Método SSA

1 SSA. CONCEPTOS BASICOS

Este resumen del método SSA está elaborado a partir del libro DeMarco T., "Structured analysis and system specification", Yourdon Press (Prentice Hall) (1979), con la intención de recoger, esperamos que fielmente, las ideas más importantes, y añadiendo alguna otra que consideramos útil para los alumnos.

1.1 Definición:

El Structured Systems Analysis (SSA) es un método clásico de especificación de los requerimientos del Software, propuesto en la segunda mitad de los años 70. Estudiaremos la versión de DeMarco T., aunque existe otra versión bastante difundida de Gane y Sarson.

Unido al SSD (Structured System Design) constituye toda una metodología de desarrollo de Software, que también contempla las labores de Ingeniería de Sistemas.

A pesar de que ya han transcurrido años desde su proposición el SSA constituye un punto de referencia muy importante en este campo, un método muy conocido por los profesionales al que dan soporte varias herramientas CASE, y el origen de una familia, la de los métodos estructurados, que está presente en campos como el de los sistemas en tiempo real y la programación orientada a objetos.

1.2 Contenido:

La proposición del método SSA, al igual que su aplicación en un proyecto, pretende alcanzar los siguientes objetivos:

- Disponer de un modelo lógico (independiente de la implementación) del sistema, inteligible para el usuario, y gráfico en la mayor medida posible, antes de proceder a implementarlo.
- Emplear un método efectivo de descomposición funcional durante el análisis.
- Dar cuenta de las características tanto lógicas como físicas de los sistemas, discerniéndolas adecuadamente.
- Obtener un documento de especificación modificable, esto es flexible para el cambio, mantenible.

El SSA hace uso de herramientas:

- Diagramas de flujo de datos **(DFD)**
- Diccionario de datos **(DD)**
- Otras: lenguaje natural estructurado, tablas y árboles de decisión.

El SSA propone un procedimiento de trabajo (ver diagrama de estructura de la fase de análisis), y un ciclo de vida (ver diagrama correspondiente), una concreción del modelo de las fases.

1.3 Contenido del Documento de Especificación:

- Recopilación de los DFDs
- Diccionario de datos
- Recopilación de descripciones de actividades (mini-especificaciones de las primitivas funcionales)

Este último aspecto (al menos así lo recomendamos nosotros) suele recogerse dentro del diccionario de datos, y no en un apartado específico.

Posteriormente, cuando estudiemos el proceso de obtención de modelos, veremos que la documentación es algo más compleja. De hecho, estos documentos podrían

encajar fácilmente en cualquiera de los formatos convencionales de documento de especificación. No sería necesario indicar. que un índice, una breve introducción y un glosario nunca estorban.

2 DESCOMPOSICION FUNCIONAL EN EL SSA

La descomposición funcional se logra en el SSA a través de los DFD,

2.1 DFD, definición y características:

Los DFD son una herramienta de representación (modelización) que representan los sistemas en términos de redes que muestran los componentes de los sistemas y sus interfaces (el flujo de información entre unos y otros, y con el exterior del sistema} -

Las características más importantes son

- se trata de una notación gráfica
- representa del flujo de información, no el flujo de control, ni la estructura organizativa.
- permiten la descomposición en submodelos.
- pueden ser inteligibles para el usuario, por lo que éste podrá revisarlos, y servirán para facilitar nuestra comunicación con el mismo.

