14
2EJ

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

ORACLE CASE COMO SOPORTE PARA LA INGENIERIA REVERSIVA

SEMINARIO DE INVESTIGACION INFORMATICA

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN INFORMATICA

P R E S E N T A N :

GARCIA GALICIA ELIA
HERNANDEZ VAZQUEZ LETICIA
PEREZ MARTINEZ ELSA LILIA

ASESOR DEL SEMINARIO:

L. C. Y M. EN C. C. MARINA TORIZ GARCIA

MEXICO, D. F. 1995

A MIS PADRES:

Por la ayuda,
Por la confianza que han depositado en mí y
Por el apoyo en los momentos más difíciles.

A MI HERMANO :

Porque sé que puedo contar con él cuando lo necesite.

ELIA

A MIS PADRES Ma DEL CARMEN RAMIREZ Y JUAN HERNANDEZ:

Quienes siempre me han apoyado y dado toda su comprensión y cariño; ya que sin ellos no hubiera logrado todas mis metas.

A MI HERMANA VERONICA:

Que siempre me proporciona su ayuda cuando la necesito.

Así también, agradezco a todas las personas que me apoyaron incondicionalmente y confiaron en mi.

LETICIA

A MIS PADRES:

Les agradezco su valiosa ayuda, comprensión y apoyo que siempre me brindaron para el cumplimiento de mis objetivos.

A MI HERMANA:

Por su cooperación y apoyo que siempre me brindaste.

AL LIC. DIEGO VARGAS COLIN:

Por el apoyo que me brindo para la realización de este trabajo.

Así también, a todas las personas que siempre me apoyaron y creyeron en mi.

ELSA LILIA

A SHL SYSTEMHOUSE DE MEXICO, EN ESPECIAL A:

ING. ROLANDO HERNANDEZ ALBIN
MSC. JUAN CARLOS DE LA TORRE
ING. RAUL CARRAL PAVON

Por su confianza,y gran disposición, para que este trabajo sea una realidad, y al demás personal que de alguna forma contribuyó con nosotros.

A L.C. Y M. EN C. C. MARINA TORIZ GARCIA:

Por su valiosa ayuda en la dirección y asesoría de esta investigación, así como, por el interés que mostró durante la elaboración de este trabajo.

INDICE

PAG.

INTRODUCCION i

CAPITULO 1. EL CONTEXTO DE LA REVERSE ENGINEERING(RE)... 1

1.1 ORIGEN Y CONCEPTOS DE LA REVERSE ENGINEERING (RE).....1
1.2 LA REVERSE ENGINEERING Y RECUPERACION DEL DISEÑO 2
1.3 UTILIDAD DE LA REVERSE ENGINEERING 3
1.4 MANTENIMIENTO DEL SOFTWARE 4
1.5 CICLOS DE VIDA Y ABSTRACCIONES.....4
1.6 TERMINOS RELATIVOS A LA INGENIERIA REVERSIVA 5
1.7 PORQUE USAR REVERSE ENGINEERING 10
1.8 UBICACION DE LA INGENIERIA REVERSIVA DENTRO DE LA REINGENIERIA DE SOFTWARE ...13
1.9 OBJETIVOS ESPECIFICOS DE LA REINGENIERIA 14
1.10 EL CONTEXTO DE LA REINGENIERIA 14
1.11 IMPORTANCIA DE LA REINGENIERIA 17
1.12 TECNOLOGIA DE REINGENIERIA PARA MEJORAR EL SOFTWARE 20
1.13 ESTRATEGIA DE REINGENIERIA DE SOFTWARE Y RIESGOS DE MIGRACION 26
1.14 AVANCES FUTUROS 27

CAPITULO 2 . HERRAMIENTAS PARA LA INGENIERIA REVERSIVA 30

2.1 LA HERRAMIENTA CASE (COMPUTER AIDED SOFTWARE ENGINEERING) . ASPECTOS GENERALES 30
2.2 SOPORTE DE LAS HERRAMIENTAS CASE A USUARIOS.....32
2.3 ARQUITECTURA DE UN SISTEMA CASE 33
2.4 ETAPAS DE CASE.....35

2.5 HERRAMIENTAS Y METODOLOGIAS CASE 37
2.5.1 CASE * METHOD (UN ENFOQUE ESTRUCTURADO "TOP-DOWN") 38
2.5.2 CASE * DICTIONARY 41
2.5.3 CASE * DESIGNER 43
2.5.4 CASE * GENERATOR 45
2.6 OTRAS HERRAMIENTAS PARA LA INGENIERIA REVERSIVA.....46

CAPITULO 3. APLICACION DE LA INGENIERIA REVERSIVA A LAS BASES DE DATOS..... 53

3.1 PASOS DE LA INGENIERIA REVERSIVA 54
3.2 OBJETIVOS DE PROYECTOS CON INGENIERIA REVERSIVA 56
3.3 EL IMPACTO DE LA INGENIERIA REVERSIVA EN PROYECTOS 57
3.4 GUIA PARA APLICAR INGENIERIA REVERSIVA 58
3.5 PASOS DE LA INGENIERIA REVERSIVA EN BASE DE DATOS RELACIONAL 62
3.6 APLICACION DE LA INGENIERIA REVERSIVA A LAS BASES DE DATOS RELACIONALES 64
3.7 PUNTOS A CONSIDERAR EN LA APLICACION DE LA INGENIERIA REVERSIVA EN LA MIGRACION DE BASE DE DATOS RELACIONALES.....66

CAPITULO 4. IMPLEMENTACION DE LA INGENIERIA REVERSIVA EN SCORE 69

4.1 GENERALIDADES DEL SISTEMA SCORE.....70
4.2 APLICACION DE LA UTILERIA DE LA INGENIERIA REVERSIVA PARA BASE DE DATOS EN EL SISTEMA SCORE 73

CONCLUSIONES 81

APENDICES

1. RELACION DE TABLAS DEL SISTEMA SCORE 83
2. REPORTE CDRF.LIS 95
3. PANTALLAS DE CASE*DICTIONARY 119
4. REPORTE ENTIDAD-RELACION 131
5. DIAGRAMA ENTIDAD-RELACION 143

GLOSARIO 144

REFERENCIA 156

BIBLIOGRAFIA 157

INDICE DE FIGURAS.

PAG.

1. Proceso para llegar a una nueva aplicación 4

2. Proceso de Reingeniería y términos relacionados 11

3. Arquitectura típica de un sistema CASE 33

4. Requerimientos de CASE 37

5. Etapas CASE 39

6. Proceso de Ingeniería Reversiva 53

INTRODUCCION

SHL SYSTEMHOUSE

Systemhouse de México, S.A. de C.V.
Avenida Jalisco No. 180, 1er. Piso
C.P. 11870, México D.F.

México, D.F. a 7 de Junio de 1995

A quien corresponda:

Systemhouse de México, S. A. de C.V. consciente de la necesidad de vincular los conocimientos teóricos con la práctica profesional ha dado la oportunidad para que las alumnas García Galicia Elia, Hernández Vázquez Leticia y Pérez Martínez Elsa Lilia de la Fac. de Contaduría y Administración de la Universidad Nacional Autónoma de México desarrollen su seminario de investigación titulado " Reverse Engineering " para aspirar a obtener el título de Lic. en Informática

Para realizar lo anterior las alumnas han asistido a la empresa durante el periodo de Mayo 1994 a Mayo 1995, trabajando en el sistema SCORE (Sistema de Control de Operaciones y Registro Estadístico) aplicable a Casas de Cambio, en donde han investigado e implementado los principios del " Reverse Engineering " de ORACLE*CASE obteniendo resultados importantes para esta empresa, así como para el cumplimiento de los objetivos de las alumnas.

Cabe comentar, que su desempeño ha sido en forma responsable y con ética profesional.
Sin más por el momento, quedo de usted.

ATENTAMENTE

MSc. JUAN CARLOS DE LA TORRE AYALA
GERENTE DE INTEGRACION DE SISTEMAS

c c p. L.C. y M. en C. Marina Toriz
María Guadalupe Torres Solís
Jefe Exámenes Profesionales
Depto. Relaciones Exteriores
Archivo Integración de Sistemas SHP Systemhouse

Teléfono: (5) 227-5200 272-3626
Fax: (5) 271-7251

INTRODUCCION

El TRATADO DE LIBRE COMERCIO(TLC), el avance tecnológico y demás sucesos contemporáneos, han obligado a las empresas a buscar un CAMBIO, un CAMBIO en su forma de trabajo, que vaya de acuerdo a las exigencias del ya tan cercano siglo XXI.

Como solución a esto surge la RE-INGENIERIA de procesos de negocios, que es un enfoque para planear y controlar dicho CAMBIO y cuya meta principal es:

AUMENTAR LA CAPACIDAD PARA COMPETIR EN EL MERCADO MEDIANTE LA REDUCCION DE COSTOS, O SEA ALCANZAR LA VENTAJA COMPETITIVA, obteniendo un NUEVO AMBIENTE EMPRESARIAL, que se convertirá en el camino del EXITO.

Para realizar la RE-INGENIERIA DE PROCESOS DE NEGOCIOS, primero hay que POSICIONAR a la EMPRESA, esto es, determinar dónde hay que aplicarla, y para esto se consideran tres componentes significativos:

- el Personal
- LA TECNOLOGIA
- el proceso mismo.

que están interrelacionados y por lo tanto trabajan en forma paralela.

Con respecto a la TECNOLOGIA se implica a la TECNOLOGIA DE LA INFORMACION, ya que cualquier empresa depende en gran medida de los SISTEMAS DE COMPUTO, determinando que ésta es un factor importante para alcanzar ese CAMBIO.

Por otro lado, la RE-INGENIERIA DE PROCESOS DE NEGOCIOS, debe aplicarse cuando se presente cualquier oportunidad significativa, por ejemplo, cuando hay NUEVAS TECNOLOGIAS que pueden REDUCIR LOS COSTOS.

Sí, la situación que viven diversas empresas es que muchas de sus aplicaciones críticas(aquellas cuyo procesamiento y análisis es base para la TOMA DE DECISIONES) están centralizadas en MAINFRAMES o minicomputadoras, sin embargo, no se puede estar ajeno de los recursos tecnológicos existentes en el mercado actual, tales como: microcomputadoras, redes de área local, estaciones de trabajo, etc., etc., y que a menudo nos los ofrecen como más costeables y que además incrementan la eficiencia de dichas aplicaciones.

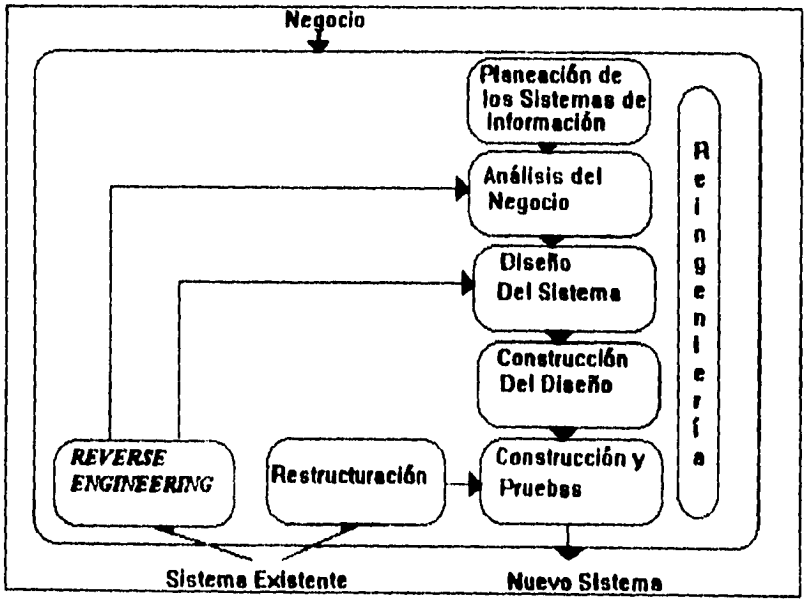
Todo esto ha hecho que los conceptos y tendencia del cómputo se modifiquen y ha motivado que la empresa tenga que determinar la ESTRATEGIA a seguir para optimizar sus operaciones.

Entre estas ESTRATEGIAS están: el DOWNSIZING, UPSIZING y RIGHTSIZING. Debido a esto, podemos ver que hay empresas que han instalado una variedad de plataformas entrando a lo que se conoce como SISTEMAS ABIERTOS, siendo su gran reto el que éstas trabajen de manera conjunta. SISTEMAS ABIERTOS es sinónimo de escalabilidad, portabilidad, interoperatividad, basándose en estándares.

Todo esto repercute rotundamente en toda la empresa.

Obviamente cualquier ESTRATEGIA de selección de hardware que se escoja, afecta ampliamente al SOFTWARE, pues de la misma manera hay que hacer una selección cuidadosa y a fondo de éste. Pero, hay que aclarar que para Tomar decisiones al respecto, es necesario hacerlo con una metodología, con un enfoque original y dinámico, y es aquí en donde entra la RE-INGENIERIA DE SOFTWARE, como solución a ésta problemática que enfrentan las empresas, por tanto cambio tecnológico de los recursos informáticos y que a diferencia de la RE-INGENIERIA DE PROCESOS DE NEGOCIOS no se basa necesariamente en la implantación de un nuevo sistema de tecnología de información(sólo que sea necesario), sino que en RE-INGENIERIA DE SOFTWARE, todo lo que se sabe que funciona, se aprovecha en el SISTEMA NUEVO, y todo lo que NO, se ignora.

El CONTEXTO de la RE-INGENIERIA DE SOFTWARE es:



En donde se puede observar que la REVERSE ENGINEERING es una parte de la RE-INGENIERIA DE SOFTWARE.

Con la INGENIERIA REVERSIVA se recupera lo esencial del diseño, la estructura y contenido de un SISTEMA DE APLICACION que SI corre y que inclusive se tienen buenos resultados para enseguida re-escribirlo.

Esto puede ser, porque se cambió de una plataforma a otra, o porque se necesita darle mantenimiento o porque se quiere una versión nueva.
Es decir, para que existe RE-INGENIERIA DE SOFTWARE, primero tiene que haber INGENIERIA REVERSIVA.

En base a esto justificamos nuestro SEMINARIO DE INVESTIGACION INFORMATICA, en donde se expone y aplica la REVERSE ENGINEERING(RE), cuya traducción al español es INGENIERIA REVERSIVA(IR) o INGENIERIA DE REVERSA(IR) por lo que durante todo el trabajo nos referimos a ésta en forma indistinta.

En cuanto a su alcance comprende una parte de investigación teórica, y otra, que es su aplicación práctica.

Agradecemos a SHL SYSTEMHOUSE DE MEXICO, S.A., de C.V., importante firma de consultoría a nivel INTERNACIONAL, por todo su apoyo para llevar a cabo la presentación de este trabajo.

La implementación de la INGENIERIA REVERSIVA, se realizó en uno de la gran cantidad de SISTEMAS DE APLICACION con que cuenta la EMPRESA. El SISTEMA en cuestión es SCORE, que significa SISTEMA DE CONTROL DE OPERACIONES Y REGISTRO ESTADISTICO, aplicable a CASAS DE CAMBIO y con el que se controla y maneja el ciclo completo de la operación cambiaria.

El ciclo comprende la promoción, operación, tesorería, ejecución, mensajería y control contable.

Las plataformas en que está implementado son PS, LANs, con los sistemas operativos OS2 y UNIX.

El desarrollo de dicho SISTEMA está en el MANEJADOR DE BASES DE DATOS ORACLE RDBMS, versión 6. Una de la ventajas de este producto es que cuenta con sus propias herramientas CASE, por lo que para la INGENIERIA REVERSIVA de dicho SISTEMA se utilizó a ORACLE CASE como una herramienta CASE INTEGRADA, ya que proporciona una gama completa de soluciones, métodos y herramientas, y en donde CASE *DICTIONARY , actúa en un nivel UPPER CASE, además de contar con mecanismos de alta seguridad para controlar a los usuarios que tienen acceso al Diccionario.

CASE*METHOD es una metodología comprobada a nivel mundial que guía a los proyectos a su culminación exitosa mediante la práctica de técnicas rigurosas de desarrollo de sistemas.

La arquitectura CASE de ORACLE proporciona una importante conexión al marco estructurado para guíar y controlar el desarrollo productivo durante todo el ciclo de vida del sistema.

Después de aplicar el proceso de INGENIERIA REVERSIVA a SCORE, se pudo observar el potencial de la herramienta ORACLE CASE, pues además de depurar la BASE DE DATOS eliminando tablas que no eran usadas que estaban desperdiciando espacio, en forma automática completó la información faltante, a nivel de definición de tablas, de índices. secuencias. También se obtuvo un Diagrama Entidad-Relación actualizado y confiable, y se generó la documentación faltante.

Para el logro de esto la organización del trabajo se encuentra de la siguiente manera:

CAPITULO 1. EL CONTEXTO DE LA INGENIERIA REVERSIVA :

En donde se abordan los orígenes y diferentes conceptos de la Ingeniería Reversiva,así como la importancia de usar Ingeniería Reversiva y su ubicación en la Reingeniería de software.

CAPITULO 2. HERRAMIENTAS PARA LA INGENIERIA REVERSIVA :

Se muestra a CASE ORACLE como herramienta para hacer Ingeniería Reversiva detallando su arquitectura, etapas , metodología, el soporte a usuarios, así como otras herramientas.

CAPITULO 3. APLICACION DE LA INGENIERIA REVERSIVA A LAS BASES DE DATOS:

Integrado por los pasos de la Ingeniería Reversiva , una guía para aplicarla y puntos de interés a considerar para su implementación en las Bases de Datos. Aquí se ve el impacto de la Ingeniería Reversiva .

CAPITULO 4. IMPLEMENTACION DE LA INGENIERIA REVERSIVA EN SCORE :

Es en este capítulo donde se prueba y comprueba el potencial de la Ingeniería Reversiva al aplicarla en el sistema SCORE (Sistema de Control , Operación y Registro Estadístico) que se usa para controlar el ciclo completo de la operación cambiaria.

CONCLUSIONES :

Como resultado de este estudio y su implementación.

APENDICES :

Cuyo fin no es empapelar, ni hacer volumen, sino mostrar el alcance y realidad del proceso.

APENDICE 1. Relación de tablas del sistema SCORE.
APENDICE 2. Reporte CDRF.lis
APENDICE 3. Pantallas de CASE * Dictionary
APENDICE 4. Reporte Entidad-Relación
APENDICE 5. Diagrama Entidad-Relación.

GLOSARIO : Para aclarar algunos términos y/o siglas utilizados en el texto de esta investigación y que se denotan con negritas.

REFERENCIAS: Los números encerrados entre "[ ]" indican una referencia bibliográfica definida al final del documento, para conocer el libro.

BIBLIOGRAFIA : Con el fin de ampliar los conocimientos al respecto, si así lo desea el lector.

CAPITULO 1

CONTEXTO DE LA INGENIERIA REVERSIVA CONCEPTOS Y UBICACION

CAPITULO I. EL CONTEXTO DE LA REVERSE ENGINEERING(RE).

1.1 ORIGEN Y CONCEPTOS DE LA REVERSE ENGINEERING(RE).

ORIGEN.

El término REVERSE ENGINEERING tiene sus orígenes en el análisis del hardware en donde la práctica de descifrar los diseños es común. En harware la REVERSE ENGINEERING, se usa para sabercomo está funcionando el sistema de cómputo, también es regularmente aplicada para mejorar la calidad de los productos, o bien para analizar los productos del competidor.

CONCEPTOS.

1. La REVERSE ENGINEERING(RE), cuya traducción al español es INGENIERIA REVERSIVA(IR) o INGENIERIA DE REVERSA(IR) es un conjunto de actividades para entender y modificar el software de los SISTEMAS. La idea principal, es identificar los componentes del SOFTWARE del SISTEMA EXISTENTE, la relación entre éstos y la creación de los altos niveles de descripción del software. También, RE se refiere al desarrollo de herramientas y técnicas para el análisis y representación de la documentación acerca del software de los SISTEMAS EXISTENTES.

2. La REVERSE ENGINEERING, es el proceso de obtener representaciones de alto nivel del código fuente y cabe aclarar, que no hay que confundir el trabajo de las herramientas de REVERSE ENGINNERING(RE) con las de REESTRUCTURACION, ya que éstas reestructuran los SISTEMAS EXISTENTES, pero producen código en el mismo lenguaje original, y que el analizar este código, no nos da una fotografía completa del diseño original, ya que la información se pierde después de la traslación del diseño en código, de una manera similar a como se pierde después de la compilación del código de un lenguaje de alto nivel y se obtiene el código binario. En la REVERSE ENGINNERING(RE) se recaptura el diseño, asi como algunos requerimientos de información.

3. La REVERSE ENGINEERING(RE) es el proceso de analizar programas existentes para entender lo que hacen. Esto se complementa con la producción de representaciones del programa en formas diferentes, partiendo del lenguaje de implementación.

Estas representaciones pueden tener una forma gráfica(estructuras gráficas), o bien, pueden ser un subconjunto de la información que presenta una vista particular del sistema, como pueden ser las dependencias de compilación.

Pero, hay que distinguir entre la RE-INGENIERIA y la REVERSE ENGINEERING(RE), aclarando que el producto de la REVERSE ENGINEERING debe de ser refinado por un humano y sometido al proceso de FORWARD ENGINEERING a través de una herramienta. Este proceso de REVERSE ENGINEERING(RE), seguido seguido por un FORWARD ENGINEERING(RE), se le denomina RE-INGENIERIA.

4. La SOFTWARE REVERSE ENGINEERING, es el proceso de analizar un sistema para identificar sus componentes e interrelaciones, asi como, el crear representaciones de éste en diferentes formas o en un alto nivel de abstracción.

Hay muchos tipos de información del programa que puede ser abstraída y examinada y si se trata de un SISTEMA VIEJO, resulta particularmente útil la abstracción y recuperación de su diseño.

1.2 LA REVERSE ENGINEERING Y RECUPERACION DEL DISEÑO.

Un diseño es recuperado juntando piezas de información que van desde el código fuente, documentación existente, experiencias personales con el sistema, hasta el dominio en el conocimiento de la información.

La recuperación del diseño difiere de la REVERSE ENGINEERING por el énfasis en recuperar el dominio del conocimiento, el cual ayuda a lograr abstracciones de alto nivel más alla de las que son creadas con el examen del código fuente, únicamente.

Ted Biggerstaff establece que:

Las abstracciones de recuperación de diseño deben incluir representaciones convencionales de ingeniería de software, tales como; especificaciones formales, división en módulos, abstracciones de datos, flujo de datos y descripciones del lenguaje del programa.

La recuperación del diseño debe producir toda la información requerida por una persona, para completar el entendimiento de lo que hace el programa, cómo lo hace, porqué lo hace, etc., etc..

Los diseñadores y desarrolladores de software modernos cuentan con técnicas y metodologías útiles derivadas de años de experiencia y observación que los guían en su trabajo. Las herramientas automatizadas han sido aplicadas en ambientes que soportan el desarrollo de software Estas herramientas engloban técnicas de ingeniería de software que soportan al desarrollo del software que está bien estructurado, entendible, de fácil mantenmiento, que contiene pocos errores y está documentado adecuadamente. En la recuperación del diseño de un sistema viejo, estas técnicas modernas pueden ser utilizadas para transformar su diseño y producir una mejor implementación. Esto es reingeniería de software.

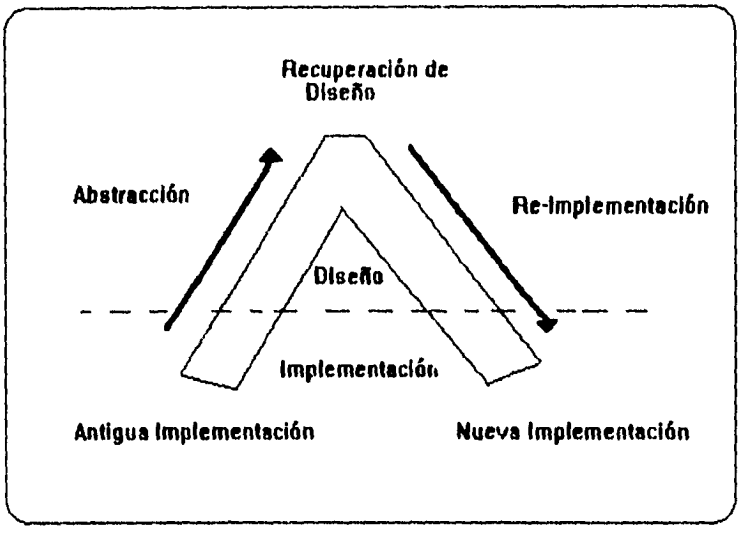


FIGURA 1. Proceso para llegar a una nueva aplicación.

En Ingeniería de Software, la RE, es usada para describir el proceso de descubrir cómo trabaja el SISTEMA.

1.3 UTILIDAD DE LA REVERSE ENGINEERING.

La REVERSE ENGINEERING sirve para

- obtener la información necesaria para la modificación racional de los SISTEMAS.
- recuperar la arquitectura de alto nivel de un SISTEMA.
- entender los detalles de los algoritmos usados.
- detección de errores.
- recuperar los componentes reusables y aprovechar su potencial posteriormente.
- transportar las políticas de una empresa en software.
- como auxiliar en la construcción de software de un SISTEMA NUEVO.

La actividad principal de la REVERSE ENGINEERING(RE) es recuperar la información de los SISTEMAS para mejorarlos. Pero también una aplicación muy especial es que la REVERSE ENGINEERING ocurre en el contexto y manejo de la empresa.

1.4 MANTENIMIENTO DEL SOFTWARE.

Para ANSI(American National Standard Institute). El Mantenimiento de Software es la modificación de un producto de software, después de haber sido liberado para su explotación, con el fin de corregir las fallas y mejorar el desempeño y otros atributos, o para adaptarlo a un ambiente al que ha sido cambiado, pero si este mantenimiento no se lo dan los diseñadores, se gastan muchos recursos para examinar y aprender acerca del sistema.

Las herramientas de la REVERSE ENGINEERING pueden facilitar esta práctica, por lo que aqí la REVERSE ENGINEERING, es la parte de un proceso de mantenimiento que ayuda a entender el sistema para que se puedan hacer los cambios necesarios y apropiados.

1.5 CICLOS DE VIDA Y ABSTRACCIONES.

Para definir adecuadamente la noción de software y de REVERSE ENGINEERING(RE), es necesario aclarar tres conceptos dependientes:

* La existencia de un modelo de ciclo de vida.
* La presencia de un sistema en cuestión.
* y la identificación de los niveles de abstracción.

Asumimos que existe el modelo de ciclo de vida del sistema en el proceso del desarrollo del software, y además que sea una interacción entre las etapas del ciclo de vida y quizá entre la repeticiones.

El sistema en cuestión puede ser un programa simple, el código de un programa , o un complejo conjunto de programas de interacción o de archivos de datos.

En un modelo de ciclo de vida las primeras etapas tratan conceptos independientes de implementación y las etapas posteriores enfatizan los detalles de ésta.

Con RE, Premerlani y Blaha exploraron el análisis de la estructura de una Base de Datos Relacional existente para identificar y entender los modelos de datos en una notación de objetos.

Ning, Engberts y Kozaczynski, describen una técnica para identificar y extraer piezas funcionales de un programa en Cobol, demostrando con esto la aplicación comercial de este proceso.

1.6 Términos relativos a la Ingeniería Reversiva.

Para una mayor comprensión se describen algunos términos relacionados con la técnica Reverse Engineering, usando solo tres etapas identificadas en el ciclo de vida de un sistema con diferentes niveles de abstracción.

1.-Requerimientos:

Especificación del problema siendo resuelto, incluyendo objetivos y restricciones reglas de negocios.

2. Diseño. Especificación de la solución.

3. implementación.- Generando y probando código de un sistema operacional.

Forward Engineering (Ingeniería hacia adelante).

El proceso tradicional de mover de un nivel superior de abstracción y lógico, los diseños de implementación independiente a la implementación física de un sistema. Se utiliza como se ve la terminología de diseño y desarrollo y se introduce un nuevo término, el adjetivo "Forward". Es necesario entender este término para distinguir el proceso de Reverse Engineering.

Forward Engineering sigue una secuencia que va desde los requerimientos a través del diseño hasta la implementación del sistema.

Reverse Engineering (Ingeniería Reversiva).

Reverse Engineering es el proceso para analizar un sistema , a través de:

1.- Identificar los componentes del sistema y sus interrelaciones.
2.- Crear representaciones del sistema en otra forma en un nivel superior de abstracción a otro nivel.

Reverse Engineering generalmente extrae artefactos de diseño y construcción o sintetiza abstracciones que son menos dependientes de la implementación.

Reverse Engineering puede aplicarse a un sistema existente, pero también puede desarrollarse empezando desde cualquier nivel de abstracción o en cualquier estado del ciclo de vida.

Reverse Engineering no involucra el cambio del sistema o el crear un nuevo sistema basado en el Reverse Engineering de un sistema ya hecho. Este es un proceso de examinación, no es un proceso de cambio o reaplicación de un sistema.

Reverse Engineering empieza desde la implementación existente, recapturando o recreando el diseño y descubriendo los requerimientos actualmente implementados por el sistema existente.

Existen varias subtareas del Reverse Engineering las cuales son : Redocumentación y Design recovery.

Redocumentation (Redocumentación).

El prefijo "RE" implica el intento de recobrar la documentación que exista acerca del sistema o mostrar o analizar la ya existente.

Es la creación o revisión de una representación semánticamente equivalente dentro del mismo nivel de abstracción. Las formas resultantes de representación son usualmente consideradas como vistas (por ejemplo diagramas de flujo, estructuras de datos y control de flujo).

La redocumentación es la más simple y vieja forma del Reverse Engineering y muchos la consideran como una forma débil de reestructuramiento.

Algunas herramientas comunes usadas para el desarrollo de la redocumentación son pequeñas impresiones (que despliegan el listado del código) , Generadores de diagramas (con los que se crean diagramas directamente del código, control del flujo o estructura del código) y manuales de referencia. El objetivo de estas herramientas es proporcionar caminos fáciles para visualizar las relaciones entre los componentes del programa, para reconocerlos y seguir sus caminos claramente.

Desing Recovery (Recuperación de Diseño).

El recuperamiento de Diseño es un subconjunto o subtarea del Reverse Engineering, en la cual domina el conocimiento, la información externa, y la deducción del razonamientos son agregadas a las observaciones del sistema para identificar los niveles de abstracción superiores significativos mas allá de esos que se obtienen directamente examinando el sistema mismo.

Design Recovery es distinguido por los orígenes y alcances de la información.

Recrea el diseño y abstracción desde una combinación de código de un diseño, obteniendo documentación disponible, experiencia personal y conocimiento general acerca del problema y la aplicación.

Debe reproducir toda la información requerida por una persona para el entendimiento completo, de lo que hace el programa, por eso trata de cubrir un rango más amplio de información que no es encontrado convencional software Engineering.

Restructuring (Reestructuración).

Es la transformación de una forma de representación a otra forma de representación de un mismo sistema relativo de abstracción. Preservando el sistema en cuanto a funcionalidad y semántica.

Una transformación de reestructuración es a menudo una de apariencia tales como los códigos alternados para mejorar su estructura en el sentido tradicional del diseño estructurado. El término reestructuración viene del uso popular de transformación de código a código que reconstruye las técnicas en remodelar los modelos de datos, planos de diseño, y requerimientos de estructuras. Muchos tipos de reestructuracion pueden ser ejecutados con un conocimiento de forma estructural pero sin un entendimiento de significado . Por ejemplo puedes convertir un conjunto de sentencias "if" en una estructura CASE o viceversa, sin conocer el propósito del programa o algo acerca del dominio del problema.

Mientras que la reestructuración crea nuevas versiones que implementan o proponen nuevos cambios al sistema este normalmente no envuelve modificaciones, debido a los nuevos requerimientos.

Reestructuración es a menudo usada como una forma de mantenimiento preventivo para mejorar el estado físico del sistema, con respecto a algunos estándares referenciados.

Reengineering (Re-ingeniería).

Se conoce como renovación, es la examinación y alteración de un sistema para reconstruirlo en una nueva forma y la subsecuente implementación de la nueva forma . Reengineering generalmente incluye algunas formas de Reverse Engineering (para lograr una descripción mas abstracta), seguidas por alguna forma de Forward Engineering o Reestructuring. Esta debe incluir modificaciones con respecto a nuevos requerimientos no incluidos en el sistema original. Por ejemplo, durante la Reengineering del sistema de manejo de información , una organización generalmente evalúa como el sistema implementa reglas de negocios de nivel superior y hace modificaciones para conformar los cambios en los negocios del futuro.

Hay alguna confusión entre los términos particularmente entre Reengineering y Restructuring.

IBM define "la aplicación de Reengineering" como el proceso de modificación los mecanismos internos de un sistema o programa, las estructuras de datos de un sistema sin cambiar su funcionalidad; en otras palabras es alterado el cómo sin afectar el qué del sistema. El Reengineering usa las tecnologías de Reverse Engineering disponibles, englobando al forward y Reverse Engineering.

El propósito primario del REVERSE ENGINEERING de un sistema de software, es incrementar la comprensión total de un sistema para el mantenimiento y el nuevo desarrollo.

La figura 2 muestra la relación entre los términos: Reverse Engineering, Restructuring y Reengineering durante el proceso de etapas de trabajo para llegar a un nuevo sistema.

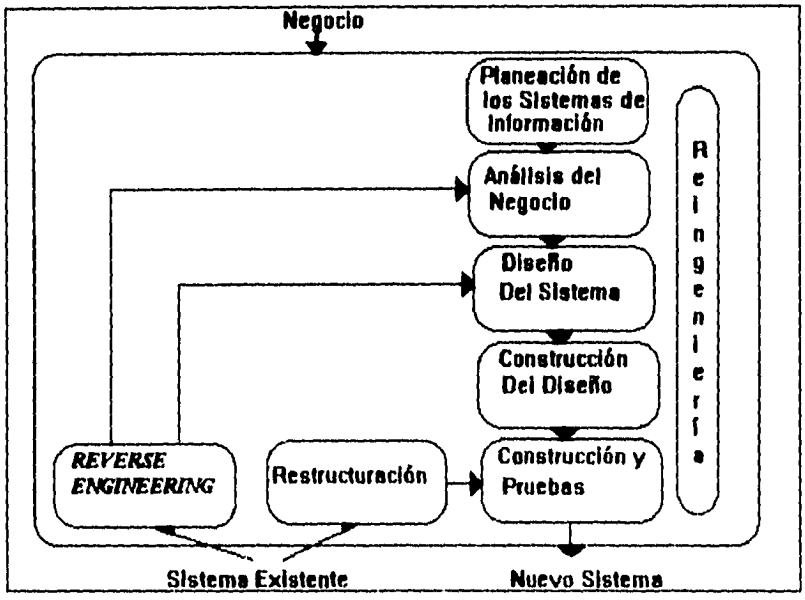


FIGURA 2. El proceso de Re-ingeniería y terminos relacionados

1.7 POR QUE USAR REVERSE ENGINEERING

El problema de reimplementar un sistema existente en un lenguaje de programación diferente ha girado por años en tres grandes alternativas :

1. Reescritura manual de los sistemas existentes.

2. Uso de un convertidor automático de lenguaje.

3. Rediseñar e implementar el sistema.

La primera proposición, reescritura manual de un sistema existente, significa una conversión manual de un lenguaje fuente a un lenguaje objeto. En esta proposición no se requieren nuevas herramientas de software. Hay flexibilidad en términos de convertir el sistema y cambiar su estructura. Sin embargo hay varias desventajas ya que:.

