FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CENTRO DE INFORMACION Y DOCUMENTACION "ING. BRUNO MASCANZONI"

El Centro de Información y Documentación Ing. Bruno Mascanzoni tiene por objetivo satisfacer las necesidades de actualización y proporcionar una adecuada información que permita a los ingenieros, profesores y alumnos estar al tanto del estado actual del conocimiento sobre temas específicos, enfatizando las investigaciones de vanguardia de los campos de la ingeniería, tanto nacionales como extranjeras.

Es por ello que se pone a disposición de los asistentes a los cursos de la DECFI, así como del público en general los siguientes servicios:

* **Préstamo interno.**
* **Préstamo externo.**
* **Préstamo interbibliotecario.**
* **Servicio de fotocopiado.**
* **Consulta a los bancos de datos: librunam, seriunam en cd-rom.**

Los materiales a disposición son:

* **Libros.**
* **Tesis de posgrado.**
* **Noticias técnicas.**
* **Publicaciones periódicas.**
* **Publicaciones de la Academia Mexicana de Ingeniería.**
* **Notas de los cursos que se han impartido de 1980 a la fecha.**

En las áreas de ingeniería industrial, civil, electrónica, ciencias de la tierra, computación y, mecánica y eléctrica.

El CID se encuentra ubicado en el mezzanine del Palacio de Minería, lado oriente.

El horario de servicio es de 10:00 a 19:30 horas de lunes a viernes.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

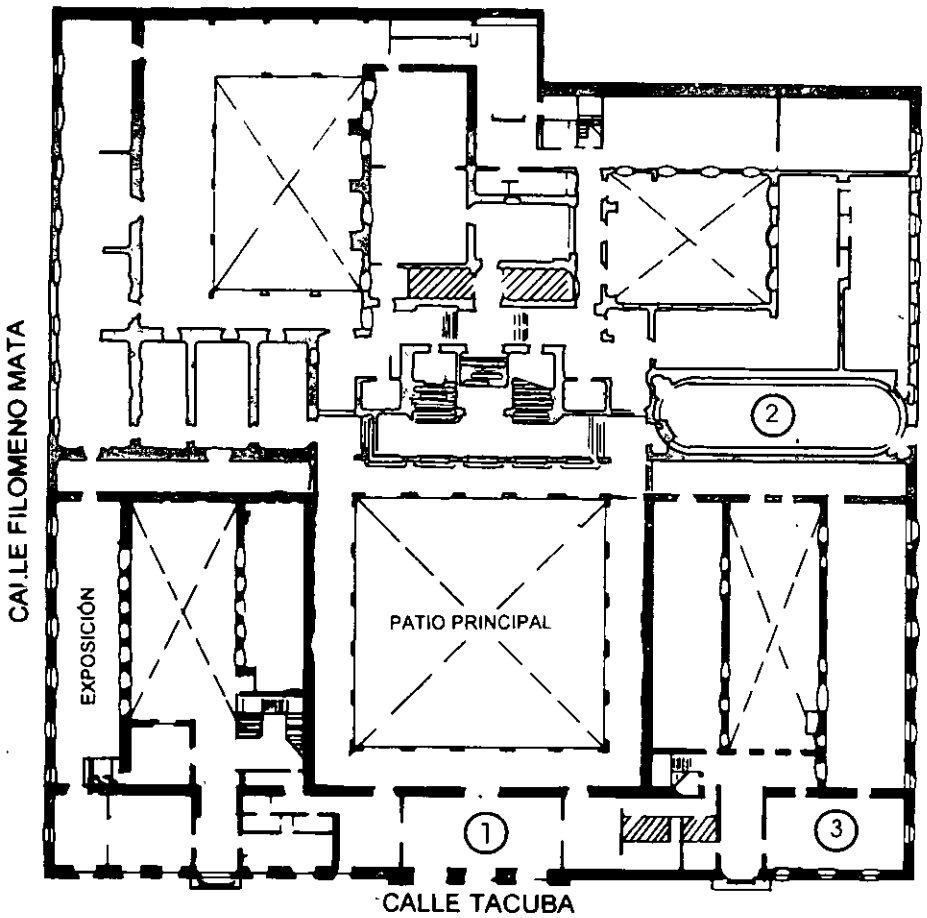
Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente

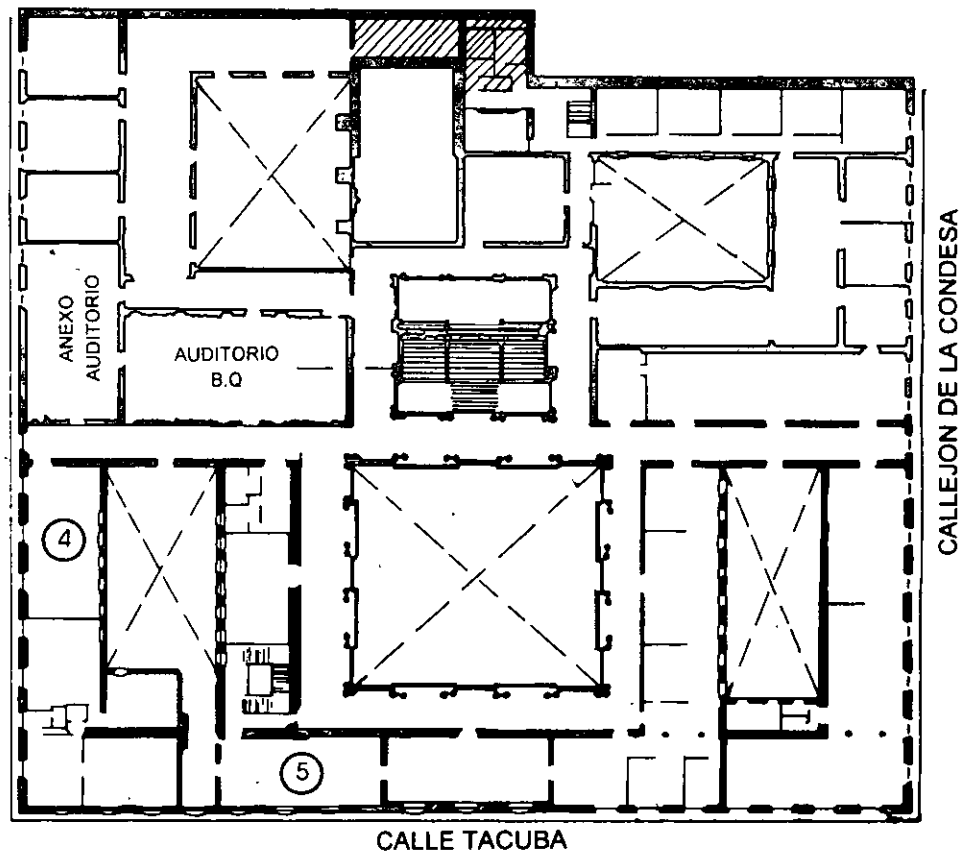
División de Educación Continua.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

