UNIVERSIDAD VERACRUZANA

Facultad de Estadística e Informática
Licenciatura en Informática

Ingeniería de Software I
Manual de Prácticas

Elaboraron:

Dr. Juan Manuel Fernández Peña
Dra. María de los Ángeles Sumano López

Este material fue utilizado en los periodos:

- febrero – julio 2010, agosto 2010 – enero 2011,
- febrero – julio 2011, agosto 2011 – enero 2012
- febrero – julio 2012, agosto 2012 – enero 2013

En el plan 2002 de la Licenciatura en Informática

Agosto 2011

Contenido

Introducción

Dentro del curso de Ingeniería de Software I, por acuerdo de maestros que forman la Academia de Ingeniería de Software I, se incluyeron los siguientes saberes heurísticos:

- Práctica en herramientas de apoyo al desarrollo de software como: IDE y CASE.
- Aplicación de la metodología escogida para la obtención de los modelos de análisis y diseño de un sistema de software utilizando el CASE e IDE practicados.
- Cálculo de métricas de calidad aplicadas a sistemas de software.
- Planeación de Pruebas de sistema aplicadas al sistema de software en desarrollo.
- Planeación y aplicación de pruebas de unidad para software realizado por cada alumno utilizando una herramienta automática para la aplicación de la prueba.

Cada saber heurístico ocupa uno o varios saberes teóricos y se adquieren realizando trabajos prácticos. El presente manual describe las prácticas correspondientes a la adquisición de los saberes mencionados.

Las prácticas que se incluyen en este manual son de suma importancia para el futuro profesional de un ingeniero de software.

Práctica de herramientas semiautomáticas de apoyo al desarrollo de software

El desarrollo moderno de software utiliza herramientas automáticas y semi-automáticas que apoyan la labor del Ingeniero de Software. Entre ellas están:

- IDE (Integrated Development Environment) que ayudan en la preparación de y codificación de los programas que conforman un sistema, ello mediante un ambiente que contiene elementos tales como: lista de componentes reutilizables, editor de código, revisión sintáctica, depuradores, formación de proyectos y una variedad de herramientas que varía de un IDE a otro.
- CASE (Computer Aided Software Engineering) que facilitan el desarrollo de modelos de desarrollo de software.
- Herramientas de prueba.

Modelos para el Análisis de Software

Al finalizar la serie de prácticas de esta parte del curso el alumno:

- Habrá aprendido a realizar el análisis de sistemas de software basados en la metodología “Análisis Estructurado Moderno” de Edward Yourdon [Yourdon, E. (1993)]
- También será consiente de todos los elementos que se deben considerar para entender lo que se pide en el desarrollo de software.

Las prácticas que forman esta parte del curso se llevan acabo de manera paralela y arrojan una serie de modelos que son: esencial, de comportamiento y de implantación del usuario.

Obtención de Métricas de Software

Esta competencia está directamente relacionada con la calidad de un software ya que son las métricas las que de alguna manera van ayudando a entender tanto los procesos como la calidad que del software se requiere. En este manual se plantea el cálculo de las siguientes métricas:

- Métrica Bang. Utilizada, principalmente, para revisar la completez de los modelos de análisis.
- Métrica de Puntos de Función. Sirve para establecer tanto los restricciones del software que se va a realizar como el esfuerzo que se requerirá para realizar su implementación.
- Métricas Card y Glass para estructura
- Métricas de módulos: Cohesión, acoplamiento y complejidad. Todas ellas dan información sobre un módulo como qué tan integrado está, si es independiente de otros módulos y cuántas deberán las pruebas mínimas que se le deberán aplicar a un módulo.

Planeación de pruebas se Sistema

En la FEI las actividades asociadas a la prueba de software se llevan a cabo conforme se va avanzando en el desarrollo del software, primero se realiza la planeación de las pruebas y luego se aplican. En el nivel de prueba de sistema la planeación ocurre paralelamente con el análisis. Lo anterior lleva varias ganancias: ayuda a entender con mayor precisión qué se quiere del software y el cliente tiene elementos que a futuro le servirán para revisar que el software entregado es lo que pidió.

Modelado de Diseño de Software

El diseño de software es la actividad ingenieril por excelencia dentro del desarrollo de software, durante el diseño se deberán considerar aspectos tales como las plataformas de hardware y software donde correrá el nuevo software; las estructuras generales y detalladas y la forma de cumplir con las restricciones impuestas.

Planeación y Aplicación de Pruebas de Unidad

Una de las últimas tareas que comprende el desarrollo de software es la aplicación de pruebas de unidad. Como la unidad es la parte más pequeña de la estructura del software, generalmente hay muchas, por ello es importante que el desarrollador se apoye en herramientas automáticas que le permitan acelerar la tarea de prueba. En este manual se presenta la práctica de una de tales herramientas: DUnit.

Práctica 1. Introducción a un IDE

I. Objetivo: El alumno entenderá la forma de trabajo de un IDE y lo manejará para construir una pequeña aplicación si incluir archivos y Bases de Datos.

II. Equipo necesario:

- Computadora con MS-Windows.
- Una herramienta IDE, preferentemente Delphi de Borland.
- Libros del lenguaje, preferentemente Object Pascal, y la herramienta IDE que lo manipula.

III. Material de apoyo:

- Ejemplo con las unidades que constituyen el la aplicación.
- Otros ejemplos de programas que vienen con el IDE

IV. Procedimiento:

1. Abrir el IDE dando clic en su icono en el escritorio o buscándolo en *Todos los programas* del menú *Inicio* de MS-Windows
2. Escoger crear un nuevo proyecto que incluya una unidad de tipo Forma: VCL Project (Projet>New>VCL Project).
3. Agregar los componentes necesarios como: campos de edición, botones, menús, para crear una ventana como en la Figura 1. Oprimir la tecla F12 para ver el código, resultará parecido al de la Figura 2 (no lo cambie).
4. Modificar los atributos de cada componente utilizando la barra de atributos que aparece a la izquierda, después de dar un clic sobre la componente deseada.
    - Escribir en Caption de Button1 “Verificar”
    - Escribir en Caption de Button2 “Calcular”
5. Dar doble clic en el botón Verificar e incluir el código de la Figura 3
6. Dar doble clic en el botón Calcula e incluir su funcionalidad asociada como en la Figura 4.
7. Crear una nueva unidad: Project>New>Unit
8. Usar el código de la Figura 5 e incluirlo en la unidad recién creada.

V. Resultados esperados:

Programa pequeño sin uso de archivos que calculará el impuesto (ISR) de una cantidad dada.

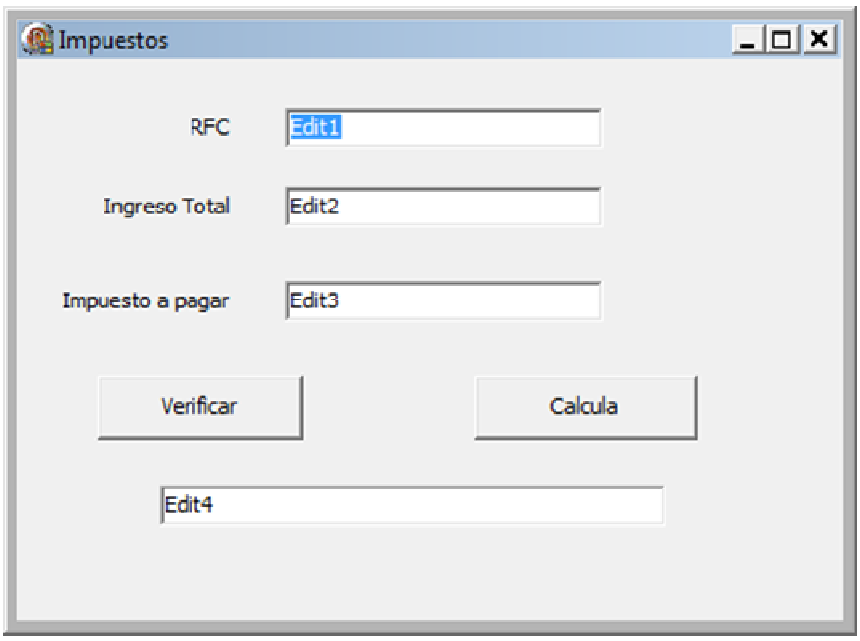

Figura 1. Ventana para el Programa sobre el cálculo de ISR

```
unit IUImpuestos;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm2 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Button1: TButton;
    Button2: TButton;
    Edit4: TEdit;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation
uses ucalcula;

{$R *.dfm}
// AQUI VA EL CÓDIGO DE LOS DOS BOTONES (FIGURAS 2-3 Y 2-4)

end.
```

Figura 2. Código de la Ventana que inserta Delphi automáticamente

```
procedure TForm2.Button1Click(Sender: TObject);
  var calcimp:pagoimpue; resp:integer;
begin
     calcimp := pagoimpue.create;
     Edit4.Text := 'Validando ...';
     resp:=calcimp.vale(Edit1.Text);
     if (resp=0) then   Edit4.Text := 'RFC parece válido'
               else
               if (resp=1) then Edit4.Text := 'RFC inválido; faltan o sobran caracteres'
                else
               if (resp=2) then Edit4.Text := 'RFC inválido; revise letras iniciales'
                else
               if (resp=3) then Edit4.Text := 'RFC inválido; revise fecha de nacimiento';
end;
```

Figura 3. Código del Botón 2 (Verificar)

```
procedure TForm2.Button2Click(Sender: TObject);
var calcimp:pagoimpue; impudebido: single;
begin
  calcimp := pagoimpue.create;
  if (StrToFloat(Edit2.Text)<0) then
    Edit4.Text := 'El total de ingresos debe ser positivo'
  else
  begin
    impudebido :=  calcimp.impuesto(StrToFloat(Edit2.Text));
    Edit3.Text := Format('%f',[impudebido] );
    Edit4.Text := ' ';
  end;
end;
```

