Índice

INTRODUCCIÓN.

A lo largo del desarrollo de un sistema software desde el estudio de requisitos hasta la implementación y puesta en marcha del mismo, se contemplan una multitud de tareas complejas.

El software desde sus orígenes de desarrollo en los que se realizaban programas de forma poco sistemática, casi artesanal, hasta hoy en día en el que existen metodologías de desarrollo muy definidas, ha evolucionado en gran medida gracias a la ingeniería del software que pretende aplicar los principios de la ingeniería al desarrollo de software de forma que se sistematice la producción del mismo.

Su objetivo es facilitar el mantenimiento del software y aumentar la calidad de éste. La ingeniería del software por tanto estudiará el conjunto de procesos para el desarrollo de productos software según un enfoque ingenieril, considerando, además de la programación, otras actividades previas de análisis y diseño.

Un diseño es un proyecto o plan. Todas las dimensiones de un sistema de información, y en particular la de las funciones o procesos, se pueden describir con distintos niveles de abstracción, que habitualmente se clasifican como conceptual, lógico, físico y de implementación.

El objetivo principal del diseño lógico de funciones es desarrollar la estructura (arquitectura) de programa, así como las relaciones entre los elementos, denominados módulos, que componen esta estructura. El diseño lógico de funciones permite establecer la transición del diagrama de flujo de datos a una descripción de la estructura del programa.

1. DISEÑO LÓGICO DE FUNCIONES.

Los tres principios básicos del diseño estructurado están basados en la descomposición, jerarquía e independencia.

1.1. Principios del diseño estructurado.

1.1.1. Descomposición.

La descomposición es la separación de una función en otras que estuvieran contenidas en la primera.
La descomposición consigue los siguientes objetivos:

- Reducir el tamaño del módulo.
- Hacer el sistema más fácil de entender y modificar.
- Minimizar la duplicidad de código.
- Crear módulos útiles.

Un módulo consiste en una unidad claramente definida y manejable, con interfaces modulares perfectamente definidas. La modularidad mejora la claridad del diseño, facilitando la implementación, la depuración, las pruebas, la documentación y el mantenimiento de un producto software.

1.1.2. Jerarquía.

El objetivo de aplicar una jerarquía de módulos es conseguir separar los módulos que realizan tareas de cálculo y edición de aquellos que toman decisiones y llaman a otros módulos.

1.1.3. Independencia.

Si los módulos individuales son completamente independientes unos de otros, entonces el esfuerzo total implicado en el desarrollo del sistema es una función lineal del número de módulos del sistema.

1.2. Calidad del diseño lógico de funciones: Acoplamiento.

Se define acoplamiento como el grado de interdependencia entre los módulos. Para que se produzca un buen diseño, se debe intentar siempre minimizar el acoplamiento; es decir, hacer los módulos tan independientes unos de otros como sea posible.

Distintos tipos de acoplamiento:

- **Acoplamiento normal**. Dos módulos A y B están normalmente acoplados si:
  - Un módulo A llama a otro B.
  - B retorna el control a A.
- **Acoplamiento de datos:** En el acoplamiento de datos los módulos se comunican mediante parámetros, y cada parámetro constituye una unidad elemental de datos.
- **Acoplamiento por estampado:** Dos módulos están acoplados por estampado si ambos hacen referencia misma estructura de datos.

- **Acoplamiento de control:** Dos módulos están acopla dos por control cuando uno de ellos pasa al otro nódulo elementos de control.
- **Acoplamiento común (global):** Un grupo de módulos están acoplados común mente cuando comparte estructura global de datos.
- **Acoplamiento por contenido**: Dos módulos presentan acoplamiento por contenido si uno hace una referencia al interior del otro.

2. DEFINICIÓN DE FUNCIONES.

Cada módulo se puede considerar un sistema en sí mismo, y, como tal, cabe definirlo mediante un modelo funcional y otro estructural.

2.1. Descripción de interfaces entre módulos del sistema.

La interfaz incluye la comunicación de información de control, así como la de los datos de la aplicación. Cada interfaz entre módulos debe incluir: Los datos comunicados y su formato y el origen y el destino de esos datos.

