ESCUELA POLITÉCNICA DEL EJÉRCITO

FACULTAD DE SISTEMAS E INFORMÁTICA

SISTEMA PARA MANEJO DE SOPORTE DE MESA (HELPDESK) PARA EXSERSA

"HDS_SYSTEM"

Previa a la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMÁTICA

POR: TORRES CAMPAÑA ALEX JAVIER

SANGOLQUÍ, Junio 07 de 2005

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. TORRES CAMPAÑA ALEX JAVIER como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA.

_____

Junio 07 de 2005

_____

ING. PATRICIA VELASTEGUI

DEDICATORIA

Dedico el presente trabajo a mi madre, a mi esposa y mis hijos símbolos de Amor, esfuerzo y constancia.

Alex J. Torres C.

AGRADECIMIENTOS

Primero doy gracias a mi mejor amigo (**Jesucristo**) que siempre estuvo a mi lado en los buenos y más aun en los momentos difíciles de mi vida, convirtiéndose en mi fortaleza y mi escudo. Gracias Señor.

A mi **Madre** Olga a mi **Padre** Carlos, así como también a mis **hermanos** quienes inculcaron en mí valores de amor, respeto, y responsabilidad, depositaron su confianza y juntos decidimos embarcarnos en este proyecto que es la Instrucción Superior, gracias a su apoyo y esfuerzo hoy puedo decirles que nuestro proyecto se ha cumplido.

A mi esposa **Doris**, mis dos tesoros **Erick** y **Josselyn** (mis hijos), quienes también contribuyeron con su amor incondicional y se convirtieron a lo largo de estos años en la razón fundamental para culminar la carrera y ejercerla con profesionalismo, teniendo siempre presente que en ellos se reflejaran nuestros valores.

A la Escuela Politécnica del Ejército **ESPE** y todo su personal, en especial al cuerpo docente quienes aportaron con sus conocimientos y su paciencia formando profesionales de alto nivel académico.

Alex J. Torres C.

RESUMEN

Uno de los puntos más sensibles en EXSERSA y cualquier organización que brinda servicios es la calidad con el que este se da a sus usuarios, siendo la atención a los mismos (tanto internos como externos) lo que se busca cubrir con mayor rapidez y eficiencia.

El proyecto contempla el estudio de procesos manuales así como el planteamiento de nuevos procesos, el mejoramiento y automatización de los mismos, el manejo de relaciones interpersonales con otras áreas ajenas ha la de IT, da la pauta para formar de manera mas profesional a quien lo realiza.

El crecimiento que la empresa ha experimentado crea la necesidad de desarrollar un sistema automatizado que permita manejar con mayor control, agilidad y seguridad los incidentes reportados por los clientes internos (Agencias y personal de Matriz) y externos (Clientes financieros y no financieros), su registro, asignación a personal técnico y solución.

HDS_System permite al área de Helpdesk administrar la información de incidentes (Problemas, reportes, solicitudes, etc.) de una manera organizada, facilita el enrutamiento o asignación de los problemas vía correo electrónico a un técnico específico, llevando así un control de tiempos de solución de los mismos. El sistema da la facilidad también para almacenar problemas y soluciones alternativas, esto agilitará la solución de incidentes y además permite que usuarios no experimentados puedan utilizar la herramienta para dar posibles soluciones a incidentes que se presenten.

CAPITULO I

EXSERSA (SERVIPAGOS)

EXSERSA es una empresa de servicios transaccionales, se conformó el 27 de Abril de 1998 gracias a la iniciativa de Ejecutivos de PRODUBANCO y el entonces BANCO POPULAR.

La empresa provee soluciones de cobros y pagos de calidad, a través de agencias, call center, internet y demás canales de distribución, brindando a los consumidores un servicio rápido y accesible.

A finales de abril de 1998, EXSERSA arranca sus operaciones con la agencia Colón y Santo Domingo en la ciudad de Quito, Escobedo y Alborada en la ciudad de Guayaquil, desde esa fecha, con la colaboración de todo los miembros de la empresa, esta ha logrando posicionarse en el mercado como una empresa diferente, que ofrece valor agregado a sus clientes como los horarios de atención, rapidez transaccional y actitud en el servicio.

Durante la crisis de los años 1999 y 2000 muchas empresas quebraron, siendo EXSERSA una de las que lograron sobresalir y fortalecerse para enfrentar un futuro que la empresa en conjunto con sus empleados lo ven con mucho optimismo.

Las labores de EXSERSA están orientadas a satisfacer las necesidades básicamente de tres segmentos:

a. Financiero: Realiza transacciones de cajas como pago de cheques, retiro de libretas de ahorro, depósitos, certificación de cheques,

cobranzas corporativas, rol de pagos, etc. Actualmente presta servicios a Produbanco, Citibank, Unibanco, Banco Amazonas, Mutualista Pichincha, Banco de los Andes, Banco Solidario y Banco Ecuatoriano de la Vivienda, por lo tanto los usuarios de estos bancos pueden hacer las transacciones bancarias en los canales de EXSERSA.

b. Público: El trabajo está dirigido al cobro de servicios públicos, la recaudación de impuestos, tasas, dentro de los principales clientes están los siguientes: Ilustre Municipio de Quito, Andinatel, Pacifictel, Empresa Muncipal de Alcantarillado y Agua Potable (EMAAP), Emelgur, Empresa Eléctrica de Quito, Emelec, Interagua, Instituto Ecuatoriano de Seguridad Policial (IESS), entre otros.
c. Privado: Brindando servicios de cobros, membresías, cuotas, pago de nómina, reembolsos de seguros, pago de pensiones, etc. Entre los clientes se destacan TV Cable, Compañía de Cervezas Nacionales, Instituto de Seguridad Social de la Policía (ISSPOL), Junta de Beneficencia de Guayaquil, Avon, Coneca, Conservera Guayas, Satnet, Beepercom, Castillo de Amaguaña, entre otros.

Cuenta con una de Red Transaccional con 30 agencias en la ciudad de Quito, 14 en la ciudad de Guayaquil y con sucursales en las ciudades de Santo Domingo, Ibarra, Cayambe, Loja, Esmeraldas, Manta, Cuenca, Quevedo, Riobamba, Ambato, Machala, Cayambe cubriendo de esta manera gran parte del país (figura 1.1).

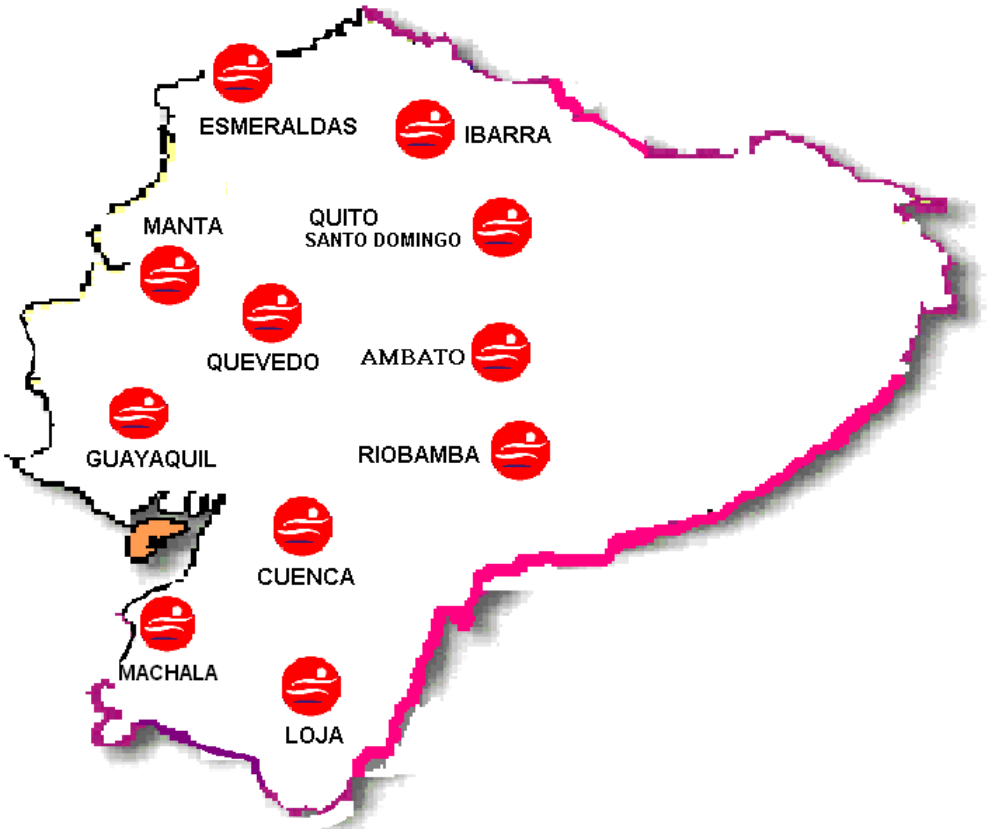

Figura 1.1: (Distribución Geográfica de Agencias a nivel País)

La red transaccional de Servipagos se la puede representar en el siguiente esquemas que refleja la configuración Wan de la empresa (figura 1.2), así como también el esquemas básico de interconexión de con varios de sus clientes (figura 1.3).

DIAGRAMA WAN EXSERSA

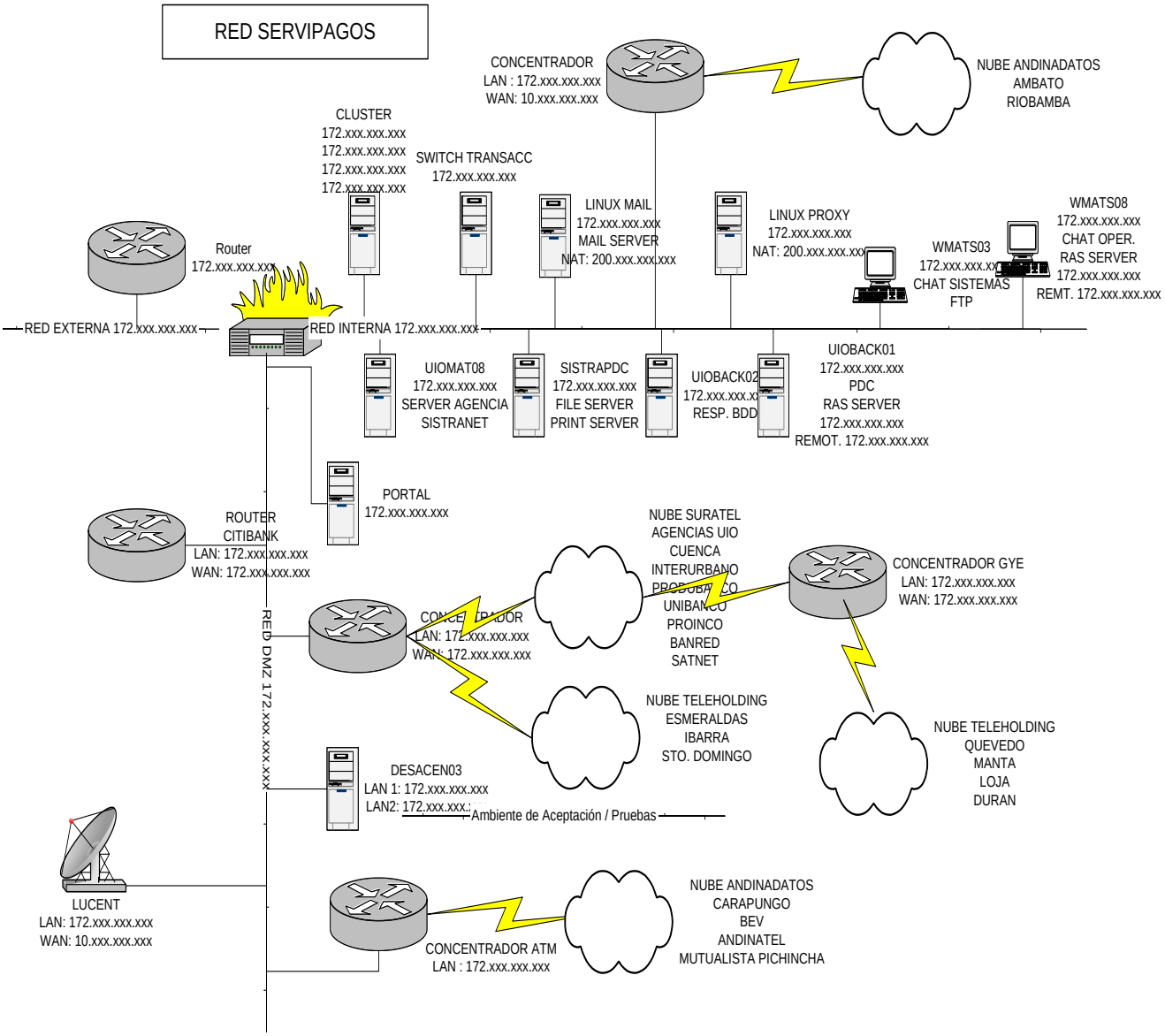

Figura 1.2: (Diagrama Red Wan EXSERSA)

CONFIGURACIÓN DE LAS CONEXIONES EXISTENTES CON CLIENTES

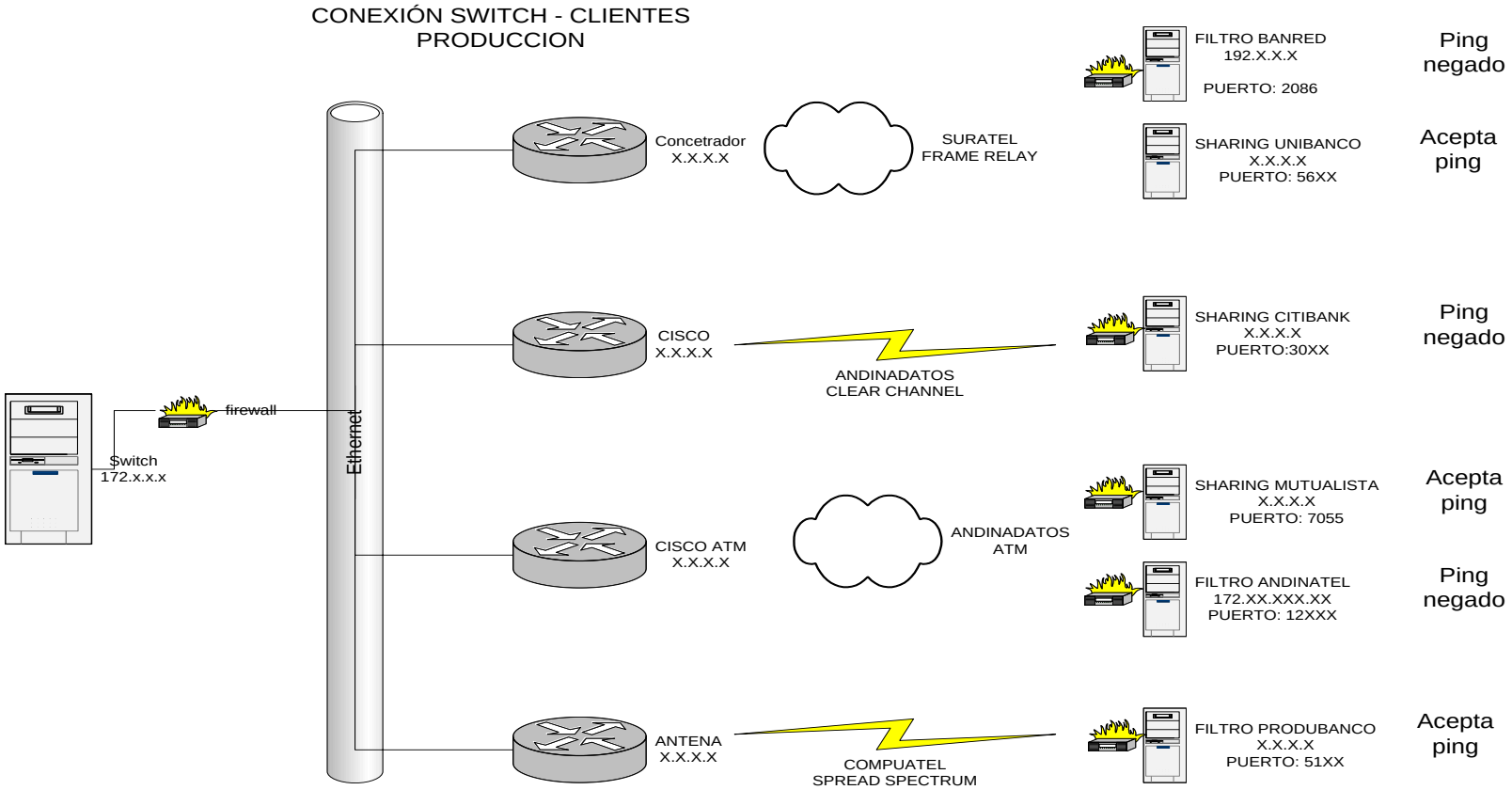

Figura 1.3: (Esquema de Conexión de Clientes Financieros)

1.2.- Misión – Visión de EXSERSA

Las acciones permanentes que constituyen la razón de ser de EXSERSA, es decir el servicio a los clientes a través de la gestión de cobros y pagos realizados con los márgenes de suficiencia y calidad, están considerados en la misión de la empresa.

El objetivo macro, el horizonte de planificación está considerado en la descripción de la visión de la empresa.

1.3.- Misión

"Proveer soluciones de cobros y pagos, de gran cobertura, a través de agencias, teléfono, Internet y demás canales de distribución, agrupando clientes para disminuir sus costos; brindando a los consumidores un servicio rápido y accesible, calidad de vida en el trabajo a sus colaboradores y rentabilidad a los accionistas; en un marco de colaboración con la sociedad".

1.4.- Visión

"Ser el mejor y más grande canal de distribución de servicios y soluciones de cobros y pagos del Ecuador con proyección internacional".

1.5.- Estructura organizacional

1.5.1.- Estructura Conceptual EXSERSA

La estructura se la resume a continuación (figura 1.4).

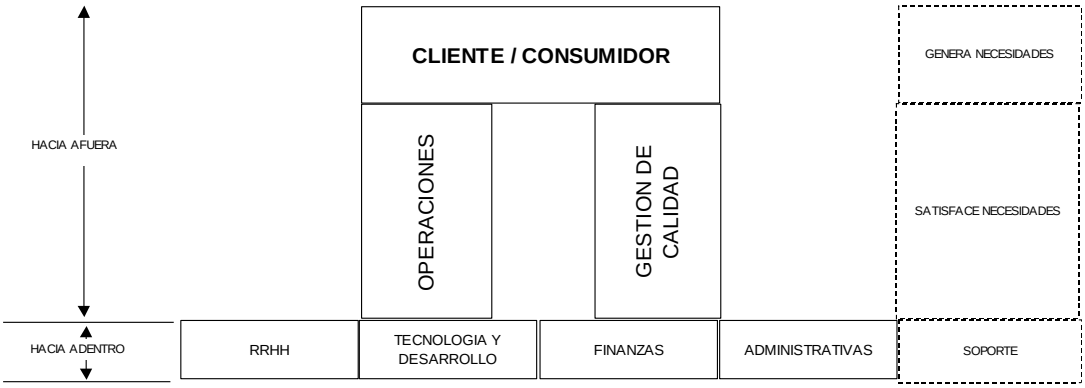

Figura 1.4: (Estructura conceptual EXSERSA)

1.5.2.- Organigrama EXSERSA

La estructura organizacional de de la empresa se la vera representada en la figura 1.5

1.5.2.1.- Gerente general

Reporta a: Presidente Directorio.

Lidera a: Gerente Administrativo, Gerente Comercial, Gerente de Operaciones, Gerente de Tecnología, Gerente Regional, Jefe de Sucursales, Auditor Interno.

Misión del Cargo: Ejercer la representación legal de las operaciones de EXSERSA; planificar, organizar, integrar y controlar todas las actividades de la compañía, dentro de las políticas y objetivos generales aprobados por el Directorio.

ESTRUCTURA ORGANIZACIONAL EXSERSA

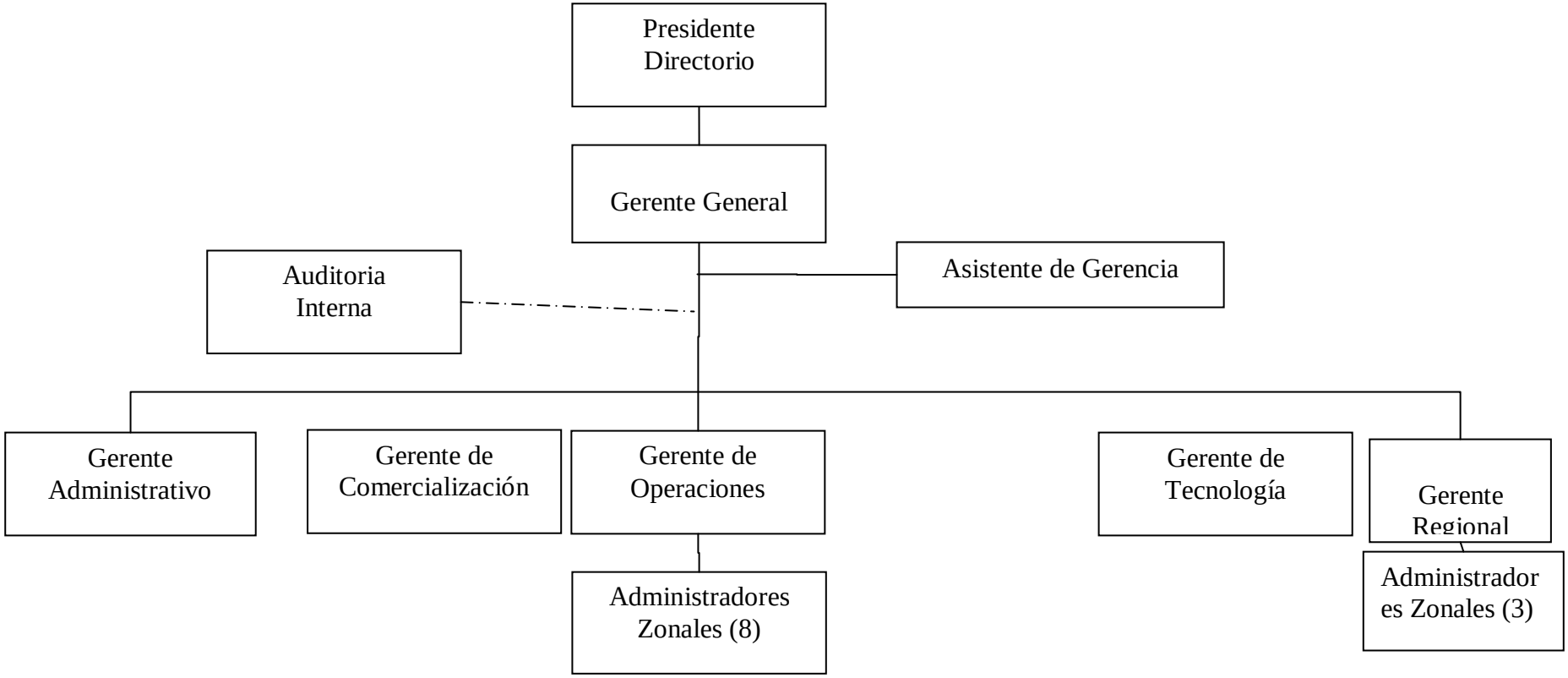

Figura 1.5: (Distribución Orgánica de EXSERSA)

1.5.2.2.- Asistente de Gerencia

Reporta a: Gerente General

Lidera a: No aplica

Misión del cargo: Llevar los reportes y cumplimiento de objetivos y planes de cada área para informar a la Gerencia.

Tareas típicas:

Elaborar presentaciones para el Directorio (Produbanco).

Recibir y realizar llamadas telefónicas.

Elaborar actas de reuniones, cartas, informes para Gerencia

1.5.2.3.- Auditoria Interna

Reporta a: Gerente General

Lidera a: Asistentes de Auditoria, Coordinador de Cambios, Administrador de Seguridades

Misión del cargo: Proponer mejoras para las áreas, mediante la evolución de procesos

Tareas típicas:

- Velar y supervisar que el proceso de Control de Cambios y Seguridades se cumpla.
- Efectuar visitas y revisiones de agencias
- Realizar auditorias a diferentes áreas
- Cumplir con objetivos institucionales de la Empresa
- Participar en los Comités de la Empresa

- Minimizar los riesgos de la Empresa
- Elaborar informes para la Gerencia General
- Administrar y verificar que se cumplan los objetivos de su área

1.5.2.4.- Gerente Administrativo

Reporta a: Gerente General

Lidera a: Jefe de Servicios Administrativos, Jefe de Cartera, Jefe de Recursos Humanos, Subcontador, Coordinadora de Compras, Asistentes

Misión del cargo: Administrar los recursos materiales, humanos y de capital para optimizar su uso, buscando el cumplimiento de los objetivos de la Institución.

Tareas típicas:

- Revisar estados financieros
- Controlar el cumplimiento de las obligaciones tributarias, de funcionamiento, legales y de prestación Social de la empresa
- Manejar inversiones
- Elaborar y controlar el presupuesto mensual de la empresa
- Mantener contacto con los bancos locales
- Negociar las Pólizas de Seguro de activos fijos, dinero y personal.
- Elaborar el plan anual de compras y mantenimiento de agencias
- Estudiar permanentemente la estructura salarial de la institución y sugerir acciones para mantener un sistema adecuado sobre la base de competitividad externa y equidad interna.
- Aprobar la nómina de pagos

- Instruir al personal sobre la técnica o el modo de aplicar los Subsistemas en Recursos Humanos y sobre Legislación Laboral.
- Analizar la estructura organizacional de la empresa.

1.5.2.5.- Gerente de Comercialización

Reporta a: Gerente General

Lidera a: Jefe de Marketing, Coordinador Publicidad, Asistente

Misión del cargo: Comercializar y distribuir las transacciones de la Empresa, mediante todos los canales existentes, para satisfacer los requerimientos de los clientes, manteniendo una buena imagen basada en estudios de mercado.

Tareas típicas:

- Definir las estrategias y objetivos del área
- Representar a la empresa frente a clientes.
- Comercializar y crear nuevos productos y canales
- Controlar el presupuesto comercial
- Mantener la imagen de la empresa frente a clientes
- Apoyar a los objetivos institucionales
- Administrar el personal a su cargo
- Mantener reuniones periódicas con clientes

1.5.2.6.- Gerente de Operaciones

Reporta a: Gerente General

Lidera a: Jefes de Operaciones, Administradores Zonales, Asistentes Operativos, Supervisores, Cajeros.

Misión del cargo: Administrar y mejorar los procesos de operación de cobros y pagos a nivel nacional

Tareas típicas:

- Cumplir y hacer cumplir los procedimientos operativos
- Ejecutar el plan estratégico en lo referente a sus responsabilidades de operación
- Administrar personal a su cargo
- Manejar las relaciones con clientes y proveedores con respecto a la relación operativa
- Liderar el mejoramiento continuo de los procesos
- Planificar, delegar, ejecutar y revisar las tareas operativas
- Elaborar informes Gerenciales
- Despachar los requerimientos de las agencias
- Realizar reuniones con Supervisores y Cajeros
- Realizar reunión de avance de objetivos

1.5.2.7.- Gerente de Tecnología

Reporta a: Gerente General

Lidera a: Jefe de Desarrollo y Jefe de Producción

Misión del cargo: Administrar los recursos Tecnológicos de la Institución.

