Técnico en

<DESARROLLO DE SOFTWARE>

Análisis de Requisitos del Sistema y del Software

(CC BY-NC-ND 4.0) Attribution-NonCommercial-NoDerivatives 4.0 International

Atribución

Usted debe reconocer el crédito de una obra de manera adecuada, proporcionar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que tiene el apoyo del licenciante o lo recibe por el uso que hace.

No Comercial

Usted no puede hacer uso del material con fines comerciales.

Sin obra derivada

Si usted mezcla, transforma o crea un nuevo material a partir de esta obra, no puede distribuir el material modificado.

No hay restricciones adicionales - Usted no puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otros hacer cualquier uso permitido por la licencia.

http://creativecommons.org/licenses/by-nc-nd/4.0/

Análisis de Requisitos del Sistema y del Software

Unidad IV

Artefactos de modelado para el desarrollo

1. Las principales metodologías estructuradas para el desarrollo de software

En las *Metodologías Estructuradas* para el desarrollo de sistemas la unidad básica de construcción es la *función*, es decir, modelan a un sistema en términos de conjuntos de instrucciones que ejecutan una tarea. En otras palabras, las metodologías estructuradas se enfocan principalmente en la descomposición funcional de un sistema. El objetivo es lograr una definición completa del sistema en términos de funciones, estableciendo los datos de entrada y salida correspondientes.

Existen tres herramientas gráficas de modelado, también llamadas artefactos, que sirven para construir una especificación de los requerimientos del usuario usando una metodología estructurada, éstas son: los Diagramas de Entidad-Relación (DER), los Diagramas de Flujo de Datos (DFD) y los Diagramas de Transición de Estados (DET).

Cada uno de ellos brinda una visión diferente del sistema. El primero pone énfasis en los datos y sus relaciones, el segundo centra la atención en la funcionalidad del sistema y el tercero en el comportamiento dependiente del tiempo. El Diccionario de Datos (DD) es un complemento a estas herramientas, el DD nos permite definir con un mayor grado de detalle los datos presentes en los diagramas.

Las metodologías estructuradas comparten un conjunto de principios fundamentales, que son los siguientes.

- Representar y comprender el dominio de la información, así como el dominio funcional de un problema.
- Subdividir el problema de forma tal que se descubran los detalles de una manera progresiva (o jerárquica). La partición se aplica para reducir la complejidad.
- Representar al sistema lógica y físicamente.

Mecanismos para el análisis del dominio de la información. Estos mecanismos se concentran en el flujo de datos y en su contenido o estructura. El flujo de datos representa datos de entrada a los que se les aplican ciertas funciones para transformarlos en los datos de salida. El contenido de los datos puede representarse explícitamente usando un mecanismo de diccionario o, implícitamente, con la estructura jerárquica de los datos. *Representación funcional*. Las funciones se describen normalmente como transformaciones o procesos de la información. Cada función puede ser representada usando una notación específica. Una descripción de la función puede desarrollarse usando el lenguaje natural, un leguaje procedural con reglas sintácticas informales o un lenguaje de especificación formal.

Definición de interfaces. Es importante definir tanto las interfaces del sistema con el usuario, como las interfaces entre los diferentes módulos del sistema.

Mecanismos para subdividir el problema (*Partición).* Normalmente los problemas son demasiado grandes y complejos para ser comprendidos como un todo. Por esta razón, partimos o dividimos los problemas en partes que puedan ser fácilmente comprendidas, y establecemos interfaces entre las partes. Durante el análisis de requerimientos, el dominio funcional y el dominio de la información del software pueden ser particionados. La partición descompone un problema en sus partes constituyentes. Se hace una representación jerárquica de la función o información partiendo un elemento superior horizontal o verticalmente.

a).- verticalmente, se incrementan los detalles.

b).- horizontalmente, se descompone funcionalmente el problema.

En la figura *1* se puede observar gráficamente esta descomposición vertical y horizontal de un problema.

Figura 1: Descomposición vertical y horizontal de un problema.

Representación de visiones físicas y lógicas. La visión lógica de los requerimientos del software presenta las funciones que han de realizarse y la información que ha de procesarse independientemente de los detalles de implementación. La visión física de los requerimientos del software presenta una manifestación del mundo real de las funciones de procesamiento y las estructuras de información.

2. Diagramas de Flujo de Datos (DFD)

Las herramientas gráficas más importantes del análisis estructurado son los Diagramas de Flujo de Datos (DFD). Un diagrama de flujo de datos (DFD), es una técnica gráfica que describe el flujo de información y las transformaciones que se aplican a los datos, conforme se mueven de la entrada a la salida, visualiza a un sistema como una red de procesos conectados entre sí. Los Diagramas de Flujo de datos son una notación operacional semiformal que ha sido ampliamente adoptada para la especificación de sistemas de información. Un DFD es independiente del tamaño y de la complejidad del sistema, consiste en un diagrama en forma de red que representa el flujo de datos y las transformaciones que se aplican sobre ellos al moverse desde la entrada hasta la salida del sistema. El DFD se apoya en otras 2 técnicas: Diccionario de Datos, y Especificaciones de procesos.

Los elementos que componen a un DFD se representan como se indica en la *figura 4.2* y son los siguientes:

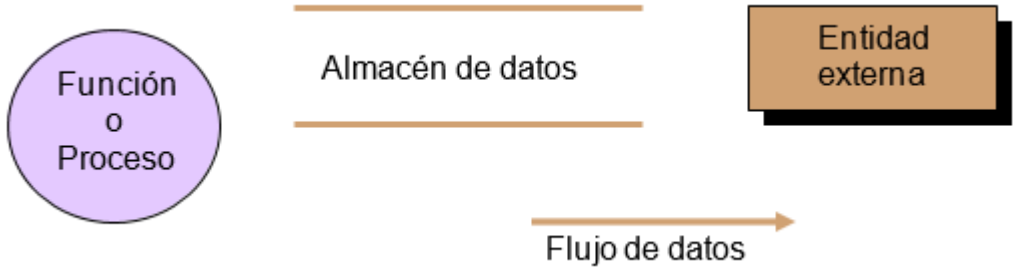
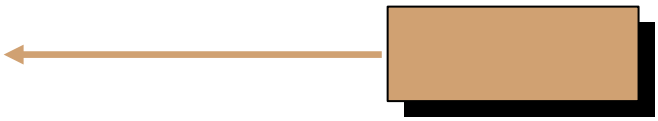


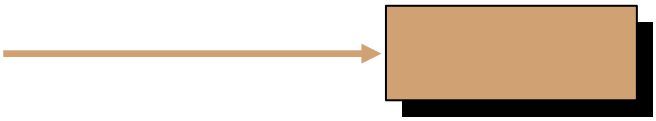
Figura 2: Elementos del Diagrama de Flujo de Datos.

- *Función o Proceso*. Representa la transformación del flujo de datos. Muestra cómo una o más entradas se transforman en salidas. Su nombre comienza con un verbo y es lo suficientemente largo y claro para que cualquier persona entienda de qué se trata. Dichas funciones van numeradas para diferenciarlas en un mismo nivel mostrando la jerarquía entre los niveles. Se representa por un círculo en cuyo interior está el nombre y el número de la función.
- *Entidad Externa:* Representa el origen o el destino de la información del sistema. Los flujos que parten o llegan a ellas definen el interfaz entre el sistema y el mundo exterior. Su representación gráfica es un rectángulo o cuadrado con el nombre.
- *Almacenamiento*. Son los datos pasivos; generalmente archivos, tablas, etc. Los almacenes de datos representan información del sistema almacenada de forma temporal por tanto representan datos en reposo; deben tener un nombre representativo, su representación gráfica son dos líneas paralelas con el nombre en medio, o también se representan con un rectángulo con el nombre adentro.

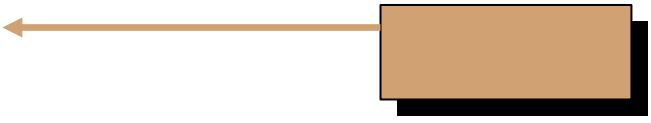
- *Flujo de dato*. Está representado por una flecha que indica su dirección, va del origen al destino. Los datos siempre van hacia y/o desde una función. Los flujos de datos representan datos en movimiento, los que se mueven hacia y desde almacenes simples no necesitan nombre si transportan toda la información del registro. Cuando se lee o escribe una porción de los elementos se debe especificar el nombre en el flujo. Existen tres tipos: *Flujo de entrada, Flujo de salida* y *Flujo de diálogo*. Los procesos pueden introducir o recuperar datos en los almacenes:
- *Flujo de salida o consulta*: Indica la utilización de la información del almacén con el proceso.



Flujo de entrada o actualización: Indica que el proceso va a alterar la información del almacén.



Flujo de diálogo: Representa como mínimo un flujo de consulta y uno de actualización que no tienen relación directa.



La conexión Entidad Externa- Almacén y viceversa solo es posible con almacenes externos que sirven de interfaz entre el sistema y una entidad externa. La conexión entre

procesos mediante un flujo de datos es posible siempre y cuando el proceso destino comience cuando el proceso origen finaliza. Los flujos de datos también se pueden dividir en *flujo síncrono* y *flujo asíncrono.*

El flujo de la información representa la manera en la que los datos cambian conforme pasan a través de un sistema. En la *figura 4.3*, la entrada se transforma en datos intermedios y más adelante se transforma en la salida.

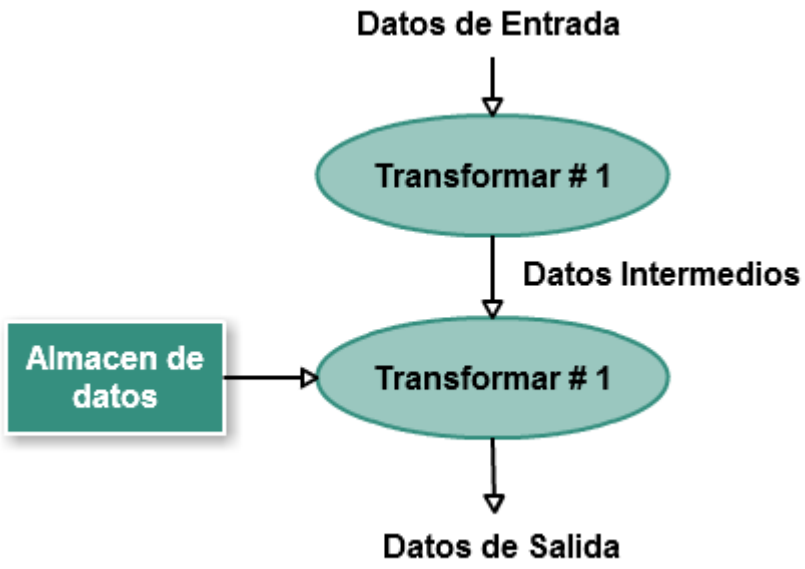


Figura 3 Estructura básica de un DFD

Un modelo de flujo de datos puede aplicarse a cualquier sistema basado en computadora independientemente del tamaño o complejidad (*figura 4.4*). El sistema acepta entradas de distintas formas; aplica un hardware, software y elementos humanos para transformar la entrada en salida; y produce una salida en distintas formas. La entrada puede ser una señal de control transmitida por un transductor, una serie de

números escritos por un operador humano, un paquete de información transmitido por un enlace a red, o un voluminoso archivo de datos almacenado en memoria secundaria. La transformación puede comprender una sencilla comparación lógica, un complejo algoritmo numérico, o un método de inferencia basado en la regla de un sistema experto. La salida puede encender un LED sencillo o producir un informe de 200 páginas.

