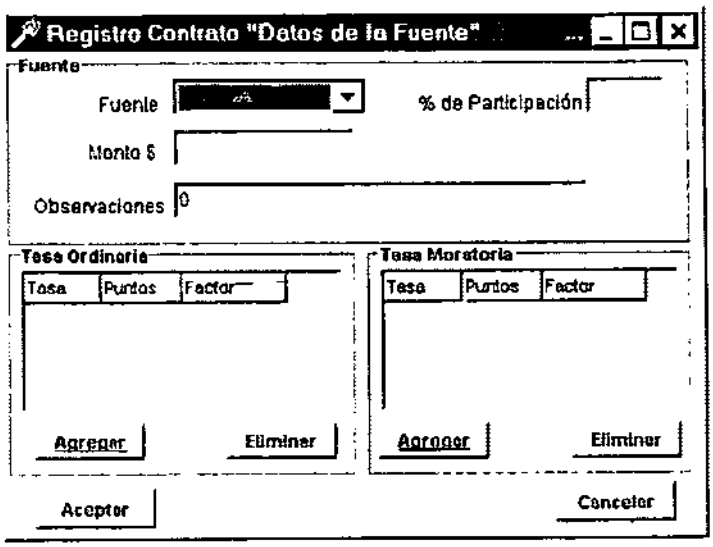
La implementación de la interfaz de usuario real es construida en paralelo con el resto del sistema, que es durante el análisis, diseño e implementación; pero lo que se ha desarrollado en este flujo de trabajo es una especificación de la interfaz de usuario.

Ejemplo
De acuerdo a las actividades anteriores, se diseño la interfaz de usuario para el caso de uso registrar los datos generales del contrato con las fuentes, el caso de uso registrar las líneas del contrato con las fuentes y el caso de uso registrar los datos de las fuentes.

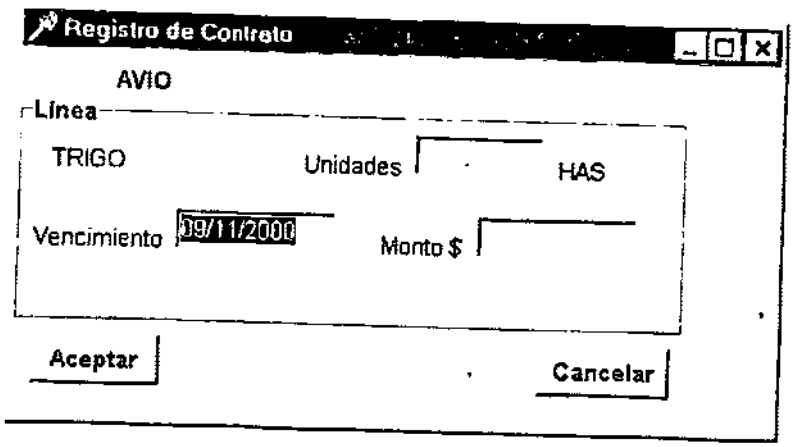
Caso de Uso Registrar los Datos Generales del Contrato con las Fuentes



Caso de Uso Registrar los datos de las fuentes



Caso de Uso Registrar las Líneas del Contrato con las Fuentes



Capitulo 3. Análisis

3.1. Introducción

En el análisis se especifican los requerimientos de una forma más precisa para refinarlos y estructurarlos. Su propósito principal es analizar los requerimientos con profundidad pero utilizando el lenguaje de los desarrolladores para describir el resultado.

Se estructuran de tal forma que permita comprenderlos, modificarlos, reusarlos y en general darles mantenimiento. La estructura está basada en las clases del análisis y los paquetes del análisis. Sirve como entrada fundamental para dar forma al sistema en su totalidad.

Es muy importante que los casos de uso:
1. Sean tan independientes uno de otro como sea posible,
2. Sean descritos usando el lenguaje del cliente y
3. Se estructuren de una forma completa especificando su funcionalidad.

El modelo del análisis organiza el sistema en piezas mas sencillas que van a representar abstracciones de subsistemas y de clases en el diseño del sistema. Los casos de uso se revisan para analizar las clases y sus objetos y desglosarlos en diagramas de colaboración.

3.2. Análisis de la Arquitectura

Durante la fase de elaboración las iteraciones iniciales se centran en el análisis y eso contribuye a obtener una arquitectura estable y sólida que facilita la comprensión de los requerimientos con mayor profundidad.

El propósito del análisis es esbozar el modelo del análisis mediante la identificación de paquetes del análisis, clases del análisis y requerimientos especiales comunes.

Actividades a Realizar
Las actividades principales que se deben realizar en el análisis son:
1. Identificar los paquetes de análisis y de servicio, sus dependencias y contenidos. (3.2.1.)
2. Analizar un caso de uso para identificar las clases del análisis y describir cómo los casos de uso son refinados en términos de colaboraciones de las clases. (3.2.2.)
3. Analizar las clases, sus responsabilidades, atributos, asociaciones, agregaciones y generalizaciones. (3.2.3.)

3.2.1. Paquetes del análisis y de servicio

Un paquete del análisis proporciona un medio para organizar el modelo del análisis en piezas más manejables, consiste de las clases del análisis, la realización de casos de uso y el análisis de

otros paquetes.

Sus características principales:
1. Deben estar fuertemente relacionados y sus dependencia uno de otro debe ser mínima.
2. Representan la separación de intereses de análisis de acuerdo a su importancia; por ejemplo en un sistema grande, el paquete del análisis puede analizarse por separado, posiblemente por diferentes desarrolladores y con conocimiento diferente del dominio.
3. Está basado en los requerimientos funcionales y en el dominio del problema pero no en los requerimientos no funcionales.
4. Es probable que los paquetes se conviertan en subsistemas en las dos capas de aplicación superiores en el modelo del diseño o se distribuyan entre ellos .

Propósitos:
1. Asegurar que sean independientes de otros paquetes como sea posible.
2. Asegurar que cumplan el propósito de realizar algunas clases o casos de uso.
3. Describir las dependencias que afectan cambios futuros que pueden estimarse.
4. Definir y mantener las dependencias con otros paquetes que contienen clases asociadas.
5. Asegurar que contengan las clases correctas.
6. Limitar las dependencias con otros paquetes.

Identificar los paquetes del análisis

Los paquetes del análisis proveen un camino para organizar el modelo del análisis por muy pequeño que sea en piezas más pequeñas. Una identificación inicial de los paquetes está basada en los requerimientos funcionales y en el dominio del problema. Una forma de identificarlos es asignar la mayor parte de casos de uso a un paquete en concreto y después revisar la funcionalidad que les corresponden dentro de ese paquete

Para realizar una distribución apropiada de casos de uso a un paquete se debe considerar lo siguiente:
1. Los casos de uso que se requieren para dar soporte a un proceso de negocios
2. Los casos de uso que se requieren para dar soporte a un actor en especifico del sistema
3. Los casos de uso que están relacionados mediante relaciones de generalización y de extensión.
4. Los casos de uso son coherentes en el sentido de que uno u otro se especializan o se amplían con otros.

Los paquetes localizan cambios en un proceso de negocio, en el comportamiento de un actor y en un conjunto de casos de uso relacionados. Esto también ayuda a distribuir o colocar los casos de uso en los paquetes. Pero se puede dar el caso de que algunos casos de uso no sean locales de un paquete sino que se cruzan entre ellos.

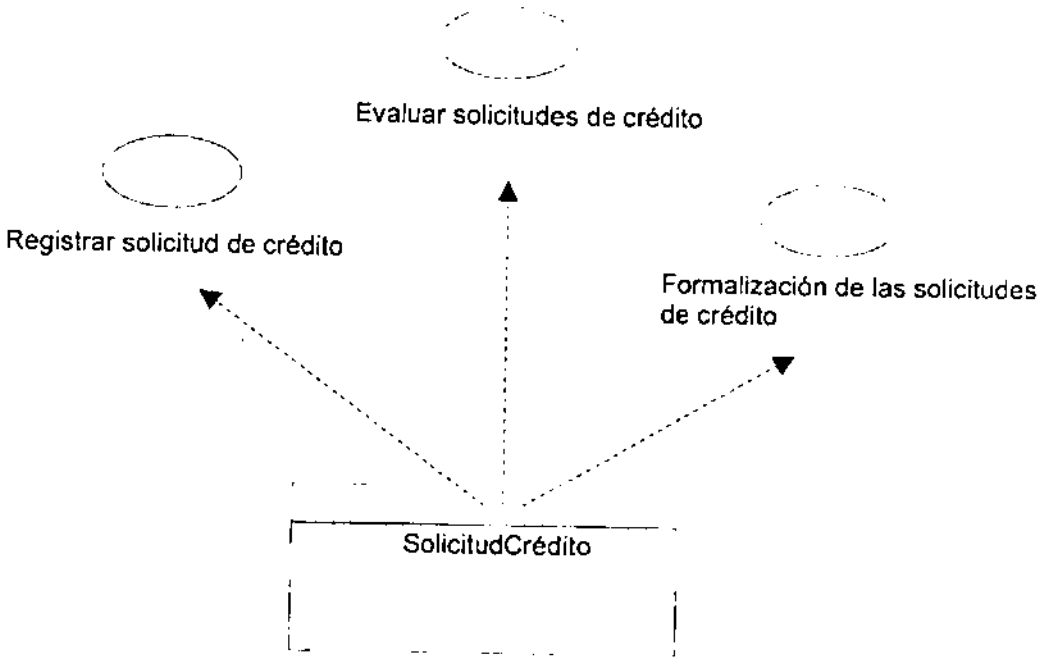
Ejemplo

Para poder ilustrar la identificación de los paquetes del análisis se tomó en cuenta todo el sistema por eso a continuación se muestra tres ejemplos de los paquetes que se identificaron como primer bosquejo ya que conforme se fue avanzando éstos se fueron refinando.

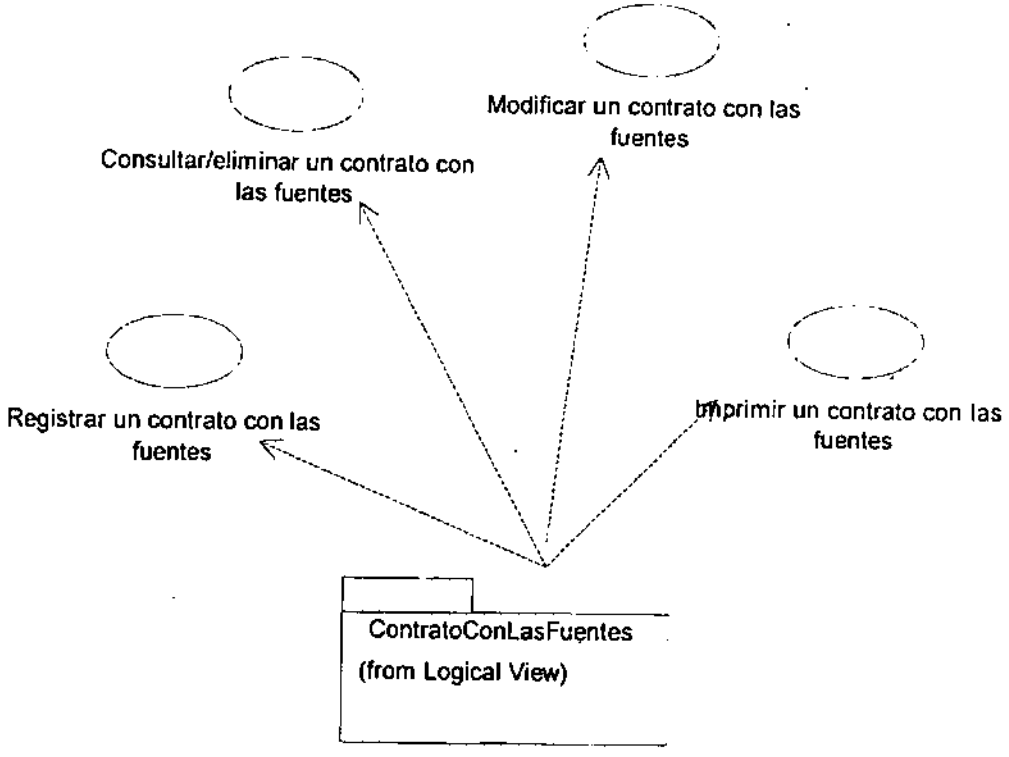
Los que recomiendan los autores[1] *es que se identifique a partir del modelo conceptual del negocio.*

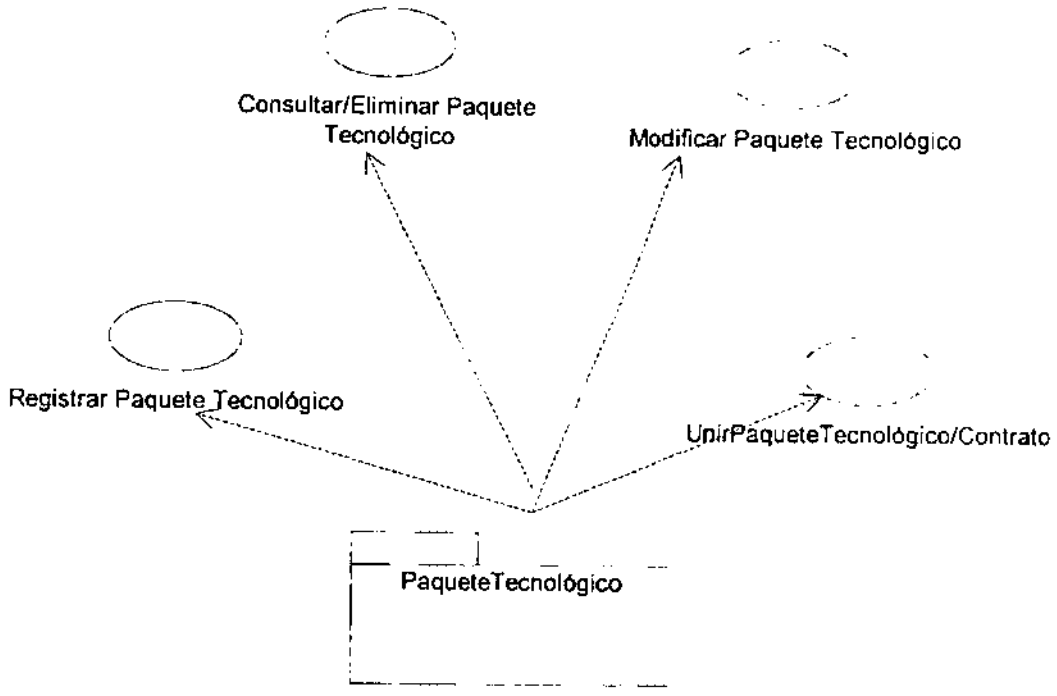
Los casos de uso registrar una solicitud de crédito, evaluar una solicitud de crédito, formalizar el crédito están todos implicados en el mismos proceso del negocio por eso pueden incluirse en un mismo paquete, el cual se le da el nombre de "SolicitudDeCcrédito" porque todos los casos de uso tienen relación con una solicitud de crédito, desde que se registra la solicitud de crédito hasta que es formalizada y se convierte en un crédito vigente.

[1] Jacobson, Booch y Rumbaugh



Otro paquete del análisis que también se identifico fue el paquete "ContratoConLasFuentes" el cual esta formado de los casos de uso registrar, consultar, modificar y eliminar un contrato con las fuentes.





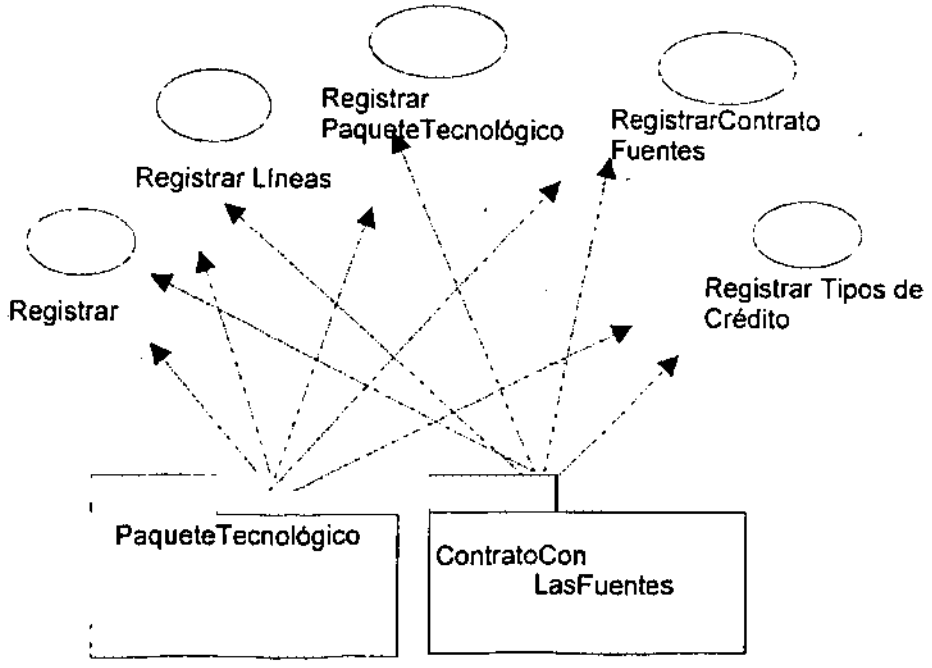
Identificar Paquetes del Análisis Comunes

Frecuentemente hay casos de uso comunes entre paquetes, es decir, cuando dos o más paquetes del análisis necesitan compartir las mismas clases del análisis. Un camino apropiado para manejarlo es extraer la clase compartida, colocarla fuera del paquete y hacer que otros paquetes sean dependientes de ese paquete o clase más general.

Ejemplo

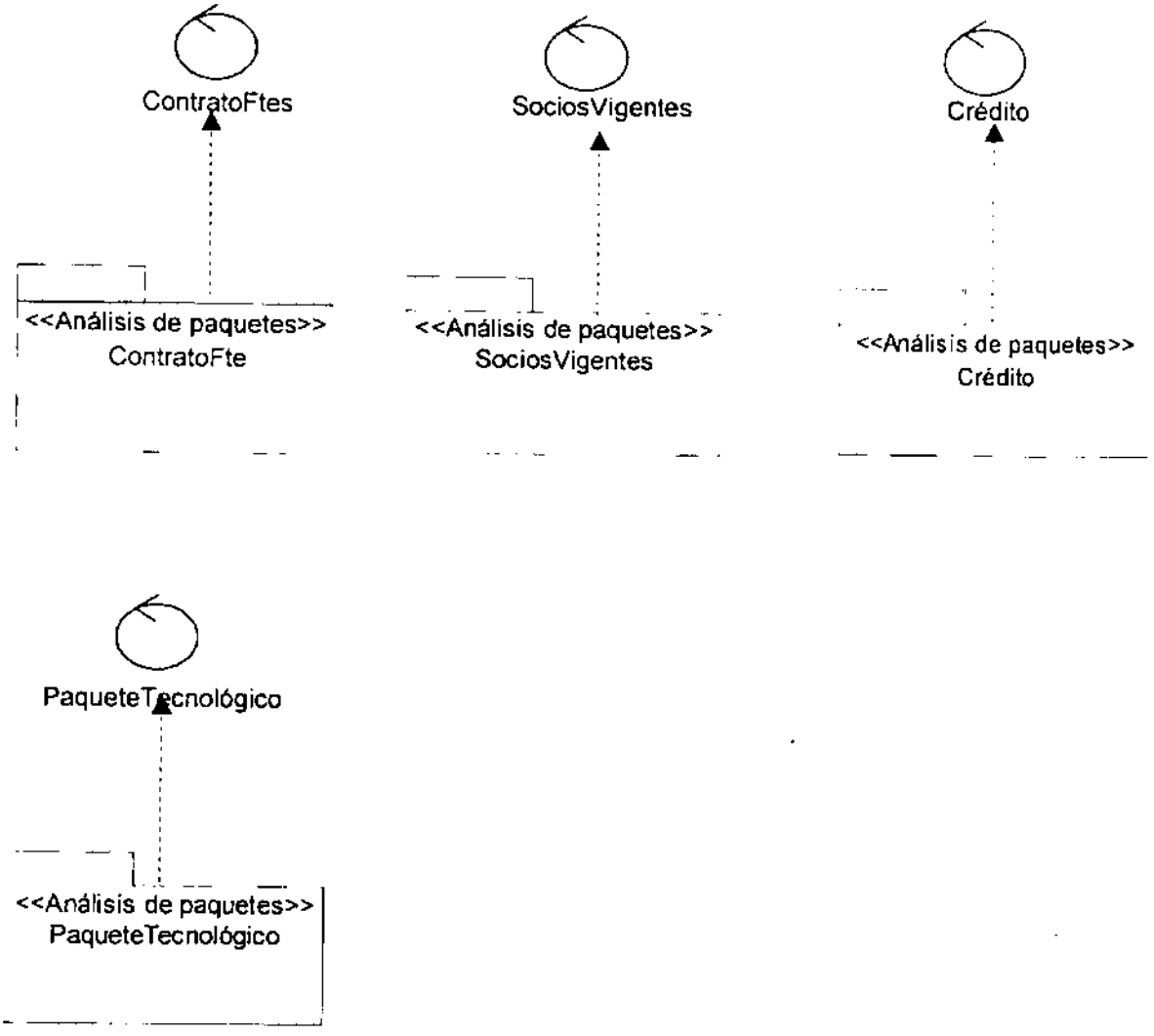
En el siguiente ejemplo se extrajeron las clases del análisis que requieren paquetes comunes y que no pueden pertenecer a uno en específico.

En los paquetes, Contrato con las fuentes y paquete tecnológico existe una dependencia de las clases: ciclos, líneas, créditos, ContratoFte y PaqueteTecnológico.



Identificar Paquetes Generales del Análisis

De acuerdo al modelo conceptual de crédito se identificaron los paquetes generales del análisis donde cada clase representa información importante del sistema y que está compartida por otros paquetes más específicos, en consecuencia se crea un paquete por cada clase



Definir las Dependencias entre Paquetes del Análisis

Se deben definir las dependencias entre los paquetes del análisis si su contenido tiene relación con cada uno y la dirección de las dependencias debe ser la misma de la relación.

Si se identifican los paquetes de una forma independiente, va a permitir que sea más sencillo realizar cambios a las clases que pertenecen a ese paquete sin afectar demasiado a otros, por eso es recomendable reducir el número de relaciones entre paquetes.

Para clarificar las dependencias, es útil tener las capas de aplicación específica y de aplicación general en el modelo del análisis; la capa más alta está formada de paquetes de aplicación específica y la capa más baja de paquetes generales.

Durante el diseño y la implementación se pueden refinar estas capas y añadir en el nivel más bajo

para un ambiente de la implementación.

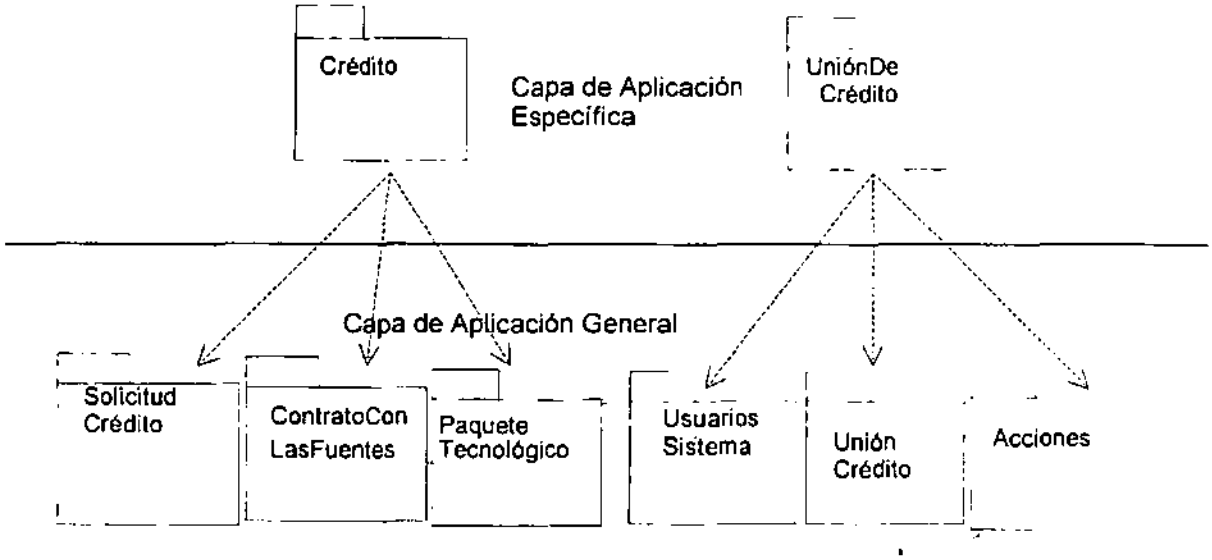
Ejemplo
En el siguiente ejemplo se puede observar la identificación de los paquetes de la capa de aplicación específica y de la capa de aplicación general. Para identificar los paquetes se tomaron como base los paquetes generales y los paquete del análisis comunes agrupándolos de acuerdo a su contenido y relación que puede haber entre ellos.

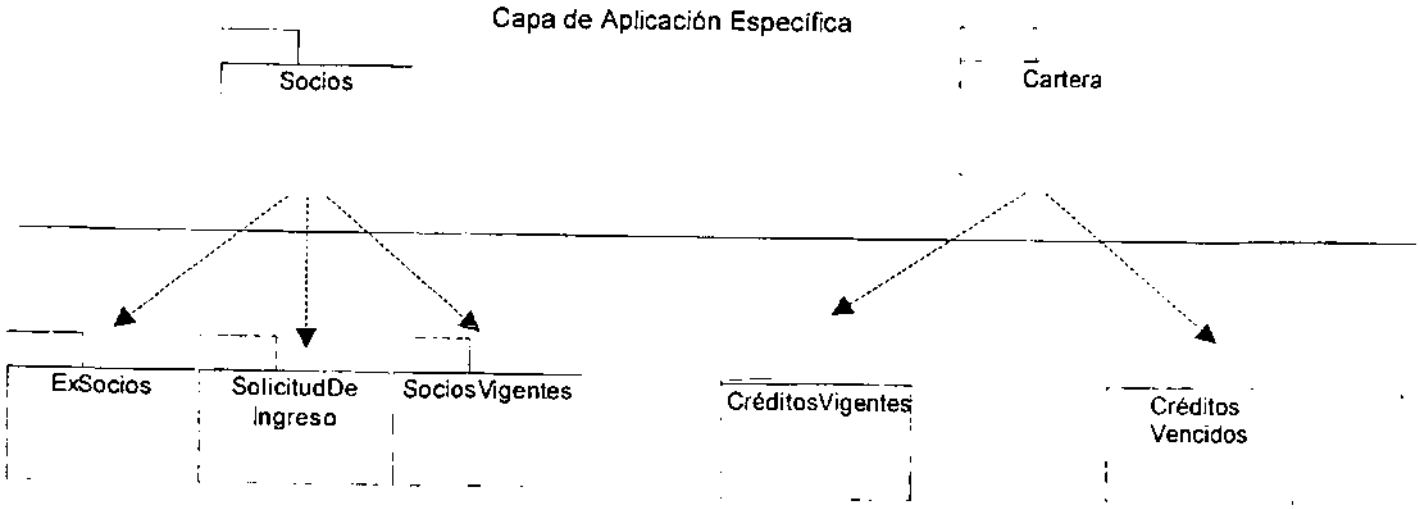
- *Los paquetes generales solicitud de ingreso, socios vigentes y exsocios forman parte de la capa de aplicación general y éstos se agruparon para integrar el paquete Socios de la capa de aplicación específica.*

- *Los paquetes generales acciones, usuarios del sistema y unión de crédito forman parte de la capa de aplicación general y agrupados integran el paquete unión de crédito de la capa de aplicación específica.*

- *Los paquetes generales créditos vigentes y cartera vencida forman parte de la capa de aplicación general y agrupados integran el paquete cartera de la capa de aplicación específica. y*

- *Los paquetes generales solicitud de crédito, contrato con las fuentes y paquete tecnológico forman parte de la capa de aplicación general e integrados forman el paquete crédito de la capa de aplicación específica.*





Paquetes de Servicio

Un servicio representa un conjunto de acciones relacionadas funcionalmente que se utilizan en varios casos de uso. Un servicio es indivisible en el sentido de que el sistema necesita ofrecerlo todo o nada. Los casos de uso son para los usuarios y los servicios para los clientes.