Tareas típicas:

- Participar en la Planificación Estratégica de la empresa

- Establecer los lineamientos y estándares tecnológicos de la empresa
- Administrar eficientemente los costos de inversión de tecnología, mediante presupuestos y un adecuado control de gastos
- Establecer planes de evaluación, implantación y operación de soluciones tecnológicas requeridas por la empresa
- Liderar proyectos de investigación, análisis, selección y puesta en marcha de nuevas tecnologías
- Asesorar a las unidades de la empresa en la adopción de nuevas tecnologías y en la solución de problemas
- Coordinar con las tareas de planificación e implementación de proyectos tecnológicos
- Coordinar el desarrollo de las aplicaciones de la empresa
- Establecer con las diferentes áreas las necesidades y especificaciones funcionales de las aplicaciones
- Establecer políticas y procedimientos que normen las actividades relacionadas con el área de la Tecnología
- Administrar los recursos físicos y humanos que forman parte del área de Tecnología
- Coordinad con Recursos Humanos los planes de capacitación y entrenamiento tecnológico al interior de la empresa
- Apoyar a que los sistemas de producción se encuentren trabajando dentro de estándares y parámetros requeridos

- Realizar planes de capacidad y actualización de la infraestructura tecnológica de la empresa.

CAPITULO II

MARCO TEÓRICO

2.1.- Helpdesk

Qué es una mesa de soporte o Helpdesk?

Un Helpdesk (HD) se lo define como una organización formal que realiza y provee funciones de soporte a los usuarios de los productos, servicios o tecnología que genera la empresa o alguna área específica de ella.

El soporte puede ser para alguna persona una función de tiempo parcial o de tiempo completo dentro de sus responsabilidades de trabajo; por ejemplo un programador, además de codificar, puede tomar llamadas telefónicas para resolver dudas con respecto a los programas que está desarrollando. Sin embargo, un HD representa un esfuerzo bien organizado y coordinado con un propósito específico.

Un área de HD puede estar formada de una sola persona que desempeñe múltiples funciones de apoyo, o por cientos o miles de personas dando soporte a las diversas funciones de un negocio.

El soporte puede ser reactivo, como en el caso en que alguien se sienta detrás de un teléfono, o puede ser proactivo, buscando formas de hacer más productivos a los usuarios. El soporte puede servir para:

a. Corregir o arreglar alguna situación.- Tal es el caso de la atención a peticiones como "mi teclado no funciona" o "se atoró el papel de la impresora".

b. Asesorar o entrenar.- Por ejemplo cuando se apoya a algún usuario a manejar correctamente un software o un dispositivo de hardware.

c. Administrar un ambiente operativo. En este caso se puede citar al personal de soporte que también está encargado de mantener funcionando en forma continua la red de la empresa atendiendo cualquier falla, incidente o realizando los mantenimientos y respaldos de rutina.

Los usuarios o clientes son quienes llaman o requieren de los servicios que provee el HD, donde los productos, servicios o la tecnología son el punto central del requerimiento. Puede tratarse de productos fabricados por la compañía, una impresora, computadora, teléfono, software o un mueble armable.

El área de HD no es tan conocida como las áreas de Desarrollo de Aplicaciones, Administración de Base de Datos o Administración de Redes. Realmente hasta este momento no es posible obtener una licenciatura en alguna carrera de HD, sin embargo para cubrir este vacío están surgiendo empresas especializadas y organismos profesionales que ofrecen programas de certificación dentro de esta disciplina. Tal es el caso de Total Service Desk (TSD).

Quienes atienden un HD pueden ser llamados Agentes, Ingenieros de Soporte, Empleados, Asociados, Ejecutivos, etc. Aunque todos estos nombres pueden ser usados indistintamente en diversas organizaciones, la única diferencia

real estriba en los conocimientos que requieren para dar el soporte en su organización de HD, más que las funciones que desempeñan y la forma de realizarlas, que en realidad pueden estandarizarse y tomar base en las mejores prácticas de operación dentro de esta disciplina.

Quiénes utilizan los servicios de un área de HD pueden ser llamados en general Usuarios o Clientes. Existen otros nombres que, dependiendo del ramo de la empresa, pueden llegar a tener los clientes en relación con el negocio, por ejemplo asegurado, si se trata de una compañía de seguros, paciente si es el caso de un servicio médico o agencia si es una empresa de servicios transaccionales.

Finalmente, al evento en el que un cliente hace contacto con un agente del HD para solicitar servicio se le puede nombrar de diversas formas como llamada, petición de servicio, incidente, problema, reporte, ticket, etc. No obstante que estos términos pudieran parecer sinónimos es muy conveniente llegar a un acuerdo semántico con respecto a su definición dentro de la disciplina del HD.

a. Llamada o petición de servicio.- Es cualquier contacto que tenga el cliente con el HD. El término no es limitativo a una conversación telefónica atendida por un Ingeniero de Soporte; puede referirse a cualquier tipo de contacto que esté habilitado incluyendo mensajes de correo electrónico, faxes, chat's, correos de voz, portales web, atención mediante sistemas IVR[1], etc.

[1] IVR **Sistemas** de respuesta interactiva de voz

b. Incidente.- Este término se refiere al suceso que motivó la llamada del cliente hacia el HD. Generalmente es un síntoma originado por un problema. Es una situación que requiere de la ayuda o del soporte por parte del agente del HD.
c. Problema.- Esta palabra se utiliza para nombrar la raíz o el origen que provocó el incidente. Es una causa cuyos efectos normalmente son los incidentes.
d. Reporte o ticket. Con estos términos se refiere al registro de la incidencia dentro del HD. Normalmente este registro queda identificado por un número o folio que permite darle seguimiento hasta la solución del problema expresado por el cliente.

Dónde se puede aplicar un Helpdesk?

Típicamente y de manera natural, un HD servirá de interfase entre usuarios o clientes y el área de Tecnologías de la Información (TI), pero las aplicaciones de un HD pueden extenderse a otros servicios como administración y seguimiento de garantías, servicios de mantenimiento y reparación de cualquier tipo de máquina o vehículo, servicios de información sobre el uso y características de los productos de una compañía, etc.

Dentro de las múltiples facetas que puede presentar un HD pueden enlistarse las siguientes:

- Soporte a clientes internos.
- Soporte a clientes externos.
- Administración de inventarios de TI.

- Administración del conocimiento.
- Control de servicios de mantenimiento.
- Control de actualizaciones, migraciones e implantaciones.
- Control de garantías.
- Control de préstamos y soportes de equipo y software.
- Control de instalaciones.
- Control de errores y mejoras a software (bug tracking).
- Otras.

2.2.- Conceptos Y Principios De Orientación A Objetos

Cuando se va a construir un sistema software es necesario conocer un lenguaje de programación, pero con eso no basta. Si se quiere que el sistema sea robusto y mantenible es necesario que el problema sea analizado y la solución sea cuidadosamente diseñada. Se debe seguir un proceso confiable, que incluya las actividades principales. Si se sigue un proceso de desarrollo que se ocupa de plantear cómo se realiza el análisis y el diseño, y cómo se relacionan los productos de ambos, entonces la construcción de sistemas de software va a poder ser planificable y repetible, y la probabilidad de obtener un sistema de mejor calidad al final del proceso aumenta considerablemente, especialmente cuando se trata de un equipo de desarrollo formado por varias personas.

El proceso a seguir para realizar desarrollo orientado a objetos es complejo, debido a la complicación que se pueda encontrar al intentar desarrollar cualquier sistema de software de tamaño medio-alto. El proceso está formado por

una serie de actividades y subactividades, cuya realización se va repitiendo en el tiempo, aplicadas a distintos elementos.

2.2.1.- Conceptos Básicos De Orientación A Objetos

Pressman [PRE97], citando a Coad y Yourdon, dos de los más importantes pensadores de la tecnología de objetos definen la orientación a objetos como:

ORIENTACIÓN A OBJETOS = Objetos + Clasificación + Herencia + Comunicación

De donde:

- **Objeto:** Instancia o representación de “algo” que tiene ciertos “atributos” verificables, sobre los cuales se pueden realizar ciertas “operaciones” y que posiblemente tenga relación con cierta “clasificación” que lo agrupe, de la cual pueda heredar tanto cualidades como comportamiento: Silla (objeto) es parte de Mobiliario (clase), Secretaria (objeto) es parte de Empleado (clase).
- **Clasificación:** Agrupamiento que encapsula las abstracciones de datos y procedimientos que se requieren para describir el contenido y comportamiento de alguna entidad del mundo real. Técnicamente, en el mundo OO, recibe el nombre de Clase.
- **Herencia:** Capacidad de un objeto de heredar atributos y operaciones de otro por lo general denominado Clase. Este concepto lleva a definir:
  - **Superclase:** Colección de clases.
  - **Subclase:** Instancia de una clase.
- **Comunicación:** Mecanismo que permite contactarse entre objetos para llevar a cabo las operaciones disponibles (Mensaje).

Como se visualiza, tanto objetos como clases, están compuestos de atributos y operaciones, los mismos que se relacionan mediante mensajes, entonces:

- **Atributo:** Característica binaria que implica que se pueda tomar un valor definido de un dominio enumerado
- **Operaciones:** Llamadas también métodos o servicios, son los algoritmos que procesan los atributos encapsulados proporcionando una representación del comportamiento de los objetos
- **Mensajes:** Medio mediante el cual los objetos interactúan. Un mensaje estimula la ocurrencia de cierto comportamiento en el objeto receptor, mismo que se realiza cuando se ejecuta una operación.

Además, para hablar de objetos de forma íntegra, se debe considerar las siguientes características:

- **Localización:** Forma en que se concentra la información dentro de un programa. En el caso OO, la clase.
- **Encapsulamiento:** Empaquetamiento de una colección de elementos (atributos y operaciones), facilitando que las estructuras de datos y las operaciones que las manipulan estén mezcladas en una entidad sencilla: la clase. Además las interfaces entre objetos encapsulados se simplifican.
- **Ocultación de información:** Supresión de detalles operativos de un componente de un programa. Esto ayuda a reducir la propagación de defectos.

- **Herencia:** La jerarquía de clases se convierte en un mecanismo a través del cual los cambios pueden propagarse inmediatamente a través de todo el sistema.
- **Abstracción:** Mecanismo que permite al diseñador de un programa centrarse en los detalles esenciales de algún componente de un programa sin preocuparse de los detalles del nivel inferior.

Como se ve, el concepto de herencia es sumamente importante, puesto que implicaría un ahorro enorme de codificación, pero es importante considerar que:

- Es probable heredar tanto atributos como operaciones de forma normal y después notificarlos según las necesidades específicas de cada clase, en cuyo caso se dice que la herencia no es transitiva. Esto se denomina ***Anulación***.
- También se puede heredar atributos y operaciones de varias clases (***Herencia Múltiple***), pero no es recomendable.
- Se puede generalizar varias clases u objetos mediante ***sobrecarga***, donde cada clase define operaciones similares que pueden ser llamadas mediante el objeto general. Esto se denomina ***polimorfismo.***
- En general, antes que determinar herencia (***generalización***), se debe primero determinar si existen relaciones parte – todo entre objetos (***agregación***)

2.2.2.- ¿Qué es el Análisis y Diseño Orientado a Objetos?

Los conceptos en forma básica y simple se pueden describir de la siguiente manera:

- **Análisis:** Es una fase de desarrollo de sistemas en la cual se investiga el problema a resolver, pero no la forma de definir la solución.
- **Diseño:** Es la fase de desarrollo de sistemas en la cual se propone una solución lógica al problema.

Entonces, el análisis y diseño orientado a objetos no son otra cosa que situar el dominio del problema y su solución lógica dentro de la perspectiva de los objetos. Esto no quiere decir, que la solución dada será necesariamente la mejor, sino que también podría ser resuelto de otra manera.

Para clarificar mejor esto, se toma como punto de partida el ejemplo básico de un Sistema de información de una biblioteca propuesto por [Larman, 1999].

Se conoce de antemano que los proyectos de sistemas son complejos, razón por la cual, se suele descomponer los mismos en bloques más fácilmente manejables. Antes del paradigma de objetos, lo usual era usar las técnicas de análisis y diseño estructuradas, que descomponen el problema básicamente por función o proceso, lo cual origina una división jerárquica de procesos constituidos por subprocesos. El análisis y el diseño orientado a objetos buscan descomponer el espacio del problema por objetos, lo que implica que posiblemente no se pueda hablar de jerarquías. La figura 2.1, describe los resultados de la aplicación de los 2 enfoques para el ejemplo indicado.

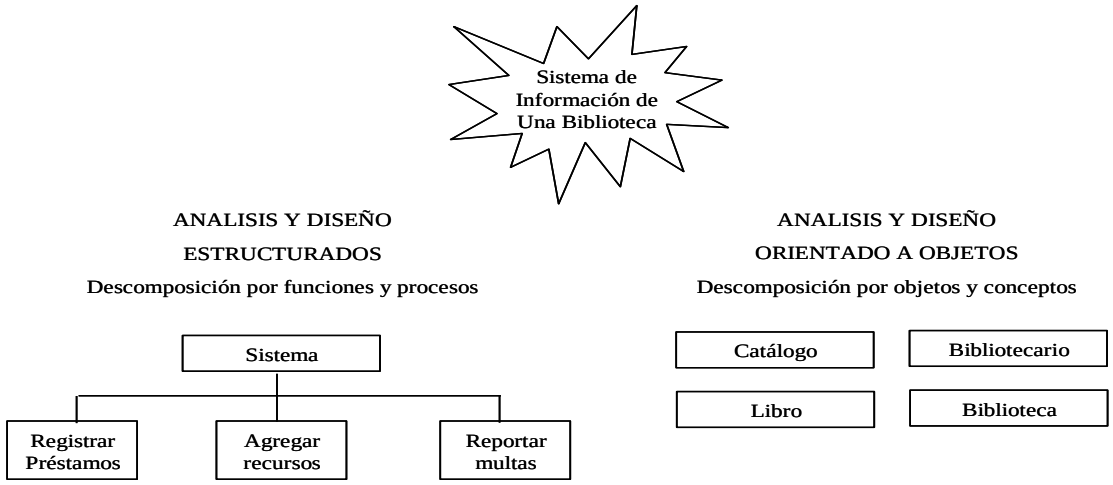

Figura 2.1: (Sistema Bibliotecario, el enfoque estructurado versus el enfoque orientado a objetos.[Larman, 1999 Figura 1.8])

Entonces, se podría decir que:

- El **análisis orientado a objetos**, busca identificar y describir los objetos o conceptos dentro del dominio del problema
- El **diseño orientado a objetos**, busca definir los objetos lógicos del software que finalmente serán implementados usando un lenguaje de programación orientado a objetos.

2.2.3.- Proceso Básico De Desarrollo Orientado A Objetos

En vista de que el enfoque es diferente, no todas las metodologías de desarrollo de software propuestas se adaptan de forma eficiente al mismo, sobretodo considerando que los sistemas orientados a objetos tienden a evolucionar con el tiempo y es sumamente difícil definir todas las clases necesarias para un gran sistema o producto al inicio del proceso.

Por este motivo, todos los autores coinciden con que el modelo evolutivo de desarrollo de componentes, basado en el modelo espiral de Boehm, es el modelo que mejor se adapta al paradigma orientado a objetos (y en general es el

tipo de proceso más eficiente y real que se puede aplicar) La figura 2.2, detalla de forma simple este modelo.

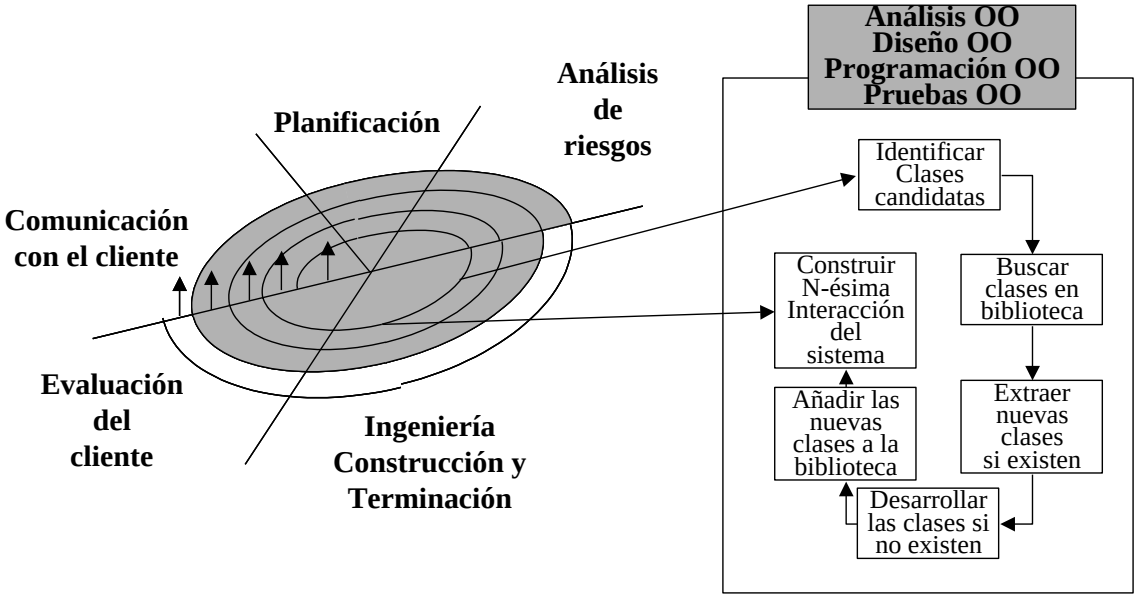

Figura 2.2: (Modelo de proceso orientado a objetos [Pressman, 1997 Figura 19.1])

2.2.4.- Metodologías De Desarrollo Y Modelado Orientado A Objetos: Proceso Unificado Y UML

Hasta antes de 1997, las principales metodologías de desarrollo orientadas a objetos eran:

- El método de Boch
- El método de Coad y Yourdon
- El método de Jacobson (ISOO)
- El método de Rumbaugh (OMT)
- El método de Wirfs – Brock

Pero ese año se presento el Lenguaje Unificado para la Construcción de Modelos, UML (Unified Modeling Language), que no era otra cosa que la

unificación de los elementos y artefactos de modelado de Booch, Rumbaugh (primero) y Jacobson (después), el cual se convirtió en el estándar de facto de la industria, promovido por la organización de investigación y estándares de objetos OMG (Object Management Group). Actualmente, este estándar de modelado es usado incluso por la mayoría de herramientas CASE del mercado.

Sin embargo, es importante indicar que “el UML es un lenguaje para construir modelos; no guía al desarrollador en la forma de realizar el análisis y diseño orientado a objetos ni le indica cual proceso de desarrollo a adoptar” [Larman, 1999] Es decir, saber UML no implica necesariamente saber analizar y / o diseñar sistemas orientados a objetos.

2.3.- Lenguaje de Modelamiento Unificado UML

El lenguaje de modelamiento unificado (UML Unified Modeling Language) tiene sus inicios a finales de los 80’s y principios de los 90’s, consiste en la sucesión de los métodos de análisis y diseños orientados a objetos, básicamente unifica los métodos de Jim Rumbaugh (OMT), Grady Booch, Ivar Jacobson y otros.

Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la cual basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran prestigio en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

UML es llamado un lenguaje de modelamiento, no un método. Los métodos consisten de los dos, tanto de un lenguaje de modelamiento y de un proceso.

El lenguaje de modelamiento es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño.

Una de las metas principales de UML es avanzar en el estado de la industria proporcionando herramientas de interoperabilidad para el modelamiento visual de objetos. Sin embargo para lograr un intercambio exitoso de modelos de información entre herramientas, se requirió definirle una semántica y una notación.

La notación es la parte gráfica que se ve en los modelos y representa la sintaxis del lenguaje de modelamiento.

Al cubrir con la semántica y con la notación se cumple con los estándares de UML.

Una herramienta de UML debe mantener la consistencia entre los diagramas en un mismo modelo. Bajo esta definición una herramienta que solo dibuje, no puede cumplir con la notación de UML.

2.3.1.- Qué es UML?

Es un lenguaje de modelamiento visual usado para especificar, visualizar, construir y documentar a cada una de las partes que comprende el desarrollo de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

Capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

Pretende unificar la experiencia pasada sobre técnicas de modelamiento e incorporar las mejores prácticas actuales en un acercamiento estándar.

No es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

Existían diversos métodos y técnicas Orientadas a Objetos (OO), con muchos aspectos en común pero utilizando distintas notaciones, se presentaban inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc., además de pugnas entre enfoques, lo que generó la creación del UML como estándar para el modelamiento de sistemas de software principalmente, pero con posibilidades de ser aplicado a todo tipo de proyectos.

2.3.2.- Objetivos del UML

- Ser un lenguaje de propósito general, estandarizado en lo posible para toda la comunidad informática.
- Ser un lenguaje simple y fácil de entender brindando facilidades de de modelar cualquier sistema.
- Manejar todos los conceptos que se originan en un sistema moderno (concurrencia, distribución, etc.), igual manera los mecanismo de ingeniería de software (encapsulación y componentes).
- Imponer un estándar mundial.

2.3.3.- Arquitectura del UML

Con una arquitectura de cuatro capas, definida a fin de cumplir con la especificación Meta CObject Facility del OMG:

a. Meta-metamodelo.- define el lenguaje para especificar metamodelos.
b. Metamodelo.- define el lenguaje para especificar modelos.
c. Modelo.- define el lenguaje para describir un dominio de información.
d. Objetos de usuario.- define un dominio de información específico.

2.3.4.- Áreas conceptuales de UML

Los conceptos y modelos de UML pueden agruparse en las siguientes áreas conceptuales:

Estructura estática:

Cualquier modelo preciso debe primero definir su universo, se refiere a los conceptos clave de la aplicación, sus propiedades internas, y las relaciones entre cada una de ellas. Este conjunto de construcciones es la estructura estática. Los conceptos de la aplicación son modelados como clases, cada una de las cuales describe un conjunto de objetos que almacenan información y se comunican para implementar un comportamiento. La información que almacena es modelada como atributos; La estructura estática se expresa con diagramas de clases y puede usarse para generar la mayoría de las declaraciones de estructuras de datos en un programa.

Comportamiento dinámico:

Hay dos formas de modelar el comportamiento, una es la historia de la vida de un objeto y la forma como interactúa con el resto del mundo y la otra es por los patrones de comunicación de un conjunto de objetos conectados, es decir la forma en que interactúan entre sí. La visión de un objeto aislado es una maquina de estados, muestra la forma en que el objeto responde a los eventos en función de su estado actual. La visión de la interacción de los objetos se representa con los enlaces entre objetos junto con el flujo de mensajes y los enlaces entre ellos. Este punto de vista unifica la estructura de los datos, el control de flujo y el flujo de datos.

Construcciones de implementación:

Los modelos UML tienen significado para el análisis lógico y para la implementación física. Un componente es una parte física reemplazable de un sistema y es capaz de responder a las peticiones descritas por un conjunto de interfaces. Un nodo es un recurso computacional que define una localización durante la ejecución de un sistema. Puede contener componentes y objetos.

Organización del modelo:

La información del modelo debe ser dividida en piezas coherentes, para que los equipos puedan trabajar en las diferentes partes de forma concurrente. El conocimiento humano requiere que se organice el contenido del modelo en paquetes de tamaño modesto. Los paquetes son unidades organizativas, jerárquicas y de propósito general de los modelos de UML.

Pueden usarse para almacenamiento, control de acceso, gestión de la configuración y construcción de bibliotecas que contengan fragmentos de código reutilizable.

Mecanismos de extensión:

UML tiene una limitada capacidad de extensión pero que es suficiente para la mayoría de las extensiones que requiere el día a día sin la necesidad de un cambio en el lenguaje básico. Un estereotipo es una nueva clase de elemento de modelado con la misma estructura que un elemento existente pero con restricciones adicionales.

Un Modelo captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle. Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés. El código fuente del sistema es el modelo más detallado del sistema (y además es ejecutable). Sin embargo, se requieren otros modelos. Cada modelo es completo desde su punto de vista del sistema, además, existen relaciones entre los diferentes modelos.

2.3.5.- Diagramas de UML

Un Diagrama es una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo conexo de arcos (relaciones) y vértices (otros elementos del modelo); es un elemento semántico, muestra representaciones de elementos semánticos del modelo, pero su significado no se

ve afectado por la forma en que son representados. Además está contenido dentro de un paquete.

La mayoría de los diagramas de UML y algunos símbolos complejos son grafos que contienen formas conectadas por rutas. La información está sobre todo en la topología, no en el tamaño o la colocación de los símbolos (hay algunas excepciones como el DS con un eje métrico de tiempo). Hay tres clases importantes de relaciones visuales:

a. Conexión.- generalmente de líneas a formas de dos dimensiones.

b. Contención.- de símbolos por formas cerradas de dos dimensiones.

c. Adhesión visual.- un símbolo que está "cerca" de otro en un diagrama.

Estas relaciones geométricas se reasignan a conexiones entre nodos en un gráfico en la forma analizada de la notación.

La notación de UML está pensada para ser dibujada en superficies bidimensionales. Algunas formas bidimensionales son proyecciones de formas tridimensionales tales como cubos, pero todavía se representan como íconos en una superficie bidimensional.

Hay cuatro clases de construcciones gráficas que se usan en la notación de UML:

a. Icono.- es una figura gráfica con un tamaño y forma fijos. No se amplía para contener a su contenido, estos pueden aparecer dentro de símbolos

de área, como terminadores en las rutas o como símbolos independientes que puedan o no conectar con las rutas.

b. Símbolos.- los de dos dimensiones tienen altura y anchura variables, y pueden ampliarse para permitir otras cosas tales como listas de cadenas o de otros símbolos. Muchos de ellos están divididos en compartimientos similares o de tipos diferentes.
c. Rutas.- se conectan con los símbolos, el arrastrar o suprimir uno de ellos afecta a su contenido y las rutas conectadas. Una ruta es una secuencia de segmentos de recta o de curva que se unen en sus puntos finales. Conceptualmente una ruta es una sola entidad topológica, aunque sus segmentos se pueden manipular gráficamente. un segmento no debería existir separado de su ruta. Las rutas siempre van conectadas en ambos extremos.
d. Cadenas.- presentan varias clases de información en una forma "no analizada", UML asume que cada uso de una cadena en la notación tiene una sintaxis por la cual pueda ser analizada la información del modelo subyacente. Las cadenas pueden existir como el contenido de un compartimiento, elementos en las listas, etiquetas unidas a los símbolos o a las rutas, o elementos independientes en un diagrama.

UML está compuesto por los siguientes diagramas: Tabla 2.1.

Tabla 2.1: (Resumen de Diagramas de UML)

Área	Vista	Diagramas	Conceptos Principales
Estructural	Vista Estática	Diagrama de Clases	Clase, asociación, generalización, dependencia, realización, interfaz.
	Vista de Casos de Uso	Diagramas de Casos de Uso	Caso de Uso, Actor, asociación, extensión, generalización.
	Vista de Implementación	Diagramas de Componentes	Componente, interfaz, dependencia, realización.
	Vista de Despliegue	Diagramas de Despliegue	Nodo, componente, dependencia, localización.
Dinámica	Vista de Estados de máquina	Diagramas de Estados	Estado, evento, transición, acción.
	Vista de actividad	Diagramas de Actividad	Estado, actividad, transición, determinación, división, unión.