PALACIO DE MINERIA

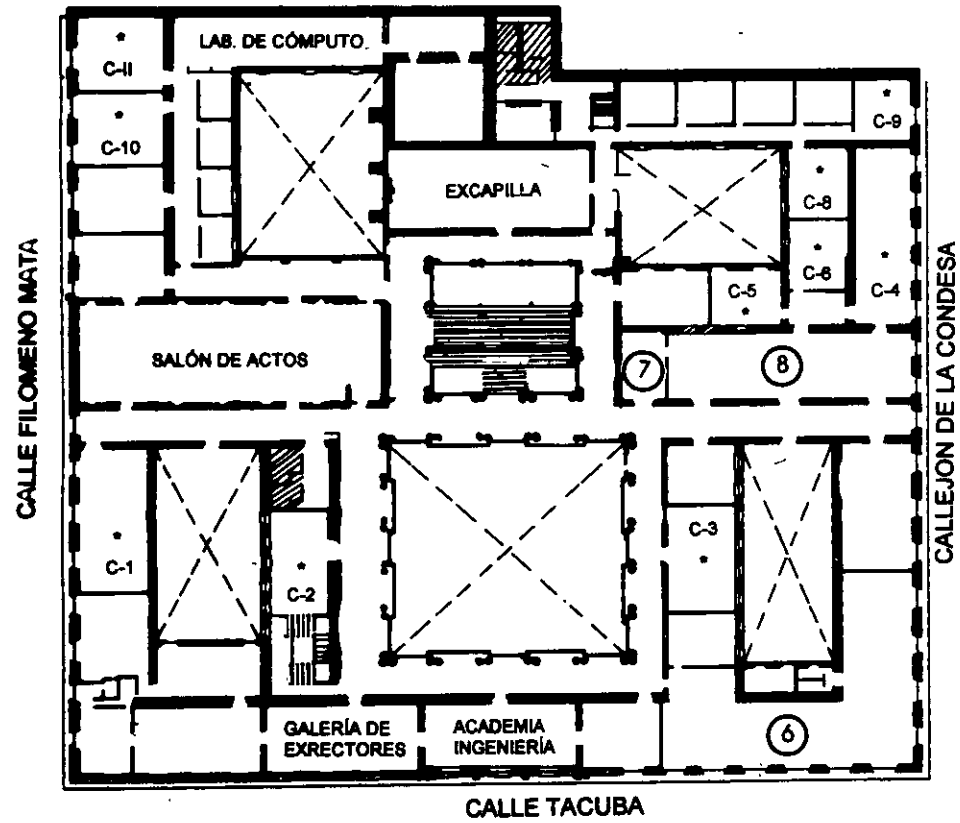


PLANTA BAJA



MEZZANINNE

PALACIO DE MINERIA



1er. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA, UNAM
CURSOS ABIERTOS

CURSO: SISTEMAS DE INFORMACION CON ORIENTACION A OBJETOS
FECHA: 9 al 13 DE DICIEMBRE, 1996

EVALUACIÓN DEL PERSONAL DOCENTE

(ESCALA DE EVALUACIÓN 1 A 10)

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACIÓN CON EL ASISTENTE	PUNTUALIDAD
ING. SALVADOR PEREZ V.				

Promedio _____

EVALUACIÓN DE LA ENSEÑANZA

CONCEPTO	CALIF
ORGANIZACION Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD DEL CURSO	
ACTUALIZACIÓN DEL CURSO	
APLICACIÓN PRACTICA DEL CURSO	

Promedio _____

EVALUACIÓN DEL CURSO

CONCEPTO	CALIF
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CALIDAD DEL MATERIAL DIDÁCTICO UTILIZADO	

Promedio _____

Evaluación total del curso_____

Continúa...2

1 ¿Le agradó su estancia en la División de Educación Continua?

SI ☐ NO ☐

Si indica que "NO" diga porqué: _____

_____

2. Medio a través del cual se enteró del curso:

Periódico *Excélsior*	
Periódico *La Jornada*	
Folleto anual	
Folleto del curso	
Gaceta UNAM	
Revistas técnicas	
Otro medio (Indique cuál)	

3. ¿Qué cambios sugeriría al curso para mejorarlo?

_____

_____

_____

_____

_____

_____

4. ¿Recomendaría el curso a otra(s) persona(s) ?

SI ☐ NO ☐

5 ¿Qué cursos sugiere que imparta la División de Educación Continua?

_____

_____

_____

_____

_____

_____

6 Otras sugerencias.

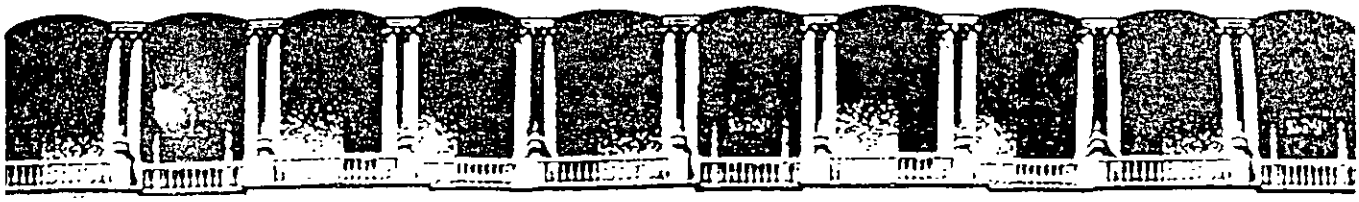
_____

_____

_____

_____

_____



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

MATERIAL DIDACTICO DEL CURSO

SISTEMAS DE INFORMACION
CON ORIENTACION A OBJETOS

DICIEMBRE, 1996

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

CONSTRUCCION DE SISTEMAS DE INFORMACION APLICANDO ORIENTACION A OBJETOS

1. Metodologías

2. Orientación a Objetos (OO)

3. Conceptos esenciales de OO

4. Análisis de Requerimientos

5. Diseño

6. Implementación
   - Lenguajes
   - Bases de Datos
   - Interfase de Usuario

7. Pruebas y Control de Calidad

8. Administración al utilizar OO

1. METODOLOGIAS PARA LA CONSTRUCCION DE SOFTWARE.

Objetivos del Software

Software Util, Económico (Costo-Beneficio), de Calidad, Oportuno, Flexible, Robusto.

Metodologías

Surgen de la necesidad de manejar la complejidad creciente de los sistemas.

Tendencias: Orientación hacia flujo de datos o Estructuras de datos.

