



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

**PROPUESTA DE ANÁLISIS Y DISEÑO BASADA EN UML Y UWE
PARA LA MIGRACIÓN DE ARQUITECTURA DE *SOFTWARE*
CENTRALIZADA HACIA INTERNET**

Haroldo Fernando Pérez Hernández
Asesorado por el Ing. José Francisco López Rodríguez

Guatemala, noviembre de 2010

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

**PROPUESTA DE ANÁLISIS Y DISEÑO BASADA EN UML Y UWE
PARA LA MIGRACIÓN DE ARQUITECTURA DE SOFTWARE
CENTRALIZADA HACIA INTERNET**

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

HAROLDO FERNANDO PEREZ HERNANDEZ
ASESORADO POR EL ING. JOSÉ FRANCISCO LOPEZ
RODRIGUEZ

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2010

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA



NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Inga. Glenda Patricia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Luis Pedro Ortiz de León
VOCAL V	P. A. José Alfredo Ortiz Herincx
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADORA	Inga. Virginia Victoria Tala Ayerdi
EXAMINADOR	Ing. Marlon Antonio Pérez Turk
EXAMINADOR	Ing. Cresencio Gertrudis Chan Canek
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

PROPUESTA DE ANÁLISIS Y DISEÑO BASADA EN UML Y UWE PARA LA MIGRACIÓN DE ARQUITECTURA DE *SOFTWARE* CENTRALIZADA HACIA INTERNET,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, el 22 de julio de 2009.

Haroldo Fernando Pérez Hernández

Guatemala, 24 de mayo de 2,010

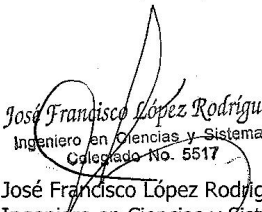
Ingeniero
Carlos Azurdia
Coordinador
Comisión de Revisión de Trabajos de Graduación
Carrera de Ingeniería en Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Respetable Ingeniero:

Por este medio hago de su conocimiento que he participado como Asesor en la realización del trabajo de graduación titulado "**Propuesta de análisis y diseño basada en UML Y UWE para la migración de arquitectura de software centralizada hacia internet.**" que el estudiante **Haroldo Fernando Pérez Hernández** desarrollara como proyecto de graduación para optar al título de Ingeniero en Ciencias y Sistemas. Este trabajo ha sido revisado y finalizado bajo los objetivos planteados inicialmente, dando todo mi apoyo y experiencia lo que me permite hacerme responsable de la calidad y del contenido completo que este incluye.

Agradeciendo la oportunidad brindada de poder colaborar con la educación e investigación universitaria, doy como finalizada mi intervención para que le sea dado el trámite correspondiente al documento.

Atentamente,


José Francisco López Rodríguez
Ingeniero en Ciencias y Sistemas
Colegiado No. 5517

José Francisco López Rodríguez
Ingeniero en Ciencias y Sistemas
No. Colegiado: 5517



Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 26 de Mayo de 2010

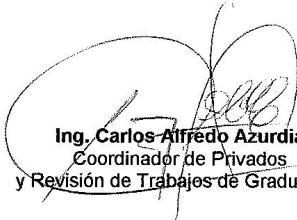
Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **HAROLDO FERNANDO PEREZ HERNANDEZ**, titulado: **"PROPUESTA DE ANALISIS Y DISEÑO BASADA EN UML Y UWE PARA LA MIGRACION DE ARQUITECTURA DE SOFTWARE CENTRALIZADA HACIA INTERNET"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,


Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación



E
S
C
U
E
L
A

D
E

C
I
E
N
C
I
A
S

Y

S
I
S
T
E
M
A
S


UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA



FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado **“PROPUESTA DE ANÁLISIS Y DISEÑO BASADA EN UML Y UWE PARA LA MIGRACIÓN DE ARQUITECTURA DE SOFTWARE CENTRALIZADA HACIA INTERNET”**, presentado por el estudiante HAROLDO FERNANDO PÉREZ HERNÁNDEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

“ID Y ENSEÑAD A TODOS”


Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería Ciencias y Sistemas



Guatemala, 04 de noviembre 2010

Universidad de San Carlos
de Guatemala



Facultad de Ingeniería
Decanato

Ref. DTG.355.2010

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **PROPUESTA DE ANÁLISIS Y DISEÑO BASADA EN UML Y UWE PARA LA MIGRACIÓN DE ARQUITECTURA DE SOFTWARE CENTRALIZADA HACIA INTERNET**, presentado por el estudiante universitario **Haroldo Fernando Pérez Hernández**, procede a la autorización para la impresión del mismo.

IMPRÍMASE.

A large, handwritten signature in black ink, appearing to read "Ing. Murphy Olympo Paiz Recinos".

Ing. Murphy Olympo Paiz Recinos
DECANO

Guatemala, noviembre de 2010



/cc

ACTO QUE DEDICO A:

DIOS

Por darme la sabiduría y fortaleza para lograr una meta más. Es la fuente de mi inspiración y es su voluntad este acto.

MIS PADRES

Por estar siempre a mi lado y apoyarme en todo momento.
No hubiera sido posible sin ustedes.

MI FAMILIA

Por su apoyo incondicional cuando fue necesario en las buenas y en las malas.

AGRADECIMIENTOS A:

MI ASESOR

Ingeniero José Francisco López

Por dedicarme su tiempo y experiencia para que este trabajo llene las expectativas profesionales necesarias.

MIS AMIGOS

Por su amistad, apoyo y compartir su tiempo a lo largo de esta carrera y el transcurso de mi vida, para no sentirme solo en este camino formativo lleno de retos y dificultades, he aprendido mucho de ustedes.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES	V
GLOSARIO	VII
RESUMEN	XI
OBJETIVOS	XIII
INTRODUCCIÓN	XV
1. MARCO TEÓRICO	1
1.1. Fundamentos de UML y UWE	1
1.1.1. Introducción al UML	1
1.1.2. Reglas del UML.....	3
1.1.3. Diagramas	3
1.1.3.1. Diagramas recomendados	5
1.1.4. Introducción a UWE.....	6
1.1.4.1. Características principales	7
1.1.4.2. Fases de desarrollo	7
1.2. Reingeniería de <i>software</i>	9
1.3. Ingeniería inversa	12
1.4. Migración de sistemas	13
1.4.1. Objetivos de migración	14
1. 5. Aplicación <i>web</i>	15
1.5.1. Características	16
1.5.2. Tipos.....	16
1.5.3. Condiciones para el desarrollo	17

1.6. Sistemas de información centralizados 18

2. ANÁLISIS Y REDISEÑO DEL SISTEMA CENTRALIZADO

USANDO UML 21

2.1. Análisis del sistema actual con ingeniería inversa.....21

 2.1.1. Especificaciones funcionales del sistema actual.....23

 2.1.2. Esquema lógico de datos24

2.2. Esquema conceptual y modelado visual con UML26

 2.2.1. La vista estática26

 2.2.1.1. Identificación de conceptos28

 2.2.1.2. Definición en UML del modelado conceptual.....29

3. REDISEÑO DEL SISTEMA DE INFORMACIÓN CON UWE 33

3.1. Diseño conceptual con UWE 33

 3.1.1. Perfiles de usuario34

 3.1.2. Escenarios de uso.....35

 3.1.2.1. Modelo de proceso de flujo de trabajo.....35

 3.1.2.2. El modelo de secuencia de tareas.....35

 3.1.2.3. El modelo de ambiente físico.....36

 3.1.3. Diagrama de casos de uso36

 3.1.4. Diseño conceptual sobre tres capas38

 3.1.4.1 .Capa de presentación39

 3.1.4.2. Capa de negocios.....40

 3.1.4.3. Capa de datos42

3.2. Diseño lógico42

 3.2.1. Organización de las estructuras lógicas.....43

 3.2.2. Transición del diseño conceptual al diseño lógico43

 3.2.3. Creación del diseño lógico44

3.2.3.1. Identificación de objetos	45
3.2.3.2. Identificación de los comportamientos	46
3.2.3.3. Identificación de atributos	48
3.2.3.4. Identificación de relaciones.....	49
3.2.4. Diagramas de clases	50
3.2.4.1. Métodos de las clases en cada una de las tres capas....	50
4. REDEFINICIÓN CON UNA ARQUITECTURA WEB USANDO	
UWE	53
4.1. Modelando el proceso de negocio <i>web</i> con UWE.....	53
4.1.1. Integración de procesos en el modelo de navegación.....	54
4.1.1.1. Modelado del espacio navegacional	54
4.1.1.2. Estructura del modelado navegacional	56
4.1.2. Afinando el modelado de procesos	57
4.1.2.1. Modelo estructural del proceso	58
4.1.2.2. Modelo de comportamiento del proceso	58
4.1.3. Soporte de procesos en el modelo de presentación	59
4.1.3.1. Estructura visual	60
4.1.3.2. Interfaz de usuario	61
5. CASO DE ESTUDIO	63
5.1. Descripción caso de estudio. Sistema de contratos de personal	63
5.2. Solución de la migración utilizando el método propuesto	64
5.2.1. Análisis y diseño del sistema actual centralizado	66
5.2.2. Rediseño del sistema de información con UWE.....	68
5.2.2.1. Nuevos escenarios de uso.....	69
5.2.2.2. Especificación y diagramas de casos de uso.....	69
5.2.2.3. Diagrama de clases	78

5.2.2.4. Diagramas de secuencia	79
5.2.2.5. Diagramas de colaboración	82
5.2.3. Modelando el sistema con una arquitectura <i>web</i>	84
5.2.3.1. Integración de procesos en el modelo de navegación.....	84
5.2.3.1.1. Modelado del espacio navegacional.....	84
5.2.3.1.2. Estructura del modelado navegacional.....	87
5.2.3.2. Afinando el modelado de procesos	91
5.2.3.2.1. Modelo estructural del proceso.....	91
5.2.3.2.2. Modelo de comportamiento del proceso.....	94
5.2.3.3. Soporte de procesos en el modelo de presentación.....	98
5.2.3.3.1. Estructura visual	98
5.2.3.3.2. Interfaz de usuario	99
5.2.4. Diagramas de componentes y despliegue	102
5.3. Prácticas adecuadas	105
CONCLUSIONES	107
RECOMENDACIONES	109
REFERENCIAS BIBLIOGRÁFICAS	111
BIBLIOGRAFÍA	113

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Impacto de cambio en el <i>Software</i> con respecto al cambio en las reglas de negocio.....	XVIII
2. Visualización del concepto.....	27
3. Asociación y multiplicidad.....	30
4. Concepto y atributos.....	31
5. Conceptualizando procedimientos.....	31
6. Arquitectura de tres capas.....	39
7. Objetos.....	46
8. Comportamiento del objeto <i>user</i>	47
9. Atributo del objeto <i>user</i>	48
10. Ejemplo de diagrama de secuencia y relaciones de objetos.....	51
11. Estereotipos y sus íconos para el modelo de navegación.....	57
12. Estereotipos y sus íconos para el modelo de presentación.....	62
13. Sistema centralizado, caso de estudio.....	65
14. Sistema migrado hacia Internet, caso de estudio.....	66
15. Abstracción de clases del sistema antiguo, caso de estudio.....	67
16. Abstracción de clases detallado del sistema antiguo, caso de estudio.....	68
17. Diagrama de caso de uso, ingreso al sistema.....	75
18. Diagrama de caso de uso, mantenimiento de empleados.....	76
19. Diagrama de caso de uso, contexto de administración de personal.....	77
20. Vista conceptual rediseñada, diagrama de clases.....	78
21. Diagrama de secuencia para la altas, bajas y consultas de empleados.....	79

22. Diagrama de secuencia, ingreso de solicitud.	80
23. Diagrama de secuencia del proceso de autorización y generación de contratos.....	81
24. Diagrama de colaboración, ingreso de solicitudes de contratos.....	82
25. Diagrama de colaboración, administración de solicitudes de contratos.	83
26. Diagrama de colaboración, mantenimiento de empleados.	83
27. Diagrama de espacio de navegación por departamento.	84
28. Diagrama de espacio de navegación para presupuesto.....	85
29. Diagrama de espacio de navegación para gerencia.....	86
30. Diagrama de estructura de navegación por departamento.....	88
31. Diagrama de estructura de navegación para presupuesto.	89
32. Diagrama de estructura de navegación para gerencia.	90
33. Diagrama de estructura de proceso para procesamiento de solicitud.	92
34. Diagrama de estructura de proceso para procesamiento de contrato y solicitud autorizada.....	93
35. Diagrama de flujo de proceso para creación de solicitud.	95
36. Diagrama de flujo de proceso para eliminación de solicitud.	96
37. Diagrama de flujo de proceso para generación de contrato.	97
38. Diagrama de presentación, estructura visual para página de solicitud.	99
39. Diagrama de presentación, interfaz de usuario, creación de solicitud.....	100
40. Diagrama de presentación, interfaz de usuario, eliminación de solicitud. .	101
41. Diagrama de componentes, estructura de páginas.	102
42. Diagrama de despliegue para usuarios de los departamentos de la empresa.....	103
43. Diagrama de despliegue para usuarios de presupuesto de la empresa.	104
44. Diagrama de despliegue para usuarios de gerencia de la empresa.	104

GLOSARIO

Actor	Es una persona o entidad externa al sistema que interactúa con los casos de uso.
Aplicación	Programa que realiza una serie de funciones y con el cual se trabaja en el computador.
Arquitectura de <i>software</i>	La manera en la cual el <i>software</i> está estructurado y construido. Diseño de alto nivel de la estructura de un sistema.
Atributo	Es la descripción de un campo con nombre de un tipo especificado en una clase.
<i>Browser</i>	Programa que permite leer documentos en la <i>Web</i> y seguir enlaces (<i>links</i>) de documento en documento de hipertexto.
<i>Bug</i>	Se refiere a un error o defecto de <i>software</i> , resultado de una deficiencia o falla durante el proceso de creación de programas para un computador.
<i>CASE</i>	(<i>Computer Aided Software Engineering</i>) Ingeniería de <i>software</i> asistida por un computador, su objetivo

reducir tiempo y dinero en el desarrollo de aplicaciones.

Caso de uso

Especificación de las secuencias de acciones, efectuadas por un sistema o clase, que interactúan con actores externos.

Clase

Descriptor de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y comportamiento.

Data warehouse

Dentro de un contexto de informática significa almacén de datos, determinado a un ámbito integrado, que ayuda a la toma de decisiones en la entidad que se utiliza.

Front-end

En diseño de *software*, se refiere a la parte que interactúa con el usuario.

Granularidad

Es el grado de detalle de una definición. La forma en que los recursos se unen o se agregan entre sí define la granularidad.

Identidad

Propiedad inherente de un objeto que permite distinguirlo de todos los demás objetos.

Implementación

La forma en que se construye o calcula algo. Fase de un sistema que describe el funcionamiento del sistema en un medio ejecutable.

Interfaz	Un conjunto de operaciones que posee un nombre y el comportamiento de un elemento.
Hardware	Conjunto de componentes que integran la parte física material de un equipo de cómputo sobre el cual funcionan programas de computadora.
HTML	<i>(Hyper Text Markup Language)</i> Lenguaje de marcas de hipertexto, lenguaje etiquetado o marcado que predomina para la creación de páginas <i>web</i> .
Mainframe	Computadora central, potente y costosa, para grandes procesamientos, su arquitectura es centralizada, su acceso es por medio de terminales físicas o virtuales.
Migración de sistemas	Consiste en trasladar un sistema informático a otro, puede cambiar, plataforma, arquitectura o datos.
Modelo	Es una abstracción semánticamente completa de un sistema.
OCL	<i>(Object Constraint Language)</i> Lenguaje de restricciones para objetos. Describe formalmente las expresiones en los modelos UML.
Página	Documento situado en una red informática, al que se accede mediante enlaces de hipertexto.

Software	Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas de una computadora.
UML	<i>(Unified Modeling Language)</i> Lenguaje de modelado visual. Un lenguaje que permite modelar problemas usando modelos organizados en torno a las ideas del mundo real.
VAX	<i>(Virtual Address eXtended)</i> Fue la primera máquina comercial de arquitectura de 32 bits, marcó un hito en la historia, lanzada en 1977.
UWE	<i>(UML based – Web Engineering)</i> Ingeniería Web basada en UML, es una propuesta de metodología que guía el proceso de desarrollo de una aplicación Web.
Web	Del inglés red, telaraña conocida como “la Web”, es el sistema de documentos interconectados por medio de enlaces de hipertexto, accesibles en Internet.
Web Server	Computadora que provee el servicio de publicación de un sistema en Internet.

RESUMEN

Se presenta una propuesta del modelo de análisis y diseño, para la migración de aplicaciones con poca funcionalidad y necesidad de cambio, construidas con arquitectura centralizada, y su traslado sea hacia una arquitectura que funcione en Internet, conjugando distintas representaciones visuales y conceptuales, que permite UML y sus respectivas extensiones para aplicaciones en Internet.

La primera fase se refiere al análisis del sistema centralizado antiguo, y el modelado conceptual del mismo con la nomenclatura UML para la especificación formal del funcionamiento de éste. En el proceso de abstracción de información del sistema centralizado se utiliza como herramienta principal la ingeniería inversa, la cual es necesaria por la poca documentación que existe para este tipo de sistemas y la investigación de los usuarios principales.

La segunda fase consiste en tomar la conceptualización obtenida del sistema centralizado y la investigación de nuevos escenarios de uso, para rediseñar el sistema, tomando en cuenta que la migración será hacia Internet. Se modifica y se amplía el diseño conceptual obtenido por medio de UWE, especificando casos de uso, secuencia de actividades y comportamiento de los objetos. Además el diseño lógico se inicia con la identificación de objetos, atributos, propiedades y las relaciones que estos tengan.

La tercera fase que se propone consiste en la redefinición del sistema con la nueva arquitectura en Internet, partiendo del nuevo diseño conceptual redefinido, agregando los diseños de navegación y presentación que sugiere la metodología propuesta para la *Web*, UWE.

OBJETIVOS

General

Brindar una propuesta práctica de análisis y diseño para llevar a cabo la migración de un sistema con arquitectura centralizada, para que funcione con arquitectura para Internet, por medio de fundamentos de UML y UWE.

Específicos

1. Dar a conocer la aplicación de estándares UML en el análisis del problema de migración, para ofrecer un mejor control del proceso.
2. Ofrecer una serie de etapas básicas necesarias para el modelado del proceso de migración de sistemas en estudio.
3. Relacionar la interface en Internet por medio de UWE y la lógica de negocios preexistentes.

INTRODUCCIÓN

El software que representan los sistemas de información suelen ser la realización de las reglas de negocios de todas las organizaciones. A medida que cambian estas reglas, el software tiene que cambiar también. En la actualidad, existen muchas compañías que poseen miles de líneas de código de computadora que codifican sus reglas de negocio.

Cuando los administradores intentan modificar esas reglas para lograr una mayor efectividad y competitividad, el software tiene que adaptarse. En algunos casos, esto implica la creación de nuevos sistemas de *software*, pero en otros significa la modificación y/o reconstrucción de aplicaciones ya existentes, para que sean competentes a la hora de satisfacer las necesidades cambiantes de los negocios del siglo XXI, entre ellas la necesidad y/o conveniencia de algunos sistemas de colocarse en Internet.

Aunque la migración y reingeniería de negocios sea una estrategia resistida por parte de la compañía, es algo que debe hacerse. Existen miles de sistemas, aplicaciones cruciales para el éxito de los negocios grandes y pequeños, que están afectados por una enorme necesidad de ser trasladados y reconstruidos hacia plataformas en Internet.

Este escenario resulta sumamente conocido, se tiene una aplicación que ha servido para las necesidades del negocio de una compañía durante diez o quince años. A lo largo de este tiempo, ha sido corregida, adaptada y mejorada muchas veces. Las personas se dedicaban a esta tarea con la mejor de las intenciones, pero las buenas prácticas de ingeniería de software siempre se omitían, por la urgencia de otros problemas.

Aun cuando este software se haya creado empleando las mejores técnicas de diseño y codificación, conocidas en la época que fueron creados (y la mayoría no lo fueron), se creó cuando los tamaños de los programas y el espacio de almacenamiento eran las preocupaciones principales. Luego se trasladó a nuevas plataformas, se ajustó para adecuarlo a cambios de equipo y sistema operativo, se mejoró para satisfacer nuevas necesidades de usuario; y todo esto se hizo sin tener en cuenta la arquitectura global y estándares de control de calidad y metodologías como UML.

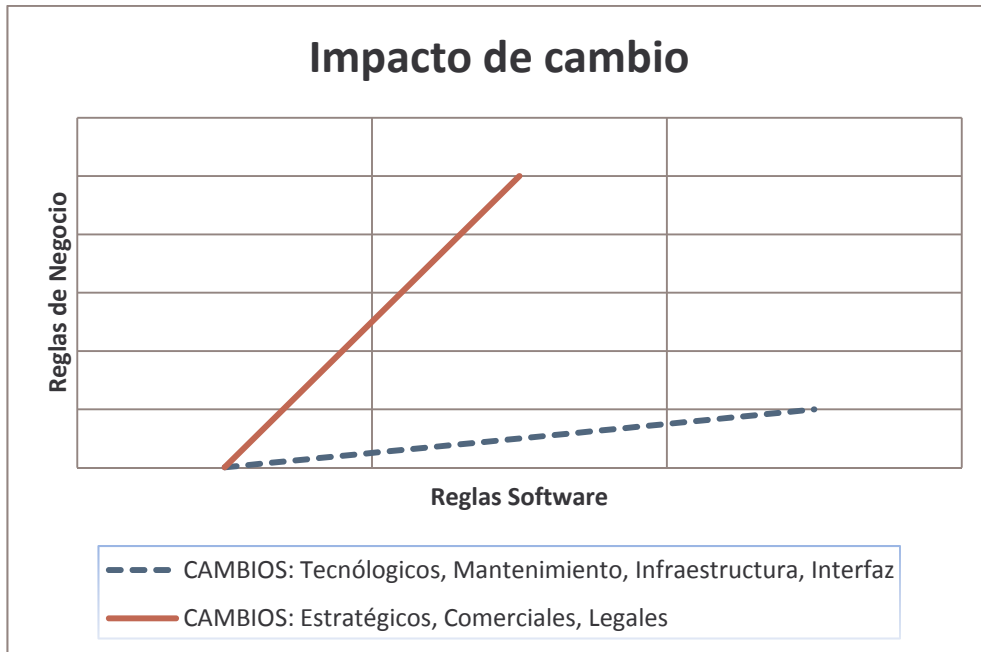
El resultado es estructura mal diseñada, mala codificación, una lógica inadecuada, una escasa documentación de los sistemas de software, y ahora se exige que continúe funcionando.

En la actualidad, la aplicación se ha vuelto inestable, sigue funcionando, pero cada vez que se intenta efectuar un cambio, se producen efectos colaterales graves e inesperados. Y sin embargo, debe seguir evolucionando.

El mantenimiento del software existente puede dar cuenta del más del 60% de las inversiones efectuadas por una organización de desarrollo, y ese porcentaje sigue ascendiendo a medida que se produce más software. [Pressman 1998,510] De él, tan sólo un 20% se invierte en corregir errores (*bugs*); el 80% restante se dedica a adaptar los sistemas existentes a los cambios de su entorno externo, a efectuar las mejoras solicitadas por los usuarios y a rehacer la ingeniería de las aplicaciones para su posterior utilización. Se puede prever en el horizonte la existencia de organizaciones de desarrollo de software truncadas en mantenimiento que ya no puedan producir software nuevo porque todos los recursos disponibles se estarán invirtiendo en el mantenimiento del software viejo.

La migración y reingeniería de sistemas de información es una actividad que absorberá recursos de las tecnologías de la información durante muchos años. Esta es la razón por la cual toda organización necesita una estrategia pragmática para la migración de sus sistemas antiguos, basada en una metodología moderna que contenga toda clase de controles.

Figura 1. Impacto de cambio en el *Software* con respecto al cambio en las reglas de negocio



1. MARCO TEÓRICO

1.1. Fundamentos de UML y UWE

1.1.1. Introducción al UML

UML (*Unified Modeling Language* en español Lenguaje Unificado de Construcción de Modelos) es la notación esquemática con la que se construyen sistemas por medio de conceptos orientados a objetos. Es un lenguaje gráfico para visualización, especificación, construcción y documentación de componentes de sistemas de software grandes y complejos, también para modelar negocios y otros sistemas que no son de software.

Especifica con notación “orientada a objetos”, cada fase de un proyecto es visualizada a través de diferentes diagramas y en conjunto representan la arquitectura del proyecto. Representa una colección de las mejores prácticas de ingeniería que han sido probadas completamente en el modelado de sistemas grandes y complejos. Provee un vocabulario y reglas de combinación con el propósito de comunicar. Su modelado produce el entendimiento de un sistema. Las reglas del lenguaje informan cómo crear y leer el flujo de los modelos formados. Su vocabulario y reglas se centran en la representación conceptual y física de un sistema de forma estándar para el diseño de software. Produce modelos explícitos que facilitan la comunicación. Algunas cosas se modelan mejor gráficamente.

UML es más que solamente símbolos gráficos, pues existe una semántica sólidamente definida detrás de los símbolos. Como lenguaje de especificación, significa la construcción de modelos precisos, completos y no ambiguos. En particular, maneja la especificación de todo el análisis, diseño y decisiones de implementación que puede hacerse en la etapa de desarrollo e implantación de un sistema grande y complejo de software. Su utilización es independiente de algún lenguaje de programación particular y de algún tipo o característica especial de proyectos. Como lenguaje de documentación, maneja la documentación de la arquitectura de un sistema y todos los detalles. Introduce diagramas que representan la parte dinámica de los procesos, logrando de esta manera identificar fallas de diseño en los procesos y por consiguiente generadoras de errores. Permite estereotipar sus elementos para que tengan un comportamiento particular.