Un paquete de servicio:

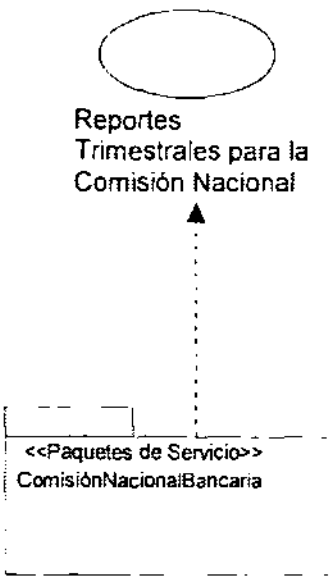
1. Contiene un conjunto de clases relacionadas.
2. Es indivisible.
3. Más paquetes pueden participar en la creación de un caso de uso, sin embargo, es común especificar cuándo participan en la realización de diferentes casos de uso.
4. Generalmente depende de otro paquete de servicio.
5. Normalmente es relevante a uno o pocos actores.
6. Puede ser mutuamente excluyente o puede representar diferentes aspectos o variantes del mismo servicio.

Identificar los Paquetes de Servicio

El servicio esta soportado por los paquetes de servicio, generalmente se identifican cuando se hayan comprendido muy bien los requerimientos funcionales y cuando ya se hayan definido la mayoría de las clases del análisis, pero se debe identificar un paquete de servicio por cada servicio opcional o que podría ser opcional.

Ejemplo

En el ejemplo del Sistema Integral de Uniones de Crédito del Sector Social lo que no hace el sistema y que desea el usuario son los reportes a la comisión nacional bancaria.



3.2.2. Analizar un Caso de Uso

Analizar un caso de uso es para:

1. Identificar las clases del análisis.
2. Distribuir el comportamiento del caso de uso entre objetos que interactúan.
3. Capturar requerimientos especiales sobre la realización del caso de uso.

Identificar las Clases del Análisis

Uno de los puntos importantes del modelo del análisis es encontrar las clases u objetos del análisis.

En las clases del análisis hay estereotipos de clases que representan el modelo conceptual de elementos del sistema que tienen una responsabilidad y un comportamiento; aunque también pueden representar una abstracción de una o algunas clases y/o subsistemas del diseño del sistema.

Características principales:

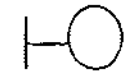
1. Manejan los requerimientos funcionales y posponen los no funcionales hasta el diseño e implementación.
2. Su comportamiento está definido por responsabilidades en un nivel menos formal.
3. Definen atributos. Los atributos son conceptuales y reconocibles en el domino del problema puesto que en el diseño e implementación son tipos de lenguajes de programación.
4. Participan en las relaciones, Por ejemplo la navegación de las asociaciones no es muy importante en el análisis pero es esencial en el diseño o la generalización puede usarse en el análisis pero no es posible usarla en el diseño si no está soportada por el lenguaje de programación.
5. Siempre deben ser apropiadas a uno de los tres estereotipos: interfaz, control o entidad.

Clases de Interfaz

Una clase interfaz modela la interacción entre el sistema y sus actores (por ejemplo usuarios y sistemas externos). La interacción frecuentemente implica la recepción y presentación de información, preguntas a los usuarios y sistemas externos, dependen de sus actores implicando

que ellos aclaren y reúnan los requerimientos de los límites del sistema. Por tanto, un cambio en la interfaz de usuario o en la interfaz de comunicación generalmente se debe aislar en una o más clases de interfaz.

Las clases de interfaz frecuentemente representan abstracciones de ventanas, formas, paneles, interfaces de comunicación, interfaces de impresión, sensores, terminales y APIs; pero no son descritas como la interacción física ya que se consideran en el diseño y la implementación de las actividades. Cada clase interfaz debe relacionarse al menos con un actor y viceversa. Se representan de la siguiente forma:



Clase Frontera o límite

Clases Entidad

Una clase entidad se usa para modelar información que posee una vida larga y debe ser persistente. Modelan la información y el comportamiento de algunos fenómenos o conceptos como una persona, un objeto o evento de la vida real.

En muchos casos, una clase entidad se deriva directamente de una clase entidad del negocio (o clases del dominio) tomada del modelo de objetos del negocio (o modelo del dominio). Sin embargo, una diferencia mayor entre las clases entidad y las clases entidad del negocio es que las clases entidad representan el manejo de objetos por el sistema y las clases entidad del negocio representan objetos presentes en el negocio (y en el dominio del problema) . Como resultado las clases entidad reflejan la información en un camino que beneficie a los desarrolladores cuando diseñan e implementan el sistema y las clases entidad del negocio (o domino de clases) describen el contexto del sistema y pueden incluir información que no es manejada por el sistema.

Una clase entidad frecuentemente muestran una estructura de datos lógica y contribuye a comprender de qué información depende el sistema. Se representan de la siguiente forma:

Clases entidad

Clases de Control

Las clases de control representan coordinación, secuencia, transacción y control de otros objetos; frecuentemente se usan para encapsular el control de un caso de uso en específico, también representan derivaciones y cálculos complejos.

Lo dinámico del sistema se modela por éstas clases debido a que manejan y coordinan las acciones y los flujos del control y posteriormente delegan el trabajo a otros objetos (por ejemplo objetos de interfaz y de entidad).

No encapsulan el resultado relacionado a la interacciones, con los actores, tampoco información de larga vida; ésta la encapsulan las clases de interfaz y de entidad. En cambio las clases de control encapsulan y aíslan, los cambios de control, la coordinación, la secuencia, las transacciones y algunas veces la lógica del negocio compleja que implica a varios objetos.

Clase de control

Para poder identificar las clases del análisis, los autores recomiendan empezar por la identificación de las clases de control, entidad e interfaz que se necesitan para realizar el caso de uso.

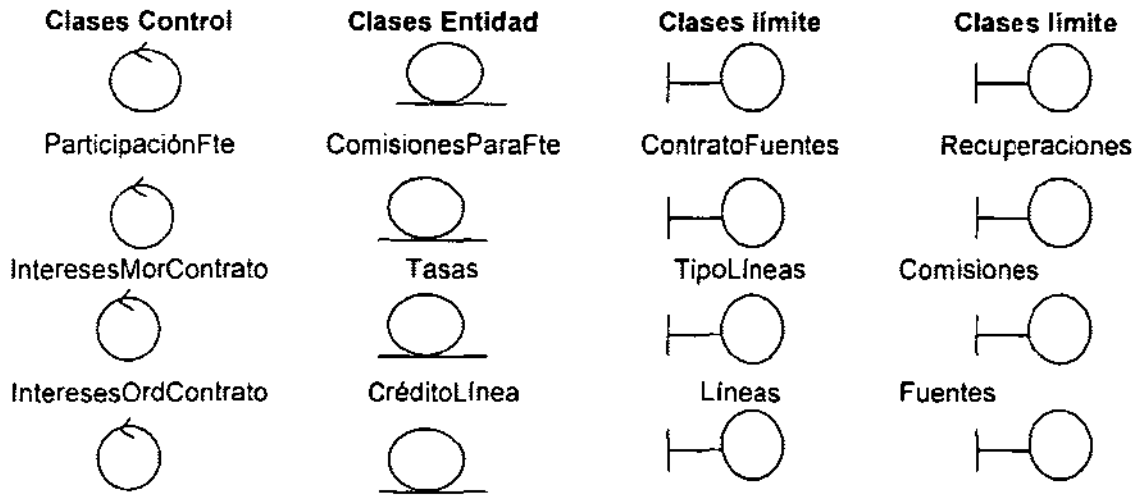
Los casos de uso descritos en los requerimientos, no siempre detallan lo suficiente para identificar las clases del análisis, por eso para identificarlas se deben refinar las descripciones de los casos de uso con respecto al sistema, este refinamiento es capturado en la descripción textual del flujo de eventos .

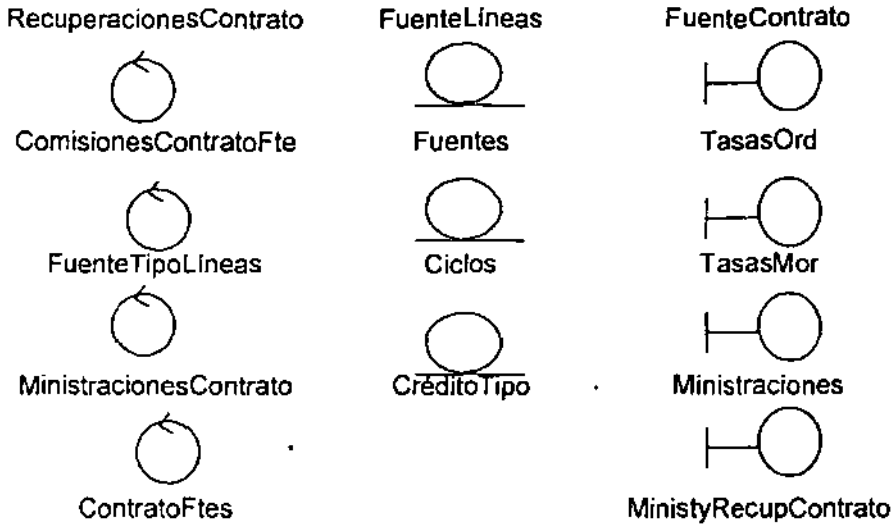
Para identificar las clases del análisis:

1. Primero se deben identificar las clases entidad, estudiando la descripción de los casos de uso y del modelo del dominio, considerando qué información se necesita utilizar y manipular en la realización de los casos de uso.
2. Identificar una clase interfaz central por cada actor humano y que esta clase represente la ventana principal de la interfaz con la que interactúa el actor. Si el actor ya interactúa con una clase interfaz, se debería reutilizar para conseguir una buena facilidad de uso de la interfaz y así minimizar el número de ventanas principales que cada actor necesita.
3. Identificar una clase de interfaz primitiva por cada clase entidad. Esta clase representa un objeto lógico que él (humano) actor interactúa con la interfaz de usuario durante el caso de uso. Puede refinarse de acuerdo a varios criterios de uso para contribuir a la creación de una buena interfaz de usuario.
4. Identificar una clase de interfaz central por cada actor que sea un sistema externo y que esta clase represente la interfaz de comunicación.
5. Identificar una clase de control responsable de manejar el control y la coordinación de la realización de los casos de uso y después refinarla acorde a los requerimientos de los casos de uso, por ejemplo, en algunos casos es mejor encapsular el control dentro de la clase de interfaz, especialmente si el actor maneja parte del control; en estos casos la clase de control no es necesaria. En otros casos el control es tan complejo que es mejor encapsularlo en dos o más clases de control y en algunos casos necesitan dividirse.

Ejemplo
Para poder ejemplificar los puntos anteriores se consideraron los casos de uso registrar el contrato con las fuentes y registrar un paquete tecnológico.
Primero se identificaron las clases de entidad, posteriormente las clases de control y por último las clases de interfaz o límite.

Caso de Uso Registrar Contrato con las Fuentes





Diagramas de colaboración
La secuencia de acciones en un caso de uso empieza cuando un actor invoca los casos de uso mediante el envío de mensajes al sistema. En el análisis se prefiere describir con los diagramas de colaboración que contienen las instancias de los actores que participan, los objetos del análisis y sus enlaces.

Los diagramas de colaboración de la realización de un caso de uso son difíciles de leer por si mismos, pero se les puede adicionar texto que lo explique. El texto puede ser escrito en términos de objetos de control que interactúan para ejecutar los casos de uso, sin embargo, la mención de algún atributo de los objetos, responsabilidades y asociaciones pueden ser difíciles de mantener porque cambian frecuentemente.

Respecto a la creación y terminación de los objetos del análisis en la realización de un caso de uso, tienen diferentes ciclos de vida:
1. Un objeto de interfaz no es particular de la realización de un caso de uso, por ejemplo, debe aparecer en una ventana y participar en dos o más instancias de casos de uso. Sin embargo, los objetos de interfaz frecuentemente son creados y terminados al mismo tiempo con la realización de un pequeño caso de uso.
2. Un objeto de entidad frecuentemente no es particular de un caso de uso, ya que tiene una vida larga y participa en la realización de varios casos de uso antes de su destrucción.
3. Las clases de control frecuentemente encapsulan el control asociado a un caso de uso en especifico, implicando que un objeto de control sea creado cuando empieza el caso de uso se destruya cuando finalice el caso de uso. Sin embargo, hay excepciones cuando un objeto de control participa en la realización de mas de un caso de uso, cuando varios objetos de control (o diferentes clases de control) participan en la misma realización de un caso de uso y cuando una realización de caso de uso no requiere ningún objeto de control.

Un diagrama de colaboración se crea comenzando por el inicio del caso de uso y continua paso a paso indicando qué interacciones de objetos del análisis y de instancias de actor son necesarias para realizarlo. En un diagrama de colaboración se puede observar lo siguiente:
1. El caso de uso es invocado por un mensaje proveniente de una instancia de un actor sobre un objeto de interfaz.
2. Cada clase del análisis tiene al menos un objeto del análisis que participa en un diagrama de colaboración; es superflua si ésta no participa en la realización de algún caso de uso.
3. Los mensajes no son asociados a las operaciones ya que no especifican operaciones para las clases del análisis. En su lugar, un mensaje denota el intento de invocar objetos cuando

interactúan con el objeto invocado.

4. Los enlaces en los diagramas frecuentemente son instancias de asociaciones entre clases del análisis y definen requerimientos sobre las asociaciones. Todas las asociaciones se describen en el diagrama clases asociadas con la realización del caso de uso.
5. La secuencia en el diagrama no es un objetivo principal, puede ser excluido si el diagrama es difícil de mantener o es muy confuso. En su lugar, la relación (ligas) entre los objetos y en los requerimientos (capturados en mensajes) en cada objeto particular puede ser el objetivo principal.
6. Los diagramas de colaboración deben tratar todas las relaciones de los casos de uso que se están realizando.

Distribuir el comportamiento del caso de uso entre objetos que interactúan

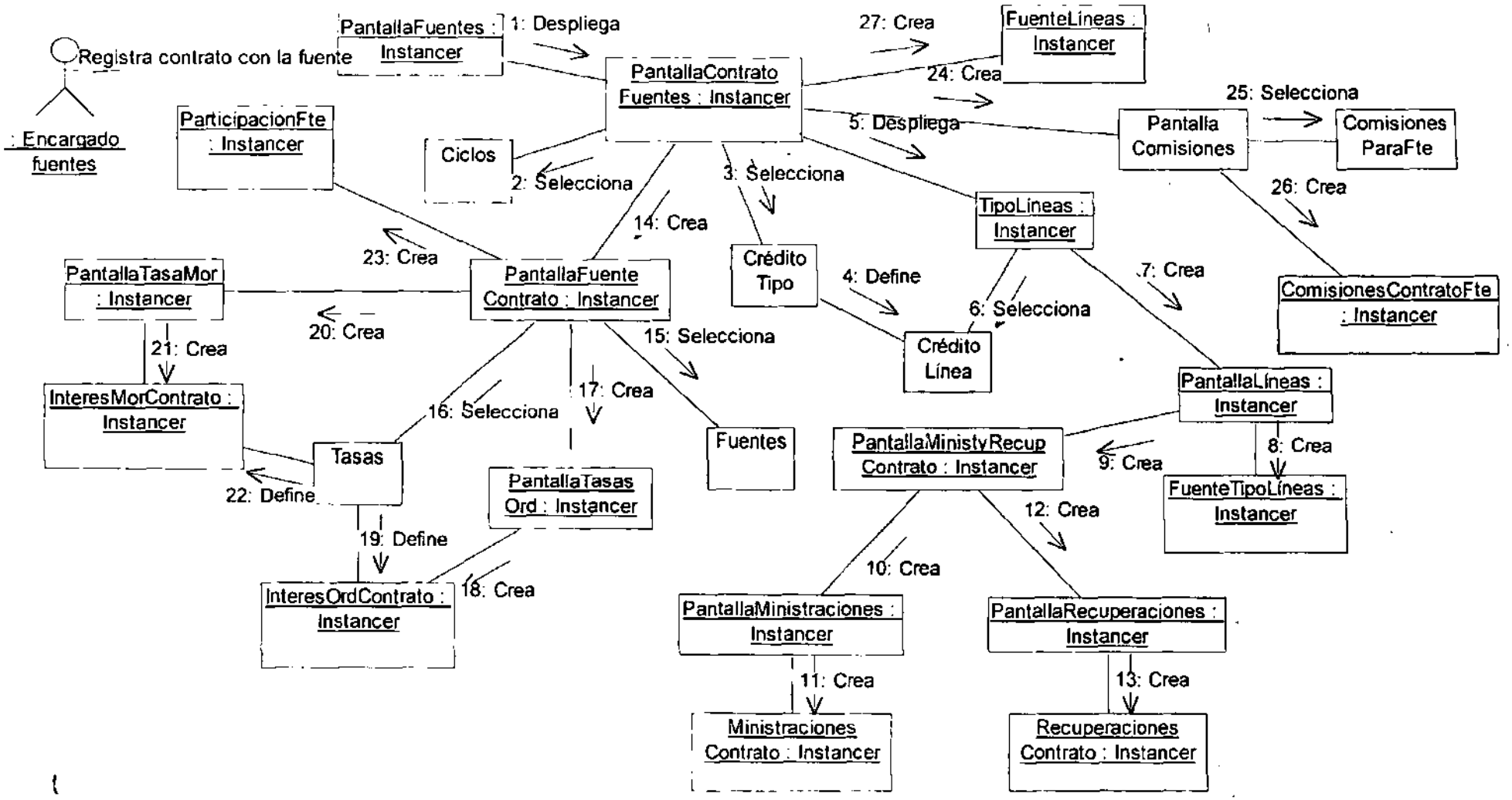
La realización de los casos de uso describen cómo un caso de uso se realiza y ejecuta en términos de clases del análisis y su interacción con los objetos del análisis.

Una realización de un caso de uso tiene una descripción textual del flujo de eventos, los diagramas de clases describen la participación de las clases del análisis y el diagrama de interacción describe la realización de un flujo particular del caso de uso en términos de interacción de objeto del análisis.

Ejemplo

A continuación se presenta el diagrama de colaboración del caso de uso registrar un contrato con las fuentes. En él se puede observar como interactúan las clases de control, de interfaz y de entidad.

Diagrama de Colaboración del Caso de Uso Registrar un Contrato con las Fuentes



En el siguiente cuadro se explica el flujo textual de un diagrama de colaboración, es decir cómo interactúan los objetos entre si para poder registrar un contrato con las fuentes.

0. El mensaje "Registrar un Contrato con la Fuente" se envía a una instancia de tipo PantallaCrédito
1. El objeto PantallaFuentes crea la instancia PantallaContratoFuentes.
2. El objeto PantallaContratoFuentes selecciona la clase ciclos.
3. El objeto PantallaContratoFuentes selecciona la clase crédito tipo.
4. El objeto crédito tipo define la clase crédito linea.
5. El objeto PantallaContratoFuentes crea la instancia PantallaTipoLíneas.
6. El objeto PantallaTipoLineas selecciona la clase Crédito Línea.
7. El objeto PantallaTipoLíneas crea la instancia PantallaLineas.
8. El objeto PantallaLíneas crea la instancia FuenteTipoLineas.
9. El objeto PantallaLineas crea la instancia PantallaMinisyRecupContratoFte.
10. El objeto PantallaMinisyRecupContratoFte crea la instancia PantallaMinistraciones.
11. El objeto PantallaMininistraciones crea la instancia MinistracionesContrato.
12. El objeto PantallaMinisyRecupContratoFte crea la instancia PantallaRecuperaciones.
13. El objeto PantallaRecuperaciones crea la instancia RecuperacionesContrato.
14. El objeto PantallaContratoFuentes crea la instancia PantallaFuentes.
15. El objeto PantallaFuentes selecciona la clase fuentes.
16. El objeto PantallaFuentes selecciona la clase tasas.
17. El objeto PantallaFuentes crea la instancia PantallaTasasord.
18. El objeto Tasas define las tasas ordinarias.
19. El objeto PantallaTasasord crea la instancia interesordcontrato.
20. El objeto PantallaFuentes crea la instancia PantallaTasasMor.
21. El objeto Tasas define las tasas moratorias.
22. El objeto PantallaTasasMor crea la instancia
23. El objeto PantallaFuentes crea la instancia participacionFte.
24. El objeto PantallaContratoFuentes crea la instancia PantallaComisiones.
25. El objeto PantallaComisiones selecciona la clase ComisionesParaFte.
26. El objeto PantallaComisiones crea la instancia ComisionesContratoFte.

3.2.3. Analizar una Clase

El propósito de analizar una clase es para:
1. Identificar y mantener las responsabilidades de una clase del análisis basándose en la realización de un caso de uso.
2. Identificar y mantener los atributos y relaciones de las clases.
3. Capturar los requerimientos especiales en la realización de las clases.

Identificar Atributos
Un atributo especifica una propiedad de una clase, algunas veces es necesaria y requerida por la responsabilidad de la clase.

Cuando se identifican los atributos se debe tener èn cuenta que:
1. El nombre del atributo sea un nombre.
2. Los tipos de atributos sean conceptuales en el análisis y no se restrinjan por el ambiente de implementación.
3. Cuando se decida un tipo de atributo se procure reusar los existentes.
4. Una pequeña instancia de un atributo no pueda compartirse por varios objetos del análisis. Si esto se requiere, los atributos se definan en su propia clase.
5. Si una clase se ha complicado al definir los atributos, éstos se puedan separar en clases independientes.
6. Los atributos de las clases interfaz que interactúan con los actores y representan sistemas, generalmente sean propiedades de una interfaz de comunicación.
7. Los atributos de las clases interfaz que interactúen con los actores generalmente representen información manipulada por los actores.
8. Los atributos de las clases de control no son muy frecuentes debido a su corto tiempo de vida, sin embargo puedan poseer atributos que representen valores acumulados o calculados durante la realización de un caso de uso.
9. Algunas veces los atributos formales no son necesarios. En su lugar, una explicación simple de una propiedad tratada por una clase del análisis sea suficiente y pueda agregarse a la descripción de las responsabilidades de la clase.
10. Si la clase tiene muchos atributos complejos, que se puedan mostrar en un diagrama de clases por separado donde se pueda ver únicamente el comportamiento de los atributos.

Identificar Responsabilidades
Las responsabilidades de una clase pueden recopilarse combinando todos los roles que éstas juegan en la realización de diferentes casos de uso. También se pueden identificar mediante los diagramas y su interacción.

Ejemplo
Roles de la clase ContratoFtes
- *El objeto contrato con las fuentes se crea cuando se va a registrar el contrato. El encargado de crédito lleva a cabo este caso de uso para registrar un contrato con las fuentes.*
- *Si el objeto contratoftes tiene recursos se verifica que exista un objeto PaqueteTecnológico y que coincidan sus líneas con el objeto contratoftes. Si las líneas del objeto paquete y del objeto contrato coinciden se une el objeto PaqueteTecnológico con el objeto contratoftes con las fuentes.*
- *Si se desea crear un objeto SolicitudCrédito debe verificar si el objeto contratoftes tiene recursos disponibles para el objeto solicitudcrédito.*

Responsabilidades de la clase ContratoFtes
El objeto ContratoFtes tiene las siguientes responsabilidades:
- *Verificar que no se repita el número de contrato.*

- *Que el monto total del contrato coincida con el monto total de las líneas.*
- *Que el tipo de crédito del contrato coincida con el del paquete tecnológico.*
- *Que las líneas y el ciclo coincidan con el paquete tecnológico.*
- *Que el porcentaje de participación de las fuentes sea igual al 100%.*
- *Que se registren las tasas ordinarias y moratorias por cada fuente.*
- *Que se registren las ministraciones y recuperaciones.*
- *Verificar que el contrato se una únicamente con un paquete.*
- *Verificar que no se elimine el contrato cuando ya se haya unido con el paquete.*

Identificar Asociaciones y Agregaciones

Los objetos del análisis interactúan unos con otros a través de enlaces en los diagramas de colaboración. Estos enlaces son instancias de asociaciones entre las clases correspondientes. Se deben estudiar los enlaces de los diagramas de colaboración para determinar las asociaciones, referencias y agregaciones entre los objetos que se pueden necesitar.

El número de relaciones entre las clases debe ser mínimo. Las relaciones que deben existir son en respuesta a las demandas de las diferentes realizaciones de caso de uso. Se deben definir la multiplicidad de las asociaciones, los nombres de los roles, autoasociaciones clases de asociación, roles ordenados, roles cualificados y asociaciones n-arias.

Las agregaciones deben usarse cuando los objetos representan:

1. Conceptos que se contienen uno al otro, como un carro que contiene el chofer y los pasajeros
2. Conceptos que son compuestos uno de otro, como cuando un carro consiste de una máquina y un par de ruedas.
3. Conceptos que forma una colección conceptual de objetos, como una familia que consiste de madre, padre e hijos.

Ejemplo

La clase contratoftes participa en el caso de uso unir un paquete tecnológico, siempre está asociado, es uno a uno, se representa mediante una asociación "1..1".

La clase contratoftes está asociado a una o varias solicitudes de crédito.

La clase contratoftes siempre está asociado a una o varias fuentes . Esto se representa mediante una asociación con la multiplicad "1.." (siempre hay al menos una fuente asociada a un contrato).