Metodologias Estructuradas:

Programación	**--- Modularización**	**----- Dijkstra, Wirth**
Diseño	**--- Descomp. funcional**	**----- Constantine, Yourdon**
Análisis	**--- DFD, Diccionario de Datos**	**----- Yourdon, DeMarco, Gane y Sarson**

Modernización del análisis estructurado ---> Yourdon

Otras Metodologías

J. Martin	**Ingenieria de Información**
Warnier	**Construcción lógica de sistemas**
Jackson	**Jackson System Development (JSD)**

- Han contribuido a resolver ciertos problemas, pero otros persisten:

+ Enfoque hacia la computadora, se enfatiza descomposición de procesos.

+ Hay disociación entre procesos y datos.

+ Hay cambio de representacion de la realidad a estructuras de Software.

+ Cambia la representación entre las diferentes etapas de desarrollo.

+ Siempre se crean soluciones únicas y para problemas específicos.

2. ORIENTACION A OBJETOS.

- Es una nueva forma de construir software para tratar de resolver los problemas clásicos del Software.

- En esencia consiste en construir Software con componentes estandar reutilizables.

- Conjunto de objetos cooperativos para lograr el objetivo del sistema.

- Construir Software como se construye el Hardware, ensamblando componentes estandarizados de diferentes proveedores.

Hardware **3 niveles:**

1. **Computadora.**
2. **Tarjetas, Drives, Fuentes de Poder,etc estandar.**
3. **Circuitos Integrados y Componentes estandar de Intel,Motorola,etc.**

Software **3 niveles:**

3. **Objetos: Componentes reusables estandar (Software-IC).**
2. **Marcos estructurales (Frameworks) Modelos de organizacion utiles en varias aplicaciones.**
1. **Aplicaciones especificas.**

3. CONCEPTOS ESENCIALES DE OO.

- Objetos

Paquetes de datos y procedimientos.
Ej: Cliente, Alumno, Contrato, Máquina, Impresora, Párrafo, etc.

- Mensajes.

Medio de comunicación entre objetos para solicitar y proporcioanr servicios activando métodos.
Estructura del Mensaje: Objeto_receptor + Operación + Parámetros.

- Clases.

Moldes para definir objetos y relaciones de dependencia entre ellos.
Permiten representar Generalizacion y Especializacion.

- Herencia.

Relación entre clases que permite definir nuevas clases en base a clases ya existentes.

- Polimorfismo.

Capacidad de los objetos de responder de distinta forma al mismo mensaje.

- Marcos Estructurales.

Modelos operacionales de la institución.

3.1 BENEFICIOS Y COSTOS DE OO.

BENEFICIOS

1. **Mejor productividad al reducir tiempos y costos reutilizarndo componentes.**

2. **Mejor calidad usando componentes ya probados.**

3. **Facilidad de Modificacion y Mantenimiento. Objetos modificables sin afectar a otros. Se reducen costos de mantenimiento.**

4. **Adaptabilidad al cambio para responder a oportunidades y demandas del mercado(Competencia) y de los clientes (Customizacion). Facilita la evolución de los sistemas.**

5. **Tener sistemas de informacion promotores del cambio y no obstaculos para cambiar.**

COSTOS

1. **Nuevas plataformas de Software y Hardware par a soportar OO.**

2. **Entrenamiento en técnias y herramientas OO.**

3. **Desarrollo de componentes de Software reusables para desarrollos futuros.**

4. **Nueva forma de pensar en los problemas de Software.**

REUTILIZACION.

- Usar componentes existentes para construir nuevos componentes.

- Los nuevos componentes serán candidatos a reutilización.

- La reutilización requiere tener componentes valiosos.

- Debe fomentarse una cultura de reutilización.

- Hay que planear y diseñar pensando en la reutilización.

COMPONENTES DE SOFTWARE REUTILIZABLES.

Código objeto o fuente
Herramientas
Diseños
Datos
Casos de prueba
Requerimientos
Estimaciones de Costos
Planes de Proyecto
Arquitectura
Documentos
Interfases humanas

PASOS PARA REUTILIZAR.

1. .Encontrar componente reusable adecuado a los requerimientos.
2. Adaptar el componente.
3. Conectarlo con otros componentes.
4. Probar.
5. Convertir el resultado en nuevo producto reusable.

Las actividades mencionadas pueden realizarse en forma automática o manual.

4. ANALISIS DE REQUERIMIENTOS utilizando OO.

- Entender, representar y documentar ciertos aspectos de la realidad (dominio del problema).

- Definir las responsabilidades del sistema.

- Analisis OO: Utilizar enfoque de objetos para realizar el análisis.