Figura 4. Código asociado al Botóan 1 (Calcula)

```
unit ucalcula;

interface
   type pagoimpue = class
     function vale(rfc:String):integer;
     function impuesto(tot:single):single;
     constructor Create;
   end;
implementation
uses sysutils;
     var Tabla: array[0..7,0..3] of single;
     //     Constructor: prepara la tabla de impuestos
     constructor pagoimpue.Create;
     begin
       //cargar tabla  2009
       Tabla[0,0] := 0.01; Tabla[0,1] := 5952.84 ; Tabla[0,2] := 0.0; Tabla[0,3] := 0.0192;
       Tabla[1,0] := 5952.85; Tabla[1,1] := 50524.92; Tabla[1,2] := 114.24; Tabla[1,3] := 0.0640;
       Tabla[2,0] := 50524.93; Tabla[2,1] := 88793.04; Tabla[2,2] := 2966.76; Tabla[2,3] := 0.1088;
       Tabla[3,0] := 88793.05; Tabla[3,1] := 103218.00; Tabla[3,2] := 7130.88; Tabla[3,3] := 0.1600;
       Tabla[4,0] := 103218.01; Tabla[4,1] := 123580.20; Tabla[4,2] := 9438.60; Tabla[4,3] := 0.1792;
       Tabla[5,0] := 123580.21; Tabla[5,1] := 249243.48; Tabla[5,2] := 13087.44; Tabla[5,3] := 0.1994;
       Tabla[6,0] := 249243.49; Tabla[6,1] := 392841.96; Tabla[6,2] := 38139.60; Tabla[6,3] := 0.2195;
       Tabla[7,0] := 392841.97; Tabla[7,1] := 0.0; Tabla[7,2] := 69662.40; Tabla[7,3] := 0.2800;
     end;
     //   Verificación de rfc
     function pagoimpue.vale(rfc:String ):integer;
     var  err, ix:integer; ll:integer; lim :integer; rfcu:string;
     begin
       err := 0;
       rfcu := UpperCase(rfc);
       ll :=  Length(rfcu);
       if (ll<12) or (ll>13) then
          err :=1;
      if (ll=12) then  lim := 3 else lim := 4;
      for ix := 1 to lim do
          if (not (rfcu[ix] in ['A'..'Z'])) then
             err := 2;
      for ix := lim+1 to lim+6 do
        if (not (rfcu[ix] in ['0'..'9'])) then
           err :=3;
      if (err <3) then
```

```
     begin
       if (rfcu[lim+3] in ['2'..'9']) or (rfcu[lim+5] in ['4'..'9']) then
           err := 3;
      end;
       vale :=err;
    end;
     //
    //   Cálculo de impuestos
    //
    function pagoimpue.impuesto(tot:single):single;
    var iz:integer; ix:integer;
    begin
       //calculo
      ix := 7;
       for iz:= 6 downto 0  do
         if (tabla[iz,1]>=tot) then
             ix := iz;
       impuesto := Tabla[ix,2] + (tot - Tabla[ix,0])*Tabla[ix,3];
      end;
end.
```

Figura 5. Código de las Funciones para Cálculo del Impuesto

Práctica 2. Delphi y Bases de Datos

I. Objetivo. En esta práctica se desarrollará una aplicación que permite consultar una base de datos que incluye imágenes, usando elementos predeterminados de Delphi y una base de datos proporcionada por Borland.

II. Equipo necesario:

- Computadora con MS-Windows.
- Una herramienta IDE, preferentemente Delphi de Borland, versión de 7, 8, Studio o Turbo.
- Libros del lenguaje, preferentemente Object Pascal, y la herramienta IDE que lo manipula.

III. Material de apoyo:

Opcionalmente, una imagen en formato BMP.

IV. Procedimiento: Se procederá como sigue:

a) Se crea un proyecto VCL (vea práctica anterior); se le hacen adaptaciones al título de la forma en el Object Inspector (ver ventana izquierda inferior).
b) Se agrega un componente *menú general* de la Tool Palette (ver ventana inferior derecha) y dando doble clic en el componente agregar dos opciones del mismo nivel: *Tabla* y *Cerrar*
c) En la opción Tabla se agregan dos subopciones: Actualizar y Buscar.
d) Desde el SO cree una carpeta con el nombre del proyecto y en ésta salve todas sus archivos usando la opción Save All del menú File de Delphi vaya nombrando los archivos como sigue:
    a. La forma con el nombre *MenúInicial* y
    b. El Proyecto con el nombre *Practica2.*
e) Se crea una nueva unidad del tipo Data Module File>New -> Other -> Data Module. Se agregan la componentes TTable y se actuslizan sus atributos como sigue:
    a. TTable se conecta, usando el Object Inspector y el atributo Database>DataBasename a una base de datos llamada DBDEMOS localizada en: C:\Archivosdeprogramas\ArchivosComunes\BorlandShared\Data
    b. Se escoge la tabla: Animals.dbf poniendo este nombre en el atributo Tablename;
    c. Se usa como campo índice NAME y
    d. Se inicia el atributo *Active* en true (ver Object Inspector).
f) También se agrega al DataModule la componente DataSource y se actualizan sus atributos:
    a. DataSource, se conecta a Table1 usando el atributo DataSet.
    b. Se salva el DataModule con el nombre *Datos*.
g) Se crea una forma (File>new -> Form) que se empleará de ventana para la consulta. Se le agrega un componente DBNavigator, un DBGrid y un DBImage. En Uses (dentro del código de la unidad) se agrega el nombre

del módulo de datos (*Datos*) y se conectan los tres elementos al Data Access usando el Object Inspector. Se salva con el nombre *Consulta*.

h) De la misma manera se crea una forma de ventana para búsqueda, pero a ésta se le agregan: un campo Edit, un botón y un campo DBImage. En Uses se agrega el nombre del módulo de datos y se conecta el DBImage con el acceso. Se salva con el nombre *Busca*.

i) En el código de la ventana principal (Form1) se agregan, en *Uses*, los nombres de las dos unidades (*Consulta* y *Busca*). En los menús correspondientes se generan eventos que hacen visibles a las ventanas (cada una por separado). En el menú cerrar se cierra la forma. Se salva.

j) En la ventana de búsqueda se crea un evento asociado al botón que permita leer un nombre del campo Edit y luego lo busque en la base de datos con la siguiente instrucción y se salva.

if not (Datos.DataModule1.Table1.FindKey([Edit1.Text]) then Edit1.Text:= 'No se encuentra ese animal');

k) Se ejecuta y prueba la aplicación (puede probarse por partes, al terminar c, e, f, g).

l) Si tiene una imagen a mano, puede agregar un registro en la ventana de consulta y agregar la imagen en la ventana DBImage. Simplemente ábrala con Paint de copiar (control+C) y pegue (control+V).

V. Resultados esperados:

Un programa ejecutable hecho en Object Pascal y que muestra el uso de diversas componentes como: un menú con las opciones de actualizar archivo, buscar registro, cerrar; un navegador de una tabla (archivo), una rejilla. Además se verá como se enlazaron las unidades con la clases tipo FORM.

Práctica 3. Utilización de una Herramienta CASE

I. Objetivo:

Conocer el manejo de una herramienta CASE mediante la introducción de modelos de análisis y diseño de un sistema de software de ejemplo.

II. Equipo necesario:

- Computadora con MS-Windows.
- Una herramienta CASE, preferentemente MS-Visio.

III. Material de apoyo:

- Diagramas que representan los modelos de análisis y diseño de un sistema de software, mismos que proporcionará el profesor.

IV. Procedimiento:

1. Buscar la herramienta MS-Visio en el menú Inicio y de allí en Microsoft Office y abrirlo
2. Para iniciar cada uno de los diagramas que le dará el profesor haga:
    - Para el DFD (Diagrama de Flujo de Datos, escoger: Archivo>New>Diagrama de flujo>Diagrama de flujo de datos.
    - Para el DE (Diagrama de Estructura), escoger: Archivo>New>Software y Base de Datos>Estructura de Programas
    - Para el DE (Diagrama de Estructura), escoger: Archivo>New>Software y Base de Datos>Modelo de Bases de Datos
3. Una vez elegido el tipo de diagrama que realizará, se deben utilizar sólo los iconos que aparecen en la ventana *Formas* (a la izquierda de la pantalla), posicionándose con el ratón sobre el símbolo o icono deseado y arrastrarlo a la ventana central donde aparece una página cuadrícula donde se conformará el dibujo.
4. Al finalizar cada diagrama de la orden de guardar como en cualquier herramienta de MS-Office.

V. Resultados esperados:

Una serie de diagramas introducidos a la herramienta CASE utilizada y el conocimiento del empleo de la misma.