Identificar Generalizaciones

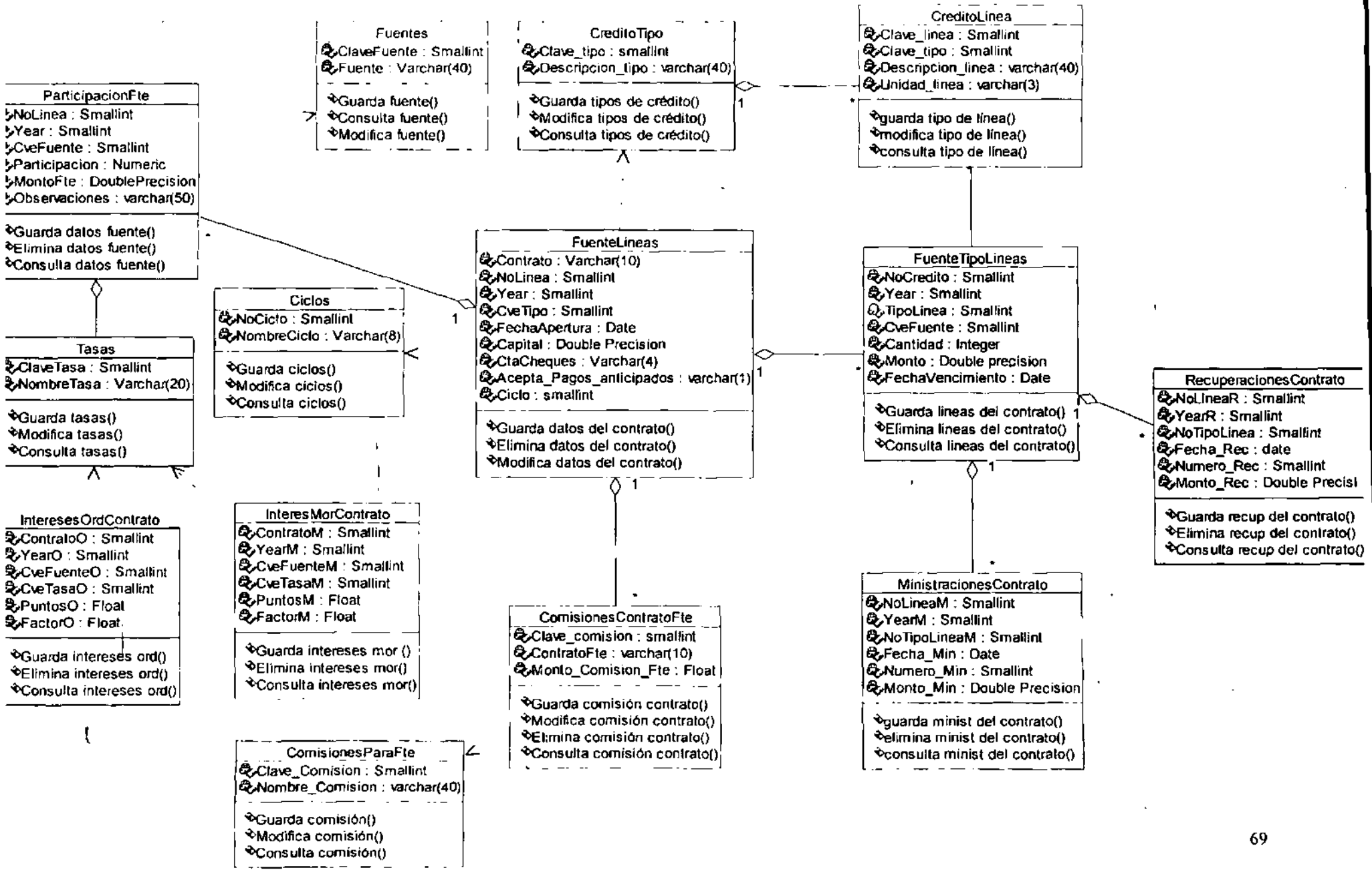
Las generalizaciones se usan durante el análisis para extraer el comportamiento compartido y común entre varias clases del análisis diferentes. La generalización se deben mantener en un nivel alto y conceptual y su propósito principal es hacer el modelo del análisis más sencillo.

3.2.4. Realizar un Diagrama de Clases

Una clase del análisis frecuentemente participan en la realización de varios casos de uso y algunas responsabilidades, atributos y asociaciones de una clase en específico son relevantes únicamente para un caso de uso. Esto es importante durante el análisis para coordinar todos los requerimientos de la clase y sus objetos que pueden tener diferentes casos de uso. Por eso es importante realizar un diagrama de clases por cada caso de uso.

Finalmente se deben reunir las clases del análisis que participan en la realización de los casos de uso en un diagrama de clases para ver la relación que se emplea en la realización de los casos de uso.

Ejemplo: En el siguiente ejemplo se aprecia el diagrama de clases del caso de uso registrar un contrato con las fuentes con su relaciones y agregaciones



Requerimientos Especiales

Los requerimientos especiales son descripciones textuales que recopilan todos los requerimientos especiales de la realización de un caso de uso.

Ejemplo
Un requisito especial que se identificó en el caso de uso registrar un contrato con la fuentes financieras incluye.

- *El número de contrato por parte de la unión debe ser consecutivo.*
- *Forzosamente se deben capturar las tasas de interés*
- *El monto del contrato debe coincidir con la participación de las fuentes.*
- *El monto de la líneas debe coincidir con el monto del contrato.*

Capítulo 4. Diseño

4.1 Introducción

El modelo del diseño describe la realización física de los casos de uso enfocándose a los requisitos funcionales y no funcionales, a otras restricciones relacionadas al ambiente de implementación y al impacto del sistema.

La entrada esencial del diseño, es el resultado del análisis ya que provee la comprensión de los requisitos utilizando el lenguaje de los desarrolladores e impone la estructura del sistema tratando de que sea preservada como sea posible en el diseño.

Los subsistemas permiten organizar el diseño en piezas más fáciles de manejar, por eso, los subsistemas y las clases del diseño representan abstracciones de componentes en la implementación del sistema; estas abstracciones son un paso entre el diseño y la implementación. En el modelo del diseño los casos de uso los realizan las clases del diseño y los objetos se representan por las colaboraciones denotadas en la realización de los casos de uso del diseño.

El modelo del diseño presenta la jerarquía de los subsistemas del diseño que contiene las clases, la realización de los casos de uso y las interfaces entre ellos.

El propósito del diseño es:

1. Comprender el resultado de los requisitos, las restricciones relacionadas al lenguaje de programación, definir el rehuso de componentes, el sistema operativo, tecnologías de distribución y concurrencia, la gestión de transacciones y las interfaces de usuario.
2. Crear un punto de partida para la implementación capturando los requisitos en subsistemas individuales, interfaces y clases.
3. Tener la capacidad de dividir la implementación en piezas más manejables para trabajar con diferentes equipos de desarrollo.
4. Capturar las interfaces entre los subsistemas.
5. Tener la capacidad de visualizar y razonar acerca del diseño usando notaciones comunes.
6. Crear abstracciones de la implementación del sistema, en el sentido que la implementación sea un refinamiento del diseño sin cambiar su estructura.

4.2 Diseño de la Arquitectura

El propósito del diseño arquitectónico es un bosquejo de los modelos del diseño y de instalación mediante la identificación de los siguientes elementos:

1. Los nodos y la configuración de la red.
2. Los subsistemas y su interfaz.
3. Las clases del diseño y las clases activas.
4. Los mecanismos de diseño genéricos que manejan los requisitos comunes.

En esta actividad se consideran varias posibilidades de reutilización, por ejemplo, la reutilización de partes de sistemas similares o productos de software generales. El resultado de los subsistemas, interfaces u otros elementos del diseño se incorporan posteriormente y se deben mantener, refinar y actualizar la descripción de la arquitectura y sus vistas arquitectónicas de los modelos del diseño y de instalación.

Las actividades que se deben realizar
1. Modelo de instalación (4.2.1.)
   1.1 Identificar los nodos y la configuración de la red
   1.2 Identificar los subsistemas y sus interfaces
   1.3 Identificar la arquitectura de las clases del diseño
   1.4 Identificar los mecanismos de diseño genéricos
2. Diseñar los Casos de Uso (4.2.2.)
   2.1 Realizar casos de uso del diseño
   2.2 Identificar las clases que participan en el diseño
   2.3 Describir las interacciones de los objetos y los subsistemas
   2.4 Identificar la participación de los subsistemas e interfaces
3. Clases del Diseño (4.2.3.)
   3.1 Diseñar una clase
   3.2 Identificar operaciones, atributos, asociaciones, agregaciones y generalizaciones
   3.3 Describir estados
   3.4 Diagrama de clases

4.2.1 Modelo de instalación

Describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre nodos computacionales. Se usa como un suministro esencial en las actividades de diseño e implementación; por eso es importante considerar que:

1. Cada nodo representa un recurso computacional, generalmente un procesador o un dispositivo de hardware.
2. Los nodos poseen relaciones que representan medios de comunicación entre ellos, como Internet, Intranets, bus, etc.
3. Puede describir diferentes configuraciones de red, incluyendo las configuraciones para pruebas y simulación.
4. La funcionalidad o procesos de un nodo está definida por componentes distribuidos en los nodos.
5. Representa una correspondencia entre la arquitectura software y la arquitectura del sistema (el hardware).

El resultado del diseño del modelo de instalación es la descripción de la configuración de red donde el sistema debe distribuirse.

4.2.1.1 Nodos y configuración de la red

La configuración física de la red frecuentemente tiene influencia sobre la arquitectura del software incluyendo las clases activas que se necesitan y la distribución de la funcionalidad entre los nodos de la red. La configuración común de una red usa un patrón de tres capas en la cual los clientes (usuarios) se dejan en una capa, la funcionalidad de la base de datos en otra y la lógica del negocio o de la aplicación en la siguiente.

En la configuración de la red se deben considerar los siguientes aspectos:
1. ¿Qué nodos se necesitan y cuál debe ser su capacidad en términos de procesamiento y tamaño de memoria?.
2. ¿Qué tipo de conexión debe existir entre los nodos y qué protocolos de comunicación pueden usar?.

3. ¿Qué características deben tener las conexiones y protocolos de comunicación como el ancho de banda, la disponibilidad y calidad?.
4. Si hay necesidad de contar con capacidad de procesamiento redundante, migración de procesos, mantenimiento de copias de seguridad de los datos o aspectos similares.

Al conocer los límites, las posibilidades de los nodos y sus conexiones, se pueden incorporar tecnologías para hacer más fácil la distribución del sistema.

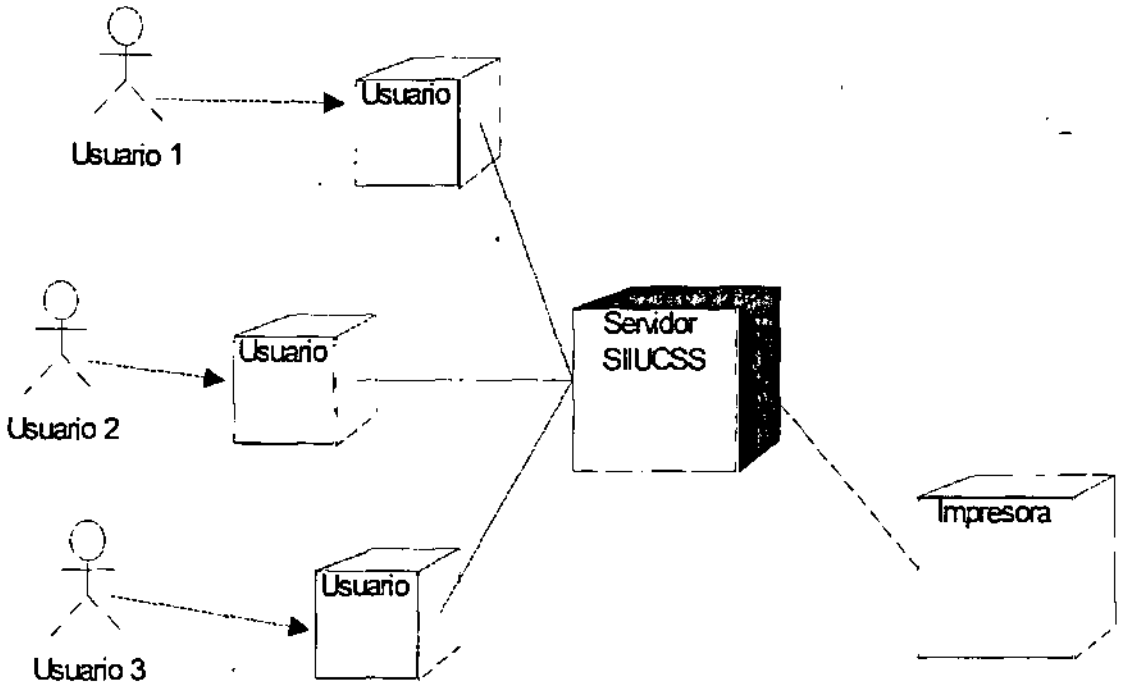
Cada configuración de la red incluye configuraciones especiales para pruebas y simulaciones, éstas se deben representar por separado en un diagrama de instalación. Una vez definidas se debe pensar cómo distribuir su funcionalidad entre ellas.

Ejemplo:

Identificar Nodos
En el sistema integral de uniones de crédito del sector social se identificaron cinco nodos; el nodo principal que es el servidor, tres nodos clientes y el nodo impresora.
1. *El sistema se ejecutará a partir de un nodo servidor, varios nodos usuarios y el nodo impresora.*
2. *El nodo servidor SIIUCSS almacenará la base de datos de la unión de crédito.*
3. *Los nodos Clientes van a guardar los componentes del sistema.*
4. *El nodo impresora lo van a usar todos los nodos clientes para generar los reportes.*

Configuración de la Red
Para la configuración de la red, se tomó en cuenta el volumen de información que se va a procesar, el tipo de computadoras que van a usar, el sistema operativo, el manejador de base de datos y el lenguaje de programación.
1. *En el nodo servidor es importante considerar su capacidad por el volumen de información que va a procesar así como el tamaño de los programas ejecutables. Se va a administrar el sistema a través de un manejador de base de datos, en este caso, Interbase Server 5.0 Borland sobre windows 95' con la restricción de que pueden conectarse hasta 5 clientes de manera simultánea.*
2. *El protocolo de comunicaciones puede ser TCP/IP o Net BUI.*
3. *El nodo servidor requiere un mínimo de 34 MB en memoria RAM y 44MB de espacio en disco duro así como de una unidad de CD Rom.*
4. *Se recomienda que sea una red de tipo árbol.*
5. *El sistema operativo que se recomienda es windows'95, también puede ser windows NT.*
6. *Para poder seleccionar los nodos clientes adecuados, se debe considerar el tamaño de los programas ejecutables y el manejador de base de datos. Los nodos clientes requieren un mínimo de 16 MB en memoria RAM y 20 MB de espacio en disco duro.*
7. *El lenguaje de programación seleccionado es Delphi 3.0 de Borland porque su manejo es sencillo y no requiere algún controlador de Base de Datos, lo que permite que las operaciones con la base de datos se realicen de una manera más rápida incluso a través de la red.*

La relación entre los nodos es a través de la conexión física especificada en el siguiente diagrama:



4.2.1.2 Subsistemas y sus Interfaces

Los subsistemas organizan el modelo del diseño en piezas más manejables que consisten de clases del diseño, la realización de casos de uso, interfaces y otros subsistemas. Se pueden identificar como una forma de dividir el trabajo de diseño ya que a medida que va evolucionando y creciendo el modelo del diseño se irán identificando los subsistemas hasta convertirse en una estructura grande que pueda descomponerse. Pero no todos los subsistemas se desarrollan, algunos representan productos reutilizados y otros son recursos existentes de la empresa. Se proveen las interfaces que representan funcionalidad en términos de operaciones.

El contenido de un subsistema debe encontrarse fuertemente asociado y sus dependencias entre unos y otros o entre sus interfaces debe ser mínima.

Poseen las siguientes características:

1. Pueden representar una separación de aspectos del diseño. Por ejemplo, en un sistema grande algunos subsistemas pueden desarrollarse por separado y de manera simultánea por diferentes desarrolladores y con diferentes aptitudes de diseño.
2. Las dos capas de aplicación de más alto nivel y sus subsistemas tienen trazas directas hacia paquetes y/o las clases del análisis.
3. Si los subsistemas representan componentes muy grandes en la implementación del sistema , éstos proporcionan interfaces compuestas creadas por componentes pequeños que se convierten en ejecutables, ficheros binarios o entidades similares que pueden distribuirse en diferentes nodos.
4. Los subsistemas pueden reutilizar productos software.
5. Los subsistemas pueden ser sistemas heredados en el modelo del diseño.

El propósito de diseñar un subsistema es para:

1. Asegurar que el subsistema sea tan independiente como sea posible de otros subsistemas y/o de sus interfaces.
2. Asegurar que el subsistema provea las interfaces correctas.
3. Asegurar que el subsistema cumpla su propósito de realizar correctamente la operaciones definidas por las interfaces.

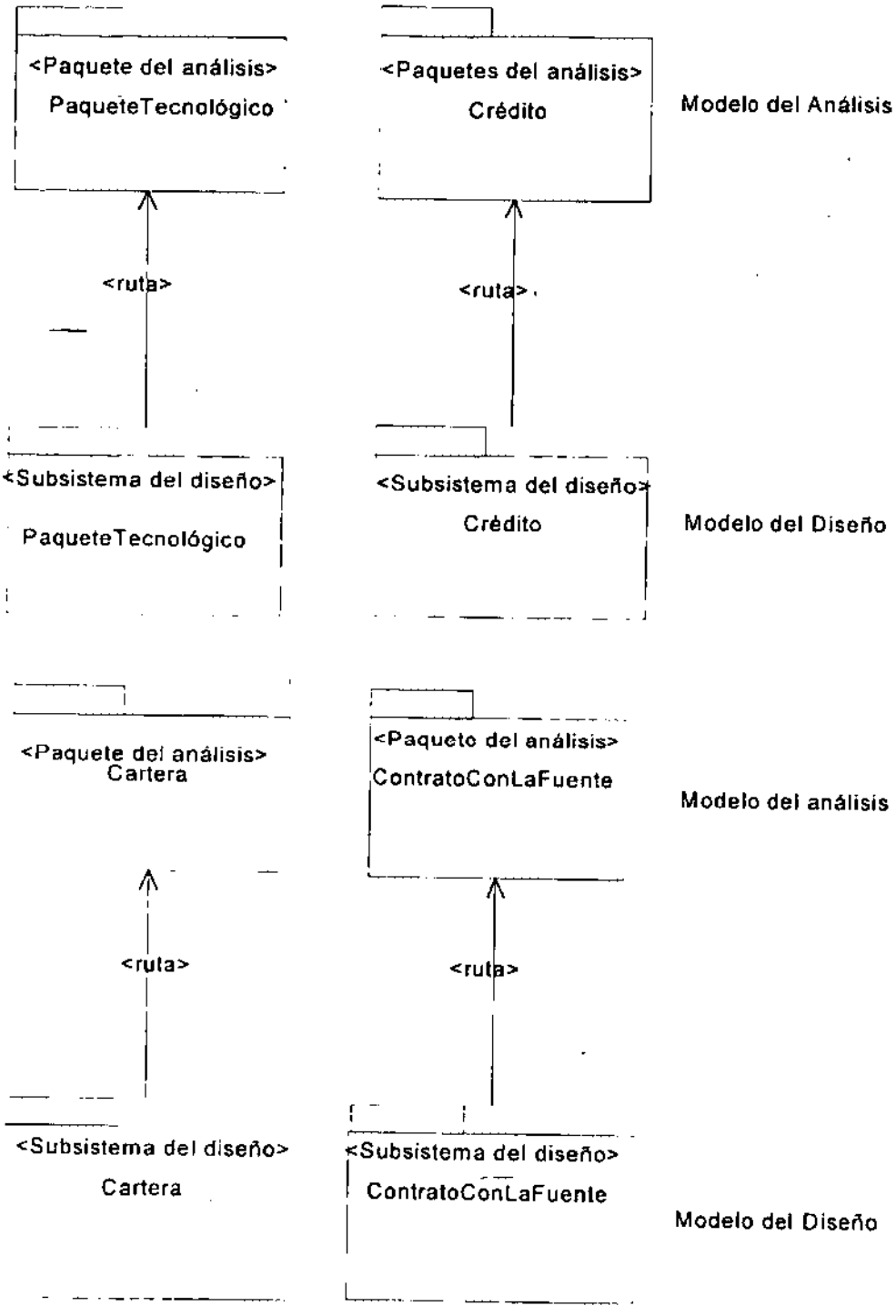
Identificar Subsistemas de la capa de aplicación específica y de la capa de aplicación general
Se deben identificar los subsistemas de la capa de aplicación específica y los de la capa de aplicación general mediante los paquetes del análisis que se identificaron durante el análisis y que se pueden descomponer; éstos se aprovechan para que no se rompa la estructura del sistema de acuerdo a los servicios que ofrece. Sin embargo, la identificación de los subsistemas permite refinarlos durante la implementación e instalación del sistema.

Un refinamiento de la descomposición inicial del subsistema comparada con los paquetes del análisis es necesario cuando:

1. Una parte del paquete del análisis corresponde con un subsistema por si misma, es decir si se encuentra que esa parte puede compartirse por varios subsistemas.
2. Algunas partes del paquete del análisis se realizan por productos software reutilizados, éstas se pueden asignar a capas intermedias o subsistemas de software del sistema.
3. Los paquetes del análisis no representan una división apropiada del trabajo.
4. Los paquetes del análisis no están preparados para una distribución en los nodos, por eso los subsistemas se necesitan descomponer en subsistemas más pequeños, de tal forma que cada uno se pueda asignar a un nodo determinado.

Ejemplo
En el siguiente ejemplo se puede observar la identificación de los subsistemas basados en los paquetes del análisis existentes.

De acuerdo a las actividades anteriores, se identificaron los subsistemas de la capa de la aplicación general y los de la capa de aplicación específica, tomando como base los paquetes del análisis; estos subsistemas no son definitivos, conforme se va avanzando en el diseño pueden surgir nuevos subsistemas, lo que permitirá que se vaya refinando el modelo del diseño.



Identificar subsistemas intermedios y de software del sistema

La capa intermedia y el software del sistema constituye el cimiento del sistema ya que toda la funcionalidad descansa sobre software como el sistema operativo, sistema de administración de base de

datos, software de comunicaciones, tecnología de distribución de objetos y tecnologías de gestión de transacciones.

Una forma de controlar las dependencias es tratar que cada producto software adquirido sea como un subsistema independiente con interfaces explícitas para el resto del sistema.

Definir y mantener dependencias entre los subsistemas

Las dependencias entre los subsistemas se deben definir por su contenido en las relaciones unas con otras y la dirección de la dependencia debe ser la misma que la dirección de la relación. Si las dependencias son un bosquejo antes de conocer el contenido de los subsistemas, se deben consideran las dependencias entre los paquetes del análisis que corresponden con los subsistemas del diseño. Así mismo, si se utilizan las interfaces entre los subsistemas, las dependencias deben ir hacía las interfaces y no hacía los subsistemas mismos.

Las dependencias se deben definir y mantener en un subsistema respecto a otros cuando los elementos contenidos en uno, estén asociados con otro. Sin embargo, es mejor depender de una interfaz que de un subsistema, ya que un subsistema puede ser sustituido con otro subsistema y las interfaces no necesitan sustituirse.

Se deben tratar de minimizar las dependencias de otros subsistemas y/o interfaces y considerar la relocalización de las clases contenidas en un subsistema que sean muy dependientes de otros subsistemas.

Ejemplo

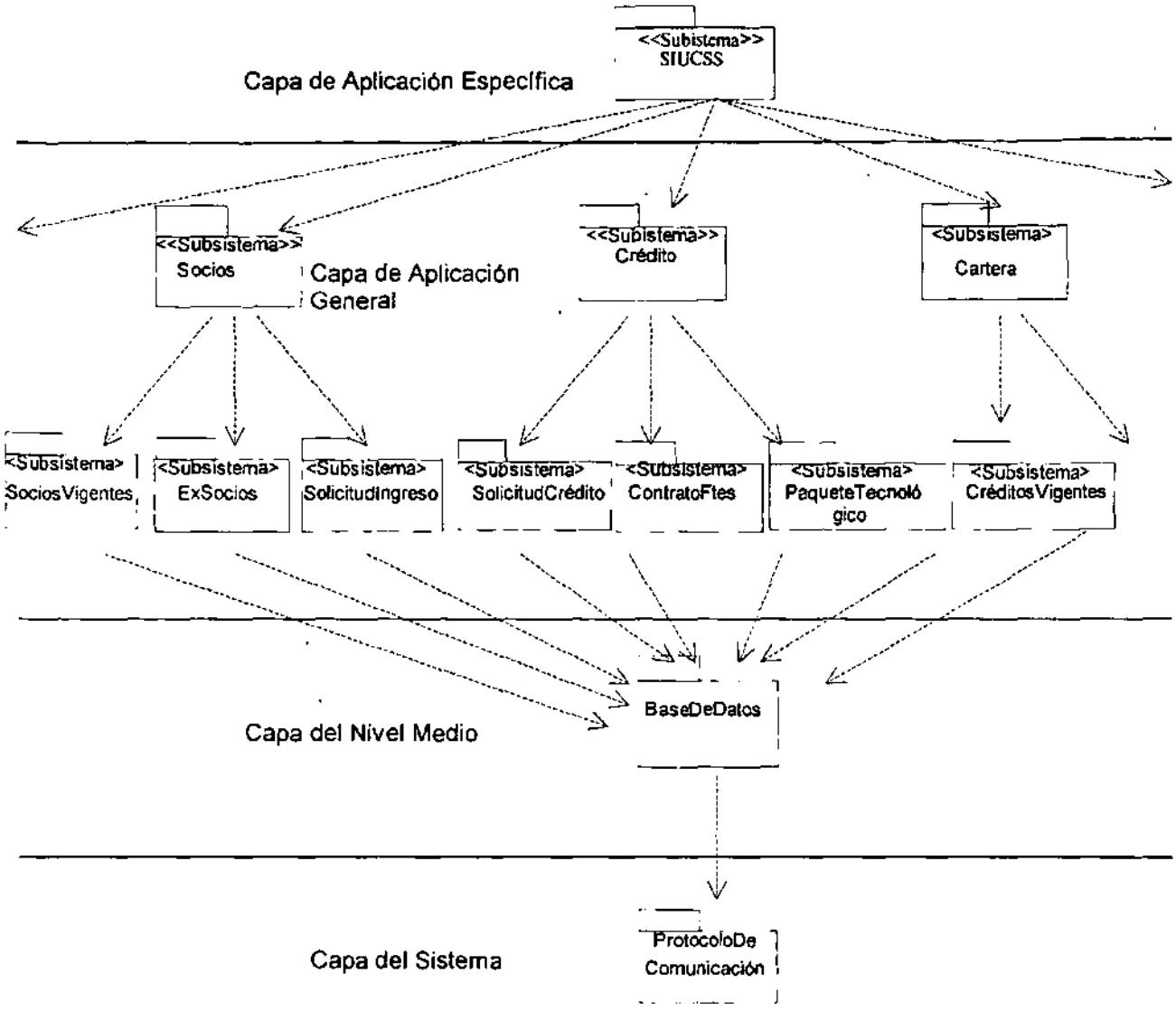
En el siguiente ejemplo se observa la identificación de los subsistemas, así como las dependencias entre ellos en las capas correspondientes.

De acuerdo al diseño del sistema integral de uniones de crédito del sector social, conforme se fue avanzando en el diseño fueron surgiendo nuevos subsistemas lo que permitió que se agruparan con los subsistemas anteriores de acuerdo a su contenido y dependencias que existen entre ellos; con esto se identificaron las capas de aplicación específica, de aplicación general, del nivel medio y la software del sistema.