La conversión manual de un código fuente muchas veces mantiene el estilo y esencia de la implementación original. Las oportunidades de cambiar el sistema no están planeadas con regularidad. Los programadores individualmente tienden a rehacer su trabajo en lugar de planear los cambios en adelante. Esta propuesta es consumir tiempo. El número de líneas de código que pueden ser convertidas manualmente por programador al día es pequeño.

Finalmente esta propuesta está propensa a error, ya que los humanos cometemos errores. No hay algo que garantice que es el sistema reescrito es igualmente funcional al original. El nuevo sistema debe ser probado con exactitud para poder tenerle confianza .

La segunda propuesta, la conversión automática, estriba en el uso de una herramienta que acepte software escrito en un lenguaje de programación fuente y generar un nuevo código fuente escrito en el lenguaje objeto. El proceso de conversión se lleva a cabo con muy poca o ninguna intervención humana, por esta razón se evitan muchos de los problemas que surgen en la propuesta de conversión manual. Esta propuesta genera nuevo código rápidamente . Sin embargo, tiene varias desventajas. El lenguaje fuente no puede producir por sí solo, la conversión del lenguaje objeto elegido.

Algunas herramientas automáticas de conversión de código solamente hacen la parte 'fácil' y las partes 'difíciles' se las dejan al humano. Los convertidores automáticos tienden a enfocar exclusivamente transformaciones de lenguaje y no direccionan el problema de las modificaciones en la estructura

del programa. El problema que comunmente se presenta con la conversión literal se puede resumir en : información incorrecta dentro, información incorrecta fuera. Si el sistema existente no está bien estructurado, en términos de su arquitectura y control de flujo, entonces el sistema resultante será de la misma calidad que el anterior.

La generación automática de código puede ser terriblemente ineficiente. El código producido por la conversión puede ser también difícil de entender e incrementar grandemente sus futuros costos de mantenimiento.

La conversión automática de lenguaje puede no proveer de una ruta fácil para completar la conversión a un nuevo sistema.

La tercera propuesta es el rediseño y la reimplementación del sistema. Esta comienza con los requerimientos para el sistema actual y construye un sistema completamente nuevo en el lenguaje objeto. Se requiere que el nuevo sistema sea equivalente funcionalmente con el sistema original, a pesar de ser derivado de éste . De las tres proposiciones esta es la que ofrece la mejor oportunidad para producir un sistema nuevo mejor. Rediseñar y reimplementar el sistema en un nuevo lenguaje nos da poder y flexibilidad en términos de llegar a un producto final. El sistema resultante puede tener costos de mantenimiento significativamente más bajos que los sistemas generados con las otras propuestas. Finalmente, rediseñando el sistema existente para su implementación en el lenguaje objeto permite tener un mejor uso de las características de ese lenguaje.

Sin embargo, esta propuesta también tiene sus desventajas .Es más difícil que hacer un diseño original por el requerimiento de emular las interfaces del sistema existente. Esta propuesta tiene un costo inicial altísimo, su equivalente será construir un nuevo sistema. La desventaja más seria es que para muchos sistemas no es posible rediseñarlos desde los requerimientos del sistema, cuando los requerimientos pueden no existir.

Para muchos sistemas viejos solamente las declaraciones correctas de las capacidades del sistema y funcionalidad es con frecuencia el código fuente por sí mismo. Muchas veces no hay especificaciones de requerimientos válidos para el sistema.

Reverse Engineering es una propuesta adicional a las tres anteriores. Si no hay requerimientos de especificación para un sistema, aplicando Reverse Engineering al sistema se puede producir un diseño que capture la funcionalidad del sistema. El diseño debe ser representado en un nivel de abstracción que borre las dependencias del lenguaje de implementación. Esto facilita la reimplementación del sistema en un nuevo lenguaje . En suma, el diseño reconstruido puede ser transformado para modernizarlo, reestructurarlo, incorporar nuevos requerimientos, etc. Esto es software reengineering (reingeniería de software) .Así la reingeniería basada en un proceso de Reverse Reegineering ofrece muchas de las ventajas de la propuesta del rediseño y la reimplementación. Esta propuesta siempre es factible en un código fuente para un sistema existente.

Es ésta una solución efectiva en costo?. Está establecido que el soporte de herramientas debe ser necesariamente hábil para aplicar efectivamente la Reverse Engineering en sistemas grandes. Un proyecto aplicó Reverse Engineering en un sistema de 24,000 líneas de código para reespecificarlo y redocumentarlo.

Con la herramienta de soporte fué posible analizar estáticamente y documentar el sistema en una semana. Fué estimado que podría haber tomado tres años-hombre documentar el sistema manualmente con el mismo nivel de detalle . La información generada fué usada para reespecificar el sistema. La reespecificación tomó 17 meses- hombre para completarlo. Esto ascendió a un mes-hombre para la especificación de 1400 líneas de código. En otro proyecto, 100,000 líneas de código repartidas en 60 programas fueron rediseñadas por la reconstrucción manual de representaciones de diseño desde el código fuente. En suma, 52,000 líneas de código del nuevo software fueron escritas. El sistema original totalizó 1.5 millones de líneas de código.

Este proyecto se completó en 21 meses.

1.8 UBICACION DE INGENIERIA REVERSIVA DENTRO DE LA REINGENIERIA DE SOFTWARE.

La Reverse Engineering dentro de la reingeniería ha tenido gran auge en los últimos años, debido a que dicha técnica forma parte de la reingeniería.

La tecnología de reingeniería de software tiene 3 objetivos:

1) entender el software,

2) actualizar el software y captura, y

3) ampliar el conocimiento acerca del software.

A continuación se define el término REINGENIERIA:

La reingeniería:

- INVOLUCRA EL ENTENDIMIENTO DEL SOFTWARE.

- PREPARA O INVOLUCRA AL SOFTWARE, GENERALMENTE INCREMENTA EL MANTENIMIENTO, REUSABILIDAD O EVOLUTIVIDAD.

En esta definición el software incluye (código fuente, documentación, gráficas y análisis). El análisis acerca del código fuente, diseño, especificaciones, prueba de datos, y otros documentos que soportan directamente al desarrollo de software o mantenimiento.

La Reverse Engineering entra en la definición anterior de reingeniería. La Reverse Engineering genera información acerca de la representación del software (como el código fuente) ayuda a entender o facilita su procesamiento, incluye las actividades de rastreo, medición, dibujo, bosquejos del software, documentación y análisis.

1.9 OBJETIVOS ESPECIFICOS DE LA REINGENIERIA.

Los propósitos especificos de reingeniería son los siguientes:

-perfeccionamiento o mejora

-renuevo

-renovación

-restauración

-modernización

-redesarrollo de ingeniería

-ingeniería de reuso

Los términos mejora, renuevo, renovación, restauración y redesarrollo de ingeniería todos tienen un significado similar: mejorar el software para evolucionar su uso futuro.

Modernización, incluye mejora de software, pero debe ir más allá de improvisar el desarrollo de software y actividades de mantenimiento del software. Reclamación e ingeniería de reuso se refiere a la reingeniería para hacer el código fuente más reusable.

1.10 EL CONTEXTO DE LA REINGENIERIA

1) las vistas de software

2) las bases de la información

3) descomposición

4) composición y

5) transformación

A continuación se describe cada uno de éstos:

1) LAS VISTAS DE SOFTWARE:

Una vista es una representación de software o un reporte acerca de éste. Una vista en software debe ser o no visualizada por un humano, pero esto es típicamente un significado intermedio de la representación de software que los humanos deben y quiere ver. La palabra vista se refiere al tipo de vista (por ejemplo: un diagrama de flujo). La Información de la vista significa la información específica en una vista (por ejemplo: el diagrama de flujo específico D), o la información de la base de conocimientos descompuestos de la información de la vista.

Ejemplos de vistas en software son las especificaciones, el código fuente, dimensiones del software , reportes derivados del análisis del código fuente estático, y prueba de los datos usados para caracterizar el software. Cuando una vista es soportada por una herramienta, rápidamente viene junto con la vista del editor, para soportar las entradas, rastreo, y cambios en la información de la vista.

2) LAS BASES DE LA INFORMACION:

Las bases de la información es el déposito de información acerca del software. Esto puede ser llamado de tres formas:

2.1) descomposición del software de los objetos y relaciones.

2.2) incremento en la construcción de objetos y relaciones conociendo la construcción de las herramientas o suma de los conocimientos en la base de información, e

2.3) importancia de la información de otras bases.

3) DESCOMPOSICION:

Es el proceso de transformación de las vistas en objetos y las relaciones almacenadas en la base de información. Por ejemplo, los compiladores comunes descomponen los programas en un árbol de representaciones abstractas.

4) COMPOSICION:

Genera información de las vistas de la información de la base. El compositor (la herramienta o la persona que hace la composición) ensambla la información de la vista para construir los objetos relevantes y relaciones en la información de la base, sumando el formato de la vista como necesidad para desplegar la información de la vista. Por ejemplo, el back-end para compilar comúnmente la generación de código por medio de una gráfica semántica del programa, o algo equivalente.

5) TRANSFORMACION

La transformación de reingeniería, es la información de una vista de software en la información de otra vista, que son de la misma clase.

La reingeniería transforma generalmente la información en la vista utilizada. La reestructuración de software es reingeniería central para transformar la estructura del código fuente (sintaxis y semántica).

La transformación está por debajo de la Reverse Engineering y la recuperación del diseño. La Reverse Engineering está ligado a la reingeniería, excepto el origen y esquema de las vistas son diferentes, el esquema de la vista empieza en una clase de la vista. Transformando código fuente, por ejemplo, los caracteres en la estructura deben ser reingeniería o Reverse Engineering. Pero la transformación del código fuente para reestructurar el código fuente es reingeniería o restructuración, no Reverse Engineering. Actualizar los comentarios del código fuente es reingeniería.

La Reverse Engineering tiene otro significado. Por ejemplo, esto puede significar el análisis de una representación detallada para descubrir un diseño inherente.

Algunas veces la gente utiliza el Reverse Engineering para determinar el código fuente del código objeto.

La recuperación de diseño, es un subconjunto de Reverse Engineering, la cual genera información acerca del software. Otra información no podría ser fácil de extraer del software y documentación asociada, y requiere considerables efectos para deducirlos.

Tres ejemplos son la generación de descripciones racionales de porque el software es una forma de generación de las especificaciones de código fuente, y generación de la caja negra del conjunto de prueba de datos para el software, así como la documentación que es generada.

El Forward Engineering: es una transformación, generalmente de una clase de la vista. Por ejemplo, la generación de código fuente de un diagrama de flujo es generalmente actividad del Forward Engineering.

Reingeniería: puede ser considerada como la transformación de la vista.

1.11 IMPORTANCIA DE LA REINGENIERIA

La reingeniería es importante por varias razones:

1) LA REINGENIERIA PUEDE AYUDAR A REDUCIR PROBLEMAS EN LA EVOLUCION DE LA ORGANIZACION.

Para extender la capacidad de software, la organización puede construir nuevo software, involucra la existencia de software, reingeniería y existencia de software, el uso de la generación de aplicaciones o la obtención de partes de software o paquetes. Cuando las dos opciones no son disponibles, las organizaciones optan por la construcción de un nuevo software o involucrar nuevas rutinas del software ya existente.

Simplemente involucra la existencia de código fuente, que aparece en una práctica, para hacer el software de acuerdo a los cambios. La construcción de software puede ser cara y poco confiable.

El software de reingeniería y los cambios, pueden ayudar al software de la organización mejor que la construcción de software o simplemente involucra el mantenimiento tradicional.

2) LA REINGENIERIA PUEDE AYUDAR A LA ORGANIZACION A RECUPERAR INVESTIGACIONES DE SOFTWARE.

Las compañías han invertido miles y millones de dólares en la construcción de software. En el software industrial se han invertido billones de dólares. Ignorando cada vez más la existencia de software, es por esto que las compañías pueden usar la reingeniería para participar en la recuperación de software. La reingeniería ayuda a la organización a construir su propio software.

Simplemente involucra la existencia de código fuente, que aparece en una práctica, para hacer el software de acuerdo a los cambios. La construcción de software puede ser cara y poco confiable.

El software de reingeniería y los cambios, pueden ayudar al software de la organización mejor que la construcción de software o simplemente involucra el mantenimiento tradicional.

2) LA REINGENIERIA PUEDE AYUDAR A LA ORGANIZACION A RECUPERAR INVESTIGACIONES DE SOFTWARE.

Las compañías han invertido miles y millones de dólares en la construcción de software. En el software industrial se han invertido billones de dólares. Ignorando cada vez más la existencia de software, es por esto que las compañías pueden usar la reingeniería para participar en la recuperación de software. La reingeniería ayuda a la organización a construir su propio software.

3) LA REINGENIERIA PUEDE AYUDAR A FACILITAR LOS CAMBIOS DE SOFTWARE.

El software puede dejar grandes dividendos. Esto depende de la productividad del mantenimiento del programador para trabajar o hacer el código mas fácil de entender. Esto da a la organización mayor flexibilidad porque este software puede ser modificado mas rápidamente para acomodar los cambios en el negocio. La reingeniería da más opciones a la organización de mejorar el software de acuerdo a sus necesidades.

4) LA REINGENIERIA AGRANDA LOS NEGOCIOS.

La mercadotecnia de 1990 para servicios de reingeniería y herramientas había sido estimada en billones. Como reporte, la estimación de servicios de reingeniería en 1990 fue de $4.6 billones. Para 1995 la estimación es de $11.9 billones. La estimación se extiende para productos de reingeniería, incluyendo el back-end de la herramienta CASE, la cual fue de $.8 billones en 1990. Para 1995 la estimación es de $2.7 billones. Muchos sistemas de software, y partes de sistemas, necesitan ser actualizados. Los proveedores de software y compañías de servicios han trabajado en esta área. Muchas organizaciones son vistas por las técnicas de reingeniería, herramientas y el uso de procesos.

5) LA REINGENIERIA EXTIENDE LA CAPACIDAD DEL CONJUNTO DE HERRAMIENTAS CASE.

La reingeniería ayuda en las técnicas nuevas y herramientas para ser aplicadas al software viejo. Esto ha traído beneficios. Para el mantenimiento, esto sigue y con las herramientas mas poderosas ayudan al mantenimiento del software. La reingeniería debe ser vista como un tipo de vehículo transferible de tecnología, siguiendo los viejos sistemas para ser actualizados en los esquemas nuevos de trabajo, por medio del poder de las nuevas herramientas.

Para vendedores de herramientas CASE, suman el conjunto de herramientas de reingeniería para abrir nuevos mercados en la existencia de software. Esto ha dado importantes cambios que son sumados al Forward Engineering solo en el conjunto de herramientas.

6) LA REINGENIERIA ES UN CATALOGO PARA AUTOMATIZAR EL MANTENIMIENTO DE SOFTWARE.

Una parte importante de la información almacenada en la base de información es valuada para ser analizada, automatizada y buscar el mantenimiento de software. La información contenida en el déposito puede ser automatizada tomando la información que aparece en la base de información.

7) LA REINGENIERIA ES UN CATALOGO PARA APLICAR LAS TECNICAS DE INTELIGENCIA ARTIFICIAL (IA) PARA SOLUCIONAR PROBLEMAS DE REINGENIERIA DE SOFTWARE.

Históricamente, el campo de la transformación automática es un conocimiento fuera de serie en trabajos de Inteligencia Artificial (IA), así como las reglas-basadas en la producción de sistemas y procesamiento de lenguajes. La transformación crece de forma lógica y matemáticamente sobreescribe sistemas.

La reingeniería ha suplido a la Inteligencia Artificial trabajando en campo aplicado a sus trabajos.[RICH 90], [HARTMAN 91].

La reingeniería tendrá significantes cambios durante los próximos 5 o 15 años. Las experiencias tenidas con aplicaciones de reingeniería provee varias maneras para actualizar el desarrollo y mantenimiento de software.

1.12 TECNOLOGIA DE REINGENIERIA PARA MEJORAR EL SOFTWARE

Estos son algunos conceptos bajo la tecnología de la ingeniería de software los cuales son: actualizar el software, entendimiento de software y captura, perseverancia, y existencia de conocimientos acerca del software. Como se muestra a continuación::

1) TECNOLOGIA PARA EL PERFECCIONAMIENTO DE SOFTWARE:

1.1 reestructuración

1.2 redocumentación, anotaciones y actualización de documentación.

1.3 Reingeniería de reuso

1.4 sistemas heredados

1.5 remodularización.

1.6 reingeniería de datos.

1.7 reingeniería en el proceso de negocios.

1.8 análisis de mantenimiento, análisis de portafolio, y análisis económico.

2) ENTENDIMIENTO DE SOFTWARE

2.1 rastreo.

2.2 análisis y medición.

2.3 Reverse Engineering y recuperación de diseño.

3) CAPTURA Y EXTENSION DE CONOCIMIENTOS ACERCA DEL SOFTWARE.

3.1 descomposición.

3.2 Reverse Engineering y recuperación de diseño.

3.3 recuperación de objetos.

3.4 entendimiento de programas.

3.5 bases de conocimientos y transformaciones.

A continuación se explica cada uno de éstos :

1) TECNOLOGIA PARA PERFECCIONAR EL SOFTWARE

1.1) REESTRUCTURACION DE SOFTWARE:

La reestructuración del software es la modificación del software para hacer más fácil el entendimiento o mantenimiento. El término connota cambios en el código fuente de la estructura de control. La Reestructuración es significativa, porque ésta es una de las más viejas y más refinadas técnicas de reingeniería. La Reestructuración fué una de las primeras tareas de la reingeniería automatizada.

Desarrollos sobre reestructuras automatizadas han visto la manera de utilizar otras herramientas de reeingeniería.

1.2) REDOCUMENTACION, ANOTACIONES, Y ACTUALIZACION DE LA DOCUMENTACION:

El software redocumentado es la creación de la actualización, acerca de la información correcta de software. El código redocumentado es una transformación del código (y otros documentos y conocimiento de programas) en una nueva o actualizada documentación acerca del código.

Normalmente esta documentación es textual (por ejemplo comentarios), pero esto puede ser gráfico.

La documentación (comentarios, diseños y especificaciones) es una de las técnicas más viejas de reingeniería. La redocumentación es importante porque mantiene bien comentado el código del programa como una guía para posteriores modificaciones al código.

Anotando las connotaciones añadidas a la documentación al código fuente pocas veces usada. Las anotaciones son importantes para entender el código ensamblador, el cual es un trabajo informativo sobre la redocumentación. El éxito de la redocumentación depende de la herramienta que se utilice para automatizarlo.

1.3) INGENIERIA DE REUSO:

El reingeniería de reuso es la modificación de software para hacer más reusable las partes del software y reconstruirlas para poder utilizar las librerías.

1.4) SISTEMAS HEREDADOS:

Son un subcampo para la ingeniería de reuso que descompone el sistema existente en objetos y relaciones que hacen ser reensambladas (reuso) en nuevos sistemas. El término 'heredar sistemas' también se refiere a la duración del sistema.

1.5) REMODULARIZACION:

La remodularización de software es el cambio de la estructura modular del sistema. Esto depende del análisis de **clusters** de los componentes característicos del sistema y su acoplamiento.

1.6) REINGENIERIA DE LOS DATOS:

La reingeniería de los datos mejoran el contenido del sistema. Esquemas que deben ser reorganizados y actualizados, esquemas múltiples, deben ser consolidados en un esquema, entradas en el diccionario de datos, deben ser hecha la consistencia de semánticas y la validación de datos debe ser borrada.

La reingeniería de los datos es una cuestión de otra tarea, como una migración de datos de otra administración de base de datos del sistema.

1.7) REINGENIERIA DE PROCESOS DE NEGOCIOS:

La arquitectura de software es flexible, así como la posibilidad tecnológica de automatizar la información, esto hace al software del negocio más ajustable al software que se tenga. Experiencias sobre el poder de productividad puede involucrar algunas veces la retención del proceso automatizado del negocio para el software. Esta retención debe resultar en el diseño del nuevo software que puede ser la base para la reingeniería, migración o evolución del software del sistema.

1.8) ANALISIS DE MANTENIMIENTO, ANALISIS DE PORTAFOLIO, Y ANALISIS ECONOMICO:

El análisis de mantenimiento del software es importante para descubrir cuantas partes del sistema deben ser de reingeniería. Típicamente la mayoría del trabajo de mantenimiento es concentrado en módulos relativos del sistema. El análisis de mantenimiento ayuda para localizar el mantenimiento de las partes altas del sistema. Estas partes tienen un impacto inicial más grande en el costo del mantenimiento.

[PERCY 81] define una metodología para determinar el mantenimiento del programa. Más recientemente, [OMAN92a] y [OMAN92b] describen que el mantenimiento es una métrica para esto. [SNEED91] discute el uso de los modelos costo-beneficio, y [CONNELL87] describe criterios sobre reingeniería.

2) TECNOLOGIA PARA ENTENDER EL SOFTWARE

2.1) BROWSING (RASTREO).

El rastreo del software, es como un editor de texto, es uno de los más viejos significados para el entendimiento de esto. Recientemente el rastreo ha venido avanzando, con el uso del hipertexto [CONKLIN87] para hacer conección entre partes relatadas y vistas múltiples del sistema [CLEVELAND89] para proveer diferentes vistas solo con el click del mouse.

2.2) ANALISIS Y MEDICION:

El análisis y medición son tecnologías importantes para entender las propiedades del programa, así como su complejidad. Técnicas relevantes para la reingeniería son programas divididos [WEISER81], así como, el control de la

medición de la complejidad del flujo [McCABE76] y el acoplamiento de medición [MYERS75].

2.3) Reverse Engineering, RECUPERACION DE DISEÑO:

Como indica, el Reverse Engineering y la recuperación de diseño generan nueva información acerca del software, generalmente en una vista diferente. Esta tecnología ha venido a ser muy popular, pero determina algunos tipos de información del diseño (diseños racionales), pero esto es todavía un componente de riesgo.

Más comúnmente, el Reverse Engineering genera estructuras de caracteres o diagramas de flujo de datos del código fuente. Esta herramienta es confiable, debido a que se puede disponer de la información rápidamente o analizar el mismo código. En Enero de 1990 la IEEE SOFTWARE mostró una buena colección de información sobre el Reverse Engineering y recuperación de diseño.

3) CAPTURA Y EXTENSION DE CONOCIMIENTOS ACERCA DEL SOFTWARE:

3.1) DESCOMPOSICION:

La descomposición de programas toma un programa y crea objetos y relaciones fuera de este. Estos objetos y relaciones son almacenados en la base de información. Los objetos y relaciones facilitan el análisis, medición, transformación y extracción más allá de la información. Trabajando en la descomposición más directamente, el código fuente guarda el trabajo para tener las partes del programa y crear los objetos y relaciones para el uso de la herramienta. Esta tarea, es la más confiable por la mayoría de los lenguajes por el tiempo-consumible para solucionar los problemas. Para la mayoría de los lenguajes esto es más fácil para la descomposición de programas.

3.2) RECUPERACION DE OBJETOS:

La recuperación de objetos obtiene objetos del código fuente. Esto sigue una de las vistas previas al código fuente no orientado a objetos en una forma orientada a objetos. La orientación de objetos (clases, herencias, métodos, abstracción de tipos de datos, etc) debe ser parcial o completa. La migración del código fuente de formas orientado a objetos ha sido recientemente estudiado. Una estructura de programa orientada a objetos debe ofrecer más programas entendibles, migratorios y la reducción de posibilidades del código no orientado a objetos. Esto ha sido una experiencia previa con la orientación de objetos cuando los sistemas se convierten de C a C++ [BREUER91].

3.3) ENTENDIMIENTO DE PROGRAMAS:

El entendimiento de programas toma diversas formas. Una es la técnica manual o automática para programadores para el mejor entendimiento del software. La otra es el cuerpo de trabajo que almacena información acerca de la programación y usos de la información para encontrar las instancias del conocimiento del código de programación. El entendimiento es evidente para extender el software con las herramientas de programación.

3.4) BASES DE CONOCIMIENTOS Y TRANSFORMACIONES:

Las transformaciones de programas son fundados hoy en día por la tecnología de reingeniería. La base de la información, asocia la transformación y programas transformados manejando el poder por la herramienta de reingeniería. La transformación de este trabajo en un programa gráfico almacena objetos gráficos en la base de conocimientos. La base de los objetos, transforman la arquitectura para las herramientas de reingeniería, con el fin de construir nuevas herramientas de reingeniería.

1.13 ESTRATEGIAS DE REINGENIERIA Y RIESGOS DE MIGRACION

La tecnología de reingeniería es tan importante como el conocimiento que se tenga de la misma.

A continuación se abarcarán las estrategias y riesgos que pueden ser inherentes en la migración de sistemas en la reingeniería:

- PROCESO DE REINGENIERIA:

El proceso de reingeniería toma muchas formas, dependiendo de sus objetivos. Los objetivos son limpieza de código, redocumentación, migración, captura de información en una base de información y reuso del código de reingeniería.

Algunos casos de estudio son fuentes frecuentes del proceso de reingeniería. Por ejemplo, [SOLVIN91] describe el mejoramiento de la estructura modular y mantenimiento de un sistema que tiene alto costo de mantenimiento.

[BRITCHER90] describe un proyecto de reingeniería en el cual la Administración Federal de Aviación tuvo el control del sistema con la reingeniería para operar en la mayoría de los modems del hardware en Pascal (instalado en código ensamblador).

Otros tienen que sistematizar el proceso de reingeniería. El proceso mejor para esto son los pasos de análisis de inventarios, posicionamiento y transformación. La fase de inventario y análisis establece la base de los componentes del software y evalúa las opciones de reingeniería como base en el inventario. Implementando mejoras en la calidad del software afectando su existencia funcional o la arquitectura. El mejoramiento facilita los cambios o análisis para soportar los cambios. La fase de transformación crea una nueva arquitectura de la existencia de la primera.

- EVALUACION DE REINGENIERIA:

Esto habla acerca de los efectos empíricos de la reingeniería. La evidencia para la reingeniería cuenta anécdotas, casos de estudio, aprendizaje de lecciones y experimentos. Una buena discusión del análisis de costo-beneficio para la reingeniería puede ser encontrada en estudios realizados por Sneed en 1991. Experimentos sobre la reestructuración de software y reingeniería, han sido

apoyados. Los últimos casos de estudio [SOLVIN91] y [BRITCHER90] han arrojado resultados positivos.

- RIESGOS DE LA REINGENIERIA: ANALISIS Y MIGRACION

La reingeniería no es algo simple de hacer. Esto es fácil si se tiene el tiempo y dinero para soportarlo. Para esto se debe de contar con un plan para aplicar la reingeniería. El plan puede evaluar y establecer los riesgos.

[ARNOLD91] catálogo riesgos típicos en las áreas de reingeniería, asociación de riesgos y migraciones.

La reingeniería tiene contingencias para riesgos de migración. En algunos casos la migración ha significado la prueba en escalas pequeñas de migración (por ejemplo, un subconjunto pequeño de programas) y riesgos evaluados antes detectados por la mayoría de los efectos de la reingeniería.

1.14 AVANCES FUTUROS:

Algunos avances en la tecnología de reingeniería pueden ser expectativos. Algunas de las áreas a las que se extiende es la herramienta CASE. La infraestructura de la herramienta CASE, opera como una herramienta-construcción-herramienta, puede ser consumible de tiempo y dinero. Sin embargo, esta tecnología de herramienta-construcción son disponibles para poder tener creaciones mas rápidas con la infraestructura de esta herramienta.

- MEDICION DE MANTENIMIENTO DE SOFTWARE:

Este es un desarrollo interesante de métricas para encontrar el mantenimiento de campo para el código. Este código debe ser un primer candidato para la reingeniería. Varias métricas de mantenimiento y armaduras existentes de trabajos de [ARNOLD82,83],[PEERCY81].

- MODELOS DE COSTO BENEFICIO DE REINGENIERIA:

La reingeniería media o larga a menudo requiere una justificación de costo-beneficio. Esto es una necesidad para organizar las entradas en los parámetros de mantenimiento y tomar una estimación razonable de los costos, beneficios y tiempo para su pago. Muchos trabajos necesitan hacer la estimación de las herramientas disponibles.

- SISTEMAS EXPERTOS PARA TAREAS DE REINGENIERIA:

Muchos experimentos existentes acerca de reingeniería,tienen un paso lógico que es capturado en un sistema experto con una base de reglas. La base de las reglas pueden ser incluidas en la Herramienta CASE para reforzar el valor de la herramienta para los usuarios. Ejemplos de áreas donde los sistemas expertos pueden ser utilizado por una prospección para reutilizar partes el código fuente [KNIGHT92], encontrando los objetos del código fuente, y su remodularización.

- MODELOS PARA Reverse Engineering:

Los modelos basados en la distribución principal de mantenimiento con diagramas noprocedurales juntan los modelos con las aplicaciones. El código fuente es generado, generalmente, con el uso de la Herramienta CASE (diagramas). El mantenimiento del código fuente es hecho por medio de diagramas, y no directamente del código fuente.

El paradigma del mantenimiento tiene grandes promesas para la reducción de costos de mantenimiento y facilidades en la evolución del software. La herramienta CASE existe para modelos basados en el mantenimiento. Las metodologías han sido desarrolladas y son refinadas, para la migración de la existencia de sistemas en modelos basados en sistemas.

- PROCESOS DE INSTRUMENTACION PARA MANTENIMIENTO DE SOFTWARE:

El proceso de software necesita ser instrumentado para capturar los cambios del software. Esto permitirá cambios en las historias para que el software pueda ser animado y mantenido por medio del play-back. Esto va mas allá de la administración de la configuración. El cambio de historias deben ser capturadas semánticamente para hacer el mantenimiento por medio del rastreo.

CAPITULO 2

HERRAMIENTAS PARA LA INGENIERIA REVERSIVA

CAPITULO 2. HERRAMIENTAS PARA LA INGENIERIA REVERSIVA

CASE COMO HERRAMIENTA DE INGENIERIA REVERSIVA

2.1 LA HERRAMIENTA CASE (COMPUTER AIDED SOFTWARE ENGINEERING).

ASPECTOS GENERALES

El CASE (Computer Aided Software Engineering), Ingeniería de Software Asistida por computadora, es una combinación de herramientas y tecnologías que conllevan a una eficaz y eficiente producción de software.

El mercado de productos CASE es variado, pero nosotros nos enfocamos a CASE ORACLE debido a que integra una amplia gama de herramientas que soportan el ciclo de vida del sistema.

La arquitectura integrada CASE de ORACLE ofrece una familia de productos que le ayuda a pasar de las necesidades del negocio a las soluciones.

CASE proporciona la importante conexión que faltaba, un marco estructurado para guiar y controlar el desarrollo productivo durante todo el ciclo de vida del sistema.

CASE es soportado por la familia de herramientas ORACLE:
CASE*Dictionary, CASE*Designer y CASE*Generator, estando apoyado por CASE*Method, la cual es una metodología comprobada a nivel mundial que guía a los proyectos a su culminación exitosa mediante la aplicación práctica de técnicas rigurosas de desarrollo de sistemas.

Las herramientas CASE contribuyen a automatizar el diseño y desarrollo de programas. Parece lógico que los programadores aprovechen la potencia de las computadoras para simplificar su trabajo; tal como hacen en favor de otras muchas personas. Una herramienta informática da apoyo a la ingeniería de software (esta sería, aproximadamente, la traducción del término CASE) y suele incluir las opciones siguientes:

- Gráficos. Una librería de símbolos para confeccionar diagramas de flujo de datos, diagramas de acción, diagramas entidad-relación, diagramas funcionales y otras técnicas semejantes.

- Metodologías. Soporte para, al menos, una metodología particular de diseño, como el diseño estructurado, análisis y diseño de sistemas, así como el controlar el desarrollo productivo durante todo el ciclo de vida del sistema.

- Diccionario de datos. Un medio para definir los elementos de datos y las estructuras de archivos, así como su utilización en las pantallas e informes que integrarán el sistema.

- Diseño de pantallas y reportes. Posibilidad de definir rápidamente la apariencia de pantallas y reportes, tanto para desarrollo de prototipos como generación automática de código.

- Documentación del sistema. Producción de listado del código fuente convenientemente documentados, de referencias cruzadas de variables y de diagramas de estructura del programa.

2.2 SOPORTE DE LAS HERRAMIENTAS CASE A USARIOS

El soporte que las herramientas CASE proveen a los usuarios para el mejor desarrollo de la metodología elegida son:

- Manipulación directa de objetos gráficos en múltiples ventanas al mismo tiempo, para la captura y actualización del diseño del producto.

- Facilidad del manejo de un lenguaje estructurado **(SQL).**

- Producción de reportes de catálogos, tablas y matrices generadas de la información almacenada en el diccionario de datos.

- Generación de otras partes del diseño del producto, basado en las partes ya definidas.

- Interfaces con otras herramientas, por ejemplo, para la administración del proyecto, realiza estimaciones, presentaciones gráficas o manejadores de prototipos.

2.3 ARQUITECTURA DE UN SISTEMA CASE

La arquitectura de un sistema CASE es la que se muestra a continuación:

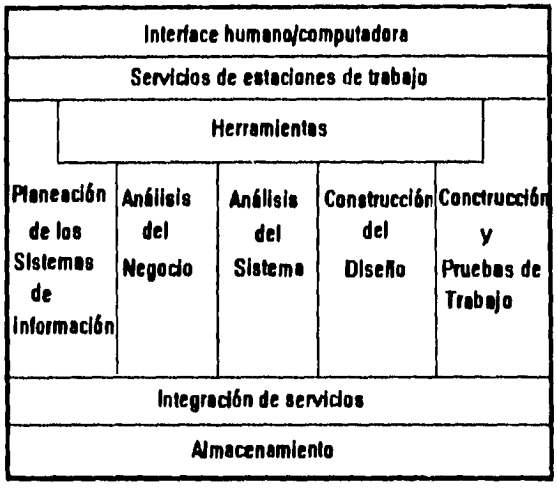


FIGURA 3. Arquitectura CASE típica de un sistema

En donde el más simple sistema CASE no requiere servicios de integración. El más complejo da soporte a todas las partes.

* El human-computer interface (Interface humano-computadora) es en donde el hombre forma una parte importante dentro del diseño del sistema, y al cual se le permite usar la computadora y accesar a diversas herramientas.

* Workstation services (servicios de estaciones de trabajo), proporciona el ambiente operacional para controlar el uso de herramientas e interfaces.

* Herramientas, son comunmente diseñadas para proveer de soporte a técnicas (o una técnica) usada en pasos de una etapa en particular. Estas pueden ser clasificadas en general en aquellas que soportan etapas predominantemente analíticas (planeación y análisis del negocio), y aquellas que soportan etapas prescriptivas (diseño del sistema, construcción y pruebas de bancos de datos). Ocasionalmente estos grupos son conocidos con los términos "Upper-Case" y "Lower-Case".

* Integración de servicios, provee una interface común entre herramientas y almacenamiento. Permite a las herramientas compartir datos aunque éstos provengan de diferentes vendedores.

* Diccionario, contiene la información del proceso de desarrollo, dentro de los datos de los componentes descubiertos, durante el proceso, así como la representación que debe generar.

Existe una gran variedad de términos para describir al almacenamiento, entre estos están: Depósito, Enciclopedia, Base de Datos, Base de Conocimientos.

Si la herramienta usada en una implementación particular para esta arquitectura proviene de un solo vendedor, el producto puede definirse como "I-CASE" (CASE Integrado).

Si la herramienta proviene de muchos vendedores el producto se define como "C-CASE" (CASE Componente), en donde el componente se refiere al hecho de que cada herramienta soporta sólo una parte del ciclo de vida del software.

2.4 ETAPAS DE CASE

CASE es una metodología flexible,debido a que puede cambiar etapas y pasos dentro de la misma.