Figura 4: Cualquier sistema computacional se puede representar mediante flujo de datos.

2.1. Características de un buen DFD.

Las Funciones o Procesos: Cada proceso debe tener Entrada y Salida, es decir, el proceso debe ser capaz de generar los flujos de salida a partir de los de entrada - El proceso no crea datos nuevos solo los transforma.

- El proceso no debe perder información.

Almacenes:

- Cada dato que sale primero debe entrar.
- No crean datos nuevos.

DFD por niveles.

Cuando el DFD es muy complejo y tiene muchas funciones, se organiza un DFD global en una serie de niveles, de modo que cada DFD de nivel inferior proporcione más detalles sobre un proceso del DFD de nivel superior.

Diagrama de Contexto: cuando construimos un DFD por niveles, el primer DFD consta de una sola burbuja, que representa el sistema completo y los flujos de datos muestran la comunicación entre el sistema y las entidades externas. A este DFD especial se le conoce como *Diagrama de Contexto*, representa el sistema de forma global, solo pueden aparecer entidades externas, flujos de datos y un único proceso que representa el sistema en su conjunto. Pueden aparecer almacenes de datos cuando son compartidos entre nuestro sistema y el exterior

Diagrama del sistema: Se le conoce como *diagrama del Sistema* o *diagrama de nivel 0*, representa las funciones principales a realizar así como la relación entre ellas. Las funciones de este diagrama deben ser lo más independientes entre sí porque esto facilita la descomposición de cada una por analistas diferentes. El DFD de nivel 0 muestra los procesos de más alto nivel del sistema y sus principales interfaces.

Procesos o funciones Primitivas: Son los procesos que no se descomponen en más diagramas de nivel inferior, ya sea porque no se puede o bien porque no interesa.

Consistencia entre niveles: Partiendo de una función de nivel superior que representa el sistema completo se va descendiendo por medio de burbujas a niveles más detallados mediante un razonamiento Top-Down hasta llegar a niveles en que las burbujas ya no

se subdividen Estas burbujas se numeran de manera adecuada. Cada burbuja *i* de un nivel particular se asocia con una figura *i* del nivel siguiente (si es que existe). Por ejemplo, la burbuja 2 del DFD de la figura 0 (nivel 0), se asocia con la figura 2 del nivel siguiente. Las burbujas en la figura *j* se numerarán *j*.1, *j*.2, *j*.3, etc. Por ejemplo, las burbujas de la figura 3 (obtenidas por la explosión de la burbuja 3 de la figura 0) se numerarán 3.1, 3.2, etc.; las burbujas obtenidas por la explosión de la burbuja 3.1, se numerarán 3.1.1, 3.1.2, 3.1.3, etc. En la *figura 5* se ve un ejemplo de explosiones.

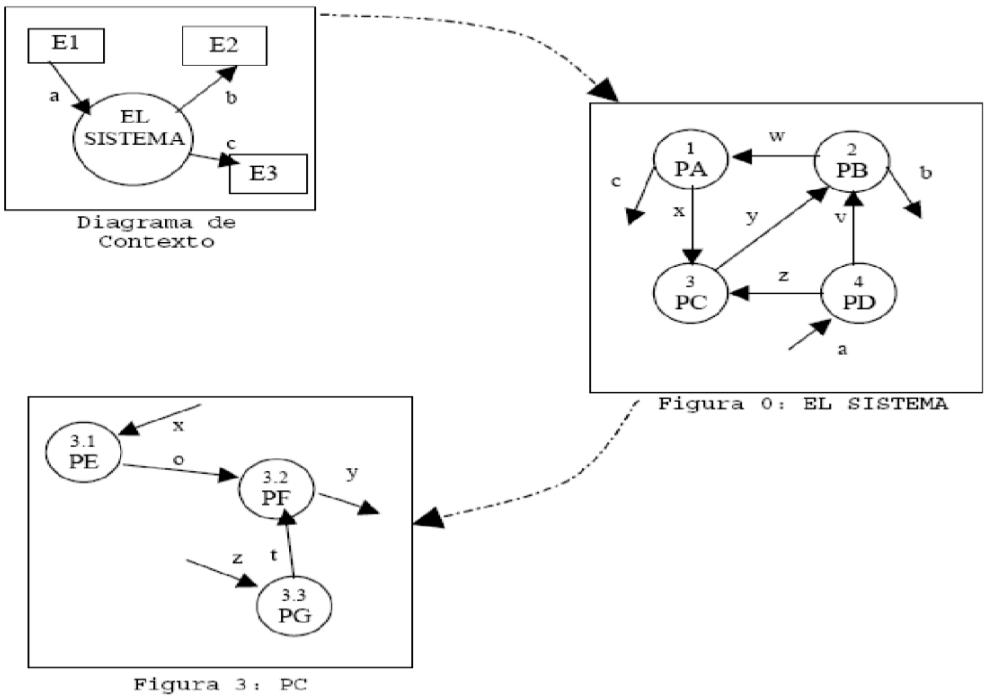


Figura 5: Ejemplo de explosiones en un DFD (Yourdon, 1993)

El nombre de la figura se hereda de la burbuja correspondiente a su explosión. Por ejemplo, si la burbuja 2 de la figura 0 se llama VALIDAR DATOS, entonces la figura 2 se deberá etiquetar “figura2: VALIDAR DATOS” y así sucesivamente para todos los niveles.

¿Cuántos niveles debe tener un DFD? Dependerá del sistema. Una regla que puede aplicarse es no poner más de media docena de procesos y almacenes relacionados. Debe caber todo en una sola hoja.

¿Deben desarrollarse todas las partes del sistema con el mismo número de niveles? No. Pueden existir partes más complejas que otras que necesiten un mayor número de niveles de partición. Pero por otro lado, si por ejemplo, al explotar el diagrama de contexto obtenemos 2 burbujas, donde la burbuja 1 es primitiva (no necesita ser explotada) y la burbuja 2 debe ser explotada en 7 niveles, significa que el modelo está desequilibrado y probablemente algunas de las porciones de la funcionalidad asignada a la burbuja 2 deban ser asignadas a otra nueva burbuja o a la burbuja 1.

¿Cómo asegurar que los niveles de un DFD sean consistentes entre sí? Se sigue una regla simple: "*los flujos de entrada y salida de una burbuja en un nivel dado deben corresponder con los que entran y salen de toda la figura asociada a dicha burbuja en el nivel inmediato inferior*". Los DFDs de la *figura 4.6* son consistentes entre sí.

¿Cómo se muestran los almacenes en los diversos niveles? La regla es la siguiente: "*mostrar un almacén en el nivel más alto donde por primera vez sirve de interfaz entre dos o más procesos, luego mostrarlo en cada nivel inferior que describa más a fondo cada una de dichas burbujas.*

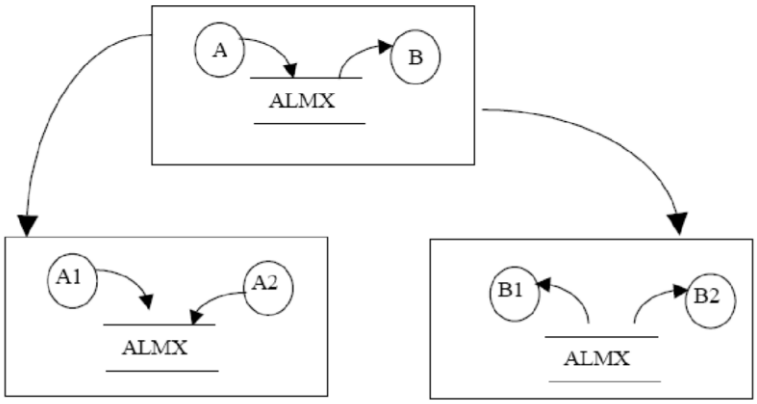


Figura 6: Ejemplo de almacén en diferentes niveles (Yourdon, 1993).

2.2. Guía práctica para la construcción de un DFD

- Escoger nombres significativos para todos los componentes del DFD.
- Numerar los procesos. Esto sirve para referirse a ellos de una manera abreviada y para construir la numeración jerárquica por niveles.
- Redibujar los DFDs tantas veces como sea necesario.
- Evitar los DFDs excesivamente complejos. En lo posible deberá caber en una sola página.
- Asegurarse que el DFD sea lógicamente consistente. Las principales reglas de consistencia son:
  - Evitar sumideros infinitos, es decir, DFDs donde solo existen flujos de entrada y ninguno de salida. o Evitar burbujas de generación espontánea. Es decir, aquellas burbujas que tienen salida pero no tienen entradas. Estas son sospechosas, sin embargo podrían existir, por ejemplo, la generación de números aleatorios. o Evitar los flujos y procesos no etiquetados. Esto podría estar ocultando un error o falta de comprensión del problema.

- Tener cuidado con los almacenes de solo lectura o solo escritura. Son sospechosos. Una excepción a esta regla son los almacenes externos al sistema que sirven de interfaz con algún otro sistema. O también el caso en el que los DFDs son muy grandes y al particionarlos podríamos encontrarnos con que una parte del sistema está modelada por un DFD en el cual un almacén es accedido como solo lectura, pero luego en otra parte del sistema, existe un DFD que accede al mismo almacén para escritura.

Ejemplo: A continuación se muestra el DFD de una visita al médico. El médico y el paciente se representan como entidades externas, los almacenes son los registros del paciente (su expediente) y los registros contables (los que tienen la información de los precios). En el DFD se observan dos procesos: el *examen médico*, que arroja tanto un diagnóstico y una medicación, como una lista de exámenes y servicios practicados, esta última sirve como flujo de entrada al segundo proceso, que es el de *contabilización*, que tiene como flujo de salida la factura para el paciente. Los flujos de datos son las flechas que sirven como entradas o salidas de los procesos, en cada flecha se pone el nombre del flujo que representa.