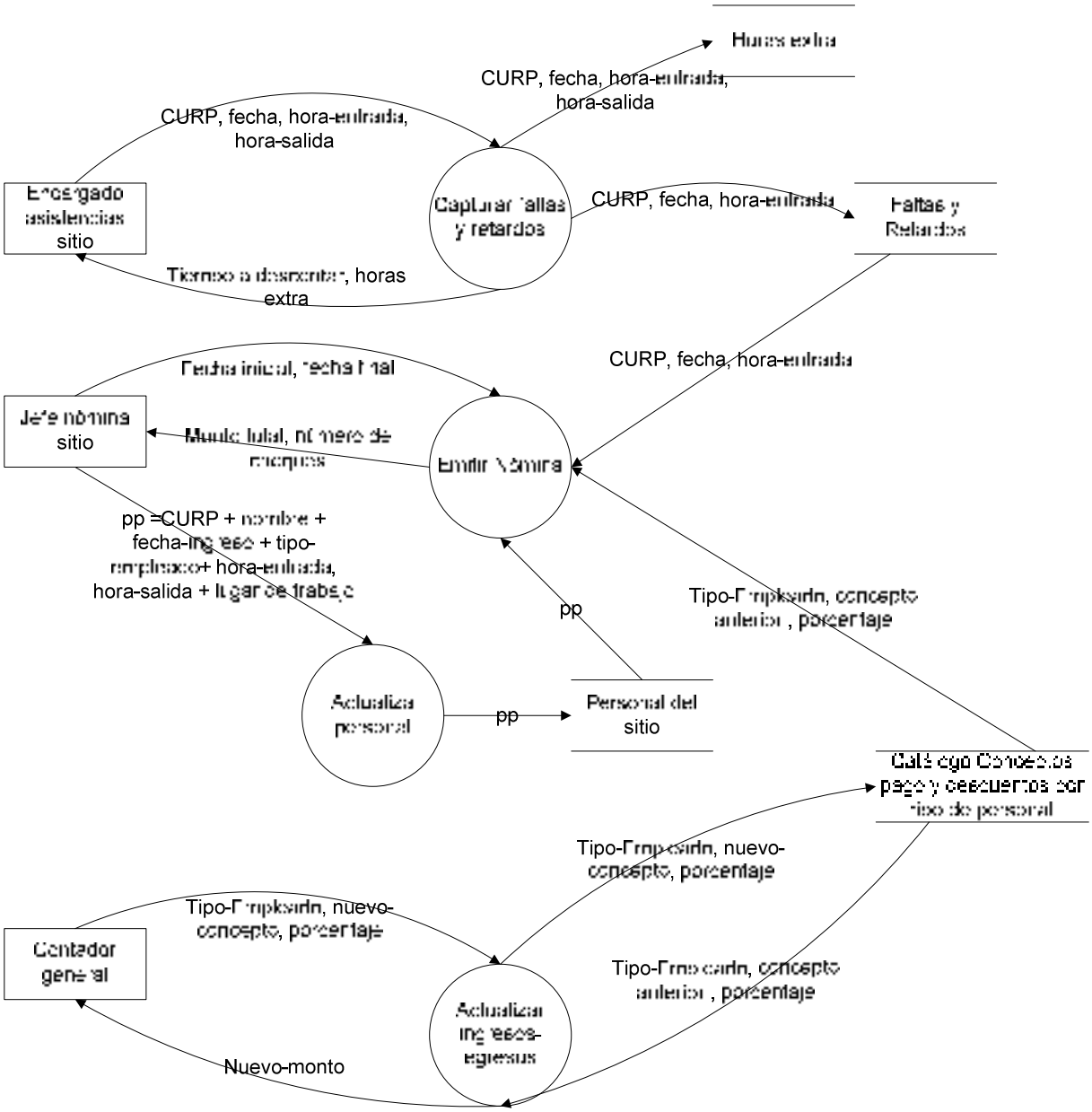

Figura 6. Ejemplo de Diagrama de Flujo de Datos

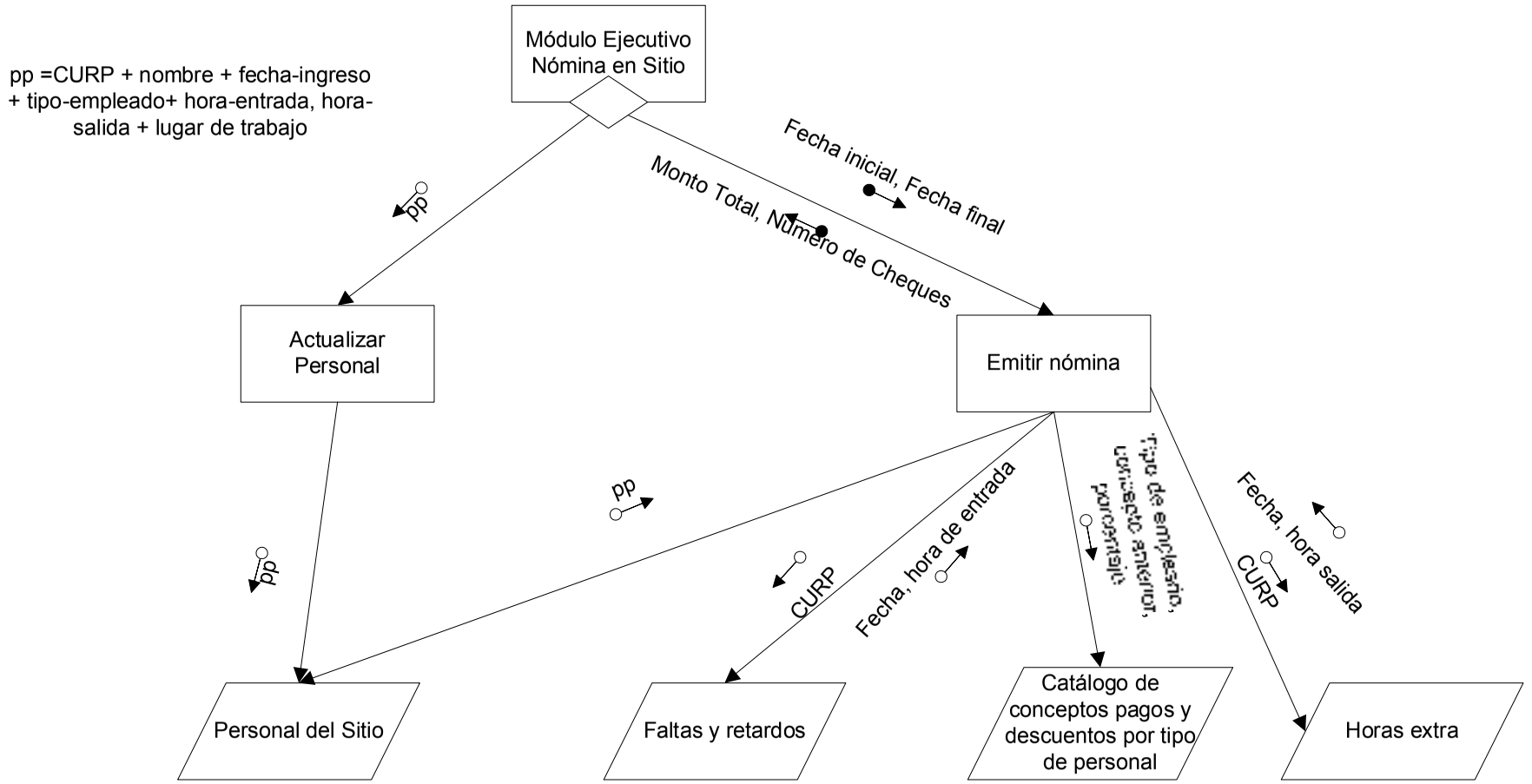

Figura 7. Ejemplo de Diagrama de Estructura

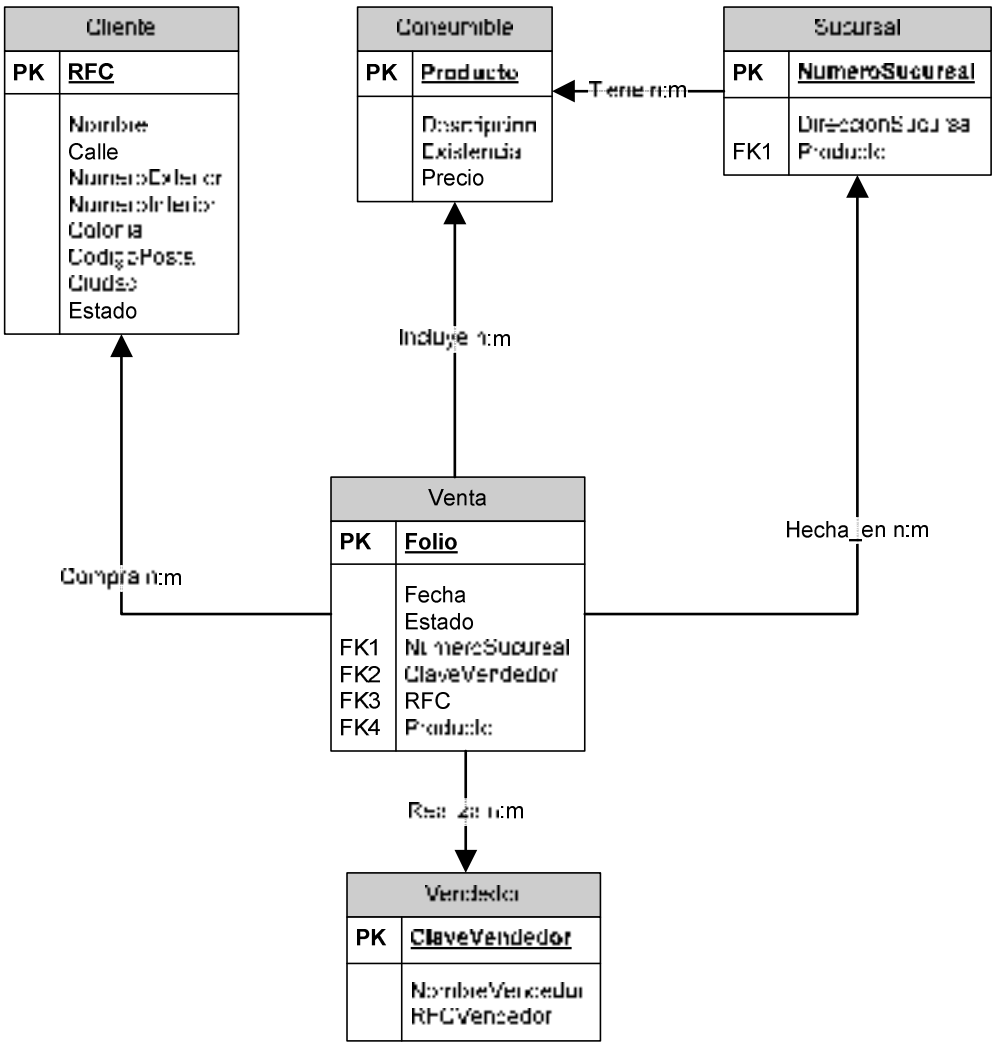

Figura 8. Modelo Entidad – Relación del Sistema de venta de Consumibles

Práctica 4. Creación del Modelo Ambiental

I. Objetivo:
Crear y poner en un archivo de documento el *modelo ambiental* de un sistema de software.

II. Equipo necesario:
- Una PC con el SO MS-Windows
- MS-Office, incluidos Word y Visio

III. Material de apoyo:
- La explicación oral, por parte del profesor, de un problema para ser resuelto mediante un sistema automatizado
- El libro de la metodología estudiada [Yourdon, E. (1993)]
- Las diapositivas sobre la metodología que se encuentran en la página www.uv.mx/asumano
- Un cuaderno para apuntes.
- Ejemplos de otros modelos ambientales dados por el profesor

IV. Procedimiento:
NOTA: Esta práctica puede llevar más de una sesión y éstas pueden ser en el salón de clase o en el Centro de Cómputo.

