TAREA 5

Diferencia entre CFD y DFD

DFD son usados para describir el flujo de información dentro de un Sistema. Estos pueden representar transformaciones en la información como en el almacenamiento. También trazan la ruta que la información que debe seguir en el sistema de inicio a fin.

CFC son usados para describir la lógica detallada de un proceso de negocio o reglas del negocio. CFC puede fácilmente ilustrar las decisiones dentro del sistema por medio de nodos de decisión que se ramifican in diferentes caminos lógicos.

Qué son las cartas de estructuras?

La Carta Estructurada del Proyecto consiste en un diagrama jerárquico modular basado en una metodología de desarrollo de sistemas TOP-DOWN es decir de lo más general a lo más específico la diferencia es el recorrido jerárquico y modular que se utiliza para su elaboración.

Un Módulo es un subsistema que agrupa funcionalmente programas, objetos, herramientas y, bases de datos según su funcionalidad y objetivos vinculantes. Por ejemplo, el objetivo de un módulo de compras es proveer a la empresa de material necesario para su funcionamiento y así sucesivamente.

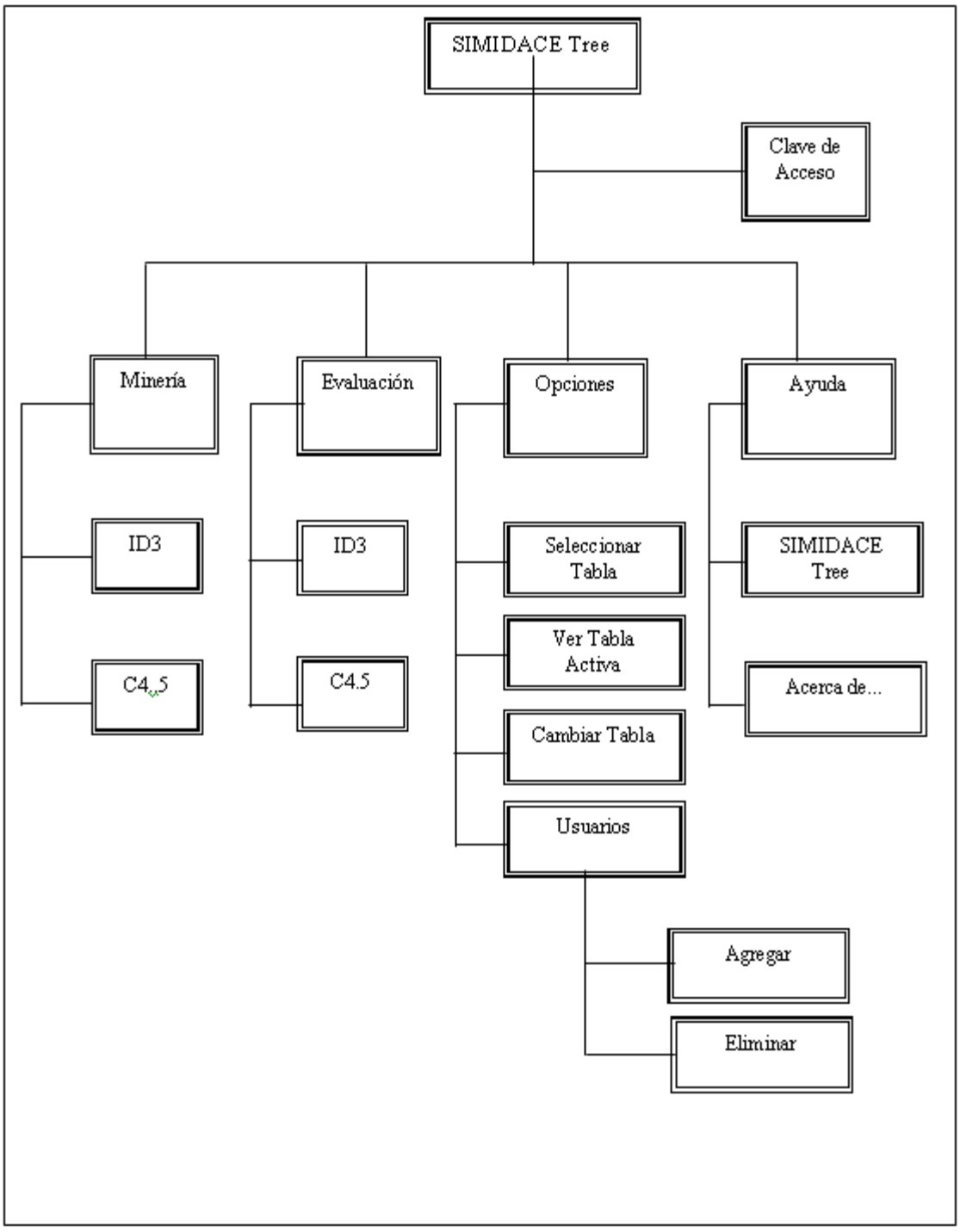
La Carta estructurada del proyecto es denominada también “modelo del producto”. Es importante diseñar la carta estructurada del proyecto antes de comenzar el proceso de diseño de un sistema de software. El software que desarrollamos es un producto, al igual que cualquier otro producto comercializable que requiere un tiempo y proceso de elaboración. Partiendo de un concepto general, se van desglosando los módulos y sub-módulos relacionados hasta llegar a un nivel donde es posible diferenciar las actividades de trabajo. La carta estructurada del proyecto permite distribuir las actividades entre los analistas, desarrolladores, jefes y gerentes involucrados en el desarrollo del proyecto. Esta carta estructurada precede a la elaboración de un Diagrama de Gantt.