El sistema quedó distribuido de la siguiente forma:

1. *Se identificó un subsistema principal, el Subsistema SIIUCSS que va a controlar los subsistemas: Socios, Cartera, Crédito, Unión de Crédito, Contabilidad, Caja, Inversiones, Mantenimiento al sistema y ayuda localizados en la capa de aplicación general.*
2. *A su vez el:*
   - *2.1 El subsistema Socios está relacionado con los subsistemas Solicitud de Ingreso, Socios Vigentes y Ex Socios.*
   - *2.2 El subsistema Crédito está relacionado con los subsistemas contrato con la fuente, paquete tecnológico y solicitud de crédito.*
3. *En la capa del nivel medio se encuentra el subsistema Base de Datos y*
4. *En la capa del sistema software se encuentra el protocolo de comunicaciones TCP/IP.*

Para comprender cómo se identifican los subsistemas y la distribución del Sistema Integral de uniones de Crédito del Sector Social, únicamente se ejemplificaron dos subsistemas principales, subsistema crédito y subsistema socios.



Interfaces

Las interfaces se usan para especificar las operaciones que proveen las clases del diseño y los subsistemas. Una clase del diseño que proporcione una interfaz debe proporcionar métodos que realicen las operaciones de dicha interfaz y un subsistema que proporcione una interfaz debe contener clases del diseño o subsistemas que proporcionen la interfaz. Las interfaces proveen un medio de separación de la especificación de la funcionalidad en términos de operaciones y de su implementación en términos de métodos. Esta separación hace independiente la implementación de la interfaz a cualquier cliente que dependa de ella. Una implementación particular de una interfaz así como las clases del diseño o subsistemas pueden sustituirse con otra implementación sin tener que cambiar los clientes.

Este tipo de interfaces se consideran como requisitos para el grupo de desarrollo de los subsistemas y pueden usarse como un instrumento de sincronización entre diferentes grupos que trabajan de manera simultánea con diferentes subsistemas.

Identificar las interfaces entre los subsistemas

Algunas interfaces las proveen las clases u otros subsistemas dentro del subsistema; cuando se da este caso, las interfaces definen operaciones que son accesibles desde fuera del subsistema.

Antes de conocer el contenido de los subsistemas y el bosquejo inicial de la interfaz, primero se deben considerar las dependencias entre los subsistemas ya que cuando un subsistema tiene una dependencia directa necesita proveer una interfaz. Sin embargo, si hay un paquete del análisis y cualquiera de sus clases están referenciadas desde el exterior del paquete puede implicar una interfaz para el subsistema.

Las interfaces de las dos capas inferiores son más fáciles de encontrar ya que en estas capas encapsulan productos software y algunos frecuentemente tienen interfaces predefinidas. Sin embargo, no es suficiente con identificarlas, también se necesita identificar las operaciones que deben definirse por cada interfaz. Esto se hace mediante el diseño de los casos de uso en términos de subsistemas y sus interfaces.

Mantener el Contenido de los Subsistemas

Un subsistema logra su propósito cuando ofrece una realización correcta de las operaciones definidas por las interfaces.

El resultado incluye:

1. Por cada interfaz que provee el subsistema, debe haber clases del diseño u otros subsistemas.
2. Para clarificar cómo el diseño interno de un subsistema realiza cualquiera de sus interfaces o casos de uso, se pueden crear colaboraciones en términos de elementos contenidos en el subsistema.

Mantener interfaces proporcionadas por el subsistema

Las operaciones definidas por las interfaces que proporciona un subsistema deben soportar todos los roles que cumple el subsistema en las diferentes realizaciones de caso de uso. De hecho, las interfaces se van refinando a medida que va evolucionando el modelo del diseño y se van diseñando los casos de uso.

La misma técnica que se utiliza para mantener las clases del diseño y sus operaciones se maneja en mantener las interfaces y sus operaciones.

Ejemplo

Una vez agrupados los subsistemas se definieron las interfaces principales, una por cada subsistema (Se le asignó el mismo nombre a la interfaz con el fin de identificarla fácilmente).

De acuerdo al diagrama anterior podemos observar que a partir del Subsistema SIIUCSS inicia la Pantalla Principal del sistema, que se le da el nombre de PantallaSIIUCSS, esta pantalla puede navegar por todo el sistema, ya que de ella dependen los subsistemas (Socios, Crédito, Cartera, Contabilidad, Inversiones, Caja, Mantenimiento del Sistema, Ayuda y Unión de Crédito). Para poder navegar debe elegir el subsistema que desea consultar por ejemplo, para poder consultar el subsistema socios vigentes debe elegir el subsistema Socios que le corresponde la pantalla Socios y posteriormente seleccionar el Subsistema Socios Vigentes que le corresponde la pantalla Socios Vigentes.

El Sistema está distribuido de la siguiente manera:
Del subsistema principal ((0) Pantalla SIIUCSS) se desprenden los subsistemas con su interfaz correspondiente:

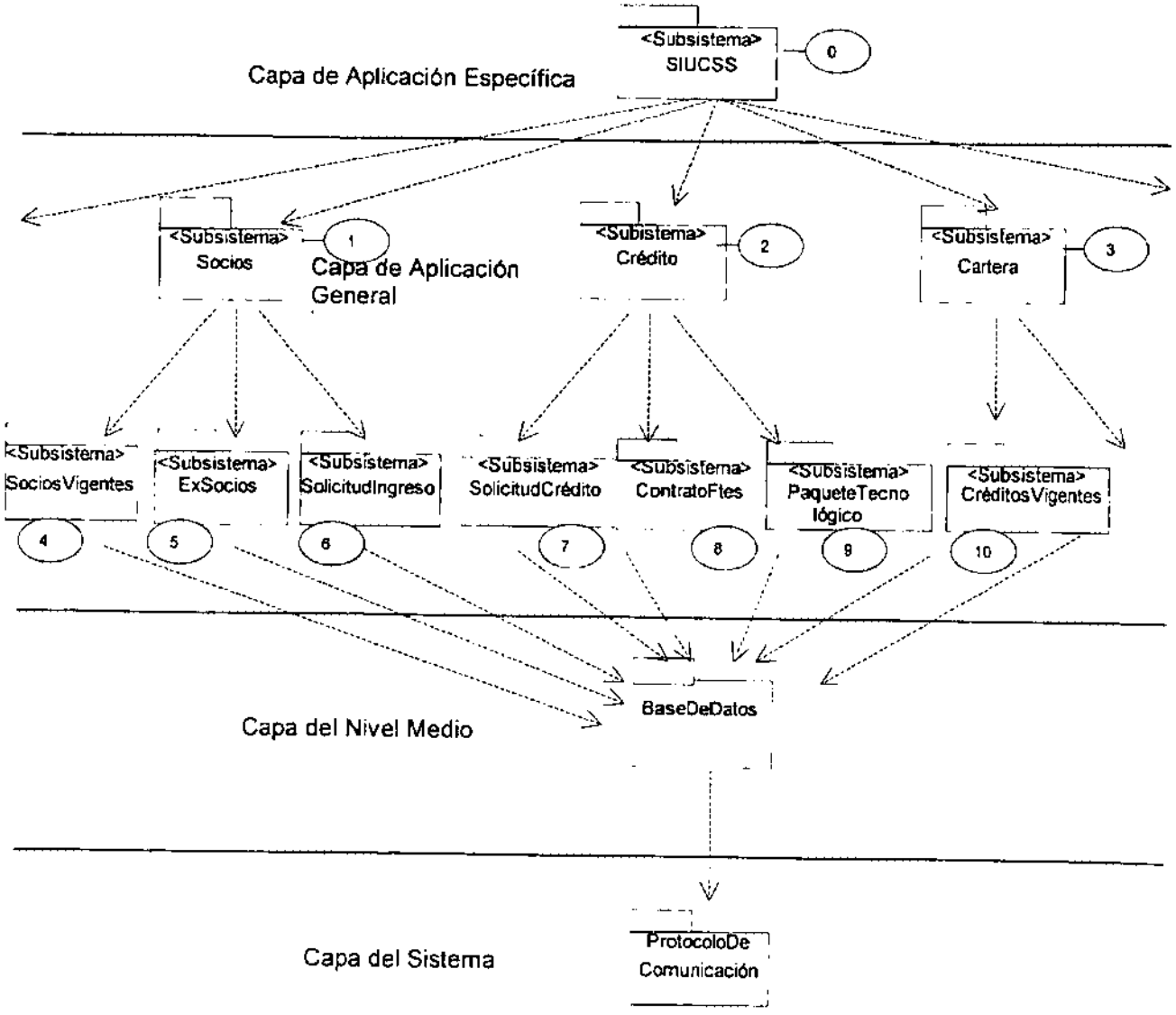
1. Subsistema Socios → Pantalla Socios
2. Subsistema Crédito → Pantalla Crédito
3. Subsistema Cartera → Pantalla Cartera

De la PantallaSocios ((2) Subsistema Socios) se desprende una interfaz por cada Subsistema, las cuales también tienen el mismo nombre que el subsistema.

3. *PantallaSolicitudIngreso*
4. *PantallaSociosVigentes*
5. *PantallaExSocios*

Para el Subsistema Fuentes (Pantalla Fuentes) se les asignó las siguientes interfaces:

6. *PantallaPaqueteTecnológico*
7. *PantallaContratoConLaFuente*
8. *PantallaSolicitudCrédito*



Subsistemas de Servicio

La identificación de los subsistemas de servicio se basan en los paquetes de servicio del modelo del análisis, generalmente es uno a uno entre ellos. Consecuentemente, los subsistemas de servicio son más comunes en las capas de aplicación especifica y de aplicación general.

Sin embargo, los subsistemas de servicio necesitan manejar mejor los resultados que los paquetes de servicio por las siguientes razones:

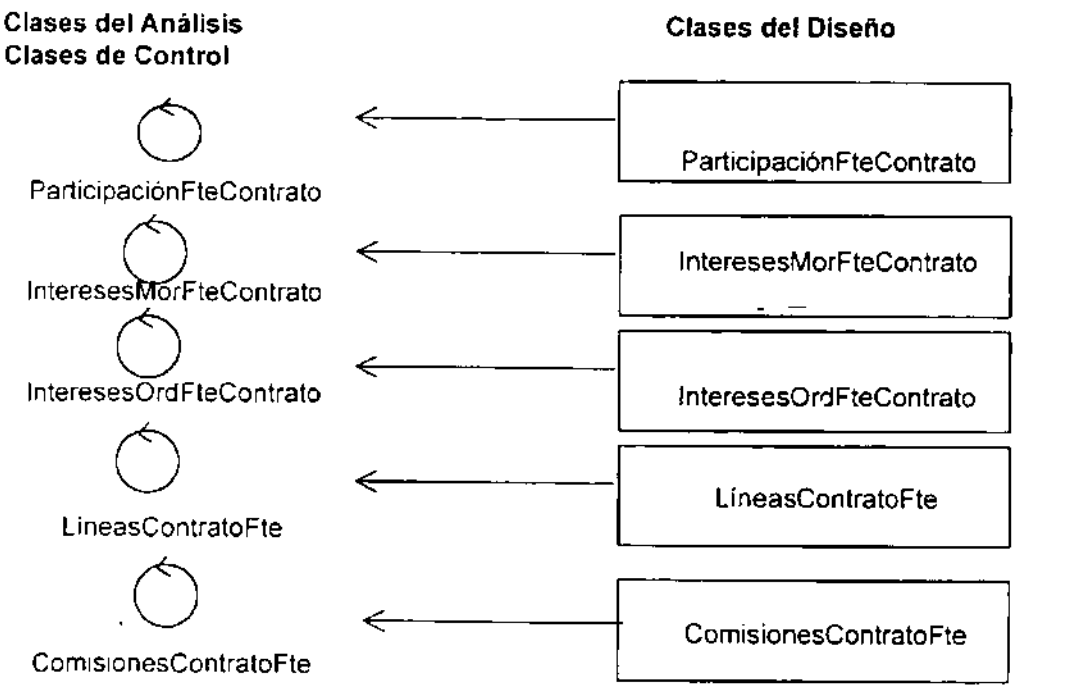
1. Los subsistemas de servicio necesitan proveer sus servicios en términos de interfaces y sus operaciones.
2. Los subsistemas de servicio contienen clases del diseño en vez de clases del análisis. Estos tienden a contener más clases que los paquetes de servicio y pueden desagregarse más adelante en subsistemas más pequeños para manejar su tamaño.
3. Un subsistema de servicio algunas veces se direcciona a un binario o a un componente ejecutable en la implementación. Pero en algunos casos, un subsistema de servicio necesita particionarse y tener cada parte en diferentes nodos, pero esto puede implicar que se necesite un binario o un componente ejecutable por cada nodo.

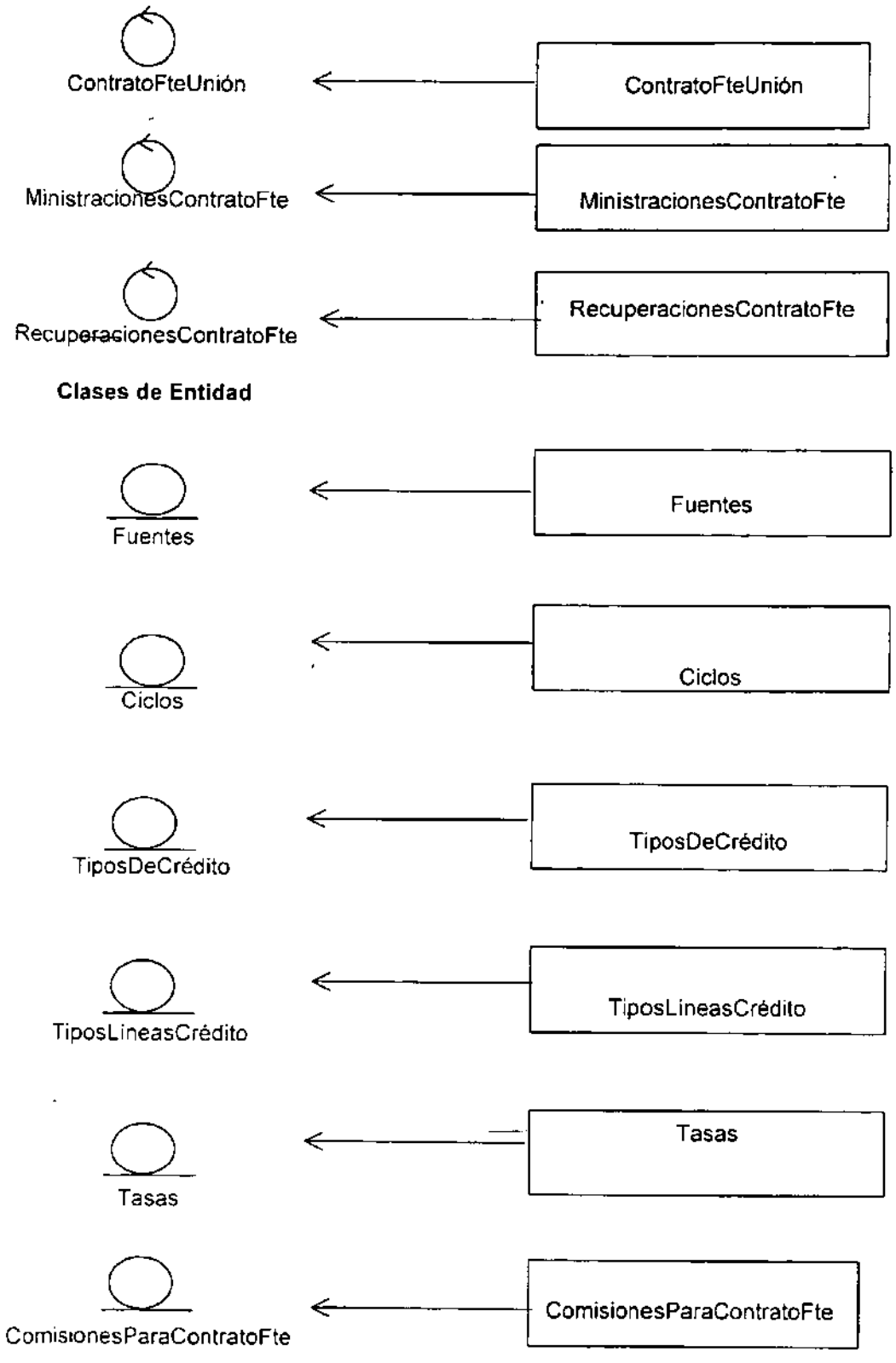
4.2.1.3 Identificar la arquitectura de las clases del diseño

Es práctico identificar las clases del diseño primeramente durante el ciclo de vida del software para iniciar el trabajo de diseño, pero se recomienda que se lleve a cabo un esbozo de las clases más importantes ya que si se identifican más clases, se puede dar el caso de que cuando se deseen utilizar para realizar los casos de uso éstas se tengan que refinar o modificar.

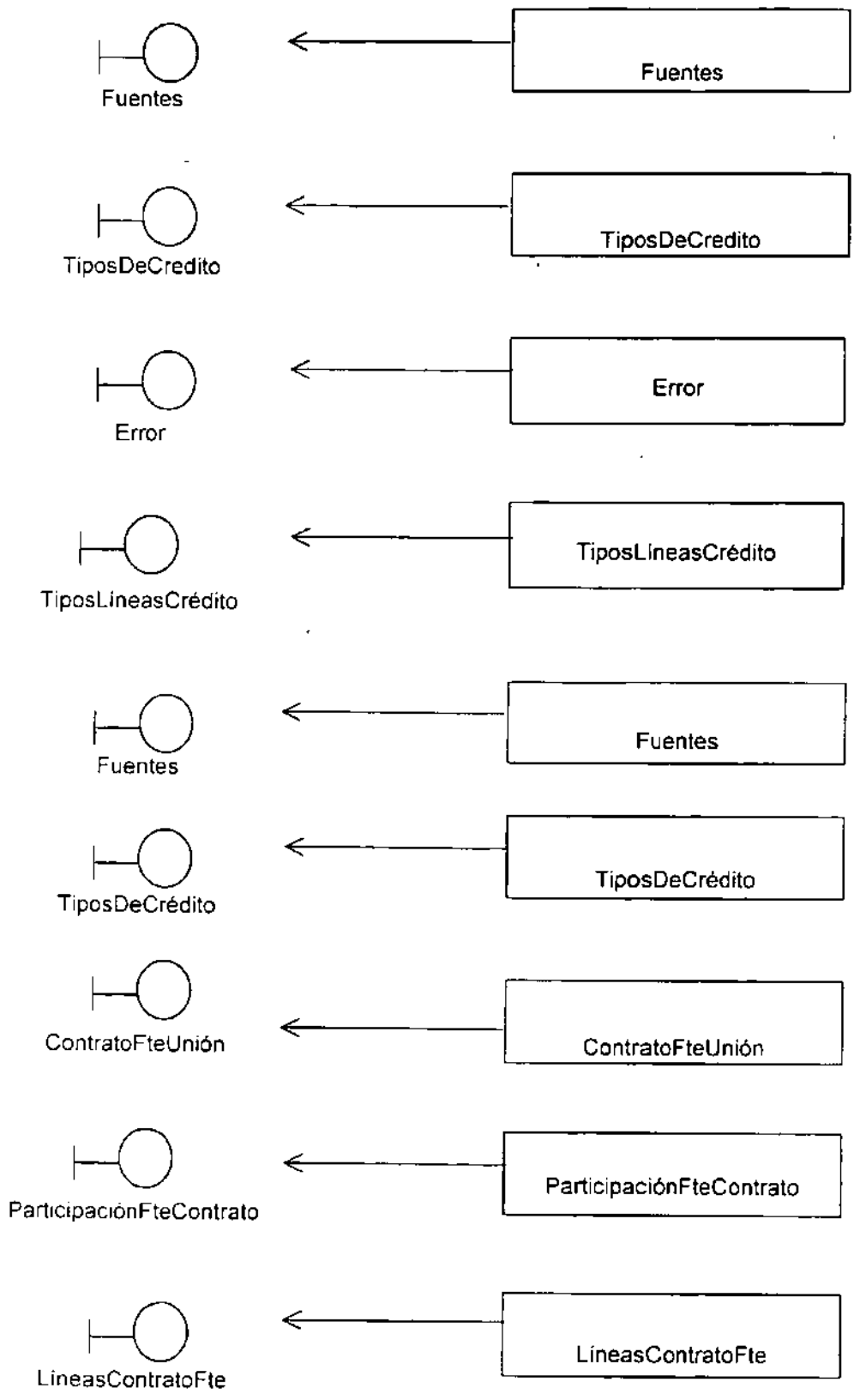
Se pueden esbozar las clases del diseño a partir de las clases que se encontraron en el análisis, también se pueden utilizar la relaciones entre esas clases del análisis para identificar un conjunto tentativo de relaciones entre las clases del diseño.

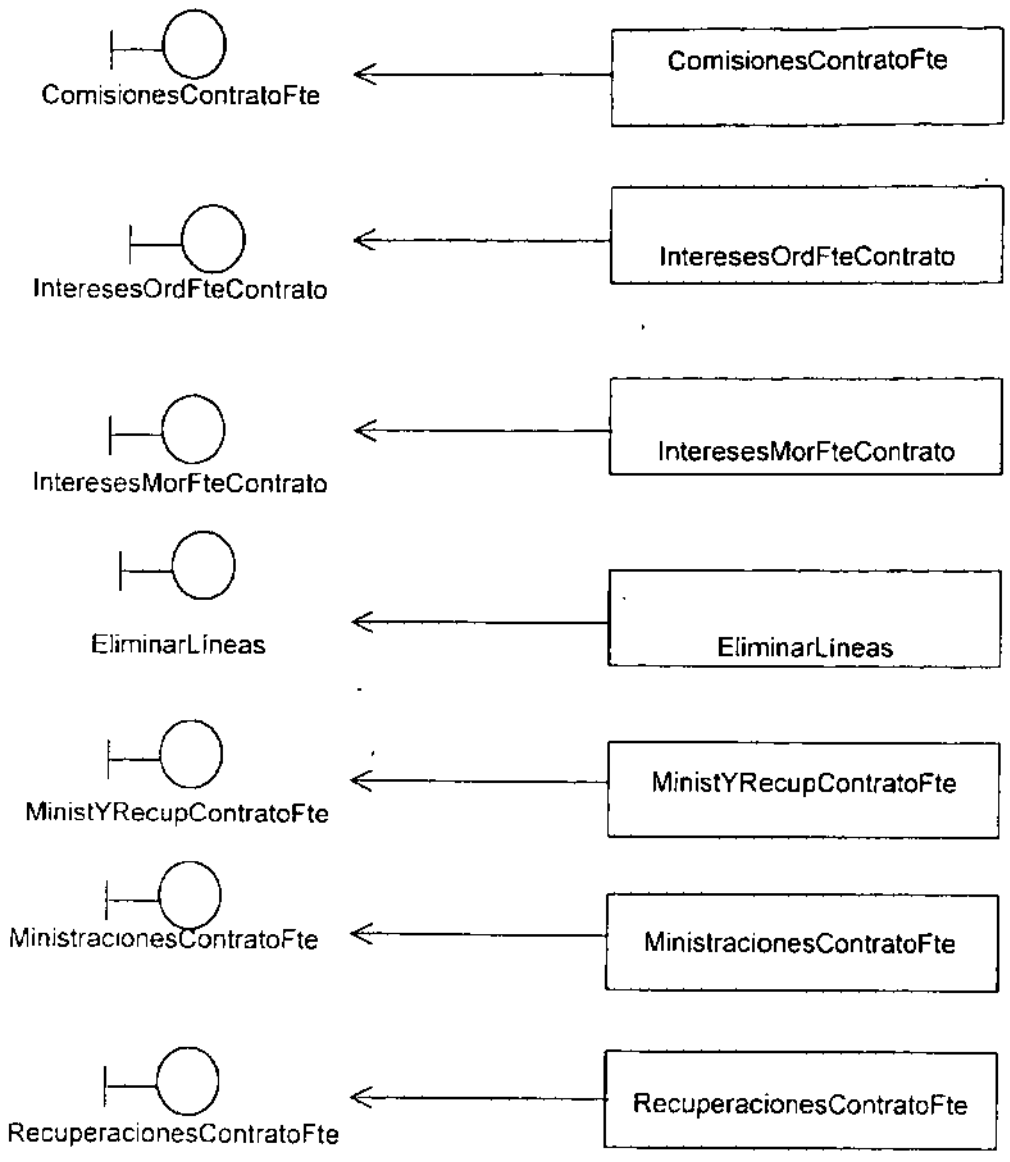
Ejemplo del caso de uso registrar un contrato de las fuentes
De acuerdo al ejemplo, se identificaron las clases del diseño a partir de las clases del análisis. Primero se identificaron las clases de control, posteriormente las de entidad y finalmente las de frontera.




ContratoFteUnión
ContratoFteUnión
MinistracionesContratoFte
MinistracionesContratoFte
RecuperacionesContratoFte
RecuperacionesContratoFte
Clases de Entidad
Fuentes
Fuentes
Ciclos
Ciclos
TiposDeCrédito
TiposDeCrédito
TiposLineasCrédito
TiposLineasCrédito
Tasas
Tasas
ComisionesParaContratoFte
ComisionesParaContratoFte

Clases Límite o Frontera





Clases Activas

Una clase activa se esboza mediante el ciclo de vida de un objeto activo que va a servir para saber cómo se van a comunicar, sincronizar y compartir información a los objetos activos.

Para realizar un bosquejo de las clases activas se pueden usar los resultados del análisis y del modelo del instalación mapeando las clases del análisis a nodos via clases activas, también se pueden utilizar los subsistemas identificados primeramente y colocar un subsistema en un nodo en específico.

Para identificarlas se toman en cuenta los requisitos de concurrencia del mismo, por ejemplo:

1. Los requisitos de rendimientos.
2. El tiempo de respuesta y disponibilidad de los diferentes actores en su interacción con el sistema.

3. Distribuir el sistema en nodos. Los objetos activos necesitan soportar la distribución en diferentes nodos los cuales requieren al menos de un objeto activo por nodo y otros objetos activos para gestionar la intercomunicación entre los nodos.
4. Otros requisitos como el de arranque y terminación del sistema, reconfiguración y la capacidad de los nodos.

Después de asignar los objetos activos a los nodos es importante considerar la capacidad de los nodos, sus procesadores, el tamaño de la memoria, las características de las conexiones, el ancho de banda y la disponibilidad. Al realizar la distribución, el tráfico de la red frecuentemente tiene un impacto substancial en los recursos computacionales (incluyendo software y hardware) requeridos por el sistema y esto puede causar un mayor impacto en el modelo del diseño por eso se debe mantener bajo un control estricto.

Una clase activa que representa un proceso grande es candidata a un componente ejecutable y debe identificarse durante la implementación. La distribución de clases activas es una entrada importante para la distribución de componentes ejecutables en los nodos durante la implementación. Sin embargo, cuando un subsistema es colocado en un nodo, todos los elementos del subsistema frecuentemente contribuyen a colocar un componente ejecutable en ese nodo.