	Vista de interacción	Diagramas de Secuencia	Interacción, objeto, mensaje, activación.
		Diagramas de Colaboración	Colaboración, interacción, rol de colaboración, mensaje.
Administración o Gestión de modelo	Vista de Gestión de modelo	Diagramas de Clases	Paquete, subsistema, modelo.
Extensión de UML	Todas	Todos	Restricción, estereotipo, valores, etiquetados.

2.3.5.1.- Diagramas de Objetos

Objeto es una entidad discreta con límites bien definidos y con identidad, es una unidad atómica que encapsula estado y comportamiento. La encapsulación en un objeto permite una alta cohesión y un bajo acoplamiento. El Objeto es reconocido también como una instancia de la clase a la cual pertenece.

La encapsulación presenta tres ventajas básicas:

- Se protegen los datos de accesos indebidos.
- El acoplamiento entre las clases se disminuye.
- Favorece la modularidad y el mantenimiento

Un objeto se puede ver desde dos perspectivas relacionadas: Como una entidad de un determinado instante de tiempo que posee un valor específico y como un poseedor de identidad que tiene distintos valores a lo largo del tiempo.

Cada objeto posee su propia identidad exclusiva y se puede hacer referencia a él mediante una denominación exclusiva que permite accederle.

El Modelado de Objetos permite representar el ciclo de vida de los objetos a través de sus interacciones. En UML, un objeto se representa por un rectángulo con un nombre subrayado.

- Objeto = Identidad + Estado + Comportamiento.
- El estado está representado por los valores de los atributos.
- Un atributo toma un valor en un dominio concreto.

La regla general para la notación de instancias consiste en utilizar el mismo símbolo geométrico que el descriptor. En la instancia se muestran los posibles valores pero las propiedades compartidas sólo se ponen de manifiesto en el descriptor. La notación canónica es un rectángulo con tres compartimientos. En el primero va el nombre del objeto, en el segundo sus atributos y en el tercero sus operaciones. Este último puede ser omitido si así se prefiere.

Oid (Object Identifier)

Cada objeto posee un oid. El oid establece la identidad del objeto y tiene las siguientes características:

a. Constituye un identificador único y global para cada objeto dentro del sistema.
b. Es determinado en el momento de la creación del objeto.
c. Independiente de la localización física del objeto, es decir, provee completa independencia de localización.
d. Independiente de las propiedades del objeto, lo cual implica independencia de valor y de estructura.
e. No cambia durante toda la vida del objeto. Además, un oid no se reutiliza aunque el objeto deje de existir.

No se tiene ningún control sobre los oids y su manipulación resulta transparente. Sin embargo, es preciso contar con algún medio para hacer referencia a un objeto utilizando referencias del dominio (valores de atributos).

Características alrededor de un objeto

a. Estado.- Evoluciona con el tiempo. Algunos atributos pueden ser constantes, el comportamiento agrupa las competencias de un objeto y describe las acciones y reacciones de ese objeto. Las operaciones de un objeto son consecuencia de un estímulo externo representado como mensaje enviado desde otro objeto.
b. Persistencia.- Designa la capacidad de un objeto trascender en el espacio/tiempo, podremos después reconstruirlo, es decir, cogerlo de memoria secundaria para utilizarlo en la ejecución (materialización del objeto). Los lenguajes OO no proponen soporte adecuado para la persistencia, la cual debería ser transparente, un objeto existe desde su creación hasta que se destruya.
c. Comunicación.- Un sistema informático puede verse como un conjunto de objetos autónomos y concurrentes que trabajan de manera coordinada en la consecución de un fin específico. El comportamiento global se basa pues en la comunicación entre los objetos que la componen

Categorías de objetos

- Activos o pasivos
- Cliente, servidores, agentes

a. Objeto activo.- posee un hilo de ejecución (thread) propio y puede iniciar una actividad.

b. Objeto pasivo.- no puede iniciar una actividad pero puede enviar estímulos una vez que se le solicita un servicio.
c. Cliente.- es el objeto que solicita un servicio.
d. Servidor.- es el objeto que provee el servicio solicitado.
e. Agentes.- estos reúnen las características de clientes y servidores. Son la base del mecanismo de delegación. Introducen indirección, es decir un cliente puede comunicarse con un servidor que no conoce directamente.

Mensajes

La unidad de comunicación entre objetos se llama mensaje. El mensaje es el soporte de una comunicación que vincula dinámicamente los objetos que fueron separados previamente en el proceso de descomposición. Adquiere toda su fuerza cuando se asocia al polimorfismo y al enlace dinámico. Un estímulo causará la invocación de una operación, la creación o destrucción de un objeto o la aparición de una señal. Un mensaje es la especificación de un estímulo.

Tipos de flujo de control

- Llamada a procedimiento o flujo de control anidado.
- Flujo de control plano.
- Retorno de una llamada a procedimiento.
- Otras variaciones.
- Esperado (balking).
- Cronometrado (time-out).

2.3.5.2.- Diagrama de Clases

Este diagrama es el principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones.

El mundo real puede ser visto desde abstracciones diferentes (subjetividad)

Mecanismos de abstracción:

- Clasificación - Instanciación.
- Composición – Descomposición.
- Agrupación – Individualización.
- Especialización - Generalización

La clasificación es uno de los mecanismos de abstracción más utilizados. La clase define el ámbito de definición de un conjunto de objetos, y cada objeto pertenece a una clase, Los objetos se crean por instanciación de las clases.
Cada clase se representa en un rectángulo con tres compartimientos, estos son los siguientes:

a. Nombre de la clase.
b. Atributos de la clase.
c. Operaciones de la clase

Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos. Por esta razón se crearon niveles de visibilidad para los elementos que son:

a. (-) Privado.- Es el más fuerte. Esta parte es totalmente invisible (excepto para clases friends en terminología C++).
b. (#) Los atributos/operaciones protegidos están visibles para las clases friends y para las clases derivadas de la original.
c. (+) Los atributos/operaciones públicos son visibles a otras clases (cuando se trata de atributos se está transgrediendo el principio de encapsulación)

Relaciones entre clases

Los enlaces entre objetos pueden representarse entre las respectivas clases y sus formas de relación son las siguientes:

- Asociación y Agregación (vista como un caso particular de asociación).
- Generalización / Especialización.

Las relaciones de Agregación y Generalización forman jerarquías de clases.

Asociación

La asociación expresa una conexión bidireccional entre objetos. Una asociación es una abstracción de la relación existente en los enlaces entre los objetos. Puede determinarse por la especificación de multiplicidad (mínima...máxima)

- Uno y sólo uno
- 0..1 Cero o uno
- M..N Desde M hasta N (enteros naturales)
- * Cero o muchos
- 0..* Cero o muchos
- 1..* Uno o muchos (al menos uno)

Agregación

La agregación representa una relación parte-de entre objetos. En UML se proporciona una escasa caracterización de la agregación. Esta relación puede ser caracterizada con precisión determinando las relaciones de comportamiento y estructura que existen entre el objeto agregado y cada uno de sus objetos componentes.

Puede el objeto parte comunicarse directamente con objetos externos al objeto agregado?

No => inclusiva

Si => no inclusiva

Puede cambiar la composición del objeto agregado?

Si => dinámica

No => estática

Diagrama de Clases y Diagramas de Objetos pertenecen a dos vistas complementarias del modelo. Un Diagrama de Clases muestra la abstracción de una parte del dominio. Un Diagrama de Objetos representa una situación concreta del dominio. Las clases abstractas no son instanciadas.

Generalización

Permite gestionar la complejidad mediante un ordenamiento taxonómico de clases, se obtiene usando los mecanismos de abstracción de Generalización y/o Especialización. La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general, se usa como nombres mas comunes (clase padre - clase hija), alternativamente son normados como (superclase - subclase, clase base - clase derivada). Las subclases heredan propiedades de sus clases padre, es decir, atributos y operaciones (asociaciones) de la clase padre están disponibles en sus clases hijas. La Generalización y Especialización son equivalentes en cuanto al resultado: La jerarquía y herencia establecidas. Generalización y Especialización no son operaciones reflexivas ni simétricas pero sí transitivas. La especialización es una técnica muy eficaz para la extensión y reutilización.

La noción de clase está próxima a la de conjunto. Dada una clase, podemos ver el conjunto relativo a las instancias que posee o bien relativo a las propiedades de la clase. Generalización y especialización expresan relaciones de inclusión entre conjuntos.

2.3.5.3.- Diagramas de Interacción

La vista de interacción describe secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Un rol clasificador, o simplemente "un rol", es la descripción de un objeto, que desempeña un determinado papel dentro de una interacción, distinto de los otros objetos de la misma clase. Esta visión proporciona una vista integral del

comportamiento del sistema, es decir, muestra el flujo de control a través de muchos objetos. La vista de interacción se exhibe en dos diagramas centrados en distintos aspectos pero complementarios; centrados en los objetos individuales y centrados en objetos cooperantes.

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Los diagramas de interacción muestran cómo se comunican los objetos en una interacción. Existen dos tipos de diagramas de interacción: el Diagrama de Colaboración (DC) y el Diagrama de Secuencia (DS).

El DS es más adecuado para observar la perspectiva cronológica de las interacciones, muestra la secuencia explícita de mensajes y son mejores para especificaciones de tiempo real y para escenarios complejos. El DS ofrece una mejor visión espacial mostrando los enlaces de comunicación entre objetos, muestra las relaciones entre objetos y son mejores para comprender todos los efectos que tiene un objeto y para el diseño de procedimientos. El DC puede obtenerse automáticamente a partir del correspondiente DS (o viceversa).

Diagramas de Secuencia

a. Muestra la secuencia de mensajes entre objetos durante un escenario concreto.
b. Cada objeto viene dado por una barra vertical.
c. El tiempo transcurre de arriba abajo.
d. Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua.

Diagramas de Colaboración

a. Son útiles en la fase exploratoria para identificar objetos.
b. La distribución de los objetos en el diagrama permite observar adecuadamente la interacción de un objeto con respecto de los demás
c. La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces

Qué es una colaboración?

Es una descripción de una colección de objetos que interactúan para implementar un cierto comportamiento dentro de un contexto. Describe una sociedad de objetos cooperantes unidos para realizar un cierto propósito. Una colaboración contiene ranuras que son rellenadas por los objetos y enlaces en tiempo de ejecución. Una ranura de colaboración se llama Rol porque describe el propósito de un objeto o un enlace dentro de la colaboración.

Un rol clasificador representa una descripción de los objetos que pueden participar en una ejecución de la colaboración, un rol de asociación representa una descripción de los enlaces que pueden participar en una ejecución de colaboración.

Un rol de clasificador es una asociación que está limitada por tomar parte en la colaboración. Las relaciones entre roles de clasificador y asociación dentro de una colaboración sólo tienen sentido en ese contexto. En general fuera de ese contexto no se aplican las mismas relaciones.

Una colaboración tiene un aspecto estructural y un aspecto de comportamiento. El aspecto estructural es similar a una vista estática, contiene un conjunto de roles y relaciones que definen el contexto para su comportamiento. El comportamiento es el conjunto de mensajes intercambiados por los objetos ligados a los roles. Tal conjunto de mensajes en una colaboración se llama Interacción. Una colaboración puede incluir una o más interacciones.

Qué es una interacción?

Es el conjunto de mensajes intercambiados por los roles de clasificador a través de los roles de asociación. Un mensaje es una comunicación unidireccional entre dos objetos, un flujo de objeto con la información de un remitente a un receptor. Un mensaje puede tener parámetros que transporten valores entre objetos.

Un mensaje puede ser una señal (comunicación explícita entre objetos, con nombre y asíncrona) o una llamada (la invocación síncrona de una operación con un mecanismo para el control, que retorna posteriormente al remitente). Un patrón de intercambios de mensajes que se realizan para lograr un propósito específico es lo que se denomina una interacción.

Qué es patrón?

Un patrón es una colaboración parametrizada, junto con las pautas sobre cuándo utilizarlo. Un parámetro se puede sustituir por diversos valores, para producir distintas colaboraciones. Los parámetros señalan generalmente las ranuras para las clases. El uso de un patrón se representa como una elipse de

línea discontinua conectada con cada una de las clases por una línea discontinua, que se etiqueta con el nombre del rol.

2.3.5.4.- Diagramas de Casos de Uso

Casos de Uso es una técnica para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje. No pertenece estrictamente al enfoque orientado a objeto, es una técnica para captura de requisitos.

- Los casos de uso (Ivar Jacobson) describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario.
- Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.
- Los casos de uso son descripciones de la funcionalidad del sistema independientes de la implementación.
- Comparación con respecto a los Diagramas de Flujo de Datos del Enfoque Estructurado.
- Los Casos de Uso cubren la carencia existente en métodos previos (OMT, Booch) en cuanto a la determinación de requisitos.
- Los casos de uso particionan el conjunto de necesidades atendiendo a la categoría de usuarios que participan en el mismo.
- Están basados en el lenguaje natural, es decir, es accesible por los usuarios.

Actores

- Principales: Personas que usan el sistema.
- Secundarios: Personas que mantienen o administran el sistema.
- Material externo: Dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- Otros sistemas: Sistemas con los que el sistema interactúa.

La misma persona física puede interpretar varios papeles como actores distintos, el nombre del actor describe el papel desempeñado.

Los Casos de Uso se determinan observando y precisando, actor por actor, las secuencias de interacción, los escenarios, desde el punto de vista del usuario. Los casos de uso intervienen durante todo el ciclo de vida. El proceso de desarrollo estará dirigido por los casos de uso. Un escenario es una instancia de un caso de uso.

UML define cuatro tipos de relación en los Diagramas de Casos de Uso:

- Comunicación.
- Inclusión: Una instancia del Caso de Uso origen incluye también el comportamiento descrito por el Caso de Uso destino. [include] reemplazó al denominado [uses].
- Extensión: El Caso de Uso origen extiende el comportamiento del Caso de Uso destino. [extend].
- Herencia: El Caso de Uso origen hereda la especificación del Caso de Uso destino y posiblemente la modifica y/o amplía.

Parámetros para la construcción de un caso de uso:

Un caso de uso debe ser simple, inteligible, claro y conciso. Generalmente hay pocos actores asociados a cada Caso de Uso. Preguntas clave:

a. Cuáles son las tareas del actor?
b. Qué información crea, guarda, modifica, destruye o lee el actor?
c. Debe el actor notificar al sistema los cambios externos?
d. Debe el sistema informar al actor de los cambios internos?

La descripción del caso de uso comprende:

a. El inicio: Cuándo y qué actor lo produce?
b. El fin: Cuándo se produce y qué valor devuelve?
c. La interacción actor-caso de uso: Qué mensajes intercambian ambos?
d. Objetivo del caso de uso: Qué lleva a cabo o intenta?
e. Cronología y origen de las interacciones.
f. Repeticiones de comportamiento: Qué operaciones son iteradas?
g. Situaciones opcionales: Qué ejecuciones alternativas se presentan en el caso de uso?

2.3.5.5.- Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas

dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas.

Un componente de sólo compilación es aquel que es significativo únicamente en tiempo de compilación. Un componente ejecutable es un programa ejecutable.

Un diagrama de componentes tiene sólo una versión con descriptores, no tiene versión con instancias. Para mostrar las instancias de los componentes se debe usar un diagrama de despliegue.

Un diagrama de componentes muestra clasificadores de componentes, las clases definidas en ellos, y las relaciones entre ellas. Los clasificadores de componentes también se pueden anidar dentro de otros clasificadores de componentes para mostrar relaciones de definición.

Un diagrama que contiene clasificadores de componentes y de nodo se puede utilizar para mostrar las dependencias del compilador, que se representa como flechas con líneas discontinuas (dependencias) de un componente cliente a un componente proveedor del que depende. Los tipos de dependencias son

específicos del lenguaje y se pueden representar como estereotipos de las dependencias.

El diagrama también puede usarse para mostrar interfaces y las dependencias de llamada entre componentes, usando flechas con líneas discontinuas desde los componentes a las interfaces de otros componentes.

El diagrama de componente hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. Esta vista proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de implementación y nodos. La vista de implementación se representa con los diagramas de componentes.

Qué es un componente?

Es una parte física reemplazable de un sistema que empaqueta su implementación y es conforme a un conjunto de interfaces a las que proporciona su realización.

Algunos componentes tienen identidad y pueden poseer entidades físicas, que incluyen objetos en tiempo de ejecución, documentos, bases de datos, etc. Los componentes existentes en el dominio de la implementación son unidades físicas en los computadores que se pueden conectar con otros componentes, sustituir, trasladar, archivar, etc.

Los componentes tienen dos características: Empaquetan el código que implementa la funcionalidad de un sistema, y algunas de sus propias instancias

de objetos que constituyen el estado del sistema. Los llamados últimos componentes de la identidad, porque sus instancias poseen identidad y estado.

Código:

Un componente contiene el código para las clases de implementación y otros elementos. Un componente de código fuente es un paquete para el código fuente de las clases de implementación. Algunos lenguajes de programación distinguen archivos de declaración de los archivos de método, pero todos son componentes. Un componente de código binario es un paquete para el código compilado. Una biblioteca del código binario es un componente.

Cada tipo de componente contiene el código para las clases de implementación que realizan algunas clases e interfaces lógicas. La relación de realización asocia un componente con las clases y las interfaces lógicas que implementan sus clases de implementación. Las interfaces de un componente describen la funcionalidad que aporta. Cada operación de la interfaz debe hacer referencia eventualmente a un elemento de la implementación disponible en el componente.

La estructura estática, ejecutable de una implementación de un sistema se puede representar como un conjunto interconectado de componentes. Las dependencias entre componentes significan que los elementos de la implementación en un componente requieren los servicios de los elementos de implementación en otros componentes. Tal uso requiere que dichos elementos sean de visibilidad pública.

Identidad:

Un componente de identidad tiene identidad y estado. Posee los objetos físicos que están situados en él. Puede tener atributos, relaciones de composición con los objetos poseídos, y asociaciones con otros componentes.

Desde este punto de vista es una clase. Sin embargo la totalidad de su estado debe hacer referencia a las instancias que contiene.

Estructura:

Un componente ofrece un conjunto de elementos de implementación, esto significa que el componente proporciona el código para los elementos. Un componente puede tener operaciones e interfaces. Un componente de identidad es un contenedor físico para las entidades físicas como bases de datos. Para proporcionar manejadores para sus elementos contenidos, puede tener atributos y asociaciones salientes, que deben ser implementadas por sus elementos de implementación. Este componente se representa con un rectángulo con dos rectángulos más pequeños que sobresalen en su lado izquierdo.

Las operaciones e interfaces disponibles para los objetos exteriores se pueden representar directamente en el símbolo de clase. Estos son su comportamiento como clase. Los contenidos del subsistema se representan en un diagrama separado.

Las dependencias de un componente con otros componentes o elementos del modelo se representan usando líneas discontinuas con la punta de flecha hacia los elementos del proveedor. Sí un componente es la realización de una

interfaz, se representa con un círculo unido al símbolo del componente por un segmento de línea.

2.3.5.6.- Diagrama de Actividades

El diagrama de actividad es una especialización del diagrama de estado, organizado respecto de las acciones y usado para especificar:

- Un método
- Un caso de uso
- Un proceso de negocio (Workflow)

Un estado de actividad representa una actividad: Un paso en el flujo de trabajo o la ejecución de una operación. Un grafo de actividades describe grupos secuenciales y concurrentes de actividades. Los grafos de actividades se muestran en diagramas de actividades. Las actividades se enlazan por transiciones automáticas. Cuando una actividad termina se desencadena el paso a la siguiente actividad.

Un diagrama de actividades es provechoso para entender el comportamiento de alto nivel de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes. Los parámetros de entrada y salida de una acción se pueden mostrar usando las relaciones de flujo que conectan la acción y un estado de flujo de objeto.

Un grafo de actividades contiene estados de actividad que representa la ejecución de una secuencia en un procedimiento, o el funcionamiento de una actividad en un flujo de trabajo. En vez de esperar un evento, como en un estado

de espera normal, un estado de actividad espera la terminación de su cómputo. Cuando la actividad termina, entonces la ejecución procede al siguiente estado de actividad dentro del diagrama. Una transición de terminación es activada en un diagrama de actividades cuando se completa la actividad precedente. Los estados de actividad no tienen transiciones con eventos explícitos, peor pueden ser abortados por transiciones en estados que los incluyen.

Un grafo de actividades puede contener también estados de acción, que son similares a los de actividad pero son atómicos y no permiten transiciones mientras están activos. Los estados de acción se deben utilizar para las operaciones cortas de mantenimiento.

Un diagrama de actividades puede contener bifurcaciones, así como divisiones de control en hilos concurrentes. Los hilos concurrentes representan actividades que se pueden realizar concurrentemente por los diversos objetos o personas. La concurrencia se representa a partir de la agregación, en la cual cada objeto tiene su propio hilo. Las actividades concurrentes se pueden realizar simultáneamente o en cualquier orden. Un diagrama de actividades es como un organigrama tradicional, excepto que permite el control de concurrencia además del control secuencial.

Notación:

Un estado de actividad se representa como una caja con los extremos redondeados que contiene una descripción de actividad. Las transacciones simples de terminación se muestran como flechas. Las ramas se muestran como

condiciones de guarda en transiciones o como diamantes con múltiples flechas de salida etiquetadas.

Una división o una unión de control se representan con múltiples flechas que entran o salen de la barra gruesa de sincronización.

Cuando es necesario incluir eventos externos, la recepción de un evento se puede mostrar como un disparador en una transición, o como un símbolo especial que denota la espera de una señal.

A menudo es útil organizar las actividades en un modelo según su responsabilidad. Esta clase de asignación puede mostrarse organizando las actividades en regiones distintas separadas por líneas en el diagrama. Debido a su aspecto, esto es conocido como **Calles**.

Un diagrama de actividades puede mostrar el flujo de objetos como valores. Para un valor de salida, se dibuja una flecha con línea discontinua desde la actividad al objeto. Para un valor de entrada, se dibuja una flecha con línea discontinua desde el objeto a una actividad.

2.3.5.7.- Diagramas de Despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos.

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de

comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

Los estereotipos permiten precisar la naturaleza del equipo:

- Dispositivos.
- Procesadores.
- Memoria

Los nodos se interconectan mediante soportes bidireccionales que pueden a su vez estereotiparse. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos. Las instancias de los nodos pueden contener instancias de ejecución, como instancias de componentes y objetos.

El modelo puede mostrar dependencias entre las instancias y sus interfaces, y también modelar la migración de entidades entre nodos u otros contenedores.

Esta vista tiene una forma de descriptor y otra de instancia. La forma de instancia muestra la localización de las instancias de los componentes específicos en instancias específicas del nodo como parte de una configuración del sistema. La forma de descriptor muestra qué tipo de componentes pueden subsistir en qué tipos de nodos y qué tipo de nodos se pueden conectar, de forma similar a un diagrama de clases, esta forma es menos común que la primera.

2.3.5.8.- Diagramas de Estado

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro.

Los Diagramas de Estado representan autómatas de estados finitos, desde el punto de vista de los estados y las transiciones. Son útiles sólo para los objetos con un comportamiento significativo. Cada objeto está en un estado en cierto instante.

El estado está caracterizado parcialmente por los valores algunos de los atributos del objeto. El estado en el que se encuentra un objeto determina su comportamiento. Cada objeto sigue el comportamiento descrito en el Diagrama de Estados asociado a su clase.

Los Diagramas de Estados y escenarios son complementarios, los Diagramas de Estados son autómatas jerárquicos que permiten expresar concurrencia, sincronización y jerarquías de objetos, son grafos dirigidos y deterministas. La transición entre estados es instantánea y se debe a la ocurrencia de un evento.

Estado:

Identifica un periodo de tiempo del objeto (no instantáneo) en el cual el objeto está esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos. Se representa mediante un rectángulo con los bordes redondeados, que puede tener tres compartimientos: Uno para el nombre, otro para el valor característico de los atributos del objeto en ese estado y otro para las acciones que se realizan al entrar, salir o estar en un estado (entry, exit o do, respectivamente).

Eventos:

Es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una de varias cosas:

- Condición que toma el valor de verdadero o falso.
- Recepción de una señal de otro objeto en el modelo.
- Recepción de un mensaje.
- Paso de cierto período de tiempo, después de entrar al estado o de cierta hora y fecha particular

El nombre de un evento tiene alcance dentro del paquete en el cual está definido, no es local a la clase que lo nombre.

Envío de mensajes

Además de mostrar y transición de estados por medio de eventos, puede representarse el momento en el cual se envían mensajes a otros objetos. Esto se realiza mediante una línea punteada dirigida al diagrama de estados del objeto receptor del mensaje.

Transición simple

Una transición simple es una relación entre dos estados que indica que un objeto en el primer estado puede entrar al segundo estado y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas. Se representa como una línea sólida entre dos estados, que puede venir acompañada de un texto con el siguiente formato:

event-signature "[" guard-condition] "/" action-expression "^"send-clause

event-signature es la descripción del evento que da lugar la transición, guard-condition son las condiciones adicionales al evento necesarias para que la transición ocurra, **action-expression** es un mensaje al objeto o a otro objeto que se ejecuta como resultado de la transición y el cambio de estado y send-clause son acciones adicionales que se ejecutan con el cambio de estado, por ejemplo, el envío de eventos a otros paquetes o clases.

Transición interna

Es una transición que permanece en el mismo estado, en vez de involucrar dos estados distintos. Representa un evento que no causa cambio de estado. Se denota como una cadena adicional en el compartimiento de acciones del estado.

Acciones:

Podemos especificar la solicitud de un servicio a otro objeto como consecuencia de la transición. Se puede especificar el ejecutar una acción como consecuencia de entrar, salir, estar en un estado, o por la ocurrencia de un evento.

Generalización de Estados:

- Podemos reducir la complejidad de estos diagramas usando la generalización de estados.
- Distinguimos así entre superestado y subestados.
- Un estado puede contener varios subestados disjuntos.

- Los subestados heredan las variables de estado y las transiciones externas.
- La agregación de estados es la composición de un estado a partir de varios estados independientes.

La composición es concurrente por lo que el objeto estará en alguno de los estados de cada uno de los subestados concurrentes. La destrucción de un objeto es efectiva cuando el flujo de control del autómata alcanza un estado final no anidado. La llegada a un estado final anidado implica la subida al superestado asociado, no el fin del objeto.

Subestados

Un estado puede descomponerse en subestados, con transiciones entre ellos y conexiones al nivel superior. Las conexiones se ven al nivel inferior como estados de inicio o fin, los cuales se suponen conectados a las entradas y salidas del nivel inmediatamente superior.

Transacción Compleja

Una transición compleja relaciona tres o más estados en una transición de múltiples fuentes y/o múltiples destinos. Representa la subdivisión en threads del control del objeto o una sincronización. Se representa como una línea vertical de la cual salen o entran varias líneas de transición de estado.

Transición a estados anidados

Una transición de hacia un estado complejo (descrito mediante estados anidados) significa la entrada al estado inicial del subdiagrama. Las transiciones

que salen del estado complejo se entienden como transiciones desde cada uno de los subestados hacia afuera (a cualquier nivel de profundidad).