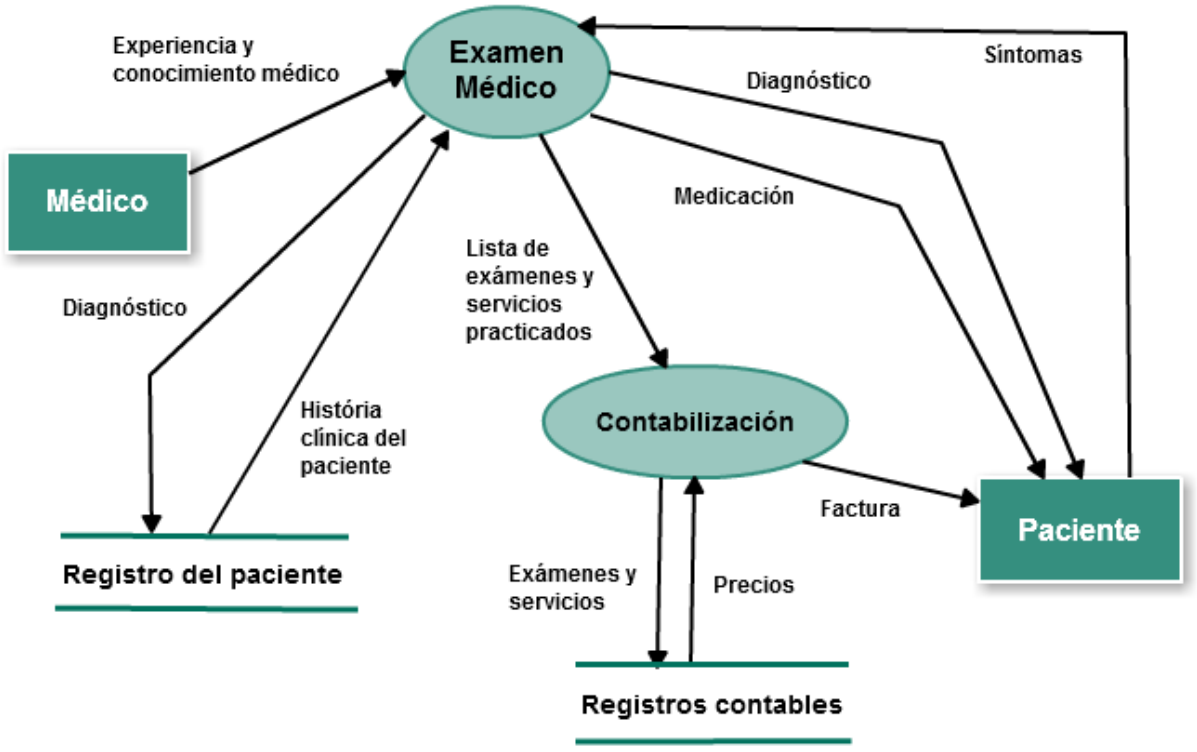


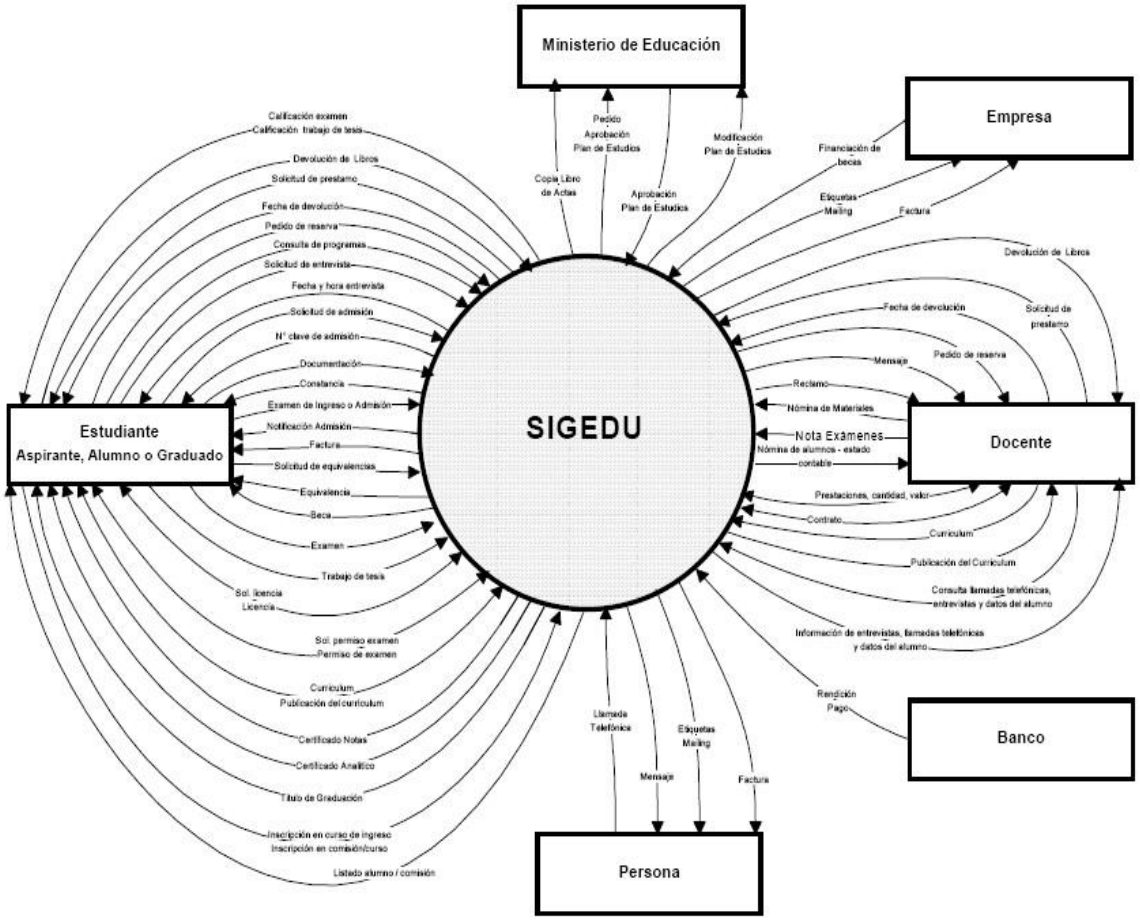
Figura 7: Diagrama de Flujo de Datos de la visita al médico (Pleeger 2002)

Para mostrar un ejemplo de un Diagrama de Flujo de Datos con Diferentes Niveles se tomó el Modelo del Sistema Integrado de Gestión Educativa (SIGEDU), de la Universidad del CEMA (Maglione y Placentino, 2001), el cual es un Sistema Informático que integra la información de las distintas áreas de la Universidad del CEMA, como por ejemplo la elaboración de programas de estudio, administración de recursos, administración de alumnos, sistema arancelario, biblioteca, docentes y graduados.

En la construcción de este modelo informático de gestión universitaria se utilizaron las técnicas de modelado de sistemas haciendo uso de las herramientas: diagrama de flujo de datos (DFD), diagrama entidad relación (DER) y diccionario de datos (DD). A continuación, se muestra el Diagrama de Contexto (una sola burbuja) en la *figura 5.8*.

Diagrama de Contexto

SIGEDU



Departamento de Ingeniería

Figura 8: Diagrama de Contexto SIGEDU (Maglione y Placentino, 2001)

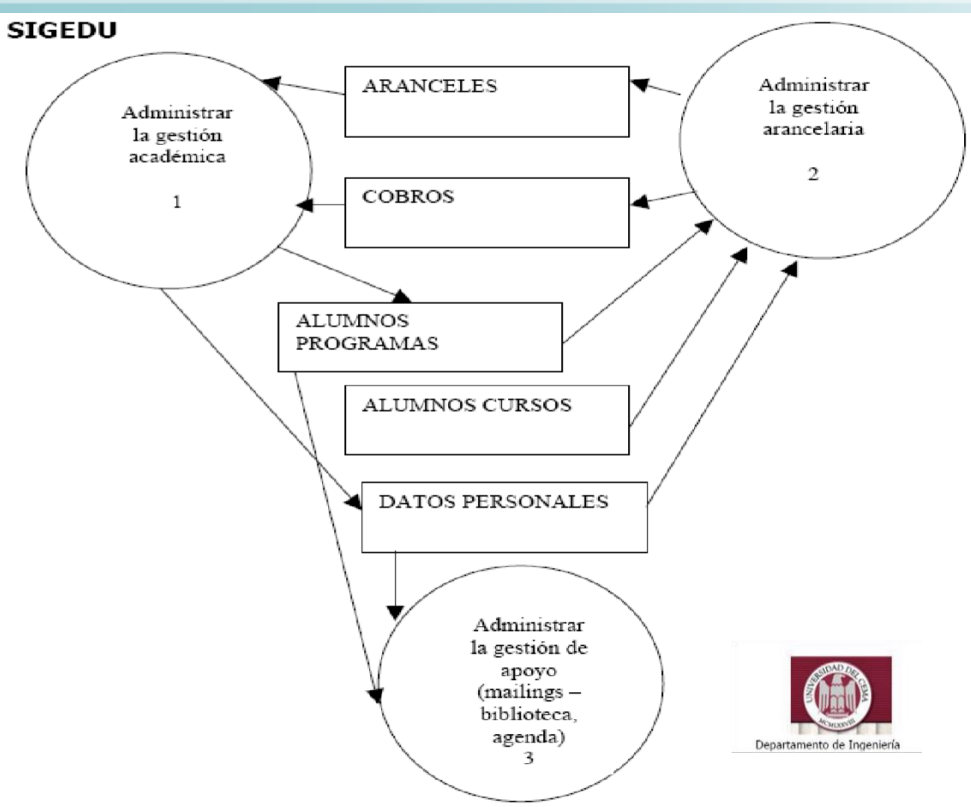


Figura 9: Diagrama del sistema (nivel 0) de SIGEDU (Maglione y Placentino, 2001)