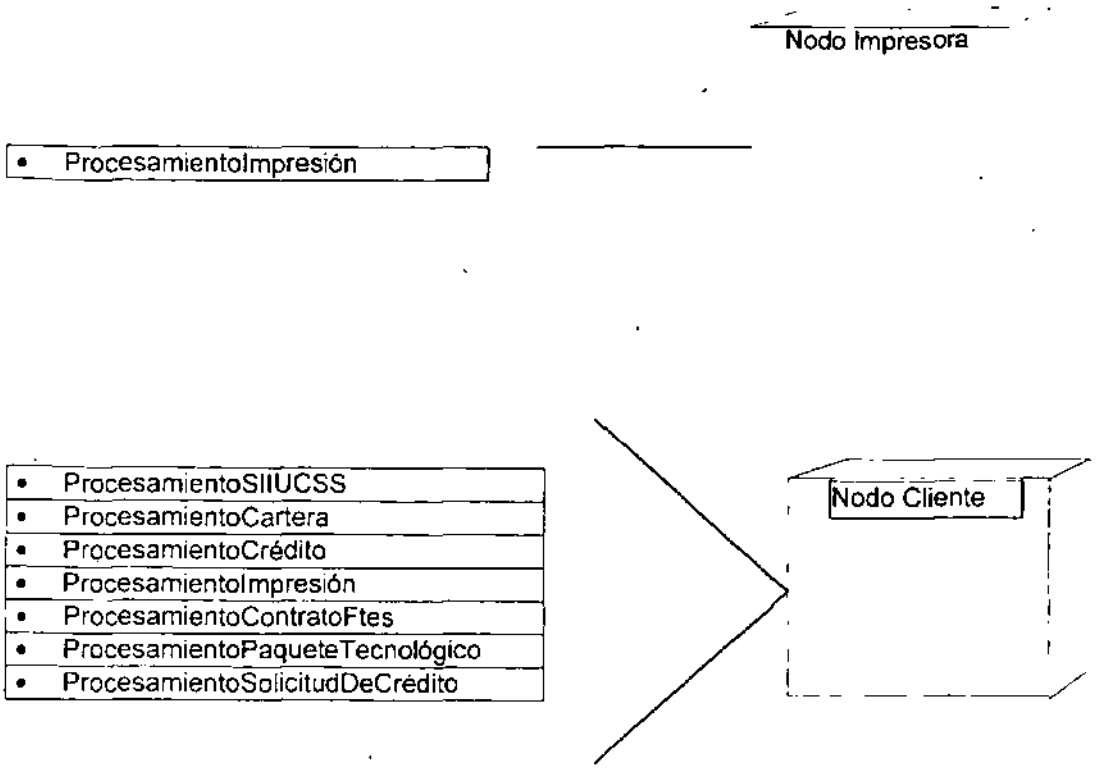
Ejemplo
De acuerdo a los puntos anteriores, para identificar fácilmente las clase activas, se tomó como base los paquetes del análisis y los subsistemas que se identificaron en la capa de aplicación especifica del modelo del diseño.

ProcesamientoCartera	ProcesamientoAccesoSIIUCSS	ProcesamientoCrédito
ProcesamientoPaquete Tecnológico	ProcesamientoSIIUCSS	ProcesamientoSolicitudDe Crédito
ProcesamientoContratoFtes	ProcesamientoImpresión	ProcesamientoConexión BaseDatos

Una vez identificados se distribuyeron en el Nodo correspondiente

Nodo Servidor

- ProcesamientoConexionBaseDatos



4.2.1.4 Identificar los Mecanismos de Diseño Genéricos

En este punto se consideran todos los requisitos comunes, identificados durante el análisis en la realización de casos de uso; el resultado de esto es una serie de mecanismos de diseño genéricos los cuales pueden manifestarse como clases, colaboraciones o subsistemas.

Los requisitos que se manejan frecuentemente están relacionados con aspectos como:

- Persistencia
- Distribución transparente de objetos
- Características de seguridad
- Detección y recuperación de errores
- Gestión de transacciones

En algunos casos el mecanismo no puede identificarse a-priori, se descubre a medida que se exploran las realizaciones de caso de uso y las clases del diseño. Cada mecanismo puede tratarse de diferentes formas con sus pros y sus contras ya que no es factible usar el mismo para todas las situaciones, es necesario proporcionar más de uno y usar el más adecuado en cada situación. También se pueden identificar colaboraciones genéricas que pueden servir como patrones para realizar diferentes casos de uso.

Los mecanismos genéricos se deben identificar y diseñar durante la fase de elaboración pero si se hace cuidadosamente, se pueden diseñar para resolver algunos aspectos difíciles del diseño permitiendo que los casos de uso sean sencillos en la fase de construcción.

identificar colaboraciones genéricas que pueden servir como patrones para realizar diferentes casos de uso.

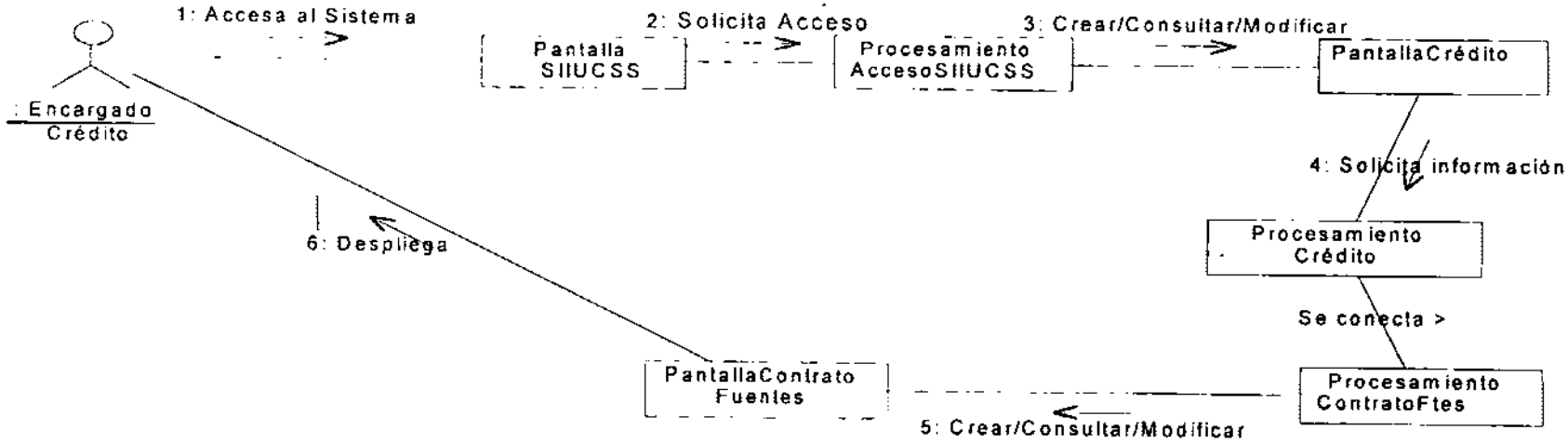
Los mecanismos genéricos se deben identificar y diseñar durante la fase de elaboración pero si se hace cuidadosamente, se pueden diseñar para resolver algunos aspectos difíciles del diseño permitiendo que los casos de uso sean sencillos en la fase de construcción.

Ejemplo

De acuerdo a los puntos anteriores, los mecanismos de diseño genéricos sirven como patrones para realizar varios casos de uso en un proceso general. Primero se identifican los casos de uso que se pueden agrupar y posteriormente las colaboraciones genéricas. A continuación se presenta un diagrama de colaboración para ejemplificar una colaboración genérica

Diagrama de colaboración genérica para un Contrato con las Fuentes, considerando tres casos de uso:

2. *Crear un Contrato con las Fuentes*
3. *Consultar un Contrato con las Fuentes*
4. *Modificar un Contrato con las Fuentes*



1. El encargado accesa al sistema para crear, consultar, modificar un contrato con las fuentes.
2. La Clase Pantalla SIIUCSS envía solicitud de acceso a la clase activa ProcesamientoAccesoSIIUCSS.
3. La clase activa ProcesamientoAccesoSIIUCSS despliega una clase de tipo PantallaFuentes.
4. El objeto PantallaFuentes solicita información a la clase activa ProcesamientoCrédito.
5. La clase activa ProcesamientoCrédito se conecta con la clase activa ProcesamientoContratoFuentes.
6. La clase activa ProcesamientoContratoFuentes despliega la información en la clase PantallaContrato Fuentes para que se pueda crear, consultar o modificar.
7. La clase PantallaContratoFuentes presenta la información al encargado.

4.2.2 Diseñar los Casos de Uso

El propósito de diseñar un caso de uso es para:

1. Identificar las clases del diseño y/o subsistemas cuyas instancias se necesitan para ejecutar el flujo de sucesos de los casos de uso.
2. Distribuir el comportamiento de los casos de uso que interactúan y/o entre los subsistemas participantes.
3. Definir los requisitos sobre las operaciones de las clases del diseño y/o subsistemas y sus interfaces.
4. Capturar los requisitos de implementación del caso de uso.

4.2.2.1 Realizar los casos de uso del diseño

La realización de un caso de uso del diseño es una colaboración en el modelo del diseño que describe cómo se realiza un caso de uso específico y cómo se ejecuta en términos de clases del diseño y sus objetos; proporciona una traza directa hacia un caso de uso del análisis, tiene una descripción textual del flujo de eventos, un diagrama de clases que describe las clases que participan y un diagrama de interacción que muestra la realización de un flujo o escenario concreto de un caso de uso en términos de interacción entre objetos del diseño. Si es necesario, los diagramas describen los subsistemas e interfaces involucradas en la realización del caso de uso. También gestiona muchos requisitos no funcionales (requisitos especiales) capturados en la realización de los casos de uso del análisis. Sin embargo, puede posponer el manejo de algunos requisitos hasta la implementación.

4.2.2.2 Identificar las Clases que participan en el diseño

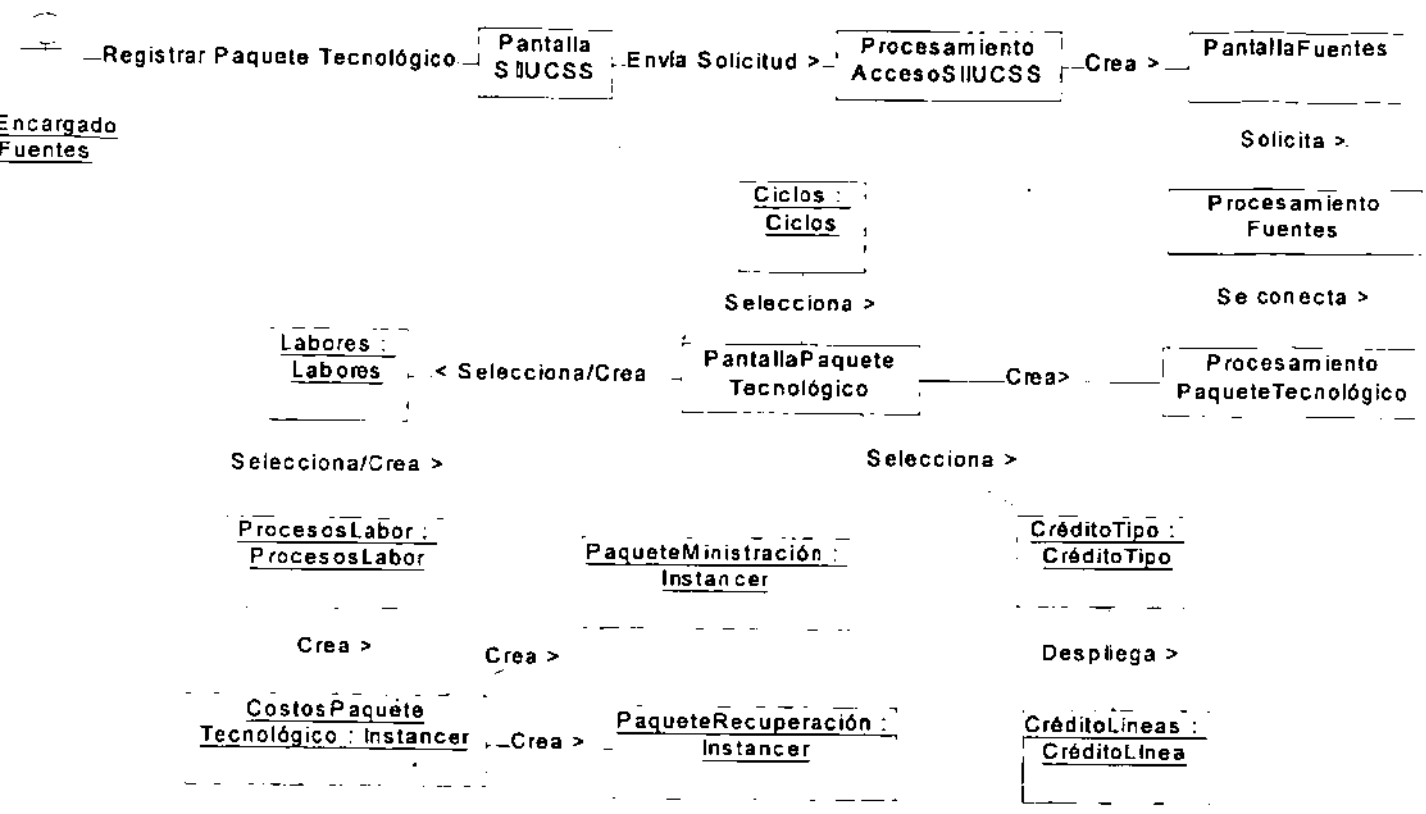
Para identificar las clases del diseño que se necesitan para realizar el caso de uso, se debe hacer lo siguiente:

1. Estudiar las clases del análisis que participan en la realización del casos de uso del análisis en especifico así como las clases del diseño que poseen una traza hacia esas clases del análisis.
2. Estudiar los requisitos especiales de la realización del caso de uso correspondientes al análisis e identificar las clases del diseño que realizan esos requisitos especiales.
3. Como resultado, se deben identificar las clases necesarias.

Las clases del diseño que participan en la realización del caso de uso se deben plasmar en un diagrama de clases asociado con la realización.

Ejemplo

En el siguiente ejemplo, se identificaron las clases que participan en la realización del caso de uso a partir de las clases del análisis.



1. El encargado accesa al sistema a través de la pantalla SIIUCSS para registrar un PaqueteTecnológico.
2. PantallaSIIUCSS envia la solicitud a la clase activa ProcesamientoSIIUCSS.
3. La clase activa ProcesamientoSIIUCSS crea la instancia PantallaFuentes.
4. El objeto PantallaFuentes solicita registrar un PaqueteTecnológico a la clase activa ProcesamientoFuentes.
5. El objetoFuentes se conecta con la clase activa ProcesamientoPaqueteTecnológico.
6. El objeto Procesamiento PaqueteTecnológico crea la instancia Pantalla PaqueteTecnológico.
7. El objeto Pantalla PaqueteTecnológico selecciona la clase Ciclos.
8. El objeto Pantalla PaqueteTecnológico selecciona la clase CréditoTipo.
9. El objeto CréditoTipo despliega la clase CréditoLínea.
10. El objeto PantallaPaqueteTecnológico selecciona/crea la instancia Labores.
11. El objeto Labores selecciona/crea la instancia ProcesosLabor.
12. El objeto ProcesosLabor crea la instancia CostoPaqueteTecnológico.
13. El objeto CostoPaqueteTecnológico crea la instancia PaqueteMinistración.
14. El objeto CostoPaqueteTecnológico crea la instancia PaqueteRecuperación.

4.2.2.3 Describir las interacciones de los objetos y los subsistemas

Diagramas de Interacción

La secuencia de acciones en un caso de uso empieza cuando un actor invoca el caso de uso para enviar algunos mensajes al sistema. En el diseño se describe con un diagrama de secuencias ya que el objetivo es encontrar secuencias de interacciones detalladas y ordenadas.

En algunos casos se incluyen subsistemas en los diagramas de secuencia para describir cuáles de ellos participan en la realización de caso de uso y qué interfaces intervienen entre los que proporcionan esos subsistemas.

Al usar un diagrama de secuencias se ilustra las interacciones mediante transferencia de mensajes entre objetos o subsistemas. Cuando un subsistema recibe un mensaje se dice que es un objeto de una clase del subsistema que envía el mensaje. Cuando un subsistema envía un mensaje es un objeto de una clase del subsistema el que recibe el mensaje. El nombre del mensaje indica una operación del objeto que recibe la invocación o de una interfaz que el objeto proporciona.

Flujo de Sucesos del Diseño

El diagrama de interacción de la realización de un caso de uso es difícil comprenderlo por sí mismo. El flujo de sucesos del diseño, es una descripción textual que explica y complementa los diagramas y sus etiquetas. El texto deberá escribirse en términos de objetos que interactúan para realizar los casos de uso o en términos de subsistemas que participan en él. Pero la descripción no debe mencionar los atributos, operaciones y asociaciones de los objetos ya que si se menciona provoca que la descripción sea difícil de mantener porque los atributos, operaciones y asociaciones de las clases cambian frecuentemente. Así mismo, en la descripción no debe mencionar alguna operación de la interfaz si las interfaces son empleadas en los diagramas.

El flujo de sucesos del diseño es útil si hay muchos diagramas de secuencia describiendo la misma realización del caso de uso o si hay diagramas que representan flujos complejos.

Por eso:

1. El flujo de sucesos del diseño de la realización de un caso de uso no es local a un diagrama de secuencias específico, pero se puede usar para describir la relación entre varios diagramas.
2. Las etiquetas de un diagrama de secuencia son locales al diagrama.

Cuando se usan etiquetas y flujos de sucesos del diseño ambas deben ser complementarias.

Describir las Interacciones entre objetos del Diseño

Para realizar los casos de uso es necesario tener un bosquejo de las clases del diseño y describir cómo interactúan sus objetos. Para representarlo se usa un diagrama de secuencias que contenga las instancias de los actores, los objetos del diseño y la transmisión de mensajes entre éstos. Si los casos de uso tienen distintos flujos o subflujos, frecuentemente se debe crear un diagrama de secuencias por cada uno de ellos para facilitar la realización del caso de uso.

Un diagrama de secuencias se crea empezando por el inicio del flujo del caso de uso y después decidir qué objetos del diseño y qué instancias de los actores se necesitan para realizar cada paso. En muchos casos, los objetos se ajustan de manera natural a la secuencia de interacciones en la realización del caso de uso.

Un diagrama de secuencias funciona cuando:

1. El caso de uso es invocado por un mensaje de una instancia de un actor hacia un objeto del diseño.

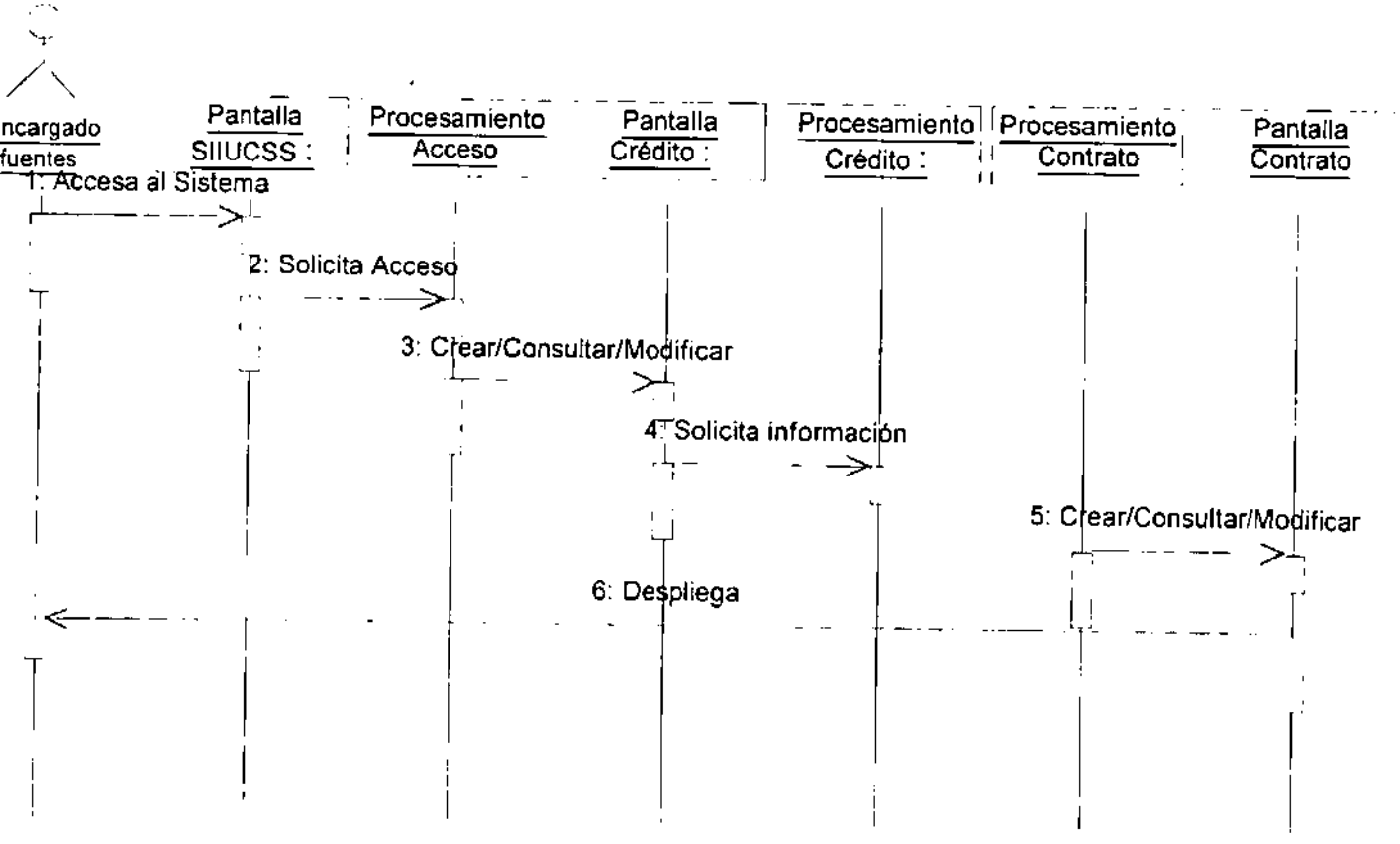
2. Por cada clase del diseño identificada se debe tener al menos un objeto participante en el diagrama de secuencia.
3. Los mensajes que realizan el caso de uso se envían entre líneas de vida de los objetos Un mensaje puede tener un nombre temporal que pasará a ser el nombre de una operación después de que ha sido identificado.
4. Se utilizan etiquetas y flujo de sucesos para complementar los diagramas de secuencia.
5. Los diagramas de secuencia deben tratar todas las relaciones del caso de uso que realiza.

A medida que se van detallando, es probable encontrar nuevas alternativas que puede tomar el caso de uso. Se puede describir en las etiquetas de los diagramas o en diagramas de interacción independientes, ya que conforme se va añadiendo más información se puede ir descubriendo nuevas excepciones que no se consideraron durante la captura o el análisis de los requisitos.

Estas excepciones incluyen:

- Manejar los tiempos cuando los nodos o conexiones se suspenden
- Suministros erróneos que los actores y máquina pueden proveer.
- Mensajes de error generados por el software y hardware.

Ejemplo
A continuación se presenta el ejemplo de un diagrama de interacción.
Diagrama de Secuencias del caso de uso Registrar un Contrato con las Fuentes



4.2.2.4 Identificar la participación de los subsistemas e interfaces

Es más apropiado diseñar un caso de uso en términos de subsistemas y/o sus interfaces que participan en él. Para empezar, se deben identificar los subsistemas que se necesitan para realizar el caso de uso mediante los siguientes pasos:

1. Primero, estudiar las clases del análisis que participan en la realización de casos de uso del análisis, posteriormente identificar los paquetes del análisis posteriormente que contienen a esas clases si existen y después identificar los subsistemas del diseño que poseen una traza hacia esos paquetes del análisis.
2. Segundo, estudiar los requisitos especiales de los casos de uso correspondientes e identificar las clases del diseño que realizan esos requisitos especiales e identificar los subsistemas que contienen esas clases.

Se usan los diagramas de clases para representar las dependencias entre los subsistemas y cualquier interfaz que se utilice en la realización del caso de uso.

Describir las interacciones entre los subsistemas

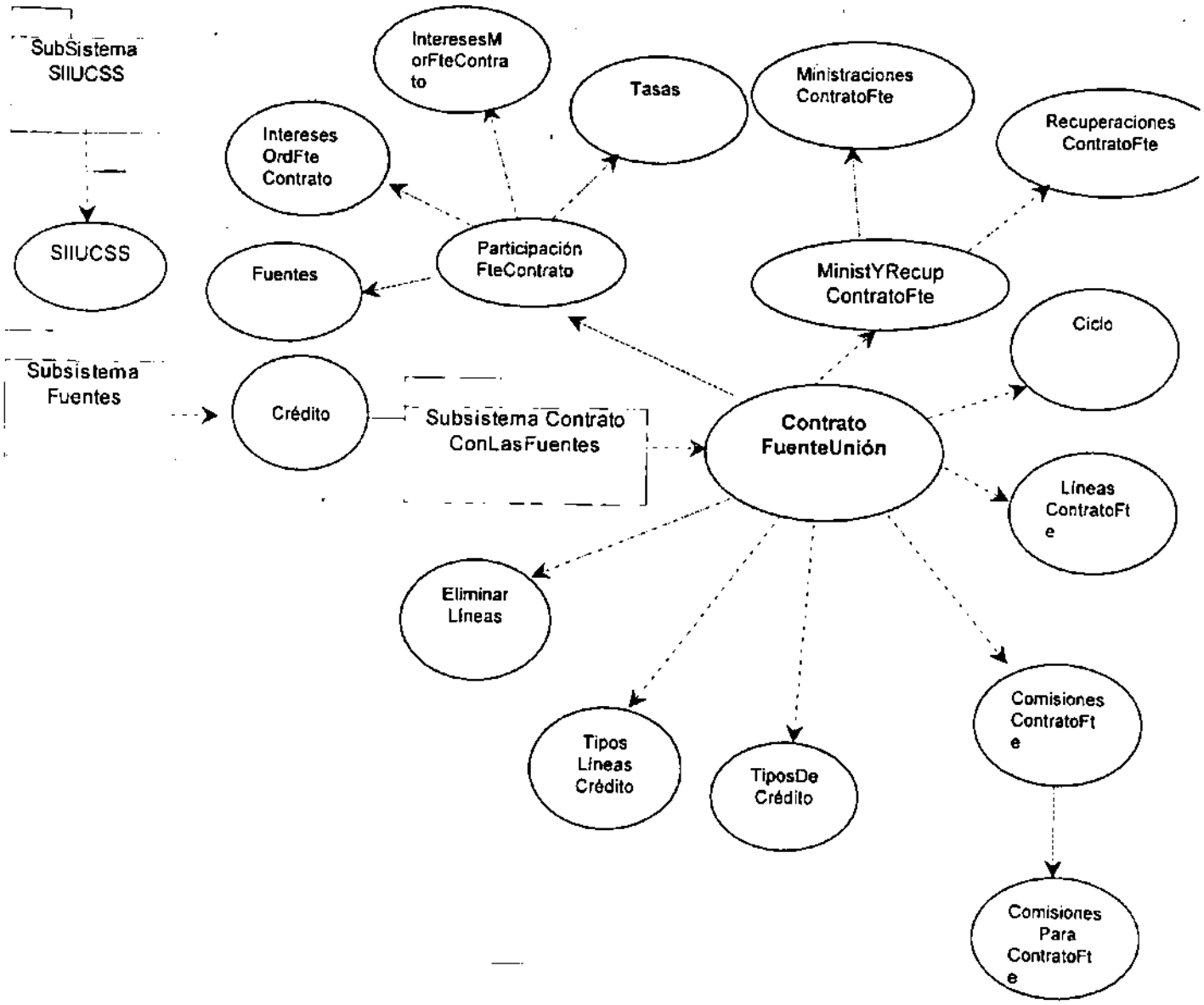
Cuando se cuenta con un bosquejo de los subsistemas para realizar los casos de uso, se necesita describir cómo interactúan los objetos de las clases.