1. En su cuaderno realice los tres elementos que conforman el modelo ambiental
2. Revise que sean considerados correctamente los tres elementos
    a. **Para la declaración de propósitos**. Que en un párrafo se listen todas las funciones que se desean del nuevo software, se entienda quienes serán los clientes y usuarios del nuevo sistema y cual sería el papel de cada uno dentro de éste. Además debe incluir los posibles sistemas externos con que interactuará el mismo.
    b. **Para el Diagrama de Flujo de Datos.** Dibujar una sola burbuja con el nombre del nuevo sistema. A partir de la burbuja conectar flujos (flechas) que incluyan paquetes de datos y que entren o salgan a terminadores (usuarios o sistemas externos) o archivos (a este nivel como bases de datos que incluyan gran parte de la información que se manejará).
    c. **Para la lista de acontecimientos.** Cada enunciado de la lista debe tener la forma: Usuario + función + objeto directo + complemento.
3. Abra MS-Visio, escoja la opción de DFD y dibuje el diagrama de contexto.
4. Usando el modelo ambiental que se hizo previamente en el salón de clase y el dibujo del DFD de contexto hecho en Visio:
    a. Abra MS-Word y escriba la declaración de propósitos,
    b. Copie de MS-Visio el DFD de contexto y péguelo en MS-Word usando Edición>pegado especial>Imagen (metarchivo mejorado) y luego, de ser necesario, dar Formato>Imagen>Diseño>en línea con el texto, con lo que se asegura que la imagen no cambié de posición al modificar el documento.

c. Finalmente, agregue en Word a lista de acontecimientos.

V. Resultados esperados:
Un documento Word con los elementos que conforman el modelo ambiental: declaración de propósitos, DFD de contexto y Lista de Acontecimientos.

Ejemplos de modelo ambiental

Ejemplo 1 Sistema Money

Declaración de propósitos:

- Se requiere hacer un sistema que ayude a las personas a llevar un control de sus gastos e ingresos y que, además, le calcule que ahorro le quedó y en donde le conviene invertir éste con base en los tipos de inversión que reporta el Banco de México a través de su Sistema Bancario. La forma en que el usuario del sistema Money utilizará la opción de inversión será utilizando Internet, mientras que el registro de ingresos y egresos se realizaría de manera local.

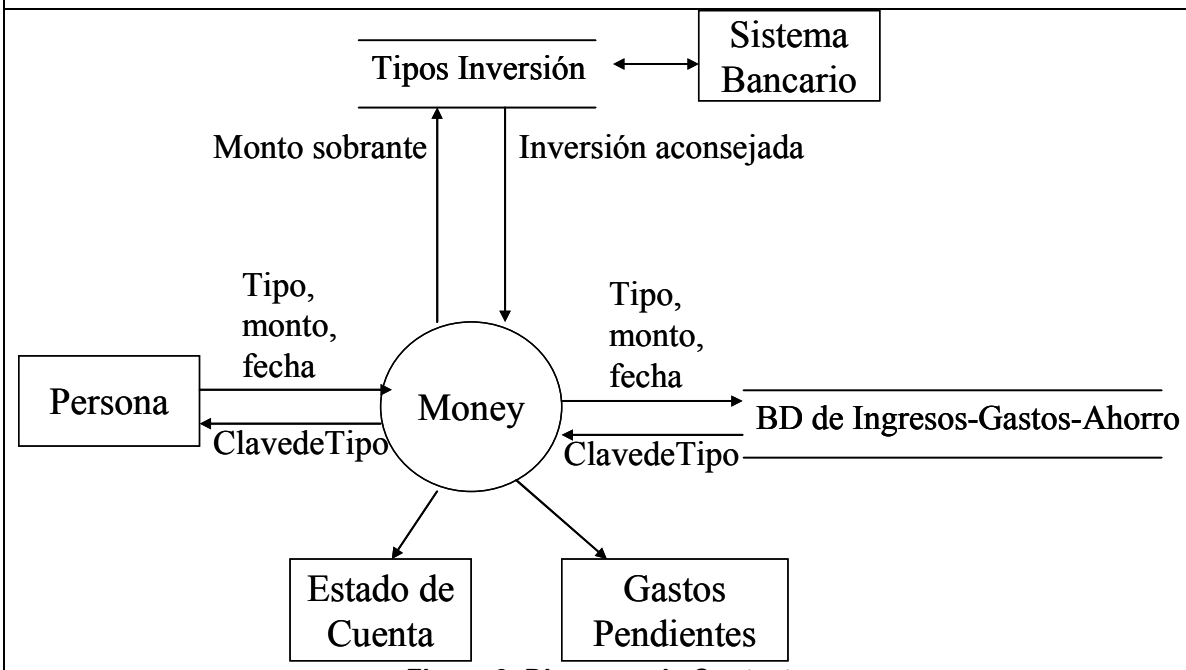

Figura 9. Diagrama de Contexto

Lista de Acontecimientos:

1. Persona registra ingreso
2. Persona registra egresos o gastos.
3. Persona actualiza tipos de ingresos y gastos.
4. Persona solicita cálculo de cantidad disponible para ahorro.
5. Persona solicita tipo de inversión con base en su ahorro del mes.

Ejemplo 2. Sistema de cajero Automático

Declaración de propósitos:
Se quiere realizar un sistema que permita al *Banco de la Ilusión,* prestar servicios de consulta de saldo y retiro de efectivo de sus cajeros automáticos (ATM). Éstos mismos servicios los deberá brindar a usuarios de otros bancos con un cargo adicional, mismo que se le cobrará de manera inmediata de la cuenta que se esté manejando.

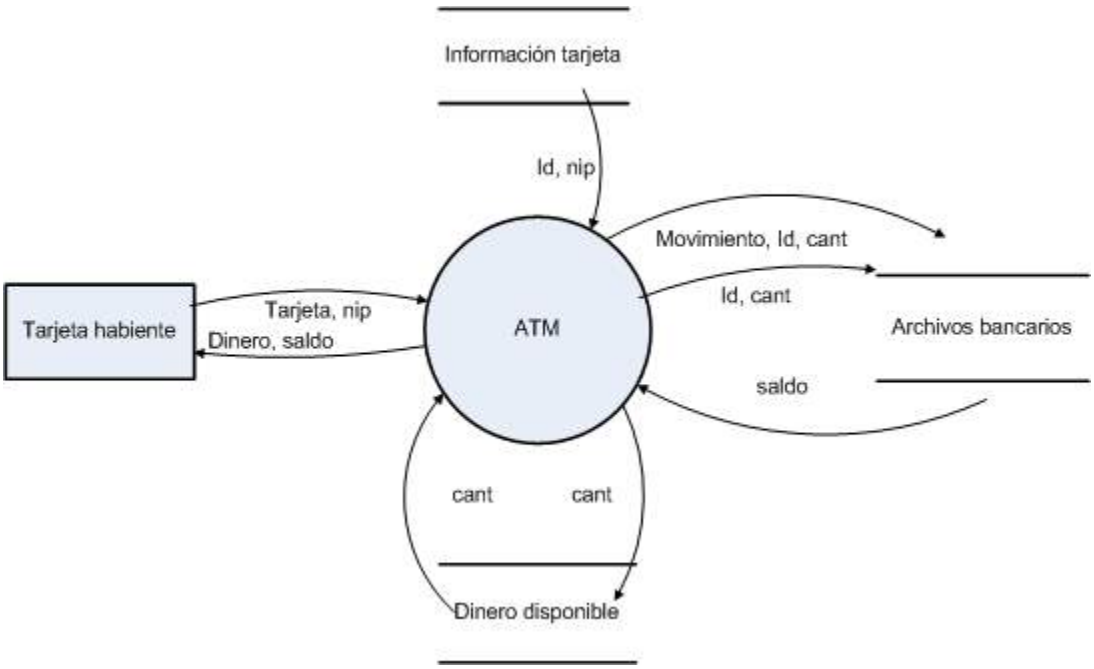

Figura 10. Diagrama de Contexto del ATM

Lista de Acontecimientos:

1. Usuario ingresa tarjeta bancaria
2. Usuario se autentifica tecleando su NIP
3. Usuario solicita saldo
4. Usuario solicita retiro
5. Banco obtiene saldo
6. Banco autoriza movimiento
7. Banco registra movimiento

Ejemplo 3. Diccionario del Español Mexicano

Tabla 1

Declaración de propósitos:
La Asociación Internacional de Vuelos Comerciales (AIVC) ha decidido realizar un sistema de apartado y venta de boletos por Internet (SAVBI). Para ello, se permite que un pasajero elija la fecha, hora y destino deseado (p1) y lo envíe vía Internet a un Agente de Viajes (AV) de su país que esté registrado en la AIVC y que debe tener un servidor automático de recepción de peticiones, mismo que a su vez manda la petición del usuario al servidor de la AIVC que revisa si existen vuelos y lugares disponibles de acuerdo a la petición del pasajero y los envía al servidor de la AV. Una vez que el pasajero está conforme con el vuelo y precio, éste debe enviar lo contenido en p2 que es: número de boletos deseados, datos de su tarjeta de crédito (tipo - VISA o MasterCard -, fecha de caducidad y número; después de ello el pasajero recibirá su confirmación mediante un boleto electrónico que el pasajero debe imprimir para llevar el día de su vuelo al aeropuerto donde le será canjeado por pases de abordar. El SAVBI debe tener actualizado los vuelos y su información asociada todo el tiempo y por ello las líneas aéreas deben estarlo actualizando.