2.2. Definición de componentes.

Se trata de definir los componentes o unidades básicas de desarrollo, describiendo, de forma breve, su funcionamiento. Para ello se parte de la descripción de las funciones y procesos realizados en la especificación funcional del sistema.

2.3. Calidad de la definición de funciones: Cohesión.

La cohesión estudia la relación que existe entre los elementos de un mismo módulo.

- **Cohesión funcional.** Todos los elementos que componen el módulo están relacionados en el desarrollo de una única función. Es el grado más alto en la cohesión.
- **Cohesión secuencial.** Existe cohesión secuencial cuando el módulo representa el empaquetamiento físico de varios módulos con cohesión funcional.
- **Cohesión comunicacional**. Un módulo con cohesión comunicacional es aquel cuyos elementos o actividades utilizan los mismos datos de entrada y salida.
- **Cohesión procedural**. Este tipo de cohesión se da cuando el módulo tiene una serie de elementos relacionados por un procedimiento efectuado por el código.
- **Cohesión temporal**. Un módulo con cohesión temporal es aquel cuyos elementos están implicados en actividades que están relacionadas en el tiempo.
- **Cohesión lógica y cohesión coincidental.**

3. DESCOMPOSICIÓN MODULAR.

3.1. Análisis de transformación.

El análisis de transformación es un conjunto de pasos de diseño que permiten a un DFD obtenido en la fase de Análisis, con características de flujo de transformación, convertirse en una estructura (template) predefinida del sistema.

3.2. Análisis de transacción.

En muchas aplicaciones software, un dato determina caminos alternativos por los que puede transitar el flujo de información; dependiendo del camino tomado varía la función realizada sobre el dato tratado. En estos casos, el elemento de datos se denomina transacción. Este elemento desencadena otro flujo de datos a lo largo de uno de los muchos caminos.

El centro del flujo de información desde el que emanan muchos caminos de acción (exclusivos entre si) se llama centro de transacción. En un flujo orientado a transacción, el flujo de información a lo largo de un camino de acción puede ser tanto de transformación como de transacción. Los pasos del diseño para el análisis de transacción son similares, y en algunos casos idénticos, a los pasos para el análisis de transformación. La principal diferencia se refiere a la conversión del DFD en la estructura del sistema.

4. TÉCNICAS DESCRIPTIVAS.

4.1. Técnica descriptiva de diseño lógico: el diagrama de estructura de cuadros.

El Diagrama de Estructura de Cuadros, permite definir cuándo, bajo que condiciones y cuántas veces se tienen que realizar los tratamientos identificados en los procesos de los DFD. Los datos se contemplan como la interfaz entre los tratamientos sucesivos.

El paso de la Fase de análisis del sistema al diseño será más fácil si se ha llegado a un nivel de detalle muy bajo en los Diagramas de Flujo de Datos.

La estructura nos da una visión de Arquitectura del sistema. Se trata pues, de no tener ninguna restricción en cuanto al número de objetos, siempre y cuando el dibujo pueda realizarse en una hoja, para no perder la referencia, y en cualquier caso poder explosionar aquella función que se descomponga.

El diagrama está compuesto por tres elementos básicos:

- Módulos.
- Conexiones entre módulos.
- Comunicación entre módulos.

4.2. Técnicas descriptivas de definición de funciones.

4.2.1. Tabla de interfaz.

Los parámetros que se pasan entre módulos se pueden representar en la Tabla de Interfaz, que permite una mejor especificación de los parámetros y sirve de apoyo a los diagramas de estructuras mejorando su claridad (cuando el número de parámetros en la interfaz es mayor que cuatro, el diagrama de estructuras puede resultar confuso). La tabla de interfaz señala, como puede verse en la tabla, para cada llamada:

1. El módulo llamado.
2. Cada parámetro formal.
3. Si el parámetro es de entrada
4. Si el parámetro es de salida.
5. El uso de cada parámetro.
6. El significado de cada parámetro.