Para entender esta metodología, se necesita construir un modelo conceptual del lenguaje y éste requiere entender tres elementos especiales: construcción de bloques básicos en el UML, las reglas que indican cómo esos bloques contruidos se pueden interrelacionar, y algunos mecanismos comunes que se aplican en todo el lenguaje. La construcción de bloques en el UML se divide en tres categorías que son: cosas, relaciones y los diagramas. Las cosas son abstracciones que están colocadas primeramente en un modelo. Las relaciones enlazan conjuntamente las cosas categorizándolas por dependencia, asociación, generalización o realización. Existen categorías de cosas en la estructura de los modelos estáticos que representan elementos conceptuales o físicos.

1.1.2. Reglas del UML

Reglas de semántica:

- Nombres, que pueden ser de cosas, relaciones y diagramas.
- Alcance, el sentido contextual que especifica a un nombre.
- Visibilidad, cómo son vistos y usados, los nombres.
- Integridad, relación entre cosas de forma apropiada y consistente.
- Ejecución, significa ejecutar o simular un modelo dinámico.

Para modelos construidos:

- Generalizados, ocultar elementos para simplificar la vista.
- Incompletos, algunos elementos pueden estar ausentes.
- Inconsistencia, la integridad del modelo no es garantizada.

1.1.3. Diagramas

Son los elementos básicos del UML, cada uno contiene una notación propia. Las diferentes formas visibles de UML son compuestas por combinaciones de diagramas, las cuales son:

- Concepción visual de casos de uso: compuesta por diagramas de, casos de uso, colaboración, estados y actividades.
- Vista de diseño: compuesta por los diagramas de clases, objetos, colaboración, estados y actividades.

- Concepción visual de procesos: compuesta por diagramas de: clases, objetos, colaboración, estados y actividades, pero repasando y afinando las clases y objetos que contienen procesos.
- Concepción visual de implementación: compuesta por diagramas de componentes, actividades, estados y colaboración.
- Concepción visual de despliegue: compuesta por diagramas de despliegue, actividades, interacción y estados.

Existen dos tipos de diagramas para representar un sistema, los de visión estática y los de visión dinámica. Los diagramas de vista estática son:

- Diagrama de clases: presenta las clases, atributos y sus respectivas relaciones. Comúnmente dan la vista estática del proyecto.
- Diagrama de objetos: presenta las instancias específicas de las clases presentadas en el diagrama de clases y su relación, da una visión particular del sistema.
- Diagrama de componentes: presenta la organización y las dependencias de los componentes del sistema.
- Diagrama de despliegue: presenta conjuntos de componentes llamados nodos, además sus relaciones. Reduce el detalle complejo de los diagramas de clases y de componentes de grandes sistemas. Resume el contenido, da una visión general del sistema.
- Diagrama de casos de uso: representa comportamientos por medio de actores y sus relaciones. Muestra qué acciones hacen los actores y las relaciones entre acciones. Modelan y organizan la lógica del negocio a nivel general, la manera que se comporta el sistema.

Los diagramas de vista dinámica son:

- Diagrama de secuencia y diagrama de colaboración: representan interacción y colaboración a través del tiempo de un conjunto de objetos, se envían mensajes entre objetos para modelar cada método.
- Diagrama de estados: representa los estados de objetos, muestra actividades y transiciones.
- Diagrama de actividades: representa el comportamiento entre objetos. Útiles para representar de que manera funciona un sistema y la fluidez de control entre los objetos.

UML proporciona un gran número de diagramas para detallar cuánto se requiera, algunos diagramas son más necesarios dependiendo el tipo de proyecto.

1.1.3.1. Diagramas recomendados

Según el tipo de sistema por crearse se recomienda una serie de diagramas básica. Dependiendo de las características del desarrollo y la experiencia, se determina los diagramas necesarios.

- Aplicaciones locales:
 - Casos de uso.
 - Clases.
 - Interacción.

- Aplicaciones orientadas a eventos:
 - Agregar: Diagrama de estados.
- Aplicaciones Cliente-Servidor (según complejidad):
 - Agregar: Diagrama de componentes
 - Agregar: Diagrama de despliegue
- Aplicaciones distribuidas:
 - Todos.

Por lo tanto, para modelar aplicaciones sencillas, se necesita de tres a seis tipos de diagramas, y para modelar aplicaciones complejas, aproximadamente nueve tipos de diagramas. La mayoría de aplicaciones no complejas necesitan a lo sumo cuatro tipos de diagramas. UML está capacitado para modelar pequeños sistemas y también sistemas complejos. Los sistemas complejos representarán cantidades millonarias de líneas de código, generadas por grandes equipos de programación, que dependen del nivel de detalle especificado en dichos diagramas.

1.1.4. Introducción a UWE

UWE (UML *Web Engineering*, en español Ingeniería *Web* basada en UML) es una metodología que permite especificar de mejor manera una aplicación Web, para el proceso de creación de aplicaciones detalla ésta, con una gran cantidad de definiciones, en el proceso de diseño lista qué debe utilizarse. Procede de manera iterativa e incremental, coincidiendo con UML, incluyendo flujos de trabajo y puntos de control.

UWE se especializa en la especificación de aplicaciones que se adaptan, y por eso hace énfasis especial en las características de personalización, y la definición de los modelos de usuario o en un patrón de características de navegación basado en preferencias, tareas o conocimiento. Otros aspectos de interés de la metodología UWE es la orientación a objetos, usuarios y la definición de un modelo de referencia que da soporte a la metodología y formaliza los modelos por el grado de restricciones y definiciones que proporciona.

1.1.4.1. Características principales

Se basa en las características principales siguientes:

- Notación estándar: el uso de la metodología UML para todos los modelos.
- Métodos definidos: pasos definidos para la construcción de cada modelo.
- Especificación de restricciones: recomendables de manera escrita, para que la exactitud en cada modelo aumente.

1.1.4.2. Fases de desarrollo

Con respecto al proceso de creación de la aplicación, UWE se vale mediante el uso de metodologías estándares reconocidas como UML principalmente y también del lenguaje de especificación de restricciones asociado OCL (*Object Constraint Language*, en español Lenguaje de restricciones para objetos).

Para recolectar los requerimientos necesarios de las aplicaciones web, esta metodología propone una ampliación utilizada en el proceso de creación, la cual se divide en las siguientes cuatro actividades:

- Análisis de requisitos: plasma los requerimientos funcionales de la aplicación Web, mediante modelos de casos de uso.
- Diseño conceptual: se define mediante un modelo de dominio, considerando los requisitos plasmados en los casos de uso, el diagrama de clases representará los conceptos con un gran porcentaje de detalle.
- Diseño navegacional: Comprende la construcción del modelo de navegación en dos pasos:
 - Modelo de espacio de navegación: su objetivo es especificar qué objetos pueden ser visitados a través de la aplicación.
 - Modelo de estructura de navegación: amplía el modelo con un conjunto de estructuras de acceso necesarias para la navegación como los índices, consultas y visitas guiadas.
- Diseño de presentación: permite la especificación lógica de la aplicación *Web*. Basada sobre este modelo lógico, una representación física puede ser construida. Representa las interfaces del usuario por medio de vistas estándares de interacción UML. Dentro de este modelo se distinguen dos diferentes vistas:
 - Estructura de vista: muestra la estructura del espacio de presentación.
 - Interfaz de usuario (UI por sus siglas en inglés de *User Interface*): Vista que presenta detalles acerca de los elementos de la interfaz de usuario dentro de las páginas.

1.2. Reingeniería de *software*

Reingeniería significa rediseñar algo de manera radical, empezar de nuevo, arrancando sin nada. Proporcionar al cliente más con menos, no significa hacer más con menos. El objetivo es hacer mejor lo que ya se está haciendo, trabajar mejor de manera inteligente, e integrar y rediseñar procesos aislados. De esta manera las compañías retoman la eficiencia que han perdido en operaciones. Dicho de otra manera la reingeniería es la revisión a los fundamentos, que establece reformas extremas, con el fin de alcanzar una espectacular mejora en medidas difíciles de rendimiento, relacionadas con costos, calidad, servicio y rapidez.

La reingeniería de software es reorganizar y modificar sistemas de software existentes, es aplicable a los subsistemas que frecuentemente requieren mantenimiento. La reingeniería agrega esfuerzo para que sea más fácil el mantenimiento del sistema. Comprende la re-documentación, reorganización y reestructura del sistema, traducción del sistema a un lenguaje de programación más moderno, actualización de estructura y adaptación de valores de datos existentes. La funcionalidad del sistema permanece igual, regularmente no se cambia y normalmente con la arquitectura se trata que permanezca igual.

Desde un punto de vista técnico, la reingeniería de software se considera una solución alternativa y poco recomendable para los problemas de evolución de sistemas.

El software arquitectónicamente no cambia mucho, pues regularmente no se actualiza y esto hace difícil descentralizar los sistemas centralizados y hacerlos distribuidos. Generalmente, es difícil cambiar radicalmente el lenguaje de programación, ya que la funcionalidad de algunos lenguajes limita la descomposición de los problemas en objetos, eventos y funciones, como otros lenguajes.

Desde un punto de vista de negocios, la reingeniería de software es una solución viable para continuar con el servicio de los sistemas, muchas veces es costoso y difícil tomar un enfoque para la evolución de los sistemas. Se debe aplicar la reingeniería cuando el soporte empieza a ser obsoleto y mientras las herramientas de reestructuración estén disponibles, si en la reestructuración el mantenimiento tiende a corromper la estructura de un programa, entonces se dificulta entender el sistema.

La reingeniería de datos puede ser necesaria debido a la inconsistencia de puntos de vista en la administración de datos. Si se convierten los datos, los costos de reingeniería se incrementan de manera notable. La reingeniería reduce el riesgo de inconsistencia, pero cuando se vuelve a desarrollar el software de una empresa, es una actividad crítica que aumenta el riesgo operativo del negocio.

La reingeniería reduce los costos a largo plazo, pues según estudios sugieren que el costo de ésta es menor que el costo de hacer de nuevo un sistema totalmente en una comparación de cuatro a uno.

El proceso de la reingeniería de software comprende varias actividades principales, las cuales son:

- Traducción de código fuente, conversión de un lenguaje de programación antiguo a una versión más actual u otro lenguaje diferente.
- Ingeniería inversa, analiza los programas extrayéndoles información de su organización para documentarla.
- Mejora de la estructura del programa, es la reestructuración de los programas para una mejor organización comprensible.
- Modularización del programa, es la forma de agrupar las partes relacionadas de los programas eliminando la redundancia.
- Reingeniería de datos, cambio de los datos según las nuevas estructuras de datos.

En el proceso de traducción de código fuente no debe cambiar la estructura y organización del programa. La traducción es conveniente hacerla por etapas debido a la actualización de arquitectura, la falta de personal capacitado en el lenguaje original, por los cambios en las políticas organizacionales, para minimizar costos con cierto lenguaje, la falta de soporte para determinado software. La traducción del código fuente es conveniente económicamente solo si se dispone de un traductor automático para una cantidad grande de traducciones. En varios casos es imposible hacer traducciones automáticas, pues las estructuras antiguas no tienen equivalente directo en el lenguaje nuevo, en estos casos hay que manipular los cambios haciendo los ajustes necesarios. La modularización comprende la reorganización del programa de forma que las partes sean agrupadas por las relaciones que más convenga. Este proceso se puede hacer por abstracciones de datos, por la funcionalidad de procedimientos que llevan a cabo tareas relacionadas, o por modelos auxiliares para apoyar los procesos del negocio.

La reingeniería de datos analiza y reorganiza las estructuras, y cuándo es necesario los valores. Prácticamente los datos se deben modificar por, la degradación de calidad de datos con el tiempo, límites no coherentes en los programas con la actualidad, evolución en la arquitectura adoptada, puesto que las arquitecturas trabajan la distribución de datos de manera diferente.

1.3. Ingeniería inversa

Es el proceso que tiene como entrada código fuente y obtiene como resultado una descripción de más alto nivel, con el fin de recuperar el diseño y la especificación. Regularmente el código es la entrada para este proceso, aunque en ocasiones no existe y se debe hacer desde el código ejecutable. La aplicación de este concepto está asociada al desarrollo de grandes sistemas o a la producción industrial de software, lo cual significa una mejora notable para mantener la competitividad del mercado.

Se debe tomar en cuenta que la ingeniería inversa no persigue el mismo objetivo con la reingeniería, puesto que la ingeniería inversa se centra en derivar el diseño o la especificación de un sistema a partir de su código fuente, y la reingeniería busca producir un sistema nuevo de fácil mantenimiento. Durante la reingeniería es necesaria la ingeniería inversa para recuperar el diseño de los programas del sistema y comprender lo que se tiene para su reorganización y reestructuración, aunque no siempre necesita seguirse de la reingeniería.

Es un trabajo a largo plazo, estas actividades, están orientadas para mejorar el mantenimiento. La productividad depende de cuanto mejora se logre en la reducción de recursos en este tipo de tarea, siendo este el objetivo principal de la reingeniería. Analizar el código fuente, el diseño actual y modificarlo, manualmente o mediante herramientas, para convertir a códigos mejor estructurados y más eficientes.

El proceso comienza con la etapa de análisis, donde el sistema se analiza a través de herramientas para descubrir su estructura, aunque esto no es suficiente para reconstruir el diseño, ya que los ingenieros necesitan el modelo estructural y el código fuente para comprender el sistema. Se usa esta información en un grafo dirigido que se vincula al código fuente del programa. Por medio del grafo se genera documentación de varios tipos como diagramas de los programas, de las estructuras de datos, también matrices de rastreo, que sirven para definir entidades y los destinos de sus referencias. Después de generar los documentos de diseño, se almacena más información al repositorio del sistema, la cual ayudará a la reconstrucción del mismo, esto implica tomar varias notas de las estructuras.

1.4. Migración de sistemas

En cualquier proceso de diseño, es normal tomar decisiones importantes y afinarlas. Las opciones que se elijan mientras se planea la migración pueden hacer que demore la distribución de algunas de las características del sistema hasta una fecha posterior.

Los pasos para crear una guía básica de migración consiste en identificar y dar prioridad a los objetivos de la migración, y comprender qué implican las opciones elegidas. Al optar por migrar a un sistema nuevo, sin duda se ha identificado ciertas características y ventajas que se estará deseando aplicar

1.4.1. Objetivos de migración

Deben verse reflejados los objetivos de la migración en su diseño. Los objetivos pueden ser en función del negocio o de la migración misma. Comúnmente los objetivos relativos al negocio impulsan la decisión de migración inicial. Una muestra de objetivos de la migración la constituye una mayor escalabilidad y mejora en la seguridad. Los objetivos en función del negocio están contenidos en la toma de decisiones de implementación y pueden utilizarse para evaluar posibles compromisos. Generalmente, se prepara una tabla de conformidad, que en etapas posteriores se emplea para identificar las tecnologías y características del producto que se va a implementar en la plataforma final. Estas tecnologías y características ayudarán a lograr los objetivos relativos al negocio.

Los objetivos con respecto a la migración incluyen aspectos como el efecto de la interrupción en sistemas de producción, el rendimiento del sistema y las maneras de aumentar la media de tiempo entre errores. Estos objetivos determinan cómo se formularán los planes de pruebas y criterios de aceptación. No están dirigidos por la necesidad de implementar características técnicas específicas de herramientas de desarrollo.

1. 5. Aplicación web

“Sistema de información que contiene grandes cantidades de datos, con un grado alto de estructuración, procesado y analizado por medio de navegadores valiéndose del protocolo HTTP (*Hyper Text Transfer Protocol*). Actualmente se maneja el mismo concepto en la comunicación cliente-servidor (navegador – *webserver*) sólo que no necesariamente el resultado de la comunicación debe provenir de la carga de una página estática. Esta puede ser el resultado de la ejecución en el servidor de alguna lógica de programación. Esto último no necesariamente se llama una aplicación Web, pero se acerca al concepto. Considérese una aplicación Web a un sitio Web donde la navegación a través de él y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio. Esencialmente, una aplicación *Web* utiliza un sitio *Web* como entrada (*front-end*) a una cierta aplicación.” [1]

Una aplicación *Web* necesariamente debe contener lógica del negocio en el servidor, de lo contrario no se cataloga como tal. Bajo este concepto una aplicación *Web* no solamente despliega información, también soporta la lógica de algún proceso operativo del negocio. Un aspecto importante es la gran interacción con el usuario, por lo tanto el diseño de la interfaz con el usuario debe ser sencillo, claro y estructurarse de manera que oriente a cualquier usuario.

1.5.1. Características

Entre los aspectos que caracterizan las aplicaciones *Web* de otras aplicaciones de software, se categorizan por tres puntos de vista:

- Usuario: debe ser fácil de usar y acceder, tanto para usuarios expertos o usuarios sin experiencia, ya que no se sabe el tipo y el número de estos.
- Plataforma: por el uso de red y las conexiones, se intensifica y es accedida desde diferentes tipos de dispositivos.
- Información: pues ayuda en la disponibilidad global de fuentes de diversa naturaleza de información, de diferentes dominios y asiste en la finalidad de la aplicación.

1.5.2. Tipos

La clasificación se puede realizar bajo ciertos criterios, como la complejidad de la aplicación, la complejidad de datos, estructura de datos, volatilidad, o la finalidad de la aplicación. La siguiente clasificación se basa en la finalidad de la aplicación:

- Interactivas: interacción con el usuario netamente.
- Transaccionales: banca electrónica, compra electrónica.
- Servicio: orientadas al servicio, ej. Asistencia financiera, simuladores.
- Portales: intermediarios en línea, centros comerciales de compra electrónica.
- Informacionales: difusión de información, independiente de personalización.

- Descarga datos: orientadas a la disponibilidad de material en servidores.
- Comunidades: servicio de subastas, foros de debate.
- Análisis de datos: aplicaciones OLAP(por sus siglas en inglés de *On-Line Analytical Processing*), *Datawarehousing*.

1.5.3. Condiciones para el desarrollo

Se debe tomar en cuenta estos requisitos para la correcta implementación.

- Calidad: evitar errores puesto que la tolerancia del usuario en este aspecto es muy limitada, información desactualizada o enlaces erróneos hacen perder la confianza del usuario en la aplicación. Por lo tanto, es de suma importancia el control que minimice dicho fracaso.
- Velocidad: considerar las condiciones de rendimiento de tecnología y telecomunicaciones, para que soporten la concurrencia y frecuencia de uso de la aplicación.
- Portabilidad: es necesario considerar el entorno cambiante y los avances de la tecnología, para que las implantaciones de una aplicación funcionen en distintas plataformas, tomando en cuenta la reutilización de código y la independencia de este por medio de marcos de trabajo.
- Seguridad: tomar en cuenta que este tipo de aplicación está expuesta ante todo tipo de usuario a través de la *web*, y para ello la transmisión de información debe protegerse a través de métodos seguros.
- Interfaz: para la presentación de la información se debe tomar muy en cuenta el diseño, para captar la atención del usuario y que sea fácil de entender y usar para que la intuición sea la guía.

- Personalización: tomar en cuenta la individualización del usuario en el diseño como valor añadido, actualizándose constantemente, pues el usuario utiliza gran variedad de aplicaciones y es necesario mantenerle un perfil.
- Integración: el manejo de contenidos de distintos formatos, estructuras y ubicaciones, es un requisito a tomarse en cuenta.
- Desarrollo rápido: el modelo de implantación regularmente para este tipo de aplicaciones es inmediato, debido al tiempo reducido influyente en el ciclo de desarrollo y necesidad de dependencia del usuario hacia más funcionalidades.

1.6. Sistemas de información centralizados

“A principios de los años 80, el procesamiento de la información de manera computarizada estaba estrictamente centralizado. Las principales características de la centralización eran las siguientes:

- Uso de una computadora principal (*mainframe*). Normalmente un gran ordenador (IBM *System* 9000) o una minicomputadora (VAX). En términos de estructura de la compañía, una unidad era responsable del mantenimiento de dicha computadora mientras sus recursos estaban disponibles para el resto del personal. A esta unidad se la llamaba unidad de informática o centro de cálculo.
- Procesamiento centralizado de la información. Todos los procesos y cálculos son ejecutados por el ordenador principal. Los diferentes departamentos tienen terminales conectados a este ordenador *mainframe*.

- Centralización de la información. La información es almacenada en dispositivos de almacenamiento bajo el control del *mainframe*.
- Control centralizado. El administrador del sistema es el único responsable del procesamiento de la información del sistema. El administrador autoriza el acceso de los usuarios, es responsable del funcionamiento, soporte y seguridad diaria del sistema.
- Servicio centralizado. El hardware y el software están mantenidos por el personal del centro de informática. Los otros departamentos no tienen técnicos informáticos.” [2]

“Entre las principales ventajas de los sistemas centralizados se pueden citar las siguientes:

- Fáciles de manejar.
- Implican menos costes de personal.
- Un ordenador es suficiente para soportar diferentes sistemas de información en una única compañía.
- Asignación de nombres uniforme (de usuarios, de ficheros, de servicios).
- La localización de los objetos no es un problema.
- El acceso y la seguridad se controlan uniformemente en todo el sistema.
- La gestión y administración puede hacerse de forma centralizada.

En resumen, los sistemas centralizados se caracterizan por su:

- Accesibilidad. Cualquiera puede usar todos los recursos.
- Coherencia. Todo funciona de igual forma en todas partes.
- Gestión centralizada. Único dominio de gestión.

Actualmente, rara vez se encuentra una situación en la que sea justificable una centralización del estilo de lo descrito anteriormente. Solo tiene sentido hablar de centralización a nivel de datos o de servicios cuando por su naturaleza es mejor tenerlos centralizados. De todas formas, ahora es bastante habitual tener lo que se ha venido en llamar clientes ligeros o *thin clients* en inglés, que son terminales de bajo coste, sin disco duro y otros recursos, pero con capacidades multimedia y que están conectados a un ordenador central más potente. En algunas situaciones en las que se desea, que un ordenador se utilice para un conjunto acotado de aplicaciones, puede ser una opción económica e interesante a tener en cuenta.” [2]

2. ANÁLISIS Y REDISEÑO DEL SISTEMA CENTRALIZADO USANDO UML

2.1. Análisis del sistema actual con ingeniería inversa

En esta sección se presenta el proceso de la ingeniería inversa y algunos de sus campos necesarios para obtener una representación estructural y generar la documentación necesaria para realizar un diseño propio y realizar la reestructuración. Los campos del proceso de ingeniería inversa dependen del trabajo que se esté realizando y de los profesionales que estén a cargo, por ello aquí se muestran sólo las secciones necesarias básicas para este tipo de trabajo aunque se puede llegar a un nivel de detalle más profundo si los profesionales a cargo lo necesitan.

Este proceso busca recuperar las cualidades y secretos del sistema en cuestión, en este caso del sistema centralizado a partir del código fuente disponible para recuperar el respectivo diseño. En la actualidad se cuenta con herramientas de tipo CASE (*Computer Aided Software Engineering*, en español Ingeniería de software asistida por un computador), que extraen la información a partir del código con cierta dependencia de un experto, la información que recupera se refiere a procedimientos, estructura de arquitectura y datos.

Aunque la aparición de multitud de herramientas facilita las labores de la ingeniería inversa, es labor humana la decisiva a la hora de completar el estudio del sistema.

Muchas veces se cuenta sólo con el código fuente para estos sistemas centralizados, aunque en algunos casos se tiene cierta información de la estructura, tanto como algún diseño no actualizado de la base de datos, una descripción incompleta de los atributos y tablas, algún manual de usuario de las primeras versiones, y algunos documentos sobre cuál debió haber sido el funcionamiento del actual sistema, todo esto puede ayudar en cierta manera a tener una visión general del sistema que se analizará.

Tomando en cuenta los sistemas obsoletos y descuidados, la ingeniería inversa se aplica para los sistemas que cuentan regularmente con las siguientes características:

- Programación sin estructura y programas muy extensos.
- Programas que no cuentan con una documentación interna. Si tienen documentación interna no está actualizada y no se entiende.
- Documentación general no existe o es obsoleta.
- La funcionalidad de la aplicación no cubre todos los requerimientos esperados.

Para iniciar el análisis del sistema centralizado que se desea estudiar, en primer lugar se debe organizar el código fuente, el cual no posee alguna estructura para que su lectura sea más fácil, y la aplicación de la ingeniería inversa pueda iniciar bien.

La actividad basada en la extracción de abstracciones es el corazón mismo de la ingeniería inversa. El analista debe evaluar el programa antiguo y partiendo del código fuente que regularmente no está documentado, debe extraer una especificación válida para el proceso de investigación y análisis que se esté realizando, además debe extraerse las estructuras de datos que se utilizan y la interfaz de interacción con el usuario.

Además juntamente con la extracción de abstracciones se produce documentación retroactiva desde el sistema existente, como parte de la investigación que lleva a cabo la ingeniería inversa, desde la escasa documentación interna del código fuente y comentarios dentro del mismo hasta la misma que genera con cada abstracción adquirida.

2.1.1. Especificaciones funcionales del sistema actual

Para comprender el procesamiento del sistema antiguo, primero la ingeniería inversa inicia por comprender abstracciones obtenidas de los procedimientos representados por el código fuente. El análisis de código se hace mediante niveles de abstracción, los cuales son: sistema, programa, componente, configuración y sentencia.

Antes que nada se debe establecer un contexto, mediante la perfecta comprensión de la funcionalidad general de todo el sistema que se dispone para iniciar con el trabajo a detalle de la ingeniería inversa.

Este contexto es necesario para proporcionar la solución a los problemas de aplicación en el sistema.

