Módulo Ingeniería del Software 2

GLADIS HELENA VÁZQUEZ ECHAVARRÍA

POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Alcaldía de Medellín
Secretaria de Educación

CONVENIO INTERADMINISTRATIVO
SECRETARÍA DE EDUCACIÓN - MUNICIPIO DE MEDELLÍN
POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID

Sergio Fajardo Valderrama
Alcalde de Medellín

Juan Camilo Ruiz Pérez
Rector Politécnico Colombiano J.I.C.

Horacio Arango Marín
Secretario de Educación

Giovanni Orozco Arbeláez
Vicerrector de Docencia e Investigación

Ana Lucía Hincapié Correa
Subsecretaria de Planeación Educativa

Gilberto Giraldo Buitrago
Vicerrector de Extensión

Juan Manuel Valdés Barcha
Subsecretario de Educación

Darío Castrillón Toro
Director de Proyectos Especiales

Teresita Aguilar García
Subsecretaria Administrativa

Luis Alfredo Aguilar Roldan
Decano Facultad de Ingenierías

Esta es una producción del Convenio Interadministrativo Secretaría de Educación Municipio de Medellín - Politécnico Colombiano Jaime Isaza Cadavid.

Coordinadora del convenio: Maryem Aliria Ruiz Nuñez

Compiladora Gladis Helena Vázquez Echavarría

Revisión: John Jairo Monsalve Restrepo

.

Impresión

POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID

Medellín. Julio de 2007

INTEGRACIÓN DEL MÓDULO POR UNIDADES

UNIDAD 0: **SABERES PREVIOS**

1. REQUERIMIENTOS
2. TIPOS DE REQUERIMIENTOS
2.1 FUNCIONALES
2.2 NO FUNCIONALES
3. PROBLEMAS Y CONSECUENCIAS
4. ACTIVIDADES
5. PRODUCTO FINAL
6. CARACTERÍSTICAS
7. DIFICULTADES Y BENEFICIOS
7.1 DIFICULTADES PARA DEFINIR LOS REQUERIMIENTOS
7.2 BENEFICIOS DE UNA BUENA TOMA DE REQUERIMIENTOS
8. PERSONAL INVOLUCRADO EN LA TOMA DE REQUERIMIENTOS
9. TÉCNICAS PARA HALLAR DATOS
9.1 LA ENTREVISTA
9.2 EL CUESTIONARIO
9.3 INSPECCIÓN DE REGISTROS
9.4 OBSERVACIÓN
10. GUÍA PARA LA OBTENCIÓN DE REQUERIMIENTOS

UNIDAD 1: **ANÁLISIS DEL SISTEMAS**

1. DEFINICIÓN Y CARACTERÍSTICAS
2. EL MODELADO DE ANÁLISIS
3. ELEMENTOS EN EL MODELO DE ANÁLISIS
3.1 MODELADO DE DATOS –DER
3.2 MODELADO FUNCIONAL Y FLUJO DE INFORMACIÓN- DFD
3.3 ESPECIFICACIÓN DE CONTROL
3.4 DICCIONARIO DE DATOS
3.5 MINIESPECIFICACIONES

UNIDAD 2: **DISEÑO DEL SISTEMA**

1. DEFINICIÓN
2. CARACTERÍSTICAS
3. PASOS PARA EL DESARROLLO DEL DISEÑO
3.1 EL DISEÑO DE DATOS
3.2 EL DISEÑO ARQUITECTÓNICO
3.3 EL DISEÑO DE LA INTERFAZ
3.4 EL DISEÑO A NIVEL DE COMPONENTES

UNIDAD 3: **IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA**

1. CONCEPTOS GENERALES
2. DEFINICIÓN DE PRUEBAS
3. PRINCIPIOS
4. RECOMENDACIONES
5. ETAPAS INVOLUCRADAS EN TODAS LAS PRUEBAS
6. ESTRATEGIAS DE PRUEBAS
7. NIVELES DE PRUEBAS