La Carta estructurada hace posible que cada participante entienda su función dentro de un contexto integral. Además, es sencillo reconocer las interrelaciones de los módulos y prever el desarrollo de interfaces entre los mismos, cuando se tiene claro el contingente de módulos, sub-módulos y jerarquías. La definición de las bases de datos puede hacerse con mayor claridad cuando tenemos que decidir si la misma entidad es compartida por varios módulos y solo hay que variar los valores de sus claves de acceso o si se trata de entidades separadas según su funcionalidad y la clase de información que contienen.

Presenta el “plano” del sistema propuesto y sirve para:

- hacer participar al usuario
- diseñar funciones detalladas
- diseñar menús
- planificar el desarrollo de programas

- monitorear el desarrollo



Mencione que tipos de cohesión existen con una pequeña definición

La cohesión tiene que ver con la forma en la que agrupamos unidades de software en una unidad mayor. Por ejemplo, la forma en la que agrupamos funciones en una librería, o la forma en la que agrupamos métodos en una clase, o la forma en la que agrupamos clases en una librería, etc.

Se suele decir que cuanto más cohesionados estén los elementos agrupados, mejor. El criterio por el cual se agrupan es la cohesión

Tipos de cohesión:

- Cohesión funcional.

  Se produce cuando agrupamos unidades de software teniendo en cuenta que todas ellas contribuyen a realizar un mismo fin. Es decir, cuando todas las unidades agrupadas, trabajando juntas consiguen un objetivo. En general, es el criterio de agrupación más deseable. Además, entre este tipo de unidades suele haber un acoplamiento relativamente alto, así que mejor que estén juntas.

- Cohesión secuencial.

  Cuando agrupamos unidades que cumplen que los resultados que produce una son los que utiliza otra para continuar trabajando. Es decir, los datos de salida de una sirven de entrada para otras. Es una forma de agrupar muy relacionada con el problema que se está tratando de resolver.

- Cohesión de datos.

  Cuando todas las unidades agrupadas trabajan sobre el mismo conjunto de datos.

- Cohesión lógica.

  Cuando todas las unidades agrupadas realizan trabajo en una misma categoría lógica, pero no necesariamente tienen relación unas con otras. Por ejemplo, librerías de funciones matemáticas... se agrupan simplemente porque realizan cálculos matemáticos, pero no necesariamente tienen relación unos con otros.

- Cohesión temporal.

  Este criterio empieza a ser algo peor. Significa que agrupamos una serie de unidades simplemente porque tienen que ejecutarse más o menos en el mismo periodo de tiempo, pero sin que tengan una relación mayor entre ellas... es decir, sin que contribuyan al mismo fin (funcional), sin que se pasen datos en secuencia (secuencial) y sin que ni tan siquiera trabajen sobre los mismos datos (de datos) ni caen dentro de una misma categoría (lógica). Simplemente, tienen que ejecutarse cerca unas de otras.

- Cohesión casual.

  Pues cualquier criterio que no caiga dentro de los anteriores se considera ya puramente casual. Mejor evitarla, si se puede... más vale tener un criterio, aunque no estemos seguros de que es bueno, que no tener ningún criterio.

Qué es un diccionario de datos?

Es un conjunto de metadatos que contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

Razones para su utilización:

1- Para manejar los detalles en sistemas muy grandes, ya que tienen enormes cantidades de datos, aun en los sistemas más chicos hay gran cantidad de datos.

Los sistemas al sufrir cambios continuos, es muy difícil manejar todos los detalles. Por eso se registra la información, ya sea sobre hoja de papel o usando procesadores de texto. Los analistas más organizados usan el diccionario de datos automatizados diseñados específicamente para el análisis y diseño de software.

2- Para asignarle un solo significado a cada uno de los elementos y actividades del sistema. En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos.

Acoplamiento

El acoplamiento informático indica el nivel de dependencia entre las unidades de software de un sistema informático, es decir, el grado en que una unidad puede funcionar sin recurrir a otras; dos funciones son absolutamente independientes entre sí (el nivel más bajo de acoplamiento) cuando una puede hacer su trabajo completamente sin recurrir a la otra. En este caso se dice que ambas están desacopladas.

Tipos de Acoplamiento

Acoplamiento normal: una unidad de software llama a otra de un nivel inferior y tan solo intercambian datos (por ejemplo: parámetros de entrada / salida). Dentro de este tipo de acoplamiento podemos encontrarnos 3 subtipos, dependiendo de los datos que intercambien las unidades de software. Para más información ver Acoplamiento normal.

Acoplamiento de datos: Dos módulos están acoplados por datos si ellos se comunican por parámetro. Siendo un parámetro una unidad elemental de datos.

Acoplamiento de marca: Dos módulos aparecen acoplados de marca si ellos se refieren a la misma estructura de datos local. Por estructura se entiende un grupo compuesto de datos en vez de argumentos simples.

Acoplamiento de control: Es cuando un modulo pasa a otro indicadores de control. Provoca dependencia de ejecución entre un módulo y otro.

Acoplamiento externo: las unidades de software están ligadas a componentes externos, como por ejemplo dispositivos de entrada / salida, protocolos de comunicaciones, etc.

Acoplamiento común: dos unidades de software acceden a un mismo recurso común, generalmente memoria compartida, una variable global o un fichero.

Acoplamiento de contenido: ocurre cuando una unidad de software necesita acceder a una parte de otra unidad de software.