Una abstracción del funcionamiento con lujo de detalles se tiene inmersa en cada programa; sólo se debe buscar y comprender. Se debe conceptualizar las iteraciones de las abstracciones obtenidas por bloques, visualizándose por medio de diagramas de comportamiento. Las funciones dentro del código fuente representan definiciones de procedimientos funcionales. Por medio de las abstracciones obtenidas también se obtienen modelos y con ellos especificaciones del sistema. En la búsqueda de procedimientos funcionales que representen procesos generales, existen secciones auxiliares de código que vuelven engorrosa la comprensión del flujo básico. Para ello se deben segmentar los programas para identificar secciones significativas del procedimiento general y aislar lo que no sirve, para formar segmentos funcionales significativos.

2.1.2. Esquema lógico de datos

Para comprender la parte estática del sistema es preciso aplicar la ingeniería inversa a las estructuras de datos internas en los programas para la comprensión de las estructuras globales. A partir del análisis de las estructuras internas de los programas se inicia la definición de clases, por medio de agrupaciones de variables relacionadas, con esto también se identifican los tipos abstractos de datos.

(Breuer 1991.145) enumera una serie de pasos para determinar las clases dentro del programa que interactúan con las estructuras globales.

1. Se identifican los indicadores y estructuras de datos locales dentro del programa que registren información importante acerca de las estructuras de datos globales.
2. Se define la relación entre indicadores y estructuras de datos locales y las estructuras de datos globales
3. Para toda variable (perteneciente al programa) que represente una matriz o archivo, se enumeran todas las demás variables que tengan una relación con ella.

Las bases de datos generalmente permiten establecer las relaciones que existen entre los objetos no importando su organización lógica y su estructura física. Para la definición de un modelo de datos base, se puede seguir los pasos definidos por [Premerlani, 1994.42]

1. Se construye un modelo de objetos inicial. Las clases definidas como parte del modelo, se pueden conseguir mediante la revisión de registros de una base de datos de archivos planos o de tablas de un esquema relacional. Los objetos contenidos en los registros o tablas pasarán a ser atributos de clases
2. Se determinan los candidatos a claves. Se examinan los atributos para determinar si se utilizarán o no para señalar a otro registro o tabla. Aquellos que sirvan como punteros pasarán a ser candidatos a claves.
3. Se refinan las clases tentativas. Se determina si ciertas clases similares pueden o no combinarse en una única clase.
4. Se descubren las asociaciones. Mediante el uso de técnicas análogas al enfoque de clases, responsabilidades y colaboraciones

2.2. Esquema conceptual y modelado visual con UML

Es la explicación de los conceptos significativos abstraídos para el dominio del problema, que mediante el sistema con que se cuenta se presenta una solución. En esencia, lo que un esquema conceptual debe ofrecer es la representación de cosas reales, no necesariamente componentes de software.

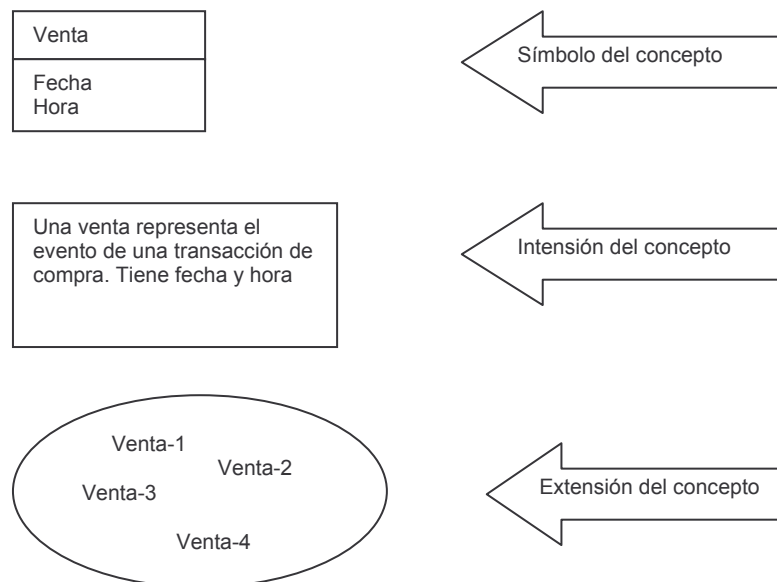
Esto se puede representar en UML con un grupo de diagramas tanto para la parte estática como para las operaciones que definen el comportamiento o sea la parte dinámica. Para ello muestra conceptos, asociaciones entre conceptos y atributos de los conceptos. La creación del esquema conceptual ayuda a descubrir y conocer la terminología del dominio del problema, modelando la comunicación y relación entre los términos importantes.

2.2.1. La vista estática

La parte estática del modelo que se abstrae por medio de la ingeniería inversa, se refiere al conjunto de elementos que tienen conceptos que significan algo en la aplicación en proceso de análisis. Ésta describe entidades de comportamiento como elementos de modelado discreto, pero no contiene los detalles de su comportamiento dinámico, los cuales solo se nombran, refieren o muestra la dependencia de la clase. Por lo tanto, los objetos y sus relaciones son los elementos esenciales de este análisis. Esta vista es la base sobre la cual se construye la vista dinámica.

Aquí se captura la estructura de los objetos abstraídos en la ingeniería inversa, aunque en esta vista se incluye lo que se refiere a las estructuras de datos tradicionales, y se enumeran las operaciones sobre los datos, prácticamente la fase de especificación. De esta manera, partiendo de estas unidades discretas de información de la arquitectura que se dispone, el diseñador puede iniciar la reestructuración donde sea necesaria. Con UML se puede representar diagramas de estructura estática que explican gráficamente el modelo conceptual. Los conceptos se pueden considerar como ideas, cosas u objetos de una manera informal, representados formalmente a partir de su símbolo, intención (definición) y extensión (instancia).

Figura 2. Visualización del concepto



2.2.1.1. Identificación de conceptos

Una estrategia válida para enfrentar la complejidad de un problema es la descomposición en unidades comprensibles. A partir del análisis estructurado tradicional del sistema dado, mediante procesos y funciones, se identifican varios conceptos del dominio del problema y se documentan los resultados en el esquema conceptual. No se debe pensar que entre menos conceptos se manejen mejor será la representación del modelo, por el contrario es mejor detallar más, para especificar un esquema conceptual.

Otra estrategia es identificar los conceptos por medio de categorías propias del dominio del problema, y también mediante la identificación de frases nominales en la especificación de los casos de uso. Pasos para construir un esquema conceptual

- Realizar una lista de conceptos adecuados a partir de una lista de categorías y de identificaciones de frases nominales.
- Graficar los conceptos.
- Agregar asociaciones para representar relaciones.
- Adicionar los atributos necesarios para satisfacer las necesidades requeridas.

Debe considerarse prácticamente que un modelo conceptual no es del todo correcto ni erróneo, sino que es una herramienta de comunicación.

2.2.1.2. Definición en UML del modelado conceptual

Para asociar y definir el modelo conceptual desde el punto de vista de análisis y diseño, se debe considerar que el término **concepto** equivale al de **clase** y **tipo** para la notación de modelado, y partiendo desde especificaciones obtenidas de un sistema existente o sea desde un punto de vista de implementación lo más correcto es modelar las estructuras abstraídas por la ingeniería inversa desde el punto de vista de diseño o sea clases, las cuales representan los conceptos con un poco más de detalle en cuanto a atributos, operaciones, métodos y relaciones.

2.2.1.2.1. Las asociaciones

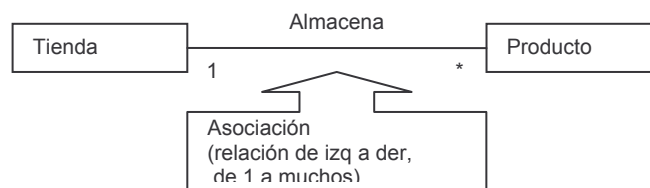
Son las relaciones significativas entre dos clases, que pueden ser comunes o eventuales pero que se preservan y son reconstruidas en el tiempo. En el modelado visual se representa como una línea entre conceptos con el nombre de la asociación, en sentido bidireccional. En los extremos puede incluirse una expresión de multiplicidad que indica las instancias de la clase. Se puede indicar el sentido de lectura con una flecha de dirección sin sentido semántico.

Es más importante identificar conceptos que asociaciones, ya que algunas asociaciones tienden a confundir en vez de aclarar. No incluir asociaciones redundantes ni derivables.

Asignación de nombre a una asociación basada en el formato NombreDeTipo-Frase nominal-NombreDeTipo.

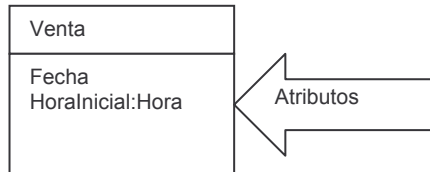
Las asociaciones que se necesitan conocer son las que facilitan la comprensión, debe verse el modelo como una herramienta de comunicación, con la cual se busca entender los conceptos importantes y sus relaciones. Se puede eliminar algunas asociaciones que no encajen y que produzcan un modelo inadecuado.

Figura 3. Asociación y multiplicidad

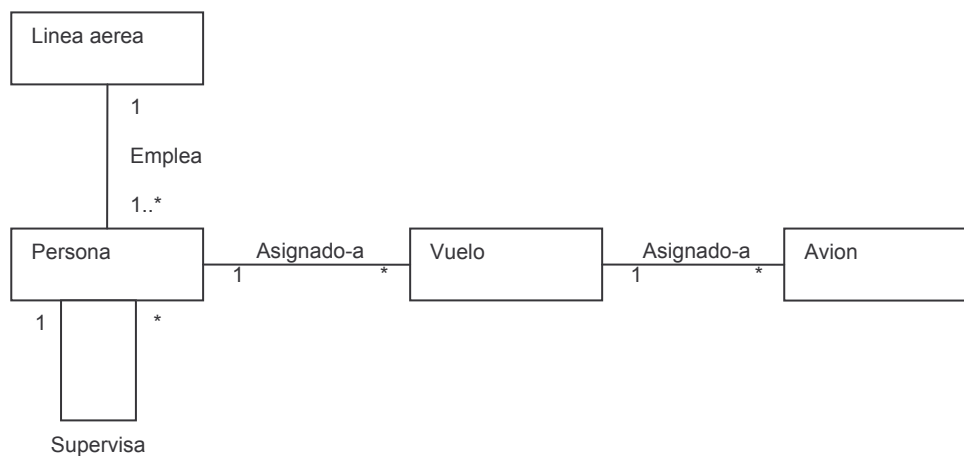


2.2.1.2.2. Los atributos

Son valores lógicos de datos de objetos. Es necesario identificar los atributos de los conceptos que se necesitan para satisfacer requerimientos de información, de los casos de uso respectivos. Los tipos primitivos de datos en la práctica suelen ser los atributos más simples, los cuales son los preferibles en un esquema conceptual.

Figura 4. Concepto y atributos

Para representar entonces el esquema conceptual, se hace uso de un diagrama de clases que facilitan las representaciones a partir de las cuales se puede iniciar la reestructuración del sistema que se desea migrar. Estos diagramas colaboran en la tarea de análisis, permite al analista una retroalimentación más directa con el cliente en su propia terminología, lo cual hace posible recabar requerimientos detallados para comprender y transformar el sistema en estudio.

Figura 5. Conceptualizando procedimientos

3. REDISEÑO DEL SISTEMA DE INFORMACIÓN CON UWE

3.1. Diseño conceptual con UWE

Ya se tiene el esquema conceptual del actual sistema, ahora debe hacerse una redefinición de lo que realmente se necesita, lo cual debería tener el funcionamiento y no se ha podido incluir. Por lo tanto se necesita diseñar un modelo conceptual, incluyendo el modelo obtenido del sistema actual, tomando en cuenta las propiedades de la nueva arquitectura de *software*. El diseño conceptual pretende definir claramente el problema que se debe solucionar y elaborar una solución para el mismo, en términos que tanto los administradores como los usuarios puedan entender.

El modelado proporciona una visión más amplia del problema, más que limitarse a enumerar los requisitos. También se trata de mantener esos requisitos en contexto y tomar decisiones racionales. El diseño conceptual captura las tareas y la información esencial necesaria, de las actividades del negocio, dando como resultado una visión de la solución, con un enfoque sobre el proceso y centrada en los usuarios. Especifica la ubicación, las capacidades y las expectativas, de los usuarios. Se considera como un análisis de actividades y consiste en la solución de negocios para el usuario y se expresa con casos de uso.

3.1.1. Perfiles de usuario

Para identificar a los actores del sistema como son llamados en UML, se hace un análisis de las personas que utilizan en la actualidad el sistema y los roles o papeles que juegan en su interacción con el mismo, en el desarrollo del diseño también se denominan responsables. Tomar en cuenta los actores externos que son necesarios para las interrelaciones que se tienen con ellos.

El actor representa el rol genérico de usuario del sistema, puede ser una persona, software o hardware. El nombre que se le dé, reflejará el papel que tendrá para el sistema. Identificar los actores permite:

- Definir los límites del sistema (Lo que forma parte del sistema y lo que no).
- Desarrollar un sistema dirigido al usuario que tome en cuenta todas las funcionalidades esperadas por los diferentes actores.

Por ello se debe conocer el funcionamiento de los procedimientos reales, para apegarse más al proceso y modelar el comportamiento. Es como volver a hacer el sistema desde cero, con el levantado de requisitos, pero con la ventaja que se tiene un modelo básico operacional el cual servirá de guía y comprensión, sobre ésta ventaja girará el nuevo sistema.

3.1.2. Escenarios de uso

“Los escenarios de uso describen los requerimientos del sistema en el contexto del usuario, mostrando cómo se efectúan los procesos de negocios, o cómo se deberían efectuar. Los escenarios de uso toman los datos que han sido recolectados, y los aplica en un documento donde paso a paso se describe qué pasa primero, luego y después en la ejecución de una tarea específica. Esto transforma los requerimientos que se han recolectado en el contexto de cómo se usan los procesos, funciones y procedimientos.”[3] Entre los distintos métodos para la construcción están los siguientes.

3.1.2.1. Modelo de proceso de flujo de trabajo

“Es usado para crear escenarios de uso, que muestran cómo trabajos específicos son ruteados a través de una organización. Estas son las condiciones necesarias para que el trabajo sea encaminado de un área a otra, y que es necesario para que un paso particular pueda darse.”[3] Es necesario definir pre y pos condiciones cuando se usa este modelo.

3.1.2.2. El modelo de secuencia de tareas

“Este modelo observa las series de acciones o secuencias de tareas que un usuario efectúa para completar una actividad.

Es posible usar este modelo con texto estructurado o no estructurado. Dependiendo que se use, se necesita identificar el rol del usuario, y escribir el escenario de uso este. El rol de usuario debe estar identificado en el escenario de uso, de manera que cualquiera que lo vea pueda saber quién efectúa que actividad.” [3]

3.1.2.3. El modelo de ambiente físico

“Los escenarios de uso también son útiles para entender el ambiente físico en el que se desenvuelve la aplicación. Esto se debe a que el diseño puede ser afectado por el lugar donde la aplicación vaya a ser usada, además de cómo y por qué. Este modelo observa el ambiente en el que la aplicación va ser usada. En este modelo, se documenta cómo las actividades se relacionan con el ambiente físico de la empresa. Esto permite determinar cómo los datos se mueven a determinadas localizaciones.” [3]

3.1.3. Diagrama de casos de uso

Basándose en la secuencia de tareas de los escenarios de uso, se crean los casos de uso, de manera que se pueda tener una idea clara, de lo que se quiere funcionalmente del sistema y de la forma en la que se realizan los procesos.

Para finalizar y poder hacer la validación del modelo conceptual se crea el diagrama de casos de uso, donde se conjuguen todos los casos de uso en un único diagrama que se analiza, para ver si cumple con las especificaciones funcionales del sistema actual. Se aplica el diagrama de casos de uso para modelar la vista de casos de uso del sistema. Esto involucra el modelado del contexto del sistema, subsistema, o clase, o modelado de requerimientos del comportamiento de estos elementos.

Su importancia radica en el hecho de visualizar, especificar y documentar el comportamiento de un elemento y hacen que el sistema sea accesible y entendible, lo que presenta una vista externa de cómo estos elementos pueden ser usados en el contexto.

Modelado del contexto del sistema: envuelve el sistema total dentro de un cuadro que dibuja los límites del modelo y los actores que interactúan con éste. Aquí, los diagramas de casos de uso especifican los actores y el significado de sus roles.

Modelado de requerimientos del sistema: especifica qué hace el sistema desde el punto de vista externo. De esta forma, el diagrama de casos de uso permite ver el sistema total como una caja negra, en el cual se puede ver las reacciones del sistema a las cosas externas, pero no se puede ver cómo ese sistema trabaja en el interior.

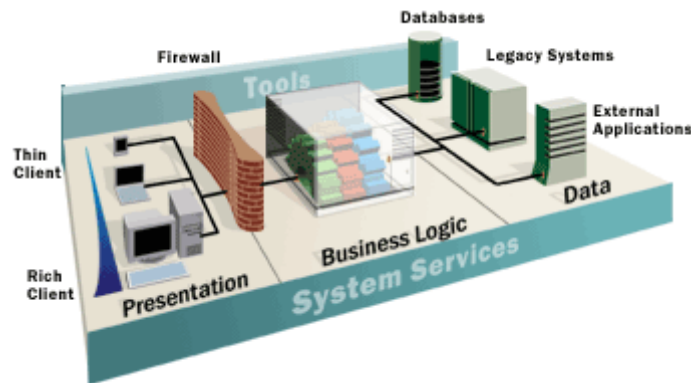
3.1.4. Diseño conceptual sobre tres capas

“La arquitectura de una aplicación es la vista conceptual de la estructura. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como está distribuido este código.”[4]

Los servicios “en la red operan de manera cooperativa para dar soporte a uno o más procesos de negocios. Una aplicación se convierte en un conjunto de servicios de usuario, negocios y datos que satisface las necesidades de los procesos de negocios.”[4] El diseño permite que los servicios estén disponibles para uso general, siguiendo lineamientos de interfaz para su publicación, la idea es que se reutilicen y se compartan entre varias aplicaciones.

La arquitectura tres capas contienen servicios ya especificados por capa, los cuales tienen comunicación entre sí, por medio de objetos o sea componentes, como se muestra en el gráfico siguiente.

Figura 6. Arquitectura de tres capas



Fuente: Diagrama de Arquitectura Windows DNA. www.msdn.com (20-12-2004)

“La arquitectura para aplicaciones distribuidas sobre Internet es un marco de trabajo para construir una nueva generación de soluciones de informática”.[4]

3.1.4.1 .Capa de presentación

Los servicios de presentación proporcionan la interfaz necesaria para presentar información y reunir datos. En esta capa se integra al usuario con la aplicación para ejecutar un proceso de negocios, y para ofrecer las capacidades de transacciones requeridas se aseguran los servicios de negocios necesarios. Estos servicios de presentación comúnmente son identificados con la interfaz de usuario, y regularmente se encuentran en un programa ejecutable localizado en la estación de trabajo del usuario final. El cliente proporciona el contexto de presentación, generalmente un navegador de internet (*browser*), que permite ver los datos remotos a través de una capa de presentación HTML.

Por medio del uso de componentes, la programación que da acceso a los datos en la base de datos y aplicaciones se aparta, desde el diseño y otros contenidos de la página *Web*. Esto asegura que los desarrolladores se enfoquen en escribir su lógica de negocios en componentes, sin la preocupación acerca de cómo el resultado se mostrará. Para el diseñador, esto le da libertad de usar herramientas familiares para modificar la interfaz. Los servicios en la capa de presentación son responsables de:

- Obtención de la información del usuario.
- Enviar información del usuario, para procesarse por medio de los servicios de negocios.
- Recibir el resultado de los servicios de negocios, después de procesarse la información.
- Presentar al usuario el resultado.

3.1.4.2. Capa de negocios

Los servicios de negocios son el puente entre un usuario y los servicios de datos. Dan respuesta a las distintas peticiones, ya sean del usuario o de otro servicio. Para ello aplican reglas de negocio por medio de procedimientos formalizados. Los datos deberán residir en la capa de datos para garantizar la correcta aplicación de estas reglas. Esto evitará que el usuario tenga interacción directa con la base de datos.

Las tareas de negocios están definidas por operaciones resultantes de los requerimientos de la aplicación, como ingresar una orden de compra o imprimir una lista de clientes. Las reglas de negocio se refieren a políticas que controlan el flujo de las tareas, estas cambian con frecuencia más que las tareas específicas de negocio que atienden, por lo tanto, es recomendable encapsularlas en componentes, que se separan de la lógica de la aplicación.

Para ayudar a los desarrolladores a construir la lógica de negocio basada en componentes de aplicaciones *Web*, existe un conjunto muy poderoso de servicios que se encargan de la comunicación en una aplicación de tres capas. Estos servicios están altamente integrados unos con otros bajo un sistema operativo, y expuestos de forma única a través de objetos componentes.

Los servicios en la capa de negocios son responsables de:

- Recibir información de la capa de presentación.
- Relacionar los servicios de datos, para la ejecución de procesos del negocio, para el cual fue diseñada la aplicación.
- Enviar los resultados procesados al nivel de presentación.

Algunos de los servicios *Web* para la capa de Negocios son los siguientes:

- Servicios de componentes y transacciones
- Servicios asíncronos
- Programación de acceso al servidor.

3.1.4.3. Capa de datos

Los servicios en la capa de datos son responsables de:

- Almacenamiento
- Recuperación
- Mantenimiento
- Integridad

Estos servicios se acomodan para administrar base de datos relacionales, sistemas de archivos y servidores de correo electrónico.

Para determinar necesidades, funcionalidades del producto u otras características percibidas, durante las etapas posteriores del modelado se hace referencia al diseño conceptual, y flexibilizar el proceso de diseño final.

3.2. Diseño lógico

Es el modelo que se construye a través de la información abstraída en el diseño conceptual, las necesidades y requerimientos del usuario son captadas con la perspectiva de la etapa previa. Las relaciones entre elementos y estructuras son establecidas en este diseño, desde el punto de vista de esta etapa se identifican objetos, servicios, interfaces de usuario y estructuras de base de datos; no importando detalles físicos de implementación, para que este diseño sea completamente independiente de modelos físicos. Entre más detalle el nivel de especificación será más independiente del diseño conceptual, y acercara mucho el modelo a la realidad del requerimiento obtenido del usuario.

Por lo tanto, el diseño lógico es el mapeo entre requerimientos de usuario conceptualizados y los objetos y servicios de negocios.

3.2.1. Organización de las estructuras lógicas

Después de la etapa de identificación de objetos, es necesario organizar todo según servicios que estos provean y relaciones entre sí. Las consideraciones esenciales a tomarse en cuenta para diseñar en varias capas son: escalabilidad, disponibilidad y eficiencia. Partiendo de estos factores importantes cualquier elemento y relación en el diseño deberá alinearse. Con respecto a la escalabilidad debe definirse el grado de granularidad entre los componentes.

3.2.2. Transición del diseño conceptual al diseño lógico

Consiste en el mapeo de objetos identificados con las reglas del negocio a través de la conceptualización obtenida del Diseño Conceptual. Regularmente la identificación de objetos y reglas obedecen a procedimientos muy propios de la teoría orientada a objetos. Según el grado de granularidad que se desee así serán los servicios que el objeto brindará. No obstante, el diseño lógico resultante no identifica tecnologías específicas. El objetivo de esta fase es analizar y comprender la funcionalidad de la aplicación antes de tomar decisiones relativas a la tecnología.

Si el diseño final de la aplicación incluye componentes personalizados, en la fase física se pueden traducir directamente los componentes correspondientes identificados en la fase lógica. Por ejemplo, si en la fase lógica se define un objeto *Users* y el equipo decide que este objeto debe ser personalizado, se repetiría el diseño lógico de dicho objeto en la fase física.

3.2.3. Creación del diseño lógico

El primer paso para la creación del diseño lógico de una aplicación es identificar los objetos (los componentes) que proporcionarán la funcionalidad necesaria. A continuación, el equipo debe identificar los comportamientos, atributos y relaciones de cada objeto, para lo cual el equipo se sirve de los escenarios de uso creados en la fase conceptual.

A continuación, el equipo analiza el escenario para identificar los aspectos fundamentales de la solución y realizar las siguientes tareas:

- Identificar los objetos del escenario.
- Identificar los comportamientos de estos objetos.
- Identificar sus atributos o propiedades.
- Identificar las relaciones lógicas entre ellos.

Una vez que se han completado y documentado estas tareas para cada escenario de uso, el equipo habrá finalizado la fase de diseño lógico.

En la fase lógica regularmente UML utiliza las siguientes secciones, en las que se describen las tareas que intervienen en la creación del diseño lógico, se ofrecen ejemplos de diagramas simples de UML.

3.2.3.1. Identificación de objetos

Al analizar un escenario de uso, lo primero que hay que hacer es identificar los objetos presentes en dicho escenario. Un objeto es, por lo general, una entidad o proceso empresarial que aparece en el escenario de uso. Por ejemplo, los objetos del siguiente párrafo aparecen en negrita:

El **usuario (User)** selecciona un **catálogo (Catalog)** en el que realiza búsquedas. Aparecerán las **categorías (Categories)** y **productos (Products)** de la raíz del catálogo seleccionado. El usuario podrá elegir entre ver los detalles de un **producto (Product)** determinado o seleccionar una categoría y ver sus productos y subcategorías.

En este escenario se utilizan los siguientes objetos:

User

Catalog

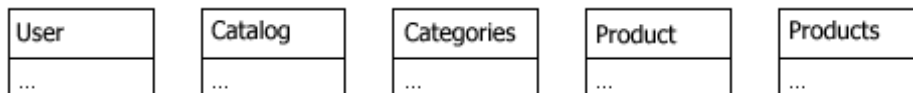
Categories

Product

Products

La siguiente figura muestra un diagrama de UML que ilustra los objetos especificados en este ejemplo:

Figura 7. Objetos



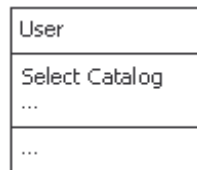
Estos cinco objetos constituyen la base de este escenario. No obstante, en determinadas situaciones, es necesario disponer de objetos adicionales para que el escenario funcione, aunque dichos objetos no aparecen de forma específica en el escenario. Estos objetos adicionales se pueden identificar examinando los comportamientos sin objetos aparentemente asociados. Para identificar estos objetos, es necesario, en primer lugar, identificar los comportamientos.