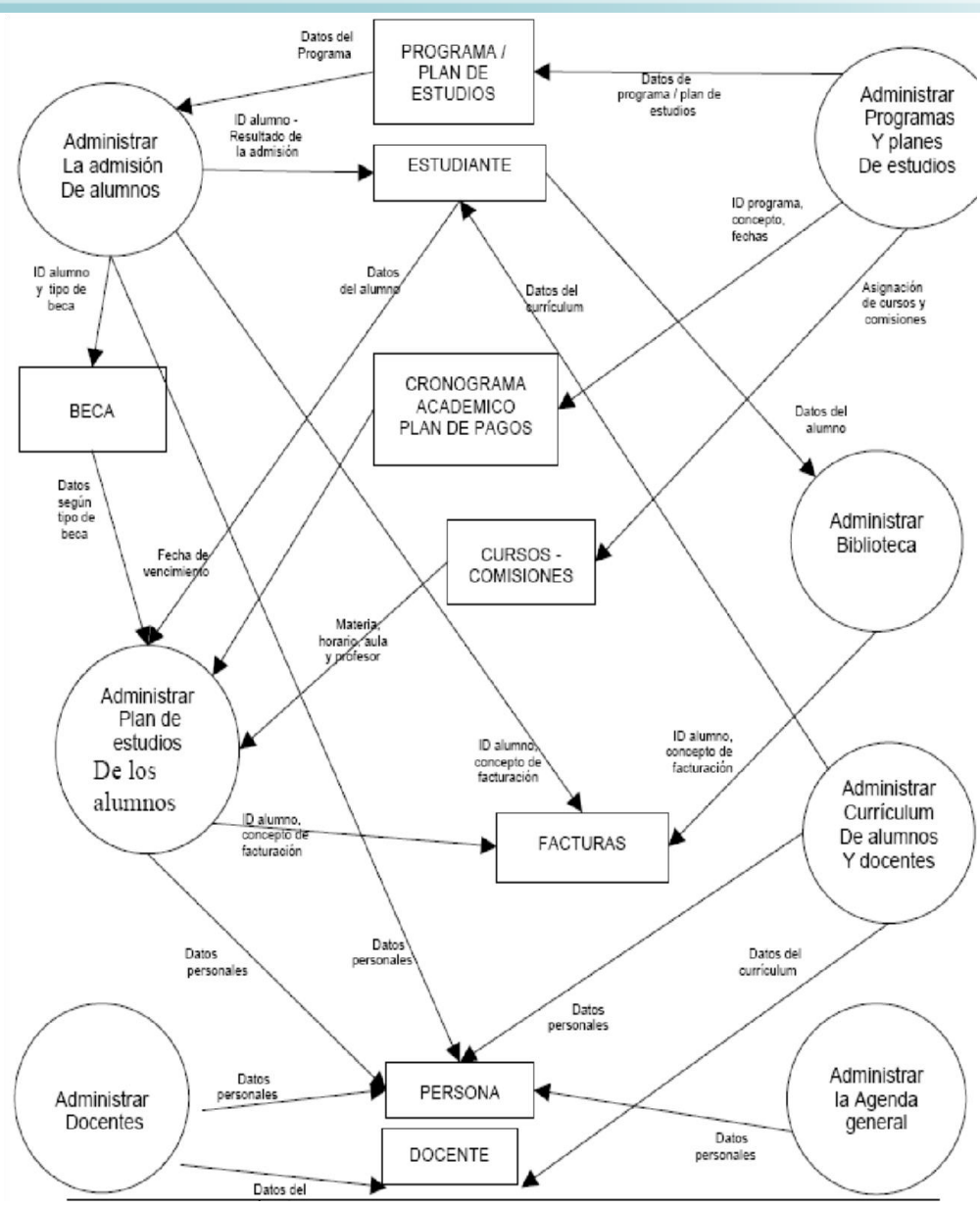


Figura 10: Diagrama de nivel 1 Burbuja 1, Administrar Gestión Académica

(Maglione y Placentino, 2001)

3. Diccionario de Datos

El Diccionario de Datos es una lista organizada de todos los datos pertinentes al sistema con definiciones precisas y rigurosas para que tanto el usuario como el analista tengan un entendimiento común de todas las entradas, salidas, componentes de almacenes y cálculos intermedios.

Un análisis del dominio de la información puede ser incompleto si solo se considera el flujo de datos. Cada flecha de un diagrama de flujo de datos representa uno o más elementos de información. Por tanto, el analista debe disponer de algún otro método para representar el contenido de cada flecha de un DFD.

Se ha propuesto el diccionario de datos como una gramática casi formal para describir el contenido de los elementos de información.

El diccionario de datos contiene las definiciones de todos los datos mencionados en el DFD, en una especificación del proceso y en el propio diccionario de datos. Los datos compuestos (datos que pueden ser además divididos) se definen en términos de sus componentes; los datos elementales (datos que no pueden ser divididos) se definen en términos del significado de cada uno de los valores que puede asumir. Por tanto, el diccionario de datos está compuesto de definiciones de flujo de datos, archivos (datos almacenados) y datos usados en los procesos (transformaciones).

El Diccionario de Datos (DD) es un listado organizado de todos los datos del sistema, con definiciones precisas. En las entradas de un DD tendremos:

- Nombre, significado y composición de los flujos y almacenes que se muestran en un DFD. Cuando un paquete de datos de los almacenes o que viaja por un flujo, es compuesto, se describirá cada una de sus partes, las cuales a su vez, si son complejas deberán aparecer como una nueva entrada del diccionario y así sucesivamente.
- Nombre, significado y composición de las entidades dependientes e independientes que se muestran en el DER (diagrama de Entidad-Relación).

Notación.

Existen varias notaciones alternativas, nosotros adoptaremos la siguiente:

= **Composición**: "*está compuesto de*" o "*es equivalente a*"

\+ **Inclusión**: y

() **Opción:** significa que el componente encerrado es opcional, es decir, lo que está entre paréntesis puede estar presente o ausente.

{} **Iteración**: cero o más ocurrencias de lo que se encuentra entre las llaves.

[] **Selección:** selección de una de las opciones encerradas entre corchetes y separadas por el símbolo "|".

texto **Comentario:** El texto entre los dos asteriscos es un comentario aclarativo a una entrada del DD.

@ **Identificador**: Se utiliza para señalar un campo o conjunto de campos que identifican cada ocurrencia de un almacén.

| **Separador**: separa opciones alternativas de construcción.

Ejemplo.

nombre=*nombre de una persona*

Primer nombre + (segundo nombre) + apellido

primer nombre = {carácter válido} segundo nombre = {carácter válido} apellido = {carácter válido} carácter válido = [A-Z | a-z | ’ | - | []

Es importante que el diccionario esté completo y sea consistente. Para controlar la completitud debemos verificar que se hayan definido todos los datos presentes en los diagramas; para la consistencia deberemos controlar que no exista más de una definición para un mismo dato y que no hayan entradas fantasmas, es decir, que no correspondan a ningún datos en los diagramas ni tampoco sean parte componente de otro dato definido en el diccionario.

4. Diagramas Entidad-Relación (DER)

Sirven para modelar un almacenamiento de datos. En muchos casos, los datos manipulados por el sistema determinan las acciones que se realizan. Puede ser útil definir los requerimientos concentrándose en los datos en lugar de las funciones. La *abstracción de datos* es una técnica para describir para qué son los datos, en lugar de cuál es su apariencia o como se denominan.

Para describir los datos se utiliza el diccionario de tipo de datos. La idea central es categorizar los datos y agrupar los elementos semejantes.

El Modelo Entidad /Relación fue desarrollado por Chen en 1976. Es un modelo muy utilizado en el campo de diseños de bases de datos. Su principal ventaja es que es traducible casi automáticamente a un esquema de bases de datos bajo modelo relacional.

El aspecto de datos es más estable que el funcional en la mayoría de los sistemas; también es mucho más difícil pensar en modelo de datos. La mayor dificultad se presenta en poder establecer la estructura de los objetos y las relaciones entre los mismos.

Elementos del Modelo Entidad/ Relación (MER)

El MER tiene sus propias estructuras que son los diagramas entidad / Relación (DER).
Entidades: Una entidad es un objeto real o abstracto de interés en una organización acerca del cual se puede y se quiere guardar información. Por ejemplo:

Curso	Alumno	Examen	Profesor

Asociado al concepto de entidad surge el de *Ocurrencia de entidad* que es una realización concreta de la misma. Por ejemplo, si las *entidades* son libro, editorial, autor, documento. Las *ocurrencias para la entidad* editorial serían: McGraw-Hill, Addison-Wesley, Alfaomega.

Atributos: Un atributo es una propiedad o característica asociada a una entidad y común a todas las ocurrencias de la misma. Por ejemplo, para la entidad Alumno se pueden

tener los atributos: nombre, grupo y calificación, o para la entidad Curso se pueden tener los atributos: unidad, nombre UEA, Carrera.

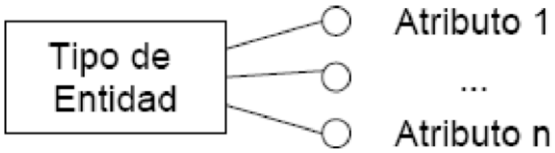
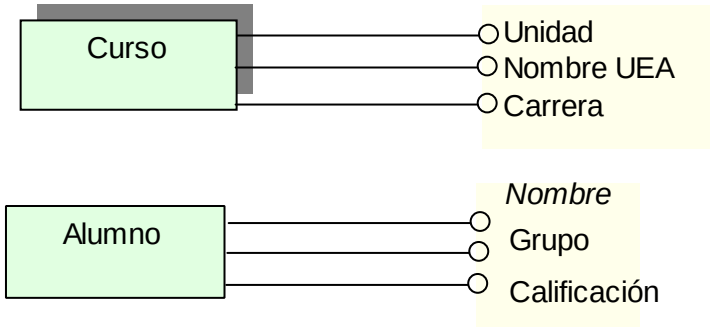
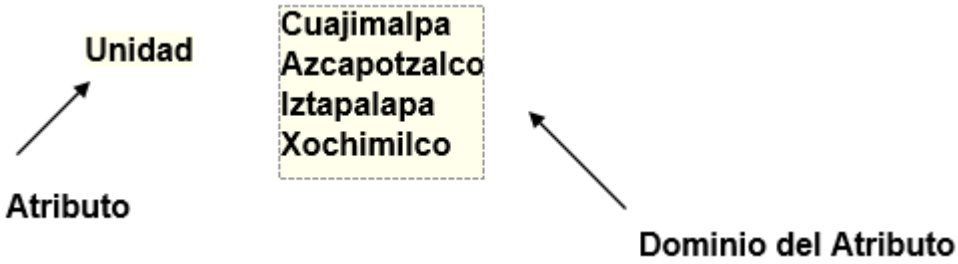


Figura 11: Tipo de Entidad y sus Atributos (De Miguel y Piattini, 2004).

Por ejemplo:



Asociado al concepto de Atributo surge el concepto de *dominio*. *Dominio* es el conjunto de valores permitidos para un atributo.



Existen 2 tipos de Atributos:

- *Atributo* ***identificador***: Distingue una ocurrencia de entidad del resto de ocurrencias. Por ejemplo *nombre* del alumno.
- *Atributo* ***descriptor***: Caracteriza una ocurrencia pero no la distingue del resto de ocurrencias de la entidad. Por ejemplo *grupo* y *calificación* del alumno.