Transiciones temporizadas

- Las esperas son actividades que tienen asociada cierta duración..
- La actividad de espera se interrumpe cuando el evento esperado tiene lugar.
- Este evento desencadena una transición que permite salir del estado que alberga la actividad de espera. El flujo de control se transmite entonces a otro estado.

Paquetes

Cualquier sistema grande se debe dividir en unidades más pequeñas, de modo que las personas puedan trabajar con una cantidad de información limitada, a la vez y de modo que los equipos de trabajo no interfieran con el trabajo de los otros.

En la figura 2.3 consta la representación de la arquitectura de un sistema con la notación de paquetes.

Un paquete es una parte de un modelo. Cada parte del modelo debe pertenecer a un paquete. Pero para ser funcional, la asignación debe seguir un cierto principio racional, tal como funcionalidad común, implementación relacionada y punto de vista común. UML no impone una regla para componer los paquetes.

Figura 2.3: (Arquitectura de un sistema utilizando paquetes)

Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general de los modelos de UML. Pueden ser utilizados para el almacenamiento, el control de acceso, la gestión de la configuración y la construcción de bibliotecas que contengan fragmentos reutilizables del modelo.

Un paquete puede contener otros paquetes, sin límite de anidamiento pero cada elemento pertenece a (está definido en) sólo un paquete.

Los paquetes contienen elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones y colaboraciones; atributos, operaciones, estados, líneas de vida y

mensajes están contenidos en otros elementos y no aparecen como contenido directo de los paquetes.

Dependencias en los paquetes

Las dependencias que se presentan entre elementos individuales, pero en un sistema de cualquier tamaño, deben ser vistas en un nivel más alto. Las dependencias entre paquetes resumen dependencias entre los elementos internos a ellos, es decir, las dependencias del paquete son derivables a partir de las dependencias entre los elementos individuales.

La presencia de una dependencia entre paquetes implica que existe en un enfoque ascendente (una declaración de existencia), o que se permite que exista más adelante en un enfoque descendente (una restricción que limita cualquier otra relación), por lo menos un elemento de relación con el tipo de dependencia indicado entre elementos individuales dentro de los paquetes correspondientes.

Las dependencias múltiples del mismo tipo entre elementos individuales se agregan a una sola dependencia entre los paquetes que contienen los elementos. Si las dependencias entre elementos contienen estereotipos, éste puede ser omitido en la dependencia del paquete, para dar una sola dependencia de alto nivel.

Una clase de un paquete puede aparecer en otro paquete por la importación a través de una relación de dependencia entre paquetes. Todas las clases no son necesariamente visibles desde el exterior del paquete, es decir, un paquete encapsula a la vez que agrupa.

En general, un paquete no puede tener acceso al contenido de otro paquete. Los paquetes son opacos, a menos que sean abiertos por una dependencia de acceso o de importación. La dependencia de acceso indica que el contenido del paquete del proveedor puede aparecer en referencias efectuadas por los elementos del paquete cliente. En general, un paquete puede ver solamente los elementos de otros paquetes que tienen visibilidad pública. Los elementos con visibilidad protegida pueden ser vistos únicamente por los paquetes que son descendientes del paquete contenedor de dichos elementos. Los elementos con visibilidad privada sólo son vistos por su paquete contenedor y anidados. La visibilidad también se aplica a las clases. El permiso de acceso y visibilidad son necesarios para hacer referencia a un elemento.

La dependencia de acceso no modifica el espacio de nombres del cliente no crea las referencias automáticamente, simplemente concede permiso para establecer referencias.

La dependencia de importación se utiliza para agregar nombres al espacio de nombres del paquete del cliente como sinónimos de los caminos completos.

Los paquetes se dibujan como rectángulos con pestañas (similar al icono "carpeta"), las dependencias se muestran como flechas con líneas discontinuas. El operador "::" permite designar una clase definida en un contexto distinto del actual.

2.3.6.- Resumen de la Notación que se utiliza den UML

a.- Iconos Utilizados en los Diagramas de Clases

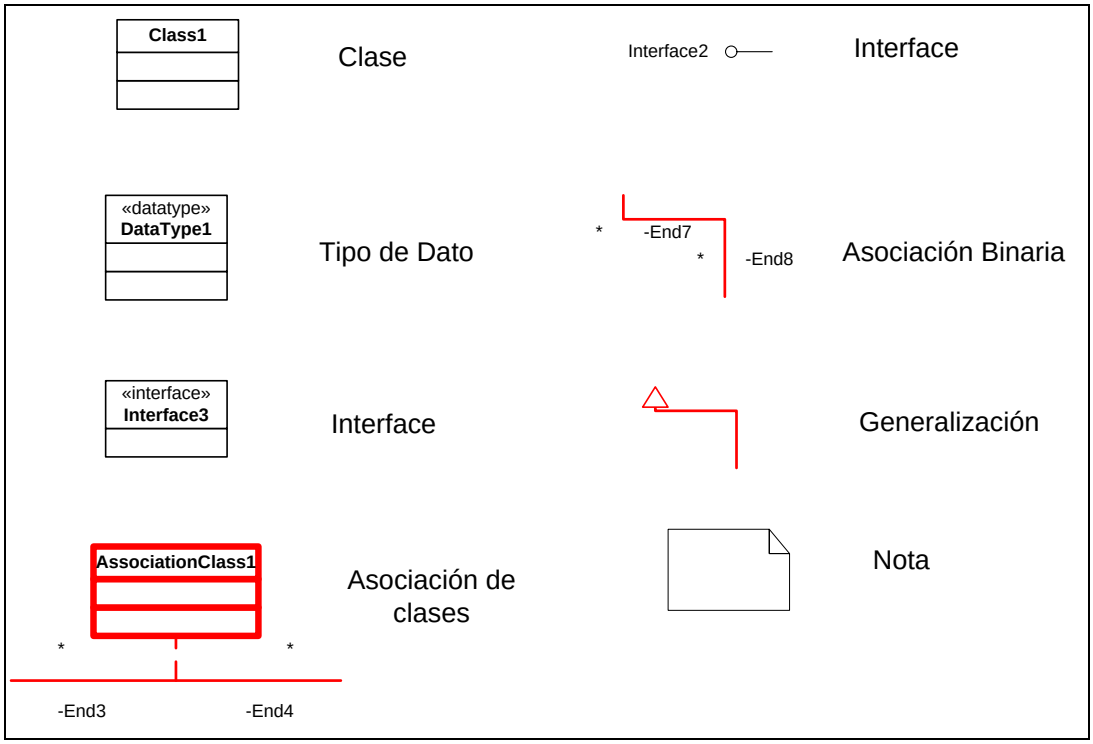

Figura 2.4: (Iconos Usados en los Diagramas de Clases)

b.- Iconos Utilizados en los Diagramas de Casos de Uso

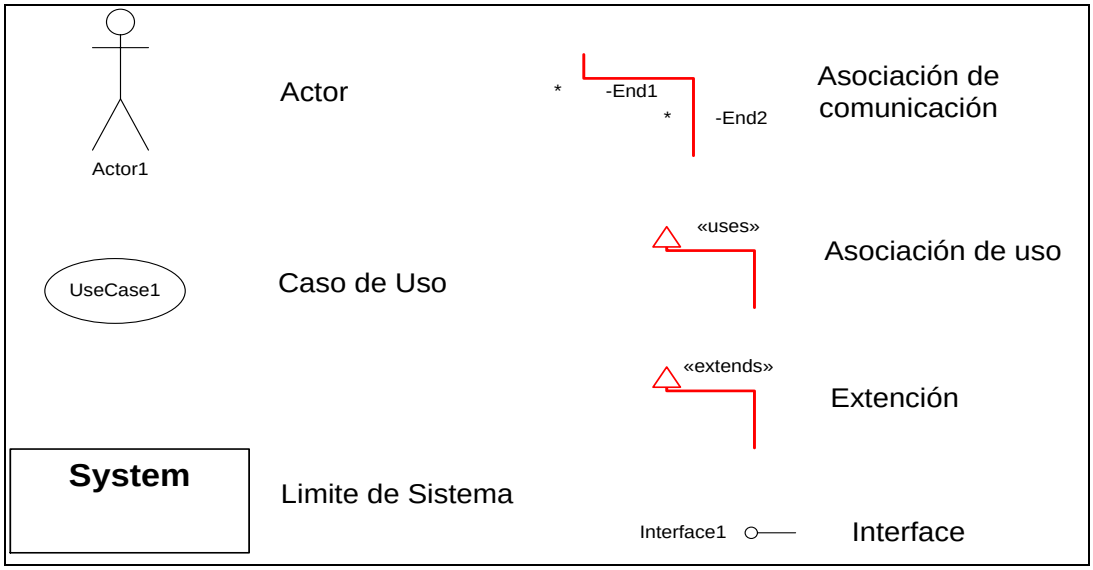

Figura 2.5: (Iconos Usados en los Diagramas de Casos de Uso)

c.- Iconos Utilizados en los Diagramas de Componentes

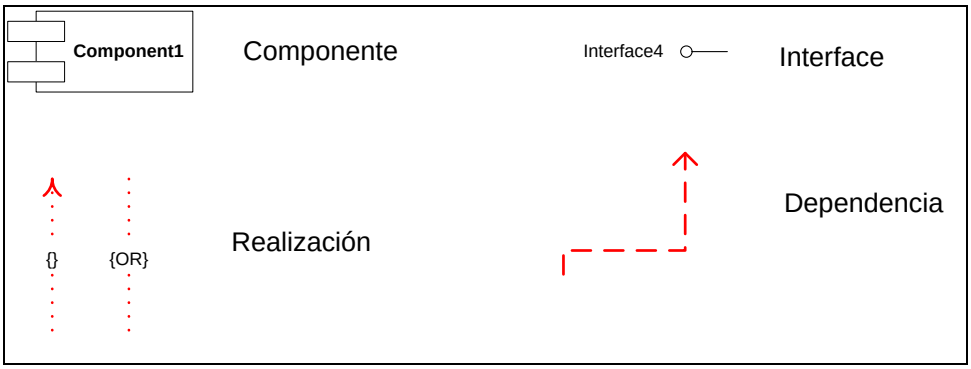

Figura 2.6: (Iconos Usados en los Diagramas de Componentes)

d.- Iconos Utilizados en los Diagramas de Despliegue

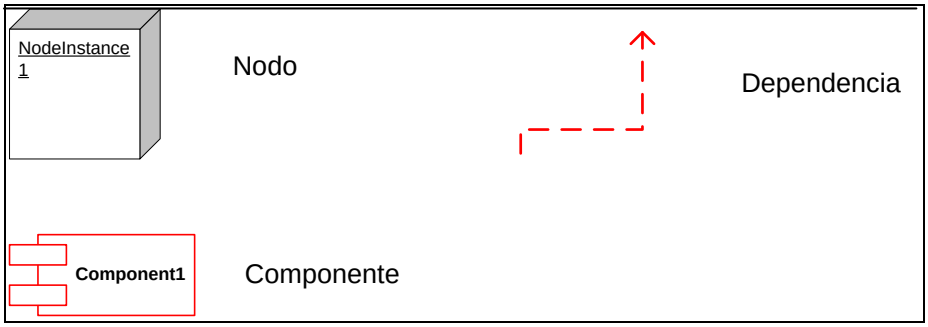

Figura 2.7: (Iconos Usados en los Diagramas de Despliegue)

e.- Iconos Utilizados en los Diagramas de Estados

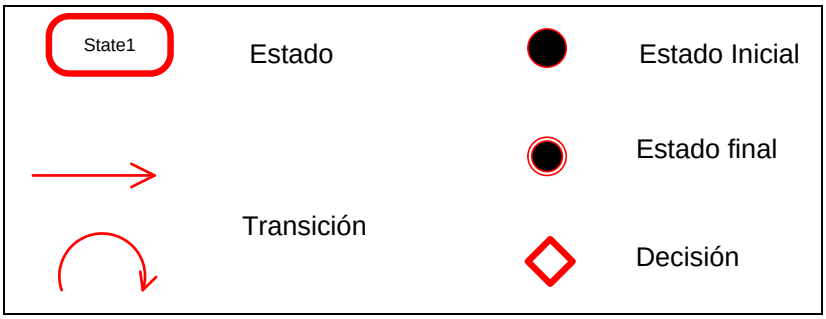

Figura 2.8: (Iconos Usados en los Diagramas de Estado)

f.- Iconos Utilizados en los Diagramas de Actividad

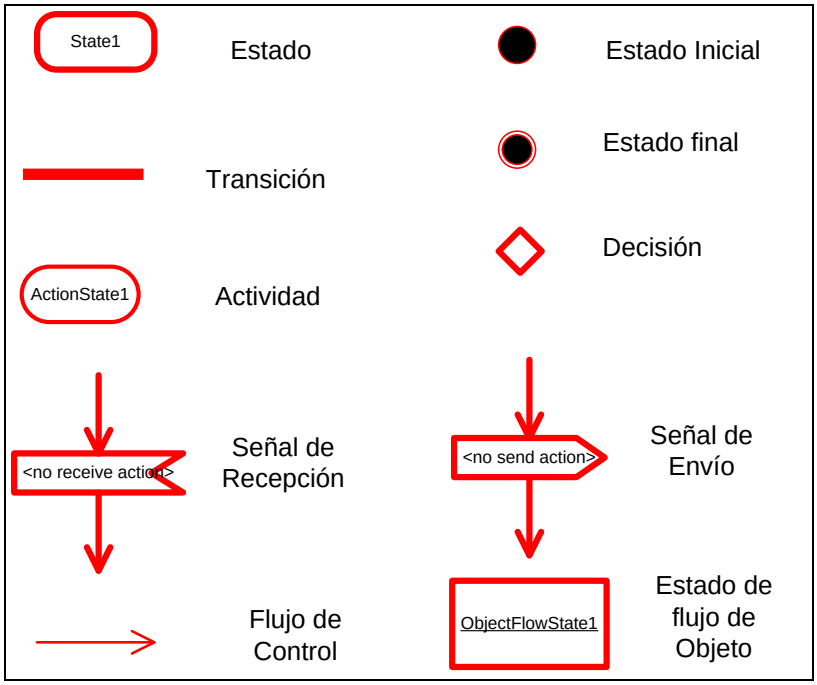

Figura 2.9: (Iconos Usados en los Diagramas de Componentes)

g.- Iconos Utilizados en los Diagramas de Secuencia

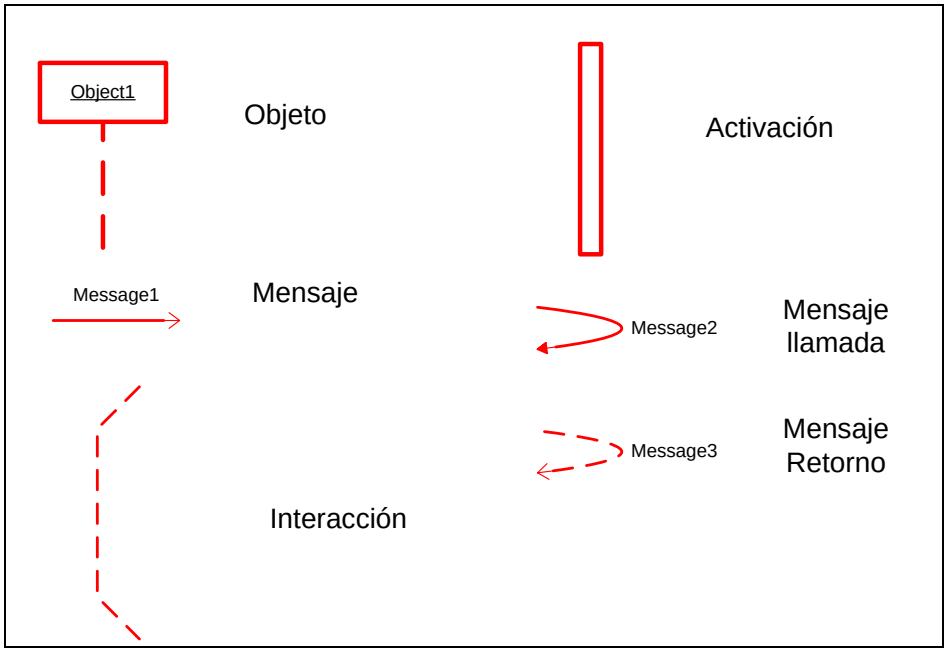

Figura 2.10: (Iconos Usados en los Diagramas de Secuencia)

h.- Iconos Utilizados en los Diagramas de Colaboración

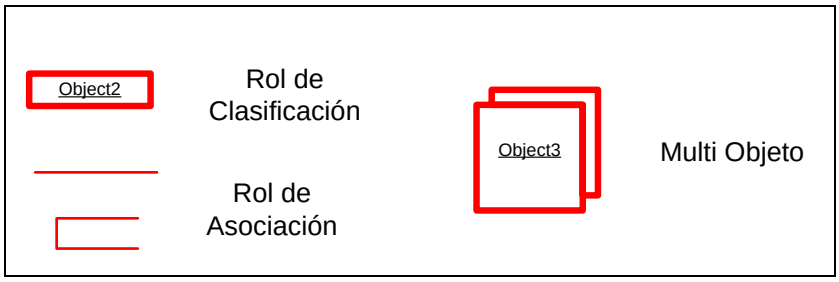

Figura 2.11 (Iconos Usados en los Diagramas de Colaboración)

i.- Iconos Utilizados para mensajes

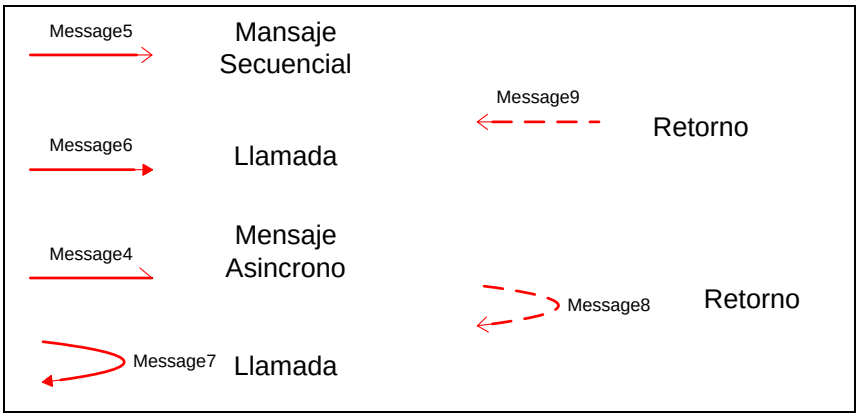

Figura 2.12: (Iconos Usados en UML para mensajes)

CAPITULO III

ESTUDIO Y RACIONALIZACIÓN DE PROCESOS

3.1.- AFIRMACIÓN DE CONOCIMIENTOS

3.1.1.- Proceso

- No existe un producto y/o servicio sin un proceso.
- No existe proceso sin un producto o servicio.

Un proceso es un conjunto estructurado de actividades que se realizan para producir un resultado específico y cumplir un objetivo determinado. Las actividades que conforman un proceso pueden involucrar a varios de los departamentos o funciones que comprenden la empresa. Las actividades del proceso pueden agruparse por homogeneidad o por ubicación dando lugar así a los procedimientos. Un procedimiento puede estar ubicado dentro de los límites de un área funcional o puede pertenecer a varias áreas funcionales.

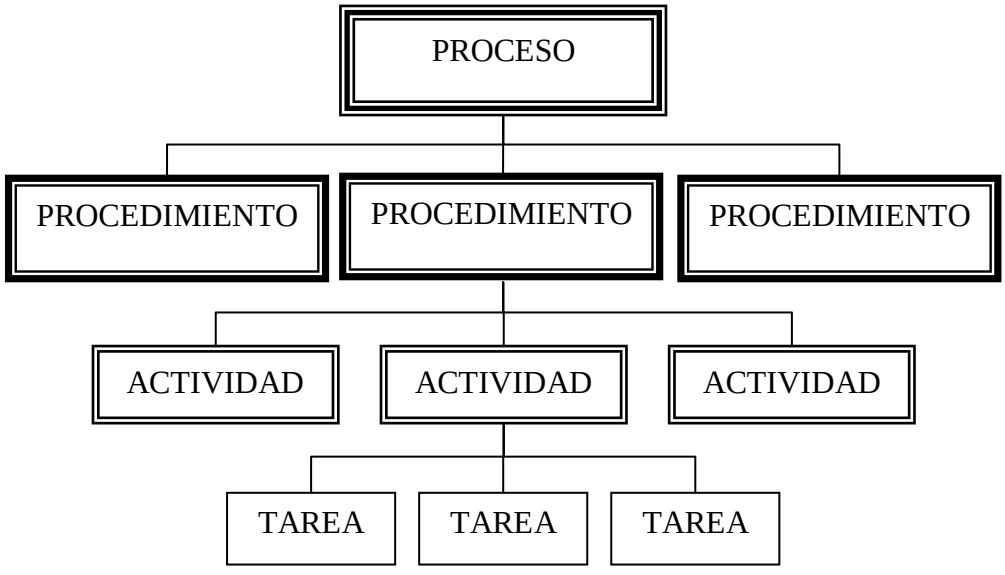

Figura 3.1: (Estructura básica de un proceso)

Todos los integrantes de una organización están involucrados en algún proceso de la misma. Figura 3.2

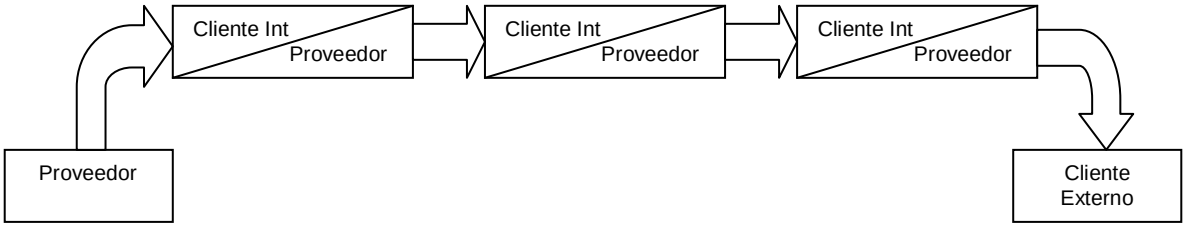

Figura 3.2: (Una organización es una cadena de procesos)

3.1.2.- Actividad

Son acciones que tienen lugar dentro de los procesos y son necesarias para generar un determinado resultado.

3.1.3.- Tarea

Consiste en realizar un trabajo en un determinado tiempo.

En resumen un proceso es una actividad o conjunto de actividades que emplean insumos (inputs), agregan valor y entregan un producto (outputs) a un cliente sea este interno o externo, como se puede observar en la figura 3.3.

3.1.4.- Principales elementos de un proceso

a. **Entrada o Activador (E):** Es el punto de inicio del proceso y corresponde a la salida o resultado de un proceso anterior.
b. **Origen:** Identifica el área o actividad que origina la entrada o activador del Proceso.
c. **Responsable:** Es el gestor del proceso y/o de las actividades que lo conforman. Existe un responsable por Proceso y por Actividad.

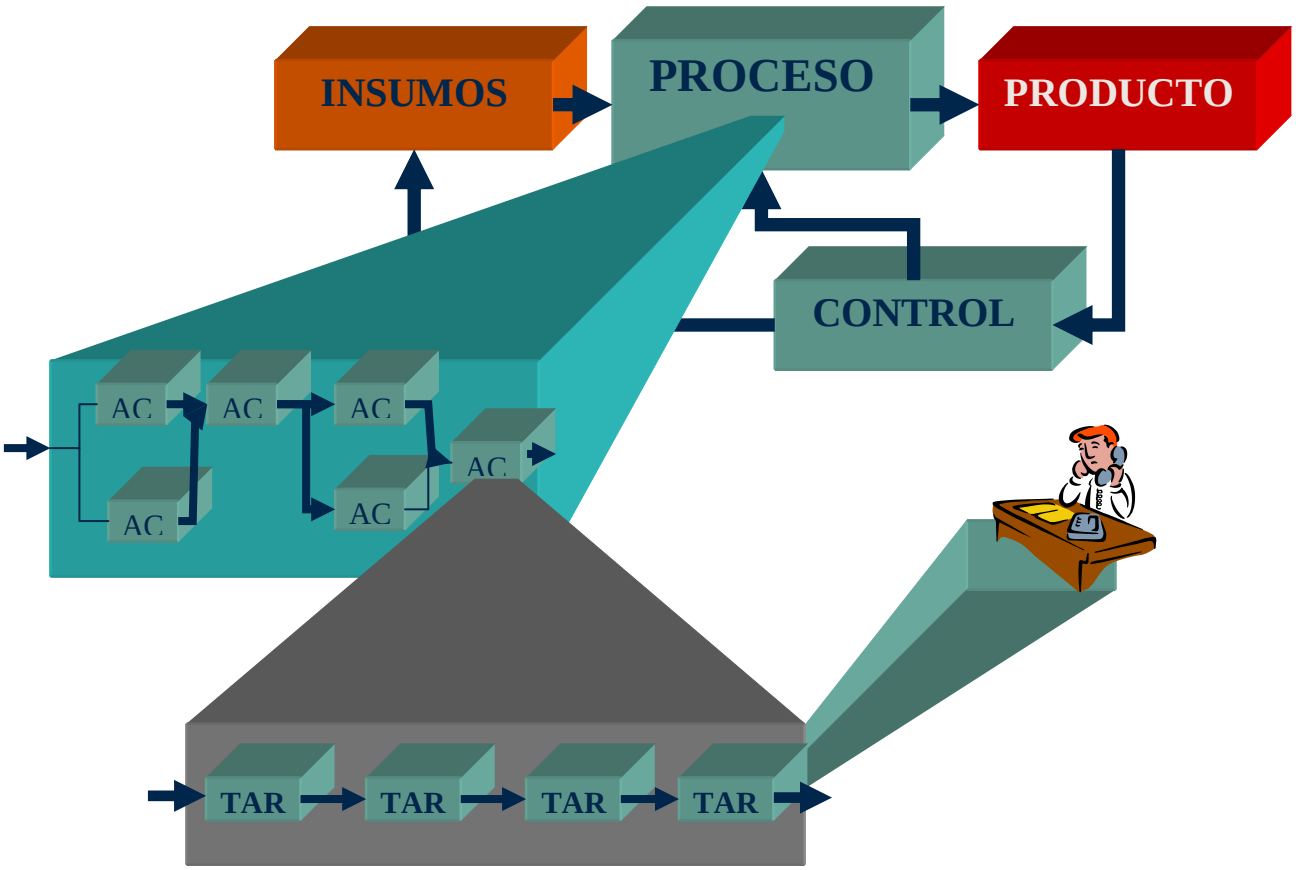

Figura 3.3: (Componentes básicos de un proceso)

d. **Actividades**: Son los grupos de tareas específicas que se ejecutan para la producción del producto o servicio.

e. **Salida**: Es el resultado o producto del proceso o de la actividad.

g. **Cliente o Receptor:** Es el que recibe la salida final del proceso. El receptor puede ser interno o externo a la Institución

Unidades de medida.- Son las que dimensionan la cantidad de entradas, recursos.

Tiempo del proceso.- Tiempo que toma la ejecución o realización de las actividades del proceso.