3.2.3.2. Identificación de los comportamientos

Una vez que se ha identificado el conjunto obvio de objetos, el siguiente paso es identificar sus comportamientos correspondientes, también denominados **métodos o servicios**. Para identificar los comportamientos de los objetos, es necesario evaluar, en primer lugar, las acciones que tienen lugar en el escenario. Por ejemplo, las acciones del siguiente párrafo aparecen en negrita: El usuario **selecciona** un catálogo en el que realiza búsquedas. **Aparecerán** las categorías y productos de la raíz del catálogo seleccionado.

El usuario podrá **elegir** entre **ver** los detalles de un producto determinado o seleccionar una categoría y **ver** sus productos y subcategorías. En primer lugar, el usuario selecciona un catálogo. La figura es un diagrama de UML que ilustra el objeto *User* con el comportamiento *Select Catalog* (Seleccionar catálogo):

Figura 8. Comportamiento del objeto *user*



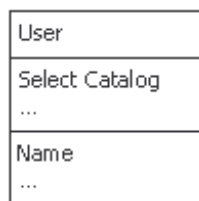
Como se mencionó anteriormente, los comportamientos sin objetos asociados aparentes deben derivar del escenario. Por tanto, debido a que el usuario selecciona un catálogo, deberá existir un tipo de mecanismo que permita seleccionar un catálogo de una lista de catálogos. Lógicamente se podría entonces asumir la existencia de un objeto *Catalogs* (Catálogos) que administra la colección de objetos *Catalog* y que, por tanto, se debe agregar a la lista de objetos definida. Una vez que se ha definido el objeto *Catalogs*, se puede definir la primera acción como *Select Catalog*; este comportamiento pertenece al objeto *Catalogs*. A continuación, se debe seguir evaluando cada una de las oraciones del escenario hasta identificar todos los objetos necesarios y sus comportamientos asociados.

3.2.3.3. Identificación de atributos

Una vez que se han identificado los comportamientos, el siguiente paso consiste en identificar los atributos (también denominados *propiedades*) de los objetos definidos. La aplicación necesita realizar un seguimiento sobre estos elementos. Se trata de marcadores de posición donde los datos se retienen o almacenan.

Los atributos se obtienen del análisis de los comportamientos del escenario y de la extracción de los elementos que se deben almacenar o controlar. Por ejemplo, en la sección anterior, el escenario de uso especifica que el usuario podrá ver un producto determinado. Una vez visualizado éste, los elementos que se muestran al usuario son los atributos del producto. Por ejemplo, si la empresa requiere que se muestre la descripción y el precio del producto, dichos elementos se convierten en atributos que se identifican en los objetos. La figura es un diagrama de UML que ilustra el objeto *User* con el atributo *Name* (Nombre):

Figura 9. Atributo del objeto *user*



3.2.3.4. Identificación de relaciones

Una vez que se han definido los objetos, sus comportamientos y atributos, el siguiente paso es identificar sus **relaciones**, que son asociaciones lógicas entre objetos. Para identificarlas, es necesario analizar la interacción entre los distintos objetos. Por ejemplo, el objeto *Categories*, que administra la colección, tiene una relación con el objeto *Category* (Categoría), ya que contiene varios de estos objetos.

Es importante resaltar que existe otro tipo de relación denominada **herencia**, que se da sólo cuando un objeto es definido por otro. Por ejemplo, si la solución que se está diseñando está destinada a la venta de comida y libros y se pretende diferenciar de forma lógica entre ambos productos, se debería definir una relación en la que los objetos *Book* (Libro) y *Food* (Comida) fueran tipos del objeto *Product*. Es decir, ambos heredan del objeto *Product*.

En las aplicaciones que referencian la arquitectura para comercio electrónico (*Reference Architecture for Commerce*), no se definen relaciones de herencia en la fase lógica. No obstante, en determinadas soluciones de comercio electrónico, las relaciones pueden resultar importantes. El resultado final del proceso es la creación de un diseño lógico que se utiliza como base para el diseño y las especificaciones técnicas.

3.2.4. Diagramas de clases

Para analizar la relación entre clases (Conjunto de objetos) se utiliza el diagrama de clases, dicho diagrama se basa en lo anteriormente mencionado.

3.2.4.1. Métodos de las clases en cada una de las tres capas

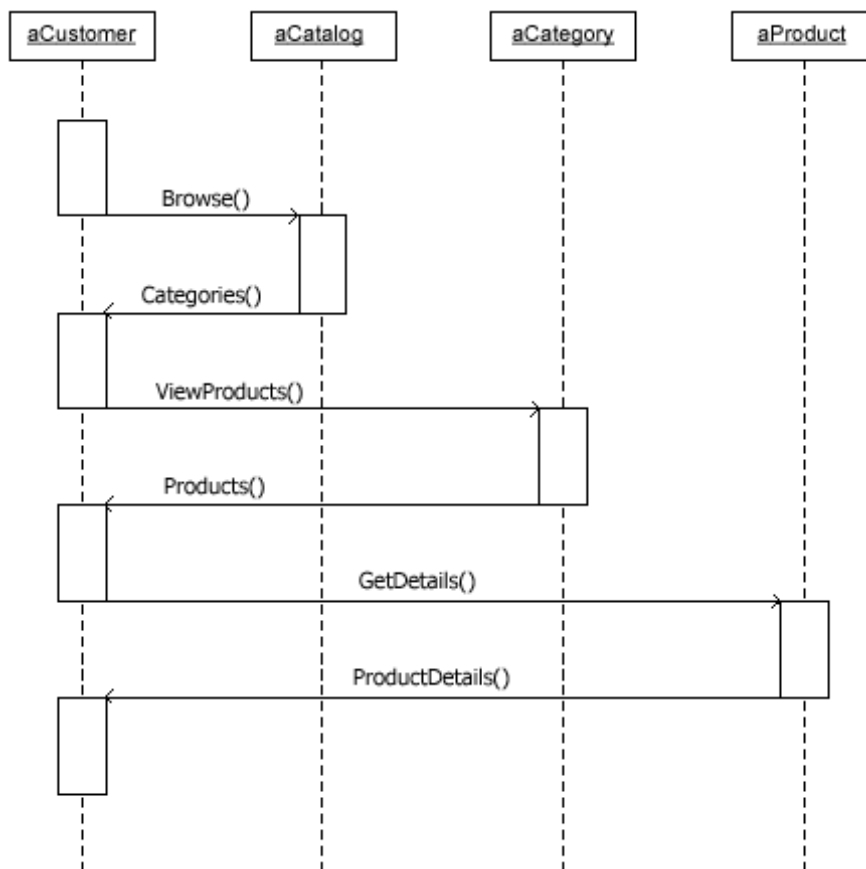
Para obtener un buen diseño de una aplicación de tres capas es necesario poder colocar los métodos de las clases correctas en las capas correctas, de esta manera es posible encontrar las características de escalabilidad y buen desempeño. En las etapas anteriores se encontraron los métodos de las clases, pero para una aplicación de tres capas es necesario que los métodos se ubiquen según su funcionalidad en la capa correspondiente.

En la capa de servicios de presentación o interfaz, deben ir los métodos que no tienen nada que ver ni con manejo de datos, ni con reglas de negocios, simplemente llamadas a métodos de otras capas y procesos de exposición de información a través de la interfaz. Por ejemplo, Ir_Registro_Anterior, etc.

En la capa del medio o de servicios de negocios, deben ir todos los métodos que contienen las reglas de negocios propiamente dichas y las llamadas a los métodos de manipulación de datos de la tercera capa. En la capa de servicio de datos deben ir todos los métodos de manipulación de datos que acceden al repositorio de datos directamente, la base de datos.

El diagrama de secuencias y el análisis de funcionalidad determinarán el lugar dónde colocar cada método. Es importante recordar que el diseño lógico resultante no identifica tecnologías específicas. Estas tecnologías se identifican en una fase física del diseño.

Figura 10. Ejemplo de diagrama de secuencia y relaciones de objetos



4. REDEFINICIÓN CON UNA ARQUITECTURA *WEB* USANDO UWE

4.1. Modelando el proceso de negocio *web* con UWE

Hasta esta etapa se ha modelado los requerimientos de la aplicación preexistente, por medio de sus especificaciones funcionales, hasta encontrar distintos aspectos de cómo trabaja la aplicación, pero ahora es necesario considerar cómo la arquitectura *web* impacta en el nuevo diseño.

El diseño de las aplicaciones de negocio *web* siguiendo esta metodología requiere de las siguientes actividades: la primera parte del diseño es una transacción de refinamiento del modelo conceptual, donde las restricciones del diseño *web* se aplican al diseño obtenido, de la conceptualización del sistema original, agrega atributos y métodos a las clases identificadas. El segundo paso consiste en integrar los procesos en el modelo de navegación, para indicar las posibilidades de navegación. La tercera actividad afina el modelo de actividades mediante una estructura de procesos y un flujo de vistas de proceso. La última, pero no menos importante, es el modelo de presentación, basado en la navegación y el modelo de procesos, mostrando cómo se combina el paradigma de navegación y las actividades del negocio.

El producto de esta fase es un diseño de implementación completo o una guía en forma de documento de especificación técnica, que el equipo de desarrollo empleará para crear la aplicación.

4.1.1. Integración de procesos en el modelo de navegación

Las actividades del modelado navegacional en UWE comprenden la construcción del modelo de navegación en dos fases.

4.1.1.1. Modelado del espacio navegacional

Primero, el objetivo es especificar **cuáles** objetos pueden ser visitados a través de la aplicación. Incorporando a este diagrama construcciones adicionales que muestran **cómo** el usuario puede alcanzar estos elementos navegando.

El modelo de navegación es representado por diagramas estereotipados de clases. Este incluye las clases de esos objetos, los cuales pueden ser visitados a través de la aplicación *Web*. UWE provee un conjunto de guías y mecanismos semiautomáticos para el modelado de navegación de una aplicación, las cuales son detalladas en manuales de referencias. Esta automatización es soportada por herramientas CASE.

UWE define un conjunto de elementos de modelado usadas en la construcción del modelo de navegación. Como primer paso la <<clase de navegación>> y el <<enlace de navegación>> y luego los estereotipos adicionales <<clase de proceso>> y <<enlace de proceso>>, los cuales se definen con la siguiente semántica.

La clase de proceso modela una clase cuya instancia es visitada por el usuario durante la navegación. Se les asigna el mismo nombre que su clase correspondiente conceptual. Este posibilita definir un mapa funcional entre la clase de proceso y los casos de uso (estos casos de usos no estereotipados como navegacionales) en una vía similar para el mapeo funcional definido entre la clase de navegación y la clase conceptual.

El enlace de proceso modela la asociación entre una clase de navegación y una clase de proceso. Este enlace de proceso necesita tener asociada información acerca del estado de proceso. Por ejemplo, pueden ser restricciones por una expresión OCL sobre el estado de proceso. Esta permite resumir actividades que contiene el proceso sobre condiciones acertadas. Las relaciones entre el espacio de navegación representan una navegabilidad directa, indicando el punto de inicio y el punto final de navegación. Para fijar las direcciones de navegación se usan flechas, en las que se indican la multiplicidad y el rol. Cuando falta el nombre en la relación, por convenio se determina de la siguiente manera, si la multiplicidad es menor o igual a uno, se toma el nombre de la clase destino para el rol, y si es mayor que uno se toma el plural del nombre. Este enlace puede ser bidireccional, en la notación puede ser una línea sin flechas.

4.1.1.2. Estructura del modelado navegacional

El segundo paso en la construcción del modelo de navegación, consiste en la ampliación del modelo, por un conjunto de estructuras de acceso necesarias para la navegación. Esta ampliación es parcialmente automatizada, esto consiste en introducir índices, visitas guiadas y consultas. Por cada una de estas construcciones, UWE define una clase estereotipada <<índice>>, <<consulta>> y <<visita guiada>> dentro de los mecanismos extendidos de UML. Adicional a esto, el modelo es enriquecido automáticamente con menús, los cuales UWE incluye con una clase estereotipada <<menú>>. Para todas las construcciones, UWE especifica elementos de modelado usando el lenguaje de restricciones de objetos (OCL) para definir invariantes de esas construcciones.

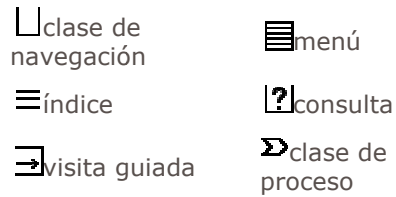
Los estereotipos e íconos asociados para estos elementos de navegación son:

Índice: permite un acceso directo a las instancias de las clases de navegación. Se modela mediante un objeto compuesto de enlaces con los nombres de las instancias de las clases de navegación a donde apuntan.

Consultas: su diseño consiste en una clase cuyo atributo es una cadena de consulta, puede ser una operación de selección definida con OCL.

Visitas guiadas: representan cómo acceder secuencialmente las instancias de una clase de navegación y deben ser controladas por el usuario o por el sistema.

Figura 11. Estereotipos y sus íconos para el modelo de navegación



Los elementos de acceso de menú se agregan siguiendo una serie de reglas que a continuación se detallan.

- Tomar en cuenta que las asociaciones tienen una clase de navegación como fuente.
- A cada clase de navegación que tiene al menos una asociación que sale se asocia una clase menú.
- Se reorganiza los menús en submenús.
- Para cada rol en el extremo de una asociación con dirección, se introduce un elemento correspondiente de menú.
- Toda relación que tiene como fuente una clase de navegación.

4.1.2. Afinando el modelado de procesos

En nivel de diseño UWE propone construir un modelo de procesos, el cual tiene una vista estructural del proceso y una vista de comportamiento o flujo de proceso. Otras vistas son la integración de vistas con el modelo navegacional, la cual es descrita definiendo entrada de procesos y puntos finales entre ejecución de procesos y de navegación.

Estos conceptos son similares para el “inicio de actividad” y el “final de actividad”, este modelo puede ser independiente de la navegación. La estructura del modelo de proceso es igual al modelo de navegación, derivado desde el modelo conceptual. La diferencia con el modelo de navegación es que el objetivo de este modelo es capturar la información relacionada que comprende una estructura de comportamiento. Se puede adherir nuevos elementos en el proceso que no necesariamente son derivados del modelo conceptual, siempre usando los estereotipos mencionados anteriormente.

4.1.2.1. Modelo estructural del proceso

El objetivo es describir las relaciones entre las diferentes clases de proceso, se crea un diagrama de clase donde cada una se representa con un estereotipo de clase de proceso. Se conjuntan estas clases y se asocian a una superclase que represente el proceso general, se agrega algunas clases si es necesario para denotar alguna interacción u operación en común y se asocian con otra superclase del mismo tipo para heredar características.

4.1.2.2. Modelo de comportamiento del proceso

La conducta de un proceso es representado mediante un diagrama de actividades UML, describiendo el flujo de una clase de proceso, lo que sucede cuando un usuario navega hacia una clase de proceso.

Se modela un diagrama de actividad para cada clase de proceso con un nombre propio del diagrama de navegación, etiquetándolo con el nombre de la clase más el flujo de trabajo, de modo que se puede ver de manera fácil que la actividad pertenece a un estereotipo. Pueden surgir varios diagramas de actividad en estas derivaciones. Para denotar interacción con el usuario y la página *web* y respondiendo a un requerimiento informativo con el sistema se usa el estereotipo <<Acción de usuario>>.

4.1.3. Soporte de procesos en el modelo de presentación

El modelo de presentación en UWE permite la especificación de la lógica de presentación de una aplicación *web*. Basada sobre el modelo lógico una presentación física puede ser construida, la cual contiene afinaciones adicionales de elementos, para el diseño físico por ejemplo letras y colores. Esta presentación física, no es el objetivo de este trabajo, pues no puede ser capturada por algún modelo UML. Dentro del modelo de presentación se puede distinguir dos vistas diferentes: la **estructura de vista** que muestra la estructura del espacio de presentación, y la **interfaz de usuario**, la cual presenta detalles acerca de los elementos de la interface de usuario en las páginas.

Este diseño se guía por algunas reglas que a continuación se presentan.

- Crear una clase de presentación por cada clase de navegación que se presente en el modelo de estructura de navegación.

- Crear una clase de presentación por cada clase menú, índice, visita guiada o consulta.
- Como soporte de la navegación se debe crear una clase de presentación.
- Adicionar enlaces a las clases de presentación.
- Señalar qué elementos deberán presentarse juntamente al usuario en alguna ventana o sección.
- Adicionar si son necesarias expresiones OCL.
- Crear los escenarios mediante serie de bocetos, representando la sucesión de vistas de interfaz de usuario.

Por medio de los diagramas de presentación se puede indicar qué clases de navegación y de procesos pertenecen a una página *web*.

4.1.3.1. Estructura visual

El objetivo de la estructura de vista en la presentación es modelar cómo el espacio de navegación es dividido, cuáles elementos de presentación son desplegados en el mismo espacio, pero no en el mismo tiempo y cómo los elementos de presentación pueden ser agrupados, o sea el flujo de presentación.

El concepto central alrededor de la estructura de espacio de presentación toma lugar en el concepto de colocación.

Por tanto se puede definir en el meta modelo de UWE una clase abstracta con el estereotipo <<colocación>>, la cual es la generalización de las siguientes clases estereotipadas <<grupo de colocación>>, <<colocación alternativa>>, y <<clase de presentación>>. La semántica de estas clases se define como sigue

- <<grupo de colocación>> clases estereotipadas son usadas para modelar la subestructura de presentación, por ejemplo un conjunto de páginas. Estas agregan una lista de sub-localidades.
- <<alternativa de colocación>> clases estereotipadas son usadas para modelar alternativas de presentación entre clases de colocación, opcionalmente una alternativa automática puede ser especificada.
- <<clase de presentación>> este estereotipo representa fragmentos lógicos de página y está compuesto de elementos de interface de usuario, presentados para el usuario de la aplicación. Cada elemento de la clase de presentación es relacionado exactamente con una clase de navegación o clase de proceso contenida dentro de los elementos de un modelo de navegación o de procesos definiendo por allí la presentación para ese elemento particular.

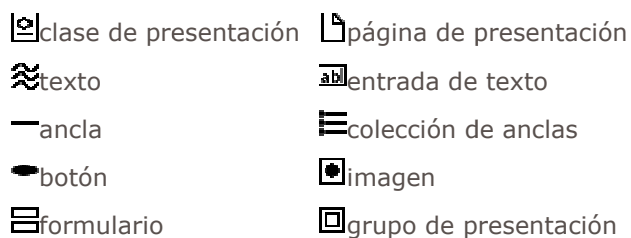
4.1.3.2. Interfaz de usuario

Muestra el contenido de cada nodo en la estructura del flujo de presentación mediante bocetos establecidos por el diseñador *web*. Se utiliza una notación alternativa de UML para mostrar la composición de la relación de los elementos de interfaz de usuario dentro del contenido visual de una clase de presentación.

Esta propuesta permite visualizar un esquema más intuitivo, como se observa la relación de composición tradicional, a la hora que se representa la vista de la interfaz de usuario. Cada tipo de elemento de interfaz de usuario tiene un estereotipo asociado. Estos son conectados por las características que subyacen de la navegación o clases de procesos, en el caso de elementos de interfaz de usuario dinámicas. Adicionalmente, este tipo de elementos de interface de usuario pueden ser usados por elementos de interface de usuario estáticas, por ejemplo las imágenes estáticas como logos o textos estáticos como encabezados.

El tipo de elemento de interface de usuario utilizado para presentar los elementos correspondientes de los modelos subyacentes depende de su tipo y el uso previsto. Un elemento de ingreso por ejemplo puede ser usado para desplegar información, así como para la entrada de información en el caso de los atributos de una clase de proceso.

Figura 12. Estereotipos y sus íconos para el modelo de presentación



5. CASO DE ESTUDIO

5.1. Descripción caso de estudio. Sistema de contratos de personal

Tomando en cuenta los conceptos descritos en este trabajo se estudia un caso práctico del mundo real. La aplicación para este caso se refiere a un módulo de administración de contratos de una institución, que por su expansión actualmente tiene sucursales distribuidas geográficamente.

El sistema con que se trabaja actualmente es de arquitectura centralizada, con un servidor que contiene la programación de las aplicaciones, la base de datos y el acceso es por medio de terminales tontas o sesiones de emulación. No puede atender procesos en línea de las sucursales.

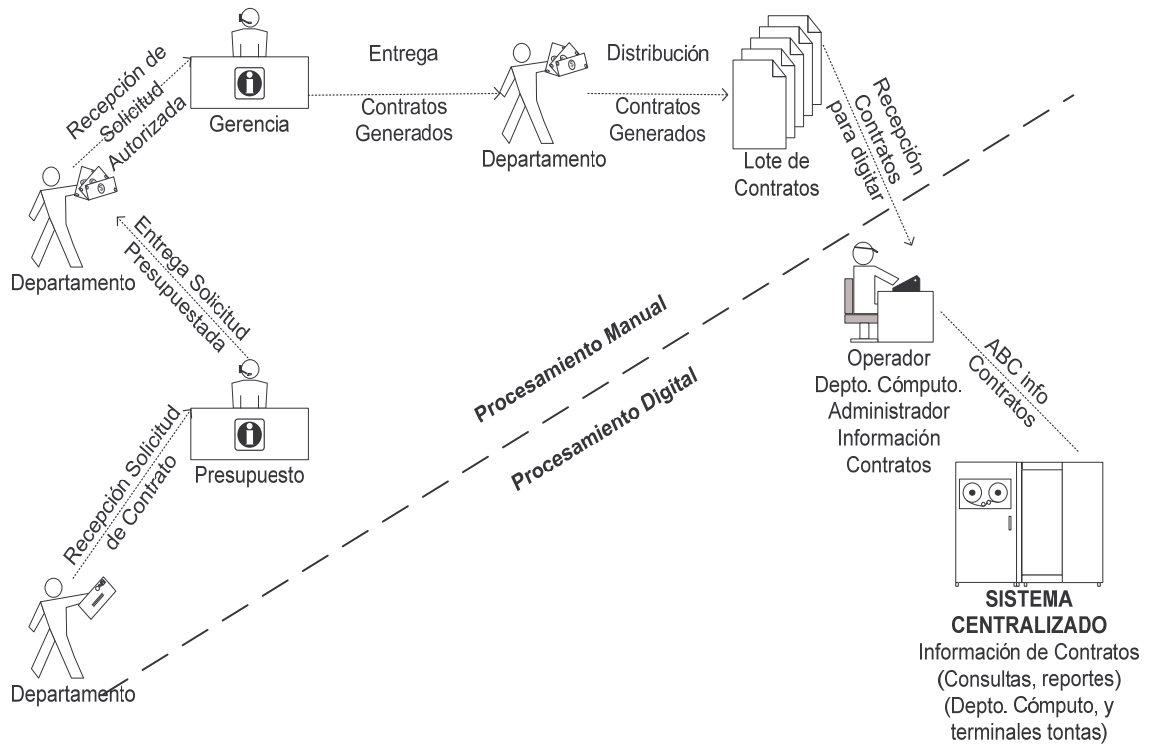
El procedimiento para administrar contratos del personal actualmente cuenta con una parte burocrática la cual consta de, redactar una solicitud de contrato para una persona durante determinado período de tiempo, por parte de cada departamento de esta institución, luego esta solicitud es llevada por el encargado del departamento a la unidad de presupuesto, aquí es donde se reciben todas las solicitudes para revisarse, autorizarse o rechazarse, posteriormente las solicitudes son regresadas a los departamentos correspondientes, para corregirse o bien si son presupuestadas las solicitudes, se llevan a la gerencia para su aprobación final y para la generación de los respectivos contratos.

Luego se acumulan lotes de contratos para trasladarlos al departamento de procesamiento de datos para su ingreso al sistema, en el cual se guarda parte de la información por el diseño limitado. Cuando se requiere por parte de cualquier departamento de la empresa información acerca de contratos y empleados, únicamente la división de procesamiento de datos y algunas estaciones de trabajo conectadas al servidor pueden acceder a ésta.

5.2. Solución de la migración utilizando el método propuesto

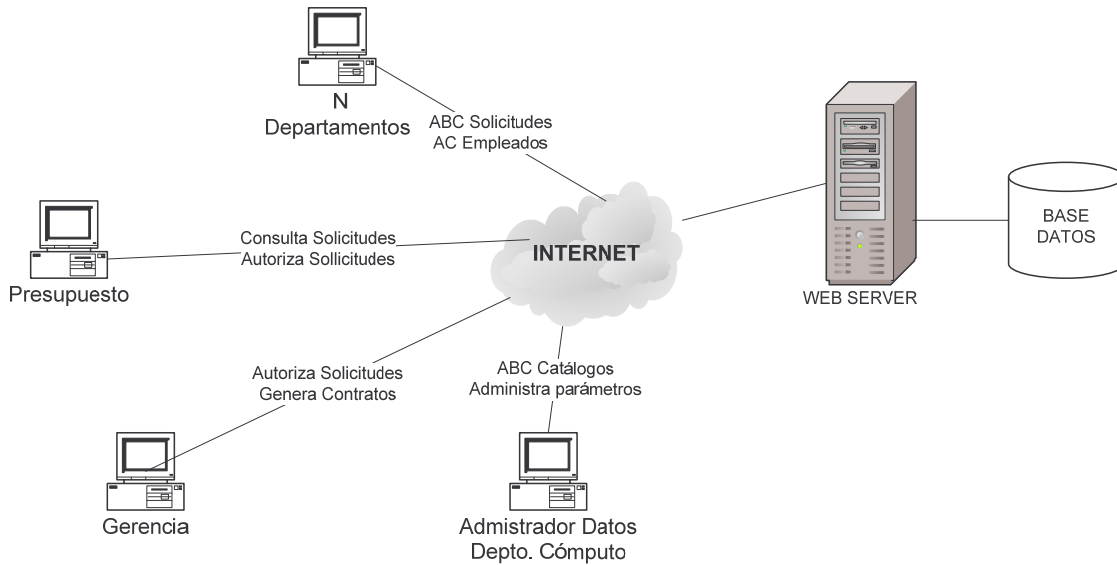
Primeramente se analiza el problema, a simple vista se detecta problema en automatización de procesos, y la necesidad imperiosa de migrar hacia una arquitectura que permita al sistema atender la cobertura geográfica que la empresa requiere, además no es sólo migrar el sistema de arquitectura, también aprovechar el rediseño y agregar funcionalidad a los procesos que existen en el sistema actual y agregar nuevos procesos. Lo que busca esta propuesta es guiar el proceso de migración de la arquitectura del sistema antiguo y su traslado hacia Internet, con un diseño basado en parte por la metodología de visualización de UML y la ingeniería para la *Web* UWE, en este caso de estudio se simplificará la solución y se resumirán las etapas, la visualización de algunos diagramas representará la esencia de las fases principales, un lujo de detalles esta fuera del alcance de los objetivos de este trabajo. En la siguiente figura se muestra cómo funciona el sistema antiguo que se dispone y el proceso completo.