7.1 PRUEBA DE UNIDAD
7.2 PRUEBA DE INTEGRACIÓN
7.3 PRUEBA DE VALIDACIÓN
7.4 PRUEBA DE SISTEMA

8. PLAN DE PRUEBAS

UNIDAD 4: **IMPLANTACIÓN Y EVALUACIÓN**

1. CONCEPTOS GENERALES
2. LA IMPLANTACIÓN
3. LA EVALUACIÓN

3.1 ¿ESTAMOS CONSTRUYENDO SOFTWARE SIN DEFECTOS?: ESTADO ACTUAL DE LA PRÁCTICA
3.2 CONTROL DE CALIDAD DEL SOFTWARE
3.3 TÉCNICAS DE EVALUACIÓN DE SOFTWARE

4. EL MANTENIMIENTO
5. LA DOCUMENTACIÓN

BIBLIOGRAFÍA

UNIDAD 0

SABERES PREVIOS

CONDUCTA DE ENTRADA

¿Qué saberes previos se deben tener para entender este programa y por que?

¿Qué es un dato, información, sistema, sistema de información, actividad, tarea, arquitectura, método, herramientas, estrategia, proceso, problema, oportunidad?

¿Qué es para usted un requerimiento?

¿Cómo identifica usted un problema o necesidad? Cite al menos tres ejemplos.

Una vez desarrollado el programa ¿Qué se espera saber y que capacidades cree que debe haber adquirido con relación a la Ingeniería del software?

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad por parte del profesor.
- Exposición magistral de los conceptos inherentes a la ingeniería de software, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Desarrollo de plenarias y exposiciones.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en Internet

PLAN DE TRABAJO

INTRODUCCIÓN

El programa de Ingeniería de Software II a desarrollarse para el grado 11°, comprende cuatro unidades de trabajo teórico practicas, con ellas, se pretende que el alumno se apropie de las herramientas necesarias para la recolección de información que le permita hacer un buen análisis y diseño de un proyecto de desarrollo de software que posteriormente deberá implementar en una empresa en particular y que se servirá como un ejercicio para futuros proyectos de desarrollo de software.

OBJETIVO GENERAL

Identificar los requerimientos operacionales de un sistema de información y aplicar técnicas de levantamiento de requerimientos en un proyecto de desarrollo de software en particular

OBJETIVOS ESPECÍFICOS

- Reconocer la importancia que tiene saber levantar requerimientos
- Identificar un problema u oportunidad.
- Diferenciar las técnicas de levantamiento de requerimientos.
- Diseñar un instrumento metodológico que le permita levantar requerimientos.
- Aplicar un instrumentos metodológico en el levantamiento de requerimientos

CONTENIDO

1. REQUERIMIENTOS

¿QUÉ ES DETERMINACIÓN DE LOS REQUERIMIENTOS?

Es el estudio del sistema actual del negocio a fin de encontrar cómo trabaja y dónde debe mejorarse, cumple un papel primordial en el proceso de producción de software porque define que se desea producir y pretende minimizar los problemas relacionados al desarrollo de sistemas

Un requerimiento es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal y debe incluir todos los aspectos necesarios para la satisfacción de las necesidades del cliente.

La especificación de los requerimientos debe ser lo suficientemente clara, para no dar pie a ambigüedades y malos entendidos que puedan llevar al diseño de una solución errónea. Generalmente la definición de los requerimientos debe ser un proceso conjunto entre desarrollador y cliente

La principal tarea de la Ingeniería de requerimientos consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema.

En síntesis la ingeniería de requerimientos es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos, entre los clientes y el equipo del proyecto

2. TIPOS DE REQUERIMIENTOS

2.1 FUNCIONALES

Definen las funciones que el sistema será capaz de realizar y describen las transformaciones que el sistema realiza sobre las entradas para producir salidas

2.2 NO FUNCIONALES

Tienen que ver con características que de una u otra forma puedan limitar el sistema. Tales como:

- El rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares

- Restricciones de diseño
- Rendimiento
- Procesos que debe soportar
- Tiempos de respuesta
- Reportes que se deben generar
- Información que debe almacenar
- Interfaz
- Almacenamiento
- Confiabilidad

3. PROBLEMAS Y CONSECUENCIAS

Los problemas más frecuentes cuando se enfrenta a levantar requerimientos son :

- La definición del alcance del sistema.
- El entendimiento común de los requerimientos entre las diferentes comunidades participantes en el desarrollo del sistema.
- Al tratar con la naturaleza volátil de los requerimientos.
- Arrastrar requerimientos.
- Documentos de requerimientos: con pérdidas, información presentada con ambigüedades o mal interpretadas, representación pobre o inadecuada.
- Oraciones inestables de los requerimientos.
- Información obsoleta.