Relaciones: Una relación es una asociación entre entidades. Entre dos entidades puede existir más de un tipo de relación. Un *objeto asociativo* es un elemento que sirve para relacionar objetos. Para que exista una instancia del mismo, deben existir instancias de todos los objetos que relaciona. Asociado al concepto de Relación surge el concepto de *ocurrencia de relación* que es la asociación concreta de ocurrencias de entidad de las diferentes entidades. . Por ejemplo: "Autor *escribe* Documento" o "Editorial *edita* Libro" (ver *Figura 4.13*).

Su representación es:

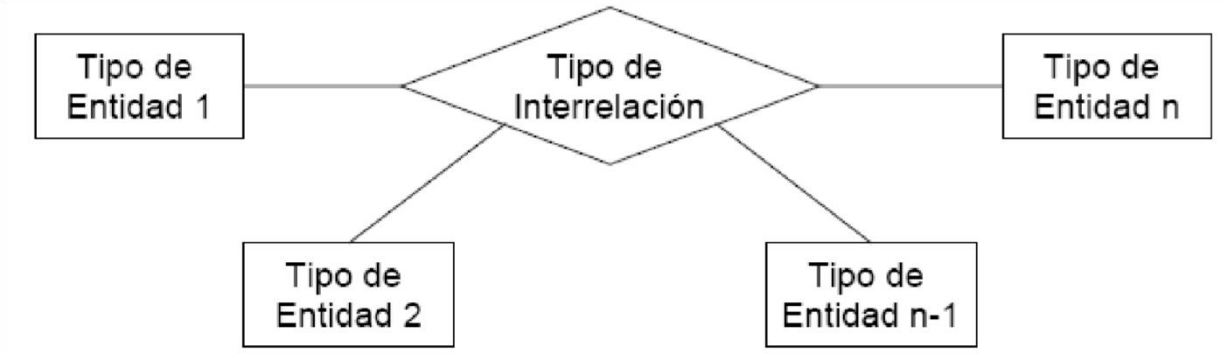


Figura 12: Tipo de Interrelación y Tipos de Entidad Relacionadas (De Miguel y Piattini, 2004).

.

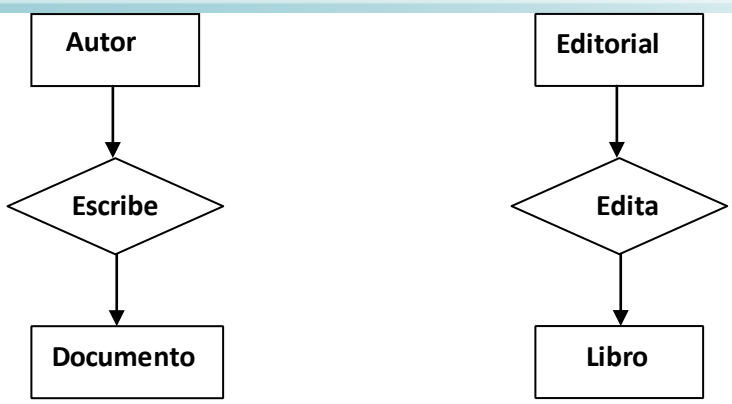


Figura 13: Ejemplos de Tipo de Interrelación entre entidades. Material de clase elaborado por Dr. Ovidio Peña (DMAS)

En la *Figura 14*: se muestra un ejemplo de Diagrama de Entidad Relación, para el mismo sistema: SIGEDU expuesto como ejemplo en la sección 2 - Diagramas de Flujo de Datos.

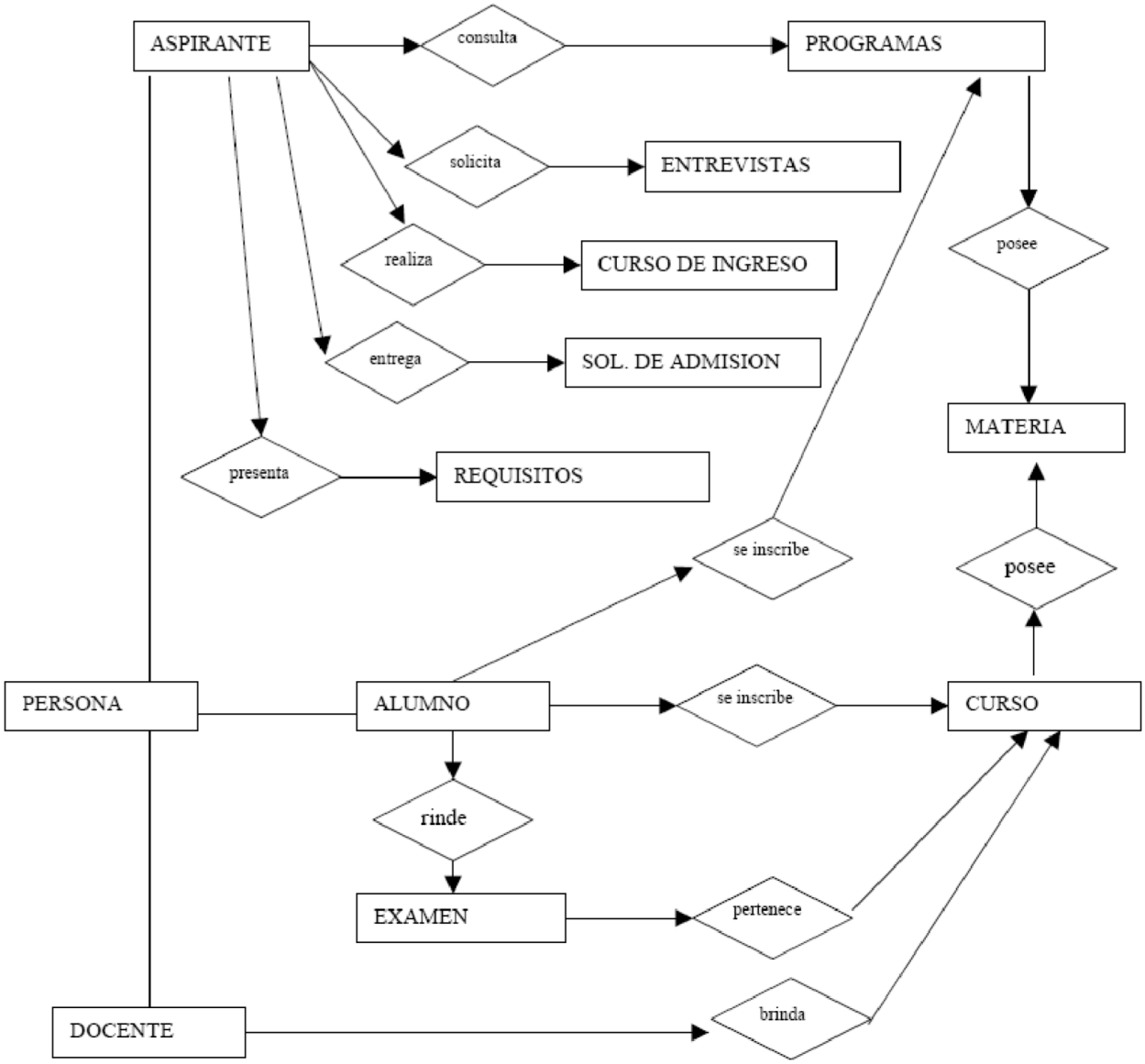


Figura 14: Diagrama Entidad-Relación de SIGEDU (Maglione y Placentino, 2001)

En este curso no se profundiza sobre los Diagramas Entidad-Relación ya que este no es el objetivo del curso. Solamente cabe mencionar que es una herramienta de análisis.

5. Diagramas de Transición de Estados (DTE)

Se utilizan cuando el aspecto más importante a considerar es el comportamiento del sistema a través del tiempo.

Los Diagramas de Transición de Estados (DTE) es una notación operacional semi-formal que permite construir modelos de comportamiento dependientes del tiempo.

Los principales componentes de un DTE son los estados y las transiciones (flechas que representan cambios de estado). Además, tenemos las condiciones y las acciones asociadas a las transiciones.

Estados

Para representar cada uno de los estados en los cuales se puede encontrar un sistema normalmente se utiliza un rectángulo o un círculo. Un estado observable del sistema corresponde a períodos en los cuales el sistema está esperando que algo ocurra en el ambiente externo o está esperando que alguna actividad que se está realizando en ese momento cambie a otra.

Entonces, decimos que "*un estado representa algún comportamiento del sistema que es observable y que perdura durante algún período de tiempo*".

Algunos ejemplos de estados pueden ser:

- Esperando que el usuario ingrese una contraseña.
- Acelerando el motor.
- Mezclando los ingredientes.

Transiciones

Para representar los cambios del sistema de un estado a otro, usamos una flecha conectando los dos estados involucrados. Por ejemplo, en la *figura 4.15* se ve un DTE con tres estados y tres transiciones.

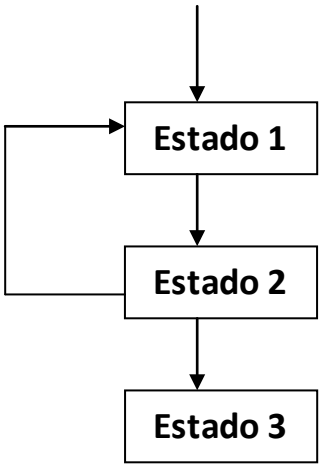


Figura 15: Ejemplo de Diagrama de Transición de Estados (DTE).

Las transiciones son las flechas que conectan dos estados involucrados y representan los cambios del sistema de un estado a otro. Nos dicen si se puede alternar entre dos estados o si una vez llegando a un estado, ya no es posible regresar al estado anterior. El sistema modelado por el DTE de la *figura 5.15* puede alternar entre el Estado 1 y el Estado 2, pero una vez que pasó del Estado 2 al Estado 3 no puede volver más a ninguno de los otros dos estados.

Dos de los estados tienen características particulares. El Estado 1 es el **estado inicial** del sistema. Se le reconoce por la flecha de entrada que no viene de ningún estado anterior. Un DTE sólo puede tener un estado inicial. El Estado 3 es un estado final. Son

estados finales aquellos estados que no tienen ninguna flecha de salida (o tienen una flecha de salida que no lleva a ningún otro estado), es decir, aquellos estados en los que una vez que se entró no se puede salir. Pueden existir múltiples estados finales. Estos son mutuamente excluyentes. Esto significa que no se puede terminar en dos o más estados finales, es decir, si se termina en un estado es imposible terminar en otro al mismo tiempo.

Condiciones y acciones.

Existen dos elementos más asociados a una transición: las condiciones que se deben satisfacer para que se produzca un cambio de estado y las acciones que el sistema lleva a cabo cuando se realiza el cambio de estado. Estos dos elementos se representan como se muestra en la *figura 4.16*.