Figura 13. Sistema centralizado, caso de estudio



Según la solución del problema, el cambio de arquitectura del sistema de manejo de contratos quedaría como se muestra en la siguiente figura.

Figura 14. Sistema migrado hacia Internet, caso de estudio

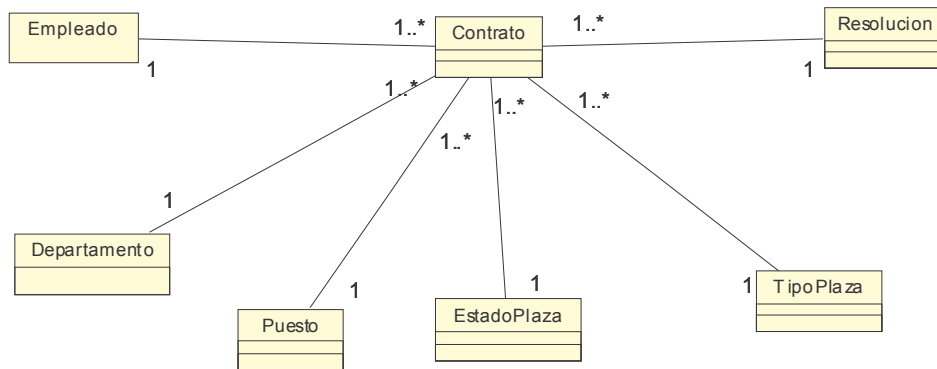


5.2.1. Análisis y diseño del sistema actual centralizado

Se debe aplicar el análisis de ingeniería inversa e investigación de código fuente, y de modelos de base de datos. De la abstracción que se obtenga de la investigación del sistema antiguo se debe crear un esquema conceptual, el cual se representa con un diagrama de clases. En la siguiente figura se muestra cómo se conceptualiza el sistema antiguo por medio de un diagrama de clases.

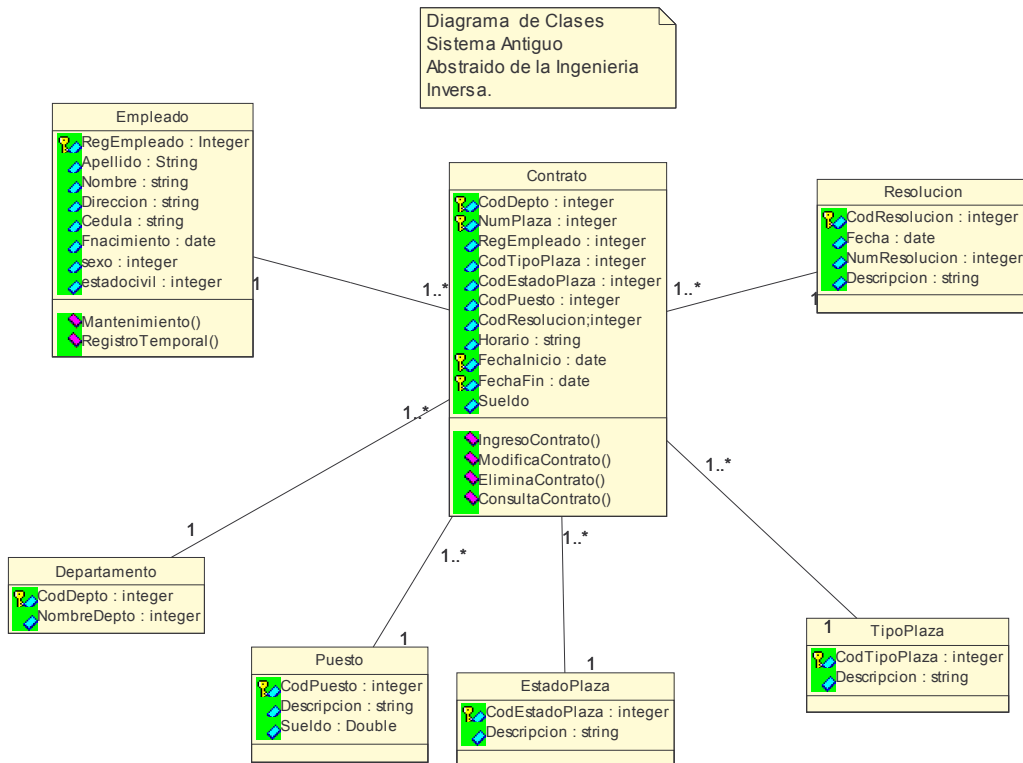
Figura 15. Abstracción de clases del sistema antiguo, caso de estudio

Diagrama de Clases
Sistema Antiguo
Abstraído de la Ingeniería
Inversa.



Después de ver un esquema generalizado del diagrama de clases, se agregan los atributos que se extraen de los campos equivalentes en entidades, tablas y archivos, además se agregan los métodos respectivos de cada clase, producidos de la lógica de los subprogramas relacionados con la clase equivalente.

Figura 16. Abstracción de clases detallado del sistema antiguo, caso de estudio



5.2.2. Rediseño del sistema de información con UWE

Con la definición de la arquitectura destino, se debe crear el nuevo diseño, partiendo del esquema conceptual que se obtuvo del sistema antiguo. Se agrega funcionalidad a los procesos antiguos, especificándose por medio de casos de uso, esto con el fin de producir la conceptualización para la arquitectura Web.

5.2.2.1. Nuevos escenarios de uso

Después del análisis y la determinación de los alcances del nuevo sistema, se puede redefinir los distintos escenarios de uso.

Mantenimiento de empleados, administración de los datos de empleados, se compone de, el ingreso de empleados nuevos, la actualización de información de los empleados existentes y la consulta de información de los mismos.

Ingreso o modificación de solicitudes de contratos, se ingresan los datos necesarios para solicitar la autorización del contrato, los cuales se hacen por medio de un formulario, esto se hace en cada departamento.

Autorización de solicitudes de contratos, administra la revisión de las solicitudes de contratos ingresadas en el sistema, con base al presupuesto se autoriza o rechaza según sea el caso, lo cual se lleva a cabo en la unidad de presupuesto.

Aprobación final y generación de contrato, la gerencia administra la aprobación de las solicitudes autorizadas ingresadas en el sistema, y genera los contratos seleccionados.

5.2.2.2. Especificación y diagramas de casos de uso

Los casos de uso proveen funcionalidad al sistema, en este sistema de administración de contratos se encuentran los siguientes casos de uso:

Nombre: **ingreso al sistema.**

Precondiciones: debe existir usuario en el sistema.

Flujo principal:

1. El usuario ingresa al sistema.
2. El sistema presenta formulario de ingreso.
3. El usuario ingresa claves y parámetros.
4. El sistema verifica y valida información ingresada.
5. El sistema presenta opciones según parámetros.

Flujo alternativo:

- 4.1 El sistema detectó parámetros incorrectos.
- 4.2 El sistema muestra mensaje de error.

Propósito: ingreso al sistema por usuarios válidos.

Nombre: **ingreso de empleado nuevo**

Precondiciones: el usuario debe haber ingresado al sistema, y estar dentro.

Flujo principal:

1. El sistema presenta formulario
2. El usuario ingresa información
3. El sistema verifica información ingresada.
4. El sistema retorna mensaje de verificación.
5. El usuario graba información
6. El sistema sigue su flujo básico

Flujo alternativo:

- 4.1 Se ingresó información incorrecta.
- 4.1 El sistema retorna mensaje de advertencia de error.
- 5.1 El sistema no pudo grabar.
- 5.2 El sistema retorna mensaje de error.

Propósito: agregar a la base de datos información de empleados que no existen en el sistema, para procesar contratos.

Nombre: **consulta de empleados**

Precondiciones: el usuario debe haber ingresado al sistema, y estar dentro. Debe existir información de empleados.

Flujo principal:

1. El sistema presenta formulario
2. El usuario ingresa parámetros de búsqueda
3. El sistema verifica información ingresada.
4. El sistema retorna información de búsqueda.
5. El sistema sigue su flujo básico

Flujo alterno:

- 2.1 El usuario ingresa parámetros de búsqueda por registro
 - 2.1.a El sistema sigue su flujo básico.
- 2.2 El usuario ingresa parámetros de búsqueda por nombre
 - 2.2.a El sistema sigue su flujo básico.
- 4.1 No existe información de parámetros de búsqueda.
- 4.2 El sistema retorna mensaje de error.

Propósito: consultar información acerca de empleados existentes en el sistema.

Nombre: **ingreso de solicitud**

Precondiciones: el usuario debe haber ingresado al sistema y estar dentro.

Flujo principal:

1. El sistema presenta formulario de solicitud.
2. El usuario ingresa información
3. El sistema verifica información ingresada.
4. El sistema retorna mensaje de verificación.
5. El usuario graba información

6. El sistema procesa la grabación.
7. El sistema sigue su flujo básico

Flujo alterno:

- 4.1 Se ingresó información incorrecta.
- 4.2 El sistema retorna mensaje de advertencia de error.
- 6.1 El sistema no pudo grabar.
- 6.2 El sistema retorna mensaje de error.

Propósito: ingreso de solicitudes de contrato, para su trámite.

Nombre: **consulta de solicitud**

Precondiciones: el usuario debe ingresar al sistema, y estar dentro. Debe existir información de solicitudes.

Flujo principal:

1. El sistema presenta listado de solicitudes ingresadas
2. El usuario selecciona solicitud deseada
3. El sistema busca información de solicitud.
4. El sistema muestra información de solicitud.
5. El sistema sigue su flujo básico

Propósito: consultar información acerca de empleados, existentes en el sistema.

Nombre: **modificación de solicitud**

Precondiciones: el usuario debe ingresar al sistema, y estar dentro.

Flujo principal:

1. El sistema presenta formulario con información de solicitud.
2. El usuario ingresa información a cambiar.
3. El sistema verifica información ingresada.

4. El sistema retorna mensaje de verificación.
5. El usuario graba información.
6. El sistema guarda información.
7. El sistema sigue su flujo básico.

Flujo alternativo:

- 4.1 Se ingresó información incorrecta.
- 4.2 El sistema retorna mensaje de advertencia de error.
- 6.1 El sistema no pudo grabar.
- 6.2 El sistema retorna mensaje de error.

Propósito: corrección de solicitudes de contrato, para su trámite.

Nombre: **autorización de solicitud**

Precondiciones: el usuario debe ingresar al sistema, y estar dentro. Debe existir información de solicitudes.

Flujo principal:

1. El sistema presenta listado de solicitudes ingresadas.
2. El usuario selecciona solicitud deseada.
3. El sistema busca información de solicitud.
4. El sistema muestra información de solicitud.
5. El usuario autoriza solicitud de contrato.
6. El sistema actualiza información.
7. El sistema sigue su flujo básico.

Flujo alternativo:

- 5.1 El usuario no autoriza solicitud de contrato.
- 5.2 El usuario ingresa observaciones de rechazo.
- 5.3 El sistema sigue su flujo básico.
- 6.1 El sistema no pudo actualizar información.
- 6.2 El sistema muestra mensaje de error.

Propósito: autorización presupuestaria de solicitud de contrato de personal.

Nombre: **aprobación de contrato**

Precondiciones: el usuario debe ingresar al sistema, y estar dentro. Debe existir información de solicitudes autorizadas únicamente.

Flujo principal:

1. El sistema presenta listado de solicitudes autorizadas.
2. El usuario selecciona solicitud deseada.
3. El sistema busca información de solicitud.
4. El sistema muestra información de solicitud.
5. El usuario aprueba contrato.
6. El sistema actualiza información.
7. El sistema sigue su flujo básico.

Flujo alterno:

- 5.1 El usuario no aprueba contrato.
- 5.2 El usuario ingresa observaciones de rechazo.
- 5.3 El sistema sigue su flujo básico.
- 6.1 El sistema no pudo actualizar información.
- 6.2 El sistema muestra mensaje de error.

Propósito: aprobación final de contrato de personal.

Nombre: **generación de contrato**

Precondiciones: el usuario debe haber ingresado al sistema, y estar dentro. Debe existir información de solicitudes autorizadas (contratos aprobados).

Flujo principal:

1. El sistema presenta lista de contratos aprobados.
2. El usuario selecciona contrato deseado.

3. El sistema busca información de contrato.
4. El sistema genera contrato.
5. El sistema sigue su flujo básico.

Flujo alternativo:

- 4.1 El sistema tuvo problemas para generar contrato.
- 4.2 El sistema muestra mensaje de error.

Propósito: generación de contratos aprobados por gerencia, con la información de las solicitudes ingresadas.

De la especificación anterior se pueden visualizar varios casos de uso, e incluir varios en un sólo diagrama, como un contexto de procesos unificados.

Figura 17. Diagrama de caso de uso, ingreso al sistema

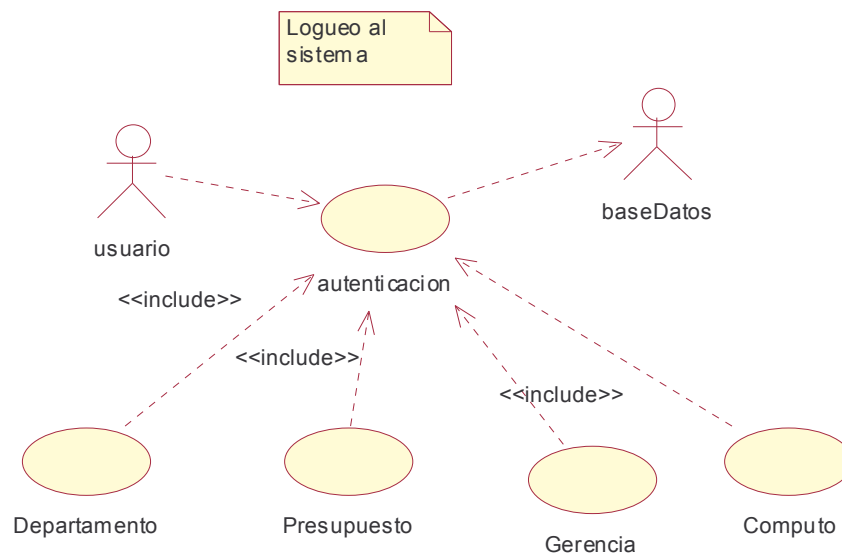


Figura 18. Diagrama de caso de uso, mantenimiento de empleados

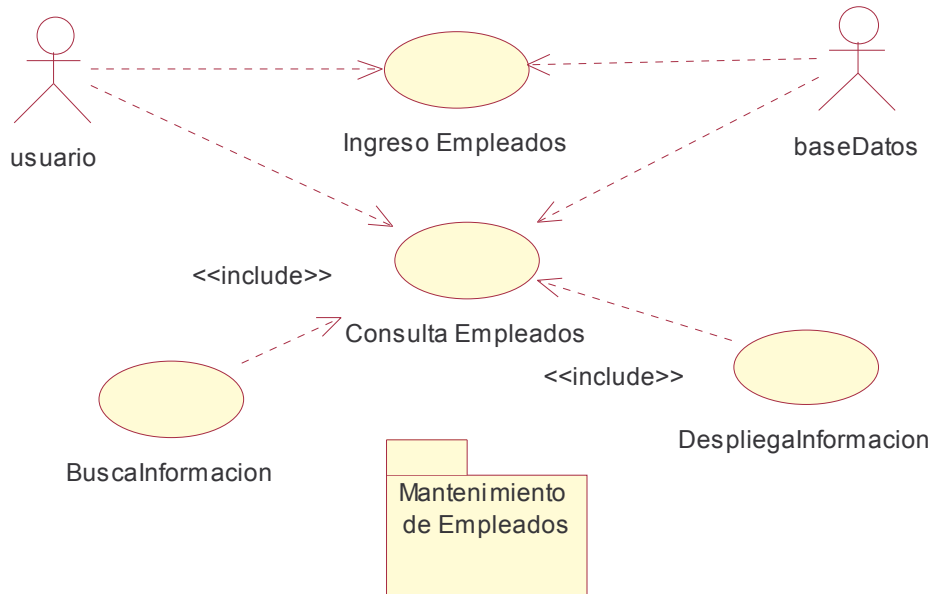
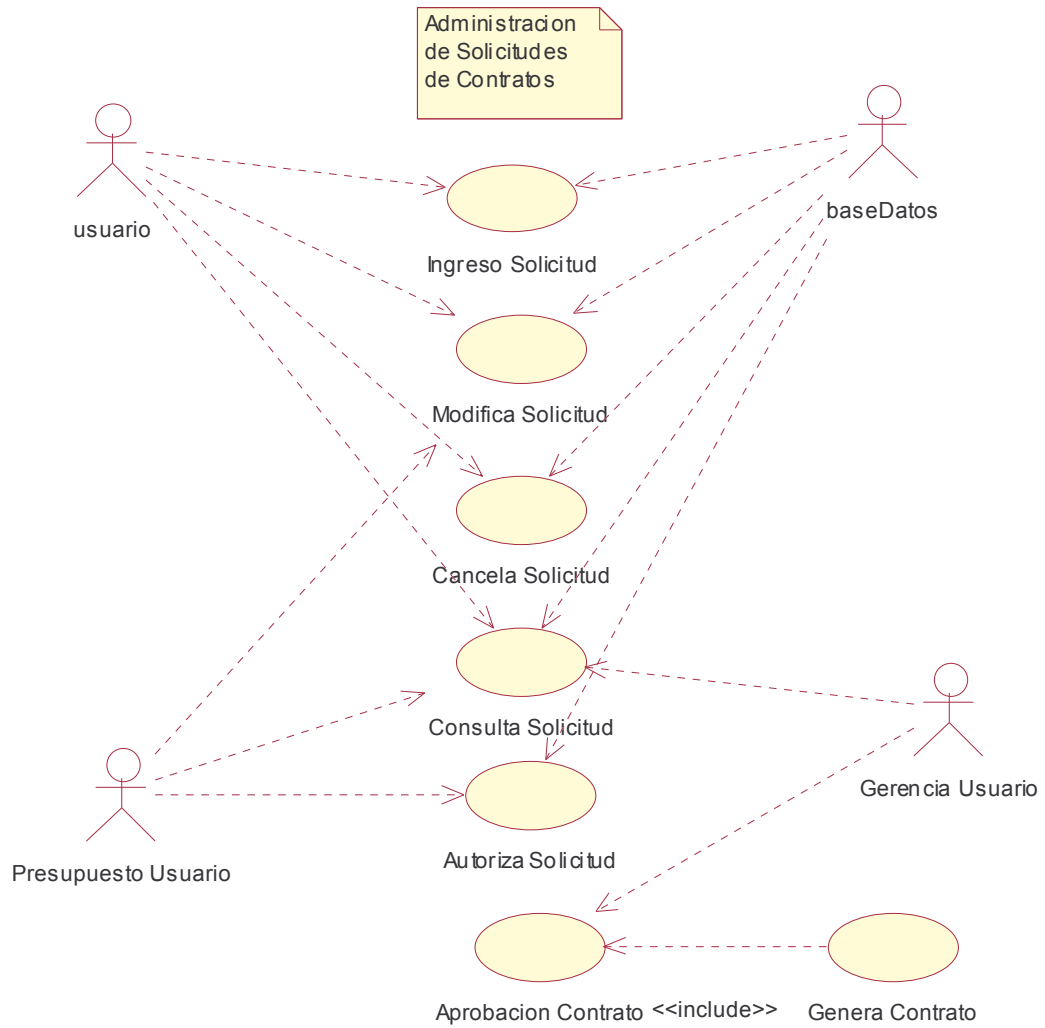
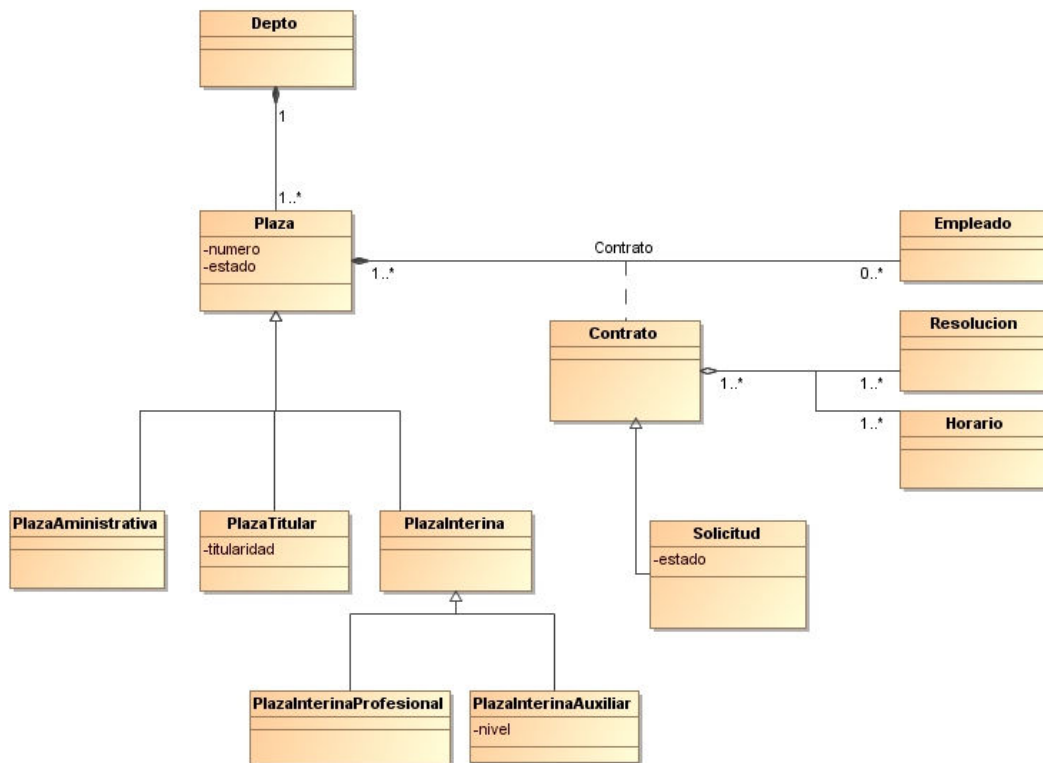