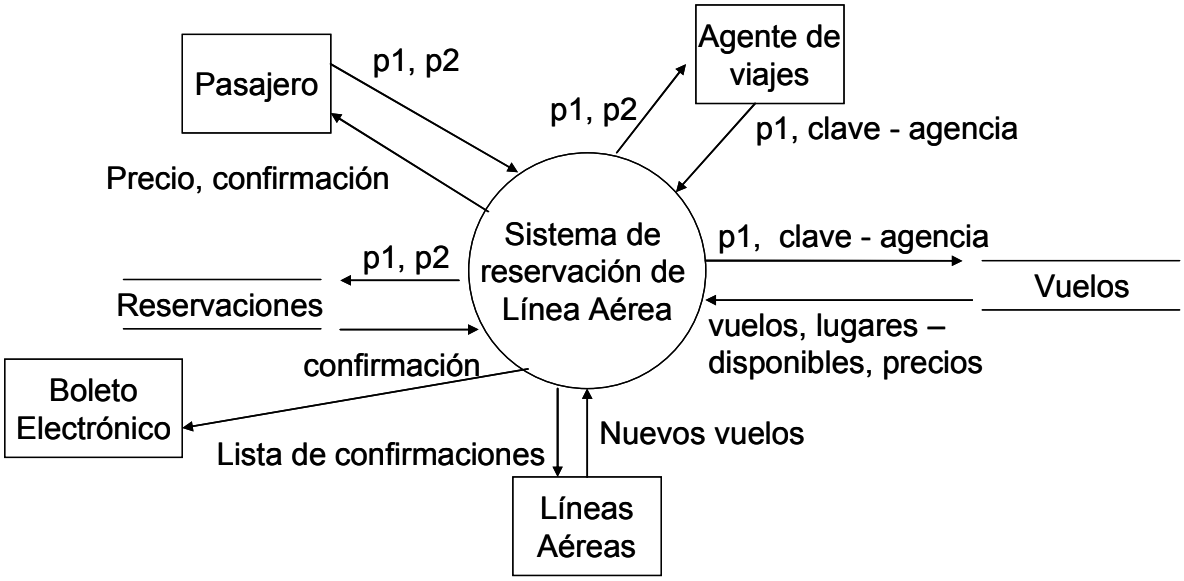

Figura 11. Diagrama de Contexto

Lista de Acontecimientos:

1. Pasajero solicita viaje con datos necesarios
2. Agencia de viajes revisa disponibilidad
3. Líneas aéreas responden disponibilidad del viaje
4. Pasajero manda datos para pagar boleto
5. Pasajero imprime boleto
6. Agencia de viaje actualiza disponibilidad
7. Líneas aéreas actualizan nuevos vuelos.

Práctica 5. Creación del Modelo de Comportamiento

I. Objetivo:
Crear y poner en un archivo de documento el *modelo de comportamiento* de un sistema de software.

II. Equipo necesario:
- Una PC con el SO MS-Windows
- MS-Office, incluidos Word y Visio

III. Material de apoyo:
- Un problema para ser resuelto mediante un sistema automatizado
- El modelo ambiental del sistema
- El libro de la metodología estudiada [Yourdon, E. (1993)]
- Las diapositivas sobre la metodología que se encuentran en la página www.uv.mx/personal/asumano
- El cuaderno para apuntes que sobre los modelos han realizado los alumnos.
- Dos ejemplos completos que permitirán ver como se desarrollarán los sistemas de los alumnos.

IV. Procedimiento:
NOTA: Esta práctica puede llevar más de una sesión y éstas pueden ser en el salón de clase o en el Centro de Cómputo.
1. En su cuaderno realice los elementos necesarios para su modelo de comportamiento
    a. DFD nivel cero que será generado a partir de la lista de acontecimientos descrita en el modelo ambiental.
    b. El DFD nivel uno
    c. DER asociado al DFD de nivel uno.
    d. DTE del sistema
    e. Diccionario de Datos (DD).
2. Revise que sean considerados los puntos de la Tabla 2-2.
3. Usando MS-Visio dibuje los DFD, DER y DTE necesarios
4. Abra el documento Word que contiene el modelo ambiental y agregue los diagramas del modelo de comportamiento.
5. Para cada diagrama escriba una explicación que permita al usuario entender lo que se está planteando en ellos.
6. Realice o actualice el DD.

Tabla 1. Elementos a revisar en los diagramas del Modelo de Comportamiento

Elemento del MC	Características a revisar
Burbujas del DFD nivel cero	Deben ser tantas burbujas como enunciados de la lista de acontecimientos. Cada burbuja se numerará a partir de uno se nombrará describiendo la respuesta que el sistema debe dar al acontecimiento asociado
Burbujas del DFD nivel 1	Deben numerarse de acuerdo a la lista de acontecimientos seguidos de punto y otro número secuencial. Ejemplo: para el acontecimiento 1, las burbujas tendrá los números 1.1, 1.2, 1.3, etc.
Diagrama Entidad Relación	Sólo debe haber entidades que aparecen en el DFD nivel 1. Debe marcarse la cardinalidad y el nombre de las relaciones que haya entre entidades.
Flujos de datos	Cada flecha que entra o sale de una burbuja debe tener encima los datos que transporta (en paquete o desglosados)
Almacenes	Cada almacén se nombrará con una frase nominal que exprese lo que almacenan y que coincida con alguna entidad o relación del DER
DTE	Tiene estado inicial y final, se han definido todos los estados, se puede llegar y salir a o de todos los estados, cada estado responde al sistema adecuadamente a todas las condiciones posibles.

V. Resultados esperados:

Un documento Word con los elementos que conforman los modelos: Ambiental corregido y de Comportamiento.

Ejemplos de Modelos de comportamiento:

