## CAPITULO 28 – METODOS FORMALES

Permiten crear especificaciones sin ambigüedades, más completas y consistentes, utilizando notación matemática. Los métodos formales son ideales para sistemas de seguridad y sistemas críticos para los negocios.

### INVARIANTE DE DATOS

Condición verdadera que contiene una colección de datos a través de la ejecución del sistema

ESTADO: representación del comportamiento observable externamente de un sistema o los datos almacenados a los que el sistema tiene acceso y altera

OPERACION: Acciones en un sistema a medida que lee o escribe datos en un estado.

### BASE DE LOS METODODOS FORMALES: METEMATICA DISCRETA

Notación y heurística asociadas con conjuntos la especificaciones constructivas (operadores de conjuntos, operadores lógicos y sucesiones). Contiene (según el lenguaje)

* **DOMINIO SINTÁCTICO:** simbología relacionada con la notación de conjuntos y el cálculo de predicados.
* **DOMINIO SEMÁNTICO:** el lenguaje expresa los requisitos concisamente.

### IMPORTANCIA:

Reducen drásticamente los errores de especificación y consecuentemente son la base del software que tiene pocos errores.

1. **Sintaxis:** Notación específica
2. **Semántica:** Universo de Objetos para describir sistemas
3. **Conjunto Relaciones:** Reglas para los objetos que satisfacen especificaciones.

LENGUAJES

FORMALES DE

ESPEFICICACIÓN

COMPONENTES

PRIMARIOS

### 10 MANDAMIENTOS:

1. Erigirás la Notación apropiada.
2. Formalizarás pero no en exceso.
3. Estimarás los costos.
4. Tendrás un experto en métodos formales a tu disposición.
5. No abandonarás tus métodos tradicionales de desarrollo.
6. Documentarás suficientemente.
7. No comprometerás los estándares de calidad.
8. No serás dogmático.
9. Probarás, probarás y probarás de nuevo.
10. Reutilizarás.

## 

## CAPITULO 29 – INGENIERIA DEL SOFTWARE DE SALA LIMPIA

Incluye la **corrección del software** a medida que se va **desarrollando**, tratando de **corregir** errores **antes de las pruebas**. Los modelos de análisis y diseño **se crean usando** una representación de **estructura de caja**.

1. Uso explícito control estadístico calidad.
2. Verifica especificación del diseño empleando demostración matemática.
3. Comprobación estadística para descubrir errores de alto impacto.

**CARACTERISTICAS**

**CAJA NEGRA:**

Especifica comportamiento del sistema o parte de un sistema. Responde a estímulos generando una respuesta.

**CAJA DE ESTADO:**

Encapsula datos y operaciones de estado. Representan las entradas y las salidas.

**CAJA TRANSPARENTE:**

Contiene el diseño procedimental de la caja de estados, de forma similar a la programación estructurada.

* Reduce verificación a un proceso finito.
* Mejora la calidad
* Permite a los equipos verificar las líneas de diseño y código.
* Nivel de defectos próximo a cero.
* Escalable.
* Produce mejor código que el test unitario.

**VENTAJAS DE LA**

**VERIFICACIÓN**

**DEL DISEÑO**

### PRUEBA DE SALA LIMPIA

Validar los requisitos del software mediante la demostración de que una muestra estadística de casos prácticos se ha ejecutado con éxito.

**OBJETIVOS DE LAS**

**PRUEBAS DE**

**SALA LIMPIA**

#### PRUEBA ESTADISTICA DE USO

Intenta probar el software de la misma manera que lo haría el usuario.

Para lograr esto, los equipos de prueba de sala limpia (también llamados equipos de certificación) deben determinar la distribución de probabilidad de utilización correspondiente al software. La especificación (caja negra) de cada incremento del software se analiza para definir un conjunto de estímulos (entradas o sucesos) que pueden dar lugar a que el software modifique su comportamiento. Basándose en entrevistas con posibles usuarios, en la creación de escenarios de utilización y en una comprensión general del dominio de la aplicación, se asigna una probabilidad de utilización de cada uno de los estímulos.

#### CERTIFICACION

Implica que la fiabilidad (medida por el tiempo mínimo de fallo, TMDF) podrá ser especificada para cada componente.

Implica 5 pasos:

* Se crean escenarios de uso.
* Se Especifica el perfil de uso.
* Se generan casos de prueba a partir del perfil.
* Se ejecución de pruebas y se registran y analizan los fallos.
* Se cálculo y certifica de fiabilidad.

#### CERTIFICACION DE FIABILIDAD

La certificación para la ingeniería del software de sala limpia requiere la creación de tres modelos:

**MODELO DE MUESTREO:** La prueba de software ejecuta “m” casos de prueba, y se certifica si no se produce ningún fallo o si se produce un número de fallos inferior al especificado.