Las consecuencias que se generan por una mala toma de requerimientos son:

- Obtención de requerimientos pobres
- Cancelación del desarrollo del sistema
- Desarrollo de un sistema que será insatisfactorio o inaceptable, con altos costos de mantenimiento o con una gran frecuencia de cambios.
- Sobrecostos

Por ende al levantar requerimientos deben establecerse claramente las responsabilidades del cliente y del desarrollador para su especificación, como también deben tenerse políticas de revisión, aprobación y anulación de los requerimientos, y por último deben tenerse políticas para el control de cambios sobre los requerimientos ya especificados y un glosario de términos para la aclaración de los documentos relacionados.

4. ACTIVIDADES

- Descubrimiento de los requerimientos: Explorar, adquirir y concretar.
- Modelar requerimientos: Representar el mundo real como descripción abstracta de los requerimientos.

- Especificación de los requerimientos: Descripción precisa y formalización.
- Validación de los requerimientos: Corrección y viabilidad.
- Administración de los requerimientos: Planeamiento, seguimiento, evaluación del impacto de cambios.

El analista debe definir: Objetos de datos, evaluar el flujo y contenido de la información, definir y elaborar todas las funciones del software, entender el comportamiento del software en el contexto de acontecimientos que afectan al sistema y descubrir restricciones de diseño adicionales.

5. PRODUCTO FINAL

Es producto final que se obtiene de una buena ingeniería de requerimientos es:

Puede ser un documento legal como un contrato, un plan de desarrollo o el documento de especificación de los requerimientos del software, el cual debe ser Internamente consistente con los documentos y prácticas del negocio, debe ser correcto y completo (necesidades usuario), Claro, debe ser la base para el diseño y las pruebas, para la Planeación y gestión del proyecto y la base para la elaboración del manual de usuario.

6. CARACTERÍSTICAS

- Necesario: si su omisión provoca una deficiencia en el sistema a construir.
- Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- Completo: si se proporciona la información suficiente para su comprensión.
- Consistente: si no es contradictorio con otro requerimiento.
- No ambiguo: cuando tiene una sola interpretación.
- Verificable: mediante el uso de métodos de verificación: inspección, análisis, demostración o pruebas.
- Nunca son iguales. Algunos son más difíciles, más riesgosos, más importantes o más estables que otros.
- Posibilidad de rastreo: porque los requerimientos están relacionados unos con otros, y a su vez se relacionan con otras partes del proceso.
- Cada requerimiento tiene propiedades únicas y abarcan áreas funcionales específicas.
- Modificable : Un requerimiento puede cambiar a lo largo del ciclo de desarrollo
- Posibilidad de utilizar durante la operación y el mantenimiento

7. DIFICULTADES Y BENEFICIOS

7.1 DIFICULTADES PARA DEFINIR LOS REQUERIMIENTOS

- Los requerimientos no son obvios y vienen de muchas fuentes.
- Son difíciles de expresar en palabras (el lenguaje es ambiguo).
- Existen muchos tipos de requerimientos y diferentes niveles de detalle.
- La cantidad de requerimientos en un proyecto puede ser difícil de manejar.

7.2 BENEFICIOS DE UNA BUENA TOMA DE REQUERIMIENTOS

- Permite gestionar las necesidades del proyecto en forma estructurada, cada actividad consiste en una serie de pasos organizados y bien definidos.
- Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados, proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- Disminuye los costos y retrasos del proyecto, reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro.
- Mejora la calidad del software ya que ésta tiene que ver con cumplir un conjunto de requerimientos.
- Mejora la comunicación entre equipos ya que la especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores.
- Evita rechazos de usuarios finales ya que obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema, por lo que se le involucra durante todo el desarrollo del proyecto

8. PERSONAL INVOLUCRADO EN LA TOMA DE

- Usuario final: usará el sistema desarrollado. Está relacionado con la usabilidad, la disponibilidad y la fiabilidad del sistema y familiarizados con los procesos específicos que debe realizar el software.
- Usuario Líder: comprende el ambiente del sistema o el dominio del problema en donde será empleado el software desarrollado. Proporciona al equipo técnico los detalles y requerimientos de las interfaces del sistema.
- Personal de Mantenimiento: Su trabajo consiste en revisar y mejorar los procesos del producto ya finalizado.
- Analistas y programadores: Son los responsables del desarrollo del producto en sí, ellos interactúan directamente con el cliente.
- Personal de pruebas: Son quienes van a validar si los requerimientos satisfacen las necesidades del cliente.
- Otros: administradores de proyecto, documentadores, diseñadores de base de datos