Figura 19. Diagrama de caso de uso, contexto de administración de personal



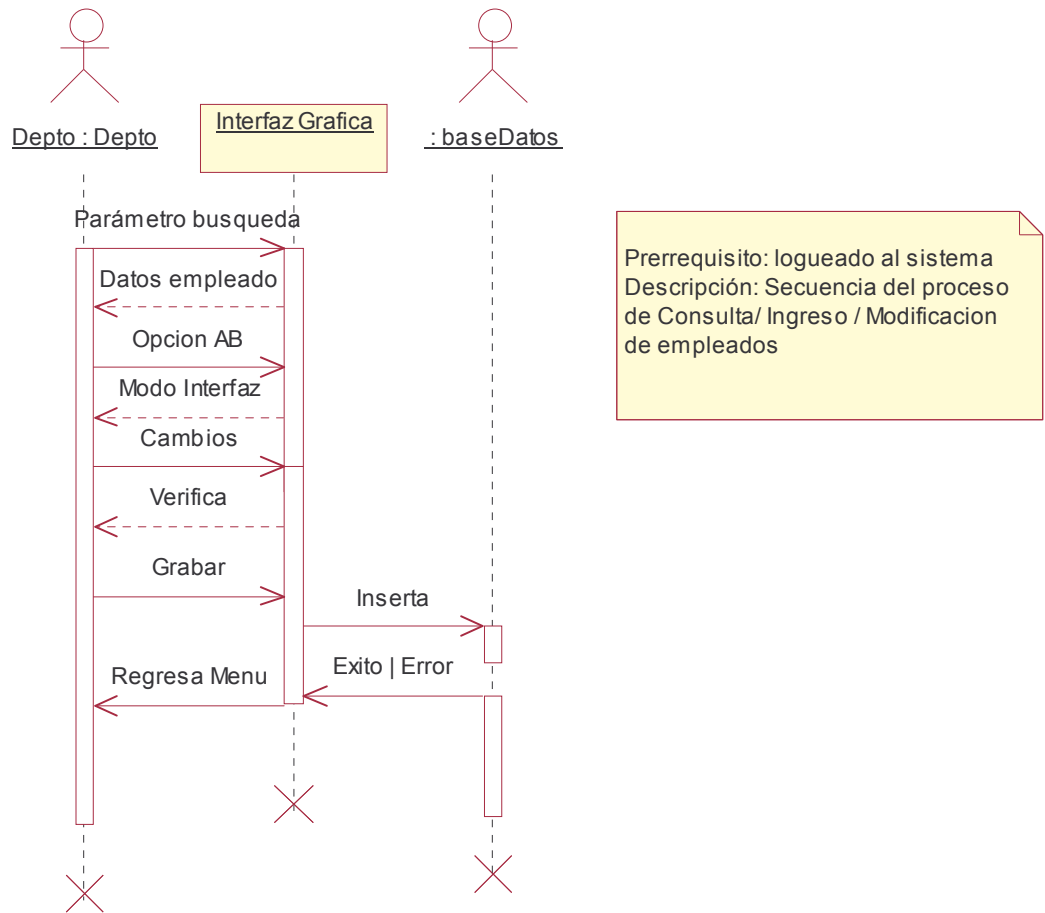
5.2.2.3. Diagrama de clases

Figura 20. Vista conceptual rediseñada, diagrama de clases



5.2.2.4. Diagramas de secuencia

Figura 21. Diagrama de secuencia para la altas, bajas y consultas de empleados



22. Diagrama de secuencia, ingreso de solicitud

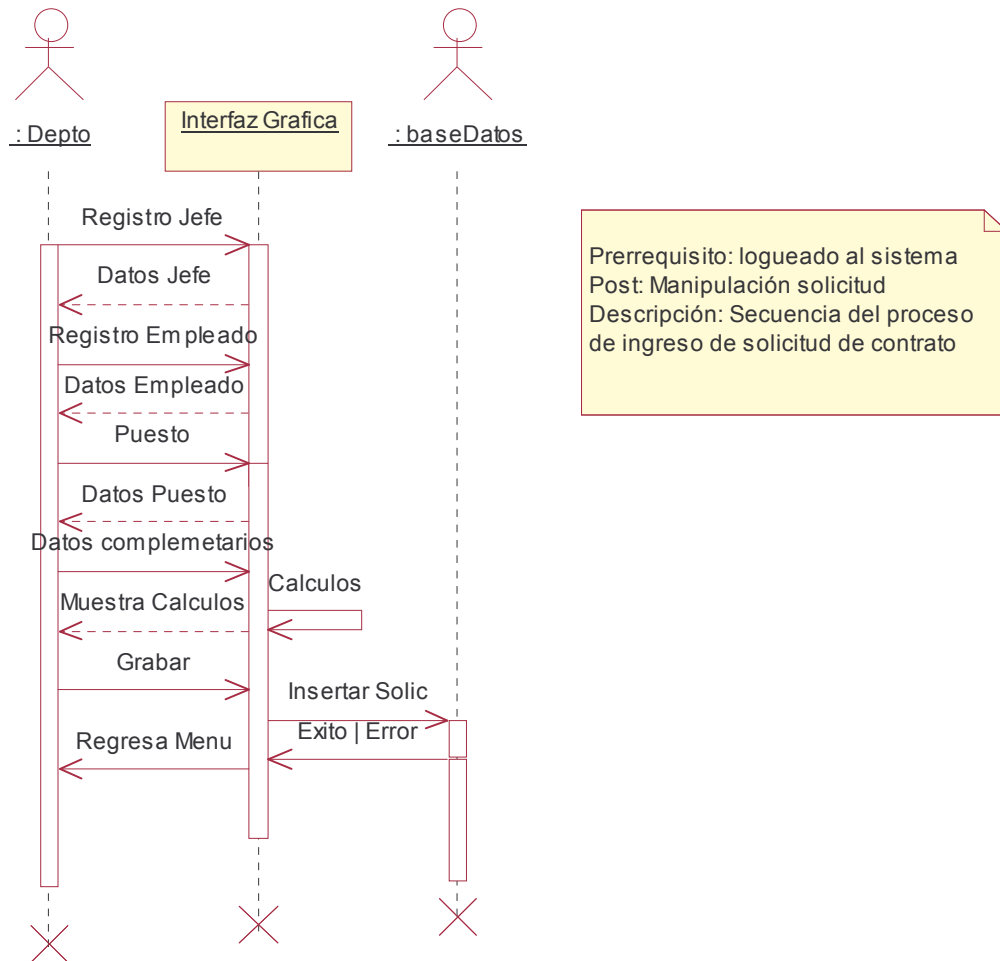
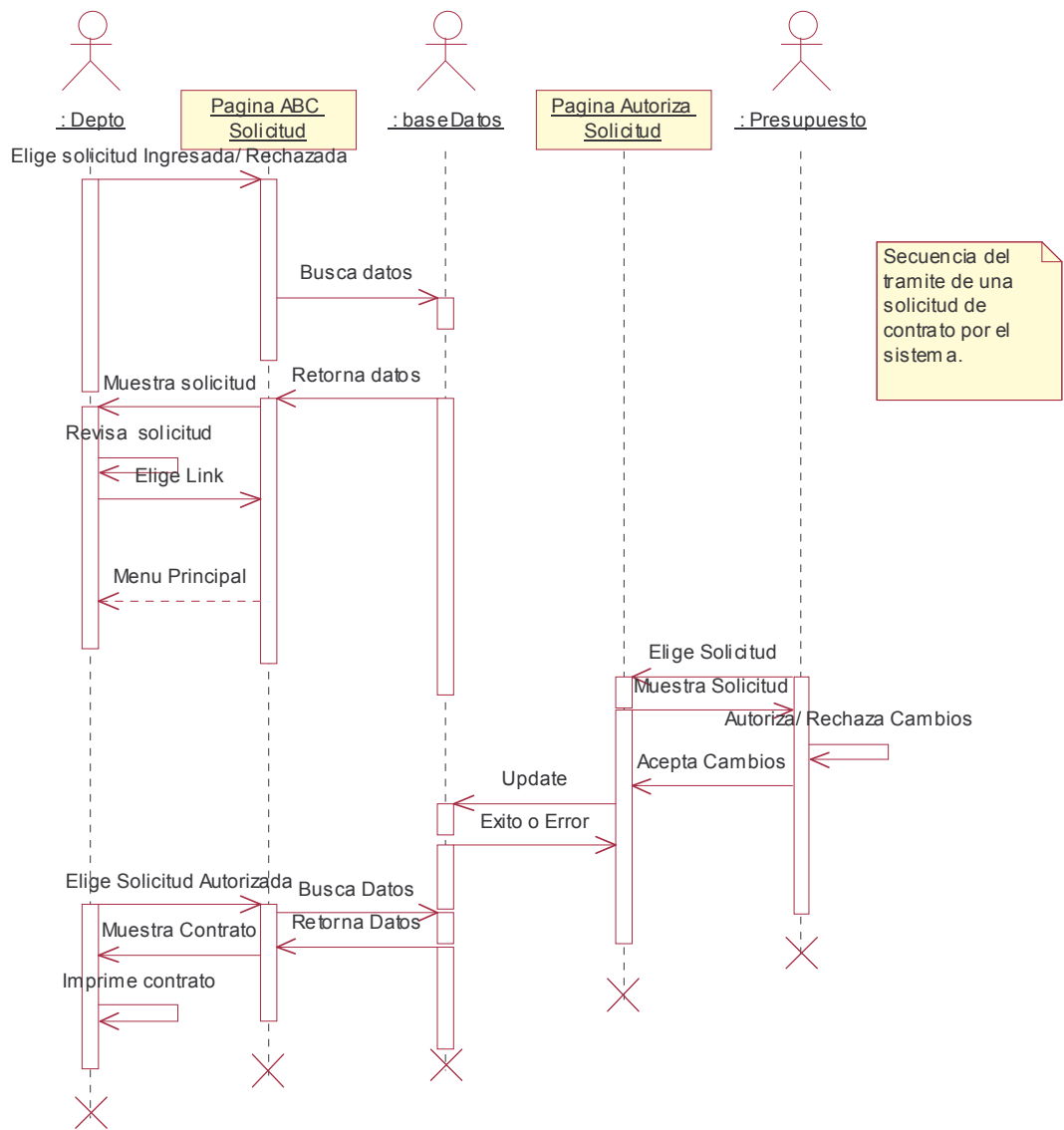


Figura 23. Diagrama de secuencia del proceso de autorización y generación de contratos



5.2.2.5. Diagramas de colaboración

Figura 24. Diagrama de colaboración, ingreso de solicitudes de contratos

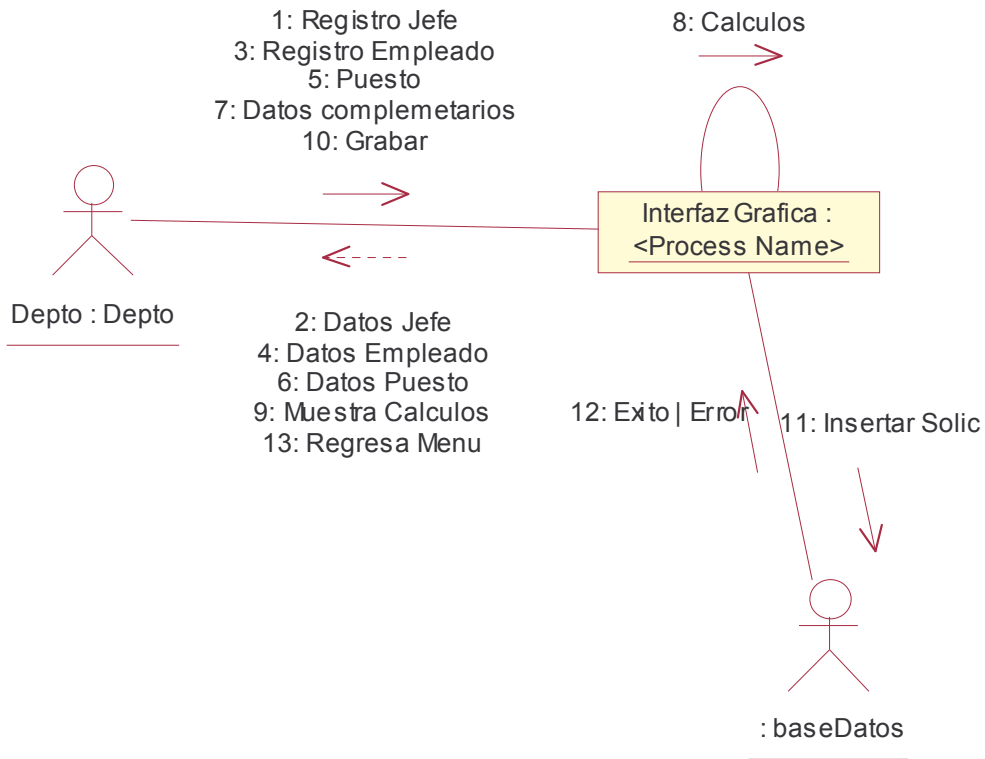


Figura 25. Diagrama de colaboración, administración de solicitudes de contratos

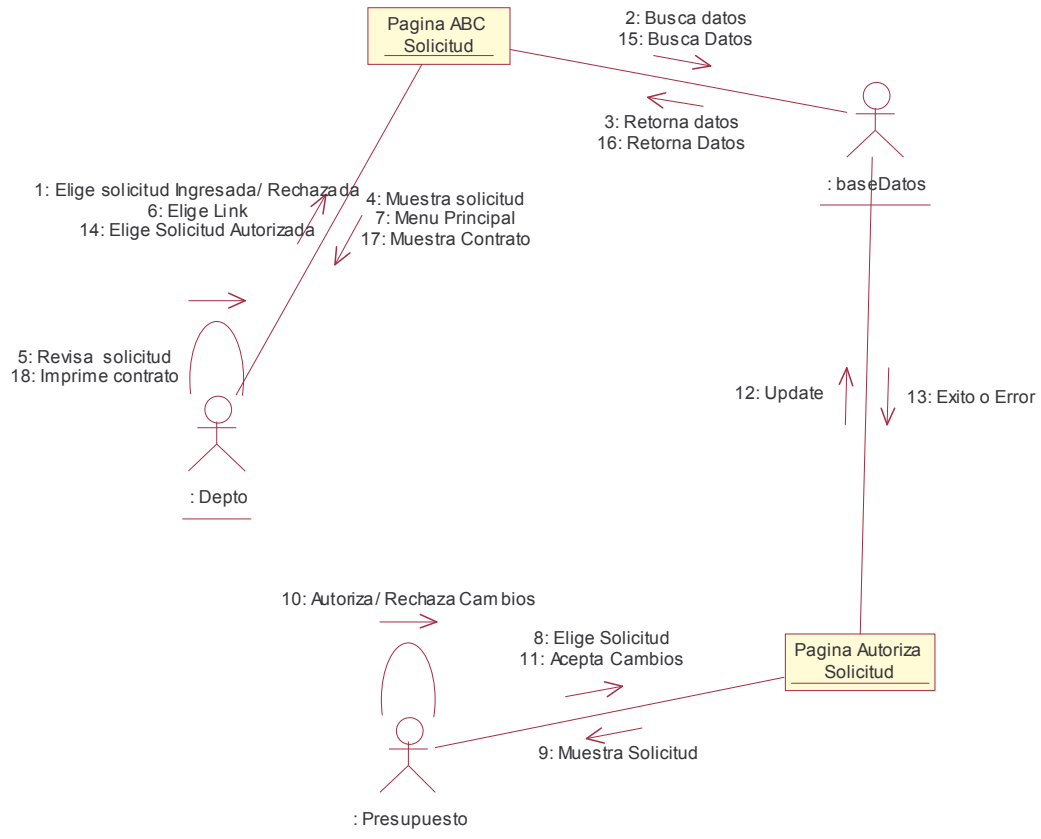
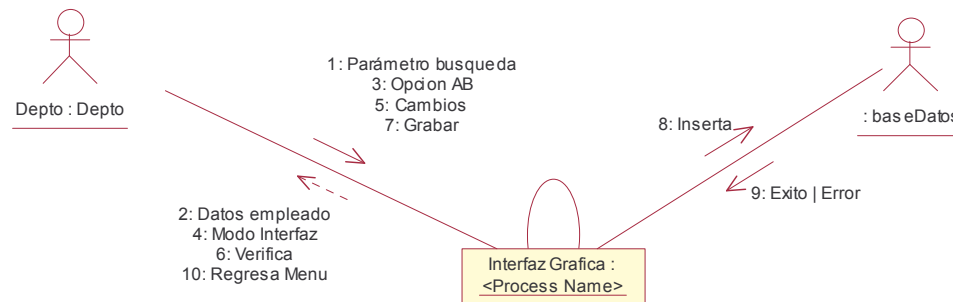


Figura 26. Diagrama de colaboración, mantenimiento de empleados



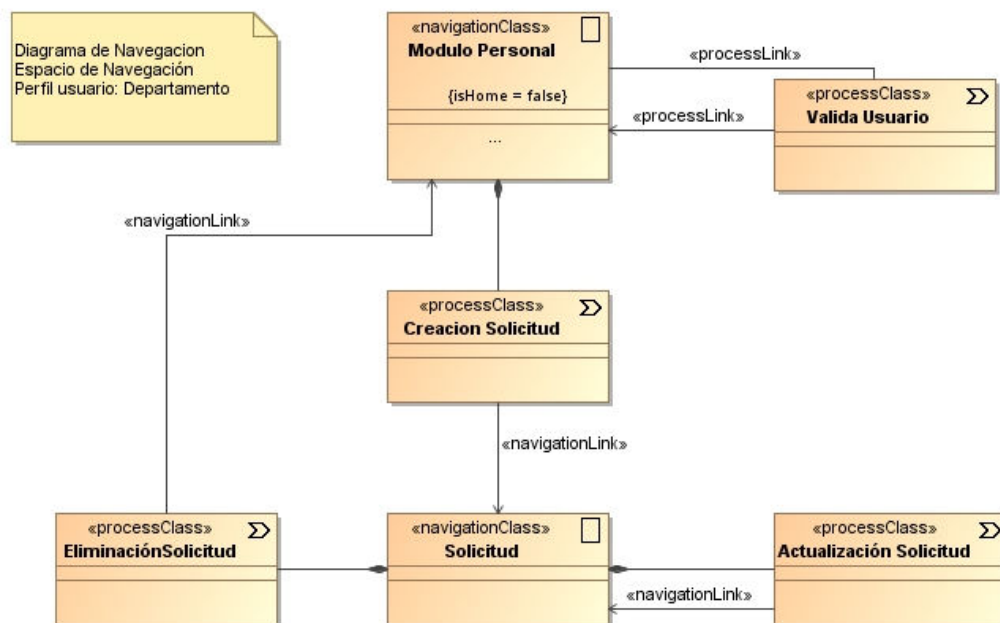
5.2.3. Modelando el sistema con una arquitectura web

5.2.3.1. Integración de procesos en el modelo de navegación

5.2.3.1.1. Modelado del espacio navegacional

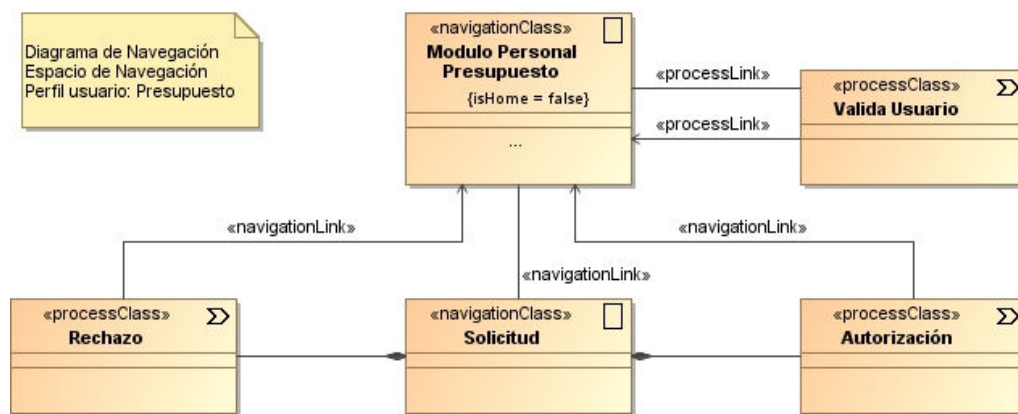
A continuación, se modela cómo alcanzar las clases de navegación y qué procesos principales intervienen para esto. Se puede observar el proceso de autenticación de usuario en cada diagrama, el cual determinará el perfil de usuario, y los diferentes objetos en el flujo de navegación.

Figura 27. Diagrama de espacio de navegación por departamento



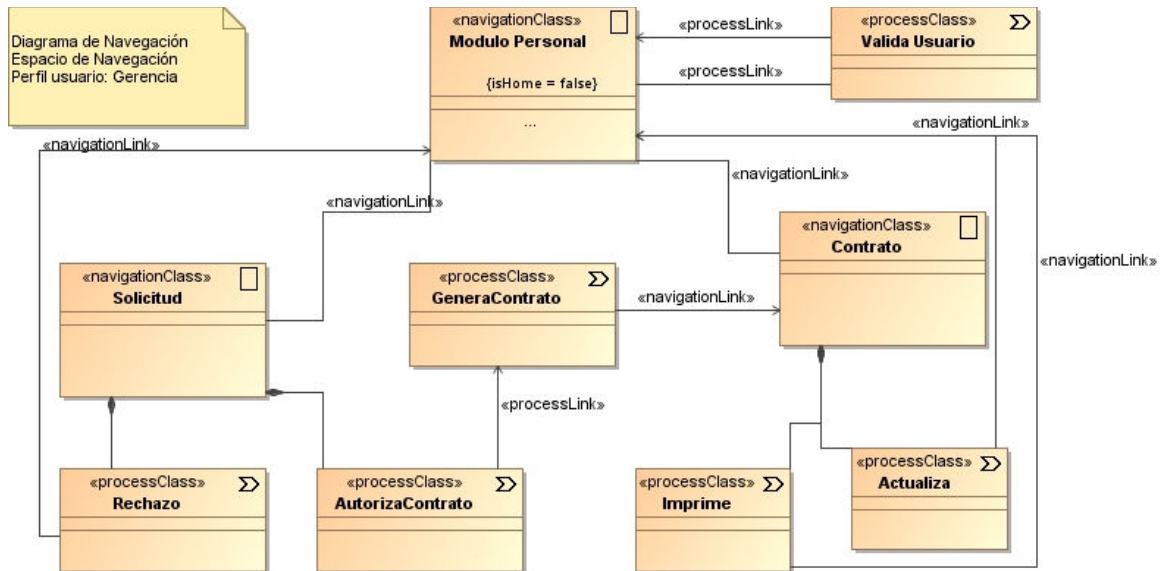
Siguiendo la descripción de los escenarios de uso, los usuarios con perfil Departamento son los encargados de las altas, bajas, modificaciones y consultas de las solicitudes para contrataciones. En la siguiente figura se observa que para los usuarios con el perfil de la unidad de Presupuesto, a partir de la página principal se puede navegar hacia la solicitud, para ser autorizada o bien rechazada, mediante dos procesos respectivos.

Figura 28. Diagrama de espacio de navegación para presupuesto



Para los usuarios con el perfil de Gerencia, a través de la página principal llegarán a la clase solicitud, con estado autorizada, ésta contiene dos procesos, uno de rechazo y el otro de autorización (aprobación de contrato), que precede la secuencia hacia el proceso de generación de contrato, el cual se dirige hacia la clase Contrato; esta clase también contiene dos procesos uno que genera la impresión del contrato y el otro que actualiza información editada.

Figura 29. Diagrama de espacio de navegación para gerencia



5.2.3.1.2. Estructura del modelado navegacional

Estas presentaciones enriquecen los diagramas anteriores con un detalle más amplio, incluyendo estereotipos propios del ambiente *web*, como son los índices, menús, consultas y vistas guiadas. Cada uno se refiere al mismo escenario de los diagramas de la sección anterior respectivamente.