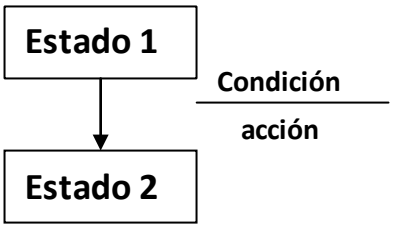
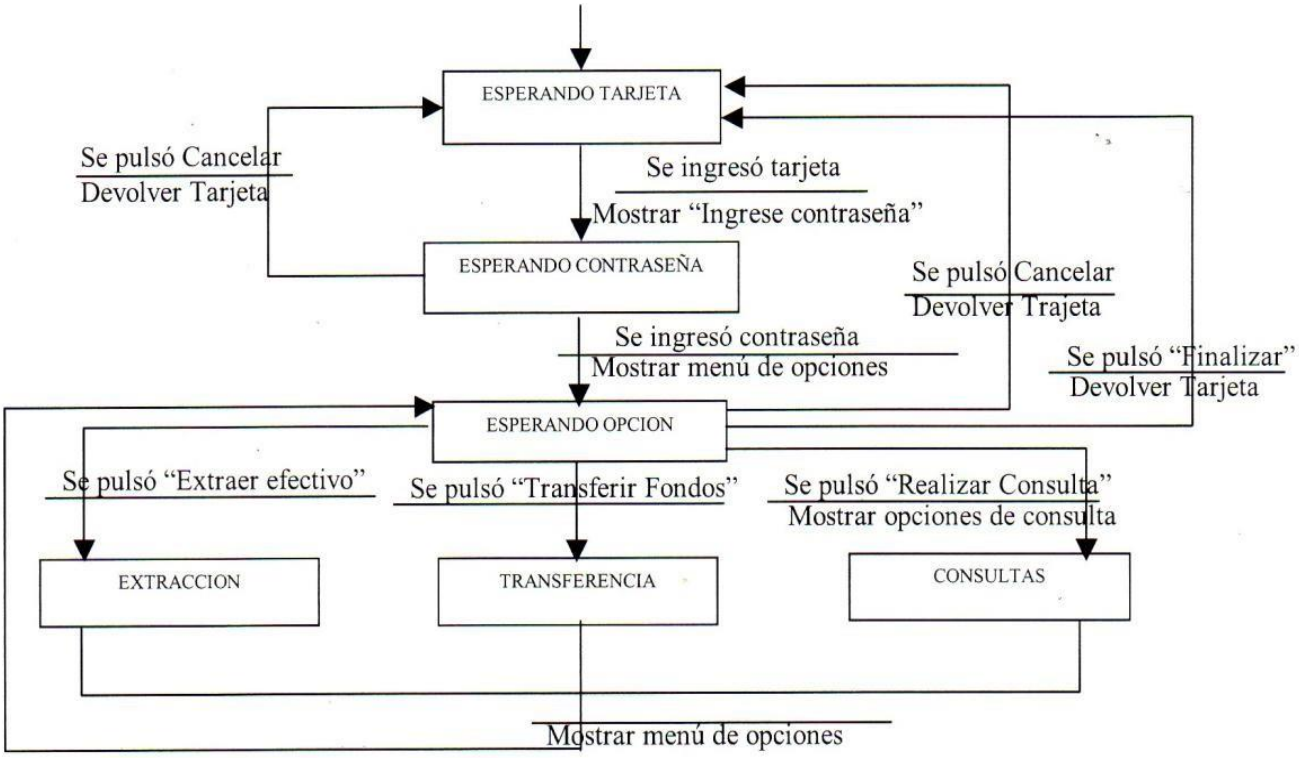


Figura 16: Condiciones y acciones en un DTE.

Particionando un DTE

Al igual que sucede con los DFDs, un DTE puede ser tan complejo que no sea posible visualizarlo cómodamente en una página. En este caso podemos descomponerlo por niveles. Cualquier estado complejo puede dar origen a un nuevo nivel el cual muestre un nuevo DTE con un estado inicial y estados finales.

El estado inicial corresponde al estado de nivel superior, es decir, se entra en el estado inicial del DTE de nivel inferior cuando se entra al correspondiente estado compuesto del nivel superior. Los estados finales del DTE de nivel inferior corresponden a las condiciones de salida del nivel superior. En la *figura 4.17* se muestra un DTE de nivel superior que modela el comportamiento de un cajero automático y en la *figura 4.18* se muestra un DTE de nivel inferior que ilustra cómo opera el estado compuesto CONSULTAS.



F*igura 17*: DTE de nivel superior para un cajero automático (Yourdon, 1993)

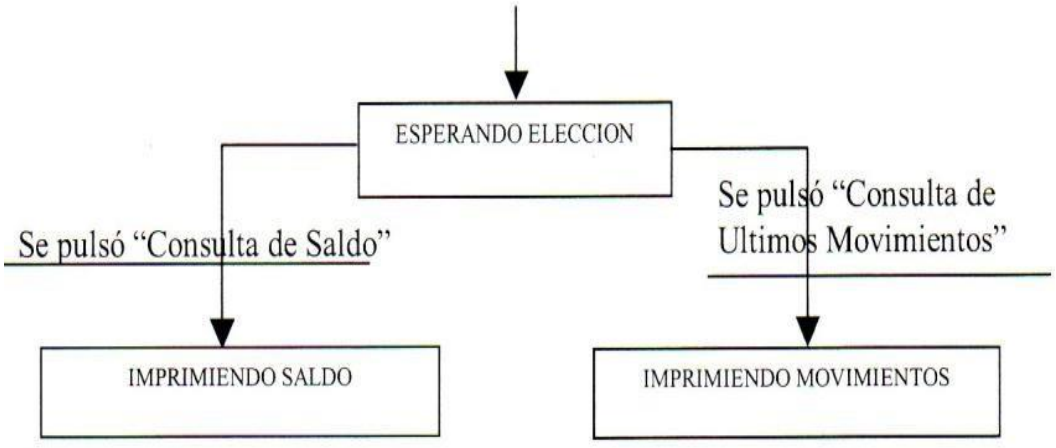


Figura 18: DTE de nivel inferior, correspondiente al estado CONSULTAS (Yourdon, 1993).

Ejemplos.- Para enriquecer el contenido de esta sección, se muestran más ejemplos de Diagramas de Transición de Estados. Cabe hacer notar que existen ligeras variaciones en la manera de expresar los DFD, DER y DET, esto se debe a que esta disciplina es relativamente nueva y todavía no existe un estándar universal, sin embargo, en esencia, se representa lo mismo.

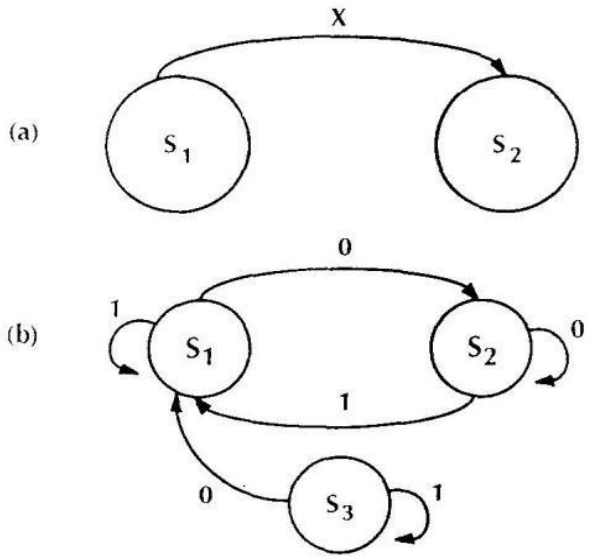


Figura 19: más ejemplos de Diagramas de Transición de Estados (Pfleeger 2002)

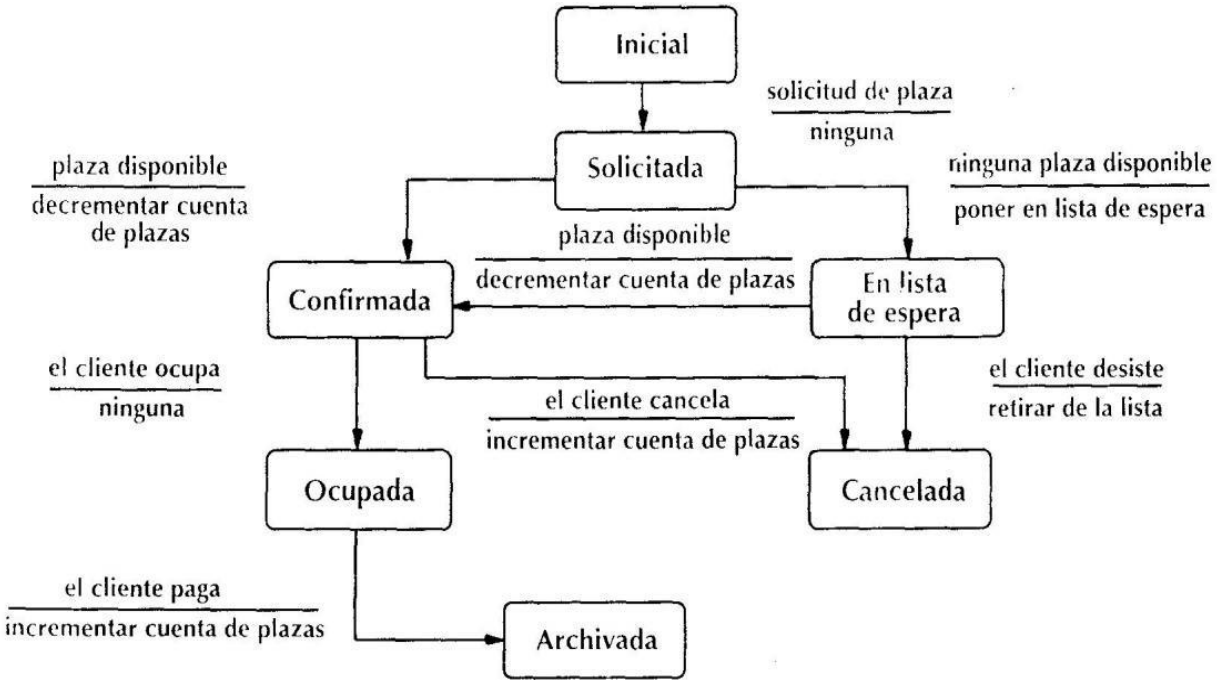


Figura 20: Diagrama de transiciones para reservas de hotel (Pleeger 2002).