9. TÉCNICAS PARA LEVANTAR REQUERIMIENTOS

9.1 LA ENTREVISTA

Se utiliza para recabar información en forma verbal, a través de preguntas que propone el analista y quienes responden pueden ser gerentes o empleados, usuarios potenciales o aquellos que proporcionarán datos o serán afectados por la solución propuesta, además puede ser personal o grupal.

VENTAJAS

- Se pueden conocer datos que no están disponibles en ninguna otra forma.
- Es la mejor fuente de información cualitativa.
- Opción para las personas que no se expresan bien en forma escrita.
- Menos incómodo para los gerentes de alto nivel que llenar cuestionarios.

Determinación del tipo de entrevista:

- Sin estructura: preguntas y respuestas libres.
- Estructuradas: preguntas estandarizadas.

De acuerdo a la forma de preguntar

- Preguntas para respuesta abierta.
- Preguntas para respuesta cerrada.

Entrevista estructurada (ventajas)

- Asegura la elaboración uniforme de las preguntas para todos los que van a responder.
- Fácil de administrar y de evaluar.
- Evaluación más objetiva tanto de quienes responden como de las respuestas a las preguntas.
- Requiere poco entrenamiento del entrevistador.
- Resulta en entrevistas más pequeñas.

Entrevista estructurada (desventajas)

- Alto costo de preparación.
- Los que responden pueden no aceptar un alto nivel en la estructura y carácter mecánico de las preguntas.
- Un alto nivel en la estructura puede no ser adecuado para todas las situaciones.
- El alto nivel en la estructura reduce responder en forma espontánea, así como la habilidad del entrevistador para continuar con comentarios hacia el entrevistado.

Entrevista no estructurada (ventajas)

- El entrevistador tiene mayor flexibilidad al realizar las preguntas adecuadas a quien responde.
- El entrevistador puede explotar áreas que surgen espontáneamente durante la entrevista.
- Puede producir información sobre áreas que se minimizaron o en las que no se pensó que fueran importantes.

Entrevista no estructurada (desventajas)

- Puede utilizarse negativamente el tiempo, tanto del que responde como del entrevistador.
- Los entrevistadores pueden inducir sus sesgos en las preguntas o al informar de los resultados.
- Puede recopilarse información extraña.
- El análisis y la interpretación de los resultados puede ser largo.
- Toma tiempo extra recabar los hechos esenciales.

Selección de entrevistados

- Personas que tienen información a la que no se puede llegar de otra forma.
- Directos involucrados en el proyecto.
- 1ª etapa: gerencia para determinar factibilidad.
- 2ª etapa: depende de quien pueda proporcionar la mayor información útil.

Antes de la entrevista

- Realizar cita por anticipado con futuros entrevistados para conocimiento previo.
- Avisar sobre la naturaleza de la entrevista.
- Planear y arreglar con cuidado las preguntas.
- Familiarizarse con el tema de la entrevista.
- Información grupal no superior a una hora.

Durante la entrevista

- Presentación, tema de la entrevista, naturaleza del proyecto.
- Preguntas generales.
- Continuar con temas y aspectos que surjan.
- No utilizar notas distrayentes.
- Preguntas específicas alternas.
- Resumen entrevista.
- Dejar posibilidad de nuevas entrevistas.
- Tacto, imparcialidad, vestimenta apropiada.

9.2 EL CUESTIONARIO

El cuestionario tiene un mayor cubrimiento y sirve para conocer varios aspectos del sistema, además suelen ser más confiables cuando se llenan sin colocar el nombre.

Formas de cuestionarios

- ***Cuestionario abierto***

  Sirve para conocer sentimientos, razones opiniones y experiencias generales.

- ***Cuestionario cerrado***:

  - Limita las respuestas.
  - Fuerza al entrevistado a tomar una decisión.
  - Respuestas de tipo si/no, de acuerdo/en desacuerdo.

Selección de encuestados

- De acuerdo a la información que las personas puedan suministrar.
- Verificar antecedentes y experiencias de encuestados.
- Que estén directamente implicados en el proyecto.