Sistema Money

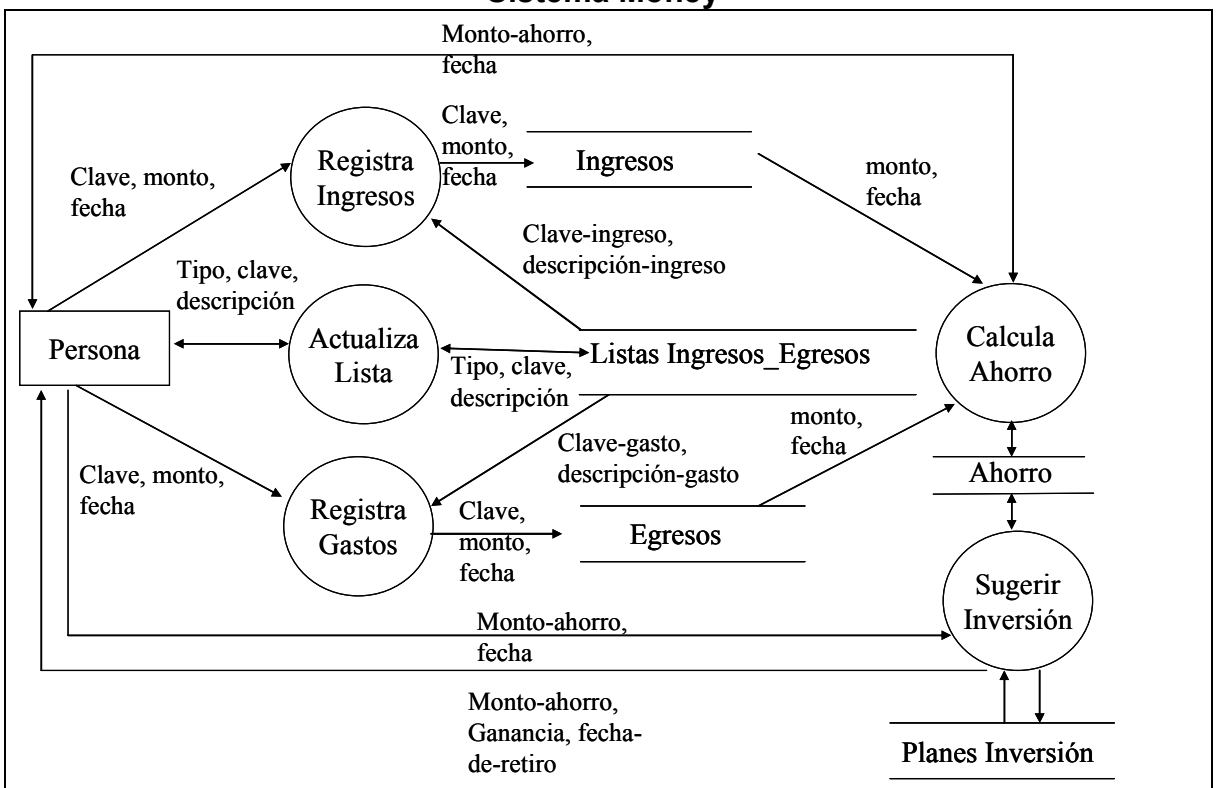

Figura 12. Diagrama de Comportamiento. Nivel 0

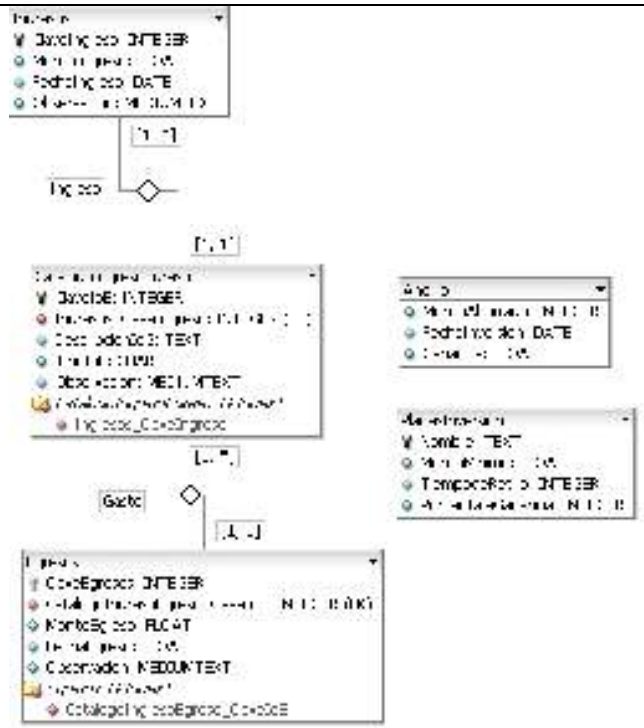

Figura 13. Diagrama Entidad - Relación

Sistema Cajero Automático

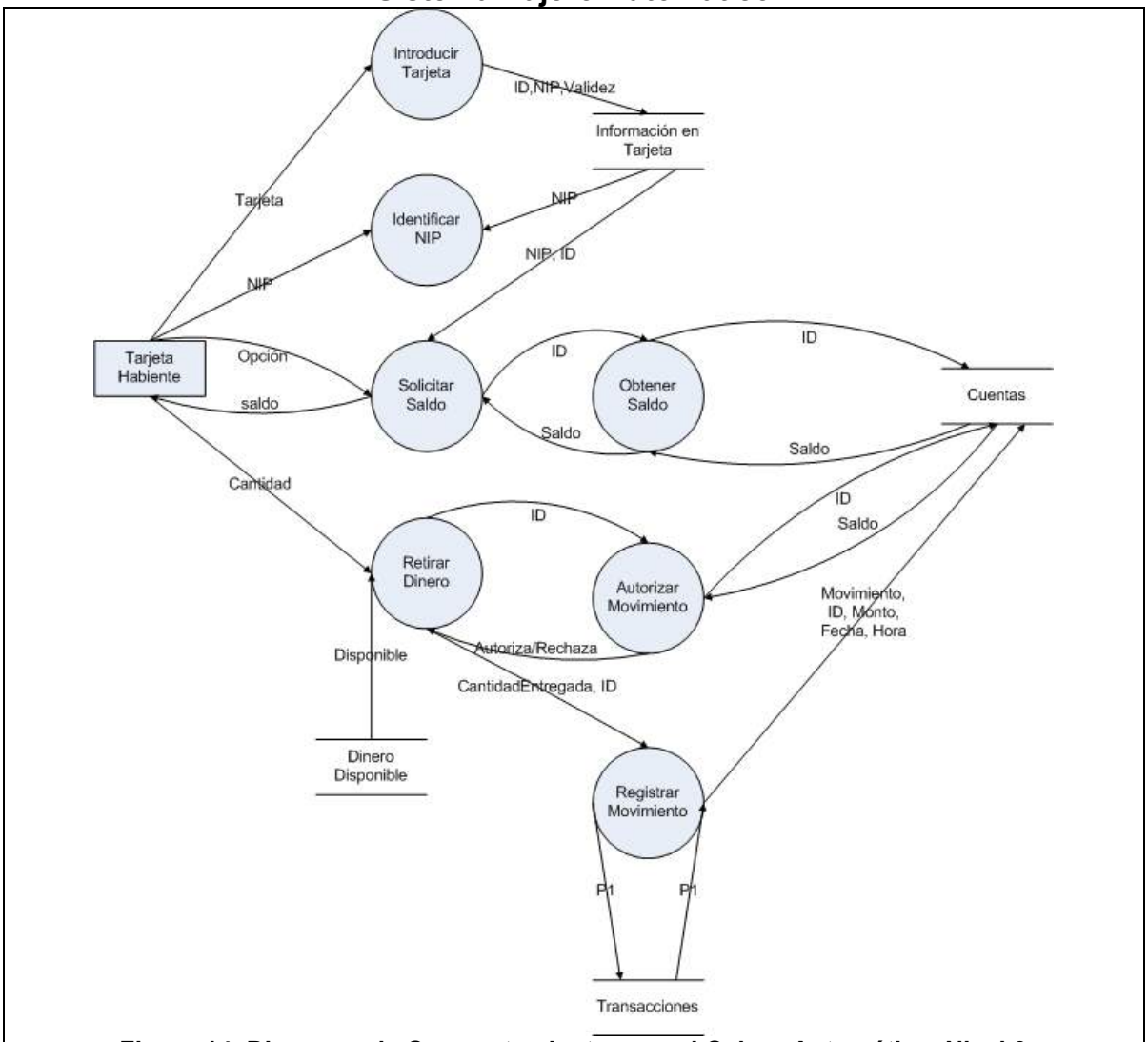

Figura 14. Diagrama de Comportamiento para el Cajero Automático. Nivel 0

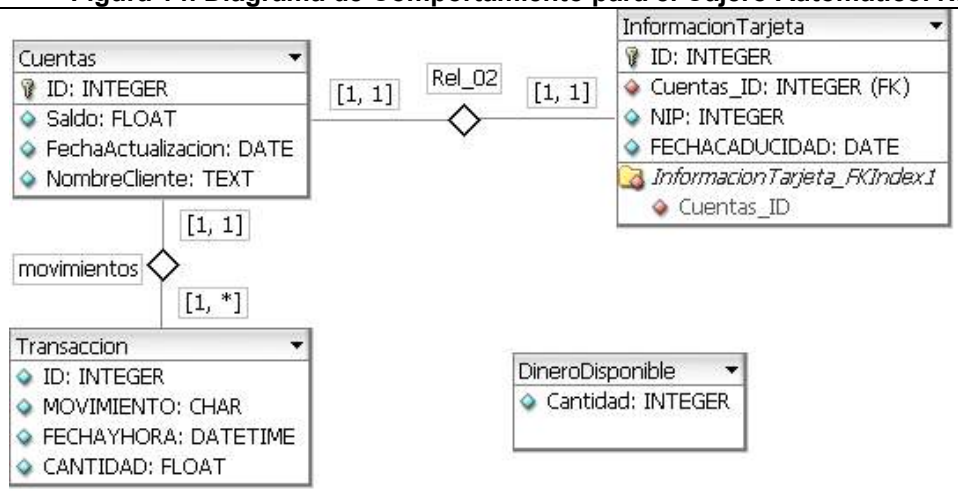

Figura 15. Diagrama Entidad - Relación del cajero Automático

SISTEMA HOGAR SEGURO (Ejemplo tomado de [4])

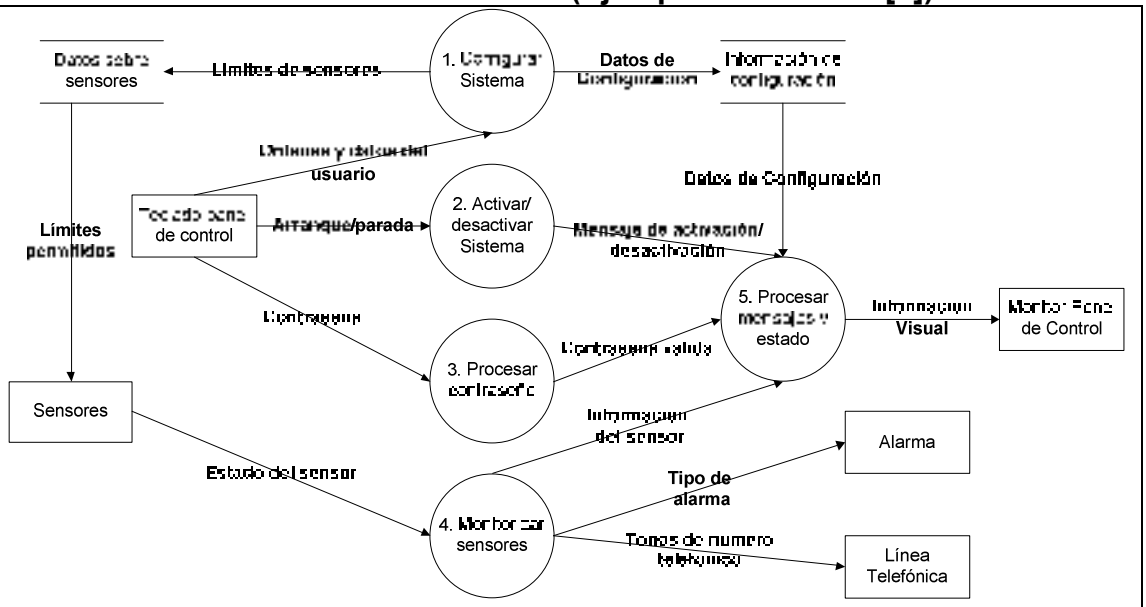

Figura 16. Diagrama de Nivel 0

Práctica 6. Creación del Modelo de Implementación del Usuario

I. Objetivo:
Realizar usando un archivo de documento el *modelo de implementación del usuario* de un sistema de software en versión de Manual Preliminar del Usuario.

II. Equipo necesario:
- Una PC con el SO MS-Windows
- Conexión a Internet
- MS-Office, incluidos Word
- IDE Delphi (cualquier versión)

III. Material de apoyo:
- El modelo de comportamiento del software que se está desarrollando.
- El libro de la metodología estudiada [Yourdon, E. (1993)]
- Las diapositivas sobre la metodología que se encuentran en la página www.uv.mx/personal/asumano
- El cuaderno para apuntes que sobre los modelos han realizado los alumnos.
- Sistemas de software semejantes que existen ya sea proporcionados por el profesor o en Internet.

IV. Procedimiento:
NOTA: Esta práctica puede llevar más de una sesión y éstas pueden ser en el salón de clase o en el Centro de Cómputo.

1. En su cuaderno realice los elementos necesarios para su modelo de comportamiento
    a. Ventanas que representarán la forma en que el sistema interactuará con el usuario. Utilice las facilidades que ofrece una GUI (Graphic User Interface) hecha utilizando Delphi.
    b. Para cada ventana poner su explicación de uso.
    c. Lista de posibles fallos a los que se puede ver expuesto el sistema. Ejemplos:
        i. Se pierde la conexión de la red.
        ii. No están la Base de Datos
        iii. Se desconecta el suministro eléctrico
        iv. El hardware falla
        v. El usuario introduce mal los datos
2. Revise que su Manual Preliminar del Usuario contenga:
    a. Introducción que incluya: Cuales son las características de su software (suponga que ya lo hizo) y que lo hace especial para que las personas decidan utilizarlo en lugar de otro.
    b. Todas las funcionalidades que describió en su modelo de comportamiento en el DFD nivel 1.
    c. Aquellas actividades manuales que deben hacerse asociadas al sistema de software que está realizando.
    d. Que en cada ventana se definan los datos que allí aparecen según el DD.
    e. Su lista de fallos posibles y la forma en que el usuario debe afrontarlos.

Práctica 7. Cálculo de las Métricas de Análisis

I. Objetivo:

Calcular dos métricas sobre los modelos de análisis con el fin de tener elementos sobre tamaño y características de calidad de éstos.

II. Equipo necesario:

- Lápiz, papel y calculadora.