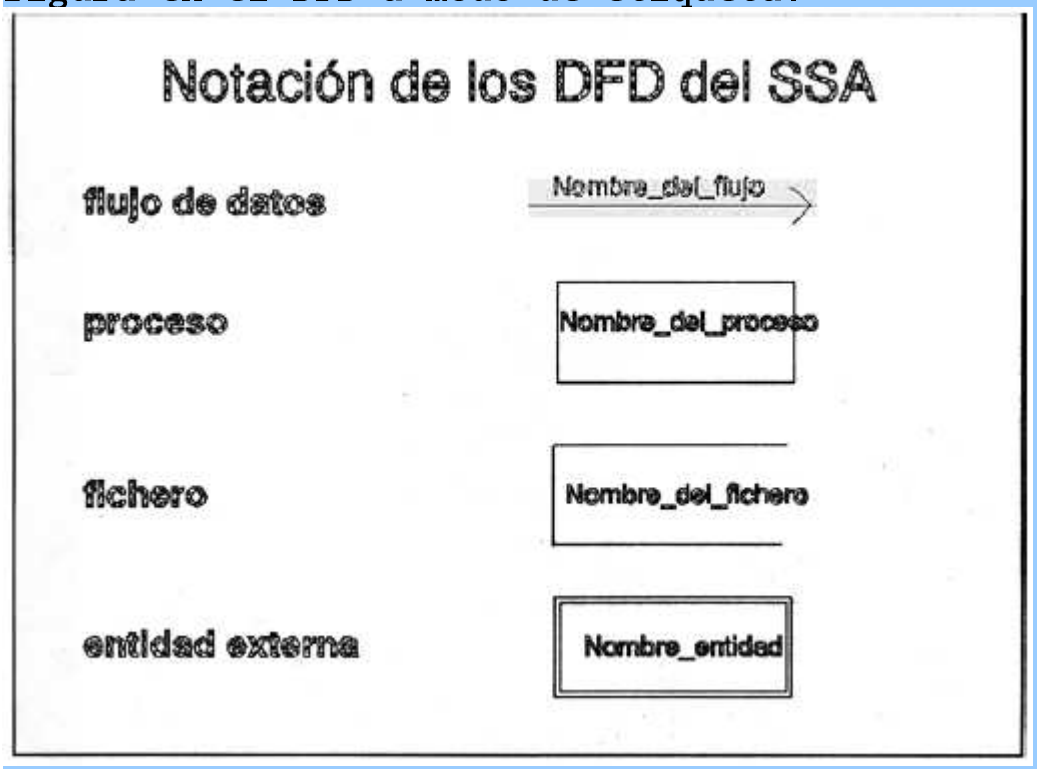
2.2 DFD, elementos y reglas:

En un DFD aparecen elementos de los tipos siguientes:

- **flujos de datos:** indican flujo de información.
- **procesos** (o actividades): trasforman la información que les llega a través de los flujos de entrada en la información que sale a través de los flujos de salida.
- **ficheros:** son almacenes de información en los que ésta se guarda a la espera de un procesamiento posterior.
- **entidades externas:** son fuentes o sumideros de

información (podría darse el caso de alguna mixta), exteriores al sistema que emiten o reciben la información que fluye a través de la interface externa del sistema. Normalmente están asociadas a personas, organizaciones, ..

Cada elemento tiene asociado un nombre, que figura en el DFD a modo de etiqueta.



Las reglas que deben respetarse en la elaboración de los DFD son:

- Los nombres deben ser unívocos, en el conjunto de diagramas de un sistema.
- Los flujos pueden converger o divergir.
- Los procesos y ficheros no pueden poseer sólo flujos de entrada, ni tampoco de salida. (Un elemento de estas características debería reflejarse como entidad externa)
- Los flujos no pueden contener información de control.

2.3 Recomendaciones:

2.3.1Sobre la elaboración de los DFD:

- La mentalidad con que debemos abordar la realización de un **DFD es** la de **"interrogar a los datos".**
- Debemos preocupamos sistemáticamente antes de los flujos de información que de los procesos.
- Es conveniente dar nombre primero a los flujos y después a los procesos.
- **El** proceso a seguir podría ser el siguiente:

  1) Identificar todos los flujos netos, de entrada y de salida, del diagrama y dibújelos en su periferia.

  2) Intentar conectar las entradas con las salidas, y las salidas con las entradas, dejándose guiar por la información procedente del usuario, intentando reflejar el flujo de información tal como es más que preguntarse porqué es así.