En la práctica las herramientas y técnicas tienen menos pasos, esto significa que, algunas herramientas pueden ser más usadas en algunas etapas que otras.

El soporte dado por CASE para almacenar es más implícito que explícito. El proceso de diseño requiere alguna forma de funcionalidad I-CASE y necesita incorporar alguna etapa para representar el diseño.

ETAPA ANALITICA

En esta etapa se lleva a cabo la planeación del sistema y análisis de negocios, CASE proporciona las herramientas que dan soporte a la información, detalles y documentación de los resultados de la investigación, así como la modelación de componentes específicos.

En CASE las etapas se aplican de forma general para todos los modelos del sistema. El propósito de la etapa analítica es dar a conocer los modelos menos desarrollados que tienden a ser trabajados en muchas metodologías.

ETAPA PRESCRIPTIVA

El diseño del sistema,diseño-construcción y la etapa de prueba de trabajo (workbench) incrementa la automatización del sistema. La traslación de modelos de datos en las especificaciones de bases de datos pueden ser hechas automáticamente, por medio de la generación de procedimientos lógicos estereotipados.

En el diseño de construcción y la prueba de trabajo (workbench), son más pasos del sistema que pueden ser automatizados por la generación del código del programa producido automáticamente, así como la descripción de bases de datos y lenguajes de control necesarios para formalizar el sistema y ligar módulos de edición y de esta forma completar la prueba de trabajo (workbench).

El nivel de automatización puede cubrir todos los pasos del diseño, CASE genera aplicaciones basadas en los resultados del análisis. Esta aplicación deberá ser vista como prototipo, pero el paso de diseño es usado para probar los requerimientos del sistema.

Muchas de estas aplicaciones pueden tener especificaciones que pueden ser reusables, debido a que el diseñador puede requerir nuevas funciones, y de esta forma CASE tiene que incorporarlos automáticamente al producto diseñado.

ETAPA DE EVOLUCION

En esta etapa CASE tiene la capacidad de administrar la evolución del sistema. El análisis y diseño del sistema puede ser modificado fácilmente.

La Ingeniería Reversiva forma parte de los elementos CASE.

Dichas herramientas pueden ser usadas para examinar la existencia de programas, bases de datos o archivos, para que de esta forma se pueda conocer cuales son las ligas que existen entre éstas.

La información de esta herramienta se almacena en el diccionario. Pueden utilizarla como soporte para el mantenimiento y evolución progresiva del sistema o como re-ingeniería para crear o usar una nueva versión de éste.

La Ingeniería Reversiva de CASE en programas es más difícil que en otras estructuras de datos. Las herramientas reestructurables deben ser usadas primeramente, para entender el programa en un formato estructurado. Sin embargo, la interpretación de dichas herramientas demanda alguna manera de determinar qué partes del programa requieren de un diseño lógico y cuales no, y de esta forma deducir cual es el propósito del diseño lógico del sistema.

Con las estructuras de datos, las reglas de normalización y las de mapeo proporcionan a la base de datos interpretaciones razonables de los datos. El área de mayor dificultad es cuando se analiza el código del programa. Este código debe ser verificado en forma manual o a través de elementos de programas lógicos que permitan realizar cambios al código.

ETAPA DE ADMINISTRACION

La administración y el control de configuración proporcionan al coordinador mecanismos para asistir la administración de los proyectos y evaluación de los mismos.

La coordinación se debe hacer en el empiezo del proyecto y en etapas diferentes del mismo.

Esto se puede aplicar cuando se requiere hacer alguna variación al sistema. Para estos requerimientos se utiliza el I-CASE.

En general CASE da soporte para administrar el uso de la metodología en el desarrollo de un sistema.

2.5 HERRAMIENTAS Y METODOLOGIAS CASE

El CASE de ORACLE provee una gama completa de soluciones, métodos y herramientas.

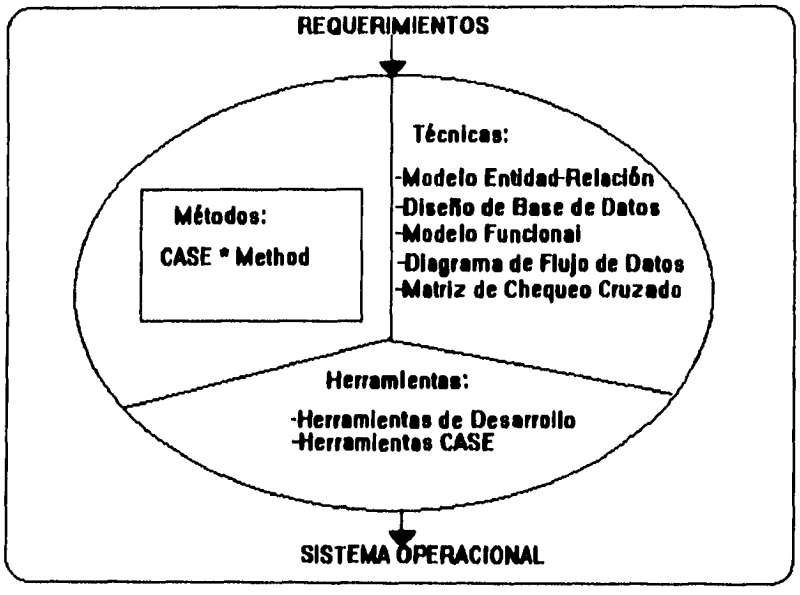


FIGURA 4. Herramientas de CASE

2.5.1 CASE *METHOD (UN ENFOQUE ESTRUCTURADO "TOP-DOWN")

CASE *METHOD comienza en la parte superior del ciclo de vida de los sistemas, con la participación tanto de los administradores principales como de los usuarios en la recopilación de los objetivos del sistema, requerimientos y prioridades de negocios que se necesitan para desarrollar un sistema, ahora y en el futuro. CASE *METHOD fomenta la comunicación entre los usuarios y los diseñadores, ya que los sistemas se definen desde un punto de vista estratégico.

A medida que prepara modelos cada vez más detallados, se podrán verificar y confirmar con los usuarios y así crear prototipos conceptuales del sistema completo.

Con el enfoque top-down, conjuntamente con la técnica de revisión cruzada botton-up, CASE *METHOD mejora la calidad a nivel de todas las etapas.

Así también, apoya todas las fases del ciclo de vida de los sistemas. Después de terminar el análisis, se crean diseños lógicos y físicos con prototipos realizados conjuntamente con los usuarios para su utilización externa, logrando óptima flexibilidad y rendimiento. La documentación y la implementación de sistemas ocurren en paralelo, a medida que se van construyendo y ejecutando las pruebas de cada unidad. La transición incluye la instrucción y capacitación del usuario, la conversión de datos y las pruebas de los sistemas, siguiendo la planificación previa efectuada durante el análisis y el diseño. Las revisiones finales durante la producción aseguran que los sistemas rindan con éxito y alta calidad, de acuerdo con la planificación.

CASE *METHOD asegura el logro de los objetivos y prioridades administrativas y aumenta gradualmente la calidad y productividad durante cada fase del ciclo de vida, además ayuda a reducir considerablemente los gastos y el tiempo de desarrollo, promover la participación y dedicación del usuario y producir un mejor entendimiento de la empresa y una terminología compartida para los negocios.

CASE *METHOD provee de una solución científica para el desarrollo de un sistema, soportando una estructura apropiada para la ingeniería de sistemas en ambientes de procesamiento de datos.

Típicamente los sistemas de información de negocios está marcados dentro del diccionario en una serie de etapas.

A través de estas etapas, los requerimientos de los sistemas de negocios son progresivamente identificados, analizados y usados para diseñar el sistema.

Existen siete etapas básicas, los cuales son:

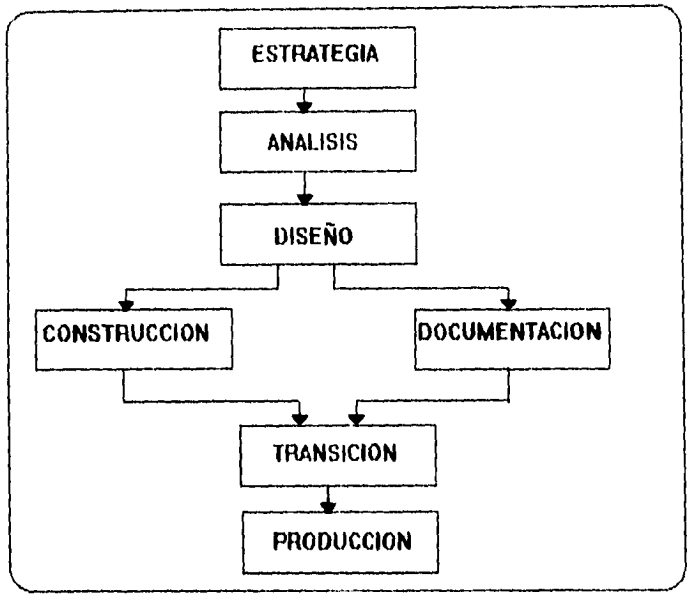


FIGURA 5. Etapas CASE

El ciclo de vida de negocios puede proveer de un reflejo de los estados de desarrollo requeridos para implementar prácticamente cualquier solución de negocios.

Las etapas y sus objetivos principales son los siguientes:

1.- ESTRATEGIA. Aquí la funcionalidad y necesidades primarias de información de negocios son establecidos y durante el estado de estrategia entidades y relaciones de entidades son definidas dentro de CASE *DICTIONARY.

Estas definiciones especifican las necesidades de la empresa para dar cabida a la información.

El modelo jerárquico de funciones es definido y dibujado en el diccionario, identificando las funciones que los negocios desarrollan o necesitan desarrollar.

2.- ANALISIS. En este mismo apartado los modelos de funciones son redefinidos y el uso de los datos de las entidades por las funciones son especificados.

Las definiciones de los diagramas de flujo pueden mostrar los flujos de información entre funciones.

3.- DISEÑO. La etapa de diseño involucra la definición de las tablas y columnas que serán utilizadas en la base de datos.

Las relaciones entre éstas, los constraints (restricciones) y validaciones son detalladas.

También contiene la definición de los módulos para poder implementar las funciones del negocio.

CASE *DICTIONARY provee de un número de utilerías y reportes para facilitar este proceso. Este incluye la utilería del diseño por default (DEFAULT DATABASE DESIGN) de la base de datos y la utilería de la especificación de módulos por default.

a) Default Database Design Utility.

La utilería genera la tabla y definición de columnas basada en la definición de entidades.

b) Default Module Specification Utility.

La utilería genera los módulos candidatos a usar en la aplicación. El módulo de datos usado y la definición de parámetros se basan en el diagrama jerárquico de datos y diagrama de flujo.

4. IMPLEMENTACION. Durante la etapa de implementación las utilerías del diccionario de datos son utilizadas para generar las instrucciones en SQL requeridas para crear la base de datos con las definiciones obtenidas durante la etapa de diseño.

En esta etapa se encuentra la CONSTRUCCION, DOCUMENTACION Y TRANSICION.

CASE *METHOD es soportado por una familia de herramientas ORACLE: CASE*Dictionary, CASE*Designer y CASE*Generator.

5.- PRODUCCION. En esta etapa se corre el sistema.

2.5.2 CASE *DICTIONARY

CASE *DICTIONARY se enfoca al desarrollo de bases de datos para los productos ORACLE CASE y provee de más funciones a la familia CASE ORACLE. Esta es la más grande contribución a la venta del uso de los productos ORACLE CASE para la construcción de sistemas. Así mismo, también provee de mecanismos de seguridad a los usuarios de ORACLE que tienen acceso al diccionario. Todos los tipos de información que pueden ser usados dentro del desarrollo de un sistema son guardados en el diccionario. Los derechos de acceso y de solo lectura pueden ser garantizados por el uso de CASE DICTIONARY.

El concepto de diccionario es usado dentro del proceso de complexión, introducción y post-implementación , y esto es usado por todas las etapas en el CASE *METHOD.

La información del diccionario está almacenada en una base de datos, la cual es construida y mantenida usando el sistema de administración de bases de datos relacionales de **ORACLE (RDBMS)**. La cual es una librería para especificar el desarrollo de formas, reportes y utilerías para entrar, manipular y procesar la aplicación de datos. Los productos son controlados por el menú del sistema y guías que existen dentro del sistema.

CASE DICTIONARY ofrece control de los elementos en la aplicación de sistemas. Una aplicación de sistemas es una colección de las funciones del negocio, entidades, programas y tablas, las cuales pueden ser descritas por la documentación de varias formas del sistema. Esto es generalmente implementado en alguna área del negocio y/o un simple sistema de cómputo, y se hace con el fin de tener un mejor control en las actividades del negocio.

Una aplicación del sistema no es restringida para una sola etapa en el ciclo de vida del sistema, sino que se deberá ver como una intersección de todas las etapas desarrolladas dentro de un sistema. Una aplicación puede ser un número de elementos (entidades, tablas, etc.) y elementos que pueden ser "enlazados" en diferentes sistemas de aplicación. Sin embargo, un elemento puede ser sólo utilizado por una aplicación del sistema y por los usuarios del CASE DICTIONARY y con el acceso a esa aplicación del sistema pueden realizar cambios en éste.

Una aplicación del sistema deberá ser hecha por el CASE DICTIONARY USER, esto puede tener variaciones para accesar en una aplicación del sistema a otros usuarios del CASE DICTIONARY. Para esto el DATA MANAGER puede aplicarse en el sistema para minimizar los niveles de control. Estos tienen derecho a accesar a la aplicación del sistema para que la gente trabaje de forma apropiada durante su desarrollo.

La funcionalidad del CASE DICTIONARY ha sido creada para soportar algún requerimiento de las entidades en la aplicación de sistemas. Esto es, una entidad puede existir en más de una aplicación del sistema, teniendo el mismo nombre pero, posiblemente, con una descripción diferente (relaciones, atributos) y de cierta forma, esto es benéfico para las organizaciones donde,la aplicación de sistemas necesita ser desarrollada.

Los sistemas de cómputo desarrollados por el CASE DICTIONARY, proporcionan beneficios que pueden ser desplegados como entidades en toda la aplicación del sistema que se está generando. Para esto se dispone del diagrama de relaciones entre las entidades.

Dentro del CASE DICTIONARY existen dos tipos de usuarios que determinan la disponibilidad de menús, los cuales son:

1) MANAGER.- Tiene acceso al CASE *Dictionary ADMINISTRATION MENU y al CASE DICTIONARY MAIN MENU.

2) USUARIO.- Tiene acceso solo al CASE *Dictiorary MAIN MENU.

Algunas de las utilerías más valuables son aquellas que provee por Default Database Design, Database Sizing, Index Design, Generation of the data Definition Language Statements en SQL para ORACLE Y DB2 y la definición de los módulos del programa usados por el CASE *Generator.

2.5.3 CASE *DESIGNER

El CASE *Designer es el nombre que se le da a un conjunto de herramientas desarrolladas para soportar conceptos de CASE.

Provee de un ambiente multiventanas, multiusuario y una interface gráfica para el desarrollo de la base de datos (CASE *Dictionary).

Siguiendo un método estructurado (CASE *Method), el CASE*Designer usa diagramas para modelar al negocio, y cómo se usa la información del negocio para soportar estas actividades.

HERRAMIENTAS DE CASE *DESIGNER

Las herramientas del CASE *Designer son las siguientes:

- Diagramador entidad-relación.
- Diagramador de funciones jerárquicas.
- Diagramador de flujo de datos.
- Diagramador de matriz.
- Otras herramientas de soporte.

- DIAGRAMADOR ENTIDAD-RELACION. Es usado durante las etapas de estrategia y análisis para definir la información del negocio como un modelo de entidad. El diagrama representa las entidades, así como las relaciones vitales entre estas y los atributos que las describen.

Estos diagramas pueden servir como material de ayuda para sesiones de retroalimentación o simplemente para almacenar el modelo en el diccionario de datos.

- DIAGRAMADOR DE FUNCIONES JERARQUICAS. Puede ser usado para transformar notas tomadas durante entrevistas en funciones estructuradas. Estas funciones describen lo que la organización hace o necesita hacer, independientemente de cómo lo hace. El diagramador crea y organiza las funciones en un orden jerárquico.

- DIAGRAMADOR DE FLUJO DE DATOS. Se usa en la etapa de análisis para crear diagramas que muestran el flujo de información de la organización, lo que afecta a la organización y dónde se almacenan los datos.

- DIAGRAMADOR DE MATRIZ. Permite desarrollar una matriz que muestra la asociación entre dos tipos de información, por ejemplo, funciones y entidades, o funciones y unidades del negocio. Esto puede ser útil para realizar un cross-reference y tener un chequeo completo.

Usando CASE *Designer, el sistema es literalmente "construido" en la pantalla en una forma que puede ser usado en un ambiente no computacional. Los beneficios ganados son rapidez y precisión, ya que la información es sustraída desde un diccionario universal de información reusable con definiciones entendibles, claras y comunes, así como visibilidad, la cual también contribuye a la velocidad y precisión.

2.5.4 CASE *GENERATOR

Pertenece a la familia de productos de software CASE *Dictionary, sirve para generar aplicación de programas en varios lenguajes como son: SQL *Forms, SQL *ReportWriter and SQL *Plus.

Esta herramienta toma las definiciones de la información guardada en el diccionario de datos y automáticamente genera un programa funcional escrito en lenguaje de cuarta generación (SQL).

Esta opción contiene la posibilidad de crear automáticamente formas, reportes y menús en CASE, tomando por default los valores del diccionario, así también, permite modificar las definiciiones y el uso de las opciones del diseñador de SQL *Forms.

2.6 OTRAS HERRAMIENTAS PARA LA INGENIERIA REVERSIVA

Existen varias herramientas en el mercado para Reverse Engineering.

MARKOSIAN, describe una tecnología que permite aplicar el software Reverse Engineering , esta tecnología soporta el rápido desarrollo de herramientas para el análisis y la modificación sistemática de los sistemas existentes. Con el uso de esta tecnología en 4 1/2 meses se pudo desarrollar y probar una herramienta para un desarrollo complejo, el análisis del flujo global de datos y la transformación de programas COBOL de 40,000 líneas de código en una simple unidad de compilación. Previamente había tomado más de 20 semanas-persona el hacer reingeniería a programas de 15,000 líneas de código usando otras herramientas, con la aplicación de esta nueva herramienta el tiempo total fué reducido a 4 horas-persona.

Esta tecnología permite :

- Automatizar procesos específicos usados por la empresa.
- Producir módulos de código fuente en el mismo estilo y formato que los producidos manualmente.
- Incorporación con el ambiente de mainframe IBM y
- Soporte automático de actividades de mantenimiento adicional más allá del próposito del proyecto.

Las ideas técnicas centrales dentro de la tecnología son :

Representar software en la forma de árboles de sintaxis abstracta anotada (ASTs annoted sintax abstract trees) en una base orientada a objetos.

Uso de un alto-nivel para especificaciones de lenguaje ejecutable para analizar y transformar representaciones de código en esta forma.

El software que se utilizó es el Refinery y el REFINE/Cobol que son productos de Reasoning Systems y que incorporan esta tecnología.

Software Refinery es una herramienta de ambiente de desarrollo para la reingeniería. Sus componentes son:

DIALECT , compilador gramático, se ha usado para análisis gramático de lenguajes como Ada, C, Cobol, Fortran.

REFINE, realiza el análisis de software y la transformación del sistema, contiene:

- Una base de datos orientada a objetos que se usa para modelar el software e información.
- Especificaciones de lenguaje ejecutables , este lenguaje también se llama REFINE.
- Un compilador para el lenguaje REFINE
- Soporte para debuggear

WORKBENCH es una librería de componentes reusables en la construcción de herramientas de reingeniería , se usa en el REFINE/Cobol para representar información requerida en la modularización

INTERVISTA es una interface gráfica de ayuda. Tiene una interface hipertexto para programar código fuente .

REFINE/Cobol es un banco de trabajo para Cobol. Carga los programas Cobol en su base objeto y genera reportes incluyendo:

* gráficas de llamadas
* gráficas de control de flujo
* modelo de datos

Tiene una interface gráfica de usuario que despliega e imprime esos reportes como diagramas, tablas y texto. [3]

ANTONINI describe un sistema que captura diseño de información de bajo nivel. Desde programas COBOL ellos pueden producir diagramas de estructuras Jackson o Warnier/Orr.

El proceso consta de 2 etapas : en la primera etapa se usa un abstractor de información para realizar un análisis estático del código COBOL que produce gráficas de control de flujo, anidamiento de árboles y listas de referencias cruzadas. Estas salidas desde la primera etapa son transformadas en estructuras gráficas o documentos Warnier/Orr.[4]

ARANGO Describe la transformación - modelo basado o TMM, el cual puede ser usado para modelar el proceso de mantenimiento. El modelo está basado en el paradigma de Draco que es un modelo de dominio enfocado a la construcción de software.

Este modelo asume que un número de programas de software similar están siendo construidos y que la ruta para una especificación de una pieza de código puede ser representada como una gráfica cíclica. La raíz simple de la gráfica es la especificación y lo que deja son programas ejecutables. Los nodos que están entre la raíz y lo dejado representan refinamientos de la especificación original en varios niveles de abstracción.

Los diferentes refinamientos pueden ser aplicados para cualquiera de las especificaciones, pero normalmente solo una ruta desde la raíz a una hoja es explorada y el movimiento es hacia arriba.

El movimiento hacia arriba es continuado hasta un nodo que contenga características no deseadas y nuevas características deseadas son encontradas. Este nodo es conocido como la última abstracción común. El movimiento entonces procede hacia abajo usando una ruta diferente hacia una hoja que representa la implementación modificada.

La dificultad de esta propuesta es el movimiento inicial hacia arriba para recapturar el diseño. El artículo no sugiere alguna propuesta partícular pero sugiere que la entrada puede venir desde varias fuentes también como el código.

Estas otras fuentes son: la experiencia previa de mantenimientos, dominio en el conocimiento del programa, entrada desde el diseñador original y documentación existente.

El autor ha puesto sus ideas en práctica por vía del sistema Draco por sí solo desde la implementación de Lisp de otro. El modelo de mantenimiento es útil pero el problema central de recapturar el diseño ha sido evitado.[4]

SOLOWAY ha empleado el concepto de 'planos' para modelar tanto el desarrollo como el mantenimiento, los 'planos' se definen como :

"Fragmentos de programa que representan secuencias de acciones estereotópicas en programación, ej. una corrida de un plano de ciclo total , un plano de ciclo de búsqueda de elementos ."

En el modelo Soloway,los programas están compuestos de planos que han sido modificados para resolver un problema particular.

Los planos son implementados usando reglas de disertación que representan convenciones de programación tales como el uso de nombre de variables que representen esa función. Soloway y Ehrlich han mostrado que las reglas de disertación son muy importantes en la comprensión del programa por la conducción de un estudio entre programadores novatos y con experiencia. En programas donde las reglas de disertación fueron seguidas, los programadores con mayor experiencia fueron muy superiores a los novatos. Sin embargo en programas donde las reglas de disertación no fueron seguidas, los resultados de los programadores experimentados no difirieron mucho de los novatos.

Un estudio hecho por Letovsky y Soloway ha mostrado que los programadores con experiencia tienen dificultad para entender programas en los cuales los planos no están localizados y se encuentran físicamente dispersos dentro del código fuente. Soloway y Johnson han descrito un sistema llamado PROUST que reconoce planos en programas de novatos. Se asume que el programa novato tiene 'bugs' (vicios) y el PROUST intenta analizar los programas recreando los métodos con los que se generaron.[4]

PUDSY es un sistema similar que se basa de igual modo en la comprensión del programa y la detección de 'bugs' (vicios). En la fase de comprensión , PUDSY divide grupos de sentencias en pedazos y evalúa las declaraciones de entrada y salida de cada pedazo. PUDSY tiene una base de conocimientos de los pedazos , pero si no se puede combinar un pedazo con otro de la base de conocimientos , emprende una ejecución simbólica del PUDSY entonces combina las declaraciones de entrada y salida del pedazo para la sección de código bajo análisis. Estas declaraciones pueden ser comparadas con la especificación original, que asume que también está expresada en términos de una declaración. [4]

Esta propuesta y la de Soloway se ven prometedoras, pero ambas han sido sistemas experimentales que solo se han aplicado en programas pequeños, habrá que probar su efectividad en programas grandes.

Ward ha descrito una proposición diferente para Reverse Engineering que se basa en transformaciones probables. El describe una representación interna para un programa y entonces se aplica a series de transformaciones para simplificar la estructura de la forma interna. Estas transformaciones pueden ser aplicadas antes de la comprensión tal como ellos han sido previstos para estar correctos.

El autor también describe un método de reducir la forma interna en una especificación. [4]

Esta propuesta para Reverse Engineering depende de aplicar las transformaciones apropiadas en el orden correcto y esta tarea esta emprendida actualmente por el humano.

Desarrollando un algoritmo o heurísticas para aplicar las transformaciones apropiadas en el orden correcto es un interesante problema de investigación.

En lo que se refiere a la herramienta CASE podemos decir que:

Los modelos de datos usados por el IE: Advantage la herramienta CASE debe ser importada del IRC (INFORMATION RESOURCE CATALOG). El IRC permite a los usuarios ligar la evidencia física del modelo de datos con una interfase que los usuarios pueden usar en el contenido del IRC, el status de los modelos de datos, y ligas entre elementos fiscos, entidades lógicas, y requerimientos del negocio y reglas.

La herramienta CASE puede solucionar la estructura física de los datos y variables de seguimiento de código. Esto es utilizado para identificar como un esquema físico es creado, actualizado, leído y borrado, pero no es suficiente para construir el esquema lógico y vistas externas conceptuales de los requerimientos de los datos.

La herramienta CASE es la solución principal para Reverse Engineering, porque asocia la estructura de datos física y segmentos de variables de código. Esto permite identificar cómo el esquema físico es creado, actualizado, leído y borrado, pero no es suficiente para la construcción de las vistas lógicas externas y conceptuales de los requerimientos de los datos. Sin embargo, expertos en la materia, aseguran que la técnica Reverse Engineering ayuda al análisis de productos que no son incluidos en las aplicaciones, liga evidencias físicas proveídas de la matriz diseñada y da soporte al modelo de administración, describiendo los servicios que ofrece.

Lo más importante, es el uso de herramientas para desarrollos inadecuados, sistemas incompletos o requerimientos de ingeniería de software. Recuperando un modelo de datos lógico normalizado junto con la asociación de las reglas del negocio, políticas y estructuras de datos físicos es difícil, el análisis desarrollado por humanos siempre es sistemático (incluye modelo de administración y modelos de datos adecuados al sistema).

Las herramientas CASE deben argumentar el análisis propuesto inicialmente para entender la implementación física del sistema., pero las herramientas CASE no han proveído la solución para recobrar los requerimientos de datos conceptuales y lógicos. Esta herramienta es diversificada y compleja. En orden para descubrir y recobrar la implementación de los requerimientos de datos en el sistema.

PRODUCTOS DE LA INGENIERIA REVERSIVA Y SUS USUARIOS

En la fase inicial de cualquier proyecto, continuamente nos preguntamos:

Cuáles son los productos producidos para Reverse Engineering?

Cómo pueden ser usados estos productos?

Cómo es utilizado Reverse Engineering en el desarrollo de las actividades de otro sistema?

Cuáles serán los productos que existirán de Reverse Engineering?

Porqué cuando se necesita orden en el Reverse Engineering se obtienen elementos de datos standard?

Los modelos de datos lógicos y los datos standard de los elementos no son solo productos Reverse Engineering. La salida de Reverse Engineering incluye:

-MODELO DE VISTAS DE ALTO NIVEL EN LA DESCOMPOSICION DE JERARQUIAS.

Esto debe ser usado en el tamaño del sistema, alcance del proyecto y definición de las vistas del modelo de datos lógico.

-DISEÑO DE LA MATRIZ

La matriz mantiene en el IRC los requerimientos substanciales para especificar la composición del modelo de datos para ligarlos especificar los estatutos de autorización, regulación y guía legalizada del sistema. La matriz puede ser usada para el análisis cuando lleguen a ocurrir cambios subsecuentes.

-RECOMENDACIONES EN LA INSERCION DE TECNOLOGIA

Se tomarán en cuenta las mejores recomendaciones para utilizar tecnología nueva en bases de datos y comunicaciones en red.

-FRAMEWORK, METODOLOGIAS Y REPORTES USADOS POR LAS HERRAMIENTAS

El lenguaje, administración del sistema de datos, complejidad de la implementación y plan estratégico para cada sistema es diferente. El esquema general es seguido por todos los sistemas en Reverse Engineering, pero identificamos los detalles para derivar el código lógico y modelos de datos del análisis estructurado. Durante cada fase refinamos y revalidamos el esquema de Reverse Engineering, metodología y uso de herramientas en cada clase de sistema.

-SISTEMA Y PLAN DE MIGRACION DE DATOS.

El plan de migración de datos preescribe los pasos necesarios para hacer la existencia legal del sistema con el concepto IE basado en la guía de existencia directiva.

-REQUERIMIENTOS DE SOFTWARE REUSABLES.

Otros productos serán reusables por los requerimientos del software: como son el modelo de datos, conjunto de reglas del dominio especifico, y tal vez, la construcción de software para soportar los requerimientos del sistema.

-DOD ENTERPRISE MODELO DE DATOS, DOD STANDARD ELEMENTO DE DATOS, INTEGRACION DE LA BASE DE DATOS ENTERPRISE.

Estos son todos los objetivos del CIM/DAPMO estandarización de datos. Todas las salidas del proyecto Reverse Engineering serán integradas a este departamento.

CAPITULO 3

APLICACION DE LA INGENIERIA REVERSIVA A LAS BASES DE DATOS

El proceso comienza con la extracción detallada de la información del diseño, y la extracción de un alto nivel de abstracción de diseño. La información detallada (bajo-nivel) del diseño es extraída desde el código fuente y la documentación existente de diseño. Esta información incluye, estructuras gráficas, descripciones de datos. Un acercamiento similar, pero automático es descrito en otro sitio para recuperar documentos Jackson y Warnier/Orr desde el código. Las representaciones de alto nivel del diseño son extraídas desde el diseño detallado y expresado usando flujo de datos y diagramas de control de flujo. A través de este documento el término 'diseño recuperado' será utilizado para denotar al diseño extraído. Los pasos del procedimiento son discutidos a continuación, la figura 6 resume el proceso:

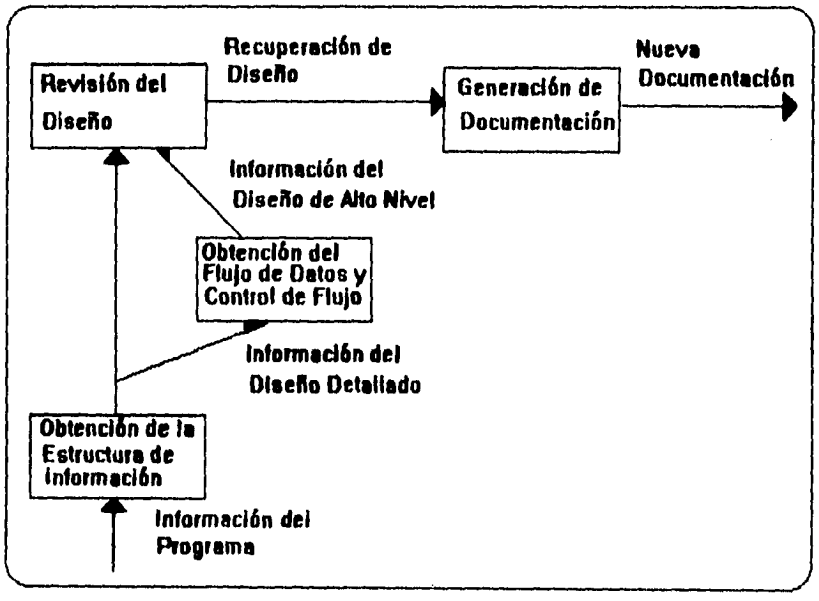


FIGURA 6. Proceso de Ingeniería Reversiva

3.1 PASOS DE LA INGENIERIA REVERSIVA

1. Recopilación de Información. Consiste en recopilar toda la información posible acerca del programa. Las fuentes de información incluyen : código fuente, documentación del diseño y documentación de llamadas al sistema y rutinas externas. La experiencia personal de los programadores con el software también debe ser identificada.

2. Examinar Información. En esta etapa se revisa la información recopilada. Este paso permite hacer que una persona se familiarice con el sistema y sus componentes. Un plan para analizar minuciosamente el programa y registrar la información recuperada puede ser formulado durante esta etapa.

3. Extraer la Estructura. Aquí se identifica la estructura del programa y ésta se usa para crear un conjunto de estructuras gráficas. Cada nodo en la estructura gráfica corresponde a una rutina llamada en le programa. La gráfica registra la jerarquía del programa. Así la gráfica almacena la llamada jerárquica del programa. Para cada nodo en la gráfica los datos son pasados a un nodo y regresados por el nodo que debe ser registrado.

4. Registro de Funcionalidad. Para cada nodo en la estructura gráfica, se registra el proceso hecho en la rutina del programa correspondiente a ese nodo. Un **PDL** puede ser usado para expresar la funcionalidad de las rutinas del programa. Para el sistema y rutinas de librerías, la funcionalidad puede ser descrita en Inglés o en una notación más fácil.

5. Registro del Flujo de Datos. La estructura del programa recuperado y **PDL** pueden ser analizados para identificar las transformaciones de datos en software. Estos pasos de transformación muestran el proceso de datos hecho en el programa. Esta información es usada para desarrollar un conjunto de diagramas de flujo jerárquicos que modele el software.

6. Registro del Control de Flujo. En esta etapa se identifica el control de estructura de alto nivel del programa y lo registra usando diagramas de control de flujo. Esto se refiere a un alto nivel de control que afecta la operación total del software, no controla el proceso a bajo nivel.

7. Revisión del Diseño Recuperado. Se revisa la consistencia y exactitud del diseño recuperado con la información disponible. Identifica cualquier elemento omitido de información e intenta localizarlos . Se revisa también el diseño para verificar que el programa está correctamente representado.

8. Generación de Documentación. El paso final es generar la documentación del diseño. La información explica el propósito del programa, vista del programa, historia, etc. que necesita ser registrada, esta información muy probablemente no esta contenida en el código fuente y debe ser recuperada desde otras fuentes.[1]

3.2 OBJETIVOS DE PROYECTOS CON INGENIERIA REVERSIVA

Los objetivos de proyectos con Ingeniería Reversiva son los siguientes:

1) Desarrollo de un cuadro sistemático para una base de datos para el Reverse Engineering y una estructura de datos para recobrar las reglas del negocio, dominio de información, requerimientos funcionales, y el uso de los datos producidos para desarrollar la normalización de datos de los modelos lógicos, construcción de la arquitectura de datos, y documentación de los datos requeridos.

2) Validar y refinar el cuadro de contexto para Reverse Engineering.

3) Desarrollar un significado de alcance y estimación futura del costo del proyecto en Reverse Engineering.

4) Desarrollar y capturar las métricas generales para asistir la decisión hecha en determinadas posibilidades económicas del futuro de reingeniería, reverse y Forward Engineering.

Reverse Engineering extrae el modelo lógico de datos de una migración múltiple de sistemas de bases de datos y archivos para reducir la redundancia, mantenimiento e inconsistencias que ocurren en los sistemas.

Esto es un soporte para la estandarización y la migración de datos.

3.3 EL IMPACTO DE LA INGENIERIA REVERSIVA EN EL DESARROLLO DE PROYECTOS

Por medio de la Ingeniería Reversiva en el desarrollo de proyectos se puede detectar y/o corregir lo siguiente:

1.- El inventario de evidencias físicas es difícil para coleccionar y analizar. Si la documentación existe, esto es frecuentemente de una calidad pobre. Además, el personal con el que se cuenta no tiene los requerimientos de conocimientos necesarios.