3.2.- Objetivo De La Racionalización

Organizar los recursos utilizados en un proceso, de tal forma que se reduzcan tanto los costos como el tiempo, manteniendo o mejorando las características de calidad esperadas del servicio. Se menciona algunos de los objetivos:

- Optimizar los recursos empleados en un proceso.
- Reducir costos.
- Reducir tiempos de respuestas.
- Mejorar la calidad del servicio.

La racionalización de procesos debe cubrir todos los procesos de la empresa.

3.3.- Proceso Soporte Técnico Sistemas

3.3.1.- Procesos Actuales

En la figura 3.4 se puede observar el flujo actual del soporte existente en EXSERSA.

En el anexo A se detalla el proceso actual, el mismo que contiene los tiempos registrados en la práctica.

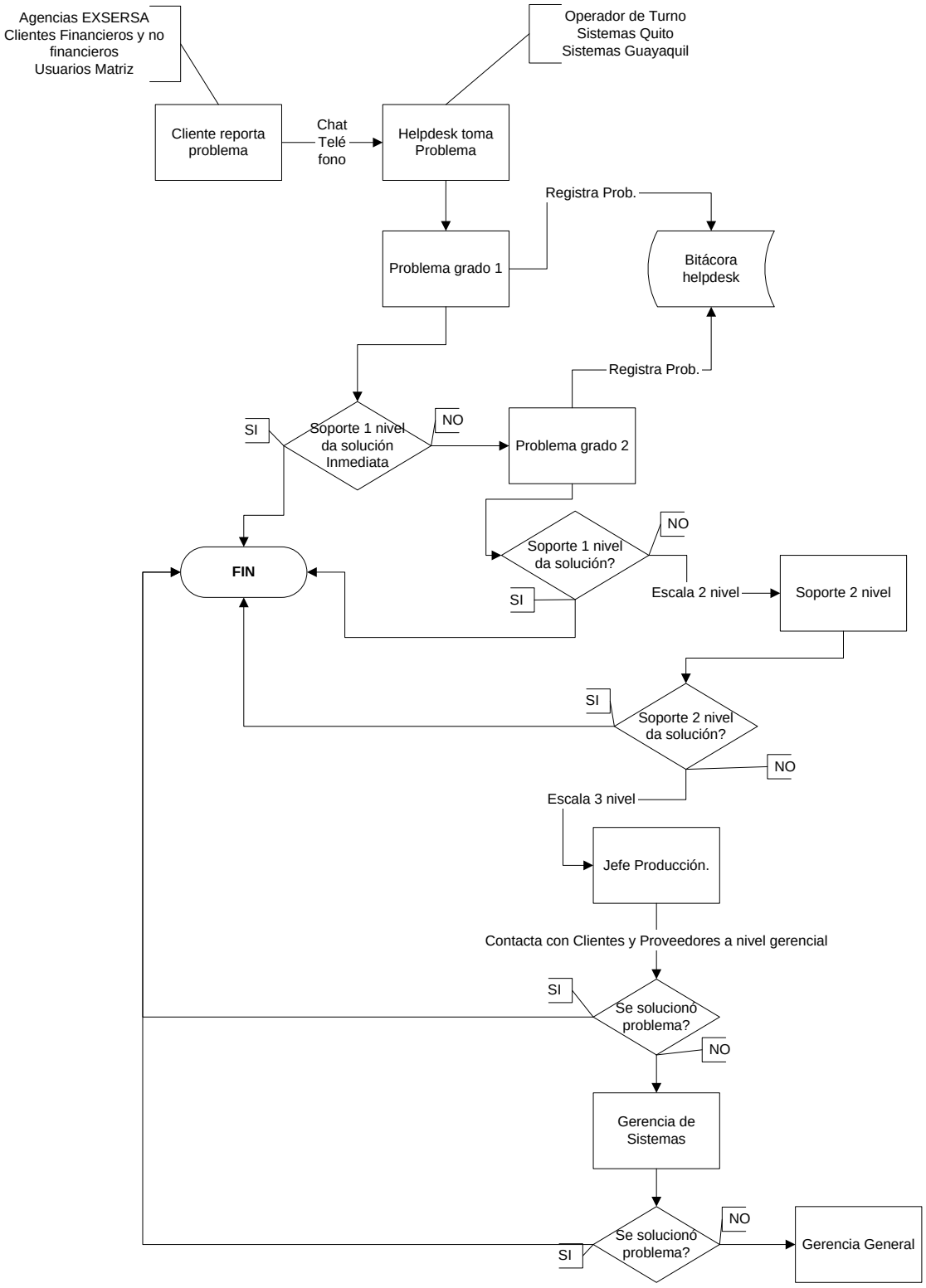

Figura 3.4: (Flujo de Soporte Existente en EXSERSA)

3.3.1.1.- Servicio técnico externo a matriz

Proceso: Recibir y validar requerimiento

Objetivo:

Permite al operador recopilar información necesaria sobre los problemas o incidentes reportados por los clientes externos a la matriz (Agencias).

Tabla 3.1: (Descripción de actividades para el proceso recibir y validar requerimiento)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Operador	Verifica problema	3
2	Operador	Califica problema (grado1, grado2)	2
3	Operador	Ingresar detalle de problema.	1

Formas: Ninguna

Políticas:

a. El operador recepta el problema vía telefónica, chat o por monitoreo permanente.
b. Ingresar los datos del problema en bitácora.

Controles: Ninguno

Proceso: Asignar técnico al problema

Objetivo:

Permite asignar o determinar a un técnico para dar solución al problema reportado por el cliente.

Tabla 3.2: (Descripción de actividades para el proceso asignar técnico al problema)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Operador	Asignar técnico en problemas grado1	2
2	Operador	Asignar técnico en problemas de grado2.	2
3	Operador	Registrar hora de asignación y técnico.	2

Formas: Ninguna

Políticas: Dependiendo del tipo de problema enrutarlo al técnico especifico.

Controles: Ninguno

Proceso: Solucionar Problema

Objetivo:

Permite dar solución a los problemas reportados por los clientes externos a la matriz.

Tabla 3.3 (Descripción de actividades para el proceso Solucionar Problema)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Técnico 1 Técnico 2	Evaluar problema Evaluar problema	5 10
2	Técnico 1 Técnico 2	Determinar solución Determinar solución	8 30
3	Técnico 1 Técnico 2	Determinar herramientas Determinar herramientas	2 10
4	Técnico 1 Técnico 2	Dar solución al problema Dar solución al problema	10 10

Formas: Ninguna

Políticas:

a. Reportar la solución dada al problema.
b. Si existe inconvenientes y no poder dar solución inmediata, escalar el problema a un nivel superior.

Controles: Ninguno

Proceso: Finalizar Solicitud De Soporte

Objetivo:

Registrar la solución dada a los problemas reportados por el cliente.

Tabla3.4 (Descripción de actividades para el proceso Cerrar Llamada)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Operador	Registrar fecha y hora de solución.	2
2	Operador	Registrar detalles de solución.	5

Formas: Ninguna

Políticas: Ninguna

Controles: Ninguno

3.3.1.2.-Servicio técnico interno a matriz

Proceso: Recibir y validar requerimiento

Objetivo:

Permite recopilar información necesaria sobre los problemas o incidentes reportados por los clientes internos de la matriz directamente al departamento de sistemas.

Tabla 3.5: (Descripción de actividades para el proceso recibir y validar requerimiento)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Técnico	Recibe problema	3
2	Técnico	Valida y verifica problema	5
3	Operador	Ingresar detalle de problema.	1

Formas: Ninguna

Políticas:

c. El Técnico recepta el problema vía telefónica o personalmente.

d. Ingresar los datos del problema.

Controles: Ninguno

Proceso: Asignar técnico al problema

Objetivo:

Permite asignar o determinar a un técnico para dar solución al problema reportado por el cliente.

Tabla 3.6: (Descripción de actividades para el proceso asignar técnico al problema)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Técnico	Asignar técnico en problemas grado1	2
2	Técnico	Asignar técnico en problemas de grado2.	2
3	Operador	Registrar hora de asignación y técnico.	1

Formas: Ninguna

Políticas: Dependiendo del tipo de problema enrutarlo al técnico especifico.

Controles: Ninguno

Proceso: Solucionar Problema

Objetivo:

Permite dar solución a los problemas reportados por los clientes externos a la matriz.

Tabla 3.7: (Descripción de actividades para el proceso Solucionar Problema)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Técnico	Evaluar daño	2
2	Técnico	Determinar solución	3
3	Técnico	Determinar herramientas	1
4	Técnico	Dar solución al problema	10

Formas: Ninguna

Políticas:

c. Reportar la solución dada al problema.

d. Si existe inconvenientes y no poder dar solución inmediata, escalar el problema a un nivel superior.

Controles: Ninguno

Proceso: Finalizar solicitud de soporte

Objetivo:

Registrar la solución dada a los problemas reportados por el cliente.

Tabla 3.8: (Descripción de actividades para el proceso Cerrar Llamada)

Ord.	Responsable	Actividad	Tiempo (min.)
1	Operador	Registrar fecha y hora de solución.	2
2	Operador	Registrar detalles de solución.	2

Formas: Ninguna

Políticas: Ninguna

Controles: Ninguno

3.3.2.- Proceso General Propuesto "EXSERSA"

En el anexo B se detalla el proceso propuesto una vez realizados los estudios respectivos.

Consideraciones Generales

En esta parte se resume el Proceso Operativo de Control, Manejo de Problemas y Atención de requerimientos, sus sub-procesos y la identificación de su alcance.

Los Sub-Procesos Operativos del Control y Manejo de Problemas y requerimientos son:

- Ingreso de requerimiento (atención telefónica, chat, radio y otros).
- Atención On Site (en el sitio).

- Atención por parte de Grupos de Solución.
- Escalamientos.
- Manejo de Incidente Crítico.
- Registro y medición de niveles de servicio.
- Acciones de mejoramiento continuo

Alcance

El alcance de Proceso Operativo fue definido de modo que provea la integración de los procedimientos, tecnologías y recursos humanos necesarios.

El Proceso comienza con:

a. La recepción de una solicitud del Usuario por parte de un responsable (Llamada al Helpdesk vía telefónica, envió de reporte por el chat, reporte por e-mail u otro recurso que permita presentar la solicitud para recibir soporte por un problema o requerir un servicio).
b. El diagnóstico de un técnico de EXSERSA sobre la ocurrencia de un problema, o,
c. La necesidad de un área por un requerimiento.

El Proceso termina con:

a. La obtención de la confirmación de la solución del Usuario para el cierre de una llamada resuelta en primer nivel o con soporte en sitio. En ambos casos el Responsable respectivo realiza una actividad de verificación del ticket para proceder a cerrarlo.

b. La confirmación del técnico asignado de EXSERSA sobre la solución del problema (en el caso de ser un problema diagnosticado por el técnico y no a través de una llamada telefónica del usuario).
c. Ingreso de los incidentes en la base de conocimientos.
d. El registro de indicadores y niveles de servicio. El input para acciones de mejoramiento continuo

El Proceso contiene:

a. Apertura del ticket (llamada, chat, e-mails, otros).
b. Resolución en 1er. nivel mediante atención telefónica.
c. Resolución en 2do. nivel de atención On-Site; cuando se trate de un problema que así lo requiera.
d. Direccionamiento de tickets para otras áreas cuando se haya diagnosticado e identificado el problema y su posible solución; así como los requerimientos que puedan suscitarse;
e. Reporte de la solución del problema o la atención del requerimiento o el escalamiento en función de los niveles de servicio.
f. Cierre del tickets.
g. Encuesta de satisfacción por muestreo.
h. Registro de indicadores.
i. Reportes predefinidos.

El Proceso de Control, Manejo de Problemas y Atención de requerimientos no contiene:

- Control de Cambios;
- Control y Seguimiento de Activos;
- Ejecución de soluciones definitivas a un problema.

Objetivos

Los objetivos del Proceso Operativo de Control, Manejo de Problemas y Atención de requerimientos son:

a. Analizar y solucionar problemas que afecten las operaciones de los clientes internos y externos.
b. Direccionar pedidos de servicios siempre que sean solicitados por el cliente interno (agencias, áreas funcionales, etc.) o externo según sea la necesidad.
c. Maximizar la resolución de llamadas (tickets) en el 1er. Nivel de Atención Telefónico.
d. Identificar y resolver los problemas antes que ellos impacten los servicios de la compañía.

Roles y Responsabilidades

Esta sección describe roles y responsabilidades del HD y de los usuarios del mismo en el soporte del Proceso Operativo de Control, Manejo de Problemas y Atención de requerimientos.

Define también los roles involucrados en el proceso así como también sus atribuciones.

a. Estos roles pueden ser ejecutados por una sola persona o por varias individualmente.
b. El Rol incluye responsabilidades, sin embargo no se limita a aquellas descritas en el mismo.
c. Las responsabilidades descritas en un rol pueden no incluir todas las actividades del Proceso de Control, Manejo de Problemas y Atención de requerimientos.

Roles:

Se observa la participación de los siguientes roles:

Cliente Interno: todos los usuarios de la compañía, subclasificados así:

a. Gerencias: Gerencia General y Gerencias funcionales
b. Jefaturas Departamentales y de Sucursales**:** Todos los jefes de cada gerencia funcional, el jefe de agencias

(Los dos niveles pueden solicitar solución de problemas y atención de requerimientos).

c. Agencias: Supervisores y Cajeros principales.
d. Personal administrativo

(Estos dos niveles pueden solicitar solución de problemas directamente y por intermedio de sus niveles definidos la atención de requerimientos)

Tabla 3.9: (Niveles definidos para atención de requerimientos)

FUNCIÓN	*SOLUCIÓN PROBLEMA*	*ATENCIÓN REQUERIM (1)*	*ATENCIÓN REQUERIM (2)*
Gerencias	***X***	***X***	
Jefaturas	***X***		***X***
Agencias	***X***		
Personal Administ.	***X***		

Helpdesk (HD): El punto de recepción, solución de 1er. Nivel y único punto de ingreso y control de solicitudes de problemas o atención de requerimientos

a. Responsable del HD: Coordinador del servicio de HD, con asiento en Quito.
b. Técnico del HD: Recurso asignado al manejo operativo del HD, con asiento en Quito.
c. Técnico de solución en sitio

Áreas de solución: Todas las áreas a las que se les asignan la solución de un problema o la atención de un requerimiento

a. **Operaciones:** Solución de problemas operativos o actividades vinculadas con la administración de agencias
b. **Tecnología:** Solución de problemas tecnológicos o la entrega de servicios vinculados a la tecnología (preparación, entrega, instalación, adición de equipos tecnológicos, administración de aplicaciones, proyectos)
c. **Servicios generales:** Materiales, suministros, equipos, herramientas requeridas por el cliente interno

d. **Gestión de Calidad:** Solución de problemas que afecten la calidad de servicio de las agencias o al cliente interno y externo de la compañía

Coordinador de mejoramiento continuo: El área de Gestión de Calidad, quien lidera la prestación del servicio, la recolección de la data, el análisis preliminar y la determinación de las necesidades de mejoramiento de procesos a partir de las estadísticas del HD.

Grupos de mejoramiento continuo: Todas las áreas involucradas en los equipos de mejoramiento continuo

Responsabilidades

Cliente Interno

a. Seguir los procedimientos de uso de servicios del Centro de Servicio al Cliente (HD).
b. Anotar mensajes de errores y de alertas recibidas.
c. Anotar el número de registro de la llamada.
d. Ejecutar las pruebas que fueran solicitadas por Centro de Servicio al Cliente (HD).
e. Responder las preguntas realizadas por el Centro de Servicio al Cliente (HD).

Help Desk

Responsable del HD

a. Conocer el procedimiento del proceso de control, manejo de problemas y atención de requerimientos.

b. Dirigir la atención del H D.
c. Garantizar el registro en la herramienta, de todos los problemas recibidos.
d. Organizar los turnos normales, horario extendido y de fin de semana.
e. Administrar la carga de trabajo de los técnicos del HD.
f. Evaluar el desempeño y actitud del personal a cargo.
g. Analizar reportes de la productividad de los agentes técnicos del HD y analizar indicadores de desempeño de la atención telefónica
h. Monitorear los tickets reasignados en las áreas involucradas en el proceso.
i. Identificar, evaluar y comunicar acciones para la mejora del servicio, problemas potenciales, tipos de problemas recurrentes.
j. Apoyar la identificación y desarrollo de planes de contingencia y continuidad del servicio
k. Cumplir los niveles de servicio, incluso utilizando planes de contingencia y continuidad de servicio
l. Aplicar escalamientos cuando la situación lo demande y dentro de los niveles de coordinación.
m. Reconocer y notificar situaciones críticas en función del manual de contingencias
n. Negociar recursos con otras áreas de la compañía para el manejo de situaciones críticas.
o. Tomar acción para minimizar el flujo de solicitudes de solución de problemas.

p. Establecer los requerimientos de entrenamiento del personal a cargo en coordinación con Recursos Humanos.
q. Estandarizar el proceso de solución de problemas a nivel nacional.

Técnico del HD

a. Conocer el procedimiento del proceso de control, manejo de problemas y atención de requerimientos.
b. Responsable por realizar la primera atención, entender, resolver y/o direccionar las solicitudes de usuario como se ha definido en el procedimiento/proceso.
c. Atender las llamadas conforme los patrones de calidad definidos.
d. Revisar el chat con una frecuencia de 2 minutos.
e. Revisar los e-mails por comprobación para agencias (previa llamada telefónica)
f. Revisar los e-mails para receptar solicitudes de requerimientos con una frecuencia de 5 minutos.
g. Registrar las llamadas, las solicitudes por chat o e-mail en el sistema con los datos e información definida en los procesos y procedimientos del proceso de control, manejo de problemas y atención de requerimientos.
h. Aperturar el ticket de solución de problema
i. Consultar al coordinador para soporte sobre cualquier duda del proceso y técnicas que no estén descritas en los procedimientos de operación.

j. Hacer uso de las herramientas de soporte para proveer soluciones para los casos reportados por los clientes (árbol problema – solución /Base de Conocimientos).
k. Direccionar las llamadas para los niveles de soporte de acuerdo a los niveles de servicio acordados con las diferentes áreas (dirigido al líder del equipo de solución o su delegado/s)
l. Entregar el número de ticket al usuario, así como el tiempo de solución estandarizada.
m. Trabajar conforme al calendario definido.
n. Monitorear las colas de trabajo definidas en el proceso así como los tiempos de respuesta entregados al usuario.
o. Estar siempre logeado a los sistemas y herramientas definidas para la atención.
p. Actuar en servicios diversos delegados por el coordinador, para la mejora de la técnica ó necesidades comprometidas con la operación.
q. Contactar al usuario para obtener información adicional de ser necesario.
r. Manejo de situaciones de reclamos de clientes, escalar el problema a su coordinador de ser necesario.
s. Identificar, evaluar y comunicar acciones para la mejora del servicio, problemas potenciales, tipos de problemas recurrentes.

CAPITULO IV

ANÁLISIS Y ESPECIFICACIÓN DE REQUERIMIENTOS

4.1.- Generalidades

En el nuevo milenio, la humanidad asiste a profundas transformaciones contextuales cuyas implicaciones, antes que presentarse como mutaciones de carácter evolutivo, representan saltos y rompimientos, que en la mayoría de los casos conllevan a la transformación de los viejos paradigmas y la sustitución abrupta.

Uno de los más significativos lo constituye la tecnología, y dentro de ella, sin duda alguna, la gerencia informática, electrónica, telecomunicaciones, y desarrollo de la Ingeniería de Software (IS) que juega un papel importante en las organizaciones.

Se avanza hacia una nueva cultura: La cultura de la información, y en consecuencia un nuevo tipo de sociedad. Organizaciones que para su desarrollo informático requieren profesionales de las distintas áreas del conocimiento, con capacidad de planear, diseñar, implementar, y liderar proyectos, que apliquen técnicas modernas de la IS, en sus diversos campos, que permitan asegurar su calidad, con base en estándares reconocidos mundialmente.

En las organizaciones actuales, se ve la necesidad de una generación de sistemas de información que se caracterizan por la flexibilidad para integrar usuarios, sistemas y datos existentes, de manera tal que permita preservar la inversión realizada y habilite la cooperación entre ellos y los nuevos sistemas de información.

4.2.- Especificación de requerimientos

El sistema de control y soporte técnico (Hepldesk), deberá tener la capacidad de mantener un registro de los pedidos (incidentes) sean estos problemas, requerimientos o consultas que se dan en cada una de las diferentes áreas a nivel interno, de agencias y proveedores a nivel externo, estos incidentes deben ser registrados por un operador del sistema, él mismo que interactúa con el sistema en busca del problema y las posibles soluciones en caso de existirlas, en caso de no existir alimentada la base de datos con la información requerida el sistema debe tener la facultad de permitir al usuario enrutar o asignar a un técnico previamente existente en la base de datos el mismo que de solución al incidente, ésta asignación se la realizará por mail que especificara el pedido reportado, así como la importancia del mismo.

Cada uno de los técnicos asignados debe reportar los tiempos de que se necesitaron para la solución y codificar la solución dada, esta información se registrará cerrando de esa forma el ticket abierto anteriormente.

El sistema debe ser capaz de alimentar la base de datos problema – solución, manteniendo de esa forma actualizada la información para uso del HD, evitando así que pedidos similares sean enrutados nuevamente a los técnicos / administradores.

El ingreso de pedidos se lo realizara a nivel nacional (Agencias Quito, Guayaquil, sucursales de las dos regiones y Provincias), por lo que la información deberá estar centralizada y actualizada para emitir reportes de pedidos categorizados o por fecha.

4.2.1.- Función del sistema

Llevar un control de pedidos (incidentes) reportados por usuarios internos y externos a la matriz de EXSERSA, generar una base de conocimientos de problema – solución, llevar una bitácora de tiempo invertido por cada uno de los técnicos en la solución de problemas, así como permitir también la emisión de reportes.

4.2.2.- Información básica que se maneja.

PEDIDOS	• Código Pedido(Secuencial) o Ticket • Descripción de pedido • Fecha / Hora
REPORTE DEL PEDIDO	• Código Pedido • Fecha inicial • Fecha inicial • Hora de inicio pedido • Hora de finalización • Hora de Salida • Problema • Solución
CATEGORIZACION DE PROBLEMAS	• Código categoría • Descripción categoría • Código subelemento • Descripción subelemento • Código elemento • Descripción elemento • Código problema • Descripción problema
MANEJO DE SOLUCIONES	• Código de Solución • Descripción solución
USUARIO	• Código Usuario

	• Nombre • Dirección • Teléfono
TECNICO	• Código técnico • Nombre • Categoría • Área • Teléfono • E-mail

4.3.- Antecedentes

Al 2004/12/05 no existe en producción un sistema para registro, enrutamiento y control de requerimientos técnicos (requerimientos efectuados por las agencias y usuarios internos), pues actualmente se lo lleva en forma manual.

La función específica que se realiza en el área de Helpdesk (HD) consiste en recopilar los datos de los problemas que se presentan en cada de una de las agencias ya sean estos de tipo software o hardware, asignar un técnico y llevar un control de los tiempos de respuesta y solución, mantenimiento de los técnicos, clientes, equipos y proveedores de servicios externos.

Dada la necesidad se propone el desarrollo de una aplicación, con la capacidad de dar apoyo a las actividades de asistencia técnica, asignación, control y supervisión de los problemas que se presentan tanto con clientes internos y externos. Todo lo que se menciona enmarcado dentro de los principios de IS.

4.4.- Justificación

Actualmente el área de HD no cuenta con un sistema de información que brinde el apoyo necesario para realizar el registro, asignación, supervisión de los

problemas y requerimientos de clientes internos (Usuarios de Matriz, Agencias a nivel país) y externos de la empresa.

Según conversaciones con usuarios finales se detecta las siguientes deficiencias del sistema manual que se lleva actualmente.

- Tiempos de respuesta (solución a los problemas) muy altos o en algunos casos negativos.
- Carencia de Reportes.
- No cumple con todos los requerimientos del área de soporte.

Cabe mencionar que el área de soporte se encuentra en las ciudades de Quito y Guayaquil.

4.5.- Descripción del Problema

EXSERSA, una red transaccional confiable y estable, afianzada en una estructura tecnológica avanzada que brinda soluciones a nivel de servicio de cobranzas y pagos a nivel nacional, con la idea de brindar mejor servicio, aumentar la calidad de sus productos, mejorar el tiempo de respuesta de los técnicos asignados para resolver los problemas internos y de sus clientes ha considerado el desarrollo de un sistema que permita apoyar al control de los servicios que presta el personal técnico de sistemas tanto de la matriz ubicada en Quito como de la sucursal técnica de Guayaquil y sus clientes (internos y externos) por medio de portales de acceso.

Teniendo en cuenta que la información es parte esencial para las empresas, el software que se desarrollará brindará a los niveles que lo requieran información veraz, oportuna y confiable.

4.6.- Objetivos

4.6.1.- Objetivo general

Implementar un sistema de información que preste la ayuda necesaria para llevar automatizado el soporte y control de servicios técnicos en EXSERSA, el mismo que brinde el apoyo necesario a los niveles gerenciales en la toma de decisiones y facilite a los usuarios involucrados en el sistema la realización de los procesos diarios.

4.6.2.- Objetivos específicos

a. Desarrollar un sistema que facilite la administración y el manejo óptimo de la información del área técnica.
b. Dar soluciones efectivas, eficientes y rápidas, satisfaciendo así los requerimientos de los usuarios.
c. Brindar facilidades a usuarios de agencias para que puedan exponer requerimientos técnicos.
d. Facilitar la personalización y configuración del sistema para emitir reportes, estadísticas y procesos de escalamiento, mediante una interfaz amigable y fácil de manipular.
e. Tener una base centralizada que sirva como fuente de contacto para usuarios y clientes.

4.7.- Alcance

HDS_System debe ser capaz de receptar los pedidos generados para las diferentes áreas, agencias, proveedores de la empresa, llevar un control de los

tiempos de atención o respuesta de los técnicos asignados, para que el personal correspondiente tomo los correctivos que sean necesario.

La funcionalidad de HDS_System para los módulos en el área de red LAN comprenderá la siguiente:

- Administración y control de los pedidos reportados por clientes internos y externos
- Administración de usuarios del sistema, tiempos de atención y notificación
- Creación de reportes y estadísticas
- Seguimiento de los incidentes reportados a niveles de mayor jerarquía.
- Asignación, generación de correos electrónicos a técnicos y niveles gerenciales dependiendo de la complejidad de los pedidos.
- Creación de una base de conocimientos alimentada con la información de los problemas y soluciones dadas a cada uno de los pedidos.

4.8.- Limitaciones

Dentro del ámbito del HDS_System no se considera:

- Realización de auditorias de las transacciones realizadas por los usuarios inmersos en el sistema
- No existirá Consultas vía Web.
- No tendrá integración directa con aplicaciones como Exchange Server para generar un Workflow, mas si utilizará Exchange como medio de envió de correo a los técnicos que se asigne.
- Uso exclusivo de herramientas Microsoft

4.9.- Enfoque del sistema de software

El software controlará los pedidos generados por los clientes, el técnico asignado y la solución aplicada a los mimos, así como dará a los niveles que lo requiera la facilidad de solicitar reportes de pedidos.

La Figura 4.1 permite tener una visión general del sistema, en el que se visualiza los usuarios del sistema y el flujo de la información.