Esto se lleva a cabo usando un diagrama de secuencias que contiene las instancias de los actores, los subsistemas y la transmisión de mensajes entre ellos. Se deben considerar los siguientes puntos al realizarlo:

- Las líneas de vida en los diagramas de secuencias denotan subsistemas en lugar de objetos del diseño.
- Cada subsistema identificado debe tener al menos una línea de vida que lo denote en un diagrama de secuencia.
- Si se asigna un mensaje a una operación de una interfaz es apropiado cualificar el mensaje con la interfaz que proporciona la operación. Esto es necesario cuando un subsistema provee varias interfaces y se debe distinguir qué interfaz se utiliza en cada mensaje.

Ejemplo
Se identificaron los siguientes subsistemas e interfaces por caso de uso ejemplificado.

Caso de Uso: Registrar Contrato Con las Fuentes



4.2.3 Clases del Diseño

Las clases del diseño son una abstracción de una clase o una construcción similar de la implementación del sistema.

La abstracción es:

1. El lenguaje usado para especificar las clases del diseño es el mismo que el lenguaje de programación. Consecuentemente, operaciones, parámetros, atributos y tipos se especifican usando la sintaxis del lenguaje de programación.

2. Especifican la visibilidad de los atributos y operaciones de las clases del diseño.
3. La relación de las clases del diseño que están involucradas con otras clases, frecuentemente tienen relación con las clases implementadas. Por ejemplo, las asociaciones y agregaciones generalmente corresponden a variables (atributos) en las clases de la implementación.
4. Los métodos (que es la realización de operaciones) de una clase del diseño tienen correspondencia directa con los métodos correspondientes en la implementación de las clases (el código). Si los métodos se especifican en el diseño, la mayoría de las veces se especifican en un lenguaje natural o en pseucódigo y se usan como comentarios en las implementaciones del método.
5. Las clases del diseño pueden posponer el manejo de algunos requisitos para la implementación.
6. Una clase del diseño frecuentemente es mapeada en el lenguaje de programación.
7. Una clase del diseño puede realizar interfaces en el lenguaje de programación.
8. Una clase del diseño puede activarse, implicando qué objetos de las clases mantiene su propio hilo de control y se ejecutan concurrentemente con otros objetos activos. Sin embargo, las clases del diseño normalmente no están activas, implicando que sus objetos se ejecuten en el espacio de direcciones y bajo el control de otros objetos activos.

4.2.3.1 Diseñar una Clase

El propósito de diseñar una clase es crear una clase del diseño que cumpla con la realización de los casos de uso y los requisitos no funcionales que se aplican a estos. Incluye mantener el diseño de clases y los siguientes aspectos:

- Sus operaciones,
- Sus Atributos,
- Las relaciones en las que participa,
- Sus métodos (que realiza sus operaciones),
- Los estados impuestos,
- Sus dependencias con cualquier mecanismo de diseño genérico,
- Los requisitos relevantes a su implementación,
- La realización correcta de cualquier interfaz que se requiera.

Bosquejo de las Clases del Diseño

Cuando una o algunas clases del análisis son la entrada en términos de clases del análisis y/o interfaz, es simple asignar a una clase del diseño para que proporcione la interfaz.

Cuando las clases del análisis son un suministro, los métodos utilizados van a depender del estereotipo de la clase del análisis:

- Diseñar clases de interfaz es dependiente de la tecnología de interfaz específica que se esté utilizando.
- Diseñar clases de entidad que representen información persistente frecuentemente implica el uso de tecnologías de bases de datos específicas.
- Es muy delicado diseñar clases de control porque encapsulan secuencias, coordinación de otros objetos o algunas veces la lógica del negocio, por eso es necesario tomar en cuenta los siguientes aspectos:
  - La distribución: Si la secuencia necesita distribuirse y manejarse por diferentes nodos en una red, se puede dar el caso de que se requiera separar las clases del diseño en diferentes nodos para realizar la clase de control.
  - El rendimiento: No es justificable tener clases del diseño por separado para realizar las clases de control, en lugar las clases de control deben realizarse por las mismas clases del diseño que están realizando algunas clases de interfaz relacionadas y/o clases entidad relacionadas.
  - Transacción: Las clases de control frecuentemente encapsulan transacciones. Su diseño debe incorporar alguna tecnología de manejo de transacciones que se estén usando.

Las clases del diseño identificadas en este punto son asignadas trazando dependencias a las correspondientes clases del análisis que se diseñaron.

4.2.3.2 Identificar operaciones, atributos, asociaciones, agregaciones y generalizaciones

Identificar Operaciones

Se deben identificar las operaciones que van a necesitar las clases del diseño y se van describir usando la sintaxis del lenguaje de programación. Esto incluye especificar la visibilidad de cada operación.

Se deben considerar los siguientes puntos:

1. Las responsabilidades de las clases del análisis que tengan traza con las clases del diseño . Una responsabilidad frecuentemente implica una o varias operaciones. Si las entradas y salidas son descritas para las responsabilidades pueden usarse como un primer bosquejo de parámetros formales y valores de retorno de las operaciones.
2. Los requisitos especiales de cualquier clase del análisis que tenga una traza con la clase del diseño. Estos requisitos frecuentemente necesitan manejarse en el modelo del diseño, posiblemente incorporando algún mecanismo de diseño genérico o tecnología de base de datos.
3. Las interfaces que las clases del diseño necesitan proveer. Las operaciones de las interfaces deben proporcionar las clases del diseño.
4. La realización de caso de uso del diseño en las que participa la clase.

Las operaciones de una clase del diseño necesitan soportar todos los roles que las clases juegan en la realización de diferentes casos de uso. Estos roles se encuentran a través de la realización de los casos de uso, verificando si las clases y los objetos están incluidos en los diagramas y en la descripción de flujos de sucesos.

Identificar Atributos

Se identifican los atributos que requieren las clases del diseño y se describen usando la sintaxis del lenguaje de programación. Un atributo especifica una propiedad de una clase de diseño, frecuentemente está implicado y requerido por las operaciones de la clase. Los atributos son identificados cuando:

1. Se consideran los atributos sobre cualquier clase del análisis que tenga relación con las clases del diseño. Estos implican la necesidad de uno o más atributos de las clases del diseño.
2. Los tipos de atributos disponibles están restringidos por el lenguaje de programación.
3. Cuando se elige un tipo de atributo, se debe intentar reutilizar alguno existente.
4. Una instancia sencilla de un atributo no puede compartirse por varios objetos de diseño. Si es necesario, el atributo debe definirse en una clase por separado.
5. Si es difícil comprender una clase del diseño por sus atributos, éstos pueden separarse en clases independientes.
6. Si existen atributos complejos de una clase, se pueden ilustrar por separado en un diagrama de clases que muestre únicamente el apartado de los atributos.

Identificar Asociaciones y Agregaciones

Los objetos del diseño interactúan unos con otros en el diagrama de secuencia. Estas interacciones frecuentemente requieren asociaciones entre sus clases, por tanto se debe estudiar la transmisión de mensajes en los diagramas de secuencia para determinar qué asociaciones son necesarias. Las instancias de las asociaciones pueden usarse como referencia con otros objetos para agruparlos en agregaciones con el propósito de enviar mensajes. El número de relaciones entre las clases debe ser mínimo.

Se deben considerar los siguientes puntos cuando las asociaciones y agregaciones son identificadas y refinadas:

1. Considerar las asociaciones o agregaciones involucradas en las clases del análisis correspondiente. Algunas veces estas relaciones implican la necesidad de una o algunas relaciones (en el modelo del diseño) que involucra las clases del diseño.
2. Refinar la multiplicidad de las asociaciones, nombres de roles, clases de asociación, roles de ordenación, roles de cualificación, y asociaciones n-arias acordes al lenguaje de programación en uso.
3. Refinar la navegabilidad de las asociaciones. Considerar el diagrama de interacción en el que se emplean asociaciones. La dirección de la transmisión de mensajes entre los objetos del diseño implica que corresponda la navegabilidad de las asociaciones entre sus clases.

Identificar Generalizaciones

Las generalizaciones se deben usar con la misma semantica definida por el lenguaje de programación. Si el lenguaje de programación no soporta la generalización (o herencia), las asociaciones y/o agregaciones se deben usar en lugar de ésta para proveer delegación desde objetos de clases más especificas a objetos de las clases más generales.

Describir Métodos

Los métodos se usan durante el diseño para especificar cómo se realizan las operaciones. Por ejemplo un método puede especificar un algoritmo que se usa para realizar una operación. El método se especifica usando un lenguaje natural o usando un pseudocódigo. Sin embargo, los métodos generalmente no se especifican durante el diseño. En lugar, se crean durante la implementación usando directamente el lenguaje de programación. Esto es porque una misma persona debe diseñar e implementar la clase.

Si las herramientas de diseño generan código para los métodos de las clases del diseño, los métodos se especifican directamente en las herramientas del diseño usando el lenguaje de programación, pero esto es considerado en la implementación.

4.2.3.3 Describir Estados

Algunos objetos del diseño son estados controlados, por lo que su estado determina su comportamiento cuando reciben un mensaje. En estos casos es importante usar un diagrama de estados para describir las diferentes transiciones de estado de un objeto del diseño.

Diagrama de Estados

Un diagrama de estados muestra un conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación junto con los cambios que permite pasar de un estado a otro.

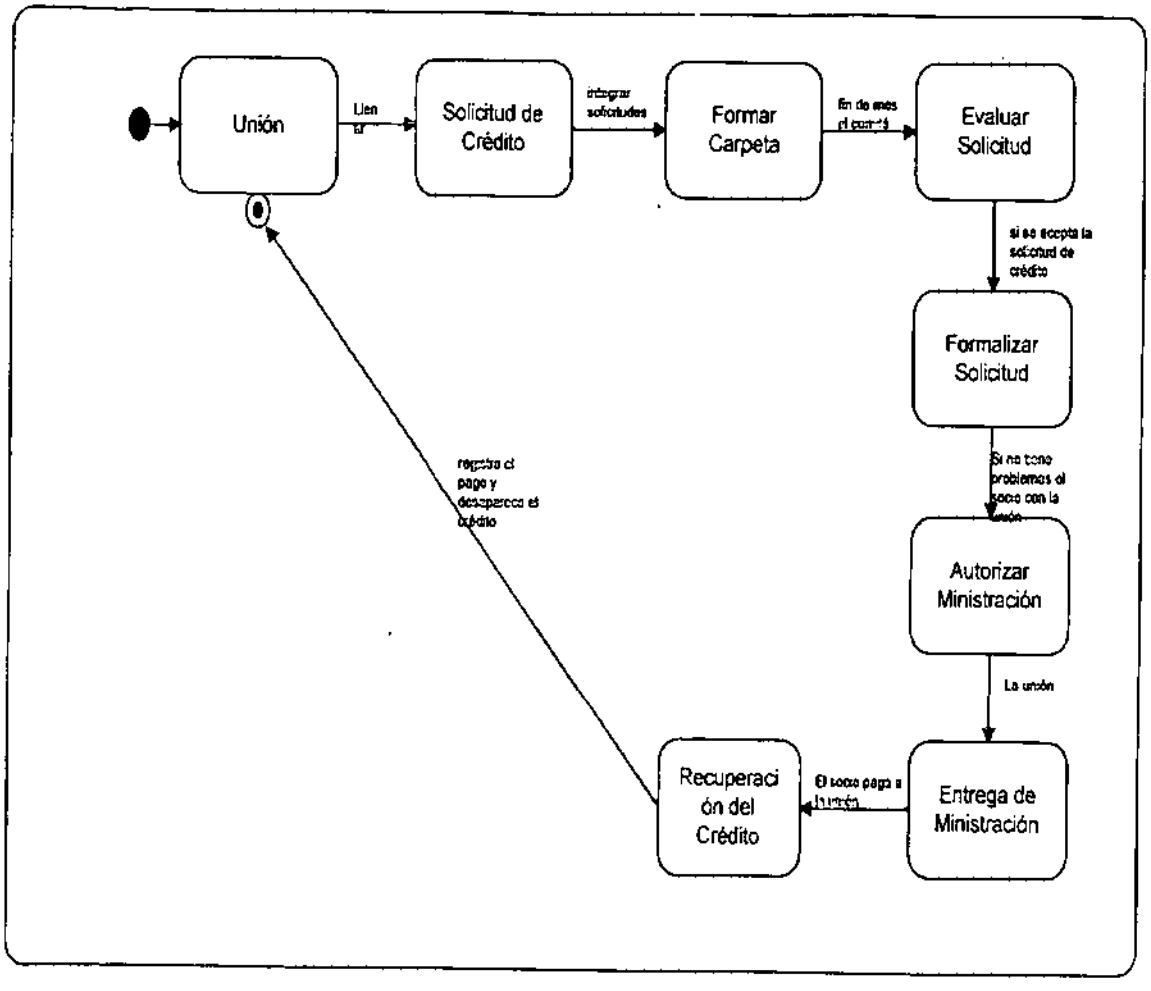
Un estado identifica un periodo de tiempo del objeto en el cual el objeto está esperando alguna operación, se representa mediante un rectángulo con los bordes redondeados; en un diagrama de estados también ocurren sucesos que pueden causar la transición de un estado a otro de un objeto; también puede representarse el momento en el cual se envían mensajes a otros objetos, esto se realiza mediante una linea punteada dirigida al diagrama de estados del objeto receptor del mensaje.

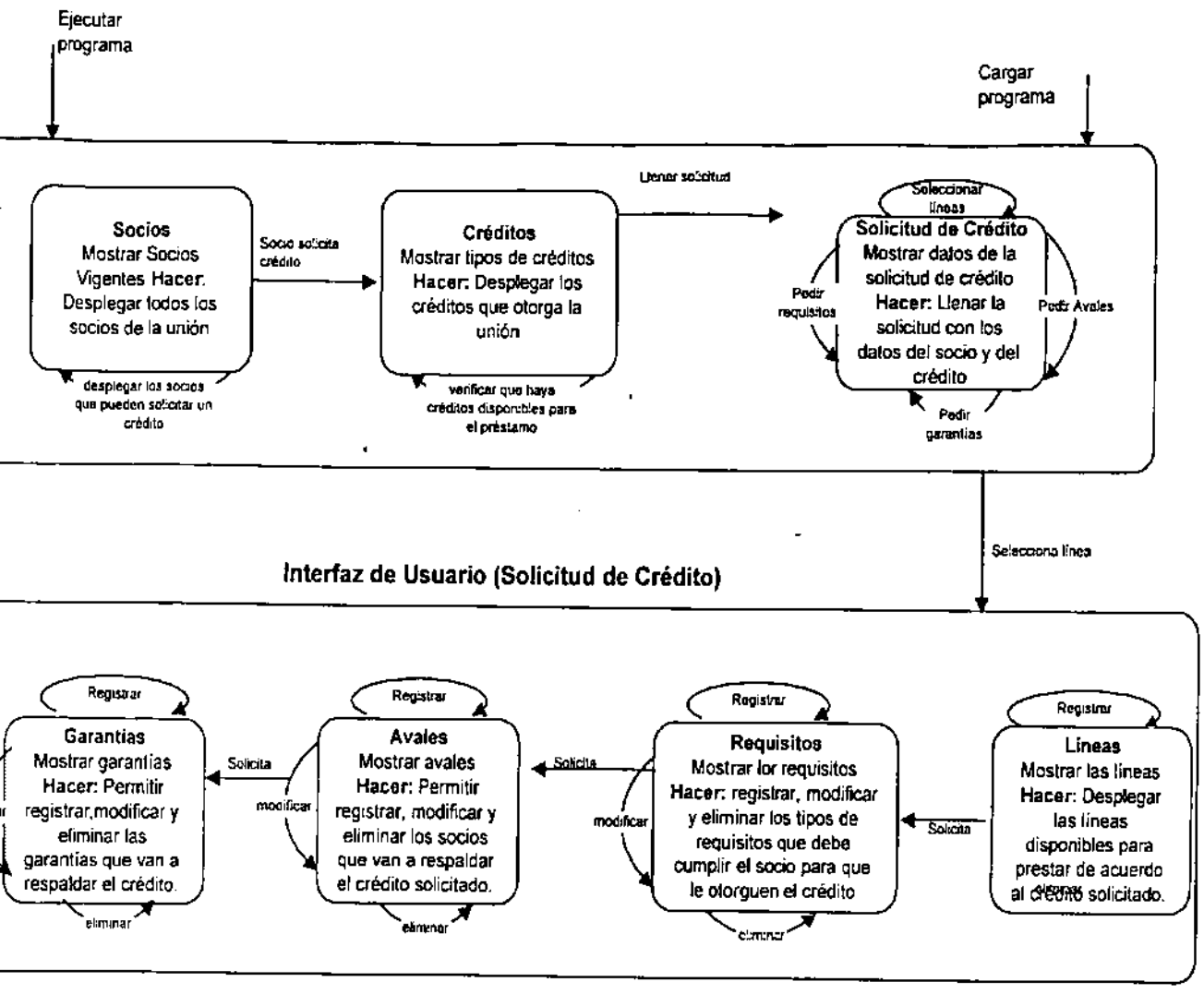
Ejemplo

El objeto Crédito cambia de diferentes estados, primero se debe llenar una solicitud de crédito por parte del socio vigente; la solicitud debe cumplir con ciertos requisitos como es solicitar garantías, el respaldo de socios vigentes que deseen ser avales y presentar la documentación que se le requiera dependiendo del tipo de crédito; posteriormente se debe formar una carpeta con las solicitudes pendientes para pasársela al comité de admisión para evaluarlas; si fue aceptada, el siguiente paso es formalizar el crédito con el socio. Una vez formalizado se debe autorizar la ministración y dársela al socio. Y finalmente devolver la recuperación, es decir recuperar el dinero del crédito.

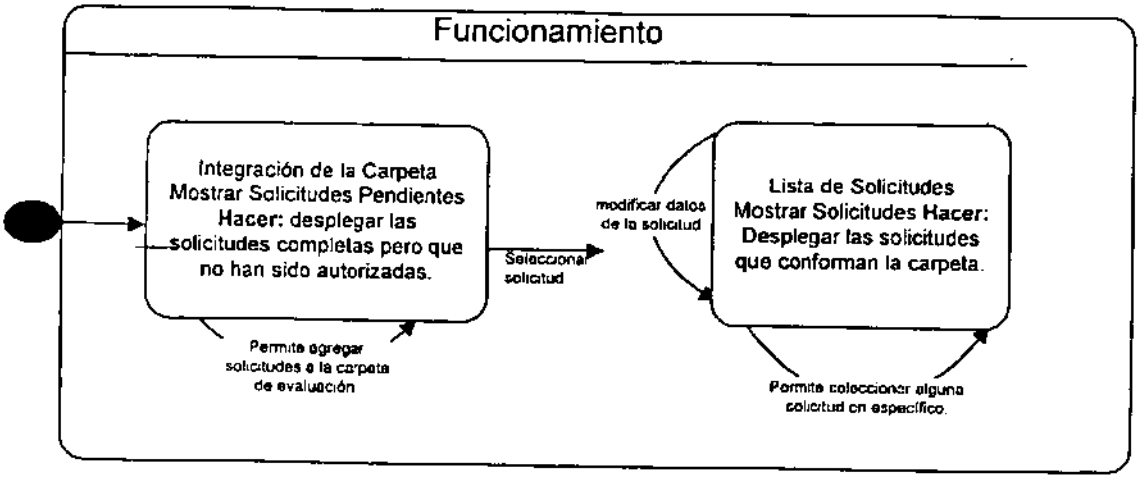
En el objeto crédito participan varias clases como son solicitud de crédito, socios vigentes, tipos de crédito, tipos de líneas, garantías, avales, requisitos solicitud ingreso, carpeta de solicitudes, ministraciones y recuperaciones.

Diagrama de Estado del Comportamiento de un Crédito

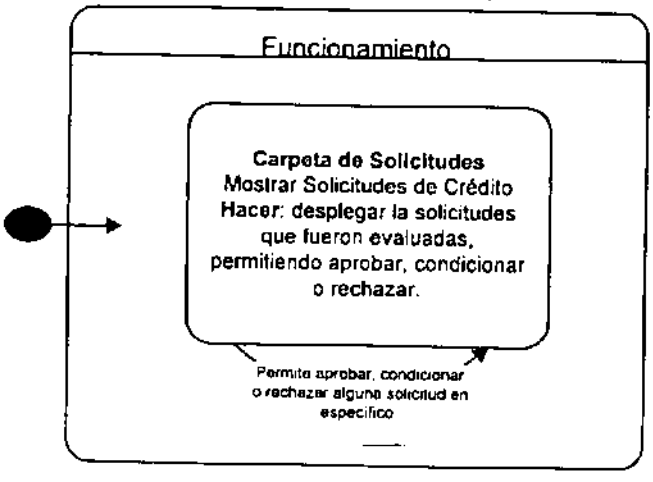



Ejecutar programa
Cargar programa
Socios
Mostrar Socios Vigentes Hacer: Desplegar todos los socios de la unión
Socio solicita crédito
Créditos
Mostrar tipos de créditos Hacer: Desplegar los créditos que otorga la unión
Llenar solicitud
Seleccionar líneas
Solicitud de Crédito
Mostrar datos de la solicitud de crédito Hacer: Llenar la solicitud con los datos del socio y del crédito
Pedir requisitos
Pedir Avales
Pedir garantías
desplegar los socios que pueden solicitar un crédito
verificar que haya créditos disponibles para el préstamo
Selecciona línea
Interfaz de Usuario (Solicitud de Crédito)
Registrar
Garantías
Mostrar garantías Hacer: Permitir registrar,modificar y eliminar las garantías que van a respaldar el crédito.
eliminar
Solicita
modificar
Registrar
Avales
Mostrar avales Hacer: Permitir registrar, modificar y eliminar los socios que van a respaldar el crédito solicitado.
eliminar
Solicita
modificar
Registrar
Requisitos
Mostrar los requisitos Hacer: registrar, modificar y eliminar los tipos de requisitos que debe cumplir el socio para que le otorguen el crédito
eliminar
Solicita
Registrar
Líneas
Mostrar las líneas Hacer: Desplegar las líneas disponibles para prestar de acuerdo al crédito solicitado.