Figura 30. Diagrama de estructura de navegación por departamento

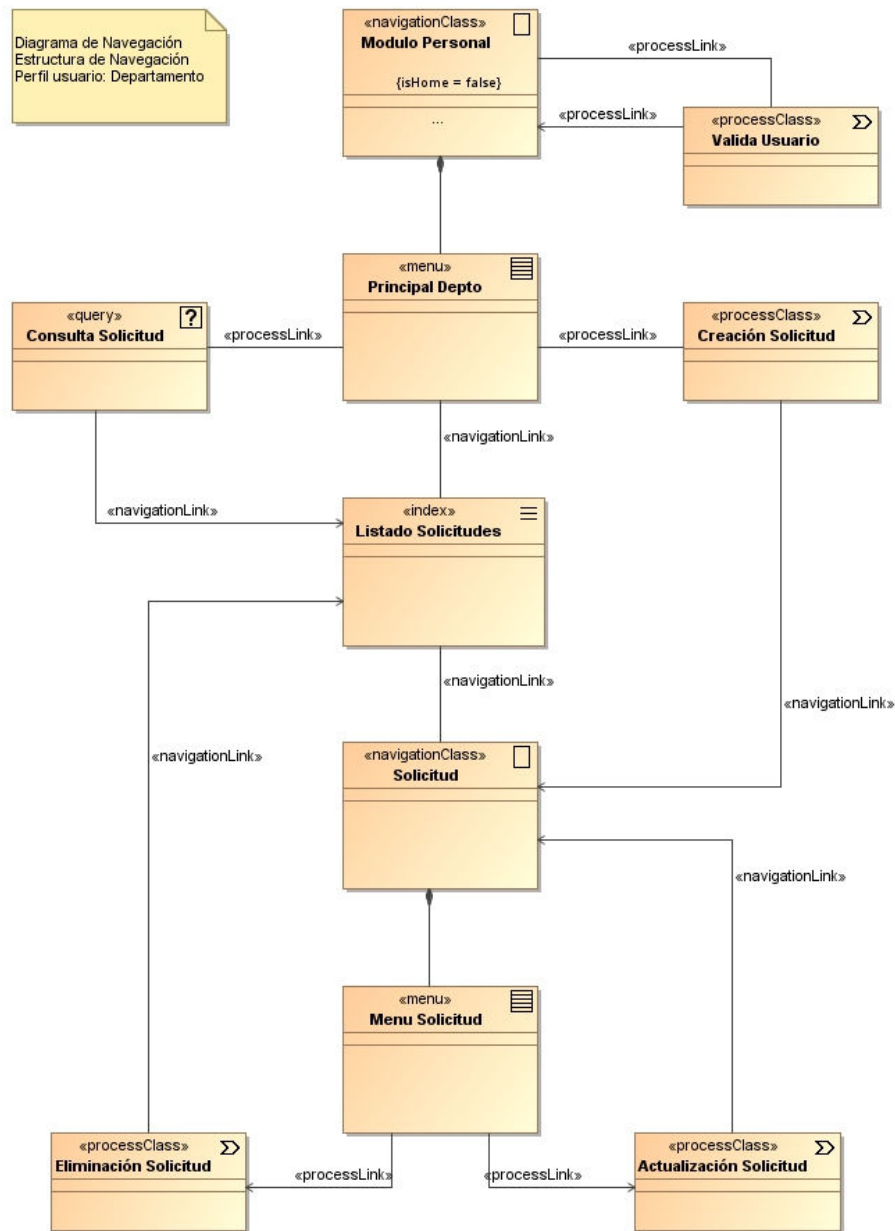


Figura 31. Diagrama de estructura de navegación para presupuesto

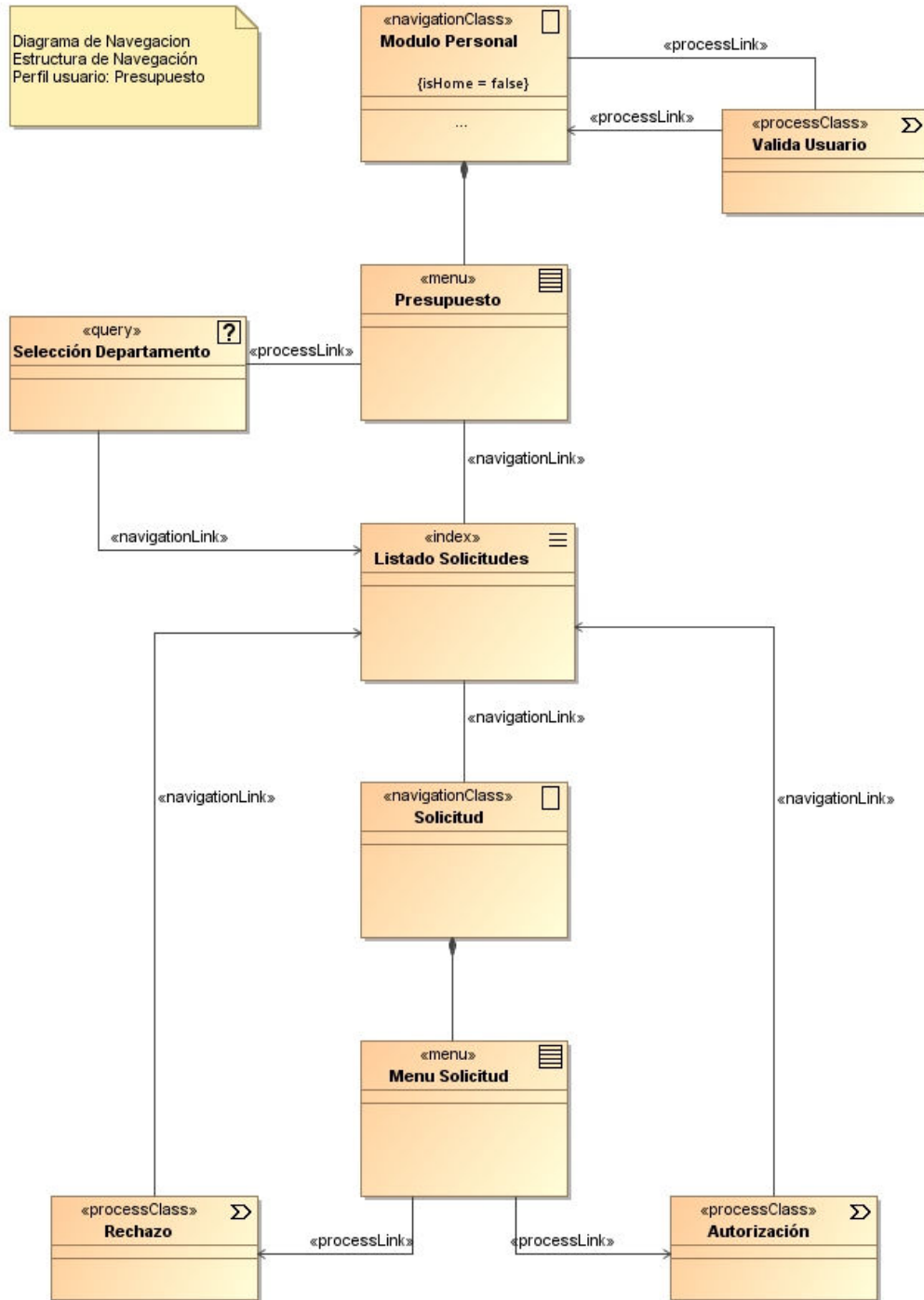
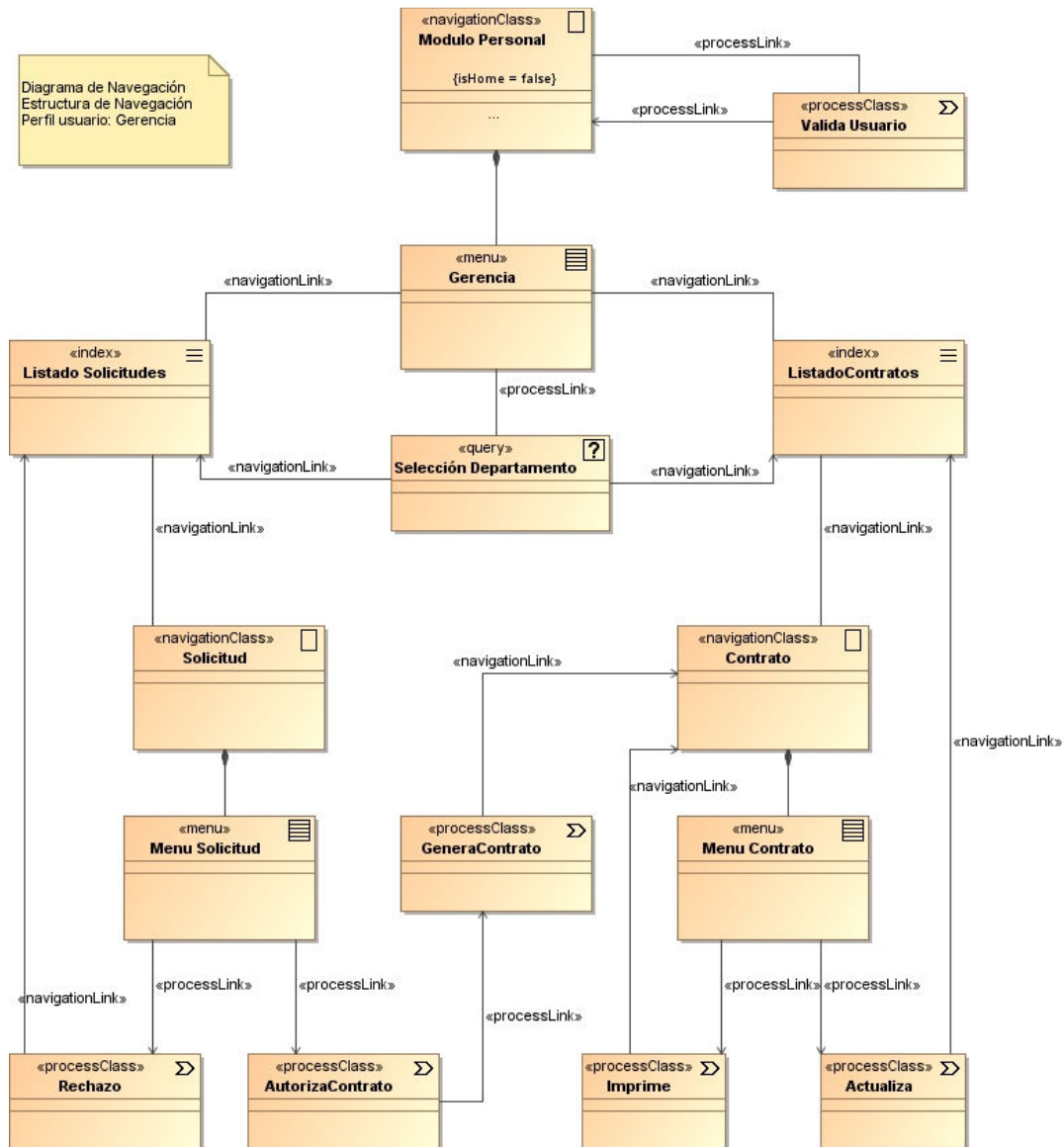


Figura 32. Diagrama de estructura de navegación para gerencia



5.2.3.2. Afinando el modelado de procesos

5.2.3.2.1. Modelo estructural del proceso

Por medio de éste se presentan las relaciones entre los distintos procesos, que se asocian a las clases principales de la aplicación en análisis; se puede observar los procesos relacionados con las clases identificadas, Solicitud, Creación y Actualización, se asocian con una superclase que se llamada ProcesamientoSolicitud, la cual les hereda características principales. Igualmente se observa la relación entre los distintos procesos de confirmación con una superclase que generaliza el proceso básico. La clase ProcesamientoSolicitud se compone de la clase IngresoInfoSolicitud que asigna valores a los atributos de la clase conceptual.

Figura 33. Diagrama de estructura de proceso para procesamiento de solicitud

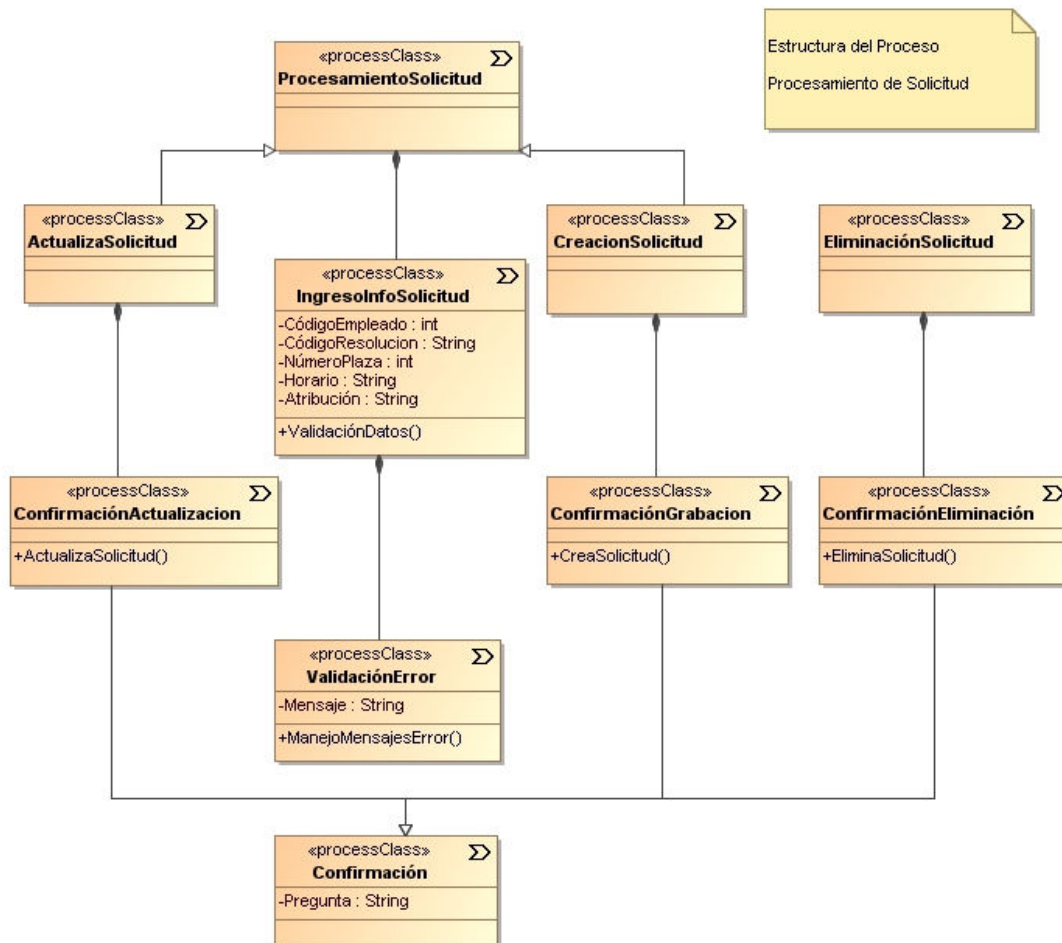
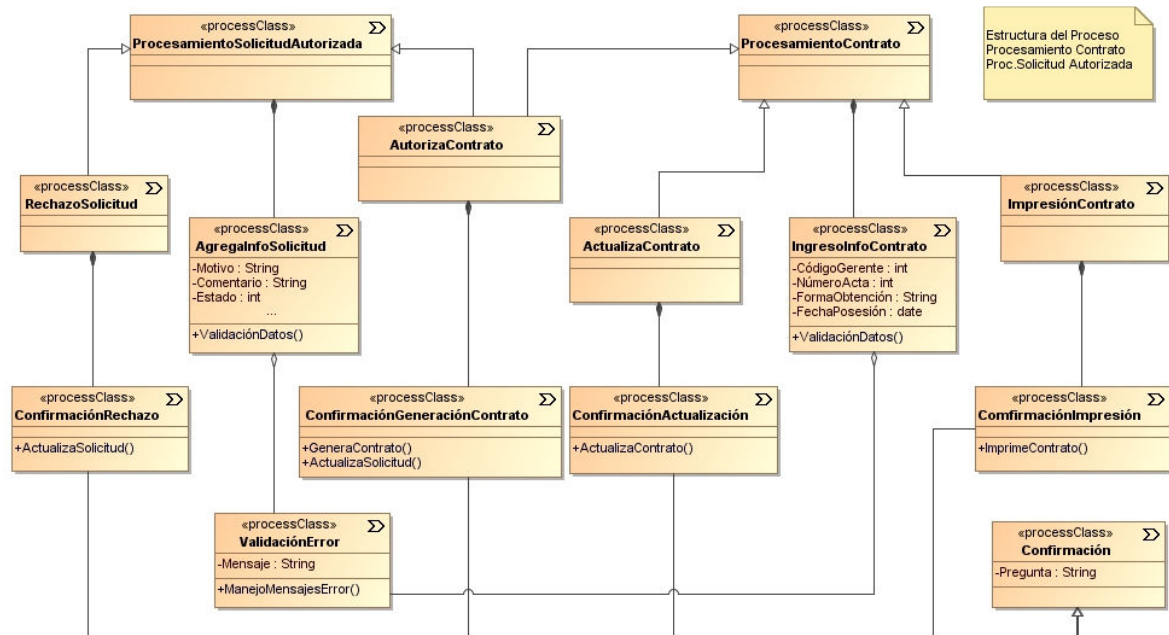


Figura 34. Diagrama de estructura de proceso para procesamiento de contrato y solicitud autorizada

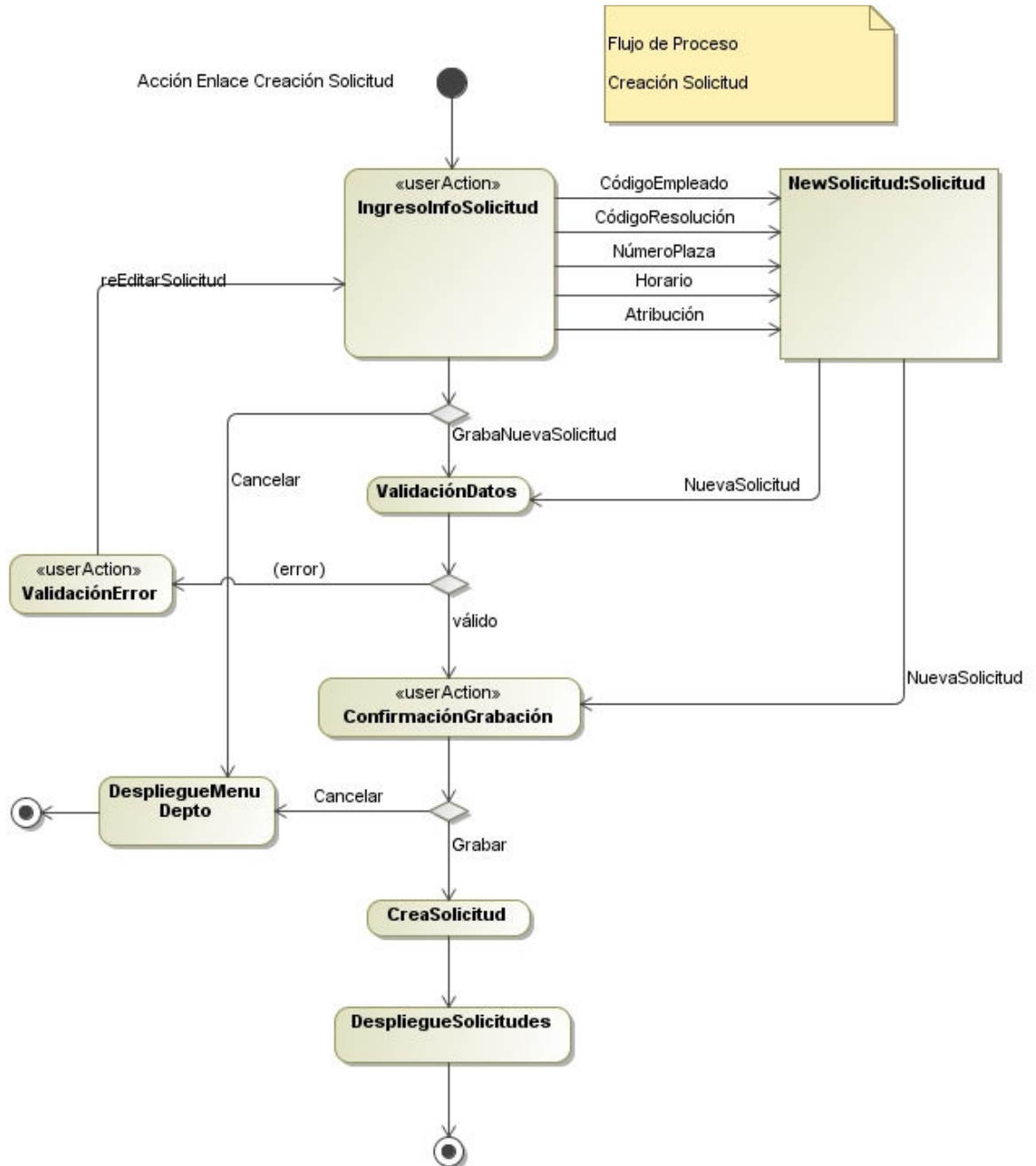


En este diagrama se puede observar la generalización del proceso Autoriza Contrato, que hereda características de las superclases de proceso ProcesamientoSolicitud y ProcesamientoContrato, ya que una solicitud es aprobada para convertirse en contrato,

5.2.3.2.2. Modelo de comportamiento del proceso

Por medio de diagramas de actividades se representa el flujo de un proceso, derivado de la acción que produzca un objeto de navegación.

Figura 35. Diagrama de flujo de proceso para creación de solicitud



En el diagrama anterior se observa el comportamiento del procedimiento que se produce cuando se selecciona la opción creación de solicitud, en el menú del módulo para los usuarios con perfil por departamento. Se observa el flujo principal entre métodos y procedimientos estereotipados, que interactúan con el usuario en la creación de una solicitud, además la relación con la instancia del objeto solicitud. El siguiente diagrama muestra el flujo de proceso que se produce al presionar el botón elimina solicitud en la clase de navegacion solicitud.

Figura 36. Diagrama de flujo de proceso para eliminación de solicitud

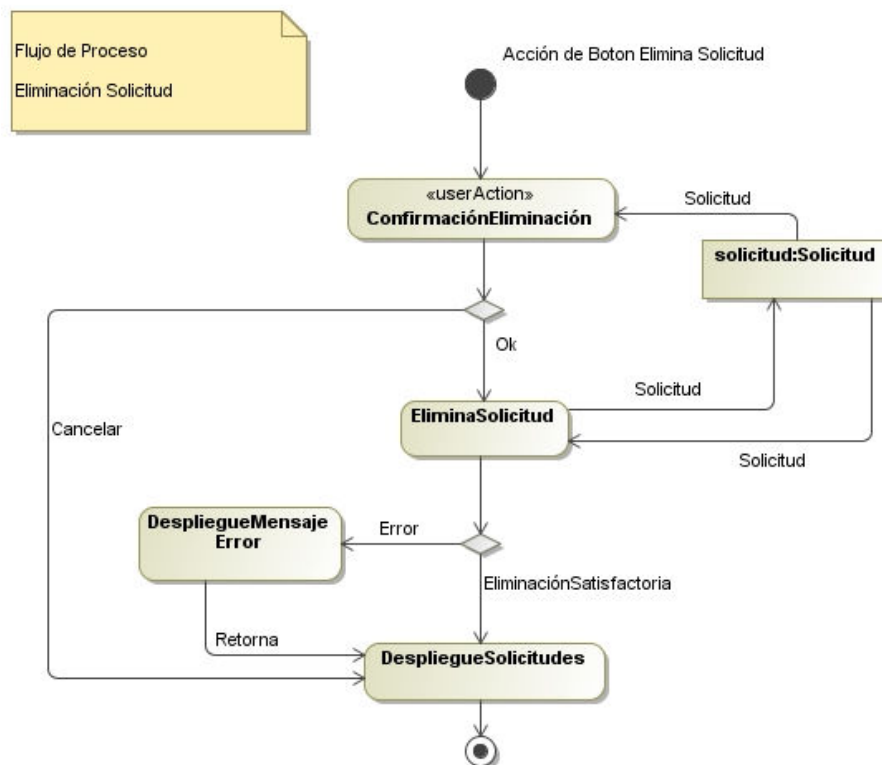
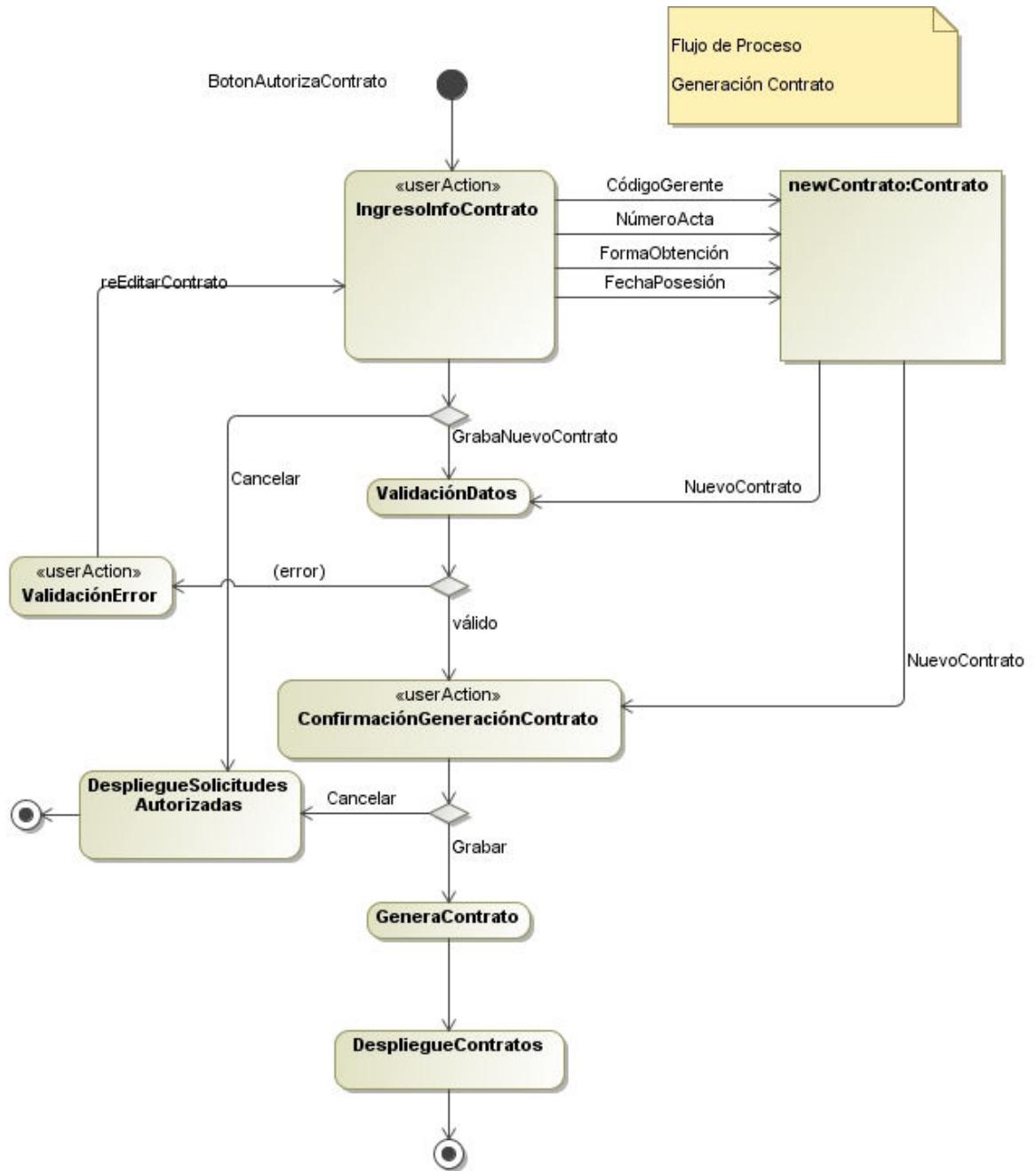


Figura 37. Diagrama de flujo de proceso para generación de contrato



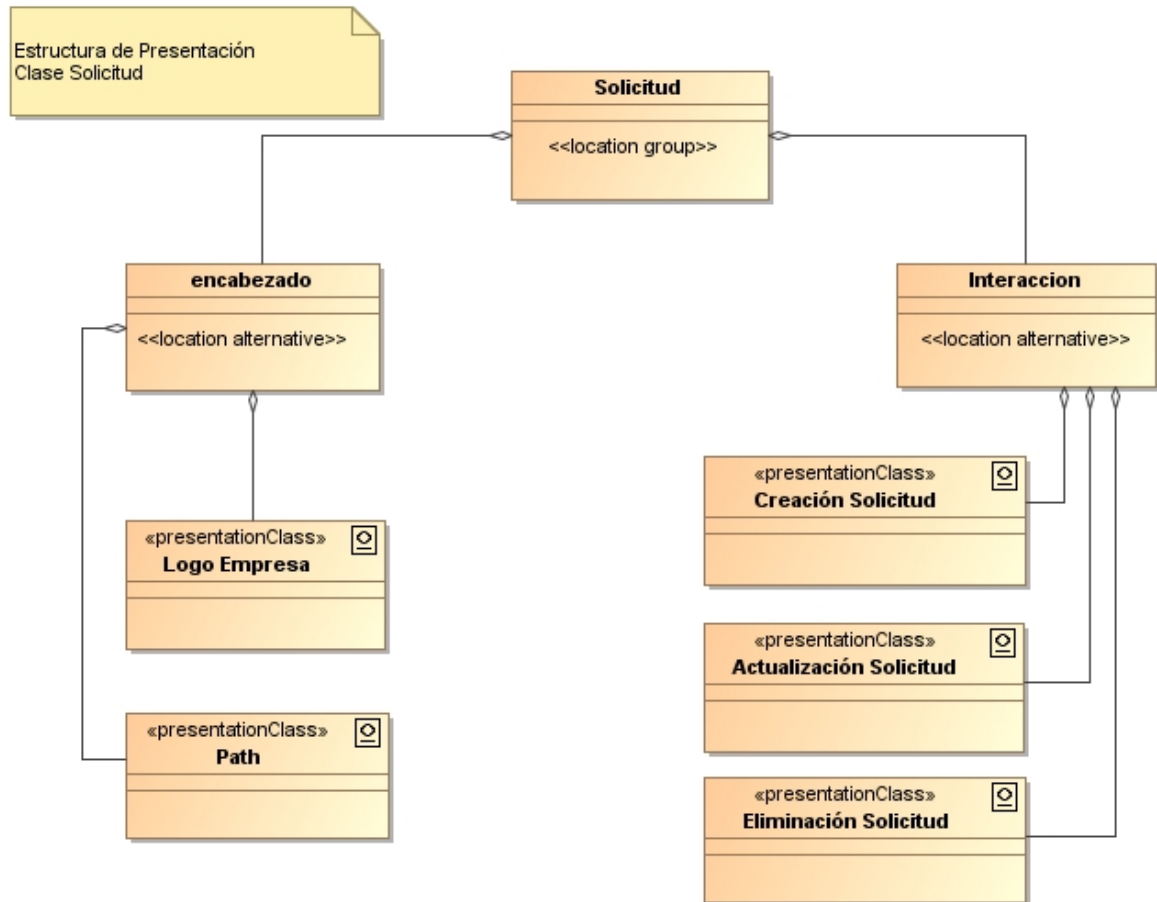
Esta secuencia de pasos se produce al presionar el botón autoriza contrato, desde la clase de navegación solicitud, con estado autorizada, la cual genera y crea una instancia de la clase contrato.