  Iremos:

  - reflejando los flujos de información que aparezcan en el sistema,
  - situando procesos (sin darles nombre en un principio) donde sea necesario procesar información para conectar unos flujos con otros,
  - introduciendo los ficheros (almacenes temporales de información) que existan en el sistema.

  3) Seguir el camino desde las entradas a las salidas, desde las salidas a las entradas, y desde el centro a la periferia:

- Examinando los flujos, su contenido, y preguntándonos:
  - ¿Qué necesito para elaborar este contenido?
  - ¿De dónde proceden sus componentes?
  - ¿Puedo elaborar (mediante un proceso) la información de algún otro flujo para obtener la
    de este?
- Examinando las cajas blancas (procesos sin nombre), que comienzan a proliferar, preguntándonos si es necesaria la información de algún otro flujo para obtener las salidas del proceso a partir de las entradas. Podremos ir dando nombre, en términos de sus entradas y salidas, a los procesos.

  Estos recorridos nos llevarán a modificar el diagrama
  convenientemente.

4) Prepararse para volver a empezar. No nos extrañe que:

- exista algún flujo de entrada desconectado del resto del diagrama (lo eliminaríamos sin más),
- exista alguna zona dentro del diagrama desconectada del resto del mismo (si no es una zona de interés la eliminaríamos)

2.3.2Sobre las denominaciones:

- Dar a los flujos nombres significativos, que reflejen su contenido en información y, si es relevante, el estado de la misma.

  Por ejemplo:

  -Numero_de_cuenta refleja el contenido del

flujo.

-Numero_de_cuenta- **validado** no sólo refleja el contenido, sino una información relevante:
que ya ha sido comprobado el número de cuenta.

Habrá que poner especial esmero en la denominación de los flujos de la interface del sistema.

- El estilo de denominación del ejemplo anterior, u otro equivalente **(NumeroDeCuenta)**, a base de frases con palabras delimitadas es más conveniente que el uso de abreviaturas (NUMCUEN).
- Un proceso bien concebido puede describirse fácilmente en términos de sus entradas y salidas (Al fin y al cabo no hace más que trasformar unas en otras). Así, es recomendable nombrar los procesos haciendo referencia a sus entradas y salidas. (Validación_de_Numero_de_cuenta)

2.3.3Sobre los flujos de los ficheros:

- A veces, asociadas a las entradas/salidas de los ficheros, existen salidas/entradas triviales, que no es necesario reflejar. (Por ejemplo, una modificación de un item de un fichero requiere una lectura previa, que no siempre reflejaremos).
- veces, en ficheros simples, sólo existe un flujo de entrada, y otro de salida, con el mismo contenido que los item del fichero. En este caso bastará con dar nombre al fichero y dejar sin nombrar los dos flujos, a los que se pondrá el mismo nombre.

2.3.4 Otras recomendaciones:

- Basta con que los **DFD** reflejen el comportamiento de los sistemas cuando están en un régimen estacionario; no es necesario ni conveniente reflejar los flujos de información correspondientes a los inicios y

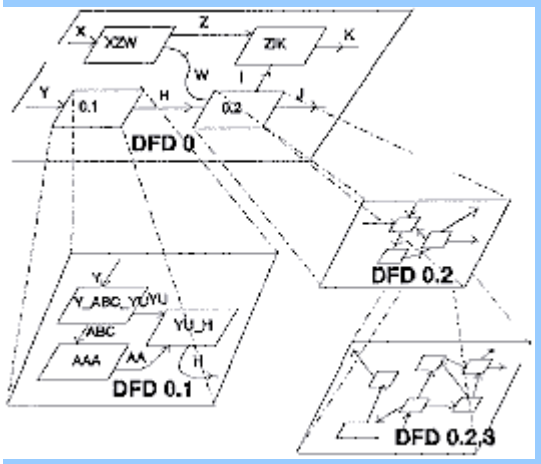
terminaciones del funcionamiento de los sistemas.