- Identificación de Objetos y Clases.

- Identificación de Relaciones entre Clases Gen-Espec y Todo-Parte.

- Identificación de subsistemas.

- Identificación de Atributos.

- Identificación de Servicios.

4.1 IDENTIFICACION DE OBJETOS.

OBJETO.

Elemento del dominio acerca del cual se conserva información y responsable de cierto comportamiento.

CLASE.

- Descripción genérica de objetos con estructura y funcionamiento comunes.
- Objetos con atributos y servicios comunes.

FUENTES DE INFORMACION PARA CLASES.

- Observación.
- Interacción y Participación (Interfases).
- Clases reutilizables del dominio u otros dominios semejantes.
- Personas, cosas o eventos.
- Roles desempeñados.
- Interacciones (Transacciones).
- Dispositivos y sistemas externos.
- Procedimientos operacionales (Reglas, Criterios).
- Lugares.
- Estructuras.

REPRESENTACION DE OBJETOS Y CLASES.

Nomenclatura apegada a vocabulario estandar del dominio.
Utilizar un sustantivo o sustantivo adjetivado.

4.2 IDENTIFICACION DE ESTRUCTURAS.

ESTRUCTURA.

Expresa la complejidad del dominio según las responsabilidades del sistema.

ESTRUCTURA GEN-SPEC.

- **Es relación IS-A.**
- **Representa Generalización-Especialización.**
- **Define explicitamente las similitudes entre clases.**
- **Se aplica a clases.**
- **Representación.**

ESTRUCTURA TODO-PARTES.

- **Es relación Part-Of.**
- **Define relaciones entre objetos.**
  - **+ Ensamble-Partes**
  - **+ Contenedor-Contenidos**
  - **+ Colección-Miembros**
- **Se aplica a objetos.**
- **Representación.**

ESTRUCTURAS GEN-SPEC.

Buscar Estructuras Gen-Spec reutilizables del dominio u otros semejantes.

Para definir estructuras Gen-Espec, se considera como Generalización a cada Clase definida y para cada Especialización potencial se considera

a) Si forma parte del dominio

b) Si es responsabilidad del sistema manejarla.

c) Si aplica herencia

d) Si la especialización cumple con los criterios de una Clase

Se hacen las mismas consideraciones para cada Generalización potencial, ahora considerando como Especialización a cada Clase definida.

Cuando hay varias especializaciones posibles, considerar primero la más simple y la más compleja y después las intermedias

La forma más común de estructura Gen-Espec es una jerarquia.

ESTRUCTURAS TODO-PARTES.

Buscar Esructuras Todo-Partes reusables del dominio u otros semejantes.

Para definir estructuras Todo-Partes, se considera cada Objeto como un Todo y para cada Parte potencial se considera

a) Si forma parte del dominio

b) Si es responsabilidad del sistema manejarla

c) Si es una abstracción util en el manejo del dominio

d) Si solo contiene valor de estatus, definirla como atributo del Todo.

Se hacen las mismas consideraciones para cada Todo potencial, ahora considerando como Parte a cada objeto.

DEFINICION DE ATRIBUTOS.

- **ATRIBUTO:** Dato relevante para los objetos de una clase.

- Revisar resultados previos para reutilizar.

- Para cada atributo debe considerarse lo siguiente:
  + Su descripción en general, en el contexto del dominio y en el contexto de las responsabilidades del sistema.

  + Debe registrar un "concepto atómico" valor simple o grupo de valores estrechamente relacionados.

- El atributo debe reflejar lo que debe conocer el objeto, lo que debe recordar a traves del tiempo y los estados en que puede encontrarse el objeto.

- La decisión de almacenar atributos derivados debe diferirse hasta el diseño, es un compromiso espacio-tiempo. Especificar el servicio para la derivación.

- El atributo debe colocarse en la Clase más adecuada (Acorde al dominio).

- En Estructuras Gen-Spec aplciar herencia y colocar los atributos más generales arriba y los especializados abajo dentro de la jerarquía.

CASOS ESPECIALES CON ATRIBUTOS.

- Para cada atributo
  + Checar valores "no aplicables"
  + Checar valores repetitivos

- Checar clases con un solo atributo.

ESPECIFICACION DE ATRIBUTOS.

- Nombre acorde al vocabulario estandar del dominio y a las responsabilidades del sistema.
- No debe incluir valores.
- Descripción del atributo.
- Especificar restricciones para reducir la especificación de servicios.
- Especificar unidades de medición del atributo.

CONEXION DE INSTANCIAS.