4.2.2. Técnica descriptiva: descripción narrada.

La descripción narrada hace referencia al lenguaje natural que se utiliza habitualmente para describir y desarrollar todo tipo de documentos. En este caso el vocabulario no está restringido, lo que hace posible la utilización de términos que no están en el diccionario de datos y por tanto pueden no tener un significado concreto.

4.2.3. Técnica descriptiva: ordinogramas.

También denominados diagramas de flujo de programas. Son representaciones gráficas que muestran la secuencia lógica y detallada de las operaciones que se van a realizar en la ejecución de un programa.

El diseño de un ordinograma debe ser totalmente independiente del lenguaje de programación utilizado en la codificación del algoritmo, evitando hacer cualquier tipo de referencia a la sintaxis del lenguaje.

El diseño de todo ordinograma debe reflejar:

- Un principio o inicio que marca el comienzo de ejecución del programa y que vie ne determinado por la palabra "INICIO".
- Una secuencia de operaciones, lo más detallada posible y siguiendo siempre el orden en el que se deberán ejecutar (de arriba-abajo y de izquierda derecha).
- Un fin que marca la finalización de ejecución del programa y que viene determinado por la palabra "FIN".

4.2.4. Técnica descriptiva: pseudocódigo.

La notación pseudocodificada o pseudocódigo se puede definir como un lenguaje inter medio entre el lenguaje natural o lenguaje humano y el lenguaje de programación seleccionado.

Todo algoritmo representado en notación pseudocodificada deberá reflejar las siguientes partes:

- Cabecera: es el área o bloque informativo donde quedará reflejado el nombre del algoritmo y el nombre del programa al que pertenece dicho diseño.
- Cuerpo: se denomina así al resto del diseño, el cual queda dividido en otros dos bloques:
- Bloque de datos: es el lugar donde deberán quedar descritos todos los elementos de trabajo necesarios en la ejecución del programa.
- Bloque de acciones: en esta zona del diseño se deben describir con la máxima claridad y detalle todas aquellas acciones que el ordenador deberá realizar durante la ejecución del programa.

4.2.5. Técnica descriptiva: diagramas de Chapin o de Nassi-Schneiderman.

Más que una metodología estructurada de programación lo que representa es una forma de representar los bloques estructurados.

Este método no facilita, de entrada, la tarea del programador; de hecho le pide una ma yor comprensión. Pero una vez pasada esta barrera, los beneficios que se obtienen de la depuración, documentación y mantenimiento de los programas compensan con creces este esfuerzo inicial.

5. DOCUMENTACIÓN.

En Métrica se ha de elaborar, como resultado de la actividad de diseño técnico del sistema, entre otros, un documento de diseño técnico. En el primer apartado del mismo, Diseño de la Arquitectura del sistema, recoge los resultados generados por las técnicas descriptivas tratadas en este tema:

- Diseño lógico: El diagrama de estructura de cuadros.
- Definición de funciones:
- Tabla de interfaz.
- Descripción narrada.
- Ordinogramas.
- Pseudocódigo.
- Diagramas de Chapin o de Nassi-Schneiderman.

CONCLUSIÓN.

Las especificaciones funcionales son documentos que amplían en detalle un requerimiento de desarrollo, incluyendo una descripción completa de la lógica esperada, los datos y los criterios de prueba, con el fin de que el desarrollador disponga de la información suficiente para que pueda diseñar una especificación técnica, construir el desarrollo y realizar las pruebas técnicas necesarias para asegurar la calidad del desarrollo entregado.

Si bien es necesario un correcto análisis de los requerimientos del negocio, para garantizar la calidad de la especificación funcional son necesarias además otras consideraciones de diseño que se deben tener en cuenta dependiendo del desarrollo requerido.

Además de una explicación clara de la forma en que se espera que funcione el desarrollo, es necesario identificar con claridad el tipo de desarrollo que se requiere y algunos elementos de diseño específicos para cada tipo de modo que el desarrollador pueda plasmar en su desarrollo estas especificaciones de forma precisa y eficaz.