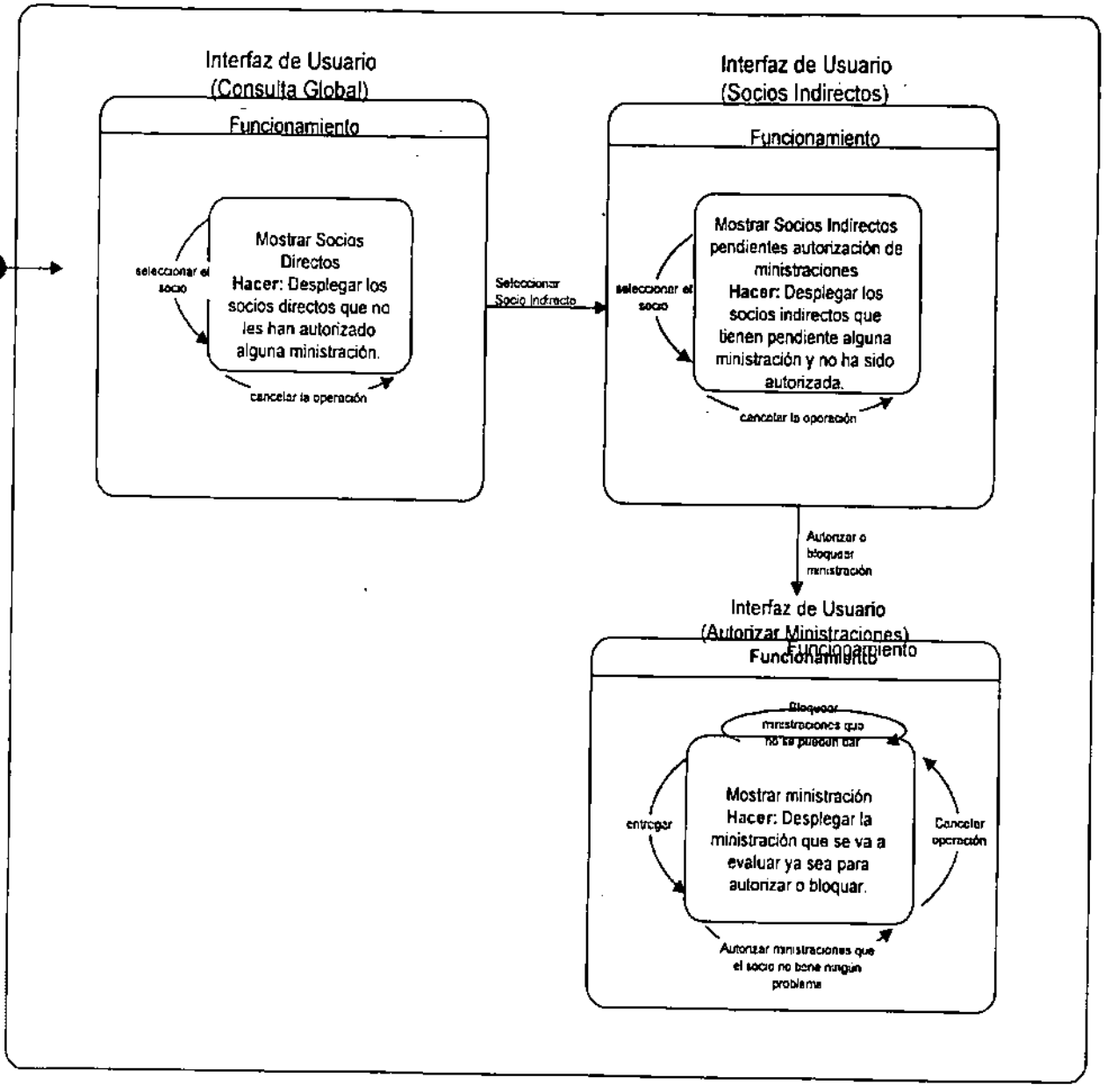
Interfaz de Usuario (Carpeta de Solicitudes)



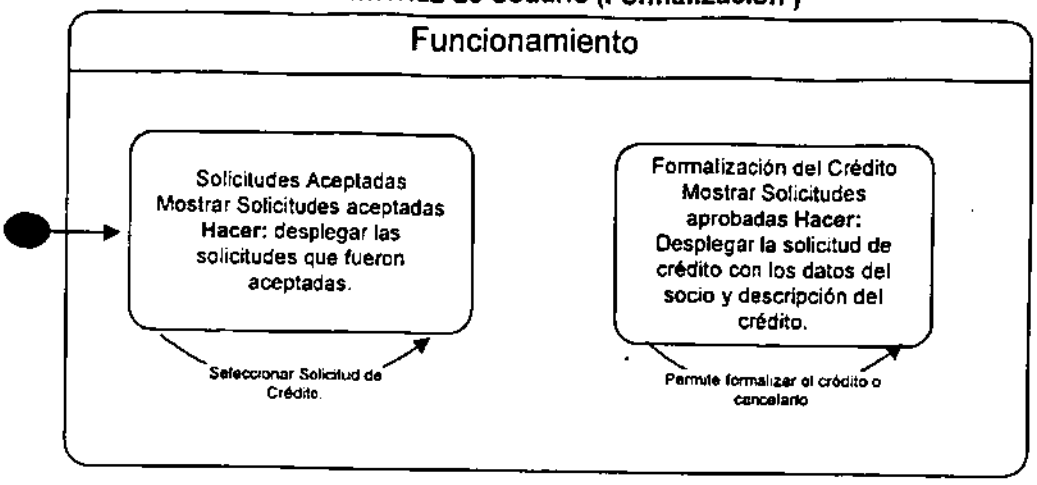
Interfaz de Usuario
(Evaluación de la Solicitud)



Interfaz de Usuario (Autorización de Ministraciones)



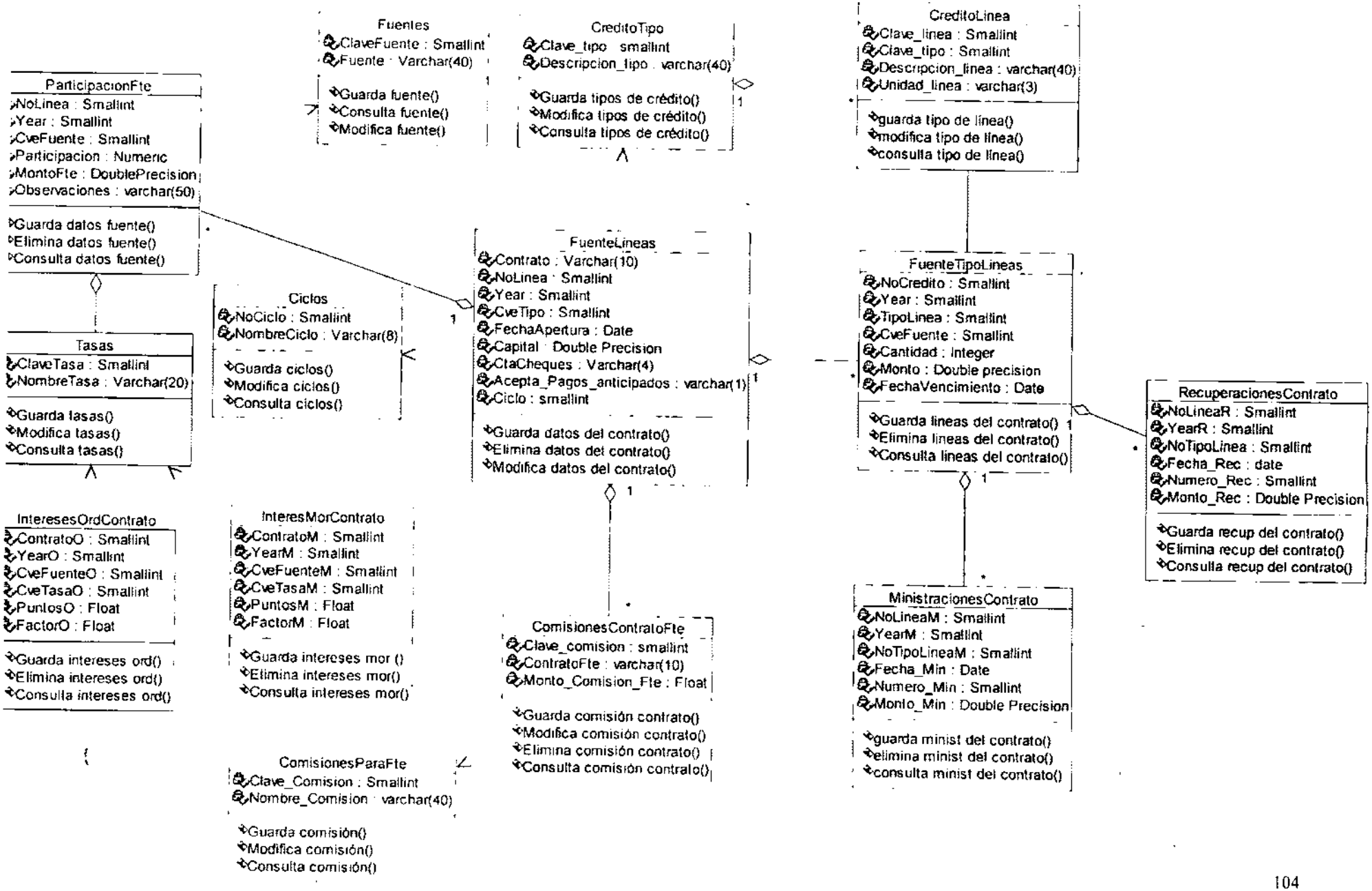
Interfaz de Usuario (Formalización)



4.2.3.4 Diagrama de Clases

Los subsistemas que contienen las clases del diseño frecuentemente participan en la realización de varios casos de uso; pero algunas operaciones, atributos, y asociaciones de una clase especifica pueden ser relevantes únicamente para la realización de un caso de uso. Por eso es importante coordinar todos lo requisitos que diferentes realizaciones de casos de uso imponen a una clase, a sus objetos y a los subsistemas que contiene. Para todo esto se utiliza un diagrama de clases conectado a la realización de un caso de uso.

Diagrama de Clases: del caso de uso Registrar un Contrato con las Fuentes



Capítulo 5. Implementación

5.2. Introducción

La implementación empieza con el resultado de diseñar e implementar el sistema en términos de componentes de tipo archivos de código fuente, binario, scripts, ejecutables y similares. Por ejemplo, se decide cómo las clases del diseño son implementadas en términos de componentes, cómo los archivos de código son ejecutables, cómo los componentes están organizados de acuerdo a la estructura y modularización de los mecanismos disponibles en el ambiente de implementación y en el lenguaje de programación en uso y cómo los componentes dependen uno de otro.

Su propósito es:

1. Planificar la integración del sistema en cada iteración, es decir, implementar el sistema en piezas pequeñas y manejables.
2. Distribuir el sistema a través del mapeo de componentes ejecutables en nodos en el modelo de instalación basado en las clases activas identificadas durante el diseño.
3. Implementar las clases y los subsistemas que se encontraron en el diseño. Las clases del diseño se implementan como componentes de tipo archivo que contengan código fuente.
4. Probar los componentes uno por uno y posteriormente integrarlos al compilarlos y enlazarlos en uno o más ejecutables.

Su objetivo es implementar el sistema planeando la integración por cada iteración como una secuencia de construcción. Primero, se esbozan los componentes principales y después se planean las integraciones que se necesitan para cada iteración. Por construcción, se debe describir la funcionalidad que se va a implementar y qué partes del modelo de implementación (subsistemas y componentes) pueden afectarse y al iniciar la implementación del siguiente constructor se debe tomar en cuenta los defectos del constructor anterior.

Durante la implementación se usan los subsistemas de implementación uno a uno con los subsistemas del diseño proveyendo las mismas interfaces. En esta actividad, se mantiene, refina y modifica la descripción de la arquitectura, la vista arquitectónica de la implementación y el modelo de construcción.

5.3. Implementación de la arquitectura

El propósito de implementar la arquitectura es esbozar el modelo de implementación y su arquitectura mediante:

- La identificación de los componentes más importantes, por ejemplo, los componentes ejecutables.
- La asignación de los componentes a los nodos.

Lo más importante durante el modelo de implementación es crear dentro de los subsistemas de implementación los componentes que implementen los subsistemas de diseño correspondientes,

refinar y actualizar la descripción y las vistas de la arquitectura de los modelos de implementación y de construcción.

Actividades
1. Identificar los componentes más importantes. (5.2.1.)
2. Identificar los subsistemas de implementación , sus dependencias, interfaces y contenido. (5.2.2.).
3. Integrar el sistema (5.2.3.).
4. Implementar una clase (5.2.4.).
5. Realizar la prueba de unidad (5.2.5.).

5.3.1. Identificar los componentes más importantes

Un componente es el empaquetamiento físico de los elementos del modelo, como las clases del diseño. Lo componentes incluyen:
1. <<executable>> es un programa que corre en un nodo.
2. <<file>> es un archivo que contiene código fuente o datos.
3. <<library>> es una librería estática o dinámica.
4. <<table>> es una tabla de base de datos.
5. <<document>> es un documento.

Cuando se crean componentes en un ambiente particular de implementación, se pueden modificar para reflejar que los componentes actuales están en su lugar.

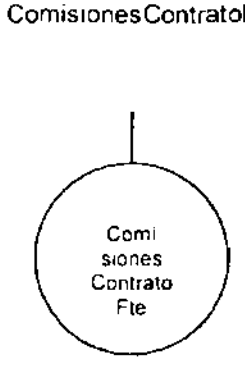
Sus características son que:
1. Tienen relación con los elementos del modelo que implementan.
2. Es común para un componente implementar varios elementos, por ejemplo varias clases.
3. Proporcionan las mismas interfaces que los elementos que implementan.
4. Pueden tener dependencias de compilación entre componentes, denotando qué componentes se requieren para compilar uno en específico.

Los componentes más Importantes
Lo más práctico es identificar primero los componentes más importantes, ya que esto va a facilitar el trabajo de implementación. Los componentes de tipo archivo es más fácil crearlos porque las clases se implementan empaquetándolos en ficheros de código fuente.

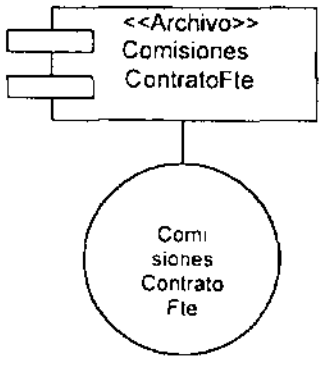
Ejemplo
Para identificar los componentes de tipo archivo y de tipo tabla del caso de uso registrar un contrato con las fuentes, se implementaron a partir de las clases del diseño; se les asignó una interfaz por cada componente asignándole el mismo nombre con el fin de identificarlo fácilmente.

Caso de Uso Registrar Contrato con las Fuentes

Modelo del Diseño



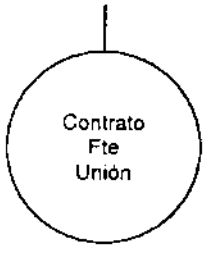
Modelo de Implementación



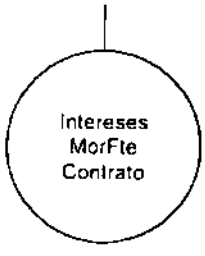
Modelo del Diseño

ParticipaciónFteContrato

ContratoFteUnión

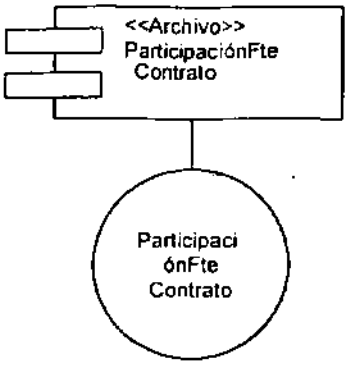


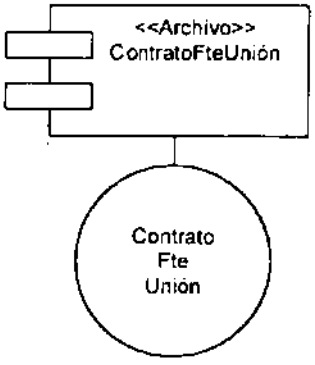
IntereresesMorFteContrato

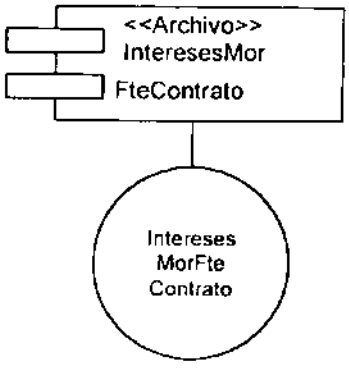


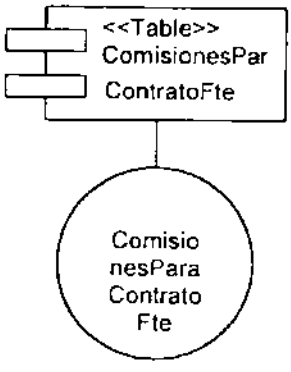
ComisionesPara
ContratoFte

Modelo de Implementación



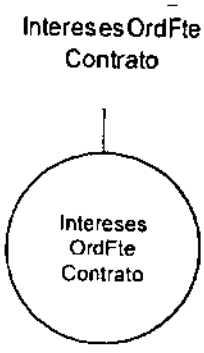


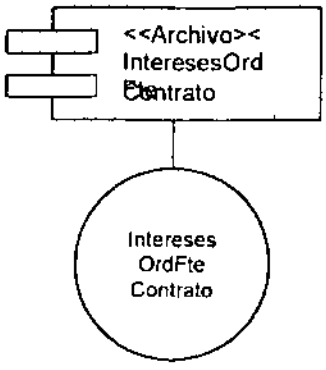


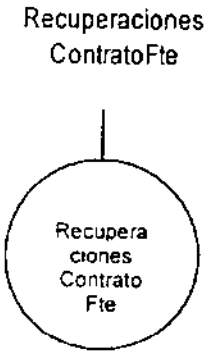


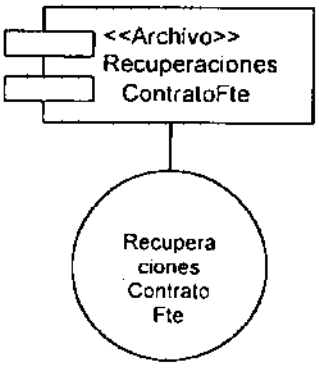
Modelo del Diseño

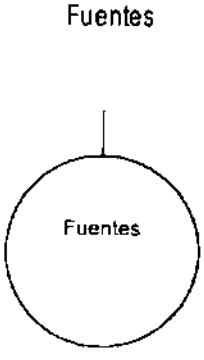
Modelo de Implementación

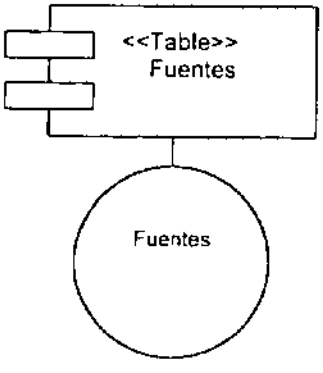


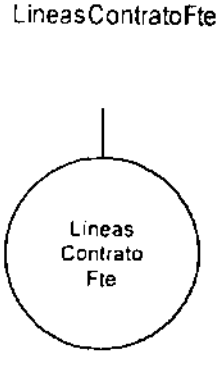


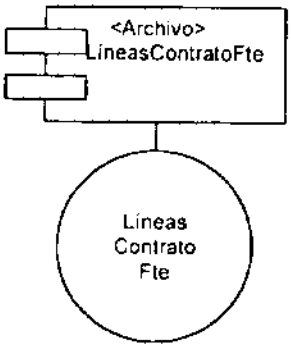




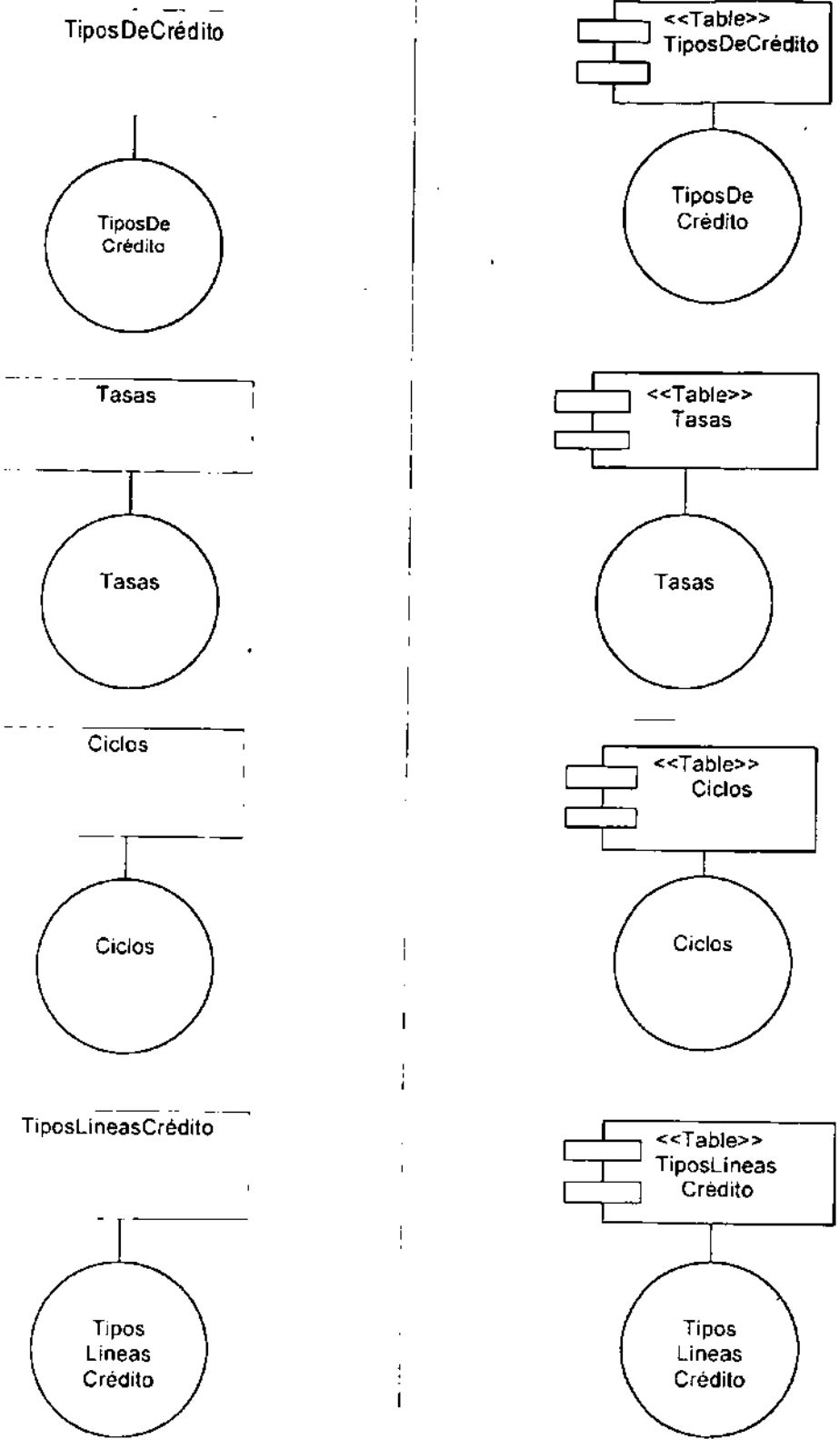








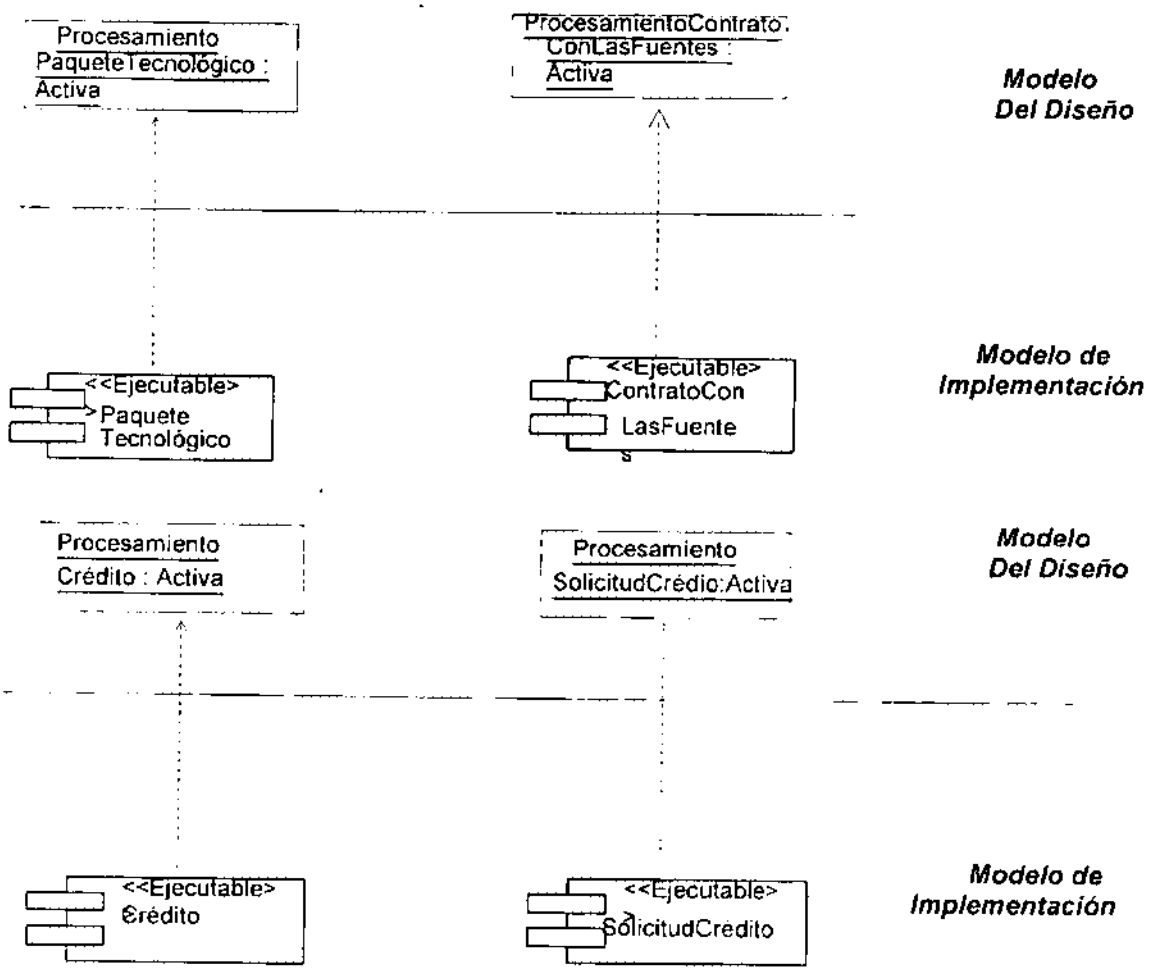
Modelo del Diseño



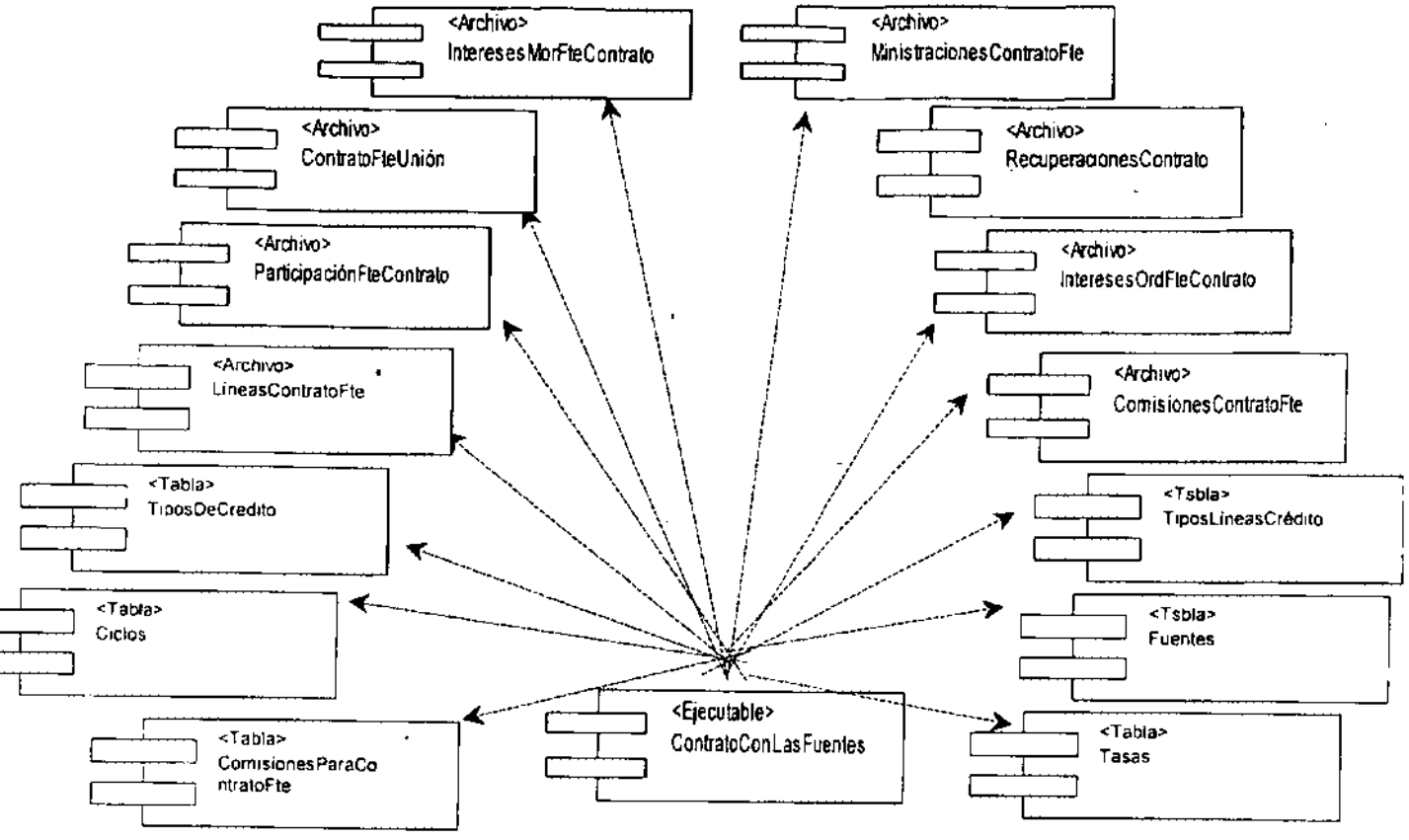
Modelo de Implementación

Identificar dependencia entre componentes
Se puede observar en el diseño y en el modelo del construcción que si hay objetos activos distribuidos en los nodos, los componentes siguen la traza de las clases activas correspondientes y deben desplegarse en los mismos nodos.