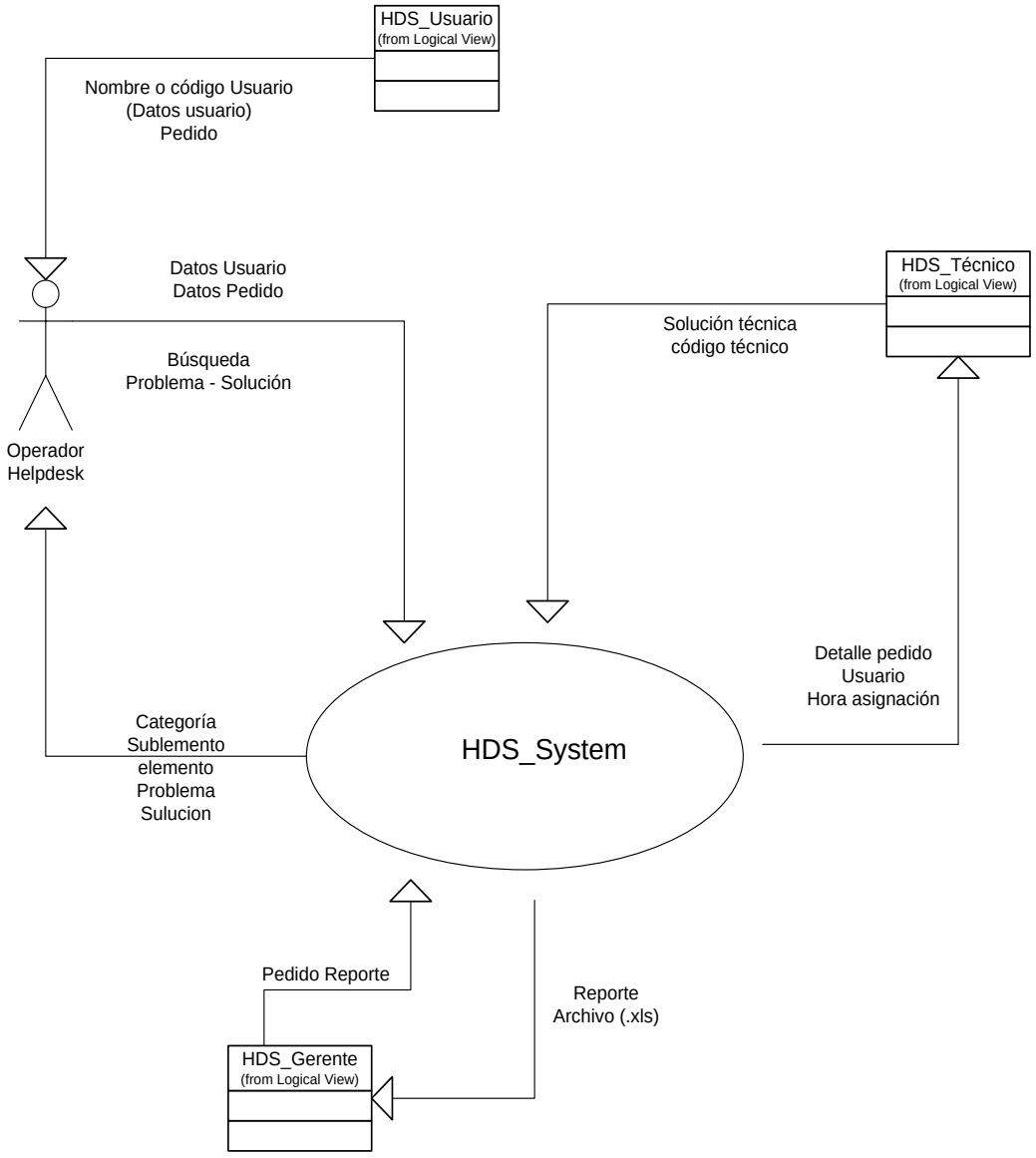

Figura 4.1: (Concepción general del sistema)

4.10.- Funciones del producto de software

- Ingreso y Mantenimiento de :
  - Pedidos generados por los clientes.
  - Usuarios del sistema.
  - Problemas y soluciones previa estandarización de los técnicos.
- Generación de reportes y estadísticas
- Envió de correos generando la asignación de los pedidos a los técnicos correspondientes.
- Interfases con la base de datos y aplicaciones en caso de requerirse.
- Generar una base de conocimientos que involucre una o varias soluciones a cada uno de los problemas.

4.11.- Especificaciones de los usuarios

Tipos de usuarios:

- Operador / Helpdesk

  Quien recibirá los pedidos generados por los clientes, dará solución a los pedidos que existan en la base de conocimientos, así como también quien coordina la asignación de lo mismos a los técnicos respectivos, además cierra los pedidos.

- Administrador.

  Tiene la facultad de administrar los usuarios y tener acceso total al sistema.

- Técnico

  Tiene la facultad de generar la información concerniente a las soluciones de los pedidos asignados, categorizarlas y registrar las mismas en la base de conocimientos.

- Gerente

  Abarcará expresamente a poder emitir reportes y estadísticas de los pedidos.

Todos los usuarios tienen la misma interfase con los permisos asignados a cada uno de ellos.

4.12.- Requerimientos de hardware y software

- **Servidor de Base de Datos**

  El servidor ubicado en la ciudad de Quito en donde estara situada la base de datos, con el siguiente hardware y software instalado.

  - Procesador Pentium IV de 2.2 Ghz
  - Memoria RAM 256 Mb
  - 1 disco duro 40 Gb
  - NIC de 100 Mbps
  - Windows NT Server / Windows 2000 Server
  - Microsoft SQL Server 2000

- **Estaciones de Trabajo**

  Las estaciones de trabajo deben tener como características básicas las siguientes.

- Pentium I de 266 Mhz
- Memoria Ram 64 Mb
- Disco duro de 4 Gb
- NIC 10/100 Mbps
- Windows 95, 98, 2000, Xp, etc.
- ODBC 32
- Microsoft Office 97 (Excel, Microsoft Outlook Express)

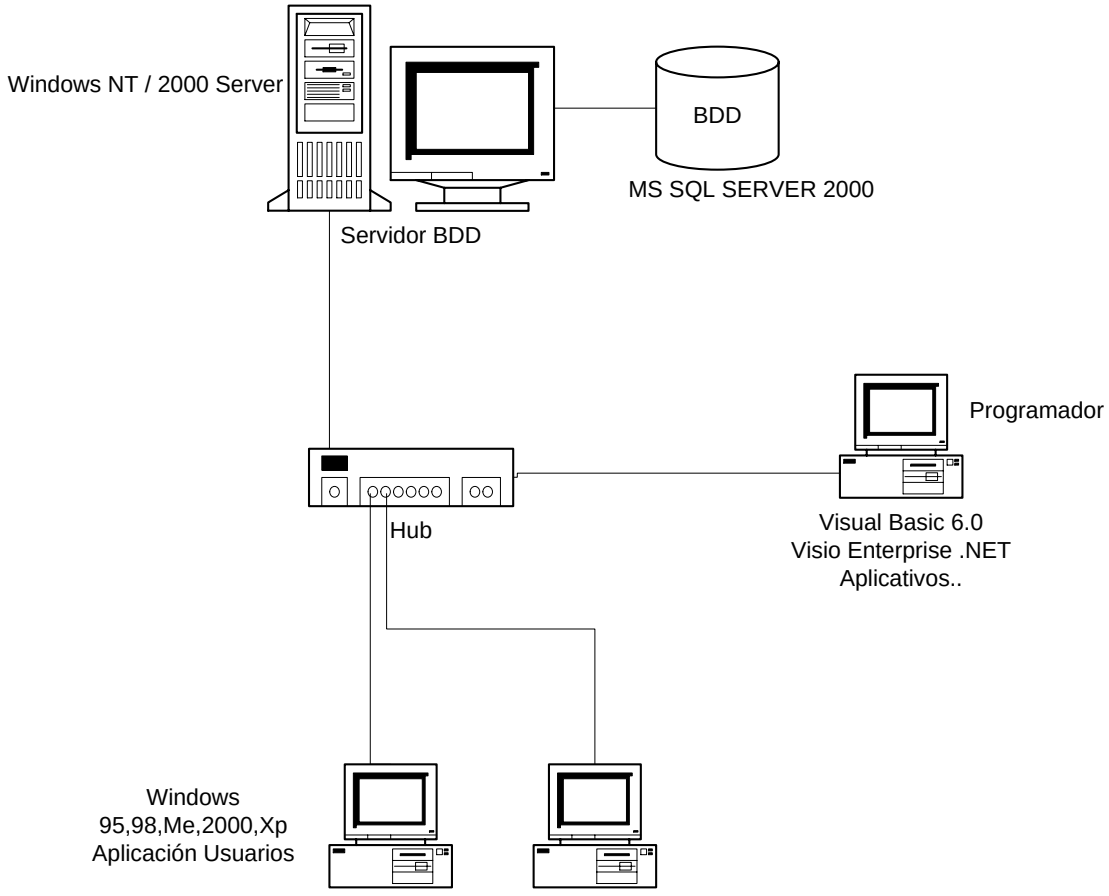

Figura 4.2: (Arquitectura Solución SQL)

4.13.- Modelo Funcional

4.13.1.- Casos de Uso

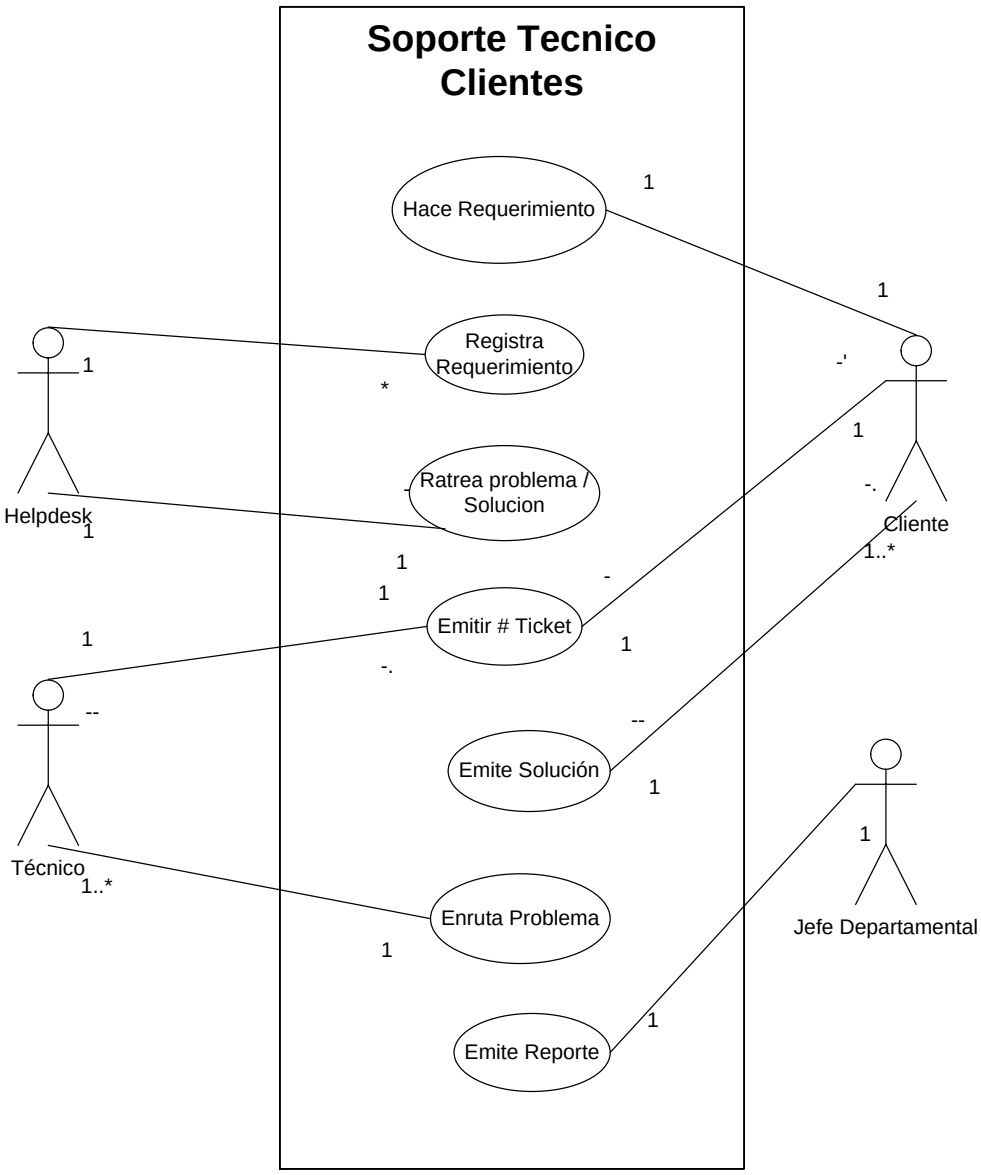

Figura 4.3: (Caso de Uso Soporte Técnico General)

4.13.1.1.- Caso de Uso para Soporte Técnico Clientes Internos

Tabla 4.1: (Caso de Uso Soporte a clientes internos)

ACTORES	
Primarios	• Operador / Helpdesk • Cliente Matriz • Cliente Agencias (Supervisores)
Secundarios	• Técnicos / Sistemas
Material Externo	• Teléfono
Otros Sistemas	• Chat, Mail, Herramientas de monitoreo.
CONDICIÓN DE ENTRADA	• Reporte de incidentes.
FLUJO DE EVENTOS	• Operador recibe incidentes • Operador registra incidente • Sistema emite # de ticket (Seguimiento) • Operador busca problema y posible solución en base de conocimientos • Operador emite solución o estado en el que se encuentra el requerimiento de cliente.

	• Operador enruta requerimiento a Técnico. • Técnico busca solución a requerimiento • Técnico emite solución y almacena nueva solución y/o problema. • Operador emite solución al cliente. • Operador cierra ticket. • Sistema almacena requerimiento en base de datos.
CONDICIÓN DE SALIDA	• Numero de ticket generado por el sistema. • Solución al incidente reportado.

4.13.1.2.- Caso de Uso para Soporte Técnico Clientes Externos

Tabla 4.2: (Caso de Uso Soporte a clientes externos)

ACTORES	
Primarios	• Operador / Helpdesk • Instituciones Financieras y No financieras

Secundarios	• Técnicos / Sistemas
Material Externo	• Teléfono
Otros Sistemas	• Mail, Chat, Herramientas de monitoreo
CONDICIÓN DE ENTRADA	• Reporte de incidentes.
FLUJO DE EVENTOS	• Operador recibe incidentes • Operador registra incidente • Sistema emite # de ticket (Seguimiento) • Operador busca problema y posible solución en base de conocimientos • Operador emite solución o estado en el que se encuentra el requerimiento de cliente. • Operador cierra ticket. • Sistema almacena requerimiento en base de datos.
CONDICIÓN DE SALIDA	• Numero de ticket generador por el sistema. • Solución al incidente reportado.

4.14.- Modelo de Objetos

4.14.1.- Modelo Conceptual

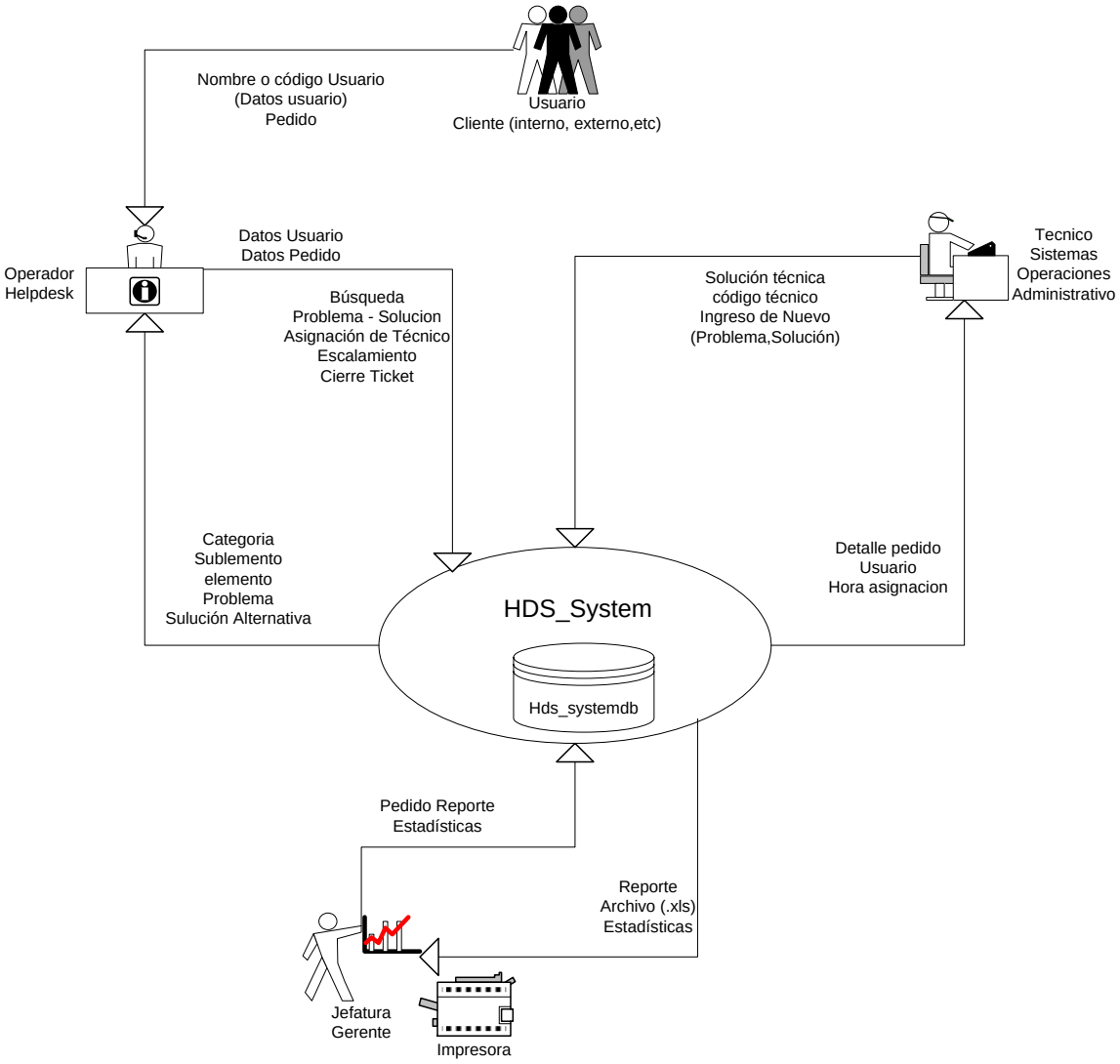

Figura 4.4: (Diagrama conceptual)

4.14.2.- Diagrama de Clases

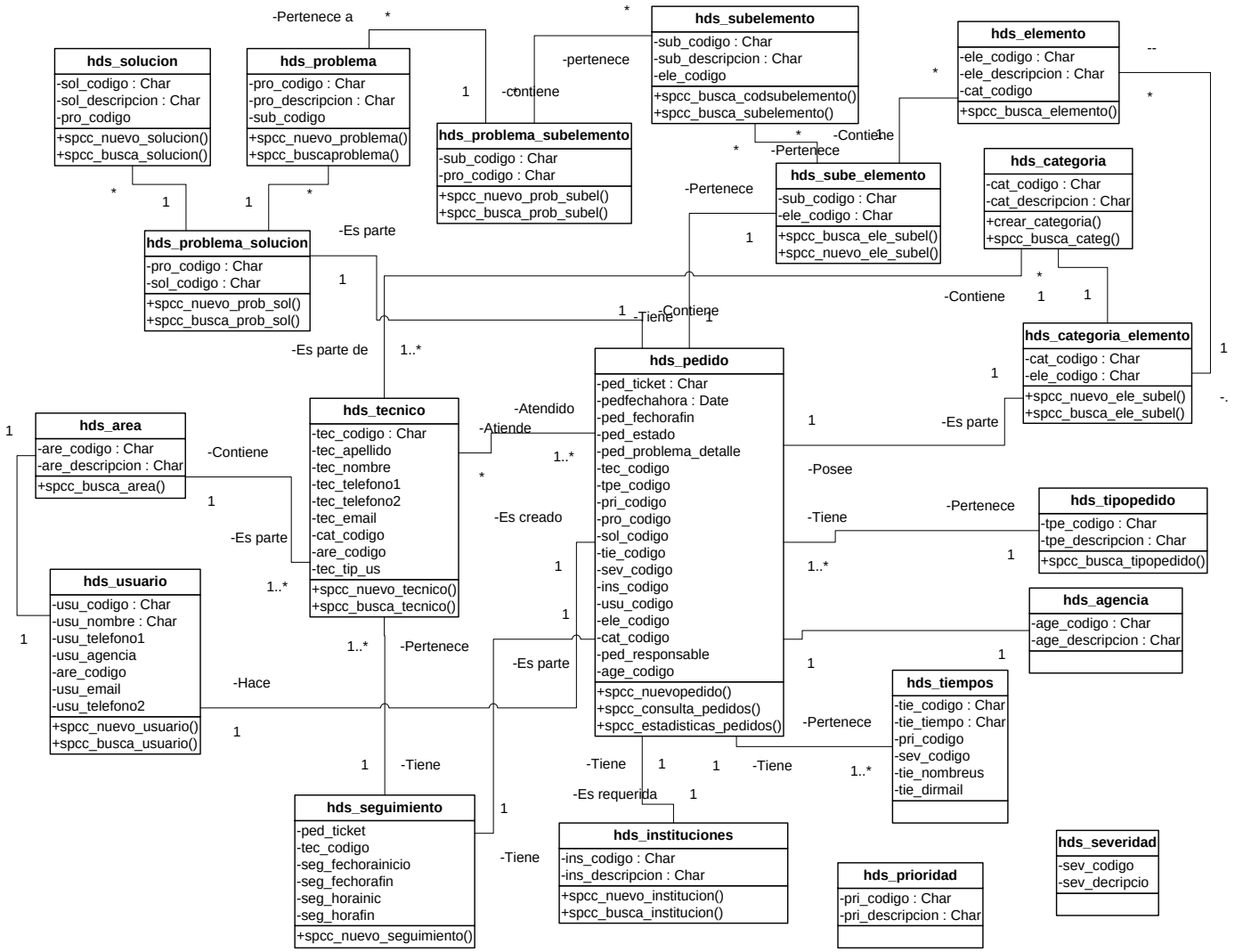

Figura 4.5: (Diagrama de Clases)

4.15.- Modelo Dinámico

4.15.1.- Diagrama de Secuencias

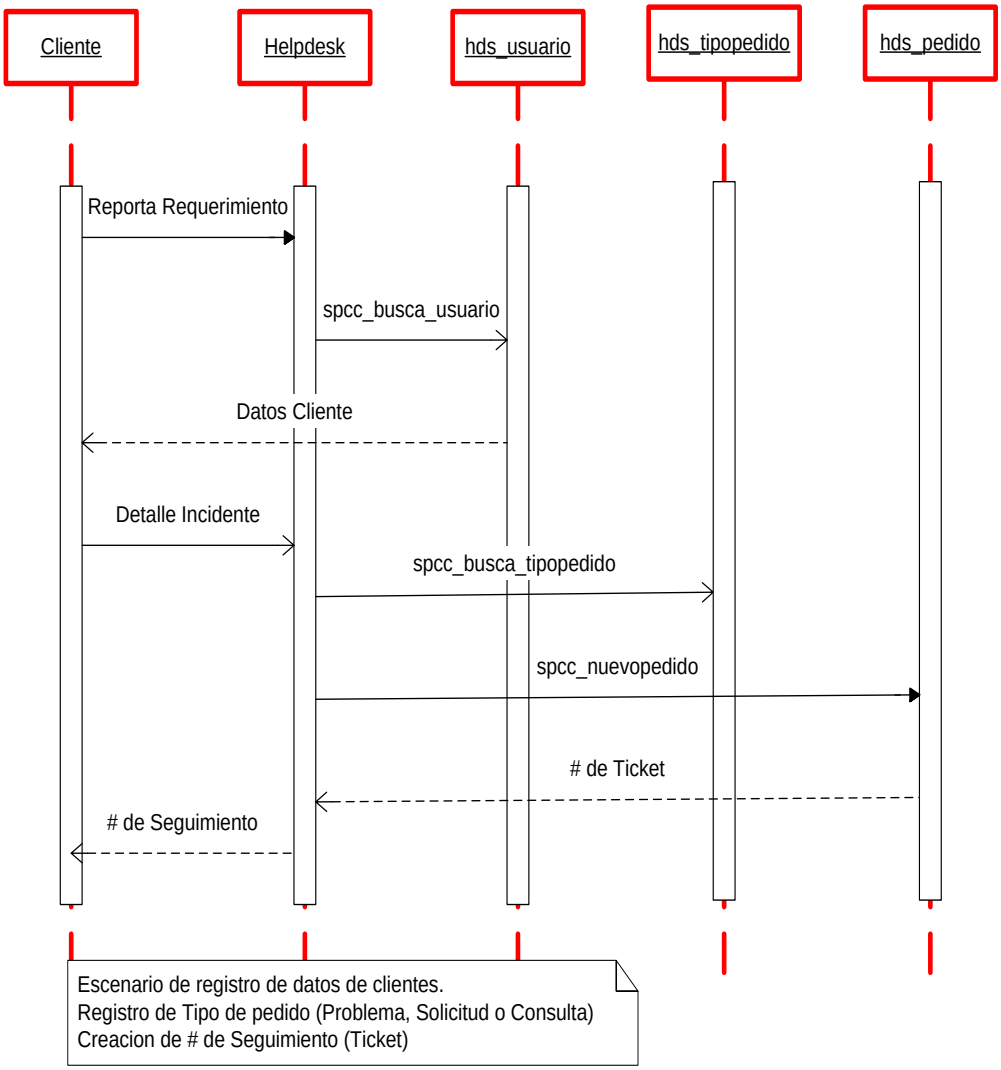

Figura 4.6: (Escenario Ingreso de Incidentes, creación de # Tickets, seguimiento)