- No es necesario reflejar exhaustivamente los caminos de flujo asociados al tratamiento de errores.
- Ya hemos indicado que los DFD no pretenden modelizar el flujo de control, pero si no ponemos cierto cuidado, sobre todo en las primeras aplicaciones del método, nuestros diagramas terminarán conteniendo flujo de información de control. Los flujos de este tipo hacen perder su carácter lógico a nuestro modelo del sistema, y pueden condicionar innecesaria y negativamente las labores posteriores de diseño.

2.4 DFD jerarquizados:

No es eficiente trabajar con diagramas muy extensos, en los que aparezcan muchos procesos. Por ello se introduce la notación jerarquizada, que no es más que un árbol de diagramas en el que los procesos complejos se expanden en DFDs, situados en el nivel inmediatamente inferior del árbol, que describe el flujo de información dentro de los mismos (Ver figura).

JERARQUIZACION DE DFDs



En una jerarquía de DFDs se distinguen los siguientes elementos:

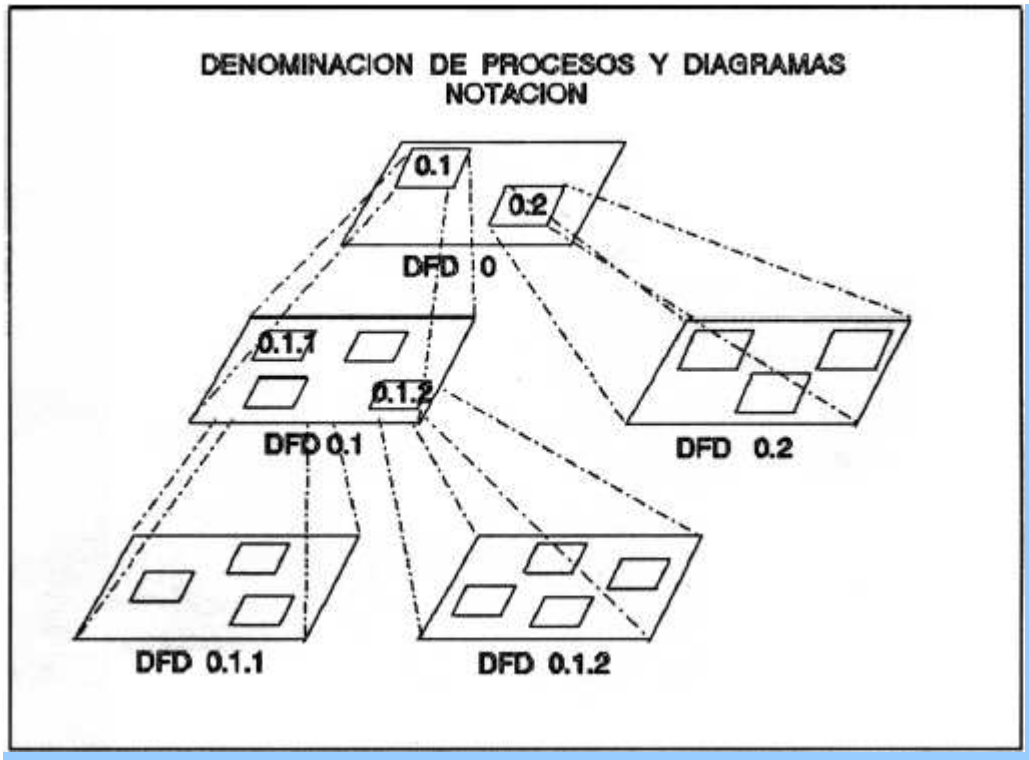
- de contexto: es el diagrama situado en la raíz del árbol, muestra su interacción con el entorno.

- intermedio: como su nombre indica son nodos del árbol no terminales.
- primitivas funcionales: son los procesos a los que no corresponde ningún **DFD** en un nivel inferior.