- Representa que un objeto necesida de otro para cumplir sus responsabilidades.

- Revisar resultados previos para reutilizar.

- Definir el rango:

Conexión opcional: limite inferior = 0.
Conexión obligatoria: limite inferior >= 1.
Conexión simple: limite superior = 1.
Conexión múltiple: limite superior > 1. Ej: Multi-versión.

CASOS ESPECIALES CON CONEXION DE INSTANCIAS.

- Checar cada conexión de instancia del tipo N-M.

- Checar cada conexión de instancia entre objetos de la misma clase.

- Checar múltiple conexión de instancia entre objetos.

- Checar conexiones de instancia faltantes.

- Checar objeto en la conexión que tenga significado especial.

DEFINICION DE SERVICIOS.

- Servicio: Comportamiento que debe mostrar un objeto.

ESTADOS DEL OBJETO.

- Revisar resultados previos para reutilizar.
- Examinar valores potenciales de los atributos.
- Determinar si cambian las responsabilidades del sistema para esos valores
  potenciales.
- Utilizar diagrama de transición de estados.

SERVICIOS REQUERIDOS.

- Servicios **SIMPLES.**
  - + Crear Crear e inicializar objetos como instancias de clases.
  - + Conectar Conectar o desconectar objetos.
  - + Acceso Accesar o modificar valores de Atributos en Objetos.
  - + Liberación Libera (desconecta y destruye) un objeto.

- Servicios **COMPLEJOS.**

  Utilizar nombres específicos del dominio.
  - + Verificar resultados previos para reutilizar.
  - + Cálculo: Calcular resultado a partir de valores de atributos del objeto. Determinar los cálculos de debe realizar el objeto con sus valores.
  - + Monitoreo: Monitorear el ambiente y manejar entradas y salidas. Puede requerir servicios complementarios de Inicialización y terminación.
  - + Responsabilidades del objeto para detectar y responder a eventos en el ambiente.

CONEXION CON MENSAJES.

Representa dependencia de procesos. Indica la necesidad de servicios ajenos para cumplir responsabilidades.

Para cada objeto:

- Checar resultados previos para reutilizar.

- Determinar los objetos de los que necesita servicios.

- Determinar los objetos que requieren de los servicios que proporciona el objeto.

- Seguir la cadena de conexión de mensajes y hacer las mismas consideraciones.

- Examinar secuencia (thread) de Conexión con Mensaje.

- Verificar omisiones mediante simulación de roles manual o automática.

- Determinar requerimientos de proceso en tiempo real cuando aplique.

ESPECIFICACION DE SERVICIOS.

- Verificar resultados previos.

- Utilizar diagramas de estados y diagramas de servicios.

- Utilizar estilo consistente en los bloques de texto.

- Definir restricciones adicionales.

- Tabla de estados y servicios para resumir servicios dependientes del estado.

MANTENIMIENTO.

- Los cambios frecuentes provocan desorden. Entropia.

- El dinamismo de las jerarquías de clases puede favorecer entropia.

- Hay incremento en la entropia conceptual al descender en la jerarquía.

- Debe preservarse la consistencia conceptual durante el mantenimiento.

- A mayor profundidad mayor posibilidad de que la clase no sea extensión o especialización consistente del concepto de la superclase.

- CRITERIOS DE CONSISTENCIA EN LAS JERARQUIAS.

1. Número de clases con la misma superclase.
   Refleja concordancia en la definición de conceptos y en la clasificación del concepto considerado.

2. Número de clases en el mismo nivel.
   Refleja concordancia en nivel de abstracción de la clase relativo a otras clases de la jerarquía.

- La modificación a la jerarquia de clases requiere compartir la visión del dominio y de la arquitectura conceptual de la jerarquía.

- La clasificacion sin criterio definido genera inconsistencias porque el número de opciones para clasificar crece al descender en la jerarquía.

Causas de la Entropia Conceptual al descender:

- Los conceptos se vuelven más complejos.

- A mayor profundidad, hay más candidatos para superclases y mayor posibilidad de mala clasificación.

- Las clases son mas especializadas. Para determinar la superclase más adecuada deben considerarse variaciones conceptuales sutiles.

REDUCCION DE LA ENTROPIA CONCEPTUAL.

Consideraciones:

- **Mejorar la calidad de la jerarquía de clases.**
- **La reestructuración de la jerarquía es costosa (pero necesaria).**

Acciones.