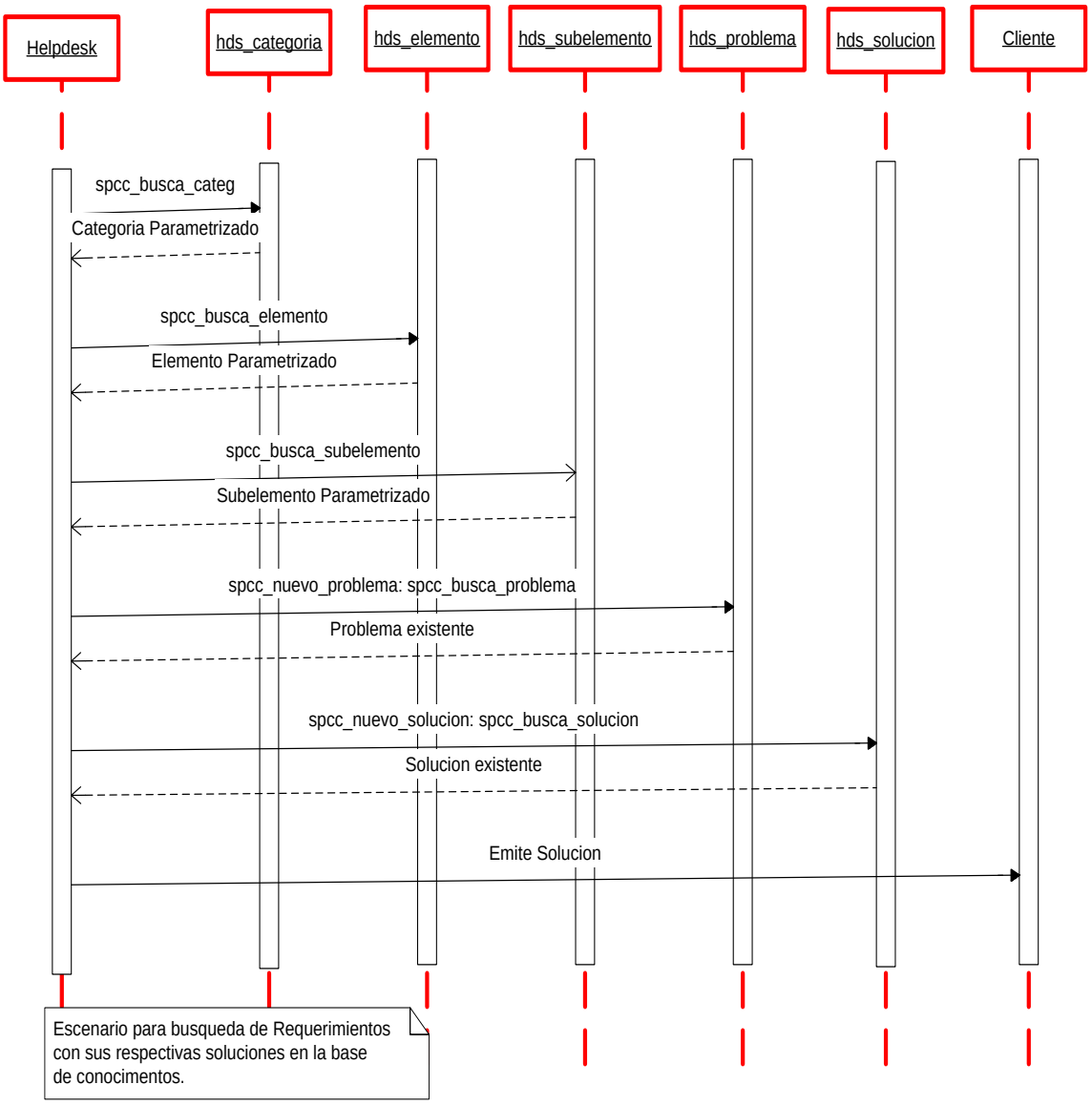

Figura 4.7: (Escenario Búsqueda de Incidentes, posibles soluciones)

4.15.2.- Diagrama de Colaboración

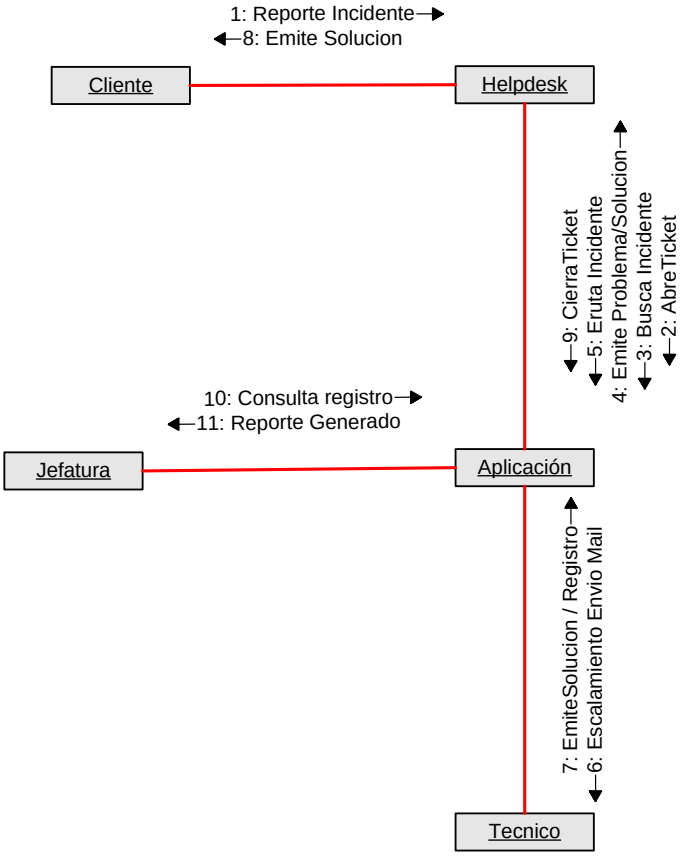

Figura 4.8: (Diagrama de Colaboración)

CAPITULO V

DISEÑO

5.1.- Modelo de clases

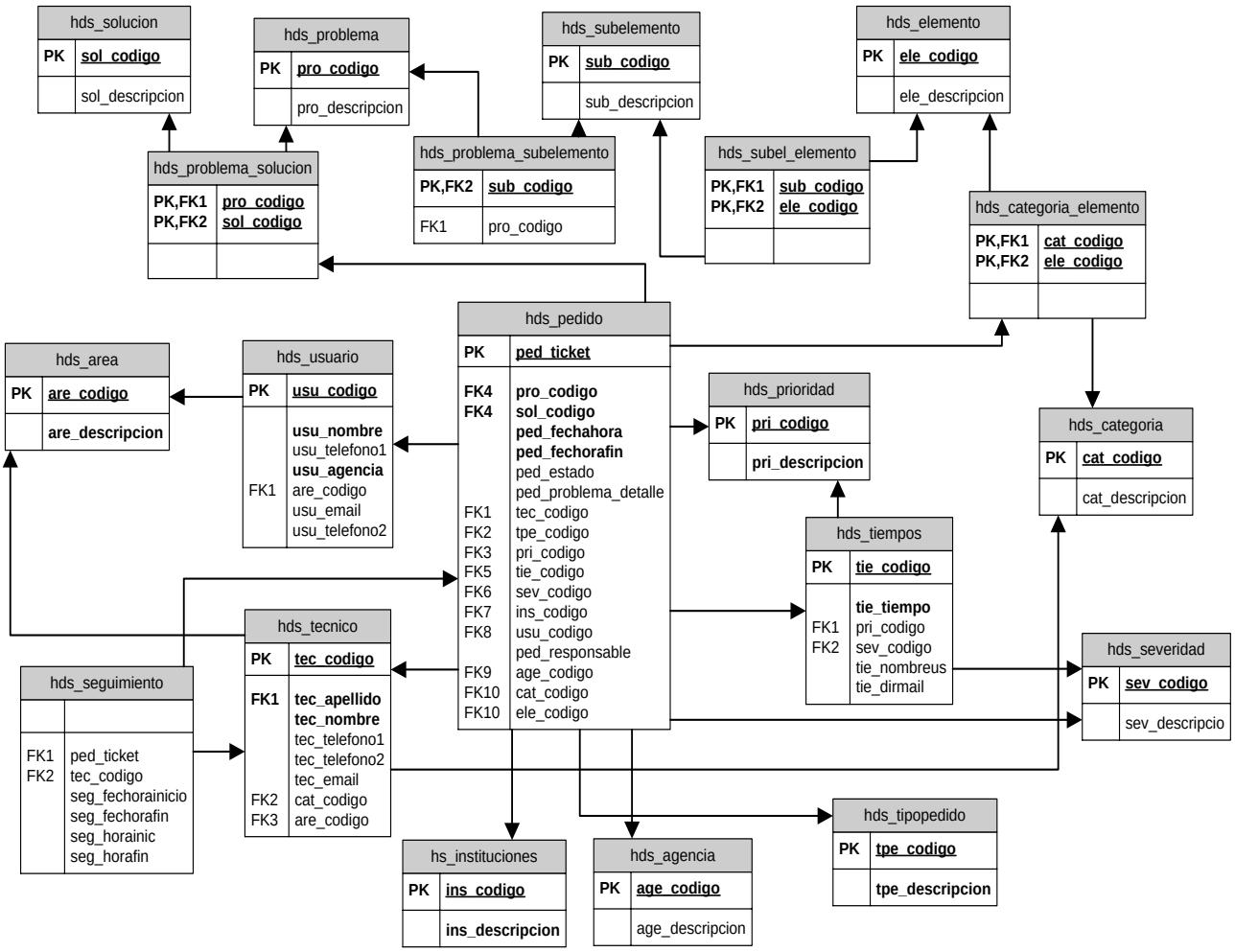

Figura 5.1: (Diagrama del Modelo de Clases)

5.2- Diseño de la Base de Datos y Generación de Scripts

5.2.1.- Resumen de Tablas

Target DBMS: Microsoft SQL Server
Número de tablas: 20
Número de vistas: 0
Número de columnas: 75
Número de foreign keys: 26
Last build date: Not built

Atributos extendidos:

Filegroup PRIMARY

Nombre de Tabla	# de Columnas	# de Claves foraneas	Descripción
hds_agencia	2	0	Contiene el código y descripción de cada una de las agencias existentes.
hds_area	2	0	Contiene el código y descripción de las areas involucradas en el sistema: Sistemas, operaciones, etc.
hds_categoria	2	0	Tabla que almacena las diferentes categorizaciones para los requerimientos ejemplo: Aplicaciones, Redes, Base de datos, etc.
hds_categoria_elemento	2	2	Tabla que permite romper la relación de muchos a muchos entre categorias y elementos.
hds_elemento	2	0	Almacena los diferentes elementos relacionados con cada una de las categorias: Comunicaciones, infraestructura, etc
hds_pedido	17	10	Tabla Principal que registra información básica para cada uno de los requerimientos.
hds_prioridad	2	0	Almacena las prioridades definidas en el estudio de procesos.

hds_problema	2	0	Registra los problemas que ingresan a la base de conocimientos.
hds_problema_solucion	2	2	Tabla que permite romper la relación de muchos a muchos entre problema y solución.
hds_problema_subelemento	2	2	Tabla que permite romper la relación de muchos a muchos entre problema y subelemento.
hds_seguimiento	7	2	Tabla que almacena información sobre cada incidente como son número de ticket, hora de incio y hora de finalización del pedido así como tambien que técnico resolvió dicho pedido.
hds_severidad	2	0	
hds_solucion	2	0	Registra las soluciones generadas para dar posibles alternativas a cada uno de los problemas existentes en la base de conocimientos.
hds_subel_elemento	2	2	Tabla que permite romper la relación de muchos a muchos entre subelemento y elemento.
hds_subelemento	2	0	Tabla de subelementos para las categorias.
hds_tecnico	8	3	Almacena información básica de cada uno de los técnicos de las diferentes areas y categorias.
hds_tiempos	6	2	
hds_tipopedido	2	0	Contiene el código y descripción de los diferentes tipos de pedido: Problema, consulta, etc.
hds_usuario	7	1	Almacena información de cada uno de los usuarios en este caso son las posibles personas que reportan o hacen pedidos.
hs_instituciones	2	0	Tabla que almacena información de cada una de las insituciones que participan en el reporte de pedidos.

5.2.2.- Descripción de Tablas

hds_agencia

Descripción: Contiene el código y descripción de cada una de las agencias existentes.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
age_codigo	char(3)	No permitido	
age_descripcion	char(40)	Permitido	

Claves foraneas	Hijo	Padre
hds_agencia_hds_pedido_FK1	hds_pedido.age_codigo	age_codigo

hds_area

Descripción: Contiene el código y descripción de las areas involucradas en el sistema: Sistemas, operaciones, etc.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
are_codigo	char(4)	No permitido	Almacena el código de área.
are_descripcion	char(40)	No permitido	Almacena la descripción de cada área.

Claves foraneas	Hijo	Padre
hds_area_hds_usuario_FK1	hds_usuario.are_codigo	are_codigo
hds_area_hds_tecnico_FK1	hds_tecnico.are_codigo	are_codigo

hds_categoria

Descripción: Tabla que almacena las diferentes categorizaciones para los requerimientos ejemplo: Aplicaciones, Redes, Base de datos, etc.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
cat_codigo	char(4)	No permitido	Almacena el código de cada categoría.
cat_descripcion	char(50)	Permitido	Almacena la descripción de cada categoría.

Claves foraneas	Hijo	Padre
hds_categoria_hds_tecnico_FK1	hds_tecnico.cat_codigo	cat_codigo
hds_categoria_hds_pedido_FK1	hds_categoria_elemento.cat_codigo	cat_codigo

hds_categoria_elemento

Descripción: Tabla que permite romper la relación de muchos a muchos entre categorías y elementos.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 2

Códigos: 0

Tipo: Tabla

Columnas	Tipo de dato	Permite Nulos	Observaciones
cat_codigo (FK)	char(4)	No permitido	Almacena el código de cada categoría relacionada con el elemento.
ele_codigo (FK)	char(10)	No permitido	Almacena el código de elemento relacionado con cada categoría.

Claves foraneas	Hijo	Padre
hds_categoria_hds_pedido_FK1	cat_codigo	hds_categoria.cat_codigo
hds_elemento_hds_categoria_elemento_FK1	ele_codigo	hds_elemento.ele_codigo
hds_categoria_elemento_hds_pedido_FK1	hds_pedido.cat_codigo hds_pedido.ele_codigo	cat_codigo ele_codigo

Detalles de clave foranea (hijo)

hds_categoria_hds_pedido_FK1

Hijo	Padre
cat_codigo	hds_categoria.cat_codigo

Tipo de relación: Identifying

Cardinalidad:	One -to- Zero-or-More
Permite Nulos:	No permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_elemento_hds_categoria_elemento_FK1

Hijo	**Padre**
ele_codigo	hds_elemento.ele_codigo

Tipo de relación:	Identifying
Cardinalidad:	One -to- Zero-or-More
Permite Nulos:	No permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_elemento

Descripción:	Almacena los diferentes elementos relacionados con cada una de las categorias: Comunicaciones, infraestructura, etc
Propietario:	dbo
Nombre BD:	hds_systemdb
Número de columnas:	2
Número de índices:	0
Número de foreign keys:	0
Códigos:	0
Tipo:	Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
ele_codigo	char(10)	No permitido	Almacena el código de cada elemento.
ele_descripcion	char(20)	Permitido	Almacena la descripción de cada elemento.

Claves foraneas	Hijo	Padre
hds_elemento_hds_subel_elemento_FK1	hds_subel_elemento.ele_codigo	ele_codigo
hds_elemento_hds_categoria_elemento_FK1	hds_categoria_elemento.ele_codigo	ele_codigo

hds_pedido

Descripción: Tabla Principal que registra información básica para cada uno de los requerimientos.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 17

Número de índices: 0

Número de foreign keys: 10

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

TextImageOnGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
pro_codigo (FK)	char(8)	No permitido	Almacena el código de problema reportado y escogido de la base de conocimientos.
ped_ticket	char(4)	No permitido	Almacena el código unico para cada incidente.
ped_fechahora	datetime	No permitido	Registra la fecha y hora en la que ingresa un incidente.
ped_fechorafin	datetime	No permitido	Registra la fecha y hora en la que un incidente es solucionado.
ped_estado	char(20)	Permitido	Registra el estado en el que un incidente se encuentra (Abierto o Cerrado)
ped_problema_detalle	text	Permitido	Registra el detalle como un incidente es reportado por cada usuario.
tec_codigo (FK)	char(10)	Permitido	Registra el código del tecnico a carga del incidente reportado.
tpe_codigo (FK)	char(2)	Permitido	Registra el código del tipo de pedido reportado (Problema, requerimiento, etc).
pri_codigo (FK)	char(2)	Permitido	Registra el código de la prioridad asiganada al incidente.
tie_codigo (FK)	char(4)	Permitido	
sev_codigo (FK)	char(2)	Permitido	
ins_codigo (FK)	char(4)	Permitido	Registra el código de la institución involucrada en el incidente.
usu_codigo (FK)	char(10)	Permitido	Registra el código de usuario que reporta el incidente.
ped_responsable	char(10)	Permitido	Registra el código del helpdesk responsable, quien toma el incidente en primera instancia.
age_codigo (FK)	char(3)	Permitido	Registra el código de agencia involucrada en el incidente reportado.
cat_codigo (FK)	char(4)	Permitido	Registra el código de la categoría asignada al incidente.
ele_codigo (FK)	char(10)	Permitido	Registra el código del elemento asignado al incidente.

Claves foraneas	Hijo	Padre
hds_tecnico_hds_pedido_FK1	tec_codigo	hds_tecnico.tec_codigo
hds_tipopedido_hds_pedido_FK1	tpe_codigo	hds_tipopedido.tpe_codigo
hds_prioridad_hds_pedido_FK1	pri_codigo	hds_prioridad.pri_codigo
hds_solucion_hds_pedido_FK1	pro_codigo	hds_problema_solucion.pro_codigo
hds_tiempos_hds_pedido_FK1	tie_codigo	hds_tiempos.tie_codigo
hds_severidad_hds_pedido_FK1	sev_codigo	hds_severidad.sev_codigo
hs_instituciones_hds_pedido_FK1	ins_codigo	hs_instituciones.ins_codigo
hds_usuario_hds_pedido_FK1	usu_codigo	hds_usuario.usu_codigo
hds_agencia_hds_pedido_FK1	age_codigo	hds_agencia.age_codigo
hds_categoria_elemento_hds_pedido_FK1	cat_codigo ele_codigo	hds_categoria_elemento.cat_codigo hds_categoria_elemento.ele_codigo
hds_pedido_hds_seguimiento_FK1	hds_seguimiento.pro_codigo hds_seguimiento.ped_ticket	pro_codigo ped_ticket

Detalles de clave foranea (hijo)

hds_tecnico_hds_pedido_FK1

Hijo	**Padre**
tec_codigo	hds_tecnico.tec_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_tipopedido_hds_pedido_FK1

Hijo	**Padre**
tpe_codigo	hds_tipopedido.tpe_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_prioridad_hds_pedido_FK1

Hijo	**Padre**
pri_codigo	hds_prioridad.pri_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_solucion_hds_pedido_FK1

Hijo	**Padre**
pro_codigo	hds_problema_solucion.pro_codigo

Tipo de relación:	Identifying
Cardinalidad:	One -to- Zero-or-More
Permite Nulos:	No permitido
Verb phrase:	has

Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_tiempos_hds_pedido_FK1

Hijo	**Padre**
tie_codigo	hds_tiempos.tie_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_severidad_hds_pedido_FK1

Hijo	**Padre**
sev_codigo	hds_severidad.sev_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hs_instituciones_hds_pedido_FK1

Hijo	**Padre**

ins_codigo	hs_instituciones.ins_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_usuario_hds_pedido_FK1

Hijo	**Padre**
usu_codigo	hds_usuario.usu_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_agencia_hds_pedido_FK1

Hijo	**Padre**
age_codigo	hds_agencia.age_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of

Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_categoria_elemento_hds_pedido_FK1

Hijo	**Padre**
cat_codigo	hds_categoria_elemento.cat_codigo
ele_codigo	hds_categoria_elemento.ele_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_prioridad

Descripción: Almacena las prioridades definidas en el estudio de procesos.
Propietario: dbo
Nombre BD: hds_systemdb
Número de columnas: 2
Número de índices: 0
Número de foreign keys: 0
Códigos: 0
Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY
Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
pri_codigo	char(2)	No permitido	Registra el código de prioridades.
pri_descripcion	char(20)	No permitido	Registra la descripción de las prioridades.

Claves foraneas	Hijo	Padre
hds_prioridad_hds_pedido_FK1	hds_pedido.pri_codigo	pri_codigo
hds_prioridad_hds_tiempos_FK1	hds_tiempos.pri_codigo	pri_codigo

hds_problema

Descripción: Registra los problemas que ingresan a la base de conocimientos.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
pro_codigo	char(10)	No permitido	Registra el código unico para cada problema.
pro_descripcion	char(100)	Permitido	Registra la descripción de cada problema.

Claves foraneas	Hijo	Padre
hds_problema_hds_pedido_FK1	hds_problema_solucion.pro_codigo	pro_codigo
hds_problema_hds_problema_su	hds_problema_subelemento.pro_codigo	pro_codigo

belemento_FK1		

hds_problema_solucion

Descripción: Tabla que permite romper la relación de muchos a muchos entre problema y solución.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 2

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK No

Columnas	Tipo de dato	Permite Nulos	Observaciones
pro_codigo (FK)	char(8)	No permitido	
sol_codigo (FK)	char(10)	No permitido	

Claves foraneas	Hijo	Padre
hds_problema_hds_pedido_FK1	pro_codigo	hds_problema.pro_codigo
hds_problema_hds_solucion_FK1	sol_codigo	hds_solucion.sol_codigo
hds_solucion_hds_pedido_FK1	hds_pedido.pro_codigo	pro_codigo

Detalles de clave foranea (hijo)

hds_problema_hds_pedido_FK1

Hijo	Padre
pro_codigo	hds_problema.pro_codigo

Tipo de relación:	Identifying
Cardinalidad:	One -to- One-or-More
Permite Nulos:	No permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_problema_hds_solucion_FK1

Hijo	**Padre**
sol_codigo	hds_solucion.sol_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	One -to- One-or-More
Permite Nulos:	No permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_problema_subelemento

Descripción:	Tabla que permite romper la relación de muchos a muchos entre problema y subelemento.
Propietario:	dbo
Nombre BD:	hds_systemdb
Número de columnas:	2
Número de índices:	0
Número de foreign keys:	2
Códigos:	0
Tipo:	Tabla

Columnas	Tipo de dato	Permite Nulos	Observaciones
pro_codigo (FK)	char(10)	Permitido	
sub_codigo (FK)	char(10)	No permitido	

Claves foraneas	Hijo	Padre
hds_problema_hds_problema_subelemento_FK1	pro_codigo	hds_problema.pro_codigo
hds_subelemento_hds_problema_subelemento_FK1	sub_codigo	hds_subelemento.sub_codigo

Detalles de clave foranea (hijo)

hds_problema_hds_problema_subelemento_FK1

Hijo	**Padre**
pro_codigo	hds_problema.pro_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hds_subelemento_hds_problema_subelemento_FK1

Hijo	**Padre**
sub_codigo	hds_subelemento.sub_codigo

Tipo de relación:	Identifying
Cardinalidad:	One -to- Zero-or-More
Permite Nulos:	No permitido

Verb phrase: has

Inverse phrase: is of

Ref. Integrity on update: No action

Ref. Integrity on delete: No action

hds_seguimiento

Descripción: Tabla que almacena información sobre cada incidente como son número de ticket, hora de incio y hora de finalización del pedido así como tambien que técnico resolvió dicho pedido.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 7

Número de índices: 0

Número de foreign keys: 2

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK No

Columnas	Tipo de dato	Permite Nulos	Observaciones
ped_ticket (FK)	char(4)	Permitido	Almacena el código asignado a un incidente y asi realizar el seguimiento respectivo.
tec_codigo (FK)	char(10)	Permitido	Almacena el código del tecnico asiganado para solucionar el incidente.
seg_fechorainicio	datetime	Permitido	Almacena la fecha en la que un incidente es asignado a un técnico específico.
seg_fechorafin	datetime	Permitido	Almacena la fecha en la que un incidente es

			dado solucion por un técnico específico.
seg_horainic	char(8)	Permitido	Almacena la hora en la que un incidente es asignado a un tecnico especifico.
seg_horafin	char(8)	Permitido	Almacena la hora en la que un incidente es dado solucion por un técnico específico.
pro_codigo (FK)	char(8)	Permitido	Almacena el código del problema.

Claves foraneas	**Hijo**	**Padre**
hds_pedido_hds_seguimiento_FK1	pro_codigo ped_ticket	hds_pedido.pro_codigo hds_pedido.ped_ticket
hds_tecnico_hds_seguimiento_FK1	tec_codigo	hds_tecnico.tec_codigo

Detalles de clave foranea (hijo)

hds_pedido_hds_seguimiento_FK1

Hijo	**Paren**
pro_codigo	hds_pedido.pro_codigo
ped_ticket	hds_pedido.ped_ticket

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_tecnico_hds_seguimiento_FK1

Hijo	**Padre**
tec_codigo	hds_tecnico.tec_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido

Verb phrase: has

Inverse phrase: is of

Ref. Integrity on update: No action

Ref. Integrity on delete: No action

hds_severidad

Descripción:

Propietario: dbo

Nombre BD: **hds_systemdb**

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
sev_codigo	char(2)	No permitido	
sev_descripcio	char(20)	Permitido	

Claves foraneas	Hijo	Padre
hds_severidad_hds_pedido_FK1	hds_pedido.sev_codigo	sev_codigo
hds_severidad_hds_tiempos_FK1	hds_tiempos.sev_codigo	sev_codigo

hds_solucion

Descripción: Registra las soluciones generadas para dar posibles alternativas a cada uno de los problemas existentes en la base de conocimientos.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
sol_codigo	char(10)	No permitido	Registra el código unico de cada solución.
sol_descripcion	char(100)	Permitido	Registra la descripción de cada solución.

Claves foraneas	Hijo	Padre
hds_problema_hds_solucion_FK1	hds_problema_solucion.sol_codigo	sol_codigo

hds_subel_elemento

Descripción: Tabla que permite romper la relación de muchos a muchos entre subelemento y elemento.

Propietario: dbo

Nombre BD: **hds_systemdb**

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 2

Códigos: 0

Tipo: Tabla

Columnas	Tipo de dato	Permite Nulos	Observaciones
sub_codigo (FK)	char(10)	No permitido	
ele_codigo (FK)	char(10)	No permitido	

Claves foraneas	Hijo	Padre
hds_elemento_hds_subelemnto_FK1	sub_codigo	hds_subelemento.sub_codigo
hds_elemento_hds_subel_elemento_FK1	ele_codigo	hds_elemento.ele_codigo

Detalles de clave foranea (hijo)

hds_elemento_hds_subelemnto_FK1

Hijo	Padre
sub_codigo	hds_subelemento.sub_codigo

Tipo de relación: Identifying

Cardinalidad: One -to- Zero-or-More

Permite Nulos: No permitido

Verb phrase: has

Inverse phrase: is of

Ref. Integrity on update: No action

Ref. Integrity on delete: No action

hds_elemento_hds_subel_elemento_FK1

Hijo	Padre
ele_codigo	hds_elemento.ele_codigo

Tipo de relación: Identifying

Cardinalidad: One -to- Zero-or-More

Permite Nulos: No permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_subelemento

Descripción: Tabla de subelementos para las categorias.
Propietario: dbo
Nombre BD: hds_systemdb
Número de columnas: 2
Número de índices: 0
Número de foreign keys: 0
Códigos: 1
Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY
Clustered PK No

Columnas	Tipo de dato	Permite Nulos	Observaciones
sub_codigo	char(10)	No permitido	Registra el código unico de cada subelemebto.
sub_descripcion	char(20)	Permitido	Registra la descripción para cada subelemento.