2.- La existencia de interfases de elementos de datos definidos para transferir las instancias de datos no son representados completamente, solo proporciona un fragmento de la estructura de datos y los requerimientos de semántica. Para obtener una esquema completo de los requerimientos de los datos, el análisis del Reverse Engineering debe determinar no solo como es la interfase de los elementos de los datos, son generados y usados siempre por el sistema, sino también identifica otras no-interfases de los elementos de los datos que son sinónimos de los sistemas disponibles.

3.- Aunque una especificación común del sistema debe haber seleccionado un grupo de sistemas legales del dominio funcional, la implementación no será exitosa si los únicos requerimientos críticos del sistema actual son identificados, documentados y direccionados por el sistema reemplazado.

La coordinación, negociación, planificación, y procesos de compra para Reverse Engineering son excepcionalmente complicados y consumibles de tiempo.

Dentro de la Ingeniería Reversiva (planificación, construcción, mantenimiento, existencia y futuros usuarios) se tiene que tomar en cuenta la cantidad de gente que manejara el sistema, con el fin de que exista coordinación en su construcción.

3.4 GUIA PARA APLICAR INGENIERIA REVERSIVA

Una guía para aplicar el diseño y desarrollo de la metodología Reverse Engineering es la siguiente:

1) MODELO DE DATOS (DIMENSION CONCEPTUAL O LOGICO)

2) TRANSFORMACION DE MODELOS (MODELOS INTERMEDIOS ENTRE DIMENSIONES FISICAS Y LOGICAS)

3) DISEÑO FISICO Y ESQUEMAS (DIMENSIONES FISICAS)

4) BASES DE DATOS Y DESARROLLO DE APLICACIONES

5) DATA/SYSTEM MIGRACION DE PLANES

6) DATA/SYSTEM PRUEBA DE PLANES

7) DATA/SYSTEM IMPLEMENTACION DE PLANES.

El impacto de la complejidad administrativa en proyectos Reverse Engineering incluye:

1.- Los objetivos estratégicos del sistema con respecto a cada sistema arriesgo/propietario debe ser identificado y prioratizado. Los objetivos del Reverse Engineering y las prioridades deben ser sincronizadas de acuerdo a las necesidades del propietario.

2.- Los proyectos de Reverse Engineering comprados y coordinados exteriormente no pueden tener éxito. La administración de alto nivel es necesaria pero no suficiente. La administración de medio nivel y sistemas personales deben también entender y soportar los objetivos del proyecto de Reverse Engineering.

3.- La negociación, planificación, y procesos de compra deben ser hechos antes del proyecto definitivo.

Bajo el cuadro de datos de Reverse Engineering, desarrolla los modelos que incluyen identificación, refinamiento, validación, y ligamento de todas las funciones del negocio, políticas, reglas y actividades de los elementos de datos, contenido del modelo, y evidencia física. Esto aproxima todas las estructuras de datos que pueden ser identificadas y ligadas para soportar procesos y minimizar el impacto de los datos cuando se cambian los procesos.

El Reverse Engineering en este aspecto, resulta de la ingeniería de datos y soporta directamente la migración de datos y el efecto de la estandarización de los datos.

Usando los tres esquemas del **paradigma** de la arquitectura de Reverse Engineering muestra las vistas lógicas de lo físico y vistas externas del usuario del ambiente del sistema. El Reverse Engineering es seleccionado para derivar los modelos de datos lógicos de la migración de sistemas, aunque esto es generalmente usado para optimizar las aplicaciones del código diseñado y perfil aerodinámico de operaciones del sistema.

Usando el Reverse Engineering también identifica, extrae e integra requerimientos críticos únicos contenidos en el sistema no legal en la migración del sistema apropiado.

Para involucrar una migración diseñada a través del sistema debe ser a través del Forward Engineering, muchos productos Reverse Engineering son tan buenos tradicionalmente que requieren ser identificados de las actividades del negocio de reingeniería. En suma, el sistema debe soportar la arquitectura de sistemas abiertos. Basándose en el Forward Engineering de modelos de datos desarrollados del análisis de Reverse Engineering hace la migración de sistemas más adaptable y fácil de mantener, reduciendo los costos de mantenimiento.

Dentro del Reverse Engineering los requerimientos principales para la migración de sistemas es recobrar la asociación de los requerimientos del negocio como son los niveles operacionales, tácticos y estratégicos.

El Reverse Engineering utiliza los requerimientos de datos y software para la migración de sistemas, lo cual es esencial para recobrar la asociación de los requerimientos del negocio al operacional, táctico y niveles estratégicos.

Los requerimientos del Reverse Engineering son comparados con los requerimientos del negocio durante el Forward Engineering. La arquitectura de la migración del sistema son evaluadas en los requerimientos técnicos. Como parte del Forward Engineering.

Los requerimientos del negocio, requerimientos técnicos y la calidad de los elementos de datos son evaluados para determinar la 'migratibilidad' e 'integridad' de migración de sistemas para formar las bases de la justificación económica de sistemas específicos en forward Engineering.

El programa Reverse Engineering soporta el análisis integrado y las actividades de rediseño/desarrollo requeridas para modernizar la selección de la migración de sistemas.

A continuación se muestra otro procedimiento que ha sido implementado para llevar a cabo el proceso de Reverse Engineering:

CASO

Los experimentos de Reverse Engineering han utilizado una serie de herramientas (OMT, CQLCODE, AWKSCRIPTS, y programas simples y análisis manuales), en nuestro estudio hemos observado un amplio rango de estilos e implementaciones inusuales que complican el proceso de Reverse Engineering.

Diccionario de Datos para Bases de Datos RelacionalesMS (Manejadores de Bases de Datos Relacionales).

Hemos escogido el diccionario de Bases de Datos para guiar el manejo del sistema de Base de Datos Relacionales

Tenemos diferentes orígenes de información:

Datos acumulados en el diccionario de datos, manuales de usuario y nuestros conocimientos para dominar el problema. Se analizaron las **llaves primarias** y secundarias para obtener una mayor información del sistema.

Primero obtuvimos el diagrama, las clases, características y llaves secundarias, preguntando al diccionario de datos. Muchas tienen una llave primaria que consiste de una característica con un identificador en el nombre lo cual asumimos que eran un llaves primarias. Algunas tablas tienen llaves secundarias múltiples examinando algunas tablas de cerca dedujimos que muchas asociaciones estaban implementadas con llaves foráneas y que la mayoría de las generalizaciones estaban implementadas con una llave primaria común.

Con el propósito de determinar generalizaciones , generamos un lista de posibles relaciones de uno a ninguno o de uno a través de preguntas que engloban a las llaves primarias. La lista resultante fue larga indicando una posible estructura generalizadora. Esta lista es el cierre de todas las subclases directas.

También consideramos asociaciones de llaves primarias. Automáticamente revisamos cada posible **llave foránea** contra cada llave secundaria para producir una lista de asociaciones posibles usando esta lista y la jerarquía heredada generamos una lista de asociaciones actuales.

Luego determinamos una multiplicidad en el query de datos un complicado factor fue que algunas veces un valor especial fue usado para representar un valor nulo que formalmente en el esquema no nos permitió anularlo. No obtuvimos los resultados correctos hasta que tomamos en cuenta el valor especial fue usado en uno o ambos finales de una asociación basados en la dirección transversal. Todo esto nos ayudo a llevar a cabo el proceso de Reverse Engineering.

3.5 PASOS DE LA INGENIERIA REVERSIVA EN BASES DE DATOS RELACIONALES

Los pasos que se siguieron en el desarrollo de Reverse Engineering se pueden explicar de la siguiente manera.

paso uno:
Prepara un modelo del objeto inicial.

Representar cada tabla como una clase tentativa. Todas las columnas de las tablas se convierten en los atributos de las clase.

paso dos:
Determinar las llaves secundarias.

Buscar los índices únicos, pero tener en cuenta que algunas llaves secundarias no pueden ser ejecutadas por índices únicos. La búsqueda automática de datos puede construir llaves secundarias potenciales. El conocimiento semántico puede ser usado para interpretar modelos de datos sugestivos.

paso tres:
Determinar los grupos de llaves foráneas.

Si se analiza la base de datos relacional se podrá determinar las llaves foráneas.
Tratar de resolver los homónimos, atributos con los mismos nombres que se refieren a diferentes cosas , y sinónimos, atributos con diferentes nombres que se refieren a una misma cosa.

El análisis de datos puede rechazar algunas hipótesis de las llaves foráneas. Aparte de los sinónimos y homónimos, el objetivo de una referencia de una llave foránea puede ser ambigua debido a su generalización. Por eso este paso se refiere a los grupos de llaves foráneas.

paso cuatro:
Refinar clases tentativas.

Aglomerar horizontalmente clases dentro de una sola clase.
Horizontalmente clases distribuidas tienen el mismo esquema.
Detectar funciones y restringir las que son representadas como tablas.

paso cinco:
Descubrir generalizaciones.

Analizar grupos de llaves foráneas, particularmente esas con cinco o diez o mas atributos relacionados. Una llave primaria que esta enteramente compuesta de una llave foránea de otra tabla debe indicar una generalización . La identidad derivada de una implementación es una generalización, la cual distingue una super clase y una tabla de subclases o la propagación de una vía de identidad de una asociación de 1 a 1.

paso seis:
Descubrir asociaciones.

Convertir una clase tentativa a una asociación cuando la llave secundaria está encadenada a dos o más llaves foráneas. Aplicar entendimiento semántico y restrictivo a algunas asociaciones como agregaciones.

La agregación es una parte de la relación.

paso siete:
Mejoramiento de la transformación:

Varias optimizaciones deben haber sido empleadas para preparar el esquema original del Bases de Datos Relacionales.

Transformaciones:

Convertir una clase a una clase unida cuando se necesite. Una clase unida es una asociación cuyas uniones pueden participar en asociaciones con otras clases.

paso ocho:

Deben ser simplemente representadas como un atributo. Por ejemplo es innecesario representar ciudad como una clase cuando el nombre de la ciudad es el único atributo de interés. Nuestra experiencia es que la mayoría de las asociaciones son binarias.

paso nueve:

Cuando sea posible debe eliminar una asociación imprecisa a una superclase en favor de una asociación mas restrictiva para una subclase.

3.6 APLICACIÓN DE LA INGENIERÍA REVERSIVA EN BASES DE DATOS RELACIONALES

REVERSE ENGINEERING REQUERIMIENTOS DE LAS BASES DE DATOS

La complejidad de las bases de datos ha creado gran impacto en proyectos Reverse Engineering. Sobre todo en la reconstrucción lógica de modelos de datos asociados a las reglas del negocio, información del dominio del negocio, requerimientos funcionales de las dependencias y la distribución de la arquitectura de datos organizacional y elementos de datos.

La Ingeniería Reversiva es una matriz diseñada para ir ligando los componentes de los modelos de datos para tener un soporte de la existencia física del sistema. Después la matriz diseñada puede ser usada para identificar el impacto de cambios de procesos y un plan de migración de datos a la migración de sistemas. Usando las herramientas CASE, el modelo de datos lógico, puede ser usado automáticamente para crear estructuras de tablas para seleccionar el DBMS, mientras la matriz diseñada es usada para desarrollar el mapeo requerido para la migración de datos del sistema legal para asegurar las tablas en el nuevo sistema.

Los aspectos técnicos incluidos dentro de esta herramienta son los siguientes:

1) dividir-dominar
2) extracción de reglas de negocios del software y estructura de datos
3) modelo de la administración
4) configuración del administrador
5) esquema de integración.

En estos casos el software es dividido en diferentes módulos para utilizar los requerimientos de la implementación funcional, funciones del negocio donde las implementan como partes de diferentes funciones de software. La estructura de datos presenta reportes en pantalla o puede ser usada para mapear elementos de datos almacenados en bases de datos o archivos. La estructura de datos definidas en el software y diccionario de datos, si existen, deben representar solo la estructura conceptual, externa o vistas físicas de elementos de datos.

El aspecto más difícil de la Reverse Engineering es descubrir las reglas del negocio y las entidades de los datos del software y estructuras de datos. Cantidad masiva de estructuras de datos y la asociación de código tiene una división-y-dominio para descubrir los elementos de datos y organizarlos en categorías. Nuestra aproximación es top-down y bottom-up. Durante el paso top-down analizaremos el material relevante para las vistas conceptuales (por ejemplo: reportes, pantallas).

Estas vistas utilizan rápidamente un conjunto de entidades conceptuales de las categorías relevantes de la estructura de datos para especificar su dominio. Usando las entidades de los datos derivadas del modelo del proceso del negocio o partición del modelo de datos del negocio a vistas.

A cada modelo de datos le corresponde una vista del proceso del negocio. Así como, las dependencias funcionales entre vistas y herencias, y las relaciones entre procesos de los modelos de alto nivel. Para identificar el acceso a la estructura de datos (creación, lectura, borrado y actualización) y aspectos del proceso.

Cuando se derivan más detalles de los modelos lógicos de datos y se ligan a esa estructura de datos para derivar las entidades de los datos usando la matriz diseñada. Todas las estructuras de datos y entidades de datos están ligadas y asociadas a los requerimientos operacionales, tácticos y/o estratégicos.

El análisis del Reverse Engineering debe examinar y analizar un sistema después de la inserción tecnológica o improvisación de ideas. Los objetivos para el Reverse Engineering también incluyen integración funcional, los cuales son usados para integrar los modelos de datos al sistema seleccionado.

La versión del Reverse Engineering del modelo de datos es asociado con una enciclopedia. La enciclopedia almacena información y lo separa en tres diccionarios. El plan diccionario contiene la planificación de la información; el dicionario de datos contiene el modelo de datos lógico y asocia la información; y el diccionario de datos contiene la estructura física y la información relatada.

Mejorar el diseño, es una evidencia física, la cual es ligada a la base de datos de Información del Catálogo Fuente (Information Resource Catalog IRC). El IRC contiene fuentes de información relevante de cada Reverse Engineering del sistema incluyendo los manuales de sistemas, código fuente, directivas, y resultados interpretados. Provee de un índice electrónico para la fuente de información reunida durante el ciclo de vida del Reverse Engineering.

El nuevo software es caro al desarrollarlo pero para mantener el viejo puede ser costoso y difícil de adaptar a los nuevos usos, el objetivo de la Reingeniería es mecánicamente reusar los anteriores esfuerzos desarrollados con el propósito de reducir el costoso mantenimiento y mejorar la flexibilidad del software.

3.7 PUNTOS A CONSIDERAR EN LA APLICACION DE LA INGENIERIA REVERSIVA EN LA MIGRACION DE BASE DE DATOS RELACIONALES

La reingeniería es aplicable a diversos software tales como códigos de programas, bases de datos e inferencia lógica. Este capítulo se enfoca en el tema de bases de datos y en particular a bases de datos relacionales. Existen varios motivos para llevar a cabo el Reverse Engineering de la base de datos:

1) Querer migrar entre los paradigmas de la base de datos de pasadas tecnologías a una moderna Base de Datos relacional o a una Base de Datos Orientada a Objectos.

Una tarea debe ser, migrar entre diferentes implementaciones de un paradigma de bases de datos, por ejemplo de un Bases de Datos Relacionales a otro Bases de Datos Relacionales.

Reverse Engineering puede dilucidar entre una pobre documentación existente del software cuando los desarrolladores no son capaces para hacerlo.

Hemos derivado beneficios substanciales de Reverse Enginnering del Software. El reverse Engineering provee un inusual origen de visión de los modelos. Los modelos orientados a objetos proveen de un lenguaje natural para facilitar el proceso de Reengineering.

Un modelo orientado a objetos puede describir la existencia del software. Este articulo adopta la técnica de moldeamiento del objeto (OMT) y anotaciones para datos de modelo.

1)Para llevar a cabo el Reverse Engineering proponemos un proceso **robusto**. Reconocemos generalizaciones en nuestro proceso de Reverse Engineering.
2) nuestro proceso enfatiza el análisis de las llaves secundarias más que de las llaves primarias.
3) Incorporamos tres fuentes de información el esquema, modelos de datos, y nuestro entendimiento semántico de aplicación. Reportamos nuestras experiencias con ejemplos industriales.

Estrategias tipicas de implantación para Ingeniería Adelantada (Forward Engineering).

Las Estrategias de Reverse Engineering deben tener en cuenta practicas pasadas. A menudo el modelo o su implementación como un esquema fue fundamentalmente violado en la practica de un buen diseño. Encontramos violaciones de la mayoría de las aseveraciones que son hechas en la existencia del método de Reverse Engineering.

-Clases. Cada clase esta usualmente implementada como una tabla con una columna para cada característica. Indices únicos usualmente están especificados en llaves primarias y secundarias. Hay diferentes acercamientos para construir una llave primaria. A menudo la llave primaria es un solo identificador generado. algunas veces uno o mas se atribuye desde el problema dominante que sirve como la llave primaria.

Finalmente una clase puede derivar su identidad de otra clase o clases a través de una relación.

Las clases deben ser divididas y combinadas para formar tablas Las clases relacionadas una a una son algunas veces mapeadas en una sola tabla. Una clase debe ser dividida en fragmentos verticales u horizontales. Un mapa vertical dividido se atribuye a columnas en tablas diferentes, si la clase no tiene atributos o características no debe ser mapeada en la tabla. A esto llamamos una clase ligera.

- Generalizaciones. El acercamiento usual para las generalizaciones es mapear cada clase en una jerarquía para una tabla con columnas, correspondientes a los atributos y a las llaves primarias comunes. Para cada registro en la tabla de superclase hay una registro para al menos una tabla de subclase. Indices únicos son creados para llaves primarias y secundarias. La mayoría de las generalizaciones son exhaustivas sin una herencia múltiple, en algunos casos hemos visto clases sin atributos que no han sido mapeados dentro de las tablas. Entonces algunos registros de las superclases no tienen registros correspondientes en esta subclase.

Un discriminador puede ser o no usado. Los valores para un discriminador o eliminador son usualmente similares a los nombres de las subclases, incluyendo cualquier subclase ligera. En un nivel múltiple de jerarquía debe haber un discriminador a cualquier nivel. Otras implementaciones de generalizaciones ocurren frecuentemente:
Mapear cada subclase a una tabla incluyendo columnas para todos los atributos heredados.

Mapear la raíz de la superclase en una tabla, incluyendo columnas para todos los atributos de las subclases.

- Asociaciones. La asociación mas común construida es la llave foránea usada para implementar asociaciones binarias que no son muchos a muchos. La llave foránea esta usualmente oculta en uno de los finales de la asociación. Los nombres de la columna de la llave foránea son nombre semánticamente significativos. Algunas veces el nombre de cada columna de llave foránea une el nombre de una columna referenciada.

Indices únicos puede ejecutar restricciones múltiples particularmente para asociaciones calificada.

Usualmente los tipos de datos de la columna foránea une los tipos de datos de las columnas de las correspondientes llaves primarias o secundarias. Hemos visto esquemas elaborados en las cuales las llaves foráneas están encadenados en otra información. Las asociaciones una a una, ninguna a una pueden ser implementadas con una llave primaria común. Esta implementación es indistinguible de una generalización.

Si una asociación envuelve una superclase ligera debe ser empujada a las subclase resultante en la llave foránea múltiple o en una llave foránea la cual podría apuntar a tablas múltiples.

Las asociaciones muchos a muchos están usualmente implementadas con una tabla de intersección las dos llaves foráneas se combinan para formar una llave secundarias.

En algunos casos la tabla de intersección debe tener otra llave secundaria típicamente un identificador generado. Las asociaciones que nos son muchos a muchos deben ser implementadas en una tabla separada a discreción del diseñador.

CAPITULO 4

IMPLEMENTACION DE LA INGENIERIA REVERSIVA EN SCORE

CAPITULO 4. IMPLEMENTACION DE LA INGENIERIA REVERSIVA EN SCORE

Los costos del software han crecido dramáticamente, convirtiéndose en la parte más costosa de muchos sistemas basados en computadora. Se fijan agendas y fechas de terminación pero raramente se cumplen. A medida que crece un sistema de software, la calidad se hace más sospechosa. Los responsables de los proyectos de desarrollo de software disponen de pocos datos históricos que se puedan usar como guías cada vez de menos control sobre el uso de un proyecto.

Desgraciadamente muchas veces por la rapidez del desarrollo de la aplicación no se tiene una adecuada documentación así como un buen análisis y diseño de la misma, con la cual se crean problemas al querer modificarla.

Una opción incluida en CASE DE ORACLE es la aplicación de Ingeniería en Reversiva que está diseñada para recuperar información de la aplicación existente en Oracle, esto implica que se podrá documentar una aplicación obteniendo los diagramas necesarios para el CASE *Dictionary, y además facilita la implementación de nuevos módulos.

En este capítulo se explica la aplicación de la Ingeniería Reversiva del CASE Oracle en un sistema existente llamado SCORE (Sistema de Control de Operaciones y Registro Estadístico) de la empresa SystemHouse de México, S.A. de C.V., ya que carecía de una adecuada implementación de dicho sistema en CASE de Oracle.

Por la fecha en que se desarrolló SCORE no se aplicó la herramienta CASE ORACLE, debido a que no se contaban con las herramientas necesarias para su aplicación.

Debido a los avances y necesidades de la empresa SYSTEMHOUSE DE MEXICO, S.A. DE C.V., se da cuenta que es necesario contar con un sistema bien documentado con el fin de hacer modificaciones y/o ampliaciones posteriores del mismo. Por lo tanto, se investiga qué herramienta daría apoyo a dicho proceso, eligiendo a la herramienta CASE de Oracle ya que sería la más adecuada para su documentación.

4.1 GENERALIDADES DEL SISTEMA SCORE

El Sistema de Control de Operaciones y Registro Estadístico (SCORE) es un software que apoya a las casas de cambio en el ciclo completo de su operación: la operación, la venta, la ejecución, la tesorería y la generación de informes estadísticos y de resultados.

El sistema puede correr en sistema operativo Unix y OS/2, así como el Manejador de Base de Datos Oracle versión 6.0 que incluya las siguientes herramientas:

* SQL FORMS v3.0
* SQL REPORTS v2.0
* **SQL (Structured Query Language)**

Para su ejecución es necesario contar con los siguientes requerimientos de hardware:

* Computadora tipo PC con procesador 80386 o superior.
* Memoria principal 4 MB mínimo.
* Disco Duro con 200 MB para la instalación de Oracle y la aplicación SCORE.

El sistema SCORE tiene 6 años de estar operando en el mercado mexicano, Monex es la casa de cambio más rentable, la cual en 1990 no figuraba entre las casas de cambio más productivas, en el año de 1991 adquirió el sistema SCORE y paulatinamente fué mejorando hasta alcanzar a la fecha el primer lugar.

Además de la Casa de Cambio Monex, Asesoría cambiaria ASECAM adquirió el sistema SCORE en 1993 y MONEX Guadalajara lo adquirió en 1994.

SCORE está compuesto por los siguientes módulos:

- PROMOCION:
  - Registra operaciones manejando varias divisas y formas de liquidación en un solo contrato.
  - Consulta costos autorizados directamente en pantalla.
  - Consulta y programa la agenda de trabajo.
  - Cierra operaciones manejando varias divisas y formas de liquidación en un solo contrato.

- PARAMETROS Y POLITICAS DE OPERACION

- OPERACION:
  - Monitorea el mercado y establece las políticas de operación.
  - Controla las operaciones extraordinarias con números de autorización.
  - Controla la posición de las divisas en línea.
  - Pacta las operaciones interbancarias (mayores).
  - Pacta operaciones para cobertura de fondos.

- ADMINISTRACION DE FONDOS:
  - Asigna cuentas de cobro y pago.
  - Controla compromisos de cobro y pago.
  - Concilia cuentas bancarias.
  - Estima saldos y programa de inversiones.
  - Optimiza el uso del flujo comprometido no cobrado basado en un modelo probabilístico.

- CONTROL CONTABLE:
  - Genera pólizas consolidadas de la operación diaria, semanal o mensual.
  - Genera archivos de pólizas para ser procesadas en un sistema de contabilidad exterior.
  - Elabora informes contables: catálogo de cuentas, lista de movimientos por cuenta (auxiliares) y balanza de comprobación.
  - Consultar saldos y movimientos contables de las cuentas.
  - Registra movimientos que no son generados automáticamente por el sistema.
  - Genera informes sobre la operación contable diaria y comprobar la dualidad de los asientos contables.

- EJECUCION:
  - Prepara comprobantes y documentos de operación.
  - Controla la impresión de documentos.
  - Controla archivo de operaciones y clientes.
  - Prepara depósitos.
  - Ejecuta transferencias de fondo.

- MENSAJERIA:
    - Elabora órdenes de mensajería.
    - Controla mensajeros.
    - Entrega operaciones.
    - Registra resultados de la mensajería.

- ADMINISTRACION DEL SISTEMA

- CATALOGOS

Dentro de los parámetros que maneja el sistema se encuentran los siguientes:
- Maneja clases de clientes (carteras con diferente riesgo)
- Determina niveles de riesgo de los métodos de pago y cobro
- Maneja el estado de las divisas mediante: efectivo, documento, remesas, transferencia, depósito.

Tomando en consideración que el sistema está desarrollado en el Manejador de Bases de Datos Oracle, se buscan alternativas por esta misma vía, encontrándose que dentro del Diccionario de CASE existe la opción de "Ingeniería Reversiva" y se decide aplicarla para saber que resultados se obtendrían.

4.2 APLICACION DE LA UTILERIA DE LA INGENIERIA REVERSIVA PARA BASE DE DATOS EN EL SISTEMA SCORE

Para la aplicación de Reverse Engineering se procedió a consultar los respectivos manuales de usuario.

Muchas de las aplicaciones existentes no tienen una documentación formal de las etapas tempranas del desarrollo, muchas veces almacenan datos valuables del negocio. Cuando hay necesidad de desarrollar un sistema para reemplazar o actualizar el existente, los pasos iniciales podrían ser:

- Dónde se están almacenando los datos

- En qué formato están los datos

- Cómo usa el sistema existente los datos

- Cómo la aplicación del sistema se ajusta a la práctica actual del negocio.

REVERSE ENGINEERING o desarrollo "bottom-up" es el proceso de:

- Derivar las definiciones de base de datos y uso de módulo de datos (por ejemplo: las definiciones reajustadas) desde el sistema real.

- Derivación del modelo del negocio desde los datos a los cuales se les aplicó el REVERSE ENGINEERING.

- Analizar los nuevos requerimientos del negocio y compararlos con los derivados del sistema existente.

- Rediseñar e implementar un nuevo sistema, si, necesariamente esto satisface los nuevos requerimientos del sistema.

CASE *Dictionary contiene una utilería para aplicar REVERSE ENGINEERING de los objetos de la base de datos (Reverse Engineer Data Base Utility), la cual parcialmente automatiza el proceso de REVERSE ENGINEERING.

- Relationship Definition, al igual que la anterior opción es necesario capturar la información requerida en este punto, la cual se refiere a las relaciones entre las diferentes entidades del sistemas que fueron dadas de alta anteriormente. (ver apéndice 4)

- Unique Identified Definition, en esta pantalla se define si algún campo va a ser llave foránea y primaria sabemos que al tener una relación y especificarla ya sea en el diagrama o en el diccionario se heredan campos a otra tabla, por lo tanto es necesario precisar qué tipo de llave se va a heredar y si formar parte de la llave primaria de la tabla de la cual se hereda este campo. (ver apéndice 3)

Al finalizar esta etapa pudimos observar que varias tablas de la Base de Datos no eran usadas por el sistema y por lo tanto se procedió a depurar la Base de Datos, así como las definiciones en CASE Dictionary.

Se obtuvo un Diagrama entidad-relación actualizado y confiable (ver apéndice 5).

Se generó la documentación faltante en CASE Dictionary.

NOTA: Si las llaves primarias y foráneas están presentes en el diccionario, éstas son automáticamente identificadas.

Para asistir a ese análisis, el diseñador del sistema y el usuario final deberán ser consultados.

Las convenciones en nombres deber ser introducida en este punto para facilitar el uso de los mismos en etapas posteriores del desarrollo. Por ejemplo, el uso de prefijos podrían ser empleados para diferenciar entre un subsistema y otro.

Hay que tener cuidado cuando los sinónimos son usados por elementos de datos que están presentes en una o más tablas. Por ejemplo, 'Clave Organización' puede ser la llave primaria en una tabla, pero puede ser referida como 'Clave Compañía' en una tabla relacionada.

CREATING MODULE DATA USAGES (CREACION DEL MODULO DE DATOS USADOS)

Las asociaciones necesitan ser definidas entre módulos y tablas/vistas/columnas para empezar una documentación de referencia cruzada para la etapa de la implementación.

El módulo de datos puede ser creado usando el Module Data Usage Screen (Pantalla de Módulo de Datos) y el Module Data Usage (Detailed) Screen (Pantalla de Módulo de Datos Detallado).

REVERSE ENGINEERING FOR SQL FORMS (REVERSE ENGINEERING PARA FORMAS DE SQL)

CASE *Generator permite al usuario trasladar definciones de aplicaciones en SQL FORMS a definiciones de módulo en CASE *Dictionary.

El REVERSE ENGINEERING de ORACLE que se aplicó, es un proceso manual y/o automático que toma un componente de un sistema existente y lo transforma en una definición lógica dentro de la herramienta CASE.

A continuación se especifican los pasos seguidos para la aplicación de REVERSE ENGINEERING en el sistema SCORE.

PASO UNO : ESTUDIO DEL SISTEMA ACTUAL

Como primer paso se procedió a la recopilación de documentación existente del sistema incluyendo Diagramas entidad-relación, reportes de tablas, vistas y columnas de la Base de Datos así como alguna información relacionada al uso del sistema. También se tuvo apoyo por parte de los diseñadores y usuarios del mismo, para de esta forma, poder familiarizarse con el sistema, su estructura y como trabaja.

A continuación se muestran un ejemplo de una tabla que forma parte de la Base de Datos del sistema Score después de haber realizado el análisis. (ver Apéndice 1).

```
tabla= COMPRA_VENTA
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_OPE                        NOT NULL NUMBER(6)
 TIPOMOV_CV                     NOT NULL CHAR(1)
 CVE_DIV                        NOT NULL CHAR(3)
 CVE_PROD                                CHAR(3)
 MONTO_CV                       NOT NULL NUMBER(16,2)
 TCMN_CV                                 NUMBER(12,6)
 MONTOMN_CV                              NUMBER(16,2)
 FECHVAL_CV                     NOT NULL DATE
 NEMOFP                                  CHAR(3)
 DETALLEFP_CV                            CHAR(150)
 TCUSD_CV                                NUMBER(12,6)
 COSTOMN_CV                              NUMBER(12,6)
 LIQUIDADO_CV                            DATE
 ENTREGA_CV                              CHAR(1)
 FUTURO_CV                               CHAR(1)
 CVE_MOV                                 CHAR(8)
 CVE_MOV1                                CHAR(8)
 FECHASG_CV                              DATE
 FECHPAGO_CV                             DATE
 CVEID_CV                                CHAR(3)
 CVEDET_CV                               NUMBER(3)
 NEMO_CTA                                CHAR(10)
 CONFIRMA_CV                             CHAR(10)
 INDI_CV                                 CHAR(1)
 REF_CV                                  CHAR(10)
 CVE_PROD2                      NOT NULL CHAR(3)
```

PASO DOS : APLICACION DE LA INGENIEIRA REVERSIVA DATA BASE UTILITY (CASE).

Esta utilería se corre desde el menú de Reconcilation Menu de CASE Dictionary, se pueden elegir cualquier combinación de tablas,**secuencias, clusters, índices y vistas**. Esta opción crea un archivo llamado CDRF.lis muestra el resultado de las opciones elegidas para el REVERSE ENGINEERING.

PARAMETROS DE ENTRADA:

Cuando se corre esta utilería se deben de introducir los siguientes parámetros de entrada:

NOMBRE DE LA APLICACION DEL SISTEMA: Aquí se introduce el nombre del sistema de aplicación al cual se le va a aplicar REVERSE ENGENEERING de los objetos de la Base de Datos.

NUMERO DE VERSION: Introducir el número de versión de la aplicación del sistema.

NOMBRE DEL USUARIO: Introducir el nombre del usuario propietario de los objetos Oracle, para hacer REVERSE ENGENEERING.

TIPOS DE OBJETOS DE LA BASE DE DATOS.Introducir el tipo de los objetos de la Base de Datos a los cuales se les va ha aplicar Reversing.

1.- Teclear el signo de ayuda (?) para obtener los tipos de objetos de la Base de Datos disponibles a elegir son :

* TS TABLESPACE
* S SEQUENCE
* T TABLE
* C CLUSTER
* I INDIDEX
* V VIEW

2.- Teclear la abreviación elegida del objeto de la Base de Datos (por ejemplo teclear T para Tabla). Será necesario teclear el nombre indivual del objeto de este tipo. Teclea un punto (.) para completar la entrada del nombre del objeto. El tipo de objeto de la Base de Datos sera desplegado.

3.- Teclea el signo de porcentaje (%) para elegir todos los tipos de objetos de la Base de Datos automáticamente.

4.- Teclear punto (.) para finalizar la entrada de tipos de objetos de datos.

VERIFICACION.

Teclear una 'Y' si se desea verificar los objetos seleccionados de la Base de Datos, y 'N' si no se desea verificarlos. Si se elige 'Y'una lista de los objetos se despliega y se pregunta si se desea continuar o salir de esta utilería. Si continua con esta utilería a los objetos de la Base de Datos seleccionados se les aplica el Reverse Engeneering y se produce un reporte con el nombre CDRF.lis. (ver apéndice 2)

NOTA: El REVERSE aplicado a Tablespace, Secuencias, Vistas e índices, no duplica los objetos existentes en el diccionario. Donde existe una tabla en el Diccionario la utilería de Revese Engeneering hará el Reverse de cualquier columna,constraints, y llave constraint que no exista en la definición de tabla y lo mismo sucede con los demas tipos de objetos.

PASO TRES: REVISION DE LA INFORMACION GENERADA DEL REVERSE ENGENEERY DENTRO DEL CASE *Dictionary.

Una vez hecho el REVERSE ENGINEERING de los objetos de la Base de Datos se procedió ha verificar las definiciones dentro de Case *Dictionary en las siguientes opciones (ver apéndice 2):

1.- Table Definition Screen.
2.- View Definition Screen.
3.- Index Definition Screen.
4.- Sequence Definition Screen.
5.- Cluster Definition Screen.
6.- Storage Definition Screen.

Algunas de estas opciones no fueron cargadas con información después del REVERSE ENGINEERING, debido a que no se contaba con datos en la Base de Oracle. Por lo tanto se procedió a analizar aquellas opciones del CASE Dictionary en las cuales se obtuvo información para completar los datos faltantes.

Al término de este paso se completo la información de las siguientes pantallas del CASE *Dictionary : Table Definition Screen, Index Definition Screen y Sequence Definition Screen. (ver apéndice 3)

PASO CUATRO : ANALISIS DE LA BASE DE DATOS DEL SISTEMA EXISTENTE.

Para este paso se revisó la información obtenida en el paso uno, pero la información concerniente a llaves foráneas y primarias no existía, sino que la dedujimos.

Los nombres de las llaves eran similares a los nombres de los campos de la tabla por lo que para encontrar llaves secundarias adicionales se hicieron consultas a la Base de Datos para obtener los campos comunes en varias tablas y de esta forma deducir que alguno de esos campos era algún tipo de llave. Mecánicamente generamos asociaciones tentativas de las llaves secundarias y foráneas. Después aplicamos transformaciones para refinar el modelo existente.