- **Arquitectura Conceptual de la jerarquia fácil de entender, explícita y consistente.**
- **Definir criterios cuantitativos para definir subclases (subclasificar) y reducir subjetividad.**
- **Asegurar aplicación consistente (repetible) de criterios de subclasificación.**

1. ARQUITECTURA CONCEPTUAL.

- **Concentrarse en atributos conceptuales de la clase y diferir lo referente a diseño e implementación.**

- **Representación no debe basarse en lenguaje de implementación.**

- **Capacidad para analizar especificaciones de la clase.**

- **Propiedades de lo que hace la clase, omitiendo la implementación. Balance de abstraccion, simplicidad y analizabilidad.**

- **Propiedades.**
  Esenciales: Esencia del concepto de la clase.
  De soporte: Elementos importantes, no cruciales.
  Incidentales: Dependientes de la implementacion.

2. Criterios cuantitativos para subclasificar.

a. Especificidad conceptual.
Asegurar que clases generales quedan mas alto.

b. Consistencia conceptual.
- Concepto de una clase consistente con el de otra clase.
- Asegurar que la subclase extiende/especializa en forma consistente el concepto de su superclase.
- Las clases conceptualmente inconsistentes no deben tener relacion clase-subclase.

Considerando a la Clase B como subclase de A

+ Ninguna propiedad de B puede negar alguna propiedad esencial de A.

+ El conjunto de valores en B para una propiedad común con A no debe ser superset de los valores para la propiedad en A.

c. Distancia conceptual.
- Mide similitud de conceptos entre 2 clases.
- Asegurar que una subclase es más similar a su superclase que cualquier otra.

- A mayor distancia conceptual, menor similitud de conceptos.
- Diferencia de propiedades esenciales y diferencia de propiedades de soporte.

- Para propiedades comunes, número de valores en que difieren.

3. Aplicacion Consistente de los criterios del punto 2.

□

LENGUAJES.

Criterios de Evaluación

- Puro o Híbrido
- Compilador o Intérprete
- "Typing" fuerte o débil
- Liga estática o dinámica
- Soporte de Herencia múltiple
- Diversidad de proveedores
- Disponibilidad de un ambiente de desarrollo

Puro.

- Smalltalk. XEROX. Alan Kay.
- Todo es objeto. Flexibilidad. Posibilidad de cambios profundos.
- Implicaciones de rendimiento.
- Obliga a la OO.

Híbrido.

- C++. AT&T. Bjarne Stroustrup.
- Extensiones OO a lenguaje existente. Lenguaje base inalterable.
- Rendimiento semejante al del lenguaje base.
- Aprovecha experiencia existente en lenguaje base.
- Riesgo de no cambiar realmente a un enfoque de OO y continuar con el enfoque procedural.

Intérprete.

Flexible y rápido para desarrollar. Lentitud en la ejecución.

Compilador.

Eficiencia al ejecutar. Lentitud en el desarrollo (Compilación y Ligado).

"Typing"

Definición del tipo de contenido de variables antes de utilizarlas.
Error de tipo si no hay coincidencia. Detectado al compilar o ejecutar.

Tipo Fuerte.

- Requiere definir tipo antes de usar variables.
- Mayor seguridad en el manejo y optimización.

Tipo Debil.

- Puede tenerse cualquier tipo de objeto durante la ejecución.
- Mayor flexibilidad. Mayor riesgo.

Ligado:

- Proceso para determinar el receptor del mensaje.
- Por default puede ser estático o dinámico.

Estático (Temprano, al-compilar)

Definido al compilar el programa.

Dinámico (Tardio, al_ejecutar)

El receptor del mensaje se define al ejecutar el programa.

Esto hace posible el polimorfismo. El objeto receptor quizá no existe al momento de compilar el programa, porque se agregó después.
El definir ligado dinámico para todos los mensajes da flexibilidad para extender pero requiere búsqueda adicional para determinar el receptor del mensaje.

Herencia Múltiple.

- Una clase hereda todos los métodos y variables de todas sus superclases.
- Hay partidarios y enemigos de la Herencia múltiple.
- Partidarios: Refleja la realidad. Su exclusión lleva a redundancia.
- Enemigos: Puede reestructurarse la herencia múltiple para tener sólo herencia simple. La herencia múltiple impone trabajo adicional en el lenguaje y en el programador.

Diversidad de Proveedores.

El tener un solo proveedor limita la capacidad de difusión.

Smalltalk:

Puro	Intérprete	Type Debil	Liga Dinámica	Her. Simple.

C++

Hibrido	Compilador	Type Fuerte	Liga SemiDinám	Her. Múltiple.

Lenguajes de Programación OO.