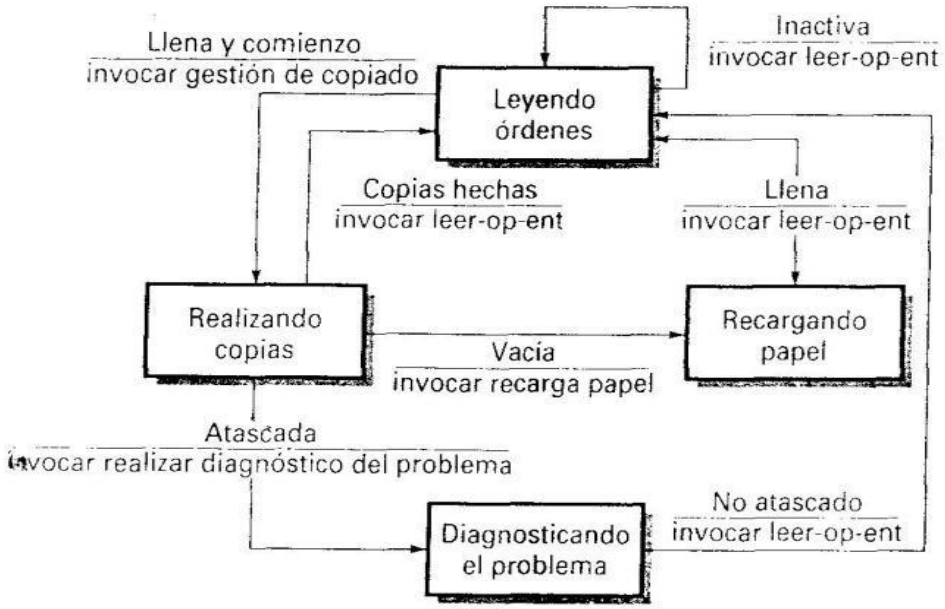


Figura 21: Diagrama de transición de estados simplificado para el software de una fotocopiadora (Presuman 2002)

Nota: Los Diagramas de transición de estado son especialmente útiles para ilustrar cuando un sistema reacciona de diferente manera ante un mismo estímulo, esta reacción depende del estado en el que se encuentra el sistema.

6. Metodologías orientadas a objetos para el desarrollo de software

En las *Metodologías Orientadas a Objetos* para el desarrollo de sistemas la unidad básica de construcción es la *clase*, es decir, modelan a un sistema en términos de objetos. Se identifican inicialmente los objetos del sistema para luego especificar su comportamiento. Existen un gran número de metodologías orientadas a objetos. Éstas utilizan diferentes *herramientas de modelado*.

Herramientas de modelado: el lenguaje UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, (*Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group), esta asociación se encarga de la definición y mantenimiento de estándares para aplicaciones de la industria de la computación. UML es un lenguaje gráfico que permite especificar, modelar, construir y documentar los elementos que forman un sistema software, principalmente orientado a objetos, sin embargo UML no está diseñado exclusivamente para software orientado a objetos.

A continuación se especifica cada una de las palabras del UML:

- **Lenguaje:** el UML es un lenguaje. Existen reglas sobre cómo deben agruparse los elementos del lenguaje y el significado de esta agrupación.
- **Modelado:** el UML es visual. Mediante su sintaxis se modelan distintos aspectos del mundo real, que permiten una mejor interpretación y entendimiento de éste.
- **Unificado**: unifica varias técnicas de modelado en una única.

La notación UML se deriva y unifica las tres metodologías de análisis y diseño Orientado a Objetos más extendidas:

- Metodología de *Grady Booch* para la descripción de conjuntos de objetos y sus relaciones.
- Técnica de modelado orientada a objetos de *James Rumbaugh* (OMT: ObjectModeling Technique).

- Aproximación de *Ivar Jacobson* (OOSE: Object- Oriented Software Engineering) mediante la metodología de casos de uso (*use case*).

UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente ayuda a visualizar el diseño y a hacerlo más accesible para otros.

UML se compone de muchos elementos de esquematización que representan las diferentes partes de un sistema de software. Los elementos UML se utilizan para crear diagramas, que representan alguna parte o punto de vista del sistema, UML contiene 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, es útil clasificarlos por su jerarquía.

Los ***Diagramas de Estructura*** muestran cuales son los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML)
- Diagrama de despliegue
- Diagrama de paquetes

Los ***Diagramas de Comportamiento*** muestran lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de transición de estados

Los ***Diagramas de Interacción*** son un subtipo de diagramas de comportamiento, que están enfocados al flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de tiempos (UML)
- Diagrama de vista de interacción (UML)

En las tres secciones siguientes se define y ejemplifica un tipo de diagrama por cada uno de los tres grupos mencionados anteriormente; los Diagramas de Clases pertenecen al grupo de los ***Diagramas de Estructura***, los *Diagramas de Casos de Uso* pertenecen al grupo de los ***Diagramas de Comportamiento*** y finalmente, los *Diagramas de secuencia* pertenecen al grupo de los ***Diagramas de Interacción*.**

Diagramas de clases

Los Diagramas de Clases pertenecen al grupo de los ***Diagramas de Estructura***, muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Se dice que los diagramas de clases son diagramas “estáticos” porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases “conocen” a qué otras clases o qué clases “son parte” de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas. En la *figura 4.22*

podemos observar un ejemplo de diagrama de clases. Los *diagramas de clases* sirven para describir los componentes esenciales de la arquitectura de un sistema. A diferencia de los Diagramas de Flujo de Datos, los Diagramas de Clases muestran relaciones de asociación entre clases y no flujo de datos entre ellas.

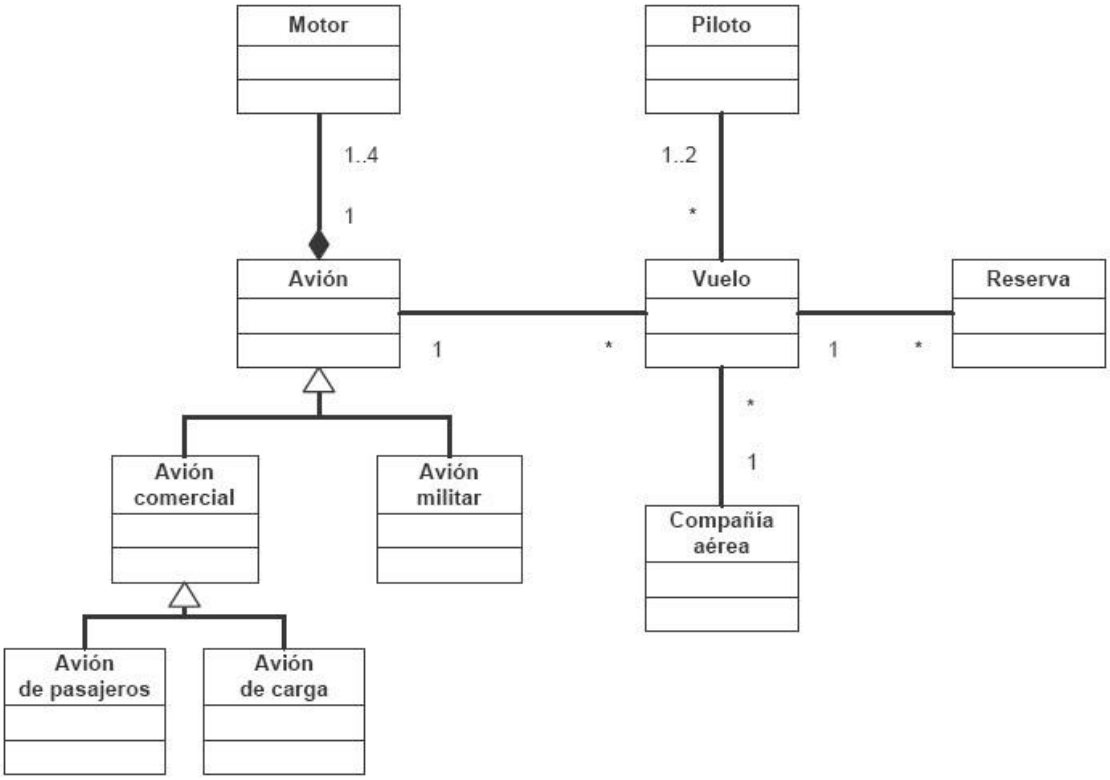


Figura 22: Diagrama de clases de un sistema de aviación.

Los *Diagramas de Subsistemas*. Se usan para describir agrupaciones de clases en un sistema

Diagramas de casos de uso

Los *Diagramas de Casos de Uso* pertenecen al grupo de los ***Diagramas de***

Comportamiento Un caso de uso es una interacción entre el sistema y una entidad externa. En su forma más simple, un caso de uso identifica el tipo de interacción y los actores involucrados. Primero se identifican los eventos externos a los que el sistema en desarrollo debe responder, y en segundo lugar, se relacionan estos eventos con los actores y los casos de uso. En las figuras *4.23* y *4.24* podemos observar ejemplos. Los *Diagramas de Casos de Uso* especifican un sistema en término de su funcionalidad. A diferencia de las metodologías estructuradas, los diagramas de casos de uso no se descomponen en funciones de programación

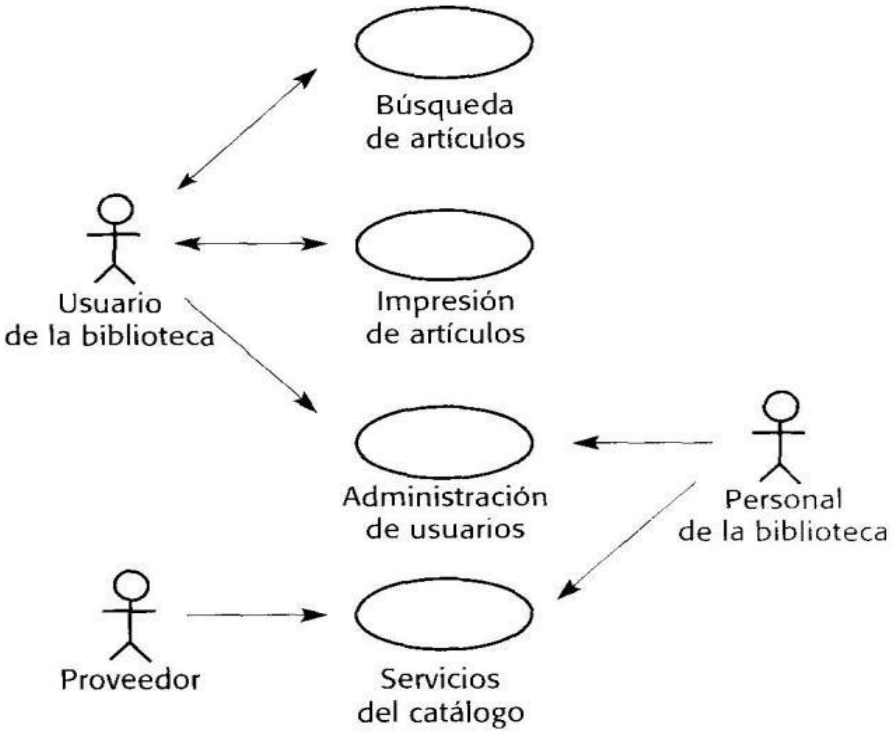


Figura 23: Casos de Uso (Sommerville 2005)

Los *Diagramas de Transición de Estado*. Describen los cambios de estado en los objetos, son similares a los DTE de la metodología estructurada, éstos también pertenecen a los Diagramas de Comportamiento.