Nuestro proceso se enfatizó en el análisis de las llaves secundarias más que en las llaves primarias, ya que la determinación de estas fué un paso importante en el proceso de REVERSE ENGINEERING, porque encontrar llaves secundarias es más fácil que encontrar llaves primarias.

La llave foránea está usualmente oculta en uno de los finales de la asociación. Los nombres de la columna de la llave foránea son nombres semánticamente significativos. Algunas veces el nombre de cada columna de llave foránea une el nombre de una columna referenciada, por lo tanto son difíciles de identificar. Generalmente los tipos de datos de la columna foránea une los tipos de datos de las columnas de las correspondientes llaves primarias o secundarias.

Las asociaciones uno a uno, ninguno a uno , pueden ser implementadas con una llave primaria común. Las asociaciones muchos a muchos están usualmente implementadas con una tabla de intersección , las dos llaves foráneas se combinan para formar una llave secundaria.

Una vez obtenidas las llaves primarias, secundarias y foráneas se procedió a la actualización del diagrama entidad-relación y la depuración de la Base de Datos. Se introdujo información necesaria para una adecuada documentación del sistema dentro de CASE Dictionary en las siguientes opciones:

- Entity/Atributte Definition, en la cual se dan de alta todas las entidades y sus campos respectivos encontrados en el análisis hecho anteriormente de la Base de Datos del sistema, ya que al hacer el REVERSE ENGINNNERING de los objetos de la Base de Datos, esta información no es llenada y es muy necesaria para seguir todo el proceso del CASE Dictionary. (ver apéndice 4)

- Relationship Definition, al igual que la anterior opción es necesario capturar la información requerida en este punto, la cual se refiere a las relaciones entre las diferentes entidades del sistemas que fueron dadas de alta anteriormente. (ver apéndice 4)

- Unique Identified Definition, en esta pantalla se define si algún campo va a ser llave foránea y primaria sabemos que al tener una relación y especificarla ya sea en el diagrama o en el diccionario se heredan campos a otra tabla, por lo tanto es necesario precisar qué tipo de llave se va a heredar y si formar parte de la llave primaria de la tabla de la cual se hereda este campo. (ver apéndice 3)

Al finalizar esta etapa pudimos observar que varias tablas de la Base de Datos no eran usadas por el sistema y por lo tanto se procedió a depurar la Base de Datos, así como las definiciones en CASE Dictionary.

Se obtuvo un Diagrama entidad-relación actualizado y confiable (ver apéndice 5).

Se generó la documentación faltante en CASE Dictionary.

CONCLUSIONES

1.- CASE ORACLE apoyado por todas las herramientas que lo componen (CASE *Dictionary, CASE *Designer, CASE *Generator, entre otras) proporciona aplicaciones confiables más rápido que las técnicas de desarrollo tradicionales.

Todas las reglas del negocio, identificadas durante el análisis se incorporan rápidamente en la aplicación. La integridad referencial, la validación del dominio y límites , las referencias cruzadas de campo y las restricciones incluídas ayudan a garantizar que todas las aplicaciones cumplan con las necesidades de los negocios sin necesidad de trabajo de desarrollo repetitivo y propenso a errores.

2.- CASE ORACLE es una fuente global de documentación que permite a los diseñadores de sistemas, llevar a cabo rediseños posteriores.

3.- Un diseño recuperado debe almacenar ligas entre la información recuperada y las fuentes originales. Esto permite un seguimiento entre el diseño recuperado y la implementación original.

4. Muchas veces la fuente de información puede estar desactualizada o incorrecta. Hay que usarlos cuidadosamente, pueden proporcionar información de un programa, pero también pueden estar mal o ser falsos. El código fuente es la sentencia final de lo que el sistema hace actualmente.

5.- Para la implementación de la Ingeniería Reversiva es necesario lograr un dominio de la información recabada de la aplicación , ya que esto facilitará entender mejor su funcionamiento y por lo tanto seguir la secuencia indicada de pasos para llegar a una exitosa culminación del proceso de la Ingeniería Reversiva.

6.- La Ingeniería Reversiva reconoce el potencial del análisis de una aplicación, así como los componentes del sistema (tablas, vistas, relaciones, índices, llaves primarias, secundarias, etc.), para el mejoramiento de la información en un plan de migración de datos.

7.- El proceso de Ingeniería Reversiva podrá con frecuencia ser hecho en un sistema de software "viejo" cuya documentación está desactualizada o no existe. Estos sistemas son los más idóneos para el rengineering : reescribir el sistema para mejorar

su entendimiento, fácil mantenimiento, eliminar código inservible, etc. Es más fácil cambiar el diseño que el código fuente.

Una nueva versión del sistema generado desde el diseño puede ser desarrollado usando técnicas modernas y lenguajes de programación. Esto dará como resultado un sistema mejor estructurado, documentado y mantenido más fácilmente que la versión anterior. Así el Reverse Engineering como una parte de la Re-ingeniería nos sirve para alargar la vida de los viejos sistemas.

8.- Como resultado de este estudio y su implementación, podemos afirmar : que aplicar la la Ingenierá Reversiva en un sistema existente es útil para poder tener un mayor control en cuanto a su documentación y como resultado los nuevos requerimientos del negocio podrán ser implementados con mayor facilidad.

9.- Una vez aplicada la Ingeniería Reversiva en SCORE obtuvimos como resultado una Base de Datos depurada, ya que nos dimos cuenta que había muchas tablas no utilizadas y que sólo estaban ocupando espacio innecesario.

10.- Se identificaron las llaves primarias y foráneas de las tablas así como sus relaciones, obteniendo la información faltante para completar los datos de la aplicación de CASE ORACLE.

11.- El proceso de Reverse Engineering podrá con frecuencia ser hecho en un sistema de software viejo cuya documentación está desactualizada o no existe. Estos sistemas son los más idóneos para el re-engineering: reescribir el sistema para mejorar su entendimiento, fácil mantenimiento, eliminar código inservible, etc. Es más fácil cambiar el diseño que el código fuente.

12.- Mientras varios autores proponen procedimientos a seguir para aplicar la Reverse Engineering, la verdad es que el procedimiento se crea dependiendo de la aplicación y el lenguaje en que está implementado, tal como lo constatamos.

13.- Es necesario contar con mayor información acerca de la Reverse Engineering ya que es un término relativamente nuevo, así mismo, las herramientas que lo aplican aún se encuentran en sus primeras etapas y creemos que aún hay mucho camino por recorrer en este campo.

APENDICES

APENDICE 1

RELACION DE TABLAS DEL SISTEMA SCORE

APENDICE 1.
Relación de tablas del sistema Score.

```
SQL> @tablas

tabla= AGENDA
 Name                           Null?    Type
 ------------------------------ -------- ----
 NEMO_PRM                                CHAR(10)
 NEMO_CLI                                CHAR(10)
 ACCION                                  CHAR(30)
 ACUERDO                                 CHAR(30)
 FECHA                                   DATE
 HORA                                    CHAR(5)
 CVE_RES                                 CHAR(3)

tabla= AUTORIZACION
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_AUT                                 CHAR(6)
 REF_AUT                                 CHAR(10)
 FECHA_AUT                               DATE
 CAUSA_AUT                               CHAR(40)
 INDI_AUT                                CHAR(1)
 LUGAR                                   CHAR(7)

tabla= AUTORIZA_CHEQUES
 Name                           Null?    Type
 ------------------------------ -------- ----
 NEMO_CTA                                CHAR(10)
 FOLIO_CHE                               NUMBER
 CVE_ACH                                 CHAR(10)
 FECH_ACH                                DATE
 MOTIVO_ACH                              CHAR(30)
 CVEID_PRM                               CHAR(10)

tabla= AUTORIZA_FACTURA
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_AFC                                 NUMBER(5)
 CVE_FAC                                 NUMBER(6)
 FECH_AFC                                DATE
 MOTIVO_AFC                              CHAR(50)

tabla= BITACORA_LOTE_CONCILIACIONES
 Name                           Null?    Type
 ------------------------------ -------- ----
 BITC_LOTE                      NOT NULL NUMBER(8)
 BITC_FECHA                              DATE
 BITC_HORA                               CHAR(5)
 BITC_USUARIO                            CHAR(10)
 BITC_TIPO_CONC                          CHAR(3)
```

```
 BITC_NUM_PARTIDAS                         NUMBER(6)

tabla= BITACORA_LOTE_TRANSMISIONES
 Name                           Null?     Type
 ------------------------------ --------- ----
 BITA_CLAVE_SISTEMA             NOT NULL CHAR(5)
 BITA_LOTE                      NOT NULL NUMBER(8)
 BITA_TIPO_TRANSMISION          NOT NULL CHAR(1)
 BITA_FECHA                              DATE
 BITA_USUARIO                            CHAR(10)
 BITA_REFERENCIA                         CHAR(30)
 BITA_HORA                               CHAR(5)
 BITA_NUM_MOVS                           NUMBER(6)

tabla= CALENDARIO
 Name                           Null?     Type
 ------------------------------ --------- ----
 FECHA_CAL                               DATE
 DIA_CAL                                 CHAR(9)
 INDI_CAL                                CHAR(1)

tabla= CHEQUES
 Name                           Null?     Type
 ------------------------------ --------- ----
 NUM_CHE                                 CHAR(10)
 FECH_CHE                                DATE
 FECONCI_CHE                             DATE
 BAN_CHE                                 CHAR(1)
 PORTADOR_CHE                            CHAR(60)
 CVE_MOV                                 CHAR(8)
 MONTO_CHE                               NUMBER(14,2)
 NEMO_MNX                                CHAR(10)
 FOLIO_CHE                               NUMBER
 CVE_OPE                                 NUMBER(6)
 CVEDET_CV                               NUMBER(3)
 FECHA_CON                               DATE
 DIAS                                    NUMBER(4)

tabla= CLASES_CLIENTES
 Name                           Null?     Type
 ------------------------------ --------- ----
 NEMO_CLS                                CHAR(10)
 NUM_CLS                                 NUMBER(2)
 NOMBRE_CLS                              CHAR(25)
 MONMAX_CLS                              NUMBER(14)

tabla= CLIENTES
 Name                           Null?     Type
 ------------------------------ --------- ----
 CVE_CLI                                 NUMBER(6)
 NEMO_CLI                       NOT NULL CHAR(10)
 NOM_CLI                                 CHAR(40)
 TEL1_CLI                                CHAR(12)
```

```
 TEL2_CLI                                  CHAR(12)
 SENSIBI_CLI                               CHAR(1)
 LIMCRE_CLI                                NUMBER(16)
 NOMOPE_CLI                                CHAR(25)
 MONTOMAX_CLI                              NUMBER(14)
 NUM_CLS                                   NUMBER(2)
 NEMO_PRM                                  CHAR(10)
 FECMOD_DP                                 DATE
 CVE_TIPO                         NOT NULL CHAR(1)
 RFC                                       CHAR(15)

tabla= CLIENTES_DIR
 Name                             Null?    Type
 -------------------------------- -------- ----
 NEMO_CLI                                  CHAR(10)
 CVE_DIR                                   NUMBER(2)
 DIR_CLI                                   CHAR(60)
 COL_CLI                                   CHAR(20)
 CIU_CLI                                   CHAR(18)
 PAIS_CLI                                  CHAR(18)
 CP_CLI                                    CHAR(6)
 NEMO_RUT                                  CHAR(5)

tabla= COMISIONES
 Name                             Null?    Type
 -------------------------------- -------- ----
 CVE_COM                                   NUMBER(3)
 MONINF_COM                                NUMBER(12)
 PTGE_COM                                  NUMBER(3)

tabla= COMPRA_VENTA
 Name                             Null?    Type
 -------------------------------- -------- ----
 CVE_OPE                          NOT NULL NUMBER(6)
 TIPOMOV_CV                       NOT NULL CHAR(1)
 CVE_DIV                          NOT NULL CHAR(3)
 CVE_PROD                                  CHAR(3)
 MONTO_CV                         NOT NULL NUMBER(16,2)
 TCMN_CV                                   NUMBER(12,6)
 MONTOMN_CV                                NUMBER(16,2)
 FECHVAL_CV                       NOT NULL DATE
 NEMOFP                                    CHAR(3)
 DETALLEFP_CV                              CHAR(150)
 TCUSD_CV                                  NUMBER(12,6)
 COSTOMN_CV                                NUMBER(12,6)
 LIQUIDADO_CV                              DATE
 ENTREGA_CV                                CHAR(1)
 FUTURO_CV                                 CHAR(1)
 CVE_MOV                                   CHAR(8)
 CVE_MOV1                                  CHAR(8)
 FECHASG_CV                                DATE
 FECHPAGO_CV                               DATE
 CVEID_CV                                  CHAR(3)
```

```
 CVEDET_CV                                 NUMBER(3)
 NEMO_CTA                                  CHAR(10)
 CONFIRMA_CV                               CHAR(10)
 INDI_CV                                   CHAR(1)
 REF_CV                                    CHAR(10)
 CVE_PROD2                        NOT NULL CHAR(3)

tabla= COMPRA_VENTA_CANCELADOS
 Name                              Null?    Type
 --------------------------------- -------- ----
 CVE_OPE                                    NUMBER(6)
 TIPOMOV_CV                                 CHAR(1)
 CVE_DIV                                    CHAR(3)
 CVE_PROD                                   CHAR(3)
 MONTO_CV                                   NUMBER(16)
 TCMN_CV                                    NUMBER(12,6)
 MONTOMN_CV                                 NUMBER(16)
 FECHVAL_CV                                 DATE
 NEMOFP                                     CHAR(3)
 DETALLEFP_CV                               CHAR(150)
 TCUSD_CV                                   NUMBER(12,6)
 COSTOMN_CV                                 NUMBER(12,6)
 LIQUIDADO_CV                               DATE
 ENTREGA_CV                                 CHAR(1)
 FUTURO_CV                                  CHAR(1)
 CVE_MOV                                    CHAR(8)
 CVE_MOV1                                   CHAR(8)
 FECHASG_CV                                 DATE
 FECHPAGO_CV                                DATE
 CVEID_CV                                   CHAR(3)
 CVEDET_CV                                  CHAR(3)
 NEMO_CTA                                   CHAR(10)
 CONFIRMA_CV                                CHAR(10)
 INDI_CV                                    CHAR(1)

tabla= CONCILIACION_BANCOS_MNX
 Name                              Null?    Type
 --------------------------------- -------- ----
 CVE_MOV                                    CHAR(8)
 FECH_CBM                                   DATE

tabla= CUENTAS_BCO_CLIENTE
 Name                              Null?    Type
 --------------------------------- -------- ----
 NEMO_CLI                                   CHAR(10)
 CVE_DIV                                    CHAR(3)
 CVE_PROD                                   CHAR(3)
 DESTINO_BCO                                CHAR(150)
 NEMO_BCO                                   CHAR(10)

tabla= CUENTAS_CONTABLES
 Name                              Null?    Type
 --------------------------------- -------- ----
```

```
 NUM_CTA                                   CHAR(16)
 DESC_CTA                                  CHAR(40)
 NIV_CTA                                   CHAR(1)
 TIPO_CTA                                  CHAR(3)
 SALDOI_CTA                                NUMBER(16)
 FECHA_CTA                                 DATE
 NEMO_CTA                                  CHAR(10)
 NAT_CTA                                   CHAR(1)

tabla= CUENTA_TRANS_DEST
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 CUEN_NEMO_CTA                             NOT NULL CHAR(10)
 CUEN_NUMERO_CUENTA                        NOT NULL CHAR(20)
 CUEN_INDICA_PROP                                   CHAR(1)
 CUEN_PLAZA_BCO                                     CHAR(5)
 CUEN_TIPT_CLAVE                                    CHAR(5)

tabla= DESTINOS_COBRO_MONEX
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 CVE_DIV                                            CHAR(3)
 DESTINO_MNX                                        CHAR(40)
 NEMO_CTA                                           CHAR(10)
 CVE_TD                                             CHAR(3)
 NEMO_MNX                                           CHAR(10)
 NUMFAX_MNX                                         CHAR(12)
 BANCHE_MNX                                         CHAR(1)
 BANTRF_MNX                                         CHAR(1)
 BANINV_MNX                                         CHAR(1)
 BANCAJ_MNX                                         CHAR(1)
 SALDO_MNX                                          NUMBER(14,2)
 CVE_PROD                                           CHAR(3)
 NUMERO_CUENTA_BCO_MNX                              CHAR(20)
 PLAZA_BCO_MNX                                      CHAR(5)
 CTAT_PROPIETARIO                                   CHAR(1)
 CTAT_TIPO_TRANS                                    CHAR(5)

tabla= DETALLE_INVERSION
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 CVE_INV                                            NUMBER
 NEMO_MNX                                           CHAR(10)
 MONTO_INV                                          NUMBER(14,2)
 INSTRUC_INV                                        CHAR(100)
 NEMOFP                                             CHAR(3)
 CVE_MOV                                            NUMBER(8)
 NUM_DET                                            NUMBER(2)
 TIPO_DET                                           CHAR(1)
 CTL_MOV                                            NUMBER(8)

tabla= DETALLE_ORDEN_MENSAJERIA
 Name                                      Null?    Type
```

```
 ------------------------------ -------- ----
 NUM_ORD                                 NUMBER(6)
 CVE_OPE                                 NUMBER(6)
 CVEDET_CV                               NUMBER(3)
 NEMO_PRM                                CHAR(10)
 NEMO_RUT                                CHAR(5)
 DIR_CLI                                 CHAR(30)
 NEMO_CLI                                CHAR(10)
 TIPOMOV_CV                              CHAR(1)
 STATUS_DOM                              CHAR(1)
 NAVISO_DOM                              NUMBER(1)
 FECHA_VALOR                             DATE

tabla= DIVISAS
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_DIV                                 CHAR(3)
 NOM_DIV                                 CHAR(20)

tabla= DOTACION_CHEQUES
 Name                           Null?    Type
 ------------------------------ -------- ----
 NUMDOT_DCH                              NUMBER(3)
 CHEINI_DCH                              NUMBER(10)
 NUMCHE_DCH                              NUMBER(5)
 ULTCHE_DCH                              NUMBER(10)
 NEMO_MNX                                CHAR(10)

tabla= ESTADO_DE_CUENTA
 Name                           Null?    Type
 ------------------------------ -------- ----
 EDOC_NUMERO                    NOT NULL NUMBER(8)
 EDOC_LOTE                      NOT NULL NUMBER(8)
 EDOC_SALDO_INICIAL                      NUMBER(18,2)
 EDOC_SALDO_FINAL                        NUMBER(18,2)
 EDOC_NEMO_CTA                           CHAR(10)
 EDOC_FECHA_RECEP                        DATE

tabla= FACTURA
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_OPE                                 NUMBER(6)
 NOMBRE                                  CHAR(40)
 DIRECCION                               CHAR(30)
 COLONIA                                 CHAR(18)

tabla= FORMA_COBRO
 Name                           Null?    Type
 ------------------------------ -------- ----
 NEMO                                    CHAR(3)
 DESCRIPCION                             CHAR(40)
 CVE_PROD                                CHAR(3)
 NIVEL                                   NUMBER(2)
```

```
 RECIBO                                    CHAR(2)
 COBRO_TESORERO                            CHAR(2)

tabla= FORMA_PAGO
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 NEMO                                      CHAR(3)
 DESCRIPCION                               CHAR(40)
 CVE_PROD                                  CHAR(3)
 NIVEL                                     NUMBER(2)
 ACCION                                    CHAR(2)
 RECIBO                                    CHAR(2)

tabla= INVERSIONES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 CVE_INV                                   NUMBER
 NEMO_MNX_O                                CHAR(10)
 NEMO_MNX_D                                CHAR(10)
 NEMO_MNX_INV                              CHAR(10)
 FECH1_INV                                 DATE
 FECH2_INV                                 DATE
 MONTO_INV                                 NUMBER(14,2)
 TR_INV                                    NUMBER(5,2)
 INSTRUC_INV                               CHAR(40)
 STAT_INV                                  CHAR(1)
 CVE_MOV                                   CHAR(8)
 UTI_INV                                   NUMBER(14,2)
 CVE_DIV                                   CHAR(3)

tabla= MOVIMIENTOS_ESTADO_DE_CUENTA
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 MOVI_EDOC_NUMERO                          NOT NULL NUMBER(8)
 MOVI_NUMERO                               NOT NULL NUMBER(4)
 MOVI_FECHA_VALOR                          NOT NULL DATE
 MOVI_IMPORTE                              NOT NULL NUMBER(18,2)
 MOVI_OBSERVACIONES                                 CHAR(50)
 MOVI_REFERENCIA                                    CHAR(20)
 MOVI_NEMO_CTA                                      CHAR(10)
 MOVI_TIPO_MOV                                      CHAR(1)
 MOVI_FECHA_CONC                                    DATE
 MOVI_PARC_NUMERO                                   NUMBER(8)

tabla= MOVS_CONTABLES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 CVE_MOV                                   NOT NULL CHAR(8)
 NUM_CTA_C                                 NOT NULL CHAR(16)
 NUM_CTA_A                                 NOT NULL CHAR(16)
 FECHA_MOV                                          DATE
 FECHVAL_MOV                               NOT NULL DATE
 MONTO_MOV                                 NOT NULL NUMBER(16,2)
```

```
 CVE_DIV                          NOT NULL CHAR(3)
 CVE_PROD                                  CHAR(3)
 REF_MOV                                   CHAR(10)
 CVEANT_MOV                                CHAR(8)
 NUMPOL_OP                                 NUMBER(7)
 CONCEPTO_MOV                              CHAR(30)
 CVEID_PRM                                 CHAR(10)
 CONCIC_MOV                                DATE
 CONCIA_MOV                                DATE
 PARC_NUMERO_C                             NUMBER(8)
 PARC_NUMERO_A                             NUMBER(8)

tabla= MOV_ADICIONALES
 Name                             Null?    Type
 -------------------------------- -------- ----
 CVE_MOV                                   CHAR(8)
 FECHA                                     DATE
 OBSERVACION                               CHAR(40)

tabla= NUMERO_AUTORIZACION
 Name                             Null?    Type
 -------------------------------- -------- ----
 LUGAR_AUT                                 NUMBER(6)
 NUMERO_AUT                                NUMBER(6)
 INDI_AUT                                  CHAR(1)

tabla= OPERACION
 Name                             Null?    Type
 -------------------------------- -------- ----
 CVE_OPE                          NOT NULL NUMBER(6)
 NEMO_CLI                         NOT NULL CHAR(10)
 NEMO_PRM                         NOT NULL CHAR(10)
 FECHA_OPE                        NOT NULL DATE
 HORA_OPE                                  CHAR(5)
 CVE_AUT                                   NUMBER(6)
 CVE_MOV                                   CHAR(10)
 REF_OPE                                   CHAR(30)
 INDI_OPE                         NOT NULL CHAR(1)
 CVE_DIR_E                                 NUMBER(2)
 CVE_DIR_R                                 NUMBER(2)
 COMISION_DP                               NUMBER(12)
 BANMOD_OPE                                CHAR(2)
 COMISION_OPE                              NUMBER(12)
 UTIMI_OPE                                 NUMBER(12)
 IMPCOM_OPE                                NUMBER(1)
 UTILIDAD_OPE                     NOT NULL NUMBER(12)
 CVEID_OPE                                 CHAR(10)
 FACTURA                                   CHAR(2)
 CAUSA                                     CHAR(10)
 CVE_TIPO                                  CHAR(1)
 N_FUTURO                                  NUMBER(5)

tabla= ORDEN_MENSAJERIA
```

Name	Null?	Type
NUM_ORD		NUMBER(6)
NEMO_PRM		CHAR(10)
NEMO_RUT		CHAR(5)
FECH_RUT		DATE
HORA_RUT		CHAR(5)
BANCAN_ORD		CHAR(1)
STATUS		CHAR(1)

tabla= PARTIDAS_CONC

Name	Null?	Type
PARC_NUMERO	NOT NULL	NUMBER(8)
PARC_FECHA		DATE
PARC_NUM_MOVS_B		NUMBER(5)
PARC_NUM_MOVS_E		NUMBER(5)
PARC_LOTE_CONC	NOT NULL	NUMBER(8)

tabla= PIZARRON_COTIZACIONES

Name	Null?	Type
CVE_DIV		CHAR(3)
CVE_PROD		CHAR(3)
FECHA_PIZ		DATE
HORA_PIZ		CHAR(9)
CVE_INS		CHAR(10)
TCCMN1_PIZ		NUMBER(12,6)
TCVMN1_PIZ		NUMBER(12,6)
TCCMN2_PIZ		NUMBER(12,6)
TCVMN2_PIZ		NUMBER(12,6)
TCCMN3_PIZ		NUMBER(12,6)
TCVMN3_PIZ		NUMBER(12,6)
INDI_PIZ		CHAR(1)

tabla= POLIZA

Name	Null?	Type
CVE_POL		NUMBER(8)
TIPO_POL		CHAR(1)
FECHA_POL		DATE
CLASE_POL		CHAR(1)
DIARIO_POL		NUMBER(3)
CONCEPTO_POL		CHAR(30)
FECHAFIN_POL		DATE
NUM_CTA		CHAR(16)
CVE_DIV		CHAR(3)
TIPO		CHAR(1)

tabla= PROMOTOR

Name	Null?	Type

```
 NEMO_PRM                                  CHAR(10)
 NOM_PRM                                   CHAR(30)
 DIR_PRM                                   CHAR(30)
 COL_PRM                                   CHAR(18)
 CIU_PRM                                   CHAR(12)
 PAIS_PRM                                  CHAR(12)
 CP_PRM                                    CHAR(5)
 TEL1_PRM                                  CHAR(12)
 TEL2_PRM                                  CHAR(12)
 CVEID_PRM                                 CHAR(10)
 CVEEXPE_PRM                               NUMBER(6)
 NEMO_PTO                                  CHAR(3)
 CVE_OFI                                   CHAR(3)
 RFC_PRM                                   CHAR(12)
 RECIBO                                    CHAR(2)

tabla= PUESTOS
 Name                           Null?     Type
 ------------------------------ --------- ----
 NEMO_PTO                                  CHAR(3)
 DESC_PTO                                  CHAR(15)
 ARCH_PTO                                  CHAR(15)

tabla= RECALCULA_SALDO
 Name                           Null?     Type
 ------------------------------ --------- ----
 FECHA                                     DATE
 REFERENCIA                                CHAR(40)

tabla= SALDO
 Name                           Null?     Type
 ------------------------------ --------- ----
 NUM_CTA                                   CHAR(16)
 CVE_DIV                                   CHAR(3)
 SALDO                                     NUMBER(18,2)

tabla= SALDOS_INICIALES
 Name                           Null?     Type
 ------------------------------ --------- ----
 FECHAI_SI                                 DATE
 NUM_CTA                                   CHAR(16)
 CVE_DIV                                   CHAR(3)
 SALDOI_SI                                 NUMBER(18,2)
 SALDOULT_SI                               NUMBER(18,2)

tabla= TASA_INTERES
 Name                           Null?     Type
 ------------------------------ --------- ----
 CVE_DIV                                   CHAR(3)
 INT_TASA                                  NUMBER(8,4)

tabla= TIPOS_CLIENTE
 Name                           Null?     Type
```

```
 ------------------------------ -------- ----
 CVE_TIPO                                CHAR(1)
 DESC_TIPO                               CHAR(12)

tabla= TIPOS_MOVIMIENTO
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_PROD                                CHAR(3)
 TIPOMOV_CV                              CHAR(1)
 CAR_DEA_MOV                             CHAR(16)
 ABO_DEA_MOV                             CHAR(16)
 CAR_ASG_MOV                             CHAR(16)
 ABO_ASG_MOV                             CHAR(16)
 CAR_PUE_MOV                             CHAR(16)
 ABO_PUE_MOV                             CHAR(16)
 CAR_CON_MOV                             CHAR(16)
 ABO_CON_MOV                             CHAR(16)

tabla= TIPO_TRANSFERENCIA_CUENTAS
 Name                           Null?    Type
 ------------------------------ -------- ----
 TIPT_CLAVE                     NOT NULL CHAR(5)
 TIPT_DESCRIPCION               NOT NULL CHAR(30)

tabla= TRANSFERENCIAS
 Name                           Null?    Type
 ------------------------------ -------- ----
 CVE_TRF                                 NUMBER
 FECHA_TRF                               DATE
 NEMO_MNX_O                              CHAR(10)
 NEMO_MNX_D                              CHAR(10)
 IMPORTE_TRF                             NUMBER(14,2)
 CVE_MOV                                 CHAR(8)
 FECHVAL_TRF                             DATE
 INSTRUC1_TRF                            CHAR(60)
 INSTRUC2_TRF                            CHAR(60)
 FECHIMP_TRF                             DATE
 INDI_TRF                                CHAR(1)
 CVE_OPE                                 NUMBER(6)
 CVEDET_CV                               NUMBER(4)
 FECHAUT_AFC                             DATE
 HORAUT_TCV                              CHAR(5)
 INSTRUC3_TRF                            CHAR(60)
 FECHA_TRANS                             DATE
 STATUS_TRANS                            CHAR(1)
 TRAN_CUENTA_DEST                        CHAR(25)
 TRAN_LOTE_TRANS                         NUMBER(8)
 TRAN_FECHA_CONC                         DATE
 CUEN_NEMO_CTA_D                         CHAR(10)

tabla= TRASPASOS
 Name                           Null?    Type
 ------------------------------ -------- ----
```

```
CVE_MOV                        NOT NULL NUMBER(8)
TIPO_TRAS                      NOT NULL CHAR(1)
CVE_DIV                        NOT NULL CHAR(3)
CVE_PROD                                CHAR(3)
FECHVAL_TRAS                   NOT NULL DATE
MONTO_TRAS                     NOT NULL NUMBER(16,2)
TCMN_TRAS                               NUMBER(12,6)
MONTOMN_TRAS                            NUMBER(16,2)
NEMOFP                                  CHAR(3)
NEMO_CTA                                CHAR(10)
```

APENDICE 2

REPORTE CDRF.LIS

```
SQL> spo off
```

APENDICE 2.
Reporte CDRF.lis
Aplicación de la Ingeniería Reversiva en el sistema Score.

```
27-AUG-94                        Retrofit Report                    Page: 1

OBJECT (object type)                      Reason not retrofitted

SYSTEM (tablespace)
CASE (tablespace)
CHEQUE_FOLIO (sequence)
CONTABLE_FOLIO (sequence)
ESTADO_CUENTA (sequence)                  Sequence already exists
LOTE (sequence)                           Sequence already exists
LOTE_CONCIL (sequence)                    Sequence already exists
MENSAJERIA_FOLIO (sequence)
NUM_CLIENTE (sequence)
PARTIDA (sequence)                        Sequence already exists
TRF_FOLIO (sequence)
AA (table)
    CAMPO1 (column)
    CAMPO2 (column)
AGENDA (table)
    NEMO_PRM (column)
    NEMO_CLI (column)
    ACCION (column)
    ACUERDO (column)
    HORA (column)
    CVE_RES (column)
    FECHA (column)
    AGE_FECH_HOR (index)
        Col no. 1 (indcol)
        Col no. 2 (indcol)
    AGE_NEMO_PRM (index)
        Col no. 1 (indcol)
AUTORIZACION (table)
    CVE_AUT (column)
    REF_AUT (column)
    CAUSA_AUT (column)
    INDI_AUT (column)
```

OBJECT (object type)	Reason not retrofitted
LUGAR (column)	
FECHA_AUT (column)	
AUTORIZA_CHEQUES (table)	
NEMO_CTA (column)	
CVE_ACH (column)	
MOTIVO_ACH (column)	
CVEID_PRM (column)	
FECH_ACH (column)	
FOLIO_CHE (column)	
AUTORIZA_FACTURA (table)	
MOTIVO_AFC (column)	
FECH_AFC (column)	
CVE_AFC (column)	
CVE_FAC (column)	
BB (table)	
CAMPO1 (column)	
CAMPO2 (column)	
BITACORA_LOTE_CONCILIACIONES (table)	Table already exists
BITC_HORA (column)	Column already exists
BITC_USUARIO (column)	Column already exists
BITC_TIPO_CONC (column)	Column already exists
BITC_FECHA (column)	Column already exists
BITC_LOTE (column)	Column already exists
BITC_NUM_PARTIDAS (column)	Column already exists
BITC_PK (constraint)	Constraint already exists,
I (key) exists for this column	This Key/Constraint already
BITACORA_LOTE_TRANSMISIONES (table)	Table already exists
BITA_CLAVE_SISTEMA (column)	Column already exists
BITA_TIPO_TRANSMISION (column)	Column already exists
BITA_USUARIO (column)	Column already exists
BITA_REFERENCIA (column)	Column already exists
BITA_HORA (column)	Column already exists
BITA_FECHA (column)	Column already exists
BITA_LOTE (column)	Column already exists
BITA_NUM_MOVS (column)	

OBJECT (object type) Reason not retrofitted

CALENDARIO (table)
    DIA_CAL (column)
    INDI_CAL (column)
    FECHA_CAL (column)
    CAL_FECH (index)
        Col no. 1 (indcol)
CALENDARIO_USA (table)
    FECHA (column)
    DIAS (column)
CAMPOS (table)
    TABLA (column)
    CAMPO (column)
    TIPO (column)
    DESCRIPCION (column)
    LIGA (column)
CC (table)
    CAMPO1 (column)
    CAMPO2 (column)
CG_FORM_HELP (table)
    HLP_APPLN (column)
    HLP_INDEX (column)
    HLP_MODTAB_NAME (column)
    HLP_GENERATED (column)
    HLP_TEXT (column)
    HLP_TYPE (column)
    HLP_SEQ (column)
    X_CG_FORM_HELP_1 (index)
        Col no. 1 (indcol)
        Col no. 2 (indcol)

OBJECT (object type)	Reason not retrofitted
Col no. 3 (indcol)	
Col no. 4 (indcol)	
CG_REF_CODES (table)	
RV_LOW_VALUE (column)	
RV_HIGH_VALUE (column)	
RV_ABBREVIATION (column)	
RV_DOMAIN (column)	
RV_MEANING (column)	
RV_TYPE (column)	
X_CG_REF_CODES_1 (index)	
Col no. 1 (indcol)	
Col no. 2 (indcol)	
CHEQUES (table)	Table already exists
NUM_CHE (column)	Column already exists
BAN_CHE (column)	Column already exists
PORTADOR_CHE (column)	Column already exists
CVE_MOV (column)	Column already exists
NEMO_MNX (column)	Column already exists
FECH_CHE (column)	Column already exists
FECONCI_CHE (column)	Column already exists
FECHA_CON (column)	Column already exists
MONTO_CHE (column)	Column already exists
FOLIO_CHE (column)	Column already exists
CVE_OPE (column)	Column already exists
CVEDET_CV (column)	Column already exists
DIAS (column)	Column already exists
CHE_CVE_MOV (index)	
Col no. 1 (indcol)	
CHE_CVE_OPE (index)	
Col no. 1 (indcol)	
CHE_FEC_CON (index)	
Col no. 1 (indcol)	

OBJECT (object type) Reason not retrofitted

CHE_FOL_CHE (index)
Col no. 1 (indcol)
CHE_NEMO_CHE (index)
Col no. 1 (indcol)
Col no. 2 (indcol)
CHE_NUM_CHE (index)
Col no. 1 (indcol)
CLASES_CLIENTES (table)
NEMO_CLS (column)
NOMBRE_CLS (column)
NUM_CLS (column)
MONMAX_CLS (column)
CLIENTES (table)
NEMO_CLI (column)
NOM_CLI (column)
TEL1_CLI (column)
TEL2_CLI (column)
SENSIBI_CLI (column)
NOMOPE_CLI (column)
NEMO_PRM (column)
CVE_TIPO (column)
RFC (column)
FECMOD_DP (column)
CVE_CLI (column)
LIMCRE_CLI (column)
MONTOMAX_CLI (column)
NUM_CLS (column)
CLI_NEMO (index)
Col no. 1 (indcol)
CLIENTES_DIR (table)