Al construir jerarquías de DFDs es necesario cumplir las siguientes reglas:

- balanceo de los diagramas: las entradas y salidas netas de un diagrama tienen que coincidir con los flujos de entrada y salida del proceso a que corresponde, en el nivel superior.
- ubicación de los ficheros: un fichero sólo puede aparecer en un diagrama, y ¿en qué nivel?: en el primero en que interaccione con más de un proceso.
- No debe indicarse, sobre los diagramas, la procedencia o destino de los flujos exteriores; esto puede averiguarse mirando en el correspondiente diagrama del nivel superior.

Para poder hacer referencia a los diagramas con facilidad, se recomienda, sobre todo si el método se aplica (esto puede ser complicado) manualmente, hacer uso de la notación que se indica en la figura.



A continuación recogemos algunos **consejos** de interés

- En los diagramas, el número de procesos no debe ser ni muy bajo, lo que multiplicaría el número de diagramas, ni muy alto (un número mayor de siete comienza a ser problemático).
- Las interfaces, como consecuencia de una división en sub-diagramas apropiada, no deben ser muy complejas, y deben guardar una cierta relación con el problema.
- Los diagramas deben ser legibles,
- ¿A partir de qué nivel debe dejar de descomponerse un diagrama?: cuando los procesos que contengan se puedan describir con facilidad; hay quien entiende que esto requiere que sólo presenten un flujo de entrada y otro de salida.
- No es necesario marcar de ninguna forma especial los diagramas terminales.

2.5 Verificación y validación de los DFD:

Aspectos a verificar:

- denominaciones
  - que todos los flujos tengan un nombre asignado,
  - que todos los flujos tengan un contenido conocido,
  - que los nombres de los procesos hagan referencia a sus entradas y salidas, y guarden relación con el contenido de los sub-diagramas que dependan de ellos,
- consistencia
  - el balanceo de los flujos,
  - que un mismo proceso no aparezca más de una vez,
  - que cada proceso pueda elaborar los flujos de salida contando sólo con la información de los flujos de entrada (conservación de los datos).
- ficheros
  - que no aparezcan ficheros que sólo reciben información (sumideros).
- legibilidad
  - que las interfaces sean sencillas,
  - que los nombres de los procesos sean significativos (evitar nombres como PROCESAR_SALIDA),
  - que la descomposición funcional sea uniforme (esto es, que la jerarquía de diagramas esté equilibrada).

3 DICCIONARIO DE DATOS

El Diccionario de datos (DD) completa la información que aparece en los DFD, aportando descripciones de todos los flujos, los ficheros, los datos que contienen unos y otros, las entidades externas, y de los procesos que se considere necesario especificar (todas las primitivas funcionales y algún otro proceso cuyo DFD, que no refleja información de

control, no describa suficientemente).

La idea más importante acerca del DD es que ni éste, por sí sólo, ni el conjunto {DFD,DD} pueden ser redundantes.

DeMarco separa del DD las descripciones (mini-especificaciones funcionales) de los procesos; no creemos que esta separación sea conveniente.

Cualquier herramienta de especificación puede ser válida para describir los items del diccionario: expresiones regulares, lenguaje natural estructurado, notaciones de estados, tablas ó árboles de decisión,...).

4 PROCESO DE OBTENCION DE MODELOS

El SSA propone un proceso de obtención de modelos sucesivos, que da cumplida cuenta de los aspectos lógicos y físicos, del análisis y diseño a nivel de sistema y del análisis de los requerimientos del software. Este proceso es el que, una vez conocidas las herramientas y técnicas básicas del método, estudiaremos a continuación.

4.1 Distinción entre aspectos físicos y lógicos:

Son aspectos físicos todos los que dependen de la implementación, y lógicos los que no dependen de ella.