9.3 INSPECCIÓN DE REGISTROS

Los manuales escritos sobre políticas y procedimientos de la empresa, le proporcionan al analista una visión clara y autónoma del sistema y le permite encontrar diferencias entre procesos planeados y reales.

9.4 LA OBSERVACIÓN

- Observar los procesos y operaciones.
- Proporciona información de primera mano.
- Constatación de los datos de la entrevista y del cuestionario.
- Identificación de problemas pasados por alto.

Cuando observar

- Cuando el analista necesita:
- Ver de primera mano los procesos.
- Manejo de documentos
- Verificar si ocurren los pasos especificados.

10. GUÍA PARA LA OBTENCIÓN REQUERIMIENTOS

El centro del modelo conceptual o foco está en :

- Las Preguntas claves.
- La Identificación de Procesos.
- La Priorización de procesos.
- El Modelamiento de los procesos

Las preguntas claves son:

- ¿Qué procesos integran el sistema?
- ¿Qué datos se utilizan en cada proceso?
- ¿Qué datos se almacenan?
- ¿Qué datos entran y salen del sistema?

Requerimientos Básicos

- Entender el proceso. ¿Cuál es el proceso básico?
- Identificación de los datos utilizados y la información producida. ¿Qué datos se utilizan o se producen durante este proceso?
- Determinación del tiempo del proceso y la cantidad. ¿Cuáles son los límites impuestos por tiempo y cantidad de trabajo?
- Identificación de controles. ¿Que controles de rendimiento se utilizan?

Preguntas básicas para identificar los afectados

- ¿Quién usará el sistema que se va a construir?
- ¿Quién desarrollará el sistema?
- ¿Quién probará el sistema?
- ¿Quién documentará el sistema?
- ¿Quién dará soporte al sistema?
- ¿Quién dará mantenimiento al sistema?
- ¿Quién mercadeará, venderá, y/o distribuirá el sistema?
- ¿Quién se beneficiará por el retorno de inversión del sistema?

Identificar a los afectados por el sistema

- Identificar a todos los afectados evita que existan sorpresas a medida que avanza el proyecto. Las necesidades de cada afectado, son discutidas y sometidas a debate.
- Saber quiénes son las personas, departamentos, organizaciones internas o externas que se verán afectadas por el sistema

Identificación de procesos

- El analista reconoce los flujos de datos.
- Comprende las transacciones y la realización de tareas.
- Identifica la entrada, proceso, almacenamiento, consulta, utilización, modificación y emisión de los datos

Entender el proceso

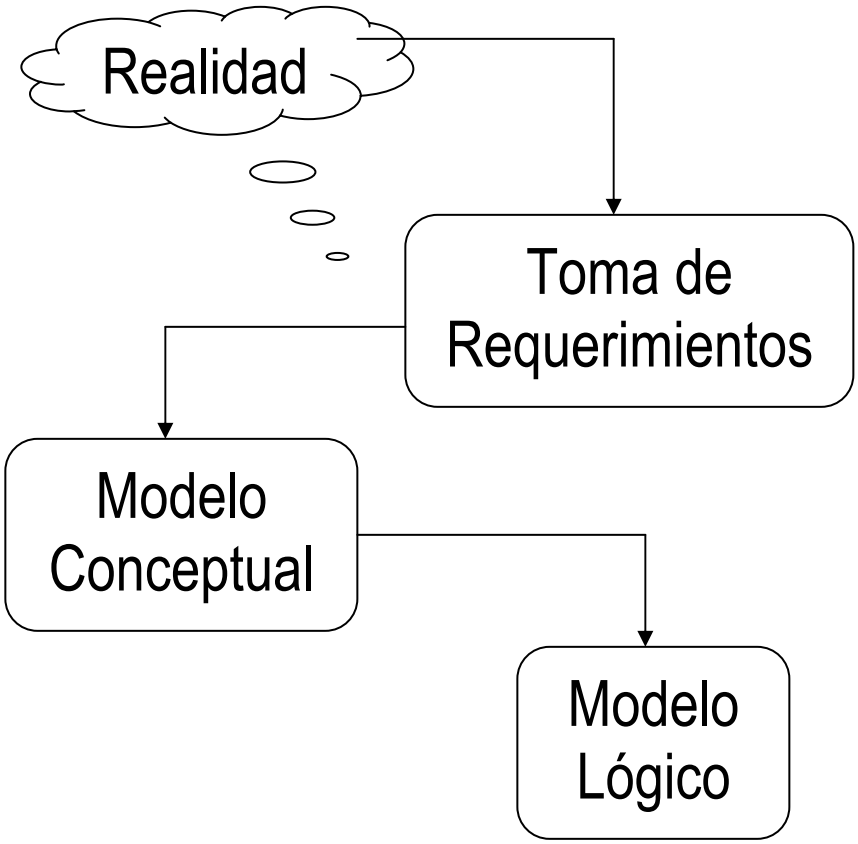
Se harán aquellas preguntas que proporcionarán un antecedente de los datos fundamentales y de las descripciones del sistema como:

¿Cuál es el propósito de esta actividad?
¿Cuáles son los pasos que se realizan?
¿Quién los ejecuta?
¿Cuánto tiempo consumen?
¿Con que frecuencia se realizan?
¿Quién utiliza la información resultante?

Priorización de procesos

- Identificar los procesos más importantes de los ya identificados.
- Determinar la dependencia entre procesos.
- Si existen procesos con igual prioridad determinar otras variables como disponibilidad de recursos.
- Determinar si el cliente quiere alguno en particular
- Es el paso intermedio entre la obtención de los requerimientos de los clientes y la construcción de un grafo (modelo lógico) que modele exactamente los datos y sus relaciones necesarias para satisfacer las necesidades del cliente.

El modelamiento conceptual tiene como objetivo mostrar funcionalmente el sistema; es decir, identificar sus elementos (entidades) y las relaciones entre ellos. De esta forma se incrementa el grado de detalle en la descripción del sistema.



En conclusión es bueno que se utilicen conectores que le permitan tener mayor claridad para hacer una buena toma de requerimientos tales como:

¿QUIÉNES están involucrados en la situación que se analiza? ¿Cuál es su papel? ¿Cuál es su nivel de entrenamiento?

¿CUÁL es la situación actual? ¿Cuáles son las manifestaciones del problema? ¿Cuáles son las funciones que se espera que haga el sistema propuesto?

¿CUÁNDO debe tenerse listo el nuevo sistema? ¿Cuándo se hará la transferencia? ¿Cuándo el cliente estará listo para instalar y probar en el sitio el nuevo sistema?

¿DÓNDE el nuevo sistema se integra con el viejo? ¿Dónde el personal actual entra al nuevo sistema?

¿POR QUÉ se piensa en un nuevo sistema? ¿Por qué los usuarios creen que sea necesario un nuevo sistema? ¿Cuál es la situación actual que requiere un cambio al nuevo sistema?

¿CÓMO debe funcionar el nuevo sistema? ¿Hay restricciones sobre su forma de operación, hardware, costos, lenguajes, etc.? Y ¿PARA QUÉ se hace el sistema?

ACTIVIDADES

Aplique en un caso práctico los siguientes ejemplos de actividades :

Ejemplo de Actividades propias de los requerimientos:

- Recepción de requerimientos
  Origen (cliente externo o interno)
  Descripción del requerimiento
  Almacenamiento y administración
- Evaluación
  Comercial (Contrato, acuerdos externos contrato, costos)
  Funcional (Roadmap producto)
- Aprobación
  Viabilidad (Desarrolladores)
  Costo (Cliente)
- Especificación
  Definición (Formato)
  Modelamiento
  Revisión
- Aprobación de especificaciones

Ejemplo Especificación de requerimientos

Situación actual y antecedentes
Objetivo del requerimiento
Descripción de proceso (Entradas, salidas)
Reglas del negocio
Excepciones
Glosarios de términos

Ejemplo definición de requerimientos registro a un centro de estudios.

Se debe poder ingresar la información de los estudiantes y permitir la consulta de esta. Debe tenerse un mecanismo de ingreso de la información personal de los estudiantes, y otro para la consulta de esta información.
Debe permitir el registro de la información personal, mediante un proceso de inscripción al centro de estudio, y la consulta por estudiante de esta información.

Debe proveer la funcionalidad del registro de estudiantes al centro de estudio. En el registro se consigna la información personal y académica del estudiante.

Información personal: Nombre, Cédula, sexo, Fecha de nacimiento, Ciudad de nacimiento, Teléfono, Dirección, estado civil.

Información académica: Código estudiante, plan académico, semestre en curso, promedio de notas.