OBJECT (object type)	Reason not retrofitted
NEMO_CLI (column)	
DIR_CLI (column)	
COL_CLI (column)	
CIU_CLI (column)	
PAIS_CLI (column)	
CP_CLI (column)	
NEMO_RUT (column)	
CVE_DIR (column)	
CLI_DIR_NEMO (index)	
Col no. 1 (indcol)	
COMISIONES (table)	
CVE_COM (column)	
MONINF_COM (column)	
PTGE_COM (column)	
COMPRA_VENTA (table)	Table already exists
TIPOMOV_CV (column)	Column already exists
CVE_DIV (column)	Column already exists
CVE_PROD (column)	Column already exists
NEMOFP (column)	Column already exists
DETALLEFP_CV (column)	Column already exists
ENTREGA_CV (column)	Column already exists
FUTURO_CV (column)	Column already exists
CVE_MOV (column)	Column already exists
CVE_MOV1 (column)	Column already exists
CVEID_CV (column)	Column already exists
NEMO_CTA (column)	Column already exists
CONFIRMA_CV (column)	Column already exists
INDI_CV (column)	Column already exists
REF_CV (column)	Column already exists
CVE_PROD2 (column)	Column already exists
FECHVAL_CV (column)	Column already exists
LIQUIDADO_CV (column)	Column already exists
FECHASG_CV (column)	Column already exists
FECHPAGO_CV (column)	Column already exists
CVE_OPE (column)	Column already exists
MONTO_CV (column)	Column already exists
TCMN_CV (column)	Column already exists
MONTOMN_CV (column)	Column already exists
TCUSD_CV (column)	Column already exists
COSTOMN_CV (column)	Column already exists
CVEDET_CV (column)	Column already exists

OBJECT (object type) Reason not retrofitted

```
    C_V_CVE_OPE (index)
        Col no. 1 (indcol)
    C_V_LLAVE (index)
        Col no. 1 (indcol)
        Col no. 2 (indcol)
COMPRA_VENTA_CANCELADOS (table)
    TIPOMOV_CV (column)
    CVE_DIV (column)
    CVE_PROD (column)
    NEMOFP (column)
    DETALLEFP_CV (column)
    ENTREGA_CV (column)
    FUTURO_CV (column)
    CVE_MOV (column)
    CVE_MOV1 (column)
    CVEID_CV (column)
    CVEDET_CV (column)
    NEMO_CTA (column)
    CONFIRMA_CV (column)
    INDI_CV (column)
    FECHVAL_CV (column)
    LIQUIDADO_CV (column)
    FECHASG_CV (column)
    FECHPAGO_CV (column)
    CVE_OPE (column)
    MONTO_CV (column)
    TCMN_CV (column)
    MONTOMN_CV (column)
    TCUSD_CV (column)
    COSTOMN_CV (column)
CONCILIACION_BANCOS_MNX (table)
    CVE_MOV (column)
    FECH_CBM (column)
CONTADOR (table)
    FECHA (column)
    NUMERO (column)
    CAMPO (column)
```

OBJECT (object type)	Reason not retrofitted
CONTROL_SCORE (table)	
DIVISA (column)	
EMPRESA (column)	
PRODUCTO (column)	
NOMBRE (column)	
RFC (column)	
DIVISA_PAIS (column)	
PRODUCTO_INTERES (column)	
NUM_CTA_INTERES_PAIS (column)	
NUM_CTA_INTERES_NOPAIS (column)	
PRODUCTO_COMISION (column)	
CONTROL_SCORE1 (table)	
DIVISA (column)	
EMPRESA (column)	
PRODUCTO (column)	
NOMBRE (column)	
RFC (column)	
DIVISA_PAIS (column)	
PRODUCTO_INTERES (column)	
NUM_CTA_POR_PAGAR (column)	
NUM_CTA_POR_COBRAR (column)	
NUM_CTA_IMTERES_PAIS (column)	
NUM_CTA_INTERES_NOPAIS (column)	
PRODUCTO_COMISION (column)	
CUENTAS_BCO_CLIENTE (table)	
CTA_BAN_NEMO (index)	
Col no. 1 (indcol)	
NEMO_CLI (column)	
CVE_DIV (column)	
CVE_PROD (column)	
DESTINO_BCO (column)	
NEMO_BCO (column)	
CUENTAS_CONTABLES (table)	Table already exists
FECHA_CTA (column)	Column already exists
SALDOI_CTA (column)	Column already exists
CTA_CON_NEMO (index)	

OBJECT (object type)	Reason not retrofitted
Col no. 1 (indcol)	
CTA_CON_NUM (index)	
Col no. 1 (indcol)	
NUM_CTA (column)	Column already exists
DESC_CTA (column)	Column already exists
NIV_CTA (column)	Column already exists
TIPO_CTA (column)	Column already exists
NEMO_CTA (column)	Column already exists
NAT_CTA (column)	Column already exists
CUENTA_TRANS_DEST (table)	Table already exists
CUEN_NEMO_CTA (column)	Column already exists
CUEN_NUMERO_CUENTA (column)	Column already exists
CUEN_INDICA_PROP (column)	Column already exists
CUEN_PLAZA_BCO (column)	Column already exists
CUEN_TIPT_CLAVE (column)	Column already exists
DEPURACION (table)	
FECHA (column)	
FECHA_DEP (column)	
NOMBRE (column)	
DESTINOS_COBRO_MONEX (table)	Table already exists
SALDO_MNX (column)	Column already exists
CVE_DIV (column)	Column already exists
DESTINO_MNX (column)	Column already exists
NEMO_CTA (column)	Column already exists
CVE_TD (column)	Column already exists
NEMO_MNX (column)	Column already exists
NUMFAX_MNX (column)	Column already exists
BANCHE_MNX (column)	Column already exists
BANTRF_MNX (column)	Column already exists
BANINV_MNX (column)	Column already exists
BANCAJ_MNX (column)	Column already exists
CVE_PROD (column)	Column already exists
NUMERO_CUENTA_BCO_MNX (column)	Column already exists
PLAZA_BCO_MNX (column)	Column already exists
CTAT_PROPIETARIO (column)	Column already exists
CTAT_TIPO_TRANS (column)	Column already exists

OBJECT (object type) Reason not retrofitted

DETALLE_INVERSION (table)
    CVE_INV (column)
    MONTO_INV (column)
    CVE_MOV (column)
    NUM_DET (column)
    CTL_PAGO (column)
    CTL_MOV (column)
    DET_INV1 (index)
        Col no. 1 (indcol)
    DET_INV2 (index)
        Col no. 1 (indcol)
    NEMO_MNX (column)
    INSTRUC_INV (column)
    NEMOFP (column)
    TIPO_DET (column)
DETALLE_ORDEN_MENSAJERIA (table)
    FECHA_VALOR (column)
    NUM_ORD (column)
    CVE_OPE (column)
    CVEDET_CV (column)
    NAVISO_DOM (column)
    DET_ORD_FEC (index)
        Col no. 1 (indcol)
    DET_ORD_NUM (index)
        Col no. 1 (indcol)
    DET_ORD_OPE (index)
        Col no. 1 (indcol)
    NEMO_PRM (column)
    NEMO_RUT (column)
    DIR_CLI (column)

OBJECT (object type)	Reason not retrofitted
NEMO_CLI (column)	
TIPOMOV_CV (column)	
STATUS_DOM (column)	
DIVISAS (table)	Table already exists
CVE_DIV (column)	Column already exists
NOM_DIV (column)	Column already exists
DOTACION_CHEQUES (table)	
NUMDOT_DCH (column)	
CHEINI_DCH (column)	
NUMCHE_DCH (column)	
ULTCHE_DCH (column)	
NEMO_MNX (column)	
ENLACE (table)	
CVE_DIV (column)	
NUM_CTA (column)	
NUM_CTA_NVA (column)	
ERR_TRANS (table)	Table already exists
MENS (column)	Column already exists
ESTADO_DE_CUENTA (table)	Table already exists
EDOC_FECHA_RECEP (column)	Column already exists
EDOC_NUMERO (column)	Column already exists
EDOC_LOTE (column)	Column already exists
EDOC_SALDO_INICIAL (column)	Column already exists
EDOC_SALDO_FINAL (column)	Column already exists
EDOC_NEMO_CTA (column)	Column already exists
EXPEDIENTE (table)	
MES (column)	
DET_COM (column)	
DET_VEN (column)	
VOL_COM (column)	
VOL_VEN (column)	
UTI_COM (column)	

OBJECT (object type) Reason not retrofitted

UTI_VEN (column)
UTI_OPE (column)
UTI_PON (column)
COBRO_CHEQUE (column)
AUTORIZACION (column)
SIN_UTILIDAD (column)
MODIFICACION (column)
NO_REALIZADA (column)
TOTAL_MN_COM (column)
TOTAL_MN_VEN (column)
MONTO_DIV_CHE (column)
MONTO_DIV (column)
EXP_NEMO_CLI (index)
Col no. 1 (indcol)
NEMO_CLI (column)
CVE_DIV (column)
FACTURA (table)
CVE_OPE (column)
FAC_CVE_OPE (index)
Col no. 1 (indcol)
NOMBRE (column)
DIRECCION (column)
COLONIA (column)
FECHAS (table)
FECHA_1 (column)
FECHA_2 (column)
FORMA_COBRO (table)
NIVEL (column)
NEMO (column)
DESCRIPCION (column)
CVE_PROD (column)
RECIBO (column)
COBRO_TESORERO (column)
FORMA_PAGO (table)

OBJECT (object type) Reason not retrofitted

NIVEL (column)
NEMO (column)
DESCRIPCION (column)
CVE_PROD (column)
ACCION (column)
RECIBO (column)
INVERSIONES (table)
FECH1_INV (column)
FECH2_INV (column)
FECHA_CONF_INV (column)
CVE_INV (column)
MONTO_INV (column)
TR_INV (column)
UTI_INV (column)
INV_FECHA2 (index)
Col no. 1 (indcol)
NEMO_MNX_O (column)
NEMO_MNX_D (column)
NEMO_MNX_INV (column)
INSTRUC_INV (column)
STAT_INV (column)
CVE_MOV (column)
CVE_DIV (column)
MENU (table)
OPCION (column)
MENU (column)
FUNCION (column)
PROGRAMA (column)
MENU1 (table)
OPCION (column)
MENU (column)
FUNCION (column)
PROGRAMA (column)
MORTALIDAD (table)

OBJECT (object type)	Reason not retrofitted
DIA (column)	
CANTIDAD (column)	
PROB (column)	
COBRADOS (column)	
MOVIMIENTOS_ESTADO_DE_CUENTA (table)	Table already exists
MOVI_FECHA_VALOR (column)	Column already exists
MOVI_FECHA_CONC (column)	Column already exists
MOVI_PARC_NUMERO (column)	Column already exists
MOVI_EDOC_NUMERO (column)	Column already exists
MOVI_NUMERO (column)	Column already exists
MOVI_IMPORTE (column)	Column already exists
MOVI_OBSERVACIONES (column)	Column already exists
MOVI_REFERENCIA (column)	Column already exists
MOVI_NEMO_CTA (column)	Column already exists
MOVI_TIPO_MOV (column)	Column already exists
MOVS1 (table)	
FECHA_MOV (column)	
FECHVAL_MOV (column)	
CONCIC_MOV (column)	
CONCIA_MOV (column)	
MONTO_MOV (column)	
NUMPOL_OP (column)	
CVE_MOV (column)	
NUM_CTA_C (column)	
NUM_CTA_A (column)	
CVE_DIV (column)	
CVE_PROD (column)	
REF_MOV (column)	
CVEANT_MOV (column)	
CONCEPTO_MOV (column)	
CVEID_PRM (column)	
MOVS_1 (table)	
MOVI_FECHA_VALOR (column)	
MOVI_EDOC_NUMERO (column)	
MOVI_NUMERO (column)	
MOVI_IMPORTE (column)	
MOVI_OBSERVACIONES (column)	
MOVI_REFERENCIA (column)	
MOVI_NEMO_CTA (column)	
MOVI_TIPO_MOV (column)	
MOVS_CONTABLES (table)	Table already exists

OBJECT (object type)	Reason not retrofitted
FECHA_MOV (column)	Column already exists
FECHVAL_MOV (column)	Column already exists
CONCIC_MOV (column)	Column already exists
CONCIA_MOV (column)	Column already exists
MONTO_MOV (column)	Column already exists
NUMPOL_OP (column)	Column already exists
PARC_NUMERO_C (column)	Column already exists
PARC_NUMERO_A (column)	Column already exists
CVE_MOV (column)	Column already exists
NUM_CTA_C (column)	Column already exists
NUM_CTA_A (column)	Column already exists
CVE_DIV (column)	Column already exists
CVE_PROD (column)	Column already exists
REF_MOV (column)	Column already exists
CVEANT_MOV (column)	Column already exists
CONCEPTO_MOV (column)	Column already exists
CVEID_PRM (column)	Column already exists
MOV_ADICIONALES (table)	
FECHA (column)	
MOV_ADI_CVE (index)	
Col no. 1 (indcol)	
CVE_MOV (column)	
OBSERVACION (column)	
TRASPASO (column)	
M_REVISION (table)	
FECHA (column)	
NIVEL_RIESGO (table)	
NUM_CLS (column)	
NIVEL_REC (column)	
NIVEL_ENT (column)	
NOTAS (table)	
NUMERO (column)	
NOTA (column)	
DATOS (column)	

OBJECT (object type)	Reason not retrofitted
NUMERO_AUTORIZACION (table)	
LUGAR_AUT (column)	
NUMERO_AUT (column)	
INDI_AUT (column)	
OPERACION (table)	Table already exists
FECHA_OPE (column)	Column already exists
CVE_OPE (column)	Column already exists
CVE_AUT (column)	Column already exists
CVE_DIR_E (column)	Column already exists
CVE_DIR_R (column)	Column already exists
COMISION_DP (column)	Column already exists
COMISION_OPE (column)	Column already exists
UTIMI_OPE (column)	Column already exists
IMPCON_OPE (column)	Column already exists
UTILIDAD_OPE (column)	Column already exists
N_FUTURO (column)	Column already exists
OPE_CVE_OPE (index)	
Col no. 1 (indcol)	
OPE_FEC (index)	
Col no. 1 (indcol)	
OPE_NEMO (index)	
Col no. 1 (indcol)	
NEMO_CLI (column)	Column already exists
NEMO_PRM (column)	Column already exists
HORA_OPE (column)	Column already exists
CVE_MOV (column)	Column already exists
REF_OPE (column)	Column already exists
INDI_OPE (column)	Column already exists
BANMOD_OPE (column)	Column already exists
CVEID_OPE (column)	Column already exists
FACTURA (column)	Column already exists
CAUSA (column)	Column already exists
CVE_TIPO (column)	Column already exists
ORDEN_MENSAJERIA (table)	

OBJECT (object type)	Reason not retrofitted
FECH_RUT (column)	
NUM_ORD (column)	
ORD_MEN_NUM (index)	
Col no. 1 (indcol)	
NEMO_PRM (column)	
NEMO_RUT (column)	
HORA_RUT (column)	
BANCAN_ORD (column)	
STATUS (column)	
PARAM_CONTROL (table)	
FECHA_CTL (column)	
FECHA2_CTL (column)	
RIESGOC_CTL (column)	
RIESGOL_CTL (column)	
MONTOMAXC_CTL (column)	
MONTOMAXL_CTL (column)	
LIMVTA_CTL (column)	
LIMCPA_CTL (column)	
UTILIMIN_CTL (column)	
MONTOMAXP_CTL (column)	
CVE_DIV (column)	
CVE_PROD (column)	
COMPUESTO (column)	
INDI_CTL (column)	
PARTIDAS_CONC (table)	Table already exists
PARC_FECHA (column)	Column already exists
PARC_NUMERO (column)	Column already exists
PARC_NUM_MOVS_B (column)	Column already exists
PARC_NUM_MOVS_E (column)	Column already exists
PARC_LOTE_CONC (column)	Column already exists
BITC_LOTE (constraint)	Constraint already exists,
1 (key)	This Key/Constraint already
exists for this column	
PASA (table)	
CAMPO1 (column)	
CAMPO2 (column)	
CAMPO3 (column)	
CAMPO4 (column)	
CAMPO5 (column)	
CAMPO6 (column)	

OBJECT (object type) Reason not retrofitted

PASA1 (table)
    CAMPO1 (column)
    CAMPO2 (column)
    CAMPO3 (column)
    CAMPO4 (column)
    CAMPO5 (column)
PASA2 (table)
    CAMPO1 (column)
    CAMPO2 (column)
    CAMPO3 (column)
    CAMPO4 (column)
    CAMPO5 (column)
    CAMPO6 (column)
PASO3 (table)
    DOS (column)
    UNO (column)
PIZARRON_COTIZACIONES (table)
    FECHA_PIZ (column)
    TCCMN1_PIZ (column)
    TCVMN1_PIZ (column)
    TCCMN2_PIZ (column)
    TCVMN2_PIZ (column)
    TCCMN3_PIZ (column)
    TCVMN3_PIZ (column)
    PIZ_COT_DIV (index)
        Col no. 1 (indcol)
    CVE_DIV (column)
    CVE_PROD (column)
    HORA_PIZ (column)
    CVE_INS (column)
    INDI_PIZ (column)
POLIZA (table)

OBJECT (object type)	Reason not retrofitted
FECHA_POL (column)	
FECHAFIN_POL (column)	
CVE_POL (column)	
DIARIO_POL (column)	
POL_FECHA (index)	
Col no. 1 (indcol)	
TIPO_POL (column)	
CLASE_POL (column)	
CONCEPTO_POL (column)	
NUM_CTA (column)	
CVE_DIV (column)	
TIPO (column)	
POSICION (table)	Table already exists
NUM_CTA_MN (column)	Column already exists
NUM_CTA_POR_COBRAR (column)	Column already exists
NUM_CTA_POR_PAGAR (column)	Column already exists
FECHAI_POS (column)	Column already exists
FECHA_CORTE_POS (column)	Column already exists
POSINIC1_POS (column)	Column already exists
TCPPI1_POS (column)	Column already exists
POSICION_CORTE (column)	Column already exists
POSINIC_POS (column)	Column already exists
POSHOY_POS (column)	Column already exists
POSFIN_POS (column)	Column already exists
CVE_DIV (column)	Column already exists
CVE_PROD (column)	Column already exists
NUM_CTA (column)	Column already exists
POSICION1 (table)	
CVE_DIV (column)	
CVE_PROD (column)	
NUM_CTA (column)	
FECHAI_POS (column)	
POSICION_CORTE (column)	
POSFIN_POS (column)	
POSINIC_POS (column)	
POSI24_POS (column)	
POSI48_POS (column)	
POSHOY_POS (column)	
POS24_POS (column)	
POS48_POS (column)	
PHOYREAL_POS (column)	
POSINIC1_POS (column)	
POSINIC2_POS (column)	
POSINIC3_POS (column)	

OBJECT (object type)	Reason not retrofitted
TCPPI1_POS (column)	
TCPPI2_POS (column)	
TCPPI3_POS (column)	
PRODUCTOS (table)	Table already exists
CVE_PROD (column)	Column already exists
DESC_PROD (column)	Column already exists
DESTINO_PROD (column)	Column already exists
PROMOTOR (table)	Table already exists
NEMO_PRM (column)	Column already exists
NOM_PRM (column)	Column already exists
DIR_PRM (column)	Column already exists
COL_PRM (column)	Column already exists
CIU_PRM (column)	Column already exists
PAIS_PRM (column)	Column already exists
CP_PRM (column)	Column already exists
TEL1_PRM (column)	Column already exists
TEL2_PRM (column)	Column already exists
CVEID_PRM (column)	Column already exists
NEMO_PTO (column)	Column already exists
CVE_OFI (column)	Column already exists
RFC_PRM (column)	Column already exists
RECIBO (column)	Column already exists
CVEEXPE_PRM (column)	Column already exists
PRM_NEM (index)	
Col no. 1 (indcol)	
PUESTOS (table)	Table already exists
NEMO_PTO (column)	Column already exists
DESC_PTO (column)	Column already exists
ARCH_PTO (column)	Column already exists
RECALCULA_SALDO (table)	
REFERENCIA (column)	
FECHA (column)	
RECEP_MENS (table)	Table already exists

OBJECT (object type)	Reason not retrofitted
LINEA (column)	Column already exists
N_LINEAS (column)	Column already exists
RESPALDO (table)	
RESP_RES (column)	
FECHA_RES (column)	
RESULTADOS_AGENDA (table)	
CVE_RES (column)	
DESC_RES (column)	
RUTAS (table)	
NEMO_RUT (column)	
DESC_RUT (column)	
SALDO (table)	
NUM_CTA (column)	
CVE_DIV (column)	
SALDO (column)	
SALDOS_INICIALES (table)	Table already exists
NUM_CTA (column)	Column already exists
CVE_DIV (column)	Column already exists
FECHAI_SI (column)	Column already exists
SALDOI_SI (column)	Column already exists
SALDOULT_SI (column)	Column already exists
SAL_INI_CTA (index)	
Col no. 1 (indcol)	
Col no. 2 (indcol)	
SEGURIDAD (table)	
NEMO_TIPO (column)	
MENU (column)	

OBJECT (object type) Reason not retrofitted

OPCION (column)
SMTIOD (table)
EMP (column)
DIA (column)
FE_UNO (column)
FE_DOS (column)
DIFERENCIA (column)
MOV_MIN (column)
MOV_MAX (column)
COMISION (column)
INTERES (column)
SMT_DIA (index)
Col no. 1 (indcol)
TABLAS (table)
TABLA (column)
DESCRIPCION (column)
SUBSISTEMA (column)
MODO_USO (column)
TASA_INTERES (table)
CVE_DIV (column)
INT_TASA (column)
TIPOS_CLIENTE (table)
CVE_TIPO (column)
DESC_TIPO (column)
TIPOS_MOVIMIENTO (table)
CVE_PROD (column)
TIPOMOV_CV (column)
CAR_DEA_MOV (column)
ABO_DEA_MOV (column)
CAR_ASG_MOV (column)
ABO_ASG_MOV (column)
CAR_PUE_MOV (column)

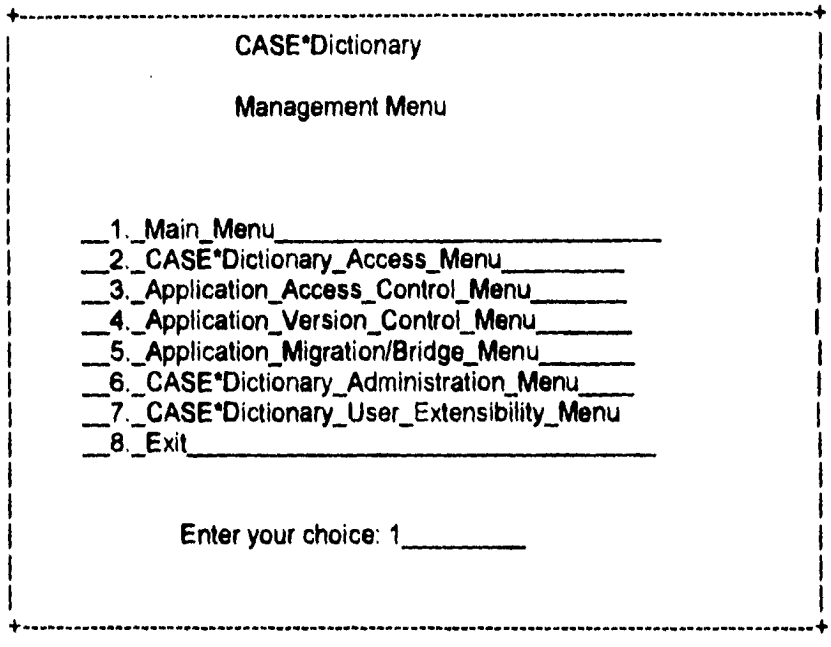
OBJECT (object type)	Reason not retrofitted
ABO_PUE_MOV (column)	
CAR_CON_MOV (column)	
ABO_CON_MOV (column)	
TIPO_TRANSFERENCIA_CUENTAS (table)	Table already exists
TIPT_CLAVE (column)	Column already exists
TIPT_DESCRIPCION (column)	Column already exists
TRANSFERENCIAS (table)	Table already exists
TRAN_CUENTA_DEST (column)	
CUEN_NEMO_CTA_D (column)	Column already exists
NEMO_MNX_O (column)	Column already exists
NEMO_MNX_D (column)	Column already exists
CVE_MOV (column)	Column already exists
INSTRUC1_TRF (column)	Column already exists
INSTRUC2_TRF (column)	Column already exists
INDI_TRF (column)	Column already exists
HORAUT_TCV (column)	Column already exists
INSTRUC3_TRF (column)	Column already exists
STATUS_TRANS (column)	Column already exists
TRAN_FECHA_CONC (column)	Column already exists
FECHA_TRF (column)	Column already exists
FECHVAL_TRF (column)	Column already exists
FECHIMP_TRF (column)	Column already exists
FECHAUT_AFC (column)	Column already exists
FECHA_TRANS (column)	Column already exists
TRAN_LOTE_TRANS (column)	Column already exists
CVE_TRF (column)	Column already exists
IMPORTE_TRF (column)	Column already exists
CVE_OPE (column)	Column already exists
CVEDET_CV (column)	Column already exists
TRF_FEC (index)	
Col no. 1 (indcol)	
TRF_FOL (index)	
Col no. 1 (indcol)	
TRF_NEMO_TRF (index)	
Col no. 1 (indcol)	

OBJECT (object type)	Reason not retrofitted
TRF_OPE (index)	
Col no. 1 (indcol)	
TRASPASOS (table)	
TIPO_TRAS (column)	
CVE_DIV (column)	
CVE_PROD (column)	
NEMOFP (column)	
NEMO_CTA (column)	
FECHVAL_TRAS (column)	
CVE_MOV (column)	
MONTO_TRAS (column)	
TCMN_TRAS (column)	
MONTOMN_TRAS (column)	
W_REVISION (table)	
DIA (column)	
ZZ (table)	
PRUEBA (column)	
CALENDARIO_HABIL (view)	
CHEQUES2 (view)	
CHEQUES3 (view)	
DEAL (view)	
EXPEDIENTE_COBRO (view)	
FORMAS (view)	
INSTITUCION_CAMBIARIA (view)	
MENSAJERIA (view)	
MENSAJERO (view)	
MOVS_BANC_A (view)	View already exists
MOVS_BANC_C (view)	View already exists
PIZARRON_MONEX (view)	
UTI (view)	
UTILIDAD	(view)

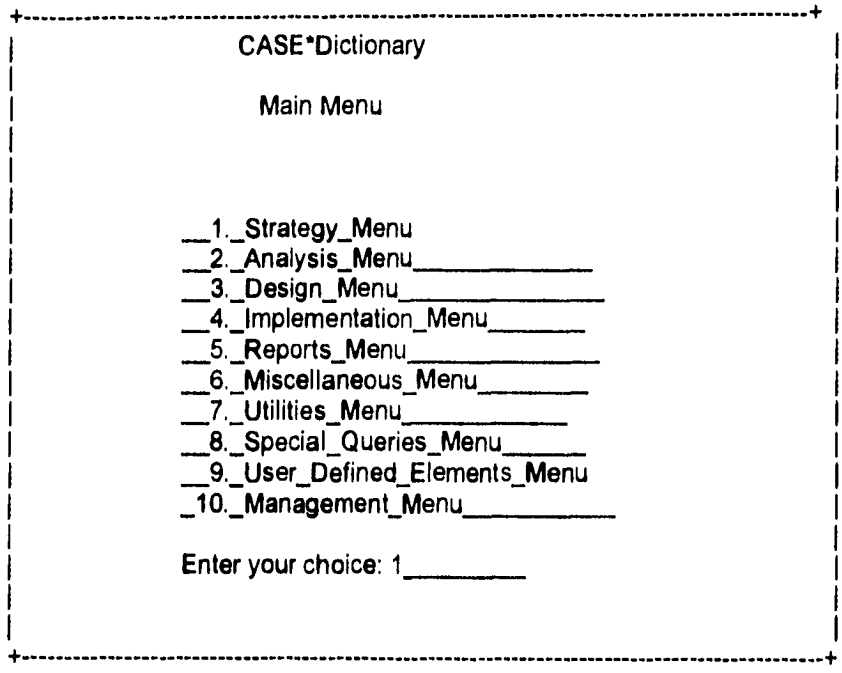
APENDICE 3

PANTALLAS DE CASE *DICTIONARY

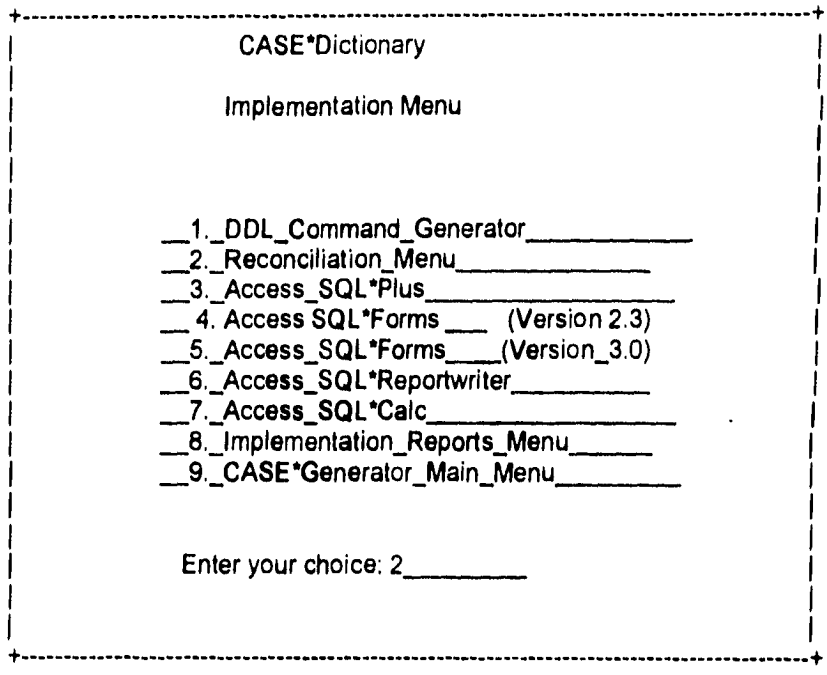
APENDICE 3.
Pantallas de CASE*Dictionary



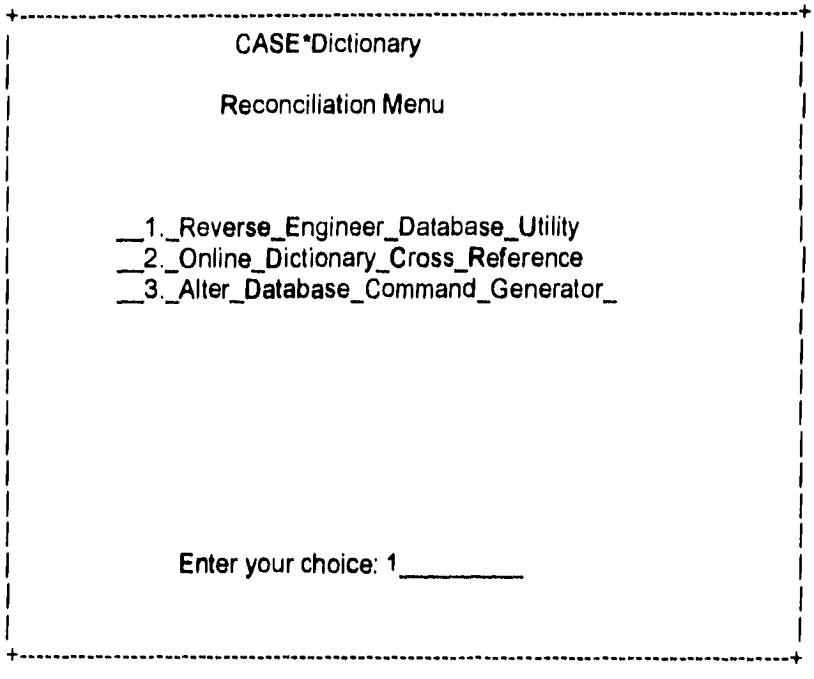
Application: CDCMAN Menu: CDCMAN v <BGM><OSC> <Rep>



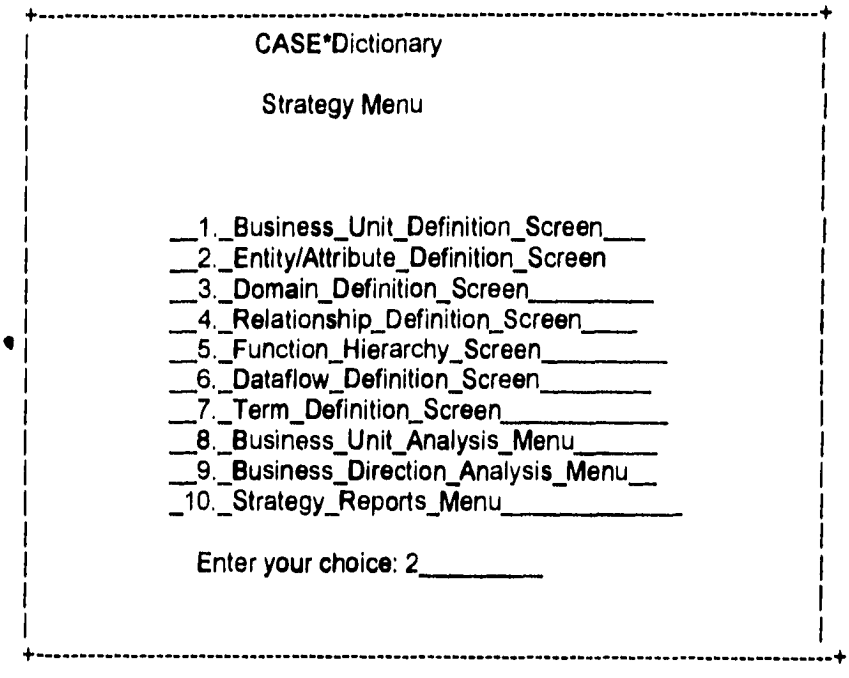
Application: CDCICT Menu: CDCDICT v <BGM><OSC> <Rep>



Application: CDCDICT Menu: CDIMPMN <BGM><OSC> <Rep>



Application: CDCDICT Menu: CDRECMN <BGM><OSC> <Rep>



Application: CDCDICT Menu: CDSTRMN <BGM><OSC> <Rep>

CASE*Dictionary Entity/Attribute Definition Page 1 of 2
18-MAR-95
Appl SCORE_____ Version 1_ Owns Entity ? YES SCORE____

+----------AQ

| Name COMPRA_VENTA_____ rt Name CV_____ Type I |

| Plural COMPRA_VENTAS_____ Volumes - Initial _____ |
| Average _____ |
| Type of _____ Maximum _____ |
| Annual growth rate (%) ___ |

+---------- Synonyms ----------+

| _____ |
| _____ |

+---------- Description ----------+

| _____ |
| _____ |
| _____ |
| _____ |

+----------+

The_name_of_an_entity_(a_thing_of_significance)_____
Count: 36 ^ v <Replace>

CASE*Dictionary Entity/Attribute Definition Page 2 of 2
18-MAR-95
Appl SCORE_____ Version 1_ Owns Entity ? YES SCORE____
AQ