**MODELO DE COMPONENTES:** Se certifica un modelo basado en n componentes. Permite que el analista determine la probabilidad de que falle el componente antes de finalizar el programa.

**MODELO DE CERTIFICACIÓN:** Se proyecta y certifica la fiabilidad global del sistema.

## CAPITULO 30– INGENIERIA DEL SOFTWARE BASADA EN COMPONENTES

Es un proceso que se centra en el diseño y construcción de sistemas basados en computadoras que utilizan componentes de software reutilizables.

La potencial ventaja es que genera productos de software altamente confiables, mejora la calidad promedio del producto.

Analiza los requisitos del sistema para determinar cuál subsistema se adapta mejor.

### TIPOS DE COMPONENTES

#### CUALIFICADOS

El desempeño, funcionalidad, fiabilidad y facilidad de uso de los componentes está garantizada por los ingenieros de software.

#### ADAPTADOS

Adaptados para modificar características no requeridas o indeseables.

#### ACTUALIZADOS

Sustituye el software según las nuevas versiones de los componentes.

### PROCESO ISBC

Cualifica, Adapta, Ensambla y Actualiza componentes.

### INGENIERIA DEL DOMINIO

Su objetivo es identificar, construir, catalogar, y diseminar, un conjunto de componentes de software que tienen aplicación en el software actual y futuro dentro de un dominio de aplicación en particular.

La meta es establecer mecanismos que permitan a los ingenieros de software compartir dichos componentes durante el trabajo en sistemas nuevos y eficientes.

#### PASOS DEL PROCESO DE ANALISIS DEL DOMINIO

1. Definir el dominio que se investigará
2. Categorizar los elementos extraídos del dominio.
3. Recopilar una muestra representativa de las aplicaciones del dominio.
4. Analizar cada aplicación en la muestra y definir clases de análisis
5. Desarrollar un modelo de análisis para los objetivos.

#### MODELADO ESTRUCTURAL Y PUNTOS DE ESTRUCTURA

El **modelado estructural** supone que en todo dominio de aplicación tiene patrones repetidos que tienen un potencial de reutilización.

Los **puntos de estructura** son una estructura bien diferenciada dentro de un modelo estructural.

1. Abstracciones con un número limitado de instancias dentro de una aplicación.
2. Las reglas que lo rigen deben entenderse fácilmente.
3. Debe implementar ocultamiento de información aislando toda la complejidad.

**CARACTERISTICAS DE**

**PUNTOS DE**

**ESTRUCTURA**

### 

### DESARROLLO BASADO EN COMPONENTES

Ocurre en paralelo a la ingeniería de dominio, esta última proporciona la biblioteca de componentes reutilizables necesarios.

Cualifica, adapta e integra los componentes para su reutilización en un sistema nuevo.

#### CUALIFICACION DE COMPONENTES

Garantiza que un componente candidato llevará a cabo la función necesaria, “encajará” adecuadamente en el estilo arquitectónico especificado para el sistema y mostrará las características de calidad necesarias para la aplicación.

Factores a tener en cuenta:

* La interfaz de programación de aplicaciones (API).
* Las herramientas de desarrollo e integración necesarias para el componente.
* Requisitos de ejecución, entre los que se incluyen la utilización de recursos (por ejemplo, memoria o almacenamiento), tiempo o velocidad y protocolo de red.

#### ADAPTACION DE COMPONENTES

Lo ideal sería que la ingeniería del dominio creara una biblioteca de componentes que pudieran integrarse fácilmente en una arquitectura de aplicaciones.

Para mitigar algunos conflictos de integración, se suele utilizar una técnica de adaptación llamada **encubrimiento de componentes**, que suelen ser las siguientes:

* **Encubrimiento de Caja Blanca**: examina detalles del procesamiento interno del componente.
* **Encubrimiento de Caja Gris**: se aplica cuando la biblioteca de componentes proporciona un lenguaje de extensión de componentes.
* **Encubrimiento de Caja Negra**: requiere la introducción de un pre o post procesamiento en la interfaz de componentes para eliminar o enmascarar conflictos.

#### COMPOSICIÓN DE COMPONENTES

Proporciona un modelo para la coordinación de componentes y servicios específicos, que hacen posible coordinar unos componentes con otros y llevar a cabo tareas comunes.