Deben proveerse consultas sobre la información de los estudiantes por los siguientes criterios.

Código de estudiante
Número de identificación
Apellido
Programa de estudio
El resultado de las consultas debe presentarse por pantalla

¿Qué pasaría si el proceso de registro implicará...?
Creación ficha académica
Creación Registros de ingreso al centro
Actualización de estadísticas de registro
Impresión automática del carné del centro.

Se requiere una especificación más detallada del requerimiento en particular: Determinar objetivos, Descripción del proceso, entradas y salidas, Modelo del proceso.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad por parte del profesor.
- Exposición magistral de los conceptos inherentes a la ingeniería de requerimientos, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en Internet
- Orientación directa en el aula en el levantamiento de los requerimientos de un proyecto de desarrollo de software para la aplicación de los conceptos teóricos explicados y la utilización de la herramienta seleccionada

RECURSOS

- Humanos: profesor y alumnos.
- Institucionales: Salón de clases.
- Materiales: Módulo de Ingeniería del Software

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación del instrumento metodológico diseñado para levantar requerimientos
- Elaboración del documento de especificación de requerimientos

CRITERIOS DE EVALUACIÓN

1. Relaciones cada una de las definiciones de la columna A con los elementos de la columna B

a) Define la estructura general de un sistema y varía de acuerdo con el tipo de sistema a desarrollarse	( ) Proceso
b) Conjunto de instrucciones y procedimientos que se incluyen en un equipo de tratamiento de datos con el fin de hacer posible la utilización eficaz de la maquina	( ) Estrategia
c) Conjunto de elementos que se interrelacionan para lograr un objetivo común	( ) Requerimiento
d) Definen las reglas para las transformaciones internas de las actividades	( ) Actividad
e) Es un conjunto de datos arreglados en forma útil para tomar decisiones	( ) Software
f) Es un plan para lograr un objetivo	( ) Sistema
g) Es el estudio del sistema actual del negocio a fin de encontrar cómo trabaja y dónde debe mejorarse	( ) Herramientas
h) Es una unidad o paso básico de un proceso	( ) Métodos
i) Son aplicaciones que apoyan la administración del proceso de software	( ) Información
j) Es un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados	( ) Arquitectura

2. Hacer un paralelo con cada unos de los instrumentos metodológicos para levantar requerimientos.

UNIDAD 1

ANÁLISIS DEL SISTEMA

INTRODUCCIÓN

Al igual que muchas de las contribuciones importantes a la ingeniería del software, el análisis estructurado no fue introducido en un solo artículo o libro clave que incluyera un tratamiento completo del tema. Los primeros trabajos sobre modelos de análisis aparecieron a finales de los 60 y principios de los 70, pero la primera aparición del enfoque de análisis estructurado fue como complemento de otro tema importante - e l «diseño estructurado»-. Los investigadores necesitaban una notación gráfica para representar los datos y los procesos que los transforman.

Esos procesos quedarían finalmente establecidos en una arquitectura de diseño. El término análisis estructurado originalmente acuñado por Douglas Ross, fue popularizado por DeMarco. En su libro sobre esta materia, este autor presentó y denominó los símbolos gráficos y los modelos que los incorporan.

En los años siguientes, Gane y Sarson y muchos otros propusieron variaciones del enfoque del análisis estructurado. En todos los casos, el método se centraba en aplicaciones de sistemas de información y no proporcionaba una notación adecuada para los aspectos de control y de comportamiento de los problemas de ingeniería de tiempo real.

A mediados de los 80, las ampliaciones para tiempo real fueron introducidas por Ward y Mellor y, más tarde, por Hatley y Pirbhai . Con esas ampliaciones, se consiguió un método de análisis más robusto que podía ser aplicado de forma efectiva a problemas de ingeniería. En la actualidad, se está intentando desarrollar una notación consistente y se están publicando tratamientos modernos que permitan acomodar el uso de herramientas CASE.

JUSTIFICACIÓN

Dentro de un proyecto de desarrollo de software se hace necesario logra integrar las necesidades y requerimientos de información de un área, en un modelo ordenado, entendible por directivos y usuarios, permitiendo la maduración y perfeccionamiento del nuevo sistema de información llegando a los más mínimos niveles de detalle

Es así como el análisis estructurado es una actividad de construcción de modelos que busca mediante una