+---------- Entity Name COMPRA_VENTA_____ ----------+
| Attribute Name Create In |
| CONFIRMA_____ Opt? Format Length UID? UIDs |
| In Domain _____ Y CHAR_____ _10__ _ 0_ |
| Notes confirma_cobro_de_operacion_____ |
| |
| Attribute Name Create In |
| COSTOMN_____ Opt? Format Length UID? UIDs |
| In Domain _____ Y NUMBER____ _12 6_ _ 0_ |
| Notes costo_divisa_____ |
| |
| Attribute Name Create In |
| CVEDET_____ Opt? Format Length UID? UIDs |
| In Domain _____ Y NUMBER____ __3 0_ _ 0 |
| Notes numero_de_detalle_de_la_operacion_____ |
| |
+---+

The_name_of_an_attribute_which_describes_the_entity_____
Count: 3 v <Replace>

CASE*Dictionary Domain Definition Page 1 of 2
18-MAR-95
Appl SCORE_____ Version 1_ Owns Domain ? YES CORE____
AQ

Name CLAVE_____ Subset of _____

Description RESTRICCIONES_PARA_LAS_CLAVES_____

Format NUMBER_ Av Len 4__Mx Len 6__ Dec Pl __ Units ___

Datatype NUMBER____ Av Len 4____ Mx Len 6____ Dec Pl _

Authority ANY_____

Default _____

Null value _____

Derivation _____

The_name_of_a_domain_____
Count: 4 ^ v <Replace>

CASE*Dictionary Domain Definition Page 1 of 2
18-MAR-95
Appl SCORE_____ Version 1_ Owns Domain ? YES SCORE____
AQ

+----------+

| Name MONTO_____ Subset of _____ |

| Description Monto_o_importe_____ |

| Format NUMBER__ Av Len ___ Mx Len 15___ Dec Pl 2_ Units |

| Datatype NUMBER____ Av Len _____ Mx Len 12___ Dec Pl 2_ |

| Authority ANY_____ |

| Default _____ |

| Null value _____ |

| Derivation _____ |

+----------+

The_name_of_a_domain_____
Count: 4 ^ v <Replace>

CASE*Dictionary Relationship Definition Page 1 of 2

18-MAR-95

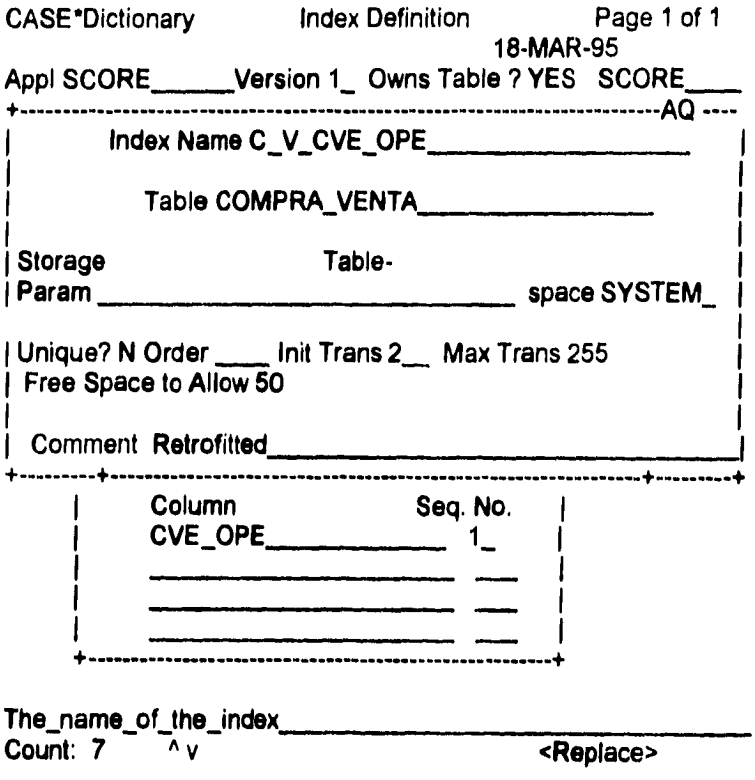
Appl SCORE___Version 1_ Owns Relationship ? YES SCORE____

AQ

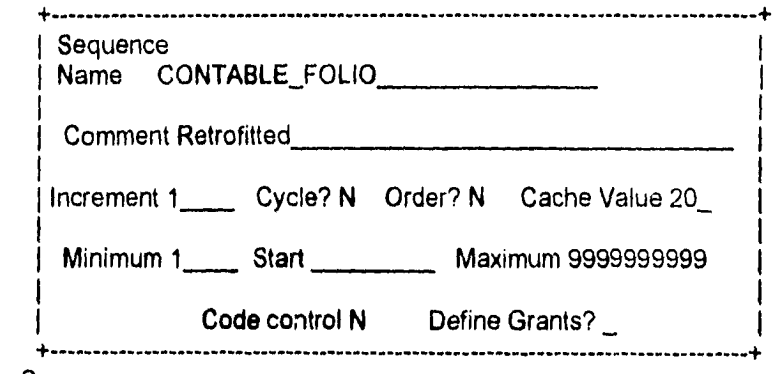
```
+---------- From Entity 1 ----------------------------------------------------------------+
|Entity Name  CLIENTE_____________  Opt? Y  In Arc ___                                     |
|                                                                                          |
|Rel. Name  tiene_____________Degree ONE_OR_MORE_____                                      |
|                                                                                          |
|Volumes - Ave ____ Min _____ Max _____  Allow Transfer Y                                  |
|                                                                                          |
|EACH_CLIENTE_MAY_BE_tiene_ONE_OR_MORE_DESTINO_CLIENTES_                                   |
+---------- To Entity 2 ------------------------------------------------------------------+
|Entity Name  DESTINO_CLIENTE_____________  Opt? N  In Arc ___                             |
|                                                                                          |
|Rel. Name  pertenece_____________ Degree ONE_AND_ONLY_ONE___                              |
|Rel. Owner SCORE_____ Version 1_                                                          |
|                                                                                          |
|Volumes - Ave _____ Min _____ Max _____  Allow Transfer _                                 |
|                                                                                          |
|EACH_DESTINO_CLIENTE_MUST_BE_pertenece_                                                   |
|ONE_AND_ONLY_ONE_CLIENTE_____________                                                     |
+------------------------------------------------------------------------------------------+
```

The_name_of_the_entity_at_this_relationship_end_____

Count: 1 v <List><Replace>


CASE*Dictionary Index Definition Page 1 of 1
18-MAR-95
Appl SCORE Version 1_ Owns Table ? YES SCORE
AQ
Index Name C_V_CVE_OPE
Table COMPRA_VENTA
Storage Table-
Param space SYSTEM_
Unique? N Order Init Trans 2 Max Trans 255
Free Space to Allow 50
Comment Retrofitted
Column Seq. No.
CVE_OPE 1_
The_name_of_the_index
Count: 7 ^ v
<Replace>

CASE*Dictionary Sequence Definition Page 1 of 1
18-MAR-95
Appl SCORE__Version 1_ Owns Sequence? YES SCORE____
AQ



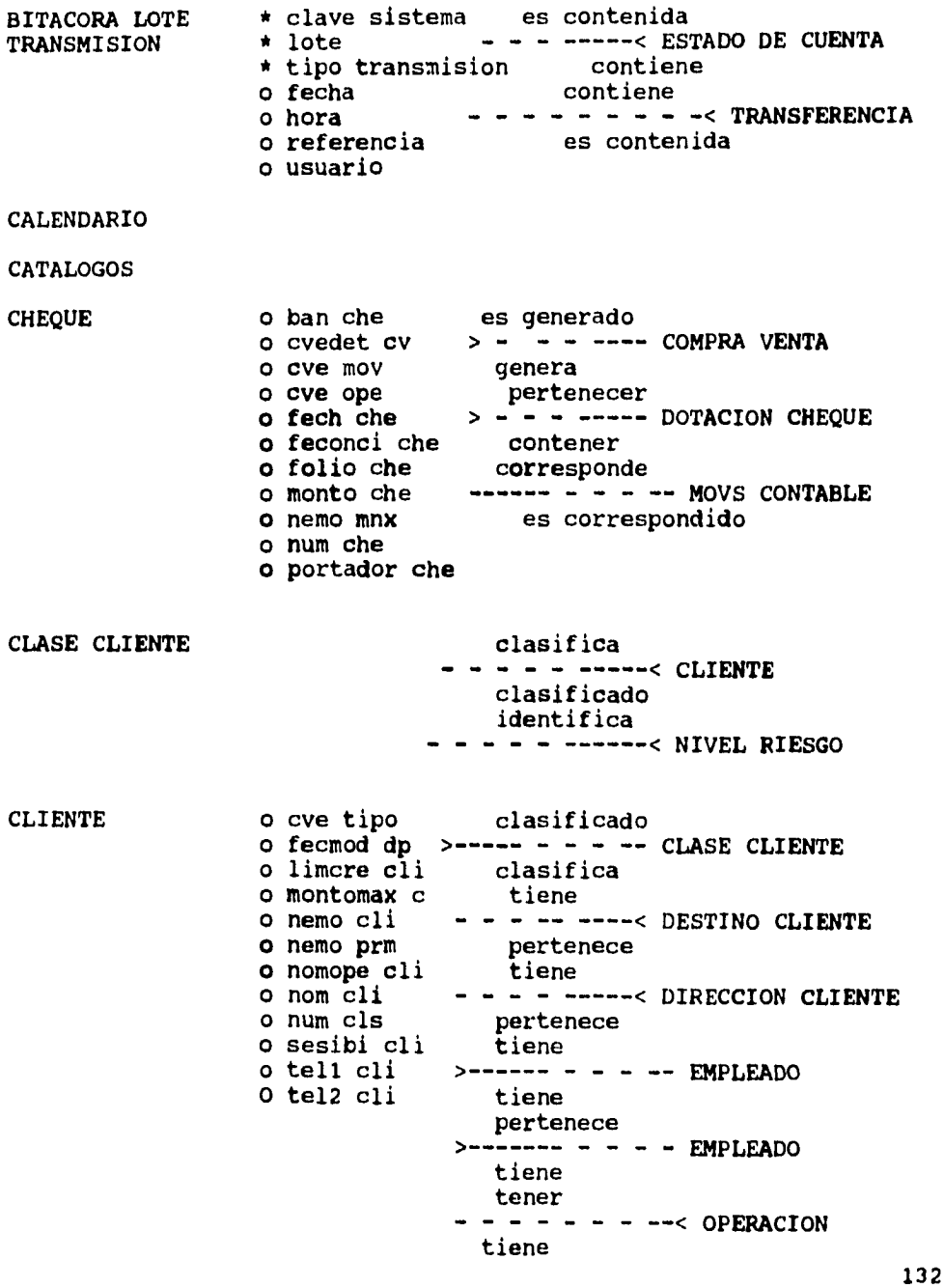
Sequence_name_____
Count: 5 ^ v <Replace>

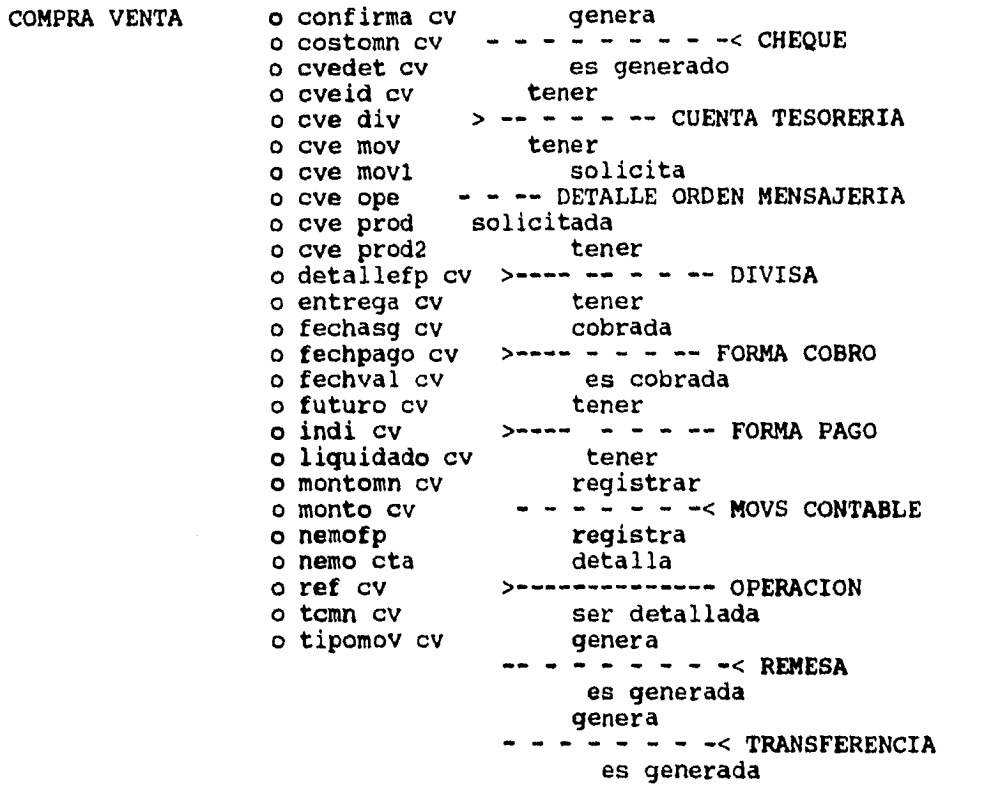
APENDICE 4

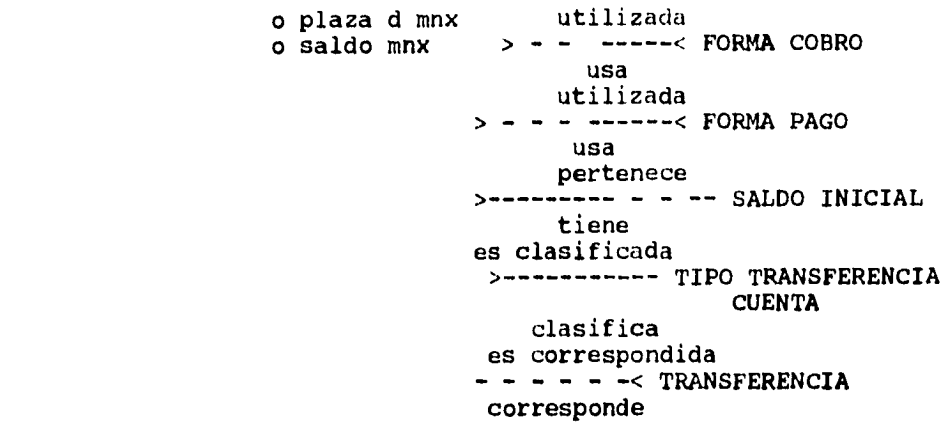
REPORTE ENTIDAD-RELACION

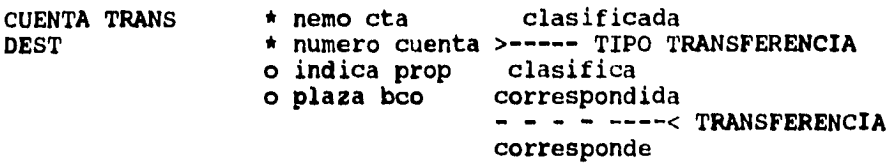
APENDICE 4
REPORTE ENTIDAD-RELACION

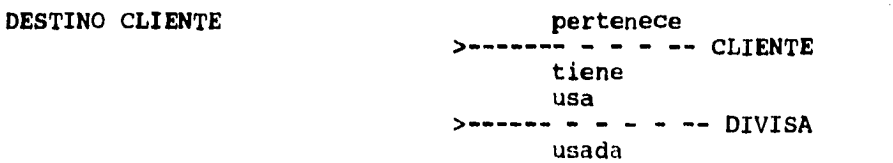
```
 CASE*Dictionary Reports

 Report          : ENTITY RELATIONSHIP DIAGRAM DETAILS
Filename         : cdmodel.lis
Run By           : SCORE
Report date      : 20-AUG-94 10:28am
+-------------------------------------------------------+
|                                                       |
|  Parameter values                                     |
|                                                       |
|  Application System : SCORE                           |
|  Version            : 1                               |
|                                                       |
+-------------------------------------------------------+

Date : 20-AUG-94                                                    Page : 1
                    Entity Relationship Diagram Details

Entity Name      Attributes     Relationships     To Entities
---------------  ----------     --------------    -----------

ADMINISTRADOR

AGENDA                                pertence
                  >---------------- - - - - - - -- EMPLEADO
                                       tiene
                                        utiliza
                   >---------------- - - - - - - -- RESULTADOS AGENDA
                                          usa



ARCHIVOS DE
TRANSFERENCIAS

BITACORA LOTE        * lote      son contenidos
CONCILACION          o fecha   - - -<  MOVIMIENTOS ESTADO DE CUENTA
                     o hora      contiene
                     o num movs        son contenidos
                     o tipo conc   - - - - - - -< MOVS CONTABLE
                     o usuario         contiene
```

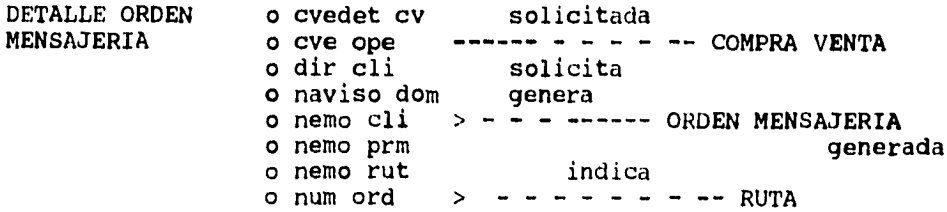

BITACORA LOTE TRANSMISION
* clave sistema
* lote
* tipo transmision
o fecha
o hora
o referencia
o usuario
es contenida
ESTADO DE CUENTA
contiene
contiene
TRANSFERENCIA
es contenida
CALENDARIO
CATALOGOS
CHEQUE
o ban che
o cvedet cv
o cve mov
o cve ope
o fech che
o feconci che
o folio che
o monto che
o nemo mnx
o num che
o portador che
es generado
COMPRA VENTA
genera
pertenecer
DOTACION CHEQUE
contener
corresponde
MOVS CONTABLE
es correspondido
CLASE CLIENTE
clasifica
CLIENTE
clasificado
NIVEL RIESGO
CLIENTE
o cve tipo
o fecmod dp
o limcre cli
o montomax c
o nemo cli
o nemo prm
o nomope cli
o nom cli
o num cls
o sesibi cli
o tel1 cli
O tel2 cli
clasificado
CLASE CLIENTE
clasifica
tiene
DESTINO CLIENTE
pertenece
tiene
DIRECCION CLIENTE
pertenece
tiene
EMPLEADO
tiene
pertenece
EMPLEADO
tiene
tener
OPERACION
tiene

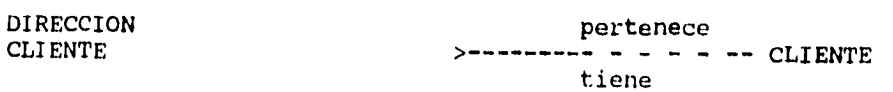

COMPRA VENTA
o confirma cv
o costomn cv
o cvedet cv
o cveid cv
o cve div
o cve mov
o cve mov1
o cve ope
o cve prod
o cve prod2
o detallefp cv
o entrega cv
o fechasg cv
o fechpago cv
o fechval cv
o futuro cv
o indi cv
o liquidado cv
o montomn cv
o monto cv
o nemofp
o nemo cta
o ref cv
o tcmn cv
o tipomov cv
genera
es generado
tener
solicita
solicitada
tener
cobrada
es cobrada
tener
tener
registrar
registra
detalla
ser detallada
genera
es generada
genera
es generada
CHEQUE
CUENTA TESORERIA
DETALLE ORDEN MENSAJERIA
DIVISA
FORMA COBRO
FORMA PAGO
MOVS CONTABLE
OPERACION
REMESA
TRANSFERENCIA
CUENTA CONTABLE
o desc cta
o fecha cta
o nat cta
o nemo cta
o niv cta
o num cta
o saldoi cta
o tipo cta
referenciar
referente
SALDO INICIAL
CUENTA TESORERIA
o bancaj mnx
o banche mnx
o baninv mnx
o bantrf mnx
o destino mn
o nemo cta
o nemo mnx
o numero cuenta mnx
o num fax mnx
tener
tener
usa
utilizada
son correspondidas
corresponden
COMPRA VENTA
DIVISA
ESTADO DE CUENTA

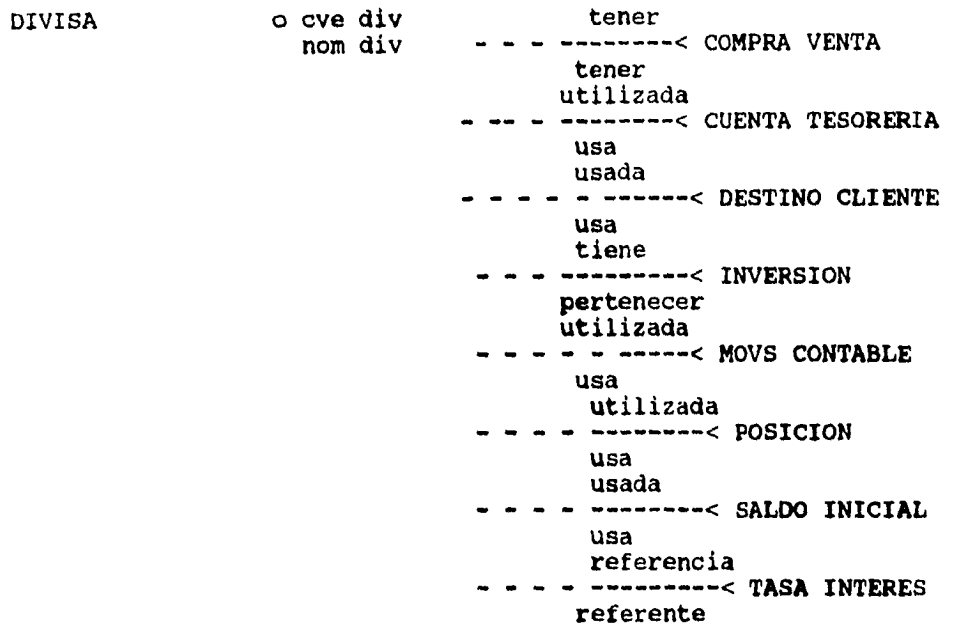

o plaza d mnx
o saldo mnx
utilizada
FORMA COBRO
usa
utilizada
FORMA PAGO
usa
pertenece
SALDO INICIAL
tiene
es clasificada
TIPO TRANSFERENCIA
CUENTA
clasifica
es correspondida
TRANSFERENCIA
corresponde

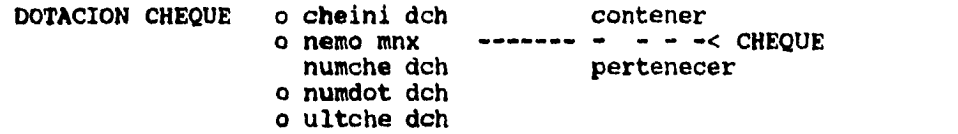

CUENTA TRANS DEST
* nemo cta
* numero cuenta
o indica prop
o plaza bco
clasificada
TIPO TRANSFERENCIA
clasifica
correspondida
TRANSFERENCIA
corresponde

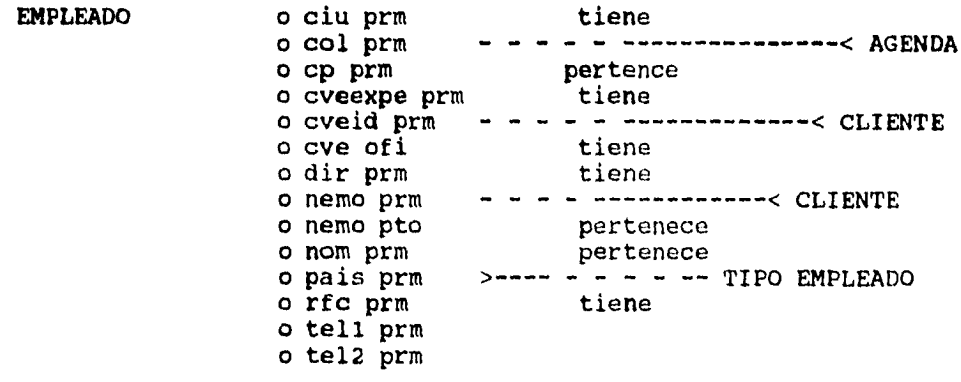

DESTINO CLIENTE
pertenece
CLIENTE
tiene
usa
DIVISA
usada

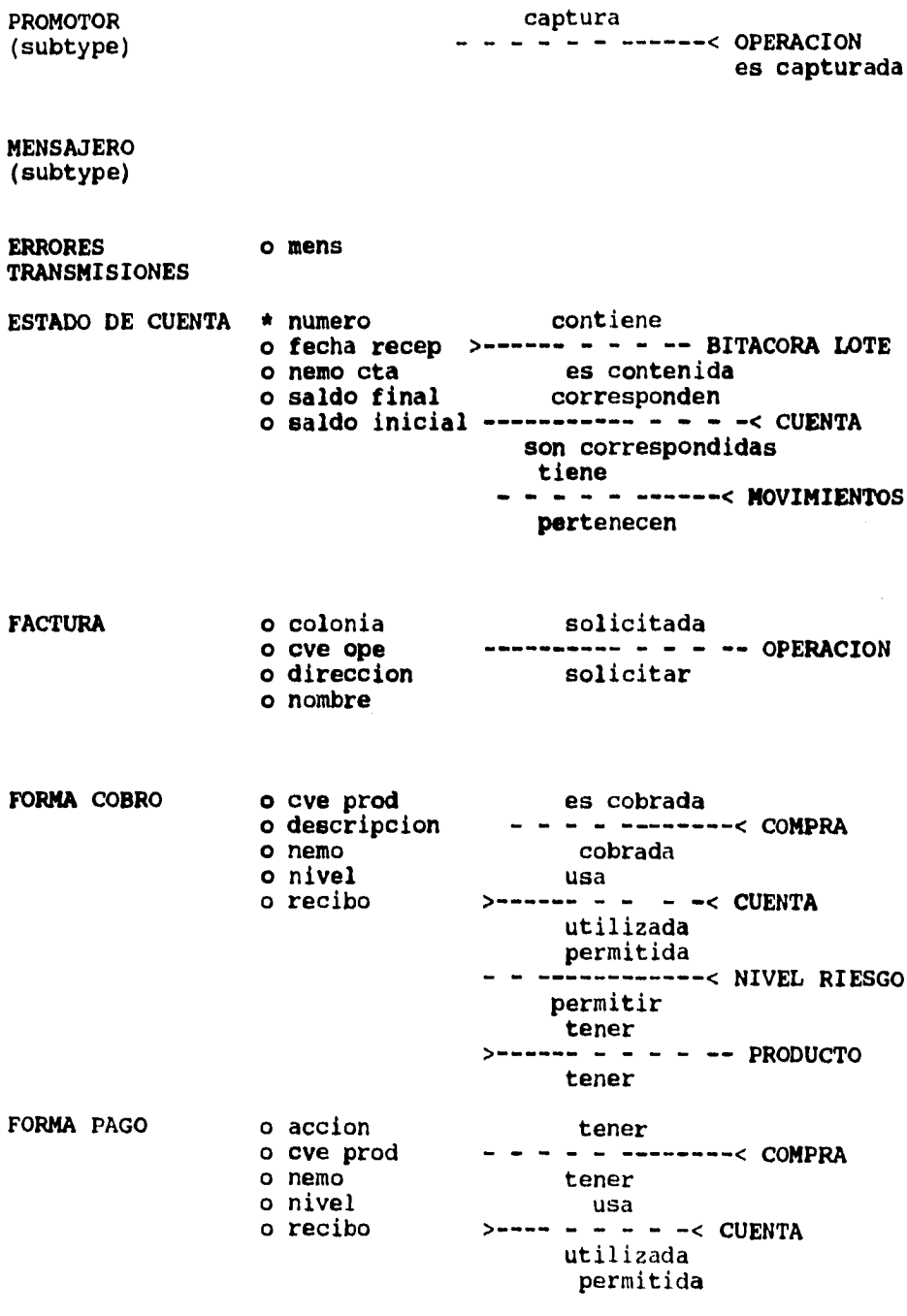

DETALLE ORDEN MENSAJERIA
o cvedet cv
o cve ope
o dir cli
o naviso dom
o nemo cli
o nemo prm
o nemo rut
o num ord
o status dom
o tipomov cv
solicitada
COMPRA VENTA
solicita
genera
ORDEN MENSAJERIA
generada
indica
RUTA
indicada


DIRECCION CLIENTE
pertenece
CLIENTE
tiene


DIVISA
o cve div
nom div
tener
COMPRA VENTA
tener
utilizada
CUENTA TESORERIA
usa
usada
DESTINO CLIENTE
usa
tiene
INVERSION
pertenecer
utilizada
MOVS CONTABLE
usa
utilizada
POSICION
usa
usada
SALDO INICIAL
usa
referencia
TASA INTERES
referente


DOTACION CHEQUE
o cheini dch
o nemo mnx
numche dch
o numdot dch
o ultche dch
contener
CHEQUE
pertenecer


EMPLEADO
o ciu prm
o col prm
o cp prm
o cveexpe prm
o cveid prm
o cve ofi
o dir prm
o nemo prm
o nemo pto
o nom prm
o pais prm
o rfc prm
o tel1 prm
o tel2 prm
tiene
AGENDA
pertence
tiene
CLIENTE
tiene
tiene
CLIENTE
pertenece
pertenece
TIPO EMPLEADO
tiene


PROMOTOR
(subtype)
captura
OPERACION
es capturada
MENSAJERO
(subtype)
ERRORES
TRANSMISIONES
o mens
ESTADO DE CUENTA
* numero
o fecha recep
o nemo cta
o saldo final
o saldo inicial
contiene
BITACORA LOTE
es contenida
corresponden
CUENTA
son correspondidas
tiene
MOVIMIENTOS
pertenecen
FACTURA
o colonia
o cve ope
o direccion
o nombre
solicitada
OPERACION
solicitar
FORMA COBRO
o cve prod
o descripcion
o nemo
o nivel
o recibo
es cobrada
COMPRA
cobrada
usa
CUENTA
utilizada
permitida
NIVEL RIESGO
permitir
tener
PRODUCTO
tener
FORMA PAGO
o accion
o cve prod
o nemo
o nivel
o recibo
tener
COMPRA
tener
usa
CUENTA
utilizada
permitida

```
                                  - - - - - -----< NIVEL RIESGO
                                        permite
                                         tener
                                >------- -- - - -- PRODUCTO
                                         tener
```

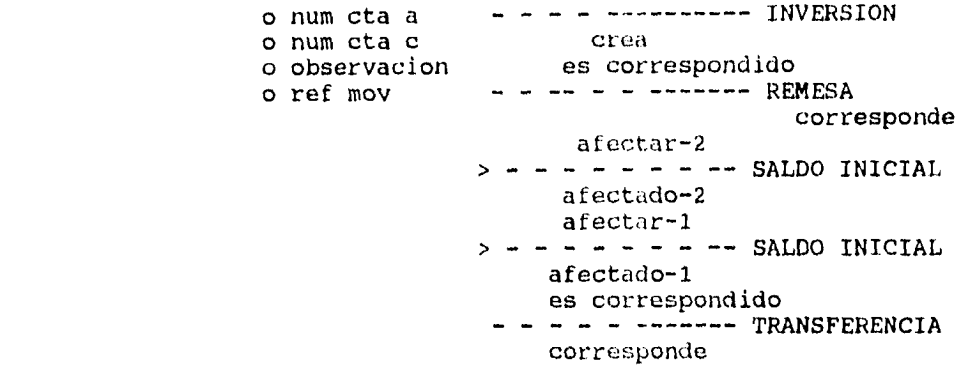
GRUPO

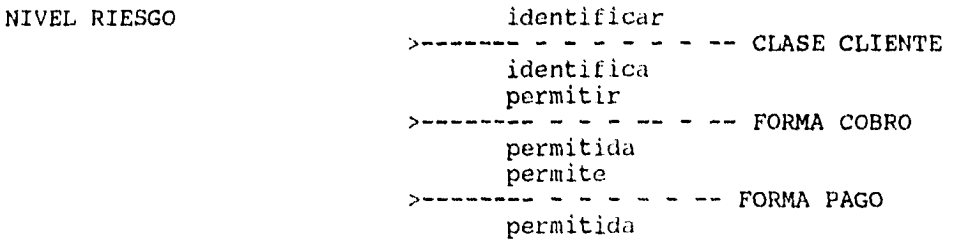
```
INVERSION          o cve inv                pertenecer
                   o cve mov          >----- - - - - - - -- DIVISA
                   o fech1 inv              tiene
                   o fech2 inv              crea
                   o instruc inv      --------- -  - - - -- MOVS
                   o monto inv               es creado
                   o nemo mnx d
                   o nemo mnx inv
                   o nemo mnx o
                   o stat inv
                   o tr inv
                   o uti inv
```