C++
Smalltalk
Objective C
Object Pascal
Eiffel
Actor
CLOS

BASES DE DATOS.

Evolución:

Archivos Indexados -- Modelo Jerárquico -- Modelo Red -- Modelo Relacional

Siguiente Generación de BD

- **Orientadas a Objetos**
- **DBMS relacional extendido.**

Aplicar la tecnología de BD a nuevos tipos de aplicaciones:

CAD, CAM, CASE, GIS, Aplicaciones médicas y científicas, Automatización de Oficinas, Aplicaciones en que DBMS tradicionales han sido inadecuados.

Se requiere conservar lo referente a manejo de concurrencia, integridad, seguridad, recuperación, rendimiento, etc.

Para soportar datos más complejos, se requiere lo siguiente:

1. **Manejo de Objetos.**
   Ej: Un componente CAD.
   Se almacenan datos y procedimientos.
   Los métodos encapsulan la semántica del objeto.
2. **Identificadores de Objetos.**
   A cada objeto se le asigna un identificador único.
3. **Manejo de Objetos compuestos en cualquier número de niveles.**
4. **Representación de relaciones entre objetos, más allá del join.**
5. **Manejo de datos Multimedia.**
   Objetos que contienen texto, imagenes, audio, video.
   Operaciones eficientes para manejo de estos tipos de datos.
6. **Aplicación de herencia en la definición de esquemas.**
7. **Manejo de versiones de los objetos.**
8. **Alto rendimiento para operaciones con objetos.**
9. **Mayor integración con los lenguajes de programación.**

EJEMPLOS DE ODBMS.

- **GemStone**
- **POSTGRES**
- **ObjectStore**
- **ORION**
- **ONTOS**
- **Iris**
- **VERSANT**
- **SIM**
- **GBase**
- **O2**

C++

```
#include <stdio.h>
/* Definicion de Clases */
class Acum
{
private:
  float Total;
public:
  void Inicia (void);
  void suma (float Cant)  {Total = Total + Cant; }
  float dispTotal (void)    { return Total; }
};

void Acum::Inicia(void)
{ Total = 0.0; }

/* Creación y eliminación de instancias.
  Depende del alcance del objeto. */

main ()
{
Acum A1;                          /* Declaración */
Acum *p_A1 = new Acum;    /* Ejecución */

/* Manejo de los Objetos. */
A1.Inicia ();
A1.suma(4);
A1.suma(3);
printf("Total con Mensajes al objeto = %f \n", A1.dispTotal() );
p_A1->Inicia ();
p_A1->suma(6);
p_A1->suma(2);
printf("Total con Mensajes a Pointer al Objeto = %f \n", p_A1->dispTotal() );
delete p_A1;
}
```

```
#include <stdio.h>
/* HERENCIA
   Clase Base Acum, Clase derivada Acum2 */

class Acum {
public:
  float Total;
  void inicio (void)                { Total = 0.0; }
  virtual void suma (float cant)  { Total = Total + cant; }
  float dispTotal (void)            { return Total; }
};

class Acum2 :public Acum {
public:
   int cont;
void inicio (void)                { Total = 0.0; cont = 0; }
virtual void suma (float cant) { cont = cont + 1;  Total = Total + cant; }
float prom (void) {
  if (cont > 0)
     return dispTotal () /cont;
   else
     return 0;
                }
};

/* POLIMORFISMO. */
main ()
{ Acum    a;
  Acum2  a2;
 a.inicio ();
 a.suma  (5);
 printf("a  %f\n" , a.dispTotal () );
 a2.inicio ();
 a2.suma  (3);
 printf("a2 %f\n" ,a2.dispTotal () );
 /* Compatibilidad de la asignación
    A un objeto de la clase base se le puede asignar un objeto de la clase
    derivada, pero no alrevés */
 a = a2;
/*  a2 = a;    No match Acum2::Acum2(Acum) */
}
□
```

BIBLIOGRAFIA

Object Oriented Design With Applications
Grady Booch
Benjamin Cummings

Object Oriented Modeling and Design
J. Rumbaugh, M. Braha
Prentice Hall

Object Oriented Software Costruction
Bertrand Meyer
Prentice Hall

Object Oriented Systems Analysis: A model driven approach
D.W. Embley, B.D. Kurtz
Prentice Hall

Essays on Object Oriented Software Enginnering
E. Berard
Prentice Hall

Designing Object Oriented Software.
R. Wirfs-Brock, B. Wilkerson
Prentice Hall

Object Oriented Programming: An evolutionary Approach
Brad Cox
Addison Wesley