Sistemas de Información

Unidad 6

METODOLOGÍA ESTRUCTURADA de DESARROLLO de SISTEMAS

INDICE DE CONTENIDOS

INGENIERÍA DE SOFTWARE 1
HERRAMIENTAS C.A.S.E. 2
COMPONENTES DE C.A.S.E. 3
VENTAJAS 4
DESVENTAJAS 5
HERRAMIENTAS AUTOMATIZADAS DE ANÁLISIS 7
DIAGRAMA DE FLUJO DE DATOS (DFD) 8
DICCIONARIO DE DATOS (DD) 10
ESPECIFICACIONES DE PROCESO (EP) 12
DIAGRAMA DE ENTIDAD – RELACIÓN (DER) 14
DIAGRAMA DE TRANSICIÓN DE ESTADOS (DTE) 15
BALANCEO DE MODELOS 17
DEL DFD Y EL DD 17
DEL DFD Y LA ESPECIFICACIÓN DE PROCESO (EP) 17
DE ESPECIFICACIONES DE PROCESO (EP) CON EL DFD Y EL DD 17
DEL DD CON EL DFD Y LAS ESPECIFICACIONES DE PROCESO (EP) 17
DEL DER CON EL DFD Y LAS ESPECIFICACIONES DE PROCESO (EP) 17
DEL DFD Y EL DTE 17

INGENIERÍA DE SOFTWARE

El desarrollo de sistemas de software complejos no es una actividad que pueda abordarse sin una preparación previa. El considerar que un proyecto de desarrollo de software puede abordarse como cualquier otro proyecto ha llevado a una serie de problemas que limitan la capacidad de aprovechar los recursos que el hardware pone a disposición.

Los problemas tradicionales en el desarrollo de software no van a desaparecer de la noche a la mañana, pero identificarlos y conocer sus causas es el único método que puede llevar hacia su solución.

Combinando métodos aplicables a cada una de las fases del desarrollo de software, construyendo herramientas para automatizar estos métodos, utilizando técnicas para garantizar la calidad de los productos desarrollados y coordinando todas las personas involucradas en el desarrollo de un proyecto, hace posible avanzar en la solución de estos problemas. De todo ello se encarga la disciplina llamada **Ingeniería de Software**.

Un primer concepto de la ingeniería de software fue: "*El establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico, que sea fiable y funcione de manera eficiente sobre máquinas reales*."

Con el correr del tiempo se llegó a la definición: **"Es la disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos (software)"**. Esta disciplina trasciende la actividad de programación, que es el pilar fundamental a la hora de crear una aplicación.

Como se puede deducir de su definición, la ingeniería de software abarca un conjunto de tres elementos clave: **métodos, herramientas y técnica (conjunto de procedimientos)**, que faciliten al gestor el control del proceso de desarrollo y suministren a los implementadores las bases para construir de forma productiva software de alta calidad.

Los métodos indican cómo construir técnicamente el software, y abarcan una amplia serie de tareas que incluyen la planificación y estimación de proyectos, el análisis de requisitos, el diseño de estructuras de datos, programas y procedimientos, la codificación, las pruebas y el mantenimiento. Los métodos introducen frecuentemente una notación específica para la tarea en cuestión y una serie de criterios de calidad.

Las herramientas proporcionan un soporte automático o semiautomático para utilizar los métodos. Existen herramientas automatizadas para cada una de las fases vistas anteriormente, y sistemas que integran las herramientas de cada fase de forma que sirven para todo el proceso de desarrollo. Estas herramientas se denominan C.A.S.E. (Computer Aided Software Engineering).

La Técnica o conjunto de procedimientos define la secuencia en que se aplican los métodos, los documentos que se requieren, los controles que permiten asegurar la calidad y las directrices que permiten a los gestores evaluar los progresos.

HERRAMIENTAS C.A.S.E.

Para el **desarrollo de sistemas de información**, es decir su **análisis, diseño e implementación**, los analistas de sistemas deben ser organizados, precisos y completar todo aquello que realicen. Para ello, resulta de gran utilidad el uso de **herramientas de productividad** creadas específicamente para mejorar sus tareas de rutina. A estos elementos se los denominan: **"Tecnologías de Ambientes Integrados"** también conocidos como Instrumentos o **Herramientas C.A.S.E. (Computer Aided Software Engineering Tools).**

Los tres principales **objetivos** que el analista sigue al adoptar estas herramientas:

- Incrementar la Productividad,
- Comunicarse con mayor eficacia con los usuarios,
- Integrar las actividades del Ciclo de Vida de Desarrollo de Sistemas, desde el principio hasta el final del ciclo.

Incrementar la Productividad

Medir la productividad de un analista no es sencillo, en especial a corto plazo. Entonces mientras que las actividades del ciclo de vida se encuentren en progreso, puede medirse la productividad de los analistas examinando y comparando el tiempo requerido o programado para cada conjunto de actividades de un proyecto con el tiempo que realmente está invirtiendo el analista a través de la utilización de herramientas C.A.S.E. En consecuencia, cuando se habla de productividad, **se habla de una mejora medible en cantidad y calidad de los resultados** que obtiene el analista con la ayuda de las nuevas tecnologías, comparada con lo que pudiera haber ocurrido posiblemente con métodos alternativos.

Existe software que permite que el analista (o quien lo use) dibuje y modifique con facilidad los diagramas. Así el analista puede ser más productivo reduciendo el tiempo de dibujo manual como las repetidas correcciones que se realizan a los diagramas.

Mejoramiento de la Comunicación Analista - Usuario

Con el fin de que el sistema propuesto llegue a utilizarse, es esencial que exista una comunicación excelente entre analistas y usuarios a lo largo del ciclo de desarrollo de sistemas. La forma en que los analistas pueden mejorar su comunicación con los usuarios es mediante el uso de diagramas generados por computadora. Con el uso de pantallas los clientes pueden ver cómo los diagramas fueron concebidos o creados y al observarlos pueden solicitar correcciones o cambios a realizar en el momento oportuno.