Para identificar los componentes ejecutables que se despliegan en los nodos se toman en cuenta las clases activas que se encontraron en el diseño y se asigna un componente ejecutable por cada clase activa. También se puede dar el caso de que se requiera identificar otros componentes de fichero o de código binario para crear componentes ejecutables.

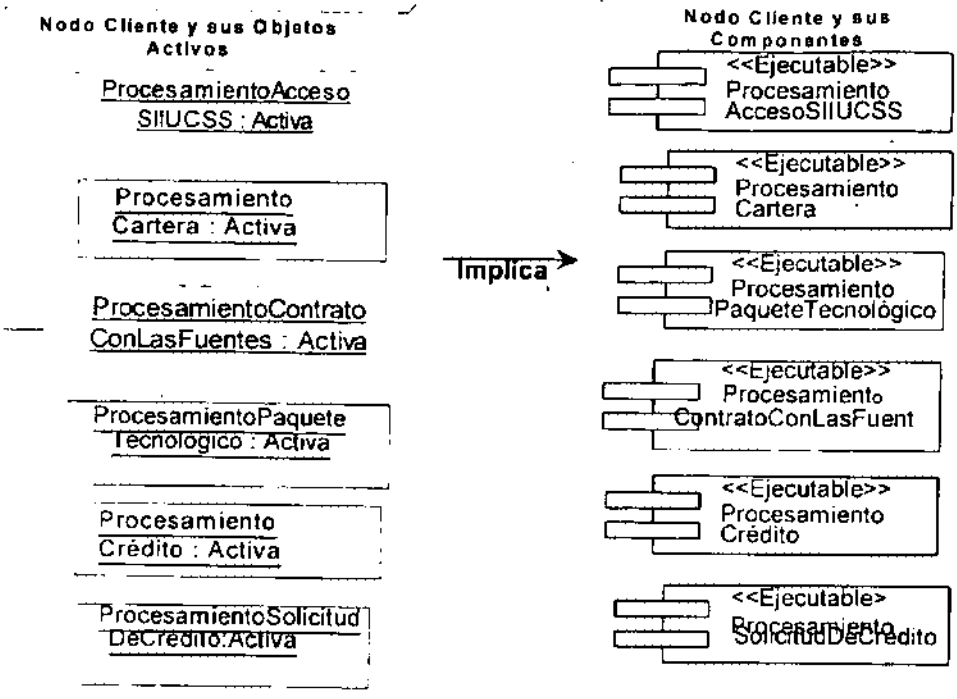


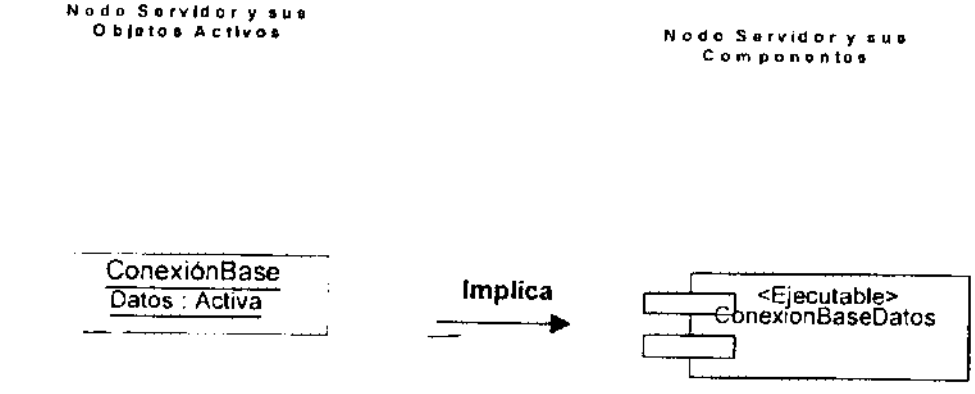
Se pudo observar que un componente puede depender de otro para crear un componente ejecutable; para identificarlos se tomo como base las clases activas identificadas en el modelo del diseño. Algunos componentes de tipo archivo se definieron como ejecutables porque dependen de otros componentes de tipo archivos y tablas, como es el caso de:
Componente Ejecutable ContratoConLasFuentes, está formado de los componentes ParticipaciónFteContrato, InteresesMorFteContrato, InteresesOrdFteContrato, RecuperacionesContratoFte, ComisionesContratoFte, ContratoFte, LíneasContratoFte. MinistracionesContratoFte, Ciclos, tipos decrédito, tipos líneas crédito, fuentes, tasas y comisiones para contrato fte.



Construcción de los Componentes en los Nodos

Una vez identificados los componentes ejecutables éstos se deben mapear a los nodos que les corresponde, tomando como base la distribución de las clases activas en los nodos del modelo del diseño.


Nodo Cliente y sus Objetos Activos
ProcesamientoAcceso SIIUCSS : Activa
Procesamiento Cartera : Activa
ProcesamientoContrato ConLasFuentes : Activa
ProcesamientoPaquete Tecnológico : Activa
Procesamiento Crédito : Activa
ProcesamientoSolicitud DeCredito:Activa
Implica
Nodo Cliente y sus Componentes
<<Ejecutable>> Procesamiento AccesoSIIUCSS
<<Ejecutable>> Procesamiento Cartera
<<Ejecutable>> Procesamiento PaqueteTecnológico
<<Ejecutable>> Procesamiento ContratoConLasFuent
<<Ejecutable>> Procesamiento Crédito
<<Ejecutable> Procesamiento SolicitudDeCredito


Nodo Servidor y sus Objetos Activos
ConexiónBase Datos : Activa
Implica
Nodo Servidor y sus Componentes
<Ejecutable> ConexionBaseDatos

5.3.2. Subsistemas de Implementación, sus dependencias, interfaces, contenido e implementación

Subsistemas de Implementación
Los subsistemas de implementación proveen una forma de organizar el modelo de implementación en piezas más sencillas de manejar. Un subsistema consiste de componentes, interfaces y otros subsistemas, sin embargo, puede implementar y proporcionar interfaces que representen la funcionalidad en términos de operaciones.

Es importante comprender que un subsistema de implementación es manifestado por un mecanismo de empaquetamiento concreto en un entorno de implementación determinado, como:
1. Un paquete en Java.
2. Un proyecto en Visual Basic.
3. Un directorio de ficheros en un proyecto de C++.
4. Un subsistema en un entorno de desarrollo integrado como Rational Apex.
5. Un paquete en una herramienta de modelado visual como Rational Rose.

Se refina un poco la semántica de un subsistema de implementación cuando se manifiesta en un ambiente de implementación específica, de hecho, los subsistemas de implementación están muy relacionados con los subsistemas de diseño son uno a uno. Un subsistema de diseño define:
1. Dependencia sobre otros subsistemas o interfaces de otros subsistemas.
2. Las interfaces que provee el subsistema.
3. Las clases del diseño u otros subsistemas del diseño dentro del subsistema deben proveer las interfaces.

Estos aspectos son importantes para el subsistema de implementación por las siguientes razones:
1. Debe definir dependencia hacia otros subsistemas de implementación o interfaces correspondientes.
2. Debe proporcionar las mismas interfaces.
3. Debe definir qué componentes o que otros subsistemas de implementación dentro del susbsistema debe proveer las interfaces proveídas por el mismo subsistema.

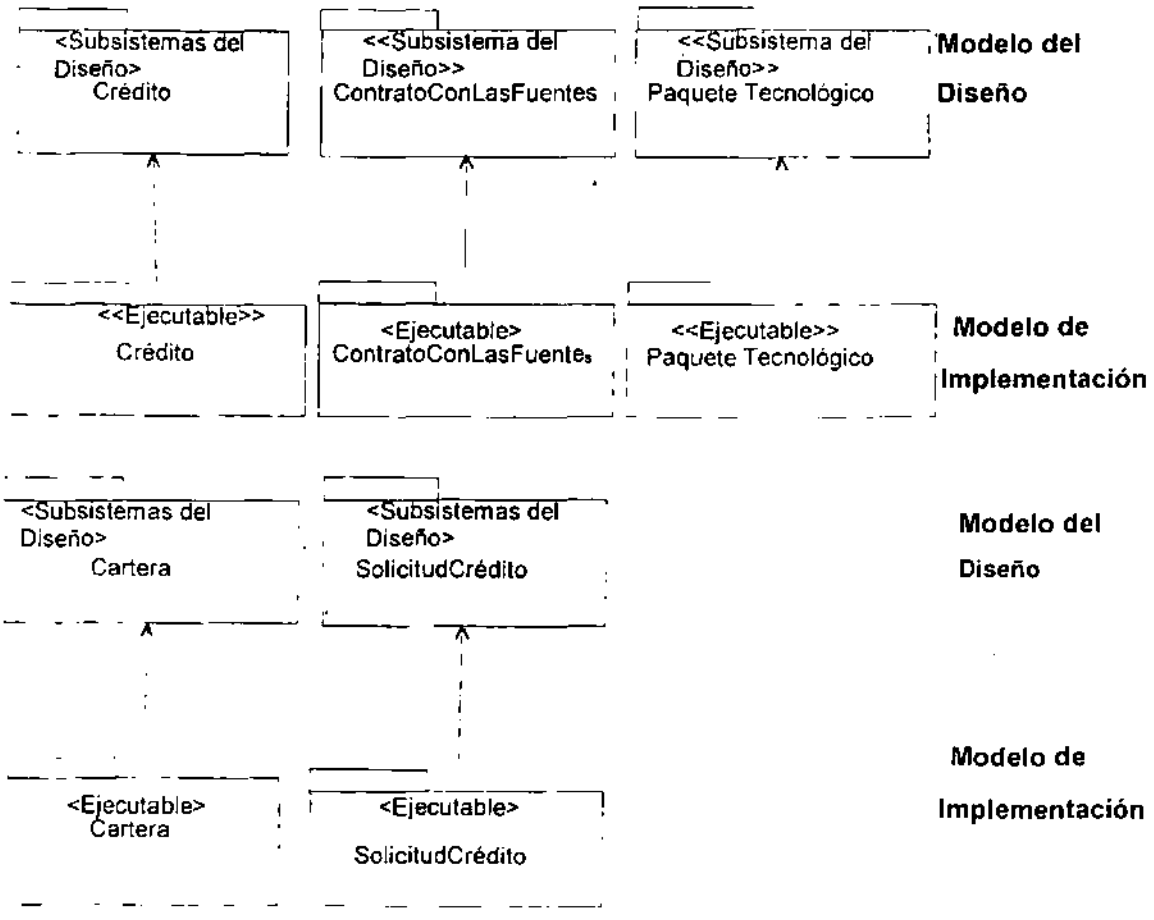
Implementar un Subsistema
El propósito de implementar un subsistema es asegurar que cumpla su rol en cada construcción para asegurar que los requisitos (escenarios y/o casos de uso) sean implementados en la construcción y los que afectan al subsistema se implementen correctamente por componentes u otros subsistemas dentro del subsistema.

Mantener el contenido de los subsistemas
Un subsistema cumple su propósito cuando los requisitos se han implementado correctamente a través de componentes dentro del subsistema.

Pero se puede dar el caso de que el contenido de un subsistema necesite refinarse; cuando se da este caso, cada clase del subsistema del diseño que se necesita en la construcción actual se debe implementar como un componente en el subsistema de implementación.

Si el subsistema de diseño contiene otros subsistemas que se requieren para la construcción que se está llevando a cabo, cada uno debe implementarse a través de un subsistema de implementación que deben estar dentro de un subsistema de implementación. Cada interfaz proporcionada por el subsistema de diseño que se requiere en la construcción debe ser proporcionada por el subsistema de implementación ya que éste debe proporcionar la interfaz.

Ejemplo
Los subsistemas de implementación están muy relacionados con los subsistemas del diseño por eso es uno a uno, como se puede observar en el siguiente ejemplo:

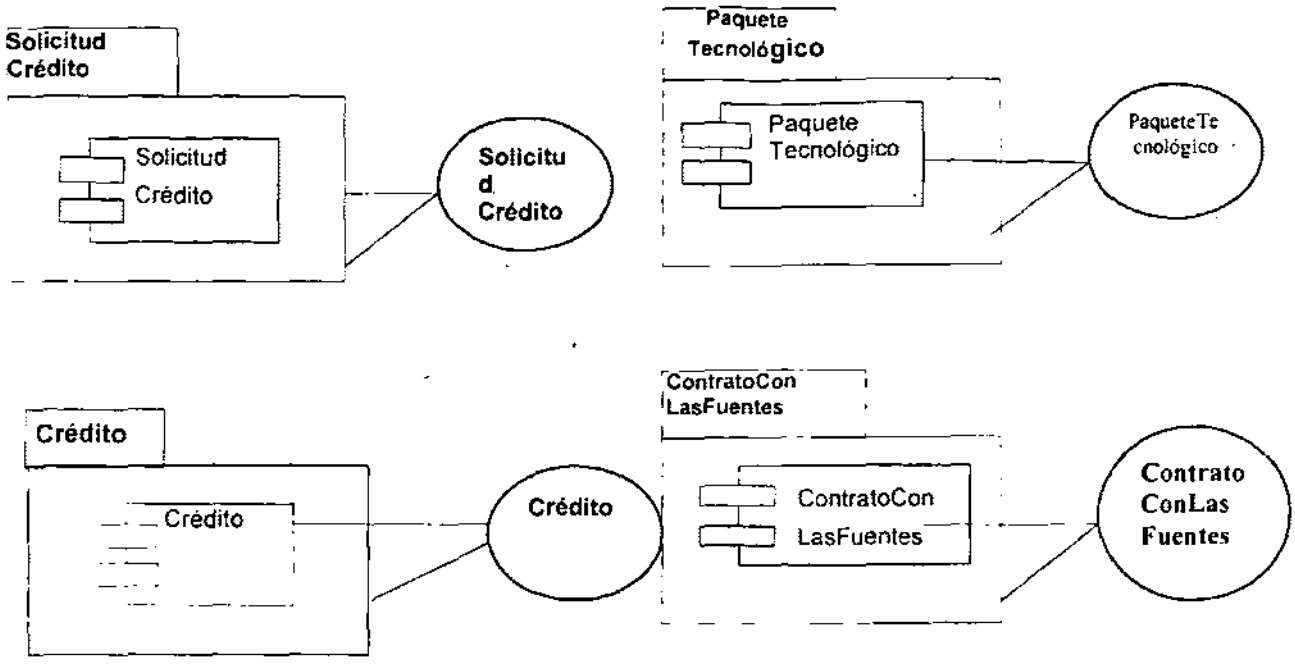


Interfaz
Las interfaces se usan en el modelo de implementación para especificar las operaciones implementadas por componentes y subsistemas de implementación. Sin embargo, éstas pueden tener "dependencias de uso" sobre interfaces.

Un componente que implementa una interfaz debe implementar todas las operaciones definidas por la interfaz de una manera correcta.

Hacer que los Componentes Provean las Interfaces
Es importante que los componentes provean las mismas interfaces que las clases del diseño que éste implementa.

Ejemplo
Generalmente cada subsistema provee su propia interfaz, se le asignó el mismo nombre para que sea más sencilla su identificación.



Integrar el Sistema
Sus objetivos son:

- Crear un plan de integración de construcciones que describa las construcciones que se necesitan por cada iteración y los requisitos de construcción.
- Integrar cada construcción antes de que se someta a pruebas de integración.

Stubs
Un Stub es un componente con un propósito especial que se usa para minimizar el número de componentes nuevos que se necesitan en cada versión (intermedia) nueva del sistema para simplificar los problemas y pruebas de integración .

Planear una Construcción
Se debe planificar los contenidos de una construcción independientemente si se cuenta con una construcción previa, con casos de uso o escenarios y con requisitos que se deben implementar.

El criterio que se debe seguir para una construcción es que:

1 Agregue funcionalidad a la construcción previa implementando casos de uso y/o escenarios completos. Las pruebas de integración están basadas en casos de uso y escenarios ya que es más fácil probar casos de uso completos que fragmentos de casos de uso.
2 No debe incluir demasiados componentes nuevos o modificados ya que es muy difícil integrarla y ejecutarle pruebas de integración. Pero es necesario que algunos componentes se puedan implementar como stubs para minimizar el número de nuevos componentes en la construcción.
3 Debe estar basada en la construcción anterior y debe empezar en las capas intermedias y de software del sistema; las construcciones subsecuentes se expanden hacia las capas generales y de aplicación específica.

Es muy importante considerar los puntos anteriores cuando se vayan a implementar los casos de uso y/o escenarios, de hecho, se recomienda llevar a cabo un compromiso con los que van a implementar para que tomen en cuenta las recomendaciones anteriores.

Por cada caso de uso implementado se debe:
1. Considerar el diseño del caso de uso identificando su realización por medio de las dependencias de traza en el modelo del diseño.
2. Identificar los subsistemas y las clases de diseño que participan en la realización del caso de uso del diseño.
3. Identificar los subsistemas y componentes de implementación que siguen la traza de los subsistemas y clases de diseño del modelo del diseño. Estos son muy importantes ya que van a ayudar a implementar el caso de uso.
4. Considerar el impacto de implementar los requisitos de los subsistemas de implementación y de los componentes de la construcción actual.

Con todo lo anterior se puede empezar a implementar los requisitos de los subsistemas y componentes de implementación en la construcción actual y llevar a cabo las pruebas individuales de cada unidad.

Hacer un Plan de Construcción
Es importante que el sistema se construya poco a poco y que en cada paso se realice una pequeña prueba. El resultado es construir una versión ejecutable del sistema y que ese ejecutable especifique una parte del sistema y se someta a una integración de pruebas antes de que se continúe con la siguiente construcción. En estos casos se recomienda llevar un control de versiones de tal forma que se puedan consultar construcciones anteriores.

Los beneficios que se obtienen de este enfoque incremental son:
1. Se puede contar con una versión ejecutable del sistema muy pronto. Esta integración de pruebas sirven para mostrar las características del sistema a los miembros del proyecto y a gente externa del sistema.
2. Los defectos son fáciles de localizar durante las pruebas de integración porque una pequeña parte del sistema es agregada o refinada en el constructor actual. En algunas ocasiones los defectos son relacionados (no siempre) con la siguiente construcción.
3. Las pruebas de integración tienden a ser más minuciosas que las pruebas del sistema completo porque se enfocan en partes más pequeñas y fáciles de manejar.

Integrar una construcción
Si la planeación se ha realizado correctamente es muy fácil integrar la construcción. Esto se hace recopilando las versiones correctas de los subsistemas de implementación y de los componentes compilándolos y ligándolos para generar una construcción. El resultado de la construcción está expuesto a pruebas de integración y a pruebas del sistema.

Integración Incremental
La integración incremental se utiliza para la integración del sistema. Usando el contexto de desarrollo iterativo en cada iteración resulta al menos un constructor. Sin embargo la funcionalidad implementada en una iteración específica, frecuentemente es compleja al ser integrada en una sola construcción; En lugar, se puede crear una secuencia de construcciones en una iteración y cada iteración resultará un incremento al sistema.

Un plan de integración de construcciones va a describir la secuencia de construcciones requeridas en una iteración. Mas específicamente, por cada construcción un plan debe describir:
1. La funcionalidad que se espera sea implementada en la construcción. Consiste de una lista de casos de uso y/o escenarios o parte de ellos.
2. Las partes del modelo de implementación son afectadas por la construcción. Es una lista de subsistemas y componentes requeridos para implementar la funcionalidad supuesta por la construcción.

5.3.3. **Implementar una Clase**

El propósito de implementar una clase es implementar una clase del diseño en un componente de tipo fichero. Incluye:

1. Un bosquejo de un componente fichero que contenga código fuente.
2. Generar código fuente a partir de las clases de diseño y la relación en las que participan.
3. Implementar las operaciones de las clases de diseño en términos de métodos.
4. Asegurar que los componentes provean las mismas interfaces que las clases de diseño.

Un bosquejo de los componentes fichero

El código fuente que implementa una clase de diseño reside en un componente fichero. Es común implementar varias clases de diseño en un mismo componente fichero, por ejemplo, cuando se usa Java se crea un componente fichero de tipo ".java" por cada clase implementada. En general, componentes de tipo fichero deben facilitar la compilación, instalación y mantenimiento del sistema.

Generar código a partir de una clase del diseño

Durante el diseño, las clases del diseño y sus relaciones se describen usando la sintaxis del lenguaje de programación, lo que permite que la generación de partes del código fuente que implementan la clase sea muy fácil. Esto también se cumple para las operaciones, relaciones y atributos de la clase en las cuales participan.

Es delicado generar código a partir de asociaciones y agregaciones, esto depende mucho del lenguaje de programación en uso.

Implementar Operaciones

Cada operación definida por la clase del diseño debe ser implementada; a menos que sea virtual (o abstracta) y sea implementada por descendientes (como subtipos) de la clase. Se usa el término método para denotar la implementación de operaciones, por ejemplo los métodos en Visual Basic, las funciones miembro en C++ y los métodos en java.

Implementar una operación involucra la elección de un algoritmo y una estructura apropiada de datos. Los métodos se especificaron durante el diseño de las clases utilizando un lenguaje natural o pseudocódigo.

5.3.4. **Realizar pruebas de unidad**

El propósito de realizar la prueba de unidad es probar los componentes implementados como unidades individuales.

Tipos de pruebas unitarias:

1. La prueba de especificación o prueba de caja negra.- verifica el comportamiento de la unidad observable externamente.
2. La prueba de estructura o prueba de caja blanca.- verifica la implementación interna de la unidad.
3. La prueba de integración y sistema para asegurar que todos los componentes se comporten correctamente cuando se integran.

También se pueden realizar otro tipo de pruebas como la prueba de rendimiento, de utilización de memoria, carga y capacidad.

Realizar pruebas de especificación

Las pruebas de especificación se realizan para verificar el comportamiento del componente sin considerar cómo se implementa el comportamiento en el componente. Las pruebas de especificación indican la salida que el componente devolverá cuando se le da una determinada entrada en un determinado estado.

Realizar pruebas de estructura

Realizar pruebas de estructura sirven para verificar que los componentes funcionen internamente como se queria y asegurar que se realicen todas las pruebas para todo el código.

Conclusiones

Tomando en cuenta las razones por las que se inició este trabajo que es proporcionar una guía que permita aprender el Proceso Unificado mediante una forma sencilla y rápida desde la definición de los requerimientos hasta la implementación, ejemplificándolo con un sistema real que se desarrolló en el Instituto de Ingeniería, se puede concluir que:

1. Este trabajo representa la síntesis de los flujos de trabajo fundamentales (identificación de requerimientos del cliente, modelado de requerimientos como casos de uso, análisis, diseño e implementación) del libro The Unified Software Development Process, identificando las actividades más importantes que se toman en cuenta para desarrollar los flujos de trabajo de una iteración de un producto software sin considerar la definición de roles.

2. Se aplicó la reingeniería a un sistema ya existente. Se ejemplificó con el Sistema Integral de Uniones de Crédito para el Sector Social porque se conocía su funcionamiento ya que se participó en la creación del mismo y ya estaba por finalizarse cuando se inició está investigación.

3. Se documentó parte del Sistema Integral de Uniones de Crédito para el Sector Social, utilizando el proceso unificado y UML. Aprovechando está investigación y la existencia del sistema, se decidió hacer algunas propuestas de mejora.

4. Los casos de uso fueron de gran ayuda para elaborar el manual de operación del sistema ya que se fue realizando a la par con los casos de uso.

5. Este trabajo se puede utilizar como una guía para que se aplique el proceso unificado en la industria de la ingeniería de software. Se pretende que este trabajo sea de gran utilidad para las empresas que se dedican a desarrollar software orientado a objetos ya que el Proceso Unificado es la unificación de las metodologías orientadas a objetos y de la experiencia de empresas y personas que se dedican a desarrollar productos software.

6. Si se llevan a cabo las actividades de los flujos de trabajo como lo indica el proceso unificado, se utiliza un lenguaje de modelado unificado (UML) y la herramienta de modelaje Rational Rose se va a lograr que el producto software cumpla con las necesidades de los clientes y usuarios del sistema.

7. Se aprendió la metodología y se va a aplicar en la vida real en proyectos posteriores.

Propuestas de Mejora al Sistema Integral de Uniones de Crédito del Sector Social

De acuerdo al objetivo de esta investigación se realizaron las actividades de los flujos de trabajo del Proceso Unificado utilizando el Sistema Integral de Uniones de Crédito del Sector Social, lo que no coincidía a lo existente se decidió documentarlo y presentarlo como una propuesta.

A continuación se mencionan las mejoras que se le pueden hacer al Sistema Integral de Uniones de Crédito del Sector Social en versiones posteriores.

Propuestas:

1. Se identificó el componente Unión de Crédito que no existía en la arquitectura anterior. En el componente herramientas se registra y consulta la información relacionada a la unión de crédito, pero por la importancia de ésta se recomienda crear el componente unión de crédito que se va a encargar de administrar toda la información relacionada a la unión, por ejemplo, datos generales de la unión, acciones y usuarios del sistema.

2. Desaparecer el componente herramientas y el contenido de éste integrarlo en los componentes ya existentes. El componente herramientas puede desaparecer porque la información de vida larga (algunas clases entidad) se puede ir registrando conforme se vaya necesitando. Por ejemplo, puede darse el caso de que al momento de registrar un contrato con las fuentes no se haya registrado determinada comisión, línea, fuente, crédito, etc. que se requiere en ese momento, por lo tanto, no se puede registrar el contrato y se tiene que cancelar el proceso e irse al componente herramientas, capturarlo y volver a iniciar todo el proceso nuevamente.

3. Al definir la interfaz de usuario, se propone que en las mismas pantallas se permita registrar y hacer uso de la información de algunas clases entidad (catálogos) que ya existe. En la interfaz de usuario se debe buscar un componente del lenguaje de programación que permita registrar el dato que se requiere y utilizar la información que ya existe.

4. Desaparecer el componente fuentes e integrarlo al componente crédito. El componente fuentes se puede incorporar al componente crédito porque tiene relación con un crédito. Por ejemplo, cuando un socio va a solicitar un crédito a la unión, el encargado de crédito verifica el contrato con las fuentes si tiene recursos en la línea que desea el socio y si hay un paquete tecnológico que coincida con el contrato por el tipo de crédito, las líneas y el ciclo. Para que se pueda otorgar un crédito debe existir un paquete tecnológico y un contrato con las fuentes con el mismo tipo de crédito, el ciclo y las líneas y a partir de estos generar un crédito.

5. Crear los componentes contabilidad, inversiones, mantenimiento al sistema, ayuda y caja.

Bibliografía

Ivar Jacobson, Grady Booch, James Rumbaugh, 1998. The Unified Software Development Process. Edit. Addison Wesley.

http://www.ctima.uma.es/turing

http://www.abits.com/Newtk/sepoct99

http://www.rational.com

http://www.objectsbydesign.com/tools/modeling_tools_sp.html

http://www.geocities.com/ResearchTriangle/7303/uml/

http://www.ctima.uma.es/turing/turing8/hardsoft/uml/uml.htm#d1

http://cgi.omg.org/news/pr99/UML_2001_CACM_Oct99_p29-Kobryn.pdf

http://www.cs.ualberta.ca/~pfiguero/soo/metod/requerimientos.html#act4

http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html

http://www.dsic.upv.es/~uml/