El término implementación no debe entenderse como relativo a código, implementación software, sino como el resultado de la asignación de funciones a los elementos de que pueda constar un sistema (en el caso de un proyecto que pretenda modificar un sistema en el que ya se hacía uso de aplicaciones informáticas, sí aparecerán aspectos físicos relativos a implementaciones software)

La siguiente relación de aspectos físicos, que podrían darse en un **DFD**, puede ser aclaratoria:

- referencias a nombres de departamentos,
- referencias a lugares concretos,
- nombres de personas,
- ficheros físicos,
- detalles procedimentales,
- dispositivos,
- sistemas informáticos ya existentes.

4.2 Proceso de obtención de modelos:

El SSA propone obtener los siguientes modelos del sistema:

1. modelo físico actual
2. modelo lógico actual
3. modelo lógico nuevo
4. modelo físico nuevo

El modelo físico actual se obtiene plasmando el flujo de información que observemos en el sistema en su configuración actual. Aunque procuraremos evitar reflejar aspectos físicos, el carácter físico de este primer modelo es inevitable:

- aún no poseemos un grado de conocimiento del sistema (de sus aspectos esenciales, del qué) como para abstraer todos los detalles físicos (lo accidental, el cómo);
- la ausencia total de referencias físicas podría dificultar la comunicación con los usuarios, justo en la fase en que más necesitamos obtener información de ellos.

4.3 Obtención del modelo lógico actual:

Se trata de eliminar los aspectos físicos del modelo del sistema actual. Para ello:

1. Re-estructurar la jerarquía de los DFD:

Construiremos DFDs expandidos (resultado de unir en uno varios DFD), en los que eliminaremos duplicidades, y volveremos a agrupar los procesos, en diagramas, con arreglo a criterios lógicos.

Cuando un DFD contiene referencias físicas es fácil que un mismo procesamiento (desde el punto de vista lógico) aparezca duplicado en el conjunto de diagramas, porque se realice en varios departamentos, o en varios lugares, o por distintas personas. Al unir los diagramas pueden detectarse estas duplicidades y ser eliminadas. También puede suceder que el mero hecho de la dispersión "geográfica" de un procesamiento genere necesidades de procesamiento que desaparecen en cuanto se elimina la dispersión; así, cuando un procesamiento se reparte entre varios departamentos surge la necesidad de intercambiar información entre los mismos, lo que provocará comprobaciones y registros en el departamento emisor y receptor, y archivos duplicados en ambos departamentos.

Al eliminar estos detalles no pretendemos que el nuevo sistema no presente ninguna dispersión geográfica, ni modificar su estructura departamental. En algunos casos pueden ser interesantes cambios de este tipo, pero no es ese el problema que ahora nos ocupa. Lo único que pretendemos es adquirir una visión del sistema suficientemente clara, disponer de una base de trabajo que nos permita proponer soluciones que se adapten a las nuevas necesidades, no ser esclavos de la estrategia de implementación vigente, aplicar tratamientos curativos a las enfermedades del sistema y no sólo paliar sus síntomas.

2. Obtener un equivalente lógico a la estructura de ficheros:

Se trataría de estudiar en conjunto los ficheros del sistema y obtener una estructura equivalente, con la misma información, pero una estructura más adecuada. Básicamente se trataría de descomponer las estructuras complejas en estructuras simples, tipo tuplas relacionales, y normalizar la estructura resultante.

Esta normalización es útil, de cara a la obtención del modelo lógico aun cuando no se piense hacer uso de un SGBD.

DeMarco propone una técnica que partiendo de los ficheros permite obtener una estructura de tablas en tercera forma normal (no estoy seguro de que efectivamente se obtenga esta forma normal, aunque sí permite obtener una estructura de ficheros sensata). Otros (Parets) proponen obtener un diagrama entidad-relación equivalente, a la estructura de ficheros, y realizar un diseño relacional convencional a partir de éste. Este diagrama E-R y los resultados del diseño relacional deberían añadirse a la documentación; también podrían figurar referencias a las entidades del E-R en el DD.