Integración de las Actividades del Ciclo de Vida

La integración de las actividades del Ciclo de Vida son importantes porque capacitan al analista para que pueda comprender lo que se encuentra dentro de la perspectiva del sistema. Esto le ayuda a asimilar adecuadamente un problema, como así también a percatarse del impacto que puede generar cualquier cambio en el desarrollo del sistema.

Las Herramientas C.A.S.E. son muy útiles ante la necesidad de realizar iteraciones o repeticiones de retroalimentación y modificación en una etapa particular del ciclo de vida.

Etapas del Ciclo de Vida de Desarrollo de Sistemas	Utilidad de Herramienta C.A.S.E.
Identificación de Problemas, Oportunidades y Objetivos	Elabora diagramas de plan de proyecto
Determinación de los Requerimientos de Información	Organiza actividades de captura de datos mediante el uso de prototipos
Análisis de las Necesidades del Sistema	Dibuja diagramas de flujo de datos, desarrollo del diccionario de datos
Diseño del Sistema Recomendado	Elabora pantallas e informes
Desarrollo y Documentación del Software	Construye diagramas de estructura y transfiere datos del diccionario de datos al código del programador
Pruebas y Mantenimiento del Sistema	Genera datos de prueba y verifica los datos de las pruebas
Implementación y Evaluación del Sistema	Uso de software de administración de proyectos para la evaluación

El compromiso del usuario es sumamente importante durante todas las etapas. La integración de actividades a través del uso de las herramientas C.A.S.E. mejora la comprensión de los usuarios sobre cómo se encuentran relacionadas las etapas del ciclo de vida, así como su interdependencia.

COMPONENTES DE C.A.S.E.

En general, estas herramientas incluyen los siguientes cinco componentes:

1. **Herramientas para Diagramación:** dan soporte al análisis y documentación de los requerimientos de una aplicación. Poseen la capacidad de dibujar diagramas de flujo de datos, además de guardar los detalles en forma interna. Cuando es necesario realizar cambios, la naturaleza de éstos se describe en el sistema, el cual puede entonces volver a dibujar todo el diagrama de manera automática. La capacidad para cambiar y volver a dibujar resuelve una actividad que los analistas encuentran tediosa y poco deseable.

2. **Depósito Centralizado de Información:** la captura de datos, su análisis, procesamiento y distribución es asistida por un depósito de información centralizado o **Diccionario de Datos (DD)**. El diccionario contiene detalles sobre los componentes del sistema, tales como datos, flujo de datos y procesos, incluyendo información sobre el volumen y frecuencia de cada una de las actividades. También incluyen controles y medidas de protección que preservan la exactitud y consistencia de los datos del sistema. El uso de niveles de autorización, validación de procesos y procedimientos para verificar la consistencia de las descripciones asegura que el acceso a las definiciones y las revisiones hechas a ellas en el depósito de información ocurran en forma apropiada y acorde con procedimientos ya establecidos.

3. **Generador de Interfaces:** las interfaces del sistema son los medios que permiten a los usuarios interactuar con una aplicación, ya sea para dar entrada a datos o para recibir información. Los generadores de interfaces ofrecen la capacidad para preparar imitaciones de las interfaces con los usuarios (menús de demostración del sistema, pantallas de presentación y formato de los informes de salida).

4. **Generadores de Código:** automatizan la preparación de software. Estos incorporan métodos que permiten convertir las especificaciones del sistema en código ejecutable. Los mejores generadores de código producen aproximadamente el 75% del código fuente de una aplicación, el resto debe ser escrito por los programadores. Los mayores beneficios se obtienen cuando los generadores de código se encuentran integrados con un depósito central de información. Esta combinación alcanza el objetivo de crear un código que pueda volverse a emplear. Cuando las especificaciones cambian, se puede volver a generar el código al alimentar los detalles del diccionario de datos a través del generador de código. El contenido del diccionario puede emplearse de nuevo para preparar el código ejecutable.

5. **Herramientas de Administración:** ayudan a los gerentes de proyecto a calendarizar las actividades de análisis y diseño así como la asignación de recursos a las diferentes actividades del proyecto. Soportan el seguimiento de los tiempos de desarrollo de un proyecto y los comparan con los planificados y realizan la misma labor con la asignación de tareas específicas al personal.

La adopción de las herramientas C.A.S.E. tiene **Ventajas** y **Desventajas**

VENTAJAS

- **Facilidad para la Revisión de Aplicaciones:** una vez que las aplicaciones se implantan, se emplean por mucho tiempo. Las herramientas C.A.S.E. proporcionan un beneficio al facilitar la revisión de las aplicaciones. Contar con un depósito central, agiliza el proceso de revisión ya que éste proporciona bases para las definiciones y estándares para los datos. Las capacidades de generación interna, si se encuentran presentes, contribuyen a modificar el sistema por medio de cambios en las especificaciones más que por ajustes al código fuente.

- **Soporte para el Desarrollo de Prototipos de Sistemas:** se desarrollan diseños para pantallas y reportes con la finalidad de mostrar la organización y composición de datos, encabezados y mensajes. Los ajustes al diseño se hacen con rapidez para alterar la presentación y las características de la interface. El desarrollo de prototipos puede producir un sistema que funcione.

- **Generación de Código:** Algunas herramientas C.A.S.E. tienen la capacidad de producir el código fuente. La ventaja más visible de esta característica es la disminución del tiempo necesario para preparar un programa. Sin embargo, la generación de código también asegura una estructura estándar y consistente para el programa (lo que tiene gran influencia en el mantenimiento) y disminuye la ocurrencia de diversos tipos de errores, mejorando de esta manera la calidad. Las características de la generación de código permiten volver a utilizar el software y las estructuras estándares para generar dicho código, así como el cambio de una especificación modular, lo que significa volver a generar el código y los enlaces con otros módulos.