Existe un conjunto de cuatro “ingredientes arquitectónicos” que se debería presentar para lograr la composición de componentes:

1. **Modelo de Intercambio de Datos:** Mecanismos que permiten a los usuarios y a las aplicaciones intercambiar y transferir datos, y que deberían estar definidos para todos los componentes reutilizables.
2. **Automatización:** Para facilitar la interacción entre componentes reutilizables se deberían de implementar herramientas, macros y guiones.
3. **Almacenamiento Estructurado:** Los datos heterogéneos dentro de un documento compuesto deben estar organizados para poder acceder a ellos como si fuera una sola estructura.
4. **Modelo de Objetos Subyacente:** Asegura que los componentes desarrollados en distintos lenguajes de programación que residan en distintas plataformas pueden ser interoperables.

###### OMG / CORBA: Arquitectura de distribución de objetos, para que otros componentes se comuniquen.

###### COM (Microsoft): Especificación para utilizar los componentes elaborados por diferentes fabricantes dentro de una aplicación única sobre el sistema operativo Windows.

###### JAVA BEAN (Sun): Portátil e independiente de la plataforma. Amplía el applet de Java para acoplar componentes-software más sofisticados.

#### MODELO 3C

#### Descripciones de los componentes

* **CONCEPTO:** Descripcion de **lo que hace** el componente.
* **CONTENIDO:** Descripción de **cómo se construye** el componente.
* **CONTEXTO:** Descripción del **dominio de aplicabilidad** del componente

**CATEGORIAS**

* **CLASIFICACIÓN ENUMERADA**
  + Jerarquía
* **CALIFICACIÓN POR FACETAS**
  + Características descriptivas
* **CLASIFICACIÓN DE VALORES Y ATRIBUTOS**

**AREAS**

* **METODOS DE BIBLIOECONOMÍA**
* **INTELIGENCIA ARTIFICIAL**
* **SISTEMA DE HIPERTEXTO**

## CAPITULO 28 –

## INGENIERIA DEL SOFTWARE DEL COMERCIO ELETRÓNICO

* **Rendimiento:** Se puede incrementar añadiendo simplemente más computadoras.
* **Compartir Recursos:** Permite a sus usuarios acceder a grandes cantidades de datos que contienen las computadoras que componen el sistema.
* **Tolerancia a Fallos:** Un sistema distribuido se puede diseñara de forma que tolere los fallos tanto del hardware como del software.

**BENEFICIOS DE LOS**

**SISTEMAS DISTRIBUIDOS**

### SISTEMAS DISTRIBUIDOS

**CLIENTES / SERVIDORES**

**Cliente:** Es una computadora que solicita los servicios que dan uno o más servidores.

**Servidor:** Es una computadora que da un servicio que requiere mucha potencia de procesamiento.

**CATEGORIAS DE SERVIDORES**

**De Archivos**

**De Base de Datos**

**De Grupo (**Agenda, Workflow, mail, etc.)

**Web**

**Correo**

**De Objetos**

**De Impresión**

**De Aplicaciones**

**SOFTWARE INTERMEDIO (MIDDLEWARE)**

Es una capa de software que se encarga de intermediar entre un cliente y un servidor. El software intermedio orientado a mensajes es el software intermedio que administra el flujo de mensajes hacia y desde un cliente.

Por ejemplo:

* El software para llevar a cabo comunicaciones utilizando el protocolo TCP/IP y otros protocolos de red.
* El software del sistema operativo, que por ejemplo, mantiene un almacenamiento distribuido de archivos.

El software MOM gestiona de forma eficaz las colas que contienen mensajes que proceden y que se envían desde servidores. El software MOM la comunicación es asíncrona, hasta el punto en que la mitad de la pareja cliente/servidor puede que no esté comunicándose con la otra parte.

### ARQUITECTURAS ESTRATIFICADAS

#### ARQUITECTURA DE DOS CAPAS

Cuando hay mucho procesamiento de datos, la arquitectura de 2 capas es ideal. Los datos se pueden almacenar en una base de datos convencional, en un archivo simple o pueden ser incluso los datos que están en la memoria. Esta capa reside en el servidor.

* LOGICA Y PRESENTACIÓN
* BASE DE DATOS

Un ejemplo de 2 capas es la de la Web.

#### ARQUITECTURA DE TRES CAPAS

Cuando una aplicación implica un procesamiento considerable, entonces comienzan a surgir problemas con la arquitectura de 2 capas, particularmente aquellas aplicaciones que experimentan cambios de funcionalidad a medida que se van utilizando.

La arquitectura de 3 capas se compone de:

* CAPA DE PRESENTACIÓN: Presentación visual de la aplicación.
* CAPA DE PROCESAMIENTO: Responsable del procesamiento.
* CAPA DE BASE DE DATOS: Contiene los datos de la aplicación.