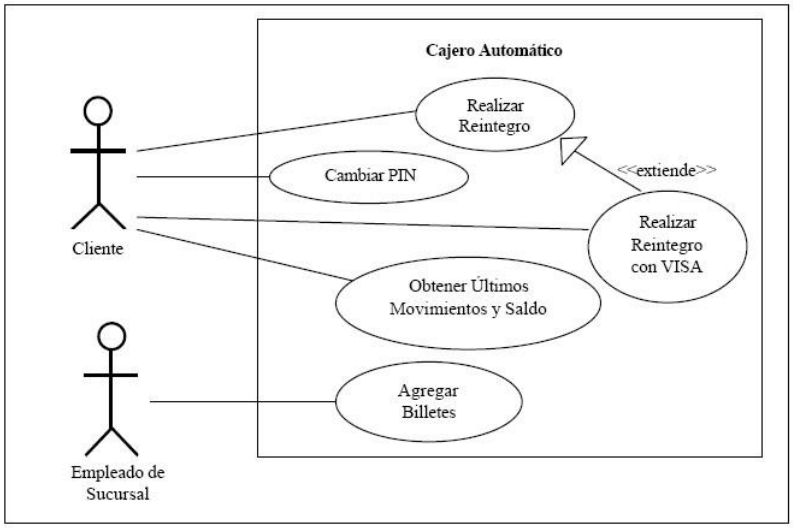


Figura 24: Casos de Uso para los actores Cliente y Empleado de Sucursal.

Diagramas de secuencia

Los *Diagramas de secuencia* pertenecen al grupo de los ***Diagramas de Interacción***, sirven para describir los aspectos dinámicos del sistema, mostrando el flujo de eventos entre objetos en el tiempo. Muestran el intercambio de mensajes (es decir la forma en que se invocan) en un momento dado. Ponen especial énfasis en el orden y el momento en que se envían los mensajes a los objetos.

Los objetos están representados por líneas intermitentes verticales, con el nombre del objeto en la parte más alta. El eje de tiempo también es vertical, incrementándose hacia abajo, de forma que los mensajes son enviados de un objeto a otro en forma de flechas

con los nombres de la operación y los parámetros. En las *figuras 4.25* y 4.*26* podemos observar ejemplos.

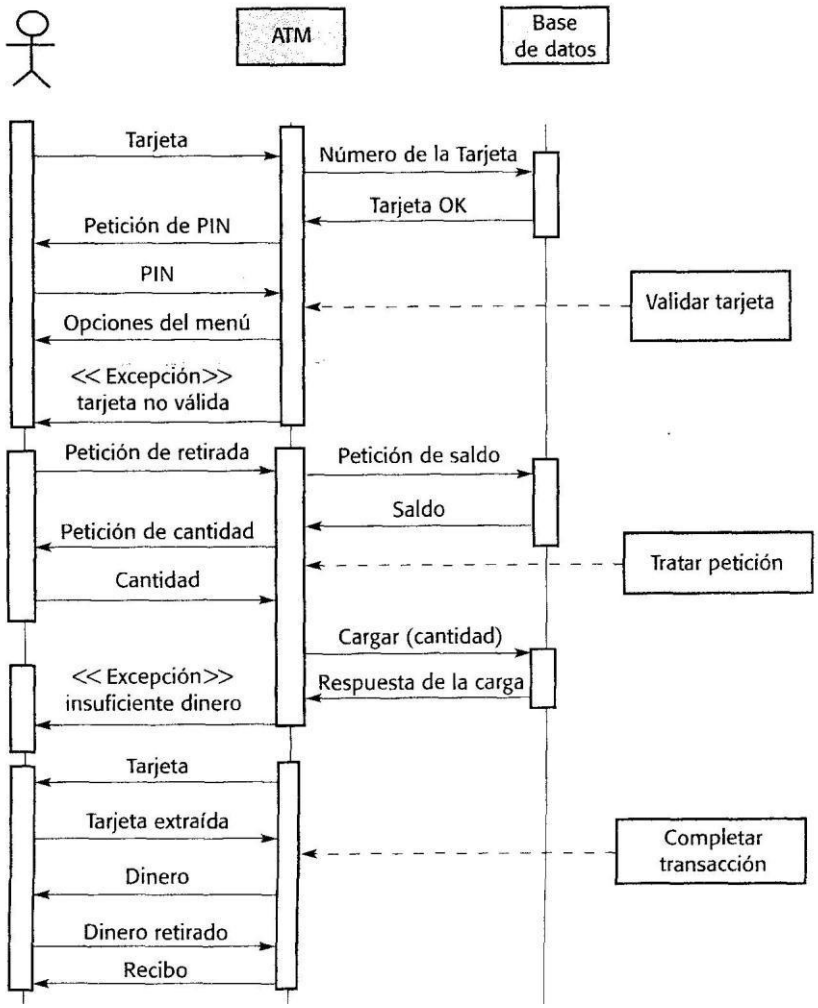


Figura 25: Diagrama de secuencia de la retirada de dinero de un cajero automático

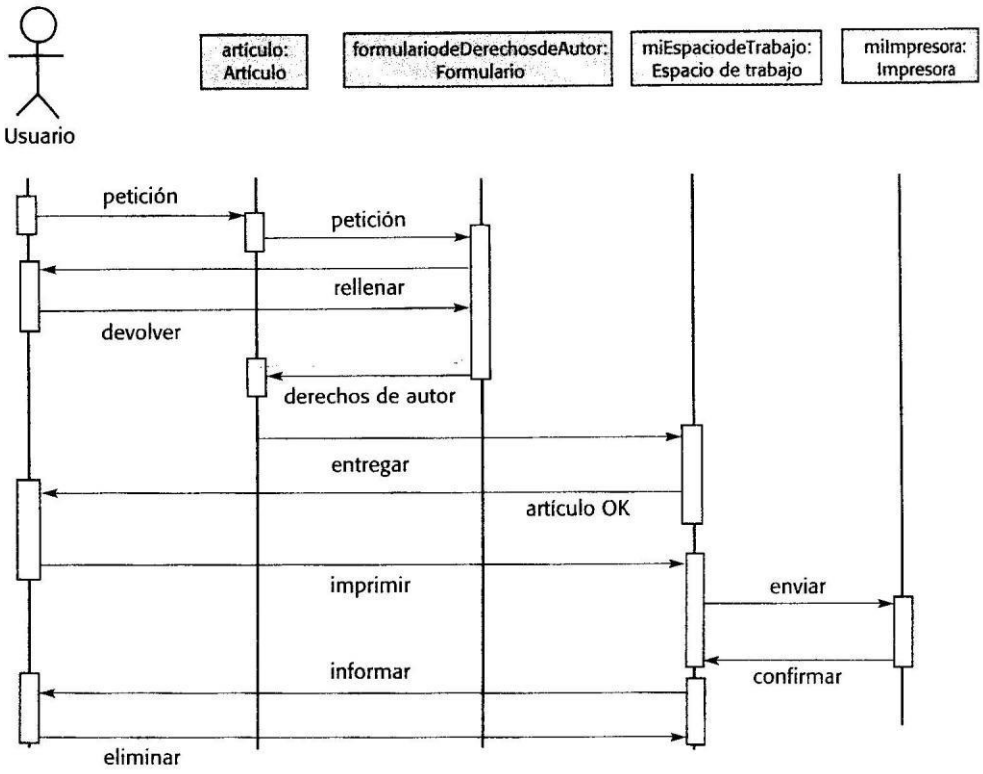


Figura 26: Diagrama de secuencia del sistema para la impresión de artículos de una biblioteca

Los *Diagramas de Colaboración*. Se utilizan para describir la comunicación entre objetos de un sistema y también pertenecen al grupo de al grupo de los ***Diagramas de Interacción***.

Las herramientas CASE

El rápido incremento en las necesidades de software en las empresas, causó que los desarrolladores de software empezaran a utilizar herramientas automatizadas como apoyo para minimizar la carga.

Hubo un gran auge en la creación de software cuando se empezó a generar con herramientas automatizadas, sin embargo, esto causó serios problemas pues existían millones y millones de líneas de código que necesitaban ser mantenidas y actualizadas. La industria de las computadoras no podía cubrir el incremento de la demanda con los métodos que se estaban usando. Esto fue reconocido como una crisis de software. Para superar este problema en el proceso de desarrollo de software se introdujeron metodologías para crear estándares de desarrollo y se creó un soporte automatizado para el desarrollo y mantenimiento de software llamado **Herramientas CASE.**

Las herramientas CASE se definen como *un conjunto de programas y procesos "guiados", que ayudan a los analistas, desarrolladores, ingenieros de software y diseñadores en una o todas las etapas que comprende un ciclo de vida, con el objetivo de facilitar el desarrollo de software*.

El objetivo general de estas herramientas es acelerar el proceso para el que han sido diseñadas, es decir, para automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas. CASE proporciona un conjunto de herramientas semiautomatizadas y automatizadas que están creando una nueva cultura de ingeniería en muchas empresas.

Las herramientas CASE se diseñaron para aumentar la productividad en el desarrollo de software y reducir su costo.

Objetivos de las herramientas CASE:

Automatizar:

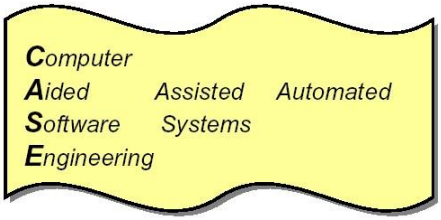
- El desarrollo del software
- La documentación
- La generación del código
- La búsqueda y corrección de errores
- La gestión del proyecto

Permitir:

- La reutilización del software
- La portabilidad del software
- La estandarización de la documentación

Las herramientas CASE son la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Variaciones en el significado de CASE.



(Instituto Nacional de Estadística e Informática, colección Cultura Informática 875-99-OI-OTDETI-INEI)

Ejemplos de Herramientas Case más utilizadas:

- ER win
- ArgoUML

- Easy Case
- Oracle Designer
- Power Designer
- System Architect
- SNAP

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.

Galileo UNIVERSIDAD La Revolución en La Educación

GES Galileo Educational System