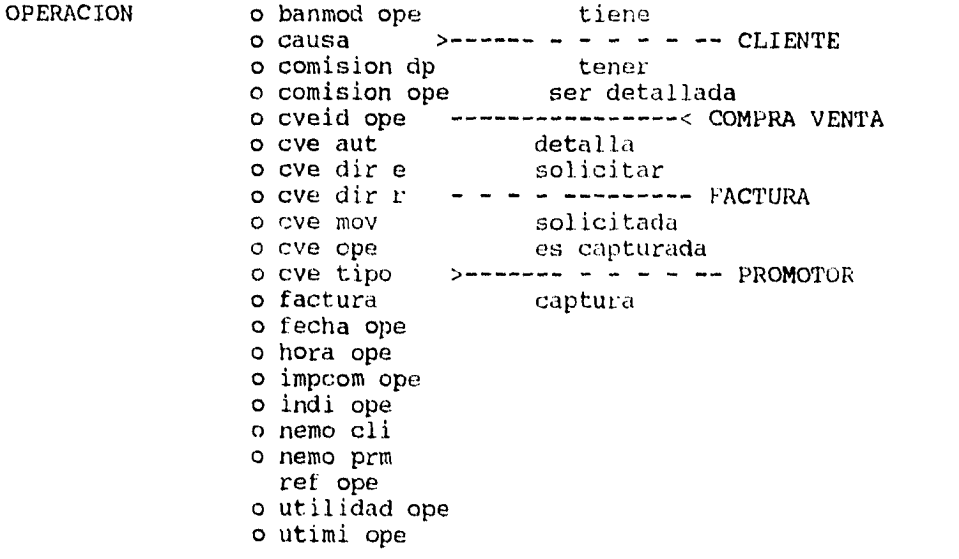
MATERIA

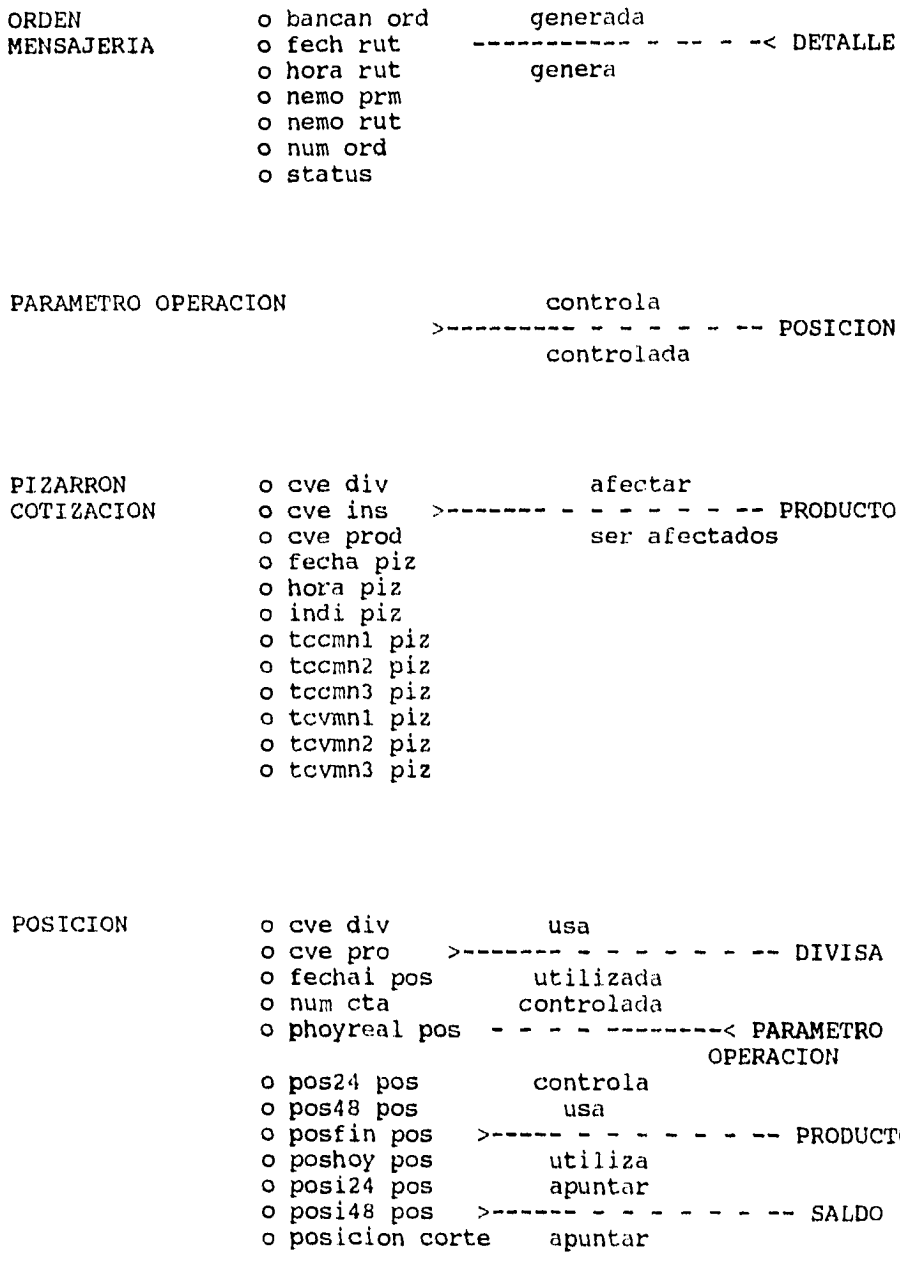
```
MOVIMIENTOS        * fecha valor          contiene
ESTADO DE CUENTA   * importe        > -- - - - - -- BITACORA LOTE
                   * numero               son contenidos
                   o fecha conc           pertenecen
                   o observaciones >------ -- - - -- ESTADO DE
                   o referencia           tiene
```

```
MOVS CONTABLE      o concepto mov          contiene
                   o concia mov      > -- - - - - - -- BITACORA LOTE
                   o concic mov             son contenidos
                   o cveant mov            es correspondido
                   o cveid prm       - - - - ---------------- CHEQUE
                   o cve div               corresponde
                   o cve mov               registra
                   o cve pro         >  - - - - - - - -- COMPRA VENTA
                   o fecha                 registrar
                   o fecha mov             usa
                   o fechval mov     > ------ - - - - -- DIVISA
                   o monto mov             utilizada
                   o numpol op             es creado
```


o num cta a
o num cta c
o observacion
o ref mov
INVERSION
crea
es correspondido
REMESA
corresponde
afectar-2
SALDO INICIAL
afectado-2
afectar-1
SALDO INICIAL
afectado-1
es correspondido
TRANSFERENCIA
corresponde


NIVEL RIESGO
identificar
CLASE CLIENTE
identifica
permitir
FORMA COBRO
permitida
permite
FORMA PAGO
permitida


OPERACION
o banmod ope
o causa
o comision dp
o comision ope
o cveid ope
o cve aut
o cve dir e
o cve dir r
o cve mov
o cve ope
o cve tipo
o factura
o fecha ope
o hora ope
o impcom ope
o indi ope
o nemo cli
o nemo prm
ref ope
o utilidad ope
o utimi ope
tiene
CLIENTE
tener
ser detallada
COMPRA VENTA
detalla
solicitar
FACTURA
solicitada
es capturada
PROMOTOR
captura


ORDEN MENSAJERIA
o bancan ord
o fech rut
o hora rut
o nemo prm
o nemo rut
o num ord
o status
generada
genera
DETALLE
PARAMETRO OPERACION
controla
controlada
POSICION
PIZARRON COTIZACION
o cve div
o cve ins
o cve prod
o fecha piz
o hora piz
o indi piz
o tccmn1 piz
o tccmn2 piz
o tccmn3 piz
o tcvmn1 piz
o tcvmn2 piz
o tcvmn3 piz
afectar
ser afectados
PRODUCTO
POSICION
o cve div
o cve pro
o fechai pos
o num cta
o phoyreal pos
o pos24 pos
o pos48 pos
o posfin pos
o poshoy pos
o posi24 pos
o posi48 pos
o posicion corte
usa
utilizada
controlada
PARAMETRO OPERACION
controla
usa
utiliza
apuntar
apuntar
DIVISA
PRODUCTO
SALDO

o posinic1 pos
o posinic2 pos
o posinic3 pos
o posinic pos
o tcppi1 pos
o tcppi2 pos
o tcppi3 pos

PRODUCTO
o cve prod
o desc prod
o destino prod

tener
- - - ---------< FORMA COBRO
tener
tener
- - - ----------< FORMA PAGO
tener
ser afectados
- - - ----------< PIZARRON
afectar
utiliza
- - - ----------< POSICION
usa

RECIBE MENS
o linea

REMESA

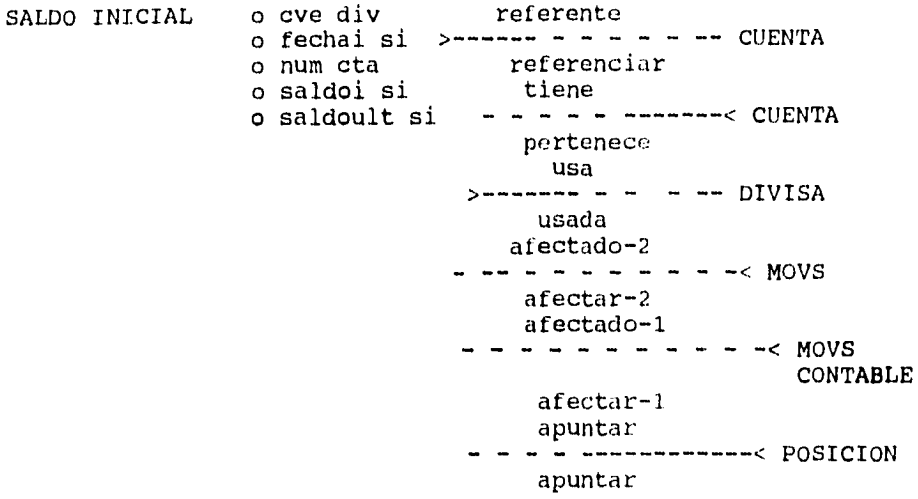
es generada
> - - - - - - - - -- COMPRA VENTA
genera
corresponde
---------- - - - - - MOVS CONTABLE
es correspondido

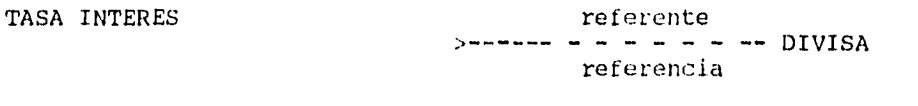
RESULTADOS AGENDA

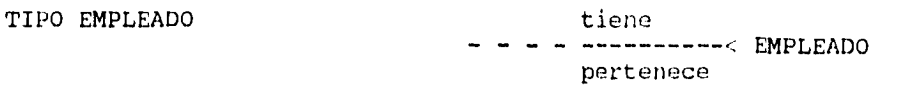
usa
- - - - - - -------< AGENDA
utiliza

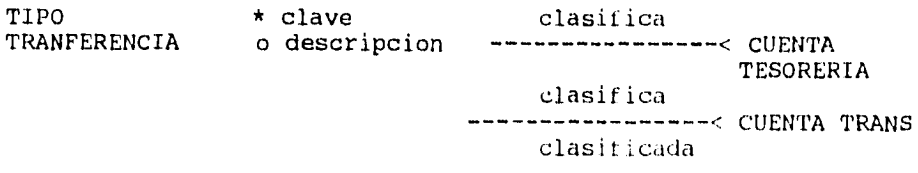
RUTA

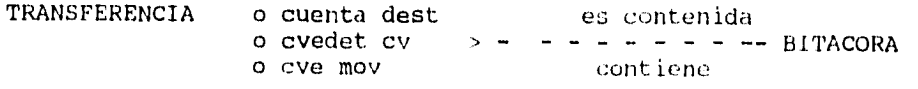
indicada
- - - - - -- - -< DETALLE ORDEN MENSAJERIA
indica

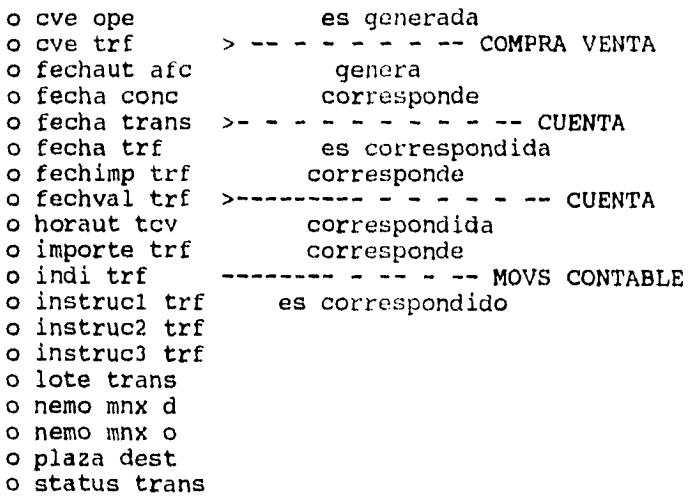

SALDO INICIAL
o cve div
o fechai si
o num cta
o saldoi si
o saldoult si
referente
CUENTA
referenciar
tiene
CUENTA
pertenece
usa
DIVISA
usada
afectado-2
MOVS
afectar-2
afectado-1
MOVS
CONTABLE
afectar-1
apuntar
POSICION
apuntar


TASA INTERES
referente
DIVISA
referencia


TIPO EMPLEADO
tiene
EMPLEADO
pertenece

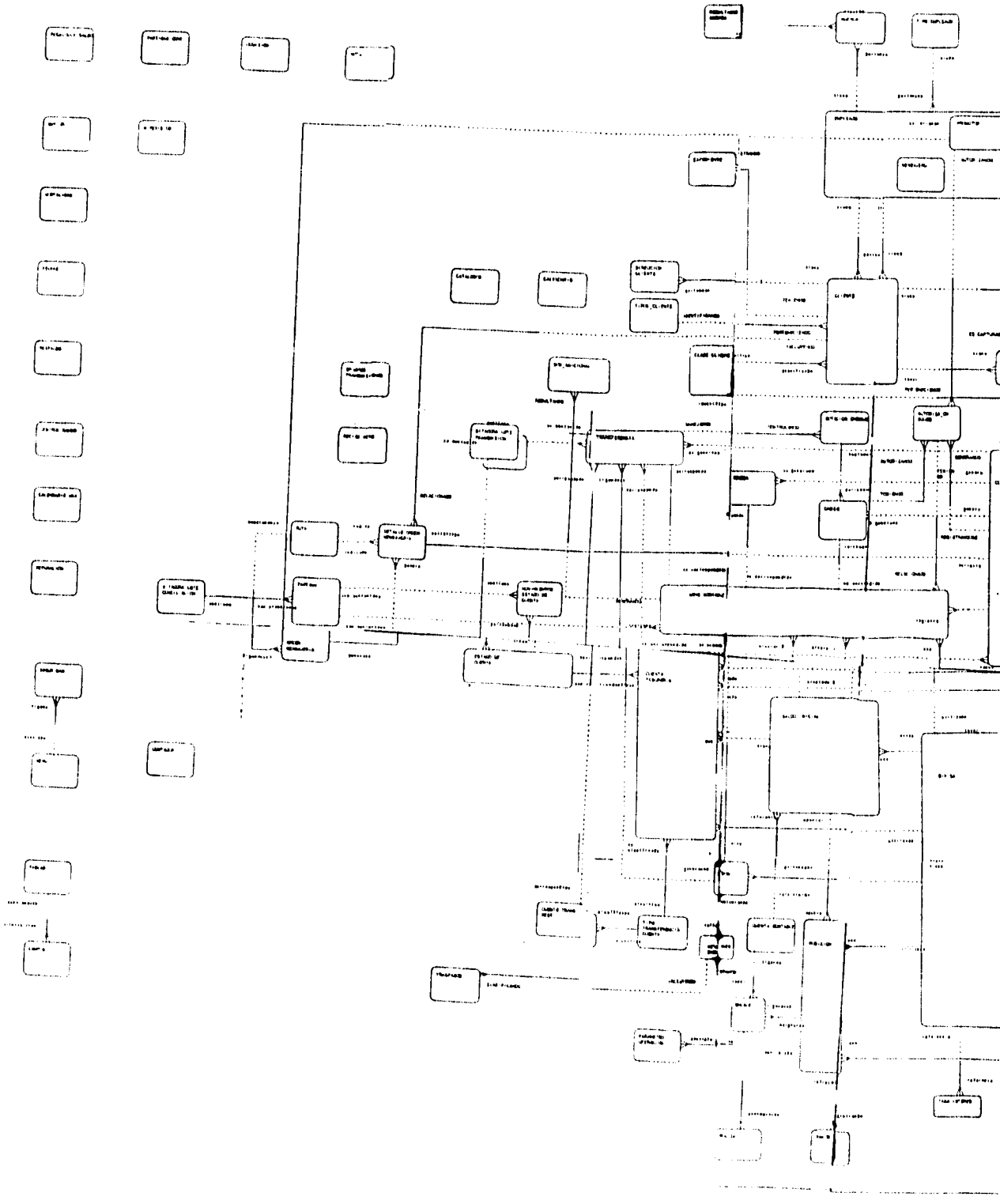

TIPO
TRANFERENCIA
* clave
o descripcion
clasifica
CUENTA
TESORERIA
clasifica
CUENTA TRANS
clasificada

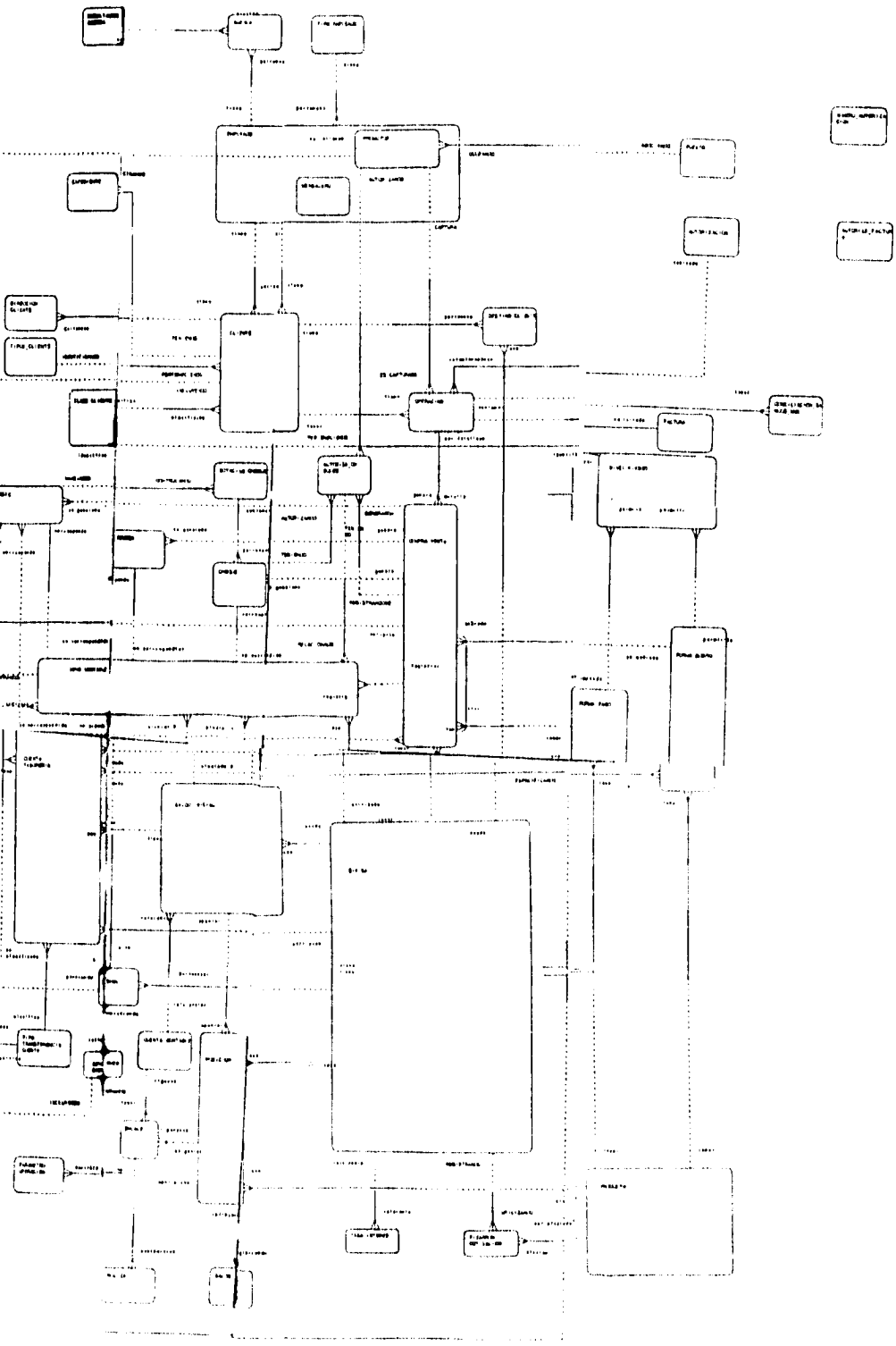

TRANSFERENCIA
o cuenta dest
o cvedet cv
o cve mov
es contenida
BITACORA
contiene


o cve ope
o cve trf
o fechaut afc
o fecha conc
o fecha trans
o fecha trf
o fechimp trf
o fechval trf
o horaut tcv
o importe trf
o indi trf
o instruc1 trf
o instruc2 trf
o instruc3 trf
o lote trans
o nemo mnx d
o nemo mnx o
o plaza dest
o status trans
es generada
> -- - - - - - - -- COMPRA VENTA
genera
corresponde
>- - - - - - - - - - -- CUENTA
es correspondida
corresponde
>---------- - - - - - -- CUENTA
correspondida
corresponde
-------- - -- - -- MOVS CONTABLE
es correspondido

APENDICE 5

DIAGRAMA ENTIDAD-RELACION





GLOSARIO

A

ABSTRACCION.

Proceso de fragmentar un programa en cada vez menos componentes especificos. Esta modularización ayuda a fragmentar tareas de programación en piezas manejables y ocultar detalles de programación al usuario.

ACM (ASSOCIATION FOR COMPUTING MACHINERY).

Una organización profesional trabajando con el propósito de apoyar los avances en la industria de la computación.

ANALISIS DE SISTEMAS.

Examen de una empresa, procedimiento, método o actividad para determinar lo que deba lograrse y de qué manera. En general, el uso se refiere al análisis de un problema para determinar la solución ideal computarizada.

APLICACION.

Problema o sistema específico al que se aplican técnicas de procesamiento de datos. A menudo se refiere a las aplicaciones como computacionales (que requieren un elevado nivel de computación) o de procesamiento de datos (que requieren un elevado nivel de manejo de estos).

ATRIBUTO.

Característica, generalmente de un campo de dato. De esa manera, los atributos de un campo dado pueden describir su formato, tamaño, nombre, etc.

- Los atributos describen entidades y son las piezas específicas de información las cuales necesitan ser conocidas.

- Son información que se necesita conocer o tener acerca de una entidad. Los atributos describen una entidad para calificar, identificar, clasificar cuantificar o expresar el estado de una entidad.

ANSI (AMERICAN NATIONAL STANDARS INSTITUTE).

Es una organización encaminado al desarrollo de los estándares de la industria americana (no precisamente en el área de computación).
ANSI está respaldada por más de 1000 compañías, organizaciones comerciales y sociedades profesionales.

ASCII (AMERICAN STANDAR CODE FOR INFORMATION INTERCHANGE).

Este es el sistema de código de caracter más usados por los sistemas de computo. Se usa para convertir valores numéricos en caracteres y dígitos que los humanos puedan entender y viceversa.

ASCII define 128 caracteres, cada uno asignado a un valor numérico entre 0 y 127.

En el ASCII extendido se usa un valor numérico entre 128 y 255 para asociarlo a un caracter.

B

BANCO DE DATOS.

Archivo amplio de datos, generalmente almacenado en un dispositivo de acceso directo y que ofrece acceso a un número grande de usuarios. Con frecuencia se conecta y actualiza mediante un sistema en tiempo real.

BASE DE DATOS.

1.- Archivo de datos estructurado de tal manera que se pueden expresar todas las relaciones lógicas entre los registros , y que es independiente de cualquier aplicación específica. De esa manera, lso programas que accesan la base de datos no imponen ellos mismos restricciones a la organización de los datos que contiene.

2.-Una colección grande y organizada de información a la que se accede mediante el software y que es una parte integral del funcionamiento de un sistema.

BASE DE DATOS RELACIONAL.

Es una base de datos que es percibida por el usuario como una colección de relaciones de tablas de dos dimensiones.

BUSQUEDA EN TABLA.

1.-Selección de un valor de función que corresponde a un argumento de una tabla de valores de función.

2.- Buscar dentro de una tabla para localizar un artículo con una llave específicada.

C

CAMPO

Es la implementación de un item o dato en una tabla. Este puede ser caracter, numérico, o de cualquier otro formato, puede ser opcional u obligatorio.
Es un atributo de una entidad.

CASE

CASE es un acrónimo de Computer-Aided System Engineering, es la combinación de gráficas, diccionario, generador y otras herramientas de software que pueden ser usadas para asistir al desarrollo de sistemas así como en el mantenimiento de los mismos con la ayuda de un método estructurado.

COBERTURA.

Monto del seguro tomado contra cualquier riesgo.
Porción del universo incluida en un estudio o inspección.

CONCILIACION.

Determinación de las partidas necesarias para que los saldos de dos o más cuentas relacionadas o estados concuerden entre sí. Ejemplo: en la conciliación del saldo bancario de un depositante; los los cheques en circulación que se no se han presentado al banco; los honorarios por cobrar, gastos de protesto y otros similares, cargados por el banco, pero no asentados en los libros del cliente; las letras cobradas , las rebajas de interes, y así sucesivamente, acreditadas por el banco, pero aún no asentadas en los libros del cliente, son conceptos cuyo importe neto tendrá que tomarse en cuenta para acordar los saldos entre el depositante y el banco, sumándolos al saldo menor de los dos restándolos del mayor.

COMISIONES.

Remuneración de un empleado o agenda, relacionado con servicios prestados en conexión con ventas, compras, cobranzas y otros tipos de transacciones mercantiles y usualmente basadas en un porcentaje de las cantidades implicadas en la operación.

CHEQUES.

Documento girado a cargo de un banco, pagadero a la vista.

CLUSTER.

En un archivo repleto, una de las clases en las que pueden agruparse los registros con conjuntos similares de identificadores de contenido.

CONSTRAINT.

Es una resticción que define el conjunto de soluciones admisibles en programación.

D

DATA BASE MANAGEMENT SYSTEM (Sistema de Gestión de Base de Datos DBMS)

Sistema especial o parte de un sistema de proceso de datos, que sirve de ayuda para almacenamiento, manipulación, informe, administración y control de datos.

DB2

Manejador de Bases de Datos de IBM para ambiente Main Frames. Utiliza SQL, y define en sí mismo un dialécto estandar.

DIAGRAMA DE FLUJO.

Es la representación gráfica más ampliamente usada para el diseño procedimental. Es un gráfico muy sencillo, un rombo representa una condición lógica y las flechas muestran el control del flujo.

DIAGRAMA DE FLUJO DE DATOS (DFD).

Es una técnica gráfica que describe el flujo de información y las tranasformaciones que se aplican a los datos, conforme se mueven de la entrada a la salida.

El DFD puede usarse para representar un sistema o software a cualquier nivel de abstracción.

DEAL.

Comerciar, traficar, negociar, tratar.

DIVISA.

Moneda , billete o efecto mercantil de cualquier país extranjero.

DOWNSIZING

El proceso conocido como downsizing consiste en migrar aplicaciones completas o funciones completas, de Mainframes centralizados a redes o sistemas centralizados menores buscando así hacercar la aplicación al usuario final. Toma ese nombre de las técnicas de "adelgazamiento" de los sistemas modernos de producción: "todo lo que no da valor agregado a la producción debe eliminarse", esto es lo que se llama producciones belta y que surge desde 1970.

E

ENTIDAD.

Una entidad en un objeto de interés para los negocios. Es también una clase o categoría de cosas.

ESTRUCTURA DE DATOS.

Es una representación de la representación lógica entre elementos individuales de datos. La estructura de datos es tan importante como la estructura de programas en la representación de la aqruitectura de software.

F

FACTURA.

Documento que muestra el caráter, la cantidad el precio, las condiciones, la forma de entrega y otras particularides de las mercancías vendidas o de servicios prestados.

FORWARD-BACKWARD.

Ascendente-descendente , presenta una entrada aditiva y otra sustractiva y permite el flujo en ambos sentidos.

H

HARDWARE.

Conjunto de elementos físicos que constituyen una computadora. Es el término contrapuesto a software.

HERRAMIENTA DE PROGRAMACION.

Programa que ayuda a desarrollar otras programas.

I

IEEE (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS).

Un organismo profesional fundado en 1963 para el reforzamiento de la educación, investigación y estándares en el campo de la electrónica. El IEEE cuenta con más de 250,000 miembros incluyendo estudiantes y profesionales de la Ingeniería Eléctrica y campos relativos.

INDICE, INDEXAR.

a) Lista, ordenada con arreglo a un cierto criterio, que contiene todas aquellas claves que se emplean para identificar cada uno de los conceptos que integran la relación.

b) Sinónimo de indicador.

c) Poner o asignar indices o indicadores a otras variables (indexar)

INGENIERIA DE SOFTWARE.

Término que describe el proceso de diseñar programas de computadora que son fáciles de escribir, comprabar, modificar, leer y funcionar. El término y intenta abarcar a la programación y a las actividades involucradas a lo largo del ciclo de vida de los programas, sobre todo en proyectos voluminosos.

L

LAN

Local Area Network. Red de área local, por lo general es la red que se encuentra en un mismo edificio.

LL

LLAVE.

Uno o más caracteres dentro de un artículo de datos que se utiliza para identificar el dato o controlar su uso.

LLAVES PRIMARIAS.

Una llave primaria (PK) es una columna o grupo de columnas que identifican de manera única a cada renglón en una tabla. Cada tabla debe tener una llave primaria y una llave primaria debe ser única. Ninguna parte de la llave primaria ser nula.

LLAVE CANDIDATA.

Una tabla puede tener más de una columna o combinación de columnas que pueden servir como la llave primaria de la tabla. Debe ser únicas y no nulas.

LLAVE FORANEA (FK).

Es una columna o combinación de columnas en una tabla, que se refieren a una llave primaria de la misma o en otra tabla.

LLAVE PRIMARIA COMPUESTA.

Es una llave primaria que consta de múltiples columnas.

N

NULL

En algunas implementaciones de manejo de sistemas de Bases de Datos una columna,campo, o dato puede ser requerido para reveservar un valor que necesita. Otras implementaciones como el ORACLE RDBMS, implementan este concepto para obtener un valor no nulo.

O

ORACLE RDBMS

El manejador de Bases de Datos Relacionales de Oracle Corporation.

P

PARADIGMA

Según el diccionario es una palabra tipo que se da como modelo para una declinación, una conjugación. Es un ejemplo o modelo a seguir.

PARTIDA.

Nombre que se da a diferentes cosas: mercancias, muebles, cuentas, etc.

POLIZA.

1.- Forma usada en un sistema de comprobantes (o polizas) a la cual se adjuntan frevuentemente facturas, recibos y otras evidencias de adeudados, mostrando la autorización para hacer el pago, los pormenores de la liquidación y otros detalles pertinentes, una poliza de desembolso.

2.- Evidencia escrita de una transición comercial o contable, contenida algunas veces en un solo documento sin anexos. Ejemplo. una póliza de diario.

PROTOTIPO

Una tecnica para demostrar rapidamente un concepto, para aceptarlo y checar su flexibilidad.

Diseño - para checar la flexibilidad de alternativas y el estilo.

Construccion - para incrementar la construccion de modulos que sean requeridos por el usuario.

Con CASE*Method el uso de un prototipo es recomendado durante:
Analisis - para checar requerimientos y descartar los que no sean validos.

ORACLE RDBMS

El manejador de Bases de Datos Relacionales de Oracle Corporation.

R

RDBMS

Relational Database Management System. Sistema Administrador de Base de Datos Relacionales.

REGISTRO

En una Base de Datos no relacional, un registro es un archivo de entrada , consistente de elementos individuales de información .
En un sistema relacional, un registro es un conjunto de campos.

RELACION

Una relación es bidireccional y representa la asociación entre dos entidades, o entre una entidad consigo misma.

REMESA.

Giro, letra de cambio.

ROBUSTO

Programa que está diseñado de manera que resiste modificaciones sin perder su esencia.

S

SALDO.

Diferencias entre el total de los débitos y el total de los créditos de una cuenta; o el total de una cuenta que contenga solamente débitos o solamente créditos.

SECUENCIAS.

Arreglo de artículos de acuerdo con un conjunto especificado de reglas.

SISTEMA DE INFORMACION.

Término genérico que denota todas las operaciones y procedimientos que participan en un sistema de procesamiento de datos.

SOFTWARE.

Todo programa ejecutable por computadora. La palabra software de base o básico se utiliza con frecuencia para designar el sistema operativo de una computadora más los programas que traducen (compilan o interpretan) lenguajes de alto nivel en lenguaje máquina, tales como compiladores, sistemas operativos, ensambladores, biblioteca de rutinas, etc. Todo el software está constituido por el básico más los programas de usuario.

SQL

Siglas de "Structured Query Language". El lenguaje de consulta y acceso a base de datos más común en la actualidad, definido como estandard por IBM, ANSI e ISO.

T

TABLA.

Estructura de datos formada por un conjunto de elementos, cada uno de ellos compuesto de dos campos: la clave y la información propiamente dicha. Se dice que es una estructura de datos de tipo asociativo porque a ella se accesa por clave.

TRANSACCION U OPERACION.

Evento o suceso , cuyo reconocimiento se expresa en forma de un siento en los registros de contabilidad.

Expuesto en cantidades monetarias, una transacción u operación, tal como la concibe un contador, se encuentra, formada por una igualdad entre créditos y débitos, los primeros representando el origen y los segundos la identificación y la disposición inmediata.

TRANFERENCIAS O TRANSPASOS.

Cesión de bienes, casi siempre junta con el título de propiedad, o de servicios prestados de una persona a otra. Las transferencias o traspasos, en un sentido más amplio, incluyen las ventas y otros conceptos de entrada.

U

UNIX

Sistema Operativo Multiusuario, desarrollado por AT&T. Es considerado muy flexible, poderoso y altamente portable. Corre en muchas plataformas de minis, y en algunas micros y mainframes.

REFERENCIAS

REFERENCIAS

[1] "Software Reverse Engineering: A CASE Estudy",
E.J. Byrne,
Software-Practice and Experience,
Vol. 21 (12), 1349-1364,
Dic. 1991.

[2] "CASE Tools Supporting ADA Reverse Engineering",
8th. Annual National Conference on ADA Technology,
1990, pp 157-164.

[3] "Using and Enabling Technology to Reingineer Legacy Systems",
Lawrence Markosian , Philipe Newcomb , Russell Brand,
Scott Burson and Ted Kitzmiller,
Comunications of the ACM,
Mayo 1994, pp 58-70.

[4] Hogshead-Davis, K. and Arora, A.K. Converting a relation database model into an entity-relationship model. In Proceedings
of the Sixth International Conference on Entity Relationship
Approach. (1987)

BIBLIOGRAFIA

BIBLIOGRAFIA

ARTICULOS :

1. M.G. Rekoff Jr. "On Reverse Engineering"
   IEEE Trans. System, Man and Cibernetics,
   Marzo-Abril 1985, pp. 224-252.

2. T.J. Biggerstaff. "Design an Recovery for Maintenance and Reuse"
   Computer World , julio 1989, pp. 36-49.

3. "Application Reingeneering", Guide Pub.
   CPP-208, Guide Int'l Corp., Chicago, 1989.

4. Blaha. M.R., Permelani, W. Object-Oriented Modeling and
   Design for Database Aplications.
   Prentice Hall, Englewood Clifts.

5. Blaha. M.R., Permelani, W.J., Bender, A.R.,Salemme, E.M.,
   Bill-of-material configuration generation. En la sexta
   conferencia internacional de ingenieria de datos.
   IEEE computer society, Los Alamitos California, 1990.

6. E.Chikofsky and J. H. Cross. Reverse En.. and Desing Recovery
   IEEE Software, 1990, 13-17.

7. K.Bennet. Automated Support of Sotware Maintenance.
   Information and Software Technology, 1991, 74-85.

LIBROS :

8. CASE*METHOD. ORACLE
FUNCTION AND PROCESS MODELLING
RICHARD BARKER, CLIFF LONGMAN
ED. ADDISON-WESLEY PUBLISHING COMPANY
E.U. , 1992
No. PAGS. 386

9. CASE*METHOD. ORACLE
TASKS AND DELIVERABLES
RICHARD BARKER, CLIFF LONGMAN
ED. ADDISON-WESLEY PUBLISHING COMPANY
E.U. , 1989
No. PAGS. 420

10. CASE*METHOD
ENTITY RELATIONSHIP MODELLING
RICHARD BARKER, CLIFF LONGMAN
ED. ADDISON-WESLEY PUBLISHING COMPANY
E.U. , 1992
No. PAGS. 295

11.CASE
CONCEPTS AND IMPLEMENTATION
LARRY E. TOWNER
ED. McGRAW HILL
NEW YORK, NY 1989
No. PAGS. 232

12.INGENIERIA DE SOFTWARE
IAN SOMMERVILLE
ED. ADDISON-WESLEY IBEROAMERICANA
MEXICO D.F., 1990
No. PAGS. 362

13.INFORMATION SYSTEM METHODOLOGIES
T. WILLIAM OLLE
ED. ADDISON-WESLEY IBEROAMERICANA
GRAN BRETANA, 1990
No. PAGS. 401

14. ORACLE "The complete reference",
George Koch,
Osborne McGrawHill
1991.

OTROS:

15. CASE *Dictionary, Reference Guide,
Version 5.0
ORACLE 1991.

16. CASE *Designer, Reference Guide,
Version 1.1
ORACLE 1991.

17. CASE *Generator for SQL *Forms
Tutorial and Reference
Version 1.0
ORACLE 1991.