Claves foraneas	Hijo	Padre
hds_elemento_hds_subelemnto_FK1	hds_subel_elemento.sub_codigo	sub_codigo
hds_subelemento_hds_problema_subelemento_FK1	hds_problema_subelemento.sub_codigo	sub_codigo

hds_tecnico

Descripción: Almacena información básica de cada uno de los técnicos de las diferentes areas y categorias.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 8

Número de índices: 0

Número de foreign keys: 3

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK No

Columnas	Tipo de dato	Permite Nulos	Observaciones
tec_codigo	char(10)	No permitido	Registra el código único de cada técnico.
tec_apellido (FK)	char(30)	No permitido	Registra el apellido del técnico.
tec_nombre	char(30)	No permitido	Registra el nombre del técnico.
tec_telefono1	char(10)	Permitido	Registra el número telefónico del técnico.
tec_telefono2	char(10)	Permitido	Registra el número telefónico alterno del técnico.
tec_email	char(40)	Permitido	Registra la dirección de correo técnico.
cat_codigo (FK)	char(4)	Permitido	Registra la categoría a la que el técnico pertenece.
are_codigo (FK)	char(4)	Permitido	Registra el codigo de área a la que el técnico pertenece.

Claves foraneas	Hijo	Padre
hds_usuario_hds_tecnico_FK1	tec_apellido	hds_usuario.usu_codigo

hds_categoria_hds_tecnico_FK1	cat_codigo	hds_categoria.cat_codigo
hds_area_hds_tecnico_FK1	are_codigo	hds_area.are_codigo
hds_tecnico_hds_pedido_FK1	hds_pedido.tec_codigo	tec_codigo
hds_tecnico_hds_seguimiento_FK1	hds_seguimiento.tec_codigo	tec_codigo

Detalles de clave foranea (hijo)

hds_usuario_hds_tecnico_FK1

Hijo	**Padre**
tec_apellido	hds_usuario.usu_codigo

Tipo de relación: Non-Identifying
Cardinalidad: One -to- Zero-or-More
Permite Nulos: No permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_categoria_hds_tecnico_FK1

Hijo	**Padre**
cat_codigo	hds_categoria.cat_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_area_hds_tecnico_FK1

Hijo	Padre
are_codigo	hds_area.are_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_tiempos

Descripción:
Propietario: dbo
Nombre BD: hds_systemdb
Número de columnas: 6
Número de índices: 0
Número de foreign keys: 2
Códigos: 0
Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY
Clustered PK Yes

Columnas	Data Tipo	Permite Nulos	Observaciones
tie_codigo	char(4)	No permitido	
tie_tiempo	char(10)	No permitido	
pri_codigo (FK)	char(2)	Permitido	

sev_codigo (FK)	char(2)	Permitido	
tie_nombreus	char(40)	Permitido	
tie_dirmail	char(50)	Permitido	

Claves foraneas	Hijo	Padre
hds_prioridad_hds_tiempos_FK1	pri_codigo	hds_prioridad.pri_codigo
hds_severidad_hds_tiempos_FK1	sev_codigo	hds_severidad.sev_codigo
hds_tiempos_hds_pedido_FK1	hds_pedido.tie_codigo	tie_codigo

Detalles de clave foranea (hijo)

hds_prioridad_hds_tiempos_FK1

Hijo	**Padre**
pri_codigo	hds_prioridad.pri_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action
Ref. Integrity on delete: No action

hds_severidad_hds_tiempos_FK1

Hijo	**Padre**
sev_codigo	hds_severidad.sev_codigo

Tipo de relación: Non-Identifying
Cardinalidad: Zero-or-One -to- Zero-or-More
Permite Nulos: Permitido
Verb phrase: has
Inverse phrase: is of
Ref. Integrity on update: No action

Ref. Integrity on delete: No action

hds_tipopedido

Descripción: Contiene el código y descripción de los diferentes tipos de pedido: Problema, consulta, etc.

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 2

Número de índices: 0

Número de foreign keys: 0

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK Yes

Columnas	Tipo de dato	Permite Nulos	Observaciones
tpe_codigo	char(2)	No permitido	Registra el código de los tipo de pedido.
tpe_descripcion	char(10)	No permitido	Registra la descripción de cada tipo de pedido.

Claves foraneas	Hijo	Padre
hds_tipopedido_hds_pedido_FK1	hds_pedido.tpe_codigo	tpe_codigo

hds_usuario

Descripción: Almacena información de cada uno de los usuarios en este caso son las posibles personas que reportan o hacen pedidos (estos son alimentados de la base de nómina de EXSERSA).

Propietario: dbo

Nombre BD: hds_systemdb

Número de columnas: 7

Número de índices: 0

Número de foreign keys: 1

Códigos: 0

Tipo: Tabla

Atributos extendidos:

OnFileGroup PRIMARY

Clustered PK No

Columnas	Tipo de dato	Permite Nulos	Observaciones
usu_codigo	char(10)	No permitido	Registra el código de usuario, estos son alimentados de la base de nómina de EXSERSA.
Usu_nombre	char(40)	No permitido	Registra el nombre de usuario, estos son alimentados de la base de nómina de EXSERSA.
usu_telefono1	char(9)	Permitido	Registra el númenro de teléfono del usuario.
usu_agencia	char(40)	No permitido	Registra la agecia a la que el usuario pertenece.
are_codigo (FK)	char(4)	Permitido	Registra el código de área a a la que el usuario pertenece.
usu_email	varchar(40)	Permitido	Registra la direccion de correo electrónico del usuario.
usu_telefono2	char(9)	Permitido	Registra el númenro de teléfono alterno del usuario.

Claves foraneas	Hijo	Padre
hds_area_hds_usuario_FK1	are_codigo	hds_area.are_codigo
hds_usuario_hds_tecnico_FK1	hds_tecnico.tec_apellido	usu_codigo
hds_usuario_hds_pedido_FK1	hds_pedido.usu_codigo	usu_codigo

Detalles de clave foranea (hijo)

hds_area_hds_usuario_FK1

Hijo **Padre**

are_codigo	hds_area.are_codigo

Tipo de relación:	Non-Identifying
Cardinalidad:	Zero-or-One -to- Zero-or-More
Permite Nulos:	Permitido
Verb phrase:	has
Inverse phrase:	is of
Ref. Integrity on update:	No action
Ref. Integrity on delete:	No action

hs_instituciones

Descripción:	Tabla que almacena información de cada una de las insituciones que participan en el reporte de pedidos.
Propietario:	dbo
Nombre BD:	hds_systemdb
Número de columnas:	2
Número de índices:	0
Número de foreign keys:	0
Códigos:	0
Tipo:	Tabla

Atributos extendidos:

OnFileGroup	PRIMARY
Clustered PK	No

Columnas	Tipo de dato	Permite Nulos	Observaciones
ins_codigo	char(6)	No permitido	
ins_descripcion	char(50)	No permitido	

Claves foraneas	Hijo	Padre

hs_instituciones_hds_pedido_FK1	hds_pedido.ins_codigo	ins_codigo

5.3.- Diseño de Interfases

Las interfaces visuales se detallan en forma general a continuación:

5.3.1.- Pantalla de presentación del Sistema

Este tipo de pantalla permite visualizar momentáneamente información básica del sistema en ejecución. Esta pantalla se visualizará al momento de inicializar el sistema.

Figura 5.2: (Pantalla de presentación del sistema)

5.3.2.- Pantalla de conexión a base de datos

Estas pantallas se representan con una ventana como se detalla la figura 5.3. En la misma se encuentra la parametrización para realizar la conexión a la base de datos.

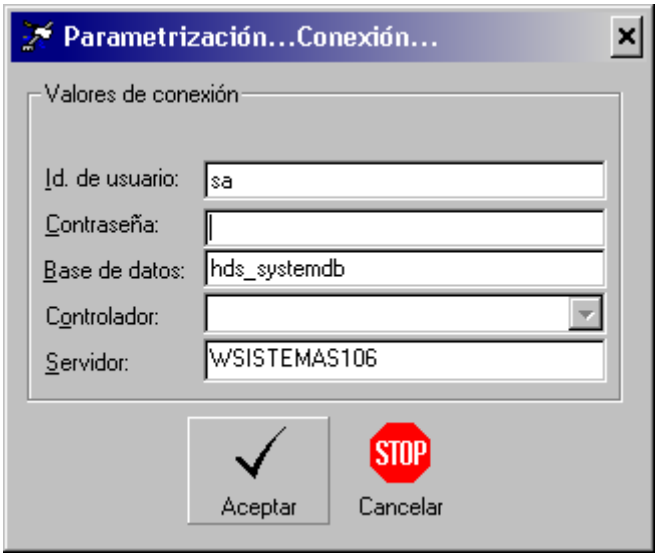

Figura 5.3: (Pantalla de parametrización de base de datos)

5.3.3.- Pantalla de ingreso y validación de usuarios

Este tipo de pantallas permiten la validación de los usuarios al ingresar al sistema, cada usuario con un perfil asignado.

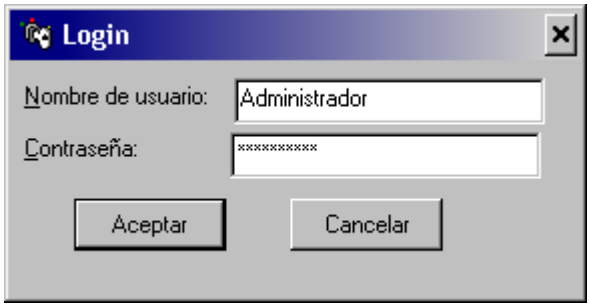

Figura 5.4: (Pantalla de validación para ingreso al sistema)

5.3.4.- Pantalla general interfase con el Usuario

Esta pantalla contendrá toda la interfaz con los operadores del sistema, dependiendo del perfil de cada uno tendrá acceso a los módulos existentes.

Constará de una barra amigable y un menú que permite ingresar a las diferentes opciones y los diferentes módulos como son:

- Requerimientos.
- Técnicos.
- Instituciones.
- Usuarios.
- Problema / Solución.
- Consultas / Estadísticas.

Cada uno de los módulos estará hábil en ventanas tabuladas y constará de los objetos que se requieran como son cajas de texto, botones, cajas de listas, etiquetas, cajas de selección, etc.

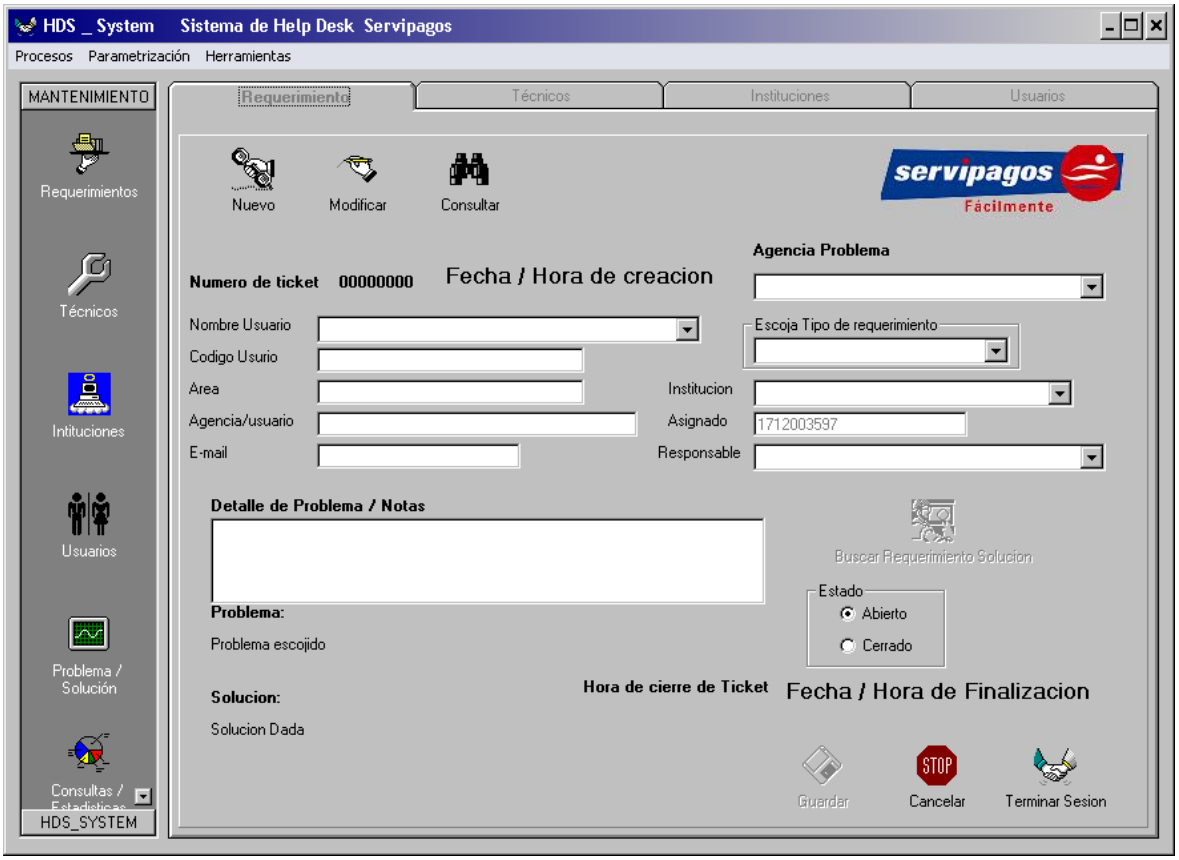

Figura 5.5: (Pantalla General de interfase con el usuario)

CAPITULO VI

IMPLEMENTACIÓN Y PRUEBAS

6.1.- Selección de Herramientas

6.1.1.- Herramientas para el Desarrollo

6.1.1.1.- Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de base de datos relacionales cliente / servidor de alto rendimiento que admite un elevado volumen de procesamiento de transacciones (como la entrada de pedidos en línea, inventario,

contabilidad o facturación), además sirve de ayuda en la toma de decisiones (como aplicaciones de análisis de ventas).

SQL Server se basa en Microsoft Windows NT Server y puede instalarse como sistema de base de datos de escritorio en máquinas con Windows NT Workstation, 95, 98 y XP.

SQL Server debido a su arquitectura abierta proporciona muchas herramientas e interfaces para trabajo de red en otros sistemas operativos de Microsoft como Win 3.1, MS DOS, UNIX.

SQL forma parte de una familia de productos integrados entre los que se incluyen herramientas de desarrollo, administración de sistemas, componentes de sistemas distribuidos e interfases de desarrollo abiertas, es parte clave de Microsoft Back Office.

Entre las herramientas que podemos mencionar están:

Herramientas de desarrollo:

- Power Builder.
- Inprise Delphi.
- Racle Power Objects.
- Microfocus Cobol.
- Microsoft Visual Studio.
- Microsoft Office (Word, Excel, Power Point, Access y Query)

Herramientas de SQL Server

- Administrador corporativo de SQL Server.
- Utilidades
  - Analizador de consultas
  - SQL Server

- Analizador de SQL Server
- Monitor de rendimiento de SQL Server
- SQL Mail
- Programación del servidor
- Procedimientos almacenados extendido
- Herramientas relacionadas
- Servicio de transformación de dato (DTS)

SQL Server

- Sistemas abiertos de datos.
- Motor de SQL Server.
- Agente SQL Server

Otros servidores de Back Office

- Windows NT Server.
- Microsoft Exchange Server.
- System Management Server.
- Microsoft Transaction Server.
- Internet Information Server.
- Microsoft Commercial.
- Internet System

Interfaces Abiertas

- ADO
- OLEDB
- OLE DB para OBDC

- OBDC

6.1.1.2.- Microsoft Visual Basic

Visual Basic es una herramienta de diseño de aplicaciones para Windows, estas se desarrollan en gran parte a partir del diseño de una interfase gráfica. En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interfase gráfica.

Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos.

6.1.1.3.- Programación Visual

La creación de un programa bajo Visual Basic lleva la siguiente secuencia de pasos:

- Creación de una interfase de usuario. Esta interfase será la principal vía de comunicación entre el hombre y la máquina, tanto para salida de datos como para entrada. Será necesario partir de una ventana conocida como Formulario a la que se añadirá los objetos necesarios.
- La definición de las propiedades de los objetos colocados en ese formulario. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para qué sirven.
- Generación del código asociado a los eventos que ocurran a estos objetos. A la respuesta a estos eventos le llamamos Procedimiento, y deberá generarse de acuerdo a las necesidades del programa.
- Generación del código del programa. Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Sin embargo, VB ofrece la posibilidad de

establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados Funciones, y otros llamados Procedimientos. Estos Procedimientos no responden a un evento acaecido a un objeto, sino que responden a un evento producido durante la ejecución del programa.

6.1.1.4.- Microsoft Visio Professional

Ayuda a profesionales técnicos como profesionales de TI, programadores e ingenieros, a visualizar ideas, información y sistemas existentes, y a crear prototipos de otros nuevos.

Visio 2002 facilita la creación de gran impacto con herramientas para crear diagramas con un aspecto sin precedentes y la posibilidad para compartirlos en el contexto de la comunicación diaria de su negocio. La línea de productos Visio 2002 es también más fácil de aprender y utilizar, con numerosas funciones clásicas de Office, y nuevas funciones compartidas con Office XP. Por último, el gran número de recursos en línea disponibles le permitirán aumentar la productividad con más facilidad que nunca.

Visio incorpora líneas y texto homogéneos con aspecto mejorado, mayor riqueza de color, Integración de imágenes, los artículos técnicos están disponibles en MSDN, TechNet y Microsoft Knowledge Base.

Las soluciones individuales de Visio le ayudarán a comprender mejor ideas, información y sistemas gracias a una mayor integración con otros productos y tecnologías.

Visio facilita dar soluciones técnicas, nuevas formas para objetos Exchange de Active Directory.

Proporciona soluciones avanzadas para la creación de diagramas de red, donde se ofrecen recursos para crear diagramas, soluciones actualizadas y formas de equipo de red específicas de cada fabricante.

El nuevo formato de archivo XML permite la interoperabilidad con aplicaciones compatibles con este formato y facilita el almacenamiento y el intercambio de información basada en diagramas, incluidos los datos no asociados con una página o forma.

Más de 90 propiedades y métodos de automatización nuevos proporcionan acceso a más datos de diagramas de Visio.

Visio permite a los programadores ampliar y mejorar Visio mediante programación y complementos COM.

La tecnología Authenticode™ de Microsoft permite a los programadores firmar digitalmente los proyectos de VBA en sus soluciones con ayuda de un certificado digital que identifica al programador como una fuente de confianza. Si se modifica un proyecto firmado, los usuarios pueden decidir deshabilitar las macros de un documento de Visio.

Los programadores pueden ampliar y mejorar la funcionalidad de Visio mediante la aplicación incorporada Visual Basic para Aplicaciones (VBA) 6.3.

Los administradores y usuarios tienen la opción de quitar Visual Basic para Aplicaciones de las instalaciones de Office de usuario único, para grupo u organizaciones.

6.2.- Descripción de Procedimientos Almacenados

Se detalla a continuación cada uno de los Scripts realizados para realizar los procesos durante la ejecución del sistema.

Tabla 4.3: (Descripción de procesos almacenados)

Nombre Stored Procedure	Parámetros entrada / salida	Descripción
spcc_busca_area	Descripción de área a buscar	Permite realizar búsquedas en la tabla de áreas.
spcc_busca_categ	Descripción de categoría a buscar.	Permite realizar búsquedas en la tabla de categorías.
spcc_busca_codsubelemento	Descripción de subelmento a buscar	Permite realizar búsquedas en la tabla de subelementos.
spcc_busca_codtecnico	Código de técnico a buscar.	Realiza búsquedas a partir del código de técnico.
spcc_busca_elemento	Descripción de categoría a buscar.	Realiza búsquedas de elementos a partir de la descripción de una categoría.
spcc_busca_institucion	Nombre de institución a buscar	Realiza búsquedas de instituciones.
spcc_busca_problema_lemento	Subelemento y elemento con el cual se relaciona el problema.	Retorna información de la tabla hds_problema.
spcc_busca_problema_nombre	Descripción del problema.	Retorna información de la tabla hds_problema.
spcc_busca_solucion	Código de solución	Despliega información de la tabla de soluciones.
spcc_busca_subelemento	Descripción de categoría	Devuelve el código, descripción de subelemento y también el código de elemento relacionado.
spcc_busca_tecnico	Nombre de tecnico a buscar	Retorna información del tecnico encontrado.
spcc_busca_tecnico_area	Descripción de área	Retorna información del tecnico encontrado.
spcc_busca_tecnico_cat	Descripción de categoría	Retorna información del tecnico encontrado.

spcc_busca_tipopedido	Descripción de tipo de pedido	Retorna registros que coincidan con esa descripción.
spcc_busca_usuario	Nombre o parte el nombre de Usuario	Realiza búsqueda de usuarios en base al nombre ordenándolos en forma alfabética.
spcc_buscaproblema	Descripción de problema.	Retorna los datos referentes a ese problema ordenado por descripción.
spcc_consulta_pedidos	Depende de la consulta que se desee realizar.	Permite realizar consultas de los pedidos con uno o varios parámetros.
spcc_estadisticas_elesubelproblema	Recibe como parámetros: elemento, subelemento, tipo de pedido, fecha inicial, fecha final	Retorna el total de pedidos por tipo pedido, categoría, elemento, problema
spcc_estadisticas_pedidos		
spcc_estadisticas_tpcateg	Tipo de pedido y fechas.	Total de pedidos por tipo pedido y categoría
spcc_estadisticas_tpctelemento	Categiria, tipo de pedido y fechas.	Total de pedidos por tipo pedido, categoría, elemento
spcc_estadisticas_tpctelsubelemento	Elemento, tipo de pedido y fechas.	Total de pedidos por tipo pedido, categoría, elemento
spcc_estadisticas_tpedido	Fechas inicial y final	Total de pedidos por tipo de pedido
spcc_nueva_parametrizacion	Código y descripción de categria, elemento y subelemento.	Crea nuevas categorías, elemento y subelemento
spcc_nueva_parametrizacion1	Código y descripción de categoría, elemento y subelemento.	Inserta nuevos elementos y subelementos.
spcc_nueva_parametrizacion2	Código y descripción de elemento y subelemento.	Inserta nuevos subelementos.
spcc_nuevo_institucion	Código y descripción de nueva institución.	Inserta nuevas instituciones.
spcc_nuevo_pedido	Todos los parámetros de la tabla hds_pedido	Inserta nuevos pedidos o incidentes en la tabla de pedidos.
spcc_nuevo_problema	Todos los parámetros de la tabla hds_problema	Inserta nuevos problemas en la tabla de problemas.

spcc_nuevo_seguimiento	Todos los parámetros de la tabla hds_seguimiento	Inserta nuevos seguimientos en la tabla de seguimientos.
spcc_nuevo_solucion	Todos los parámetros de la tabla hds_solucion	Inserta nuevas soluciones en la tabla de soluciones.
spcc_nuevo_tecnico	Todos los parámetros de la tabla hds_tecnico	Inserta nuevos registros en la tabla de técnicos.
spcc_nuevo_usuario	Todos los parámetros de la tabla hds_usuario	Inserta nuevos registros en la tabla de usuarios.
spcc_problema_solucion	Código de solución, código de problema	Retorna elemento, problema, solución
spcc_valida_cedula	Numero de cedula o Ruc.	Valida si un numero de cedula o Ruc son correctos.
spcc_valida_clave	Usuario, password	Valida si son correctos los datos para dar acceso al sistema.

6.3.- Pruebas

6.3.1.- Pruebas de Caja blanca

Sinónimos:

- Pruebas estructurales
- Pruebas de caja transparente

Estas se basan en el conocimiento de la lógica interna del código del sistema las pruebas contemplan los distintos caminos que se pueden generar gracias a la estructuras condicionales.

Lograr una buena cobertura con pruebas de caja blanca es un objetivo deseable; pero no suficiente a todos los efectos. Un programa puede estar

perfecto en todos sus términos, y sin embargo no servir a la función que se pretende.

Por ejemplo, si escribimos una rutina para ordenar datos por orden ascendente, pero el cliente los necesita en orden decreciente; no hay prueba de caja blanca capaz de detectar la desviación.

Las pruebas de caja blanca convencen de que un programa hace bien lo que hace; pero no de que haga lo que necesita el usuario.

Para el caso de HDS_System las pruebas de Caja Blanca se realizaron en conjunto con las personas involucradas en el sistema.

6.3.2.- Pruebas de Caja negra

Sinónimos:

- Pruebas de caja opaca
- Pruebas funcionales
- Pruebas de entrada/salida
- Pruebas inducidas por los datos

Las pruebas de caja negra no considera la codificación dentro de sus parámetros a evaluar es decir no están basados en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requerimientos establecidos y en la funcionalidad del programa. Por ello se denominan pruebas funcionales, y el evaluador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro, debido a que se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación.

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de cobertura de especificación para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea.

Para el caso de HDS_System las pruebas de Caja Negra se realizaron en conjunto con las personas involucradas en los requerimientos del sistema y los operadores del Helpdesk.

CAPITULO VII

CONCLUSIONES Y RECOMENDACIONES

7.1.- Conclusiones

- El estudio y mejoramiento de los procesos que una empresa utiliza sea este en una área determinada conlleva al perfeccionamiento de los mismos, a su vez sugiere su automatización.

- El cambio de un sistema manual a un sistema automatizado que maneje la información en forma ordenada, segura y confiable, mejora tanto en la administración del Helpdesk, así como en la administración de la información.

- El sistema facilitará y agilitará la resolución de incidentes por cuanto maneja una base de conocimientos, la misma que se alimentará conforme las necesidades de los usuarios.

- Realizada la implantación del sistema se espera optimizar los tiempos de respuesta a los incidentes reportados, así también llevar un control de quienes lo resuelven, dando así a los niveles que lo requieran los reportes necesarios para mejorar los procesos.

- El trabajo elaborado contribuyó a aprender nuevas herramientas de diseño y desarrollo, así como también a reforzar los conocimientos académicos de quienes participaron en la planificación y el desarrollo de este proyecto.

7.2.- Recomendaciones

- El uso de sistemas en las empresas cada vez es más común, esto facilita el trabajo y ayuda a dar un servicio más eficiente y oportuno, más aún en un área tan sensible como lo es la atención a los clientes. Este tipo de sistemas son los que las organizaciones en busca de dar un valor agregado a los servicios que prestan, deberían tratar de implantar en la actualidad.

- El automatizar procesos conlleva a disciplinar al personal involucrado por lo que se recomienda instruir al mismo sobre las mejoras realizadas, así como también del manejo del sistema a implantarse.

BIBLIOGRAFÍA

Roger S. Pressman (1997). McGraw-Hill. Ingeniería del software. Un enfoque práctico, Cuarta Edición.

Ed. Yourdon (1993), Análisis estructurado moderno, Prentice-Hall Hispanoamericana.

Bernd Bruegge & Allen H Dutoit (2000), Object-Oriented Software Engineering, Conquering Complex and Changing Systems, Prentice Hall, New Jersey.

Craig Larman (1999), UML y patrones. Introduccion al analisis y diseño orientado a objetos, Segunda edicion. Prentice-Hall.

G. Booch, J. Rumbaugh, I. Jacobson (1999). El Lenguaje Unificado de Modelado. Addison Wesley Iberoamericana.

BIBLIOGRAFÍA ELECTRÓNICA

http://www.clikear.com/manuales/uml/procesodesarrollo.asp

http://www.sitecpro.com/home/principal/Productos/ivr

http://www.creangel.com/uml/diagramas.php

http://manuales.astalaweb.com/Manuales/UML.asp

http://manuales.astalaweb.com/Manuales/VisualBasic.asp

http://manuales.astalaweb.com/Manuales/SQLServer.asp

http://www.dcc.uchile.cl/~psalinas/uml/