La idea de las tres capas encaja con las prácticas orientadas a objetos de hoy en día: todo el procesamiento tiene lugar por medio de los mensajes que se envían a los objetos y no mediante trozos de código asociados a cada objeto en la capa de presentación que se está ejecutando.

La diferencia entre la capa intermedia del servidor de la aplicación y el middleware, es que mientras el primero describe el software de la aplicación que media entre el cliente y el servidor, el segundo se reserva para el software de sistemas.

##### PROTOCOLOS

Siempre que se utiliza un mecanismo para la comunicación en una red, existe un protocolo.

* IP e ICMP
* POP3
* SMTP
* HTTP

### SISTEMA DE COMERCIO ELECTRONICO

La aplicación de la tecnología de sistemas distribuidos que apoya las operaciones comerciales. El comercio electrónico podría definirse como un sistema para simular alguna actividad comercial en tiempo real utilizando tecnología de red.

### TECNOLOGÍAS USADAS PARA EL COMERCIO ELETRONICO

Existen varias tecnologías basadas en red que se utilizan para las aplicaciones de comercio electrónico. El mejor ejemplo es el uso de la tecnología de base de datos relacionales para proporcionar almacenes de datos a gran escala.

Muchas tecnologías antiguas todavía se utilizan para aplicaciones de comercio electrónico.

#### CONEXIONES (Sockets)

Un socket es un tipo de conducto que se utiliza para conectarse a una computadora conectada a una red y basada en TCP/IP. El socket se configura de tal manera que los datos pueden ser bajados desde el cliente y devuelto al mismo.

#### OBJETOS DISTRIBUIDOS

Un objeto distribuido es aquel que reside en una computadora, normalmente un servidor, en un sistema distribuido. Los objetos distribuidos representan un nivel más alto de abdstracción que las conexiones (sockets) a parte de algún código de inicialización, el programador no es consciente del hecho de que el objeto reside en otra computadora.

Actualmente existen 3 tecnologías de objetos distribuidos:

* **RMI –** Esta es la tecnología asociada al lenguaje de programación Java. Es ideal para sistemas cerrados de Java, estos sistemas generalmente tendrán pocas conexiones o ninguna con otros sistemas
* **DCOM –** Es la tecnología desarrollada por Microsoft y permite que programas escritos en lenguajes como Visual Basic y Visual J++ se comuniquen con los objetos que están en computadoras remotas.
* **CORBA –** Esta es la tecnología de objetos distribuidos más sofisticados. Fue desarrollada por un consorcio de compañías informáticas, clientes y compañías de software. La característica más importante del enfoque CORBA es que es multilenguaje, donde los programadores pueden utilizar diferentes lenguajes de programación para enviara mensajes a objetos CORBA: las interfaces CORBA existen para lenguajes tales como Java, Fortran, Lisp, Ada y Smalltalk.

#### ESPACIOS

Esta es una tecnología que se encuentra en un nivel de abstracción incluso más alta o que los objetos distribuidos. La tecnología de espacios concibe un sistema distribuido en base a un gran almacén de datos persistentes donde las computadoras de un sistema distribuido para leer o escribir.

#### CGI – Common Gateway Interface

Es la interfaz con el servidor Web al cual se puede acceder mediante los programas que se ejecutan en el servidor. Gran parte de la interactividad asociada a las páginas Web se implementa programando el acceso a la CGI. Por ejemplo, cuando el usuario de un navegador accede a una página que contiene un formulario, éste lo rellena y lo envía al servidor Web, programa que accede al CGI, procesa el formulario, y lleva a cabo la funcionalidad asociada al formulario, por ejemplo recuperando los datos solicitados por el formulario.

### DISEÑO DE SISTEMAS DISTRIBUIDOS

A nivel de análisis hay poca diferencia entre un sistema distribuido y un sistema local, y se basa en que el modelo de análisis de un sistema no contendrá ningún dato de diseño como el hecho de que tres computadoras, y no una sola, están llevando a cabo algún procesamiento.

#### PRINCIPIOS DE DISEÑO – SISTEMAS DISTRIBUIDOS

1. Correspondencia del volumen de transmisión con los medios de transmisión.
2. Mantenimiento de los datos más usados en un almacenamiento típico.
3. Mantenimiento de datos cerca de donde se aplican.
4. Utilizar la duplicación de datos.
5. Eliminar cuellos de botella.
6. Minimizar la necesidad de un gran conocimiento del sistema.
7. Agrupar datos afines en la misma ubicación.
8. Utilizar servidores dedicados a funciones frecuentes.
9. Correspondencia de la tecnología con las exigencias de rendimiento.
10. Empleo de paralelismo.
11. Compresión de datos
12. Diseño para el fallo
13. Minimizar la latencia.