5.2.3.3. Soporte de procesos en el modelo de presentación

5.2.3.3.1. Estructura visual

Por medio de éste se muestran los elementos de presentación que ocupan el espacio de una clase de navegación, pero no en el mismo tiempo. A continuación el diagrama describe la clase de navegación Solicitud, y la estructura como un grupo de ubicación, con la clase estereotipada <<location group>>, que contiene una sección de encabezado y una parte de interacción por medio de secciones alternativas <<location alternative>>. Encabezado, agrega una clase para logotipo y otra para la ruta de navegación Path. La sección Interacción, agrega tres distintas clases de presentación, las cuales son de creación, actualización y eliminación.

Figura 38. Diagrama de presentación, estructura visual para página de solicitud

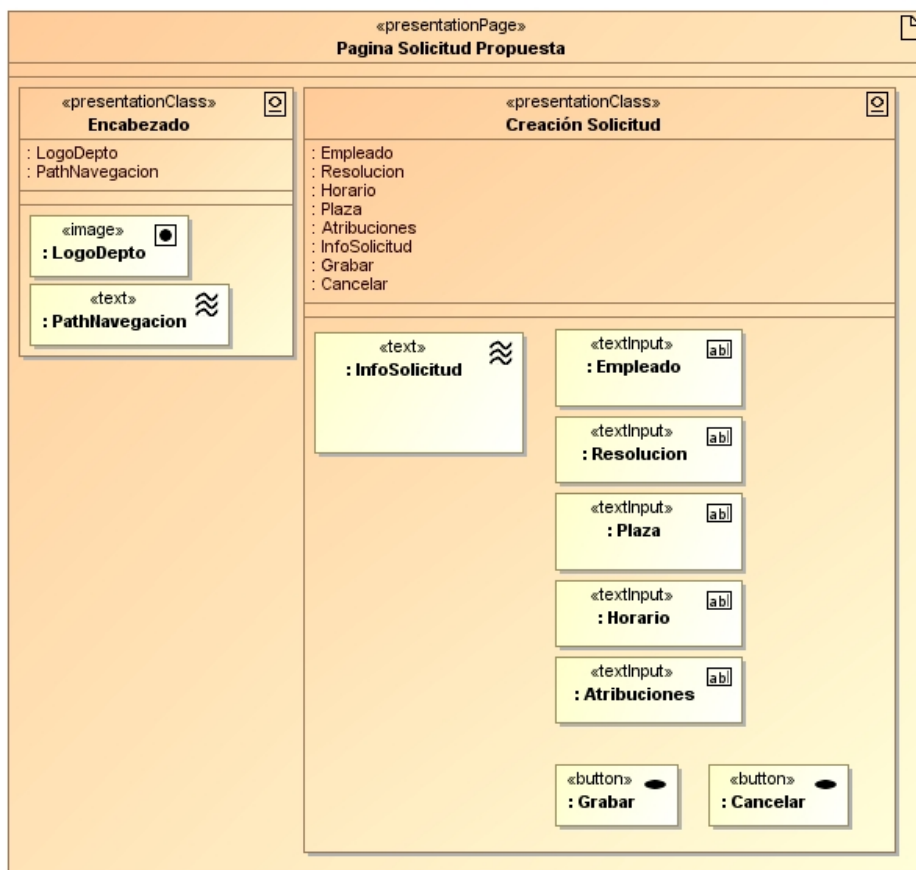


5.2.3.3.2. Interfaz de usuario

Muestra el contenido visual por medio de elementos físicos de interfaz de usuario. En el siguiente diagrama se observa el contenido para la página de solicitud de propuesta, correspondiente a la clase de navegación llamada Solicitud. La estructura detallada en el diagrama anterior, se puede ver aplicada, la sección de encabezado y la sección de interacción.

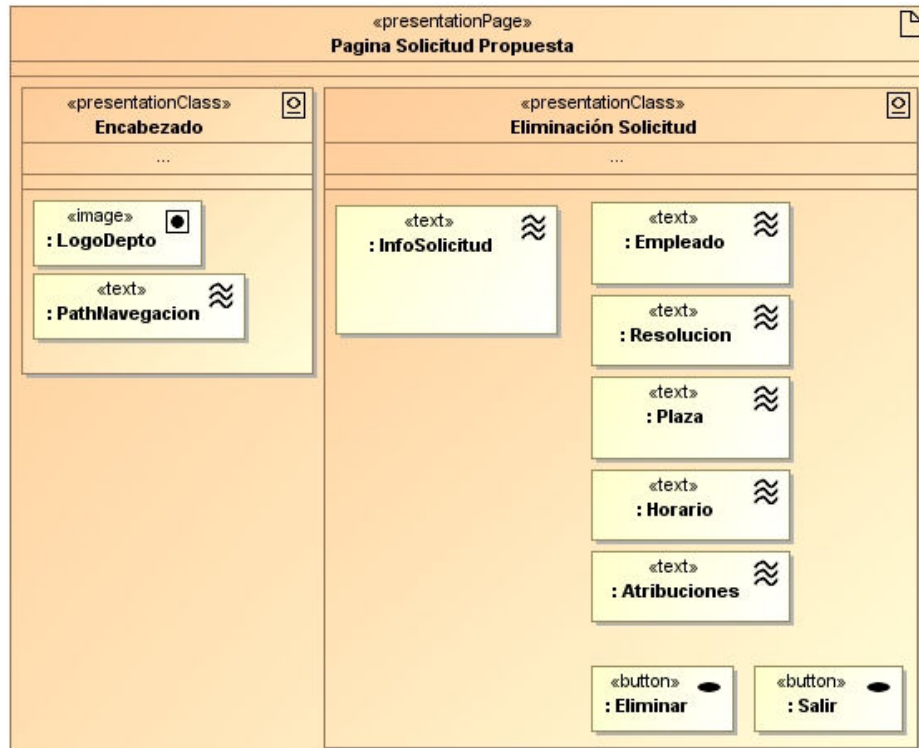
Para este caso particular, se muestra el contenido presentado en el tiempo de ejecución correspondiente a la sección creación de solicitud, se pueden ver sus respectivos elementos de interfaz de usuario, elementos de entrada de texto, botón grabar, botón cancelar.

Figura 39. Diagrama de presentación, interfaz de usuario, creación de solicitud



En el siguiente diagrama se puede notar algunas diferencias con respecto al anterior, entre ellas el objetivo, una de creación y la otra de eliminación, el tipo de estereotipo para presentar los datos cambia, el anterior como entrada de datos y el siguiente como texto de despliegue y no de edición.

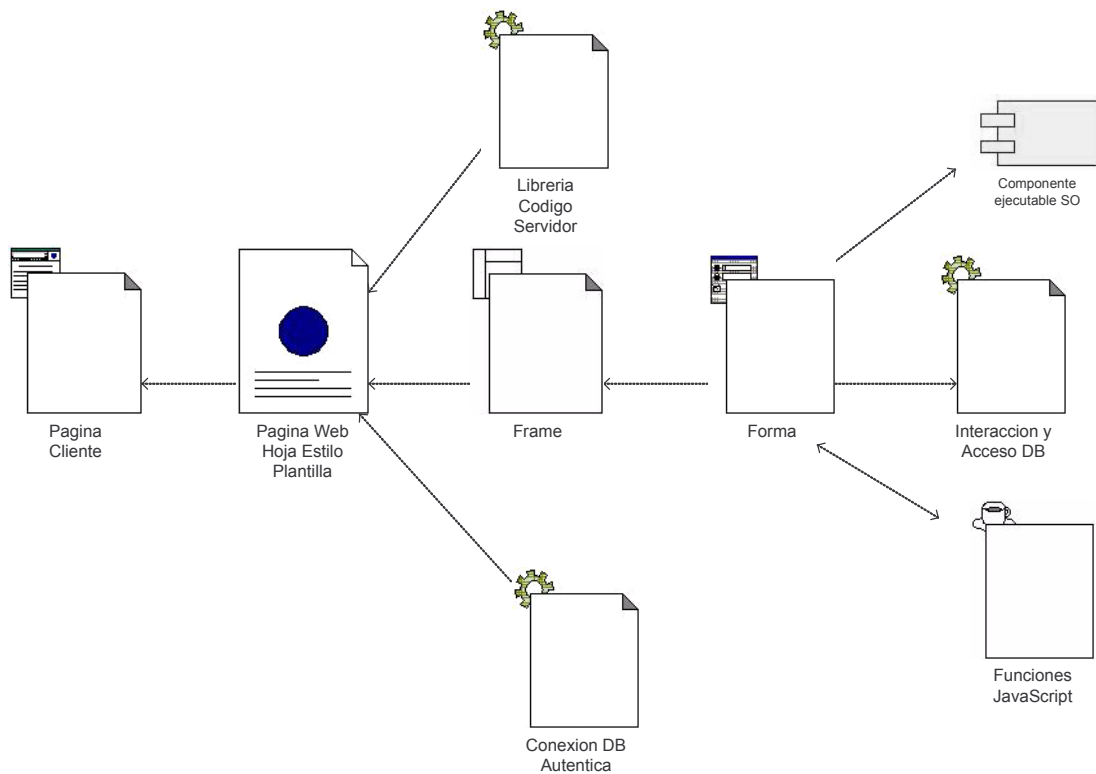
Figura 40. Diagrama de presentación, interfaz de usuario, eliminación de solicitud



5.2.4. Diagramas de componentes y despliegue

Para modelar la organización y dependencia entre objetos físicos del sistema como ejecutables, archivos, librerías, tablas y artefactos de este tipo, UML sugiere los diagramas de componentes, y sugiere los diagramas de despliegue para mostrar la configuración de nodos en tiempo de ejecución. A continuación se muestra la estructura de las páginas para el diseño de la aplicación *Web* en estudio, la cual constará de varios componentes sugeridos como se puede observar, páginas de estilo, librerías de autenticación, funciones java script y componentes propios del sistema operativo.

Figura 41. Diagrama de componentes, estructura de páginas



En los siguientes diagramas se puede observar la configuración de instancias de componentes y sus relaciones físicas en tiempo de ejecución, para los distintos perfiles que manejará la interacción con la aplicación.

Figura 42. Diagrama de despliegue para usuarios de los departamentos de la empresa

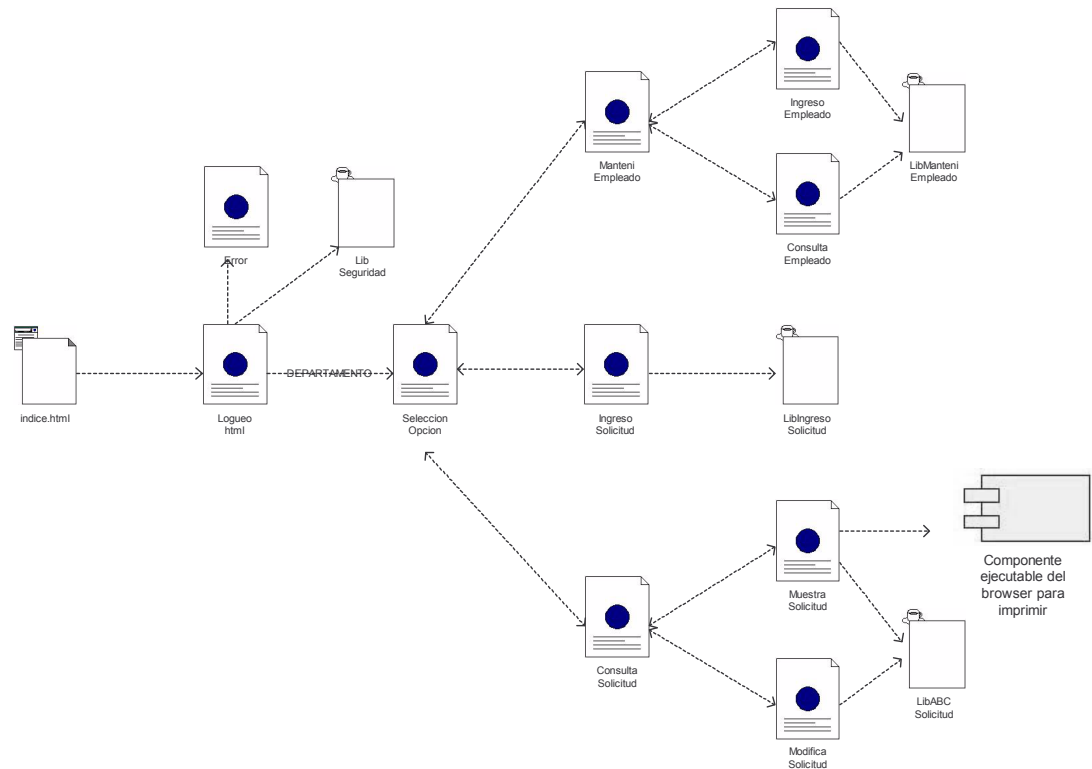


Figura 43. Diagrama de despliegue para usuarios de presupuesto de la empresa

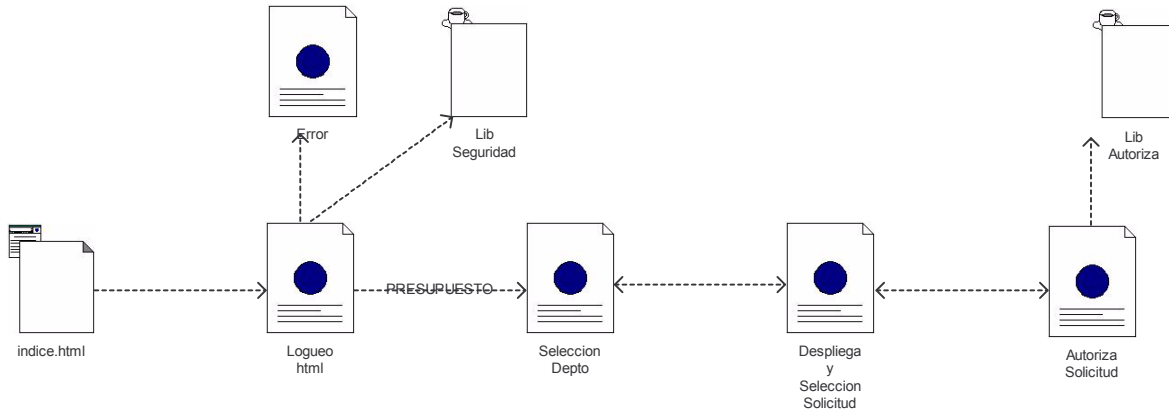
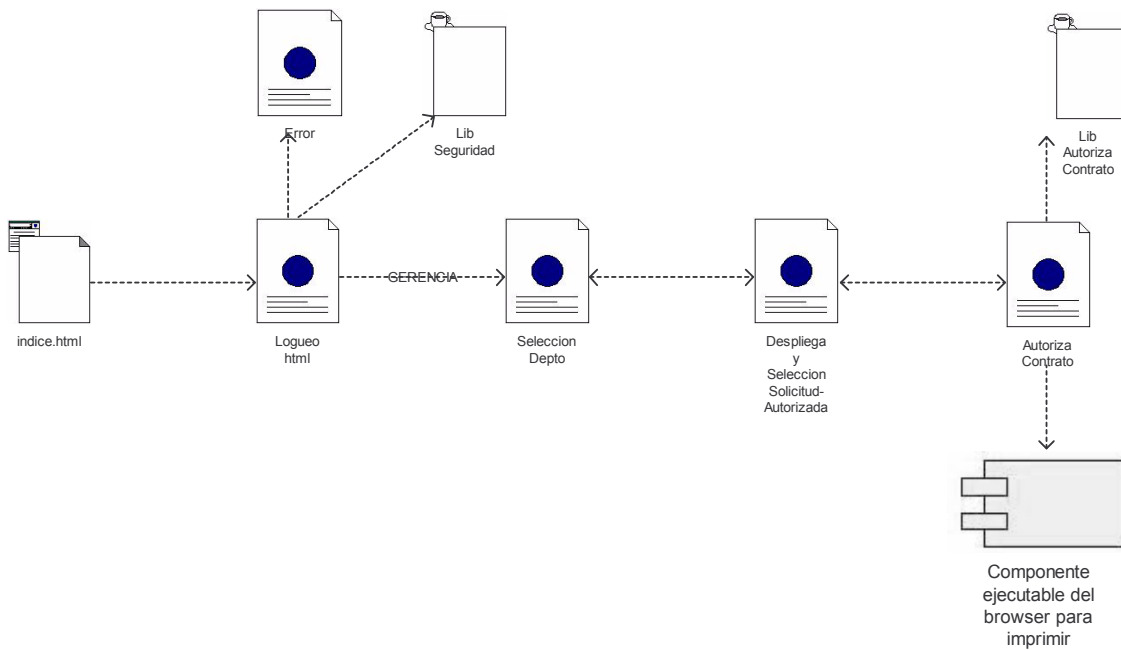


Figura 44. Diagrama de despliegue para usuarios de gerencia de la empresa



5.3. Prácticas adecuadas

Plasmar toda clase de detalles sugeridos en la conceptualización del sistema incluyendo la abstracción del anterior para que la migración no sea un fracaso.

Los esquemas conceptuales del sistema antiguo y del nuevo sistema son de mayor ayuda si antes de aplicar nomenclatura propia de clases y casos de uso son modelados como conceptos y relaciones en lenguaje natural.

Los procedimientos utilizados para el modelado de un proyecto de cambio de arquitectura son importantes para alcanzar el objetivo final, los cuales deben ser revisados constantemente, con el fin de asegurar que todos los usuarios involucrados perciban sus procesos operativos aplicados.

La comunicación debe ser abierta y directa ya que la expectativa del usuario crece con respecto a la información en línea, este requiere de resultados operativos más rápidos.

La experiencia en el diseño con UML y UWE es necesaria para no perder tiempo valioso en el aprendizaje de la misma, puesto que la investigación es necesaria solamente para extraer conocimiento del funcionamiento del sistema.

Recabar toda la información necesaria acerca de las distintas arquitecturas tanto origen como destino, puesto que detalles propios de éstas pueden influir en el comportamiento de los sistemas y eso es necesario para el modelado.

Contar con la colaboración de usuarios expertos que conocen el rol de los actores claves en los procesos funcionales.

Para el rediseño del sistema es recomendable crear un entorno de prueba lo más apegado a la realidad, para descubrir los detalles y casos particulares que muchas veces no se contemplan.

El procedimiento de migración de arquitectura de *software* no variará dependiendo de la dimensión del sistema, una vez se tenga clara la conceptualización de éste.

CONCLUSIONES

1. El análisis y diseño de un proceso de migración de arquitectura de software será más efectivo y conciso si se apoya en una metodología rica en detalles de conceptualización visual.
2. La aplicación de los estándares que sugieren las metodologías de modelado visual UML y UWE, ofrecen un control más detallado de especificaciones.
3. Por medio de las etapas que sugiere esta propuesta, un proceso de migración de arquitectura de software facilitará el desarrollo de la aplicación final.
4. La conceptualización y el diseño que permite UWE para representar los requerimientos del negocio para el funcionamiento en la *Web*, se acopla perfectamente con la lógica preexistente conceptualizada con UML.
5. La representación de las metodologías UML y UWE permiten validar juntamente con el usuario la conceptualización obtenida.

6. Esta propuesta garantiza el éxito del proyecto en función de la conceptualización, percepción y especificación que el analista plasme en el lenguaje de modelado propuesto.

RECOMENDACIONES

1. Trazar objetivos a largo y corto plazo, definiendo hitos y puntos de referencia, para hacer una migración de arquitectura de software de un sistema, para mejor control.
2. Para facilitar la conceptualización de una ingeniería inversa será de gran ayuda si se apoya en la metodología UML.
3. No caer en el error de considerar que por estar diseñando una aplicación orientada por un lenguaje de modelado visual, la interfaz gráfica al usuario de una aplicación es lo más importante, cada sección sugerida en esta propuesta tiene su importancia.
4. No confundir los detalles de las metodologías de modelado UML y UWE con lenguajes de programación lineal, pues estos tratan de modelar sistemas de manera que sean bastante entendibles al usuario sólo en etapa de análisis y diseño de un proyecto de aplicación de software.
5. Investigar los comportamientos particulares de las arquitecturas de *software* en la etapa de análisis, para tomarse en cuenta desde el diseño de la aplicación.

6. Seguir con los estándares que dictan las metodologías UML Y UWE para la documentación, en las fases posteriores de desarrollo.

REFERENCIAS BIBLIOGRÁFICAS

1. **Aplicación Web.**
<http://ieee.udistrital.edu.co/computer/paginas.php> (02-06-2003)
2. **Tecnologías de la información al servicio de la historia clínica electrónica.**
<http://www.seis.es/informes/2003/PDF/CAPITULO6.pdf> . (06-10-2004)
3. **Tecnología computacional**
<http://www.monografias.com/trabajos14/tecnolcomp/tecnolcomp2.shtml>
(02-02-09)
4. **Arquitectura en capas**
http://www.docirs.cl/arquitectura_tres_capas.htm (18-06-2003)

BIBLIOGRAFÍA

1. Aguirre Ordoñez, Zulma Karina. Modelado visual con el lenguaje unificado de modelado (UML). Tesis Ing. Sis. Guatemala, Universidad de San Carlos de Guatemala, Facultad de Ingeniería, 2000. 181pp.
2. Alvarado Mansilla, Luis Giovanni. Consideraciones para la migración de la tecnología de información actual hacia la nueva generación Downsizing/Rightsizing. Tesis Ing. Sis. Guatemala, universidad de San Carlos de Guatemala, Facultad de Ingeniería, 1997. 112 pp.
3. **Conallen, Inc. *HomePage*.**
<http://www.conallen.org/technologyCorner/webextension> .
(28-07-2004)
4. Conallen, Jim. ***Building Web Applications with UML***. Editorial Adison Wesley Longman, 1999.
5. **Construyendo aplicaciones web con una metodología de diseño orientada a objetos.**
http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r22_art5_c.pdf.
(20-07-2004)
6. **Desarrollo de aplicaciones WEB con UML.**
http://www.vico.org/TRAD_obert/TRAD_WAE_abierto.html .
(27-10-2003)

7. **Ingeniería inversa.**
http://www.pue.udlap.mx/~tesis/lis/lopez_a_aa/capitulo4.pdf.
(03-11-2004)
8. Larman, Craig. **UML y patrones. Introducción al análisis y diseño orientado a objetos.** México: Prentice Hall, 1999. 536pp.
9. **Metodologías para el desarrollo de aplicaciones Web: UWE**
http://alarcos.inf-cr.uclm.es/doc/aplicabdd/DASBD-MetodologiasParaElDesarrolloDeaplicacionesWeb_UWE.pdf .
(28-10-2005)
10. Migration Guide. ***A guide to migrating the basic software components on server and workstation computer.*** Berlin: KBSt Publication Service, 2003. 414pp.
11. **Modelando aspectos de navegación y presentación en aplicaciones hipermediales.**
<http://www.dlsi.ua.es/~ccachero/papers/jisbd00.pdf>. (30-11-2004)
12. Ortiz Ibáñez, Maria José y Jesús García Molina. **Un proceso basado en UML para aplicaciones Web.** Cursos de Verano de la Universidad de Burgos, Universidad de Murcia, España: 2001. 129pp.
13. Pressman, Roger S. Tr. Rafael Ojeda Martín, Joaquín Sánchez, Virgilio Galaup, Julio Zurdo Chávez. **Ingeniería del software.** 4º. ed. España: McGraw Hill – Interamericana. 581pp.

14. Schuller Joseph. **Aprendiendo UML en 24 horas**. México: Pearson Educación, 2000. 448pp.
15. Sommerville, Ian. **Ingeniería de Software**. 6ª ed. México: Pearson Educación, 2002. 712pp.
16. **Tecnologías de la información al servicio de la historia clínica electrónica**.
<http://www.seis.es/informes/2003/PDF/CAPITULO6.pdf> .
(06-10-2004)
17. **Terminología general**
es.wikipedia.org . (30/03/2010)
18. **UML**. <http://usuarios.lycos.es/oopere/uml.htm> .(19-04-2004)
19. **Web Software Arquitectura**.
<http://www.dlsi.ua.es/~ccachero/papers/websatr.pdf> . (28-07-2004)