- **Mejora la habilidad para satisfacer los requerimientos del usuario:** la satisfacción de los requerimientos del usuario guarda relación directa con el éxito del sistema. Tener los requerimientos correctos mejora la calidad de las prácticas de desarrollo. Las herramientas C.A.S.E. disminuyen el tiempo de desarrollo, algo que es importante para los usuarios. Las descripciones gráficas y los diagramas, los prototipos de reportes y la composición de las pantallas, contribuyen a un intercambio de ideas más efectivo.

- **Soporte iterativo para el proceso de Desarrollo:** El desarrollo de sistemas es un proceso iterativo. Las herramientas C.A.S.E. soportan pasos iterativos al eliminar el tedio manual de dibujar diagramas, elaborar catálogos y clasificar. Los analistas repasarán y revisarán los detalles del sistema con mayor frecuencia y en forma más consistente.

- **Alcanzar una Ventaja Competitiva:** Las Herramientas C.A.S.E. pueden proporcionar un nivel competitivo sobre otros que no las hubieran adoptado o que no hayan capitalizado por completo su potencial en la productividad.

- **Estandarización de Métodos Internos y Externos:** A través de la estandarización en la elaboración de diagramas, los analistas de sistemas pueden comprender con rapidez el trabajo que se ha realizado en el sistema. Esto elimina la necesidad de establecer primero qué técnica de elaboración de diagramas se empleó antes de llegar al contenido. También tienen una ventaja externa, ya que un analista con experiencia en el uso de herramientas C.A.S.E., permitirá que el cliente se comunique de una manera fácil.

- **Ver problemas viejos de nuevas maneras:** crean un nuevo contexto para problemas viejos de tal forma que se estimula la creación de nuevas ideas. Ej.: Cuando los diagramas de flujo pueden elaborarse con rapidez y luego corregirse, es más fácil comparar la conceptualización de los flujos de datos que se tenían con anterioridad. Este tipo de tormenta de ideas gráfica sirve para la creación de nuevas ideas.

- **Adopción de la Automatización como rutina:** cuando los analistas adoptan las Herramientas C.A.S.E. mejoran su credibilidad ante sus clientes, ya que muestra con su ejemplo que la automatización les confiere beneficios a los usuarios.

DESVENTAJAS

- **Confiabilidad en los Métodos Estructurados:** Muchas herramientas C.A.S.E. están construidas teniendo corno base a las metodologías del análisis estructurado y del ciclo de vida de desarrollo de sistemas, lo que puede convertirse en la principal limitante ya que no todas las organizaciones emplean métodos de análisis estructurado. El nivel de conocimiento de los métodos estructurados es alto entre los profesionales de sistemas de información. Si la organización o el analista no utilizan los métodos propios del análisis estructurado o no consideran su uso, entonces el valor de C.A.S.E. disminuye.

- **Falta de niveles estándar para el Soporte de la Metodología:** Aún no aparece un conjunto "estándar" de herramientas C.A.S.E. que puedan dar soporte a los diagramas que emplea una metodología, sus reglas y sus procesos.

- **Conflictos en el uso de los Diagramas:** Las herramientas difieren en el uso que hacen de los diagramas. Algunas son herramientas exclusivamente para gráficas, que se abocan al dibujo de diagramas para el análisis de entrada y salida de datos. Este tipo de herramientas pueden restringir ya sea el proceso de desarrollo normal seguido por una organización o el estilo particular de trabajo de los analistas.

- **Diagramas No Utilizados:** Los productos C.A.S.E. emplean gráficas para modelar y generar informes sobre el análisis y desarrollo durante todo el proceso de desarrollo de sistemas. En algunos casos, las herramientas gráficas no se emplean del todo o tal vez no se utilicen en la forma en que deberían emplearse. Algunos analistas prefieren para algunas tareas un lenguaje estructurado o descriptivo. Los profesionales de sistemas no hacen uso de herramientas gráficas en el desarrollo de software; más bien las emplean para automatizar la producción de informes y documentación del sistema.

- **Función Limitada:** Aunque una herramienta puede apoyar varias fases del ciclo de vida de desarrollo de sistemas o adaptarse a diferentes metodologías de desarrollo, por lo general su enfoque está dirigido hacia una fase o método específico. Ej.: los encargados de desarrollar un nuevo producto pueden afirmar que éste apoya todo el proceso de análisis y diseño y sin embargo sus capacidades de comprobación y verificación de errores son más rigurosas en el área de análisis o en la de diseño, pero no en ambas.

- **Alcance Limitado:** Aunque muchas herramientas incluyen la capacidad de verificar las especificaciones para determinar su completitud o consistencia, virtualmente no llevan a cabo ningún análisis de los requerimientos de la aplicación. Por tanto, el alcance de las actividades de desarrollo asociado con las herramientas existentes es bastante limitado. La mayor parte de productos C.A.S.E. describe (documenta) pero no analiza. No ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas.

- **Las Tareas Humanas siguen siendo Críticas:** Las herramientas deben estar en manos de personas con experiencia y deben "adaptarse" a la arquitectura de la información así como a las metodologías de desarrollo utilizadas por la organización. Las actividades críticas no son el desarrollo de gráficas que documenten al sistema existente sino que son aquellas tareas donde las personas interactúan entre sí: determinación y verificación de requerimientos con el usuario. Obtener y comprender los requerimientos son tareas realizadas por humanos, y lo más probable es que se continúe de tal forma.

- **Cambios de Métodos de Trabajo:** El cambio de tecnología implica también un cambio de los métodos de trabajo. La resistencia al cambio es parte de la naturaleza humana, nunca será fácil dejar atrás un antiguo método de trabajo por otro nuevo.

- **La inversión en paquetes comerciales costosos:** Los costos iniciales del software C.A.S.E. pueden ser significativos. Debe ponderarse los incrementos en productividad y realizar un análisis costo/beneficio para determinar su adquisición.

HERRAMIENTAS AUTOMATIZADAS DE ANÁLISIS

La **Especificación Funcional** asociada a un sistema de información puede contener una gran cantidad de diagramas que:

- identifican relaciones entre componentes del sistema y con entidades externas al mismo;
- explican la estructuración de los procesos que lo componen, el flujo de los datos que se capturan o se generan, las relaciones entre ellos y los lugares donde se almacenan;
- definen el sentido y composición de los bloques de datos y sus repositorios;
- explican el funcionamiento interno de los procesos y sus cursos de acción posibles;
- describen el comportamiento del sistema y los estados que pueda asumir en el tiempo.