Finalmente, si lo que se pretendía era simplemente mejorar (quitarle características físicas) la estructura de ficheros, sin que se vaya a hacer un uso posterior de un SGBD relacional, las tablas relacionales aparecerán reflejadas como ficheros independientes. Si, en cambio, se va a hacer uso posterior de un SGBD puede resultar interesante reflejar un sólo fichero, la base de datos, que aparecerá en los primeros niveles de la jerarquía, y de la que saldrán y entrarán multitud de flujos correspondientes a las tuplas. No se extrañe el lector de que propongamos la anticipación de un detalle físico del nuevo sistema, la existencia de una base de datos. Este tipo de actuaciones, en general, está proscrito: no deben introducirse características físicas, de la implementación, del nuevo sistema antes de tiempo; y esto, para no condicionar las labores de diseño posteriores. Pero, no nos engañemos, muchas veces

la decisión de hacer uso de un SGBD es una decisión apriorística, en cuyo caso tenerlo en cuenta es de sentido común. Además, aún quedarían detalles físicos por especificar (el diseño físico, la posibilidad de una base centralizada o distribuida, de usar un esquema cliente-servidor,...), cuya elección sigue totalmente abierta.

3. Revisar la estructura resultante, de arriba hacia abajo, eliminando detalles físicos residuales:

No está de más hacernos acompañar de los usuarios en alguno de estos recorridos, rememorando las características físicas eliminadas, para asegurarnos de la completitud del modelo, y familiarizarlos con el nuevo modelo lógico.

4.4 *Obtención del modelo lógico nuevo*

El proceso a seguir es el siguiente:

1. Identificar los cambios a introducir:

Se trata de releer el estudio de viabilidad, donde se indican las mejoras a introducir en el sistema.

2. Redefinir el dominio del cambio:

Hacer un recorrido de la jerarquía de DFDs detectando las partes del sistema que se verán afectadas por los cambios.

Es interesante substituir esas zonas por macro-procesos, porque así conseguimos delimitar visualmente el/los dominio/s del cambio, y determinar el intercambio de información a través de la interface entre estas zonas y el resto del sistema, que no ha de cambiar (la interface tampoco ha de cambiar).

3. Identificar el dominio del cambio:

Ahora el analista tiene manos libres para modificar convenientemente, drásticamente, si es necesario, las zonas de cambio. Esto, por supuesto, a nivel lógico, introduciendo nuevos flujos y procesos, sin anticipar detalles físicos de la nueva implementación del sistema.

Esta labor debe hacerse poniendo en práctica todas las indicaciones dadas anteriormente sobre cómo debe ser una buena descomposición funcional, y cuidando la completitud y coherencia del DD.

Todos los trabajos posteriores se basarán en lo que ahora definamos, por tanto, los modelos anteriores se conservarán sólo a título ilustrativo. Así, sería interesante que el **DD** no reflejara información sobre procesos y flujos ya desaparecidos. No esta claro que esto pueda conseguirse fácilmente, en todos los casos, ni siquiera contando con la ayuda de herramientas automáticas.

4. Verificación y validación:

Deben someterse los diagramas al proceso de verificación y validación expuesto anteriormente (2.5).

4.5 Obtención del modelo físico nuevo:

Ahora se trata de seguir el proceso inverso, revistiendo de detalles físicos el modelo lógico nuevo para obtener una nueva implementación del sistema. Esta, como la anterior son fases creativas, no de análisis sino de diseño: de diseño a nivel de sistema (no de diseño de los subsistemas software).

La estrategia recomendada es la de proponer varias opciones, labor creativa, a la que sucederá una labor "destructiva" de evaluación de opciones (análisis de costos/beneficios, estudios de viabilidad,. ..) y de selección de la más adecuada, o de proponer nuevas opciones si ninguna de las disponibles se considera viable. En todo el

proceso la participación de los usuarios debe ser activa, y en la selección decisiva.