III. Material de apoyo:

- El modelo de comportamiento del software que se está desarrollando.
- Las diapositivas sobre el cálculo de la métricas que se encuentran en la página www.uv.mx/personal/asumano
- Apuntes sobre Puntos de Función, sección 2.7 del libro de Áncora [Sumano, A. (2006)].
- Ejemplos sobre el cálculo de las métricas: Bang y Puntos de Función que se adjuntan dos ejemplos de sistemas de software con el cálculo de las métricas.
- Archivo Excel de apoyo al cálculo de Puntos de Función

IV. Procedimiento:

1. Para la métrica Bang:
    a. Usando el DFD nivel 1, calcule la primitiva funcional PFu
    b. Usando el DER, calcule la primitiva de relaciones RE
    c. Saque el coeficiente RE / PFu
    d. Si su aplicación es:
        - Dominio de datos y el coeficiente RE / PFu es menor de 1.5, revise y modifique, en su casa, sus DER y DFD.
        - Dominio de función y el coeficiente RE / PFu es mayor de 0.6, revise y modifique, en su casa, sus DER y DFD.
        - Híbrida y el coeficiente RE / PFu es mayor de 1.4 o menor de 0.8, revise y modifique, en su casa, sus DER y DFD.
    e. Volver a calcular coeficiente RE / PFu hasta que se ajuste a la cantidad adecuada.
2. Para la métrica de PF:
    a. Haga una lista de indicadores de datos (ALI, AIE) y de transacciones (EE, SE y CE)
    b. Ingrese los datos pedidos en la hoja Excel para que ésta calcule la complejidad de cada indicador y los punto de función sin ajustar (T).
    c. Haga la lista de catorce modificadores (Estimadores) y elija los que tendrán influencia en su sistema. Redacte dos renglones por cada modificador donde se justifique por qué lo eligió.
    d. Asigne el grado de influencia (de cero a cinco) de cada modificador escogido usando las descripciones que vienen en las notas.
    e. Sume los grados de influencia y llame M a la suma.

f. Observe los puntos de función que se calcularon en la hoja Excel utilizando la fórmula.
g. Haga una interpretación sobre su resultado sobre: complejidad y posible costo.

V. Resultado esperado:

- Un coeficiente RE / PFu que le permitirá darse cuenta de la validez del modelo de comportamiento.
- El cálculo de PF que le servirá para estimar esfuerzo y como medida histórica que **permita entender nuevos sistemas.**

Práctica 8. Planeación de Pruebas de Aceptación de Sistema

I. Objetivo:

Planificar cómo el cliente y usuarios podrán verificar que el software que se les está construyendo hará lo que ellos desean, para lo cual tendrán una serie de casos de prueba que incluirán diversas formas de verificar el comportamiento del software desde el normal o esperado hasta los posibles fallos que se pudieran cometer y a los cuales el software debe responder con ayudas y prevención de defectos.

NOTA: Está práctica no se terminará en un par de horas ni en un día, trate de avanzar con una función del software en construcción y avance de función en función. Como lo marca un modelado iterativo e incremental.

II. Equipo necesario:

- Computadora con MS-Office.

III. Material de apoyo:

- Los modelos de análisis del software que se está desarrollando, incluyendo el Diccionario de Datos y Manual Preliminar del Usuario.
- Los apuntes sobre prueba de software, Capítulos 1 y 2, que se encuentran en la página www.uv.mx/personal/jfernandez. Observe bien los ejemplos.
- El cuaderno donde empezó a establecer sus dominios y particiones y a realizar sus casos de prueba.

IV. Procedimiento:

1. Definir dominios de cada pieza elemental del DD.
    a. Use la Tabla del DD (ver el ejemplo de Money), agregue dos columnas, llámelas Dominio y Particiones y proceda como sigue:
        i. En la columna Dominio especifique, usando notación de conjuntos, rango o con lenguaje natural, los dominios de las piezas elementales de información de flujo de datos y en los almacenes de datos, mismas que deben aparecer en el DD.
        ii. En la columna Particiones, describa las clases de equivalencia dictadas por el comportamiento de la función en que interviene cada variable de entrada.
2. Para cada funcionalidad (acontecimiento)
    a. Establecer los valores de entrada con los que se establecerán los casos de prueba de cada función del software en desarrollo. Identifique las entradas y salidas; luego escoja:
        i. Un valor de cada clase de equivalencia para cada variable de entrada.
        ii. Escoja dos valores que sean los extremos cada clase de equivalencia.
3. Identifique las salidas correspondientes y escriba los casos de prueba, uno por cada combinación de los valores escogidos en las diversas variables. No varíe todos los valores de las variables a la vez.

V. Resultado esperado:
Un conjunto de casos de prueba que permitirán revisar y aprobar o rechazar cada una de las funcionalidades que se prometieron en la fase de análisis.

Práctica 9. Creación de Modelos de Diseño

I. Objetivo:

Crear algunos de los modelos que conforman el diseño dentro de la metodología: Análisis Estructurado Moderno de Yourdon, a saber: modelo de procesador y modelo de implantación de programas

II. Equipo necesario:

- Computadora con MS-Office, incluido MS-Visio.

III. Material de apoyo:

- Los modelos de análisis del software que se está desarrollando, incluyendo el Diccionario de Datos y Manual Preliminar del Usuario.
- Las diapositivas sobre la metodología utilizada que se encuentran en la página www.uv.mx/personal/asumano.
- El cuaderno donde empezó a establecer sus modelos de diseño.
- Los ejemplos: Money y ATM que se le entregaron previamente.

IV. Procedimiento:

1. Definir modelo de procesador. En una Tabla de tres columnas en MS-Word
    a. A la columna uno póngale de título la palabra “Procesador” y agregue un renglón por cada conjunto de procesadores equivalentes.
    b. A la columna dos póngale como título “Procesos”, escriba cuáles serán los procesos (burbujas del DFD nivel 1) que correrán en cada uno de los procesadores que se definieron.
    c. A la columna tres titúlela como “Almacenes”, escriba la lista de almacenes (entidades y relaciones del DER) que se guardarán en cada tipo de máquina (procesador).
2. Para cada procesador se debe realizar un DE.
    a. Ponga un módulo ejecutivo llamado como el procesador
    b. Si una burbuja no tiene burbujas derivadas, póngala como módulo del primer nivel del DE.
    c. Las burbujas derivadas, póngalas en el segundo nivel asociadas a la burbuja padre.
    d. Si percibe que la cohesión de las burbujas no es correcta o que la burbuja es muy general, dibuje más módulos derivados.
    e. Incluya los almacenes que cada módulo de último nivel empleará.
    f. Incluya los parámetros de entrada salida: de control - aquellos que modificarán el comportamiento del módulo que lo está recibiendo- o de datos -aquellos que se obtienen de un almacén o que son datos necesarios para el cálculo o consulta que realiza el módulo que lo recibe-.
3. Para cada módulo, escriba su pseudocódigo.

V. Resultado esperado:

Los modelos de diseño: de procesador (tabla con procesadores, procesos y almacenes), de implantación de programas (DE por cada procesador, pseudocódigo de cada módulo).

Práctica 10. Plan de Pruebas de Integración

I. Objetivo:
Realizar el Plan de Pruebas de Integración del Sistema de Software que se está realizando en el curso que servirá para que Administrador del Proyecto de Software e Ingeniero de Pruebas puedan tanto planificar los casos de prueba como su aplicación.

II. Equipo necesario:
- Computadora con MS-Office.

III. Material de apoyo:
- Los modelos de diseño del software que se está desarrollando, incluyendo el Modelo de Procesador y Modelo de Implementación de Programas.
- Las diapositivas sobre prueba de integración de software que se encuentran en la página www.uv.mx/personal/jfernandez. Observe bien los ejemplos.
- El cuaderno donde empezó a establecer su plan de prueba.

IV. Procedimiento:
Usando el estándar IEEE 829 para especificar Planes de Prueba haga:

1. Abra un documento de MS-Word
2. Escriba el título: *Plan de Prueba de Integración del Sistema* <<nombre de su sistema de software>>
3. Ingrese una tabla de dos columnas y doce renglones.
4. Escriba los apartados de cada renglón como sigue:

1. Identificación	
2. Elementos a probar	
3. Enfoque	
4. Criterio aceptación	
5. Criterio suspensión	
6. Productos a entregar	
7. Tareas	
8. Necesidades ambientales	
9. Responsabilidades	
10. Personal	
11. Calendario	
12. Riesgos y contingencias	

5. Llene cada renglón con lo que se necesita.

V. Resultado esperado:
Una tabla para cada Diagrama de Estructura de cada procesador.

Práctica 11. Planeación de Pruebas de Unidad

I. Objetivo:
Crear una serie de casos de prueba de unidad sobre el algoritmo que le tocó realizar y que corresponde a su sistema de software que está realizando.

II. Equipo necesario:
- Una PC con el SO MS-Windows

III. Material de apoyo:
- Un módulo codificado en Pascal o pseudocódigo
- Las notas sobre el método de caminos básicos en la página: www.uv.mx/personal/jfernandez
- Un cuaderno para apuntes.

IV. Procedimiento:
1. Realice el grafo de control empleando MS-Visio y un diagrama de flujo de datos.
2. Calcule usando las tres fórmulas la complejidad ciclomática
3. Abra MS-Word
4. Escriba el título: "Casos de Prueba para el módulo: <nombre del módulo que usted codificó>"
5. En el siguiente renglón escriba: "Autor: <su nombre>"
6. Cree una tabla de dos columnas como sigue:

Tabla 0-1. Valores a considerar para la prueba del módulo: <su módulo>

Dato de entrada	Dominio	Particiones

7. Ingrese una Tabla de cuatro columnas y a cada columna póngale el nombre que sigue:

Tabla 0-2. Tabla para Casos de Prueba de Unidad

Camino básico	Num. de caso de prueba	Datos de entrada	Salida esperada

8. Para cada camino básico haga, al menos, un caso de prueba.

V. Resultados esperados:
Un reporte de prueba de unidad que deberá ser enviado por Internet al profesor con:
1. El código en Pascal o pseudocódigo
2. El grafo de control que representa el algoritmo y su complejidad ciclomática
3. La lista numerada de caminos básicos
4. La tabla de valores de entrada con los posibles valores a considerar
5. La tabla de casos de prueba.

Práctica 12. Utilización de una Herramienta para Pruebas de Unidad

I. Objetivo:

Conocer el manejo de una herramienta para la prueba de unidad y aplicar las pruebas de unidad a un módulo codificado previamente.

II. Equipo necesario:

- Una PC con el SO MS-Windows
- Delphi 2006 o Delphi 7

III. Material de apoyo:

- Su módulo que codificó en Pascal bajo el ambiente Delphi
- Notas sobre DUnit en: www.uv.mx/personal/jfernandez.

IV. Procedimiento:

1. Para Delphi 2006:
    a. Entre a la opción Archivo (File)
    b. Escoja crear un nuevo proyecto (New)
    c. De clic en la opción otros (other)
    d. Pida la opción UnitTest
    e. Escoja Test Project
    f. Ponga el nombre del proyecto, por ejemplo: *Prueba1*
2. Además, debe crear una Unit con el código del programa que realizó. Póngale un nombre, por ejemplo: *Mi-modulo*
3. Se debe crear una nueva unidad que contendrá los casos de prueba :
    a. Escoja a File>new>other>Test Unit
    b. De el nombre de la unidad a probar (source file, ejemplo: *Mi-modulo*)
    c. Ponga nombre a su unidad de prueba.
4. Codifique los casos de prueba como en el ejemplo del *máximo común divisor* (ver notas sobre DUnit).
5. Corra la aplicación (F9).

V. Resultados esperados:

Un proyecto de prueba de unidad DUnit con tres archivos de Delphi: el proyecto, la unidad a probar y la unidad de prueba. Además se deberá ver la corrida de los casos de prueba ingresados como lo muestran las notas sobre DUnit.

EJEMPLO DE UN PROYECTO DUnit

CÓDIGO DEL PROYECTO:

```
program ProyPruebaAritmetica;
{

  Delphi DUnit Test Project
  -------------------------
  This project contains the DUnit test framework and the GUI/Console test
runners.
  Add "CONSOLE_TESTRUNNER" to the conditional defines entry in the
project options
  to use the console test runner.  Otherwise the GUI test runner will be used
by
  default.

}

{$IFDEF CONSOLE_TESTRUNNER}
{$APPTYPE CONSOLE}
{$ENDIF}

uses
  Forms,
  TestFramework,
  GUITestRunner,
  TextTestRunner,
  Aritme in 'Aritme.pas',
  TestAritme in 'TestAritme.pas';

{$R *.RES}

begin
  Application.Initialize;
  if IsConsole then
    TextTestRunner.RunRegisteredTests
  else
    GUITestRunner.RunRegisteredTests;
end.
```

CÓDIGO DE LA UNIDAD A PROBAR: PROGRAMA QUE CALCULA EL MÁXIMO COMÚN DIVISOR ENTRE DOS ENTEROS

```
unit Aritme;

interface
type Aritmetica=class
  function maxcomdiv(x,y:integer):integer;
end;
implementation
function Aritmetica.maxcomdiv(x,y:integer):integer;
begin
 if ((x<=0) or (y<=0)) then
   maxcomdiv:=-1   //indica error
 else
 begin
   while (x<>y) do
     if (x>y) then x:=x-y
     else y:=y-x;
   maxcomdiv:=x;
  end;
 end;
end.
```

CASOS DE PRUEBA Y CÓDIGO DE LA UNIDAD QUE CONTIENE LOS CASOS DE PRUEBA

Tabla 0-1. Casos de Prueba de Aritmética.maxcomdiv

Número de caso	Entrada	Salida esperada	Comentario
1	0 y 8	- 1	Un parámetro es ilegal
2	6 y -2	- 1	Un parámetro es ilegal
3	10 y 3	1	No hay factor común excepto 1
4	4, 28	4	El menor es el factor común
5	30 y 30	30	Números iguales
6	42 y 231	21	El divisor común es un valor medio diferente a ambos

```
nit TestAritme;
{

  Delphi DUnit Test Case
  ----------------------
  This unit contains a skeleton test case class generated by the Test Case
Wizard.
  Modify the generated code to correctly setup and call the methods from the
unit
  being tested.

}

interface

uses
  TestFramework, Aritme;
type
  // Test methods for class Aritmetica

  TestAritmetica = class(TTestCase)
  strict private
    FAritmetica: Aritmetica;
  public
    procedure SetUp; override;
    procedure TearDown; override;
  published
```

```
    procedure Testmaxcomdiv1;
    procedure Testmaxcomdiv2;
    procedure Testmaxcomdiv3;
    procedure Testmaxcomdiv4;
  end;

implementation

procedure TestAritmetica.SetUp;
begin
  FAritmetica := Aritmetica.Create;
end;

procedure TestAritmetica.TearDown;
begin
  FAritmetica.Free;
  FAritmetica := nil;
end;

procedure TestAritmetica.Testmaxcomdiv1;
var
  ReturnValue: Integer;
  y: Integer;
  x: Integer;
begin
// TODO: Setup method call parameters
// Caso de prueba 1
  x:=0; y:=8;
  ReturnValue := FAritmetica.maxcomdiv(x, y);
  Check(ReturnValue=-1,'no debe ser menor o igual a cero');
end;

Procedure TestAritmetica.Testmaxcomdiv2;
var
  ReturnValue: Integer;
  y: Integer;
  x: Integer;
begin
// Caso de prueba 2
  x:=6; y:=-2;
  ReturnValue := FAritmetica.maxcomdiv(x, y);
  Check(ReturnValue=-1, 'no debe ser menor o igual a cero');
end;

Procedure TestAritmetica.Testmaxcomdiv3;
var
  ReturnValue: Integer;
```

```
  y: Integer;
  x: Integer;
begin
// Caso de prueba 3
  x:=10; y:=3;
  ReturnValue := FAritmetica.maxcomdiv(x, y);
  Check(ReturnValue=1, 'segundo menor sin factor común');
end;

Procedure TestAritmetica.Testmaxcomdiv4;
var
  ReturnValue: Integer;
  y: Integer;
  x: Integer;
begin
// Caso de prueba 4
  x:=4; y:=28;
  ReturnValue := FAritmetica.maxcomdiv(x, y);
  Check(ReturnValue=4, 'el primero menor, exacto');
end;

initialization
  // Register any test cases with the test runner
  RegisterTest(TestAritmetica.Suite);
end.
```

Práctica 13. Cálculo de Métricas de Diseño

I. Objetivo:

Medir los elementos estructurales que conforman los modelos de diseño del sistema que se está elaborando.

NOTA: Esta práctica puede llevarle más de dos sesiones y parte de ella se realiza en equipo y otra parte es personal.

II. Equipo necesario:

- Computadora con Windows XP o superior y, opcionalmente conectada a Internet
- Lápiz, papel y calculadora.

III. Material de apoyo:

- El IDE Delphi 7 ó superior
- Hoja de cálculo como MS-Excel
- Los modelos de implantación de programas del software que se está desarrollando: Diagramas de estructura de cada procesador y pseudocódigo de todos los módulos.
- Las diapositivas sobre el cálculo de métricas que se encuentran en la página www.uv.mx/personal/asumano. Fijarse en los ejemplos sobre el cálculo de las métricas: Complejidad Total del Sistema, Complejidad Relativa del Sistema, cohesión y acoplamiento
- Las notas sobre el método de prueba de Caminos Básicos que están en la página www.uv.mx/personal/jfernandez. Fijarse en los ejemplos sobre el cálculo de la complejidad ciclomática v(G).

IV. Procedimiento:

1. Para las métricas Complejidad Total del Sistema y Complejidad Relativa del Sistema (esta parte de la práctica es por equipo):
   a. Construya en una hoja de cálculo, como en Excel, una Tabla de 5 columnas y llámelas así:

Tabla 0-1. Tabla para el cálculo de métricas CTS y CRS

Módulo	f_{out} (i)	v(i)	S(i)	D(i)

   b. Para cada DE de cada procesador haga lo siguiente:
      i. Agregue una fila a la Tabla 2-3 por cada módulo de los DE
      ii. Anote las medidas f_{out} (i) y v(i) de cada módulo
      iii. Incluya la fórmula para S(i) y D(i)
   c. Agregue un renglón al final de la Tabla 2-3 y en la celda intersección columna S(i) agregue la función suma (∑) de toda la columna.
   d. En la celda intersección columna D(i) agregue la función suma (∑) de toda la columna.

e. Ya con las sumas calcule CTS y CRS.

2. Para las métricas de nivel componente (módulo, subrutina o procedimiento) haga (esta parte de la práctica es personal y es sólo para un módulo):
    a. Codifique un módulo en Pascal bajo el ambiente Delphi. El módulo a codificar debe devolver un resultado numérico y sus entradas pueden ser numéricas o alfanuméricas, NO debe realizar ningún tipo de acceso a almacenes (tablas, archivos).
    b. Compile y corra su módulo utilizando una interfaz gráfica que convierta los campos Edit.text en cantidades numéricas. Está interfaz será provisional y servirá para prueba de la funcionalidad.
    c. Para calcular la complejidad ciclomática:
        i. Haga el grafo de control del módulo codificado.
        ii. Calcule v(G) con las tres formas. Comprueba que sean resultados iguales; de no ser así, vuelva a calcular.
    d. Para calcular la cohesión.
        i. Realice la lista de tokens.
        ii. Haga las rebanadas de cada parámetro de salida.
        iii. Cuente los adhesivos y superadhesivos.
        iv. Haga las divisiones para el cálculo de la cohesión funcional fuerte.
        v. Haga las divisiones para el cálculo de la cohesión funcional débil.
    e. Para calcular el acoplamiento
        i. Cuente las variables de dato y de control de entrada (d_i y c_i).
        ii. Cuente las variables de dato y de control de salida (d_o y c_o).
        iii. Revise si hay variables globales y cuéntelas (g_d y g_c).
        iv. A la variable w asígnele el número de módulos que manda llamar su módulo en estudio (puede ser cero).
        v. A la variable r asígnele el número de módulos que llaman a su módulo en estudio (al menos debe ser uno).
        vi. Calcule el indicador de acoplamiento m_c.

V. Resultado esperado:

- Medida de estructura arquitectónica Complejidad Total del Sistema y Complejidad Relativa del Sistema (CRS). Note que sí CRS es mayor a 26.5 debe simplificar su modelo.
- Medidas a nivel componente:
    - Complejidad ciclomática v(G).
    - Cohesión. Si el cálculo de la CFF ó CFD resultan una fracción cercana a uno su módulo es cohesivo, lo cual es bueno, si están muy cercanos a cero, modifique su diseño, tal vez su módulo necesite dividirse.
    - Acoplamiento. Si el indicador de acoplamiento es cercano a uno tiene un acoplamiento bajo, de lo contrario, revise su codificación para lograr mayor generalidad.

Bibliografía

1. Fernández Peña, J.M., Diapositivas sobre prueba para el curso de Ingeniería de Software I, http://www.uv.mx/personal/jfernandez
2. Sumano López, M.A., Diapositivas del curso de Ingeniería de Software I, http://www.uv.mx/personal/asumano
3. Yourdon, E. (1993), Análisis Estructurado Moderno, Prentice-Hall
4. Pressman, R. (2005), Ingeniería de software, un enfoque práctico; quinta edición, McGraw-Hill