Además del trabajo que necesario para crear y mantener dichos diagramas se requiere de una gran cantidad de trabajo para verificarlos con el fin de asegurar que sean consistentes y estén completos.

El desarrollo de poderosas estaciones de trabajo gráficas llevó a que varios proveedores de software ofrezcan productos que dibujan estos tipos de diagramas, además de llevar a cabo una variedad de labores de revisión de errores, lo que produjo la creación de modelos de sistemas de mayor calidad y visualmente más atractivos.

La revisión de errores automatizada y la generación de programas directamente desde las especificaciones funcionales merced a las herramientas C.A.S.E. elimina problemas de productividad, confiabilidad, reduciendo la necesidad de programadores.

Existen varias herramientas de modelado de sistemas siendo las más importantes:

- El Diagrama de Flujo de Datos.
- El Diccionario de Datos.
- El Diagrama de Entidad - Relación.
- Las Tablas de Decisión.
- El Diagrama de Transición de Estados.

DIAGRAMA DE FLUJO DE DATOS (DFD)

Es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por conductos y tanques de almacenamiento de datos.

Componentes del DFD:

Proceso: Es equivalente a una FUNCIÓN o TRANSFORMACIÓN. Muestra una parte del sistema que convierte entradas en salidas. Se nombran mediante frases VERBO + OBJETO. Ej: VALIDAR PEDIDO.

Flujo: Describen el movimiento de paquetes o bloques de información de una parte del sistema a otra. Representan Datos en Movimiento (a diferencia de los Almacenes). Pueden representar Materiales Físicos.

Almacén: Modela una colección de Datos en Reposo. Se nombran con el plural de las etiquetas de paquetes que entran o salen de ellos.

Terminador: Representan entidades externas con las cuales el sistema se comunica. Suelen ser personas, grupos o departamentos de la empresa. Son externos al sistema que se está modelando. Las relaciones entre los terminadores no se muestran en el DFD.

Guía para la construcción de un DFD

Las siguientes son reglas que se deben seguir para no elaborar un DFD incompleto o lógicamente inconsistente y que se dibuje un DFD grato a la vista:

- ☑ **Escoger nombres con significado para los procesos, flujos, almacenes y terminadores:** es importante etiquetar con precisión el proceso, de modo que quienes leen el DFD puedan confirmar que se trata de un modelo preciso. Si el proceso lo hace una sola persona se recomienda identificar el papel o función que la persona está representando y no su identidad o nombre.

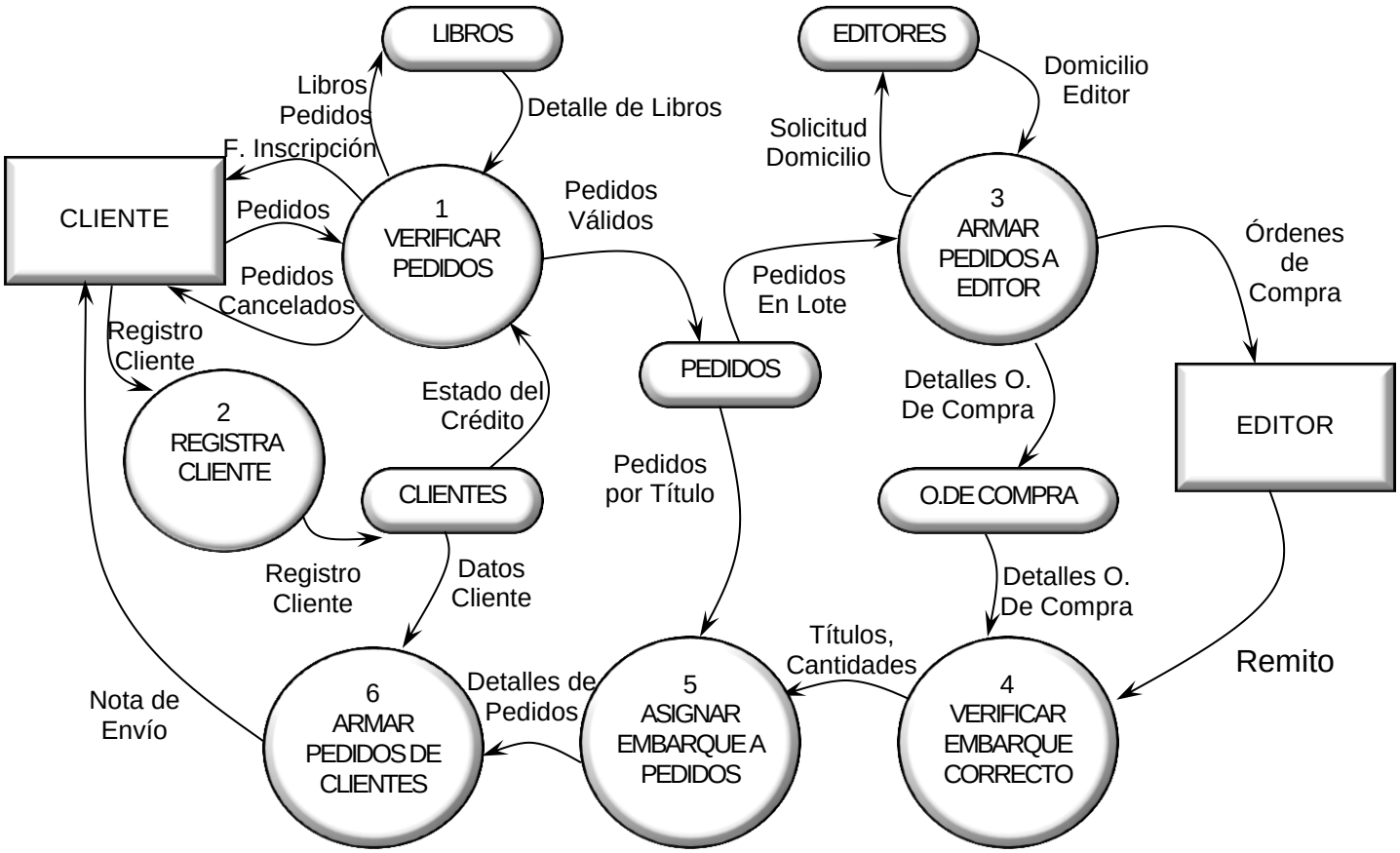
- ☑ **Numerar los procesos:** es una forma conveniente de referirse a los procesos cuando se discute sobre un DFD (decir burbuja 1 es más fácil que EDITAR TRANSACCIONES y REPORTAR ERRORES), y es más importante el hecho de que los números se convierten en base para la numeración jerárquica existen DFD de varios niveles.

- ☑ **Redibujar el DFD tantas veces como sea necesario estéticamente:** tendrá que dibujarse y volverse a dibujar, ser técnicamente correcto, ser aceptable para el usuario y estar lo suficientemente bien dibujado para mostrarlo a la dirección de la organización.

☑ **Evitar los DFD excesivamente complejos:** el diagrama debe ser fácilmente entendido, y placentero a la vista, el DFD debe caber cómodamente en una hoja normal.

☑ **Asegurarse de que el DFD sea internamente consistente y que también lo sea con cualquier DFD relacionado con él:** evitar burbujas que tengan entradas pero no salidas (sumideros infinitos), evitar burbujas de generación espontánea (tienen salidas sin tener entradas), tener cuidado con los flujos y procesos no etiquetados, con los almacenes de solo lectura o solo escritura (deben tener entrada tanto como salida, excepto los externos que sirven de interfaz entre el sistema y algún terminador externo).

Ejemplo DFD



DFD por Niveles

Es conveniente organizar el DFD global en una serie de niveles de modo que cada uno proporcione sucesivamente más detalles sobre una porción del nivel anterior.

El DFD del primer nivel consta sólo de una burbuja, que representa el sistema completo, los flujos de datos muestran las interfaces entre el sistema y los terminadores externos, junto con los almacenes externos que pueda haber. Es el DFD que sigue del Diagrama de Contexto, se conoce como la figura 0, y representa la vista de más alto nivel de las principales funciones del sistema, al igual que sus principales interfaces, cada una de estas burbujas debiera numerarse para una referencia conveniente, los números también sirven como una manera adecuada de relacionar una burbuja con el siguiente nivel del DFD que la describe más a fondo.

DICCIONARIO DE DATOS (DD)

El Diccionario de Datos es un listado organizado 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.

Define datos haciendo lo siguiente:

- ☑ Describe el significado de los flujos y almacenes que se muestran en los DFD.
- ☑ Describe la composición de agregados de paquetes de datos que se mueven a lo largo de los flujos, es decir paquetes complejos (domicilio de un cliente) que pueden descomponerse en unidades más elementales (ciudad, estado y código postal).
- ☑ Describen la composición de los paquetes de datos en los almacenes.
- ☑ Especifica los valores y unidades relevantes de piezas elementales de información en los flujos de datos y en los almacenes de datos.
- ☑ Describe los detalles de las relaciones entre almacenes que se enfatizan en un diagrama de entidad-relación.

Notación del Diccionario de Datos

En la mayoría de los sistemas en los que se trabaja, los paquetes de datos o elementos de datos, serán lo suficientemente complejos y será necesario describirlos. Los elementos complejos de datos se definen en términos de elementos más sencillos, y los sencillos en términos de los valores y unidades legítimos que pueden asumir. Se vuelve tedioso describir la composición de los elementos de datos en una forma narrativa. Es necesario adoptar una notación concisa y compacta (como los diccionarios normales de palabras ordinarias). El esquema más usado es el siguiente:

= : Está compuesto de... Ej: A=B+C....A compuesto de B y C.
+ : y Ej: Idem anterior
() : Optativo. Ej: Nombre = Nombre + (2° Nombre) + Apellido.
{} : Iteración. Ej: Solicitud = Nombre del Cliente + Domicilio de Envío + {Artículo}
[] : Selección de Alternativas. Ej: Sexo = [Masculino | Femenino]
| : Separa opciones alternativas
****** : Comentario.
@ : Identificador (Campo Clave) para un almacén.
.
Ej.: Nombre Completo = Titulo de Cortesía + Nombre + (Segundo Nombre) + Apellido

Título de Cortesía = [Sr. | Srta. | Sra. | Dr. | Profesor]

Nombre = {Carácter Válido}

Segundo Nombre = {Carácter Válido}

Apellido = {Carácter Válido}

Carácter Válido = [A - Z | a - z | 0 - 9 | ‘ | - | |]

Para definir un dato se comienza con el símbolo **=**, y se lee "se define como, significa, o está compuesto de…". Se debe incluir el significado del dato dentro del contexto de la aplicación del usuario, usando la notación ******; se debe incluir la composición del dato (si se compone de partes elementales con significado) y también los valores que puede tomar el dato.
Por ejemplo:

Peso = *peso del paciente al ser admitido al hospital*
unidades: Kilogramos; gama 1-200

Estatura = *estatura del paciente al ser admitido al hospital*
unidades: centímetros; escala: 20-200

Hay términos que se definen por si solos, es decir, cuyo significado es universal para todos los sistemas de información, o que no necesitan mayor aclaración, en estos casos no se necesita un comentario narrativo, se puede usar la notación ** para indicar sin comentarios cuando estos datos se definan solos.

Por ejemplo:

Peso Actual= **
unidades: libras; escala: 1-400
Sexo = **
valores: [M | F]

La notación de iteración "**{ }**" se usa para indicar la ocurrencia de un componente de un dato, y se lee como "cero o más ocurrencias de…". En muchas ocasiones se querrá indicar los límites inferior y superior de la iteración.

Es correcto especificar sólo el límite inferior, sólo el superior, ambos, o ninguno, lo cual se puede indicar así: a = 1 {b}; a = {b} 10; a = 1 {b} 10; a = {b}.
Solicitud = nombre del cliente + domicilio de envío + 1 {artículos} 10

El alias, es una alternativa de nombre para un dato, cuando se insiste en utilizar nombres distintos para decir lo mismo, el alias se incluye en el diccionario de datos para que esté completo, y se relaciona con nombres primarios u oficial del dato.

Por ejemplo:
Comprador = * alias de cliente *

Como mostrar el Diccionario de Datos

El diccionario de datos lo crea el analista durante el desarrollo del modelo del sistema, pero el usuario debe ser capaz de leerlo y entenderlo para poder verificar el modelo.

"Construir un Diccionario de Datos es una de las labores más tediosas, y largas, del análisis de sistemas, pero también es una de las más importantes: *sin un diccionario formal que defina el significado de los términos, no se puede esperar precisión*".

ESPECIFICACIONES DE PROCESO (EP)

Es la descripción de qué sucede en cada burbuja primitiva de nivel más bajo en un Diagrama de Flujo de Datos. Define lo que debe hacerse para transformar entradas en salidas, es una descripción detallada de política de negocios del usuario que cada burbuja lleva a cabo.

Existe una variedad de herramientas para producir una Especificación de Proceso: Tablas de Decisiones, Lenguaje Estructurado (español, inglés, etc.), Pre/Post - Condiciones, Diagramas de Flujo, Diagramas de Nassi / Shneiderman, etc. Se puede usar cualquier método mientras cumpla tres requisitos cruciales:

1. La Especificación del Proceso debe expresarse de una manera que puedan verificar tanto el usuario como el analista (razón por la cual se evita el lenguaje narrativo).
2. El Proceso debe especificarse en una forma que pueda ser comunicado efectivamente al público amplio que esté involucrado.
3. No debe imponer o implicar decisiones de diseño e implantación arbitrarias. El analista debe especificar la esencia del proceso y no cómo se lleva a cabo hoy en día.

Las Especificaciones de Proceso sólo se desarrollan para los procesos de más bajo nivel en un conjunto de diagramas por niveles en un DFD. Las tres herramientas principales de especificación de proceso son:

- ☑ Lenguaje Estructurado (español, inglés, etc.)
- ☑ Pre/Post Condiciones
- ☑ Tablas de Decisión

Lenguaje Estructurado

Es un subconjunto de todo el idioma con importantes restricciones sobre el tipo de frases que pueden utilizarse y la manera en que pueden juntarse dichas frases. Su propósito es hacer un balance razonable entre la precisión del lenguaje formal de programación y la informalidad del lenguaje cotidiano. Por ejemplo, una frase puede ser una ecuación algebraica como X = (Y * Z) / (Q + 14) o una frase imperativa que consista en un verbo y un objeto como por ejemplo: CALCULAR X = (Y * Z) / (Q + 14).

El lenguaje estructurado permite que se combinen frases en unas cuantas formas limitadas que se toman de las construcciones acostumbradas de la programación estructurada.

Pre/Post Condiciones

Son una manera conveniente de describir la función que debe realizar el proceso, sin decir mucho acerca del algoritmo o procedimiento que se utilizará, siendo útil cuando:

- ☑ El usuario tiende a expresar lo ejecutado por un proceso en términos de un algoritmo;
- ☑ El analista está seguro de que existen muchos algoritmos distintos que podrían usarse;
- ☑ El analista desea que el programador explore varios de estos algoritmos.

Un ejemplo de especificaciones de proceso escritas con este enfoque es:

ESPECIFICACIONES DEL PROCESO 3.5: CALCULAR EL IMPUESTO SOBRE VENTAS

Precondición 1
Ocurre DATOS-VENTA con TIPO-ITEM que corresponde con CATEGORIA-ITEM en CATEGORIAS-IMPUESTO

Postcondición 1
IMPUESTO-SOBRE-VENTA se hace igual a MONTO-VENTA * IMPUESTO

Precondición 2
Ocurre DATOS-VENTA con TIPO-ITEM que no concuerde con CATEGORÍA-ITEM en CATEGORIA-IMPUESTO Postcondición 2

Postcondición 2
Se genera MENSAJE-ERROR

Las **Precondiciones** describen lo que debe darse antes de que el proceso pueda comenzar a ejecutarse. Describen lo siguiente:

- Qué entradas se encuentran disponibles,
- Qué relación debe existir entre las entradas,
- Qué relaciones deben existir entre entradas y almacenes de datos,
- Qué relaciones deben existir entre almacenes o dentro de un almacén dado.

Las **Postcondiciones** describen que debe darse cuando el proceso ha concluido. Describen lo siguiente:

- Las salidas que generará o producirá el proceso,
- Las relaciones que existirán entre los valores de salida y los valores originales de entrada,
- Las relaciones que existirán entre valores de salida y los valores en los almacenes,
- Los cambios que se hayan dado en los almacenes.

Tablas de Decisión

Si el proceso debe producir alguna salida o tomar alguna acción basada en decisiones complejas es preferible una tabla de decisiones, si las decisiones se basan en diversas variables distintas, por ejemplo: datos de entrada, y si dichas variables pueden tomar diversos valores, entonces la lógica expresada por el lenguaje estructurado o por las pre/post condiciones probablemente sea tan compleja que el usuario no la comprenderá.

Una tabla de decisiones se crea listando todas las variables relevantes o entradas y todas las acciones relevantes en su lado izquierdo, las variables y acciones están separadas por una línea horizontal gruesa.

A continuación se lista en columnas separadas cada combinación posible de valores de las variables, cada columna usualmente se conoce como regla. Una regla describe una acción o acciones que deben llevarse a cabo para una combinación específica de valores de las variables. Por lo menos debe especificarse una acción para cada regla.

Es preferible expresar variables binarias (V/F). Si existen N variables con valores binarios, existirán 2 a la N reglas distintas. Caso contrario se procede de la siguiente manera:

Total de Reglas= Posibilidades Var.1 X Posibilidades Var.2 X …Posibilidades Var.n

En el Ejemplo: Total de Reglas= 2 x 2 x 2 = 8

Compra Anual > $ 100.000?	S	S	S	S	N	N	N	N
Buena Historia de Pagos?	S	S	N	N	S	S	N	N
Más de 5 años trabajando con la Empresa?.	S	N	S	N	S	N	S	N
Ningún Descuento				X				X
Descuento 5 %			X			X	X	
Descuento 10 %		X			X			
Descuento 25 %	X							

Deben discutirse cada regla con el usuario para asegurarse que se ha identificado la acción o acciones correctas para cada combinación de variables. La ventaja del enfoque de la tabla de decisiones es que el analista se puede concentrar en una regla a la vez y de que no implica ninguna forma particular de implantación habiendo total libertad para su elección.

DIAGRAMA DE ENTIDAD – RELACIÓN (DER)

Es un modelo de red que describe con un alto nivel de abstracción la distribución de datos almacenados en un sistema. En un sistema las estructuras de datos y las relaciones pueden ser tan complejas que es deseable enfatizarlas y examinarlas mediante un modelo independiente del proceso que se llevará a cabo. Este modelo se muestra a los usuarios ejecutivos de mayor nivel en una organización quienes generalmente no están interesados en los detalles funcionales cotidianos del sistema, sino que se preocupan más por los datos que se requieren para manejar su negocio, quién los tiene y quién tiene acceso a ellos.

Componentes de un DER

Tipos de Objetos: colección o conjunto de cosas cuyos miembros individuales pueden identificarse de manera única por algún medio. Juegan un papel necesario dentro del sistema pudiendo describirse por uno o más datos.
El objeto es algo material y el tipo de objeto es su representación en el sistema. Se denotan con un sustantivo singular Ej: CLIENTE, COMPRA, ARTÍCULO, etc.

Relaciones: Representan conjunto de conexiones entre objetos. Cada instancia de la relación representa una asociación entre cero o más ocurrencias de un objeto y cero o más ocurrencias del otro. Una relación puede conectar dos o más instancias del mismo objeto.

Ejemplo:

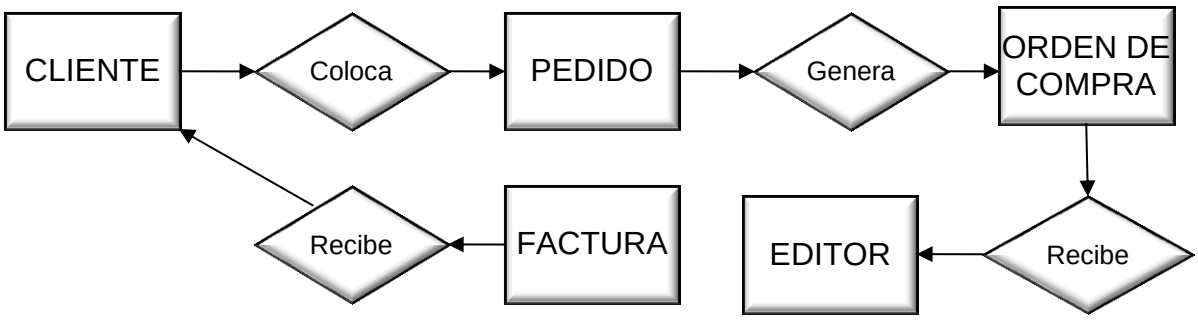


DIAGRAMA DE TRANSICIÓN DE ESTADOS (DTE)

Enfatiza el comportamiento dependiente del tiempo del sistema (sistemas de tiempo-real).

Notación de los DTE

Los principales componentes son Estados y Flechas que representan los Cambios de Estado.

Estados del Sistema: Cada rectángulo representa un estado en el que se puede encontrar el sistema, un estado es un conjunto de circunstancias o atributos que caracterizan a una persona o cosas en un tiempo dado, forma de ser, condición.

Los

un

Cambios de Estado válidos se muestran conectando pares relevantes de estados con una flecha. Cualquier estado puede llevar número arbitrario de estados sucesores.

La mayoría de los sistemas tienen un estado inicial y un estado final reconocibles. El estado inicial suele ser el que se dibuja en la parte superior del diagrama y lo identifica la flecha "desnuda" no conectada a otro estado, el estado final suele ser el que se dibuja en la parte inferior, lo que lo identifica es la ausencia de una flecha que salga de él. Un sistema puede tener un solo estado inicial pero puede tener múltiples estados finales.

Las condiciones que causan un cambio de estado y las acciones que el sistema toma cuando cambia se muestran junto a la flecha que conecta dos estados relacionados.

Construcción del Diagrama de Transición de Estados

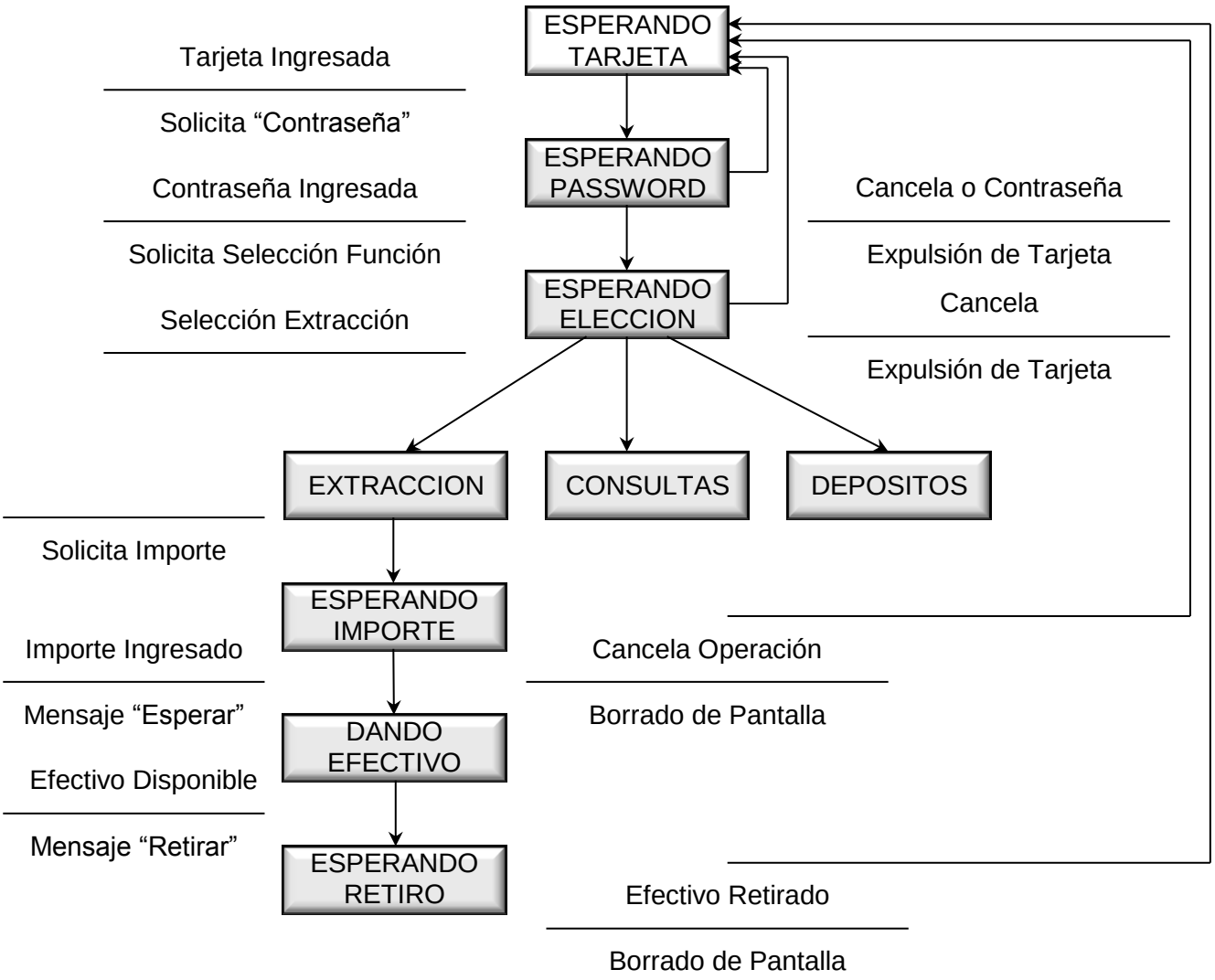
Pueden seguirse dos enfoques:

- Comenzar por identificar todos los posibles estados del sistema y representar cada uno por separado y luego explorar todas las conexiones con significado entre las cajas (cambios de estados).
- Comenzar por el estado inicial y luego metódicamente ir siguiendo un camino hasta el o los restantes, luego del o los estados secundarios, proseguir a los terciarios, etc.

Al terminar su construcción, deben seguirse las siguientes reglas para verificar la consistencia:

- ☑ ¿Se han definido todos los estados?
- ☑ ¿Se pueden alcanzar todos los estados? ¿Se han definido estados que no tengan caminos que lleven a ellos?
- ☑ ¿Se puede salir de todos los estados? Excepto los finales, todos los demás estados deben tener un sucesor.
- ☑ En cada estado ¿el sistema responde adecuadamente a todas las condiciones posibles? ¿Ocurren condiciones normales, condiciones inesperadas?

Ejemplo:



Relación del DTE con los demás componentes del modelo

El DTE puede usarse por sí solo como herramienta de modelado. Sin embargo puede y en general debiera ser utilizado en conjunto con otras herramientas. El DTE representa una especificación de proceso para una burbuja de control en un DFD.

BALANCEO DE MODELOS

Una especificación estructurada se dice que está balanceada cuando todas las herramientas se han verificado entre sí para asegurar su consistencia. Las reglas son:

Del DFD y el DD

- Cada flujo de datos y cada almacén deben estar definidos en el diccionario de datos.
- De manera inversa, cada dato y cada almacén que se define en el diccionario de datos debe aparecer en alguna parte del DFD.

Del DFD y la ESPECIFICACIÓN DE PROCESO (EP)

- Cada burbuja del DFD debe asociarse con un DFD de nivel inferior o con una EP, pero no ambos, ya que el modelo sería redundante.
- Cada EP debe tener una burbuja de nivel inferior asociada en el DFD.
- Las entradas y salidas deben coincidir en el DFD y deben ser evidentes en las EP.

De ESPECIFICACIONES de PROCESO (EP) con el DFD y el DD

Cada referencia de un dato en las EP debe seguir una de las siguientes reglas:

- Coincidir con el nombre de un flujo de datos o almacén conectado a la burbuja descripta por la especificación de procesos o,
- Ser un terminador local, definido explícitamente en la especificación de proceso o,
- Aparecer en una entrada del DD para un flujo o almacén conectado con la burbuja.

Del DD con el DFD y las ESPECIFICACIONES DE PROCESO (EP)

- Cada entrada del DD debe tener referencias en una EP, un DFD, u otro DD.

Del DER con el DFD y las ESPECIFICACIONES DE PROCESO (EP)

- Cada almacén del DFD debe corresponder con un tipo de objeto, una relación o una combinación de un tipo y una relación, es decir un tipo asociativo de objeto, en el DER.
- Los nombres de objetos en el DER y de almacenes de datos en el DFD deben coincidir.
- Las entradas del DD deben aplicarse tanto al modelo del DFD como al de DER.

Del DFD y el DTE

- Cada burbuja de control del DFD se asocia con un DTE como su especificación de proceso, de manera similar cada DTE en el modelo global del sistema debe asociarse con una burbuja o proceso de control del DFD.
- Cada condición del DTE debe corresponder con un flujo de datos de entrada al proceso de control asociado con el DTE (y viceversa).
- Cada acción en el DTE debe corresponder con un flujo de control de salida del proceso de control asociado con dicho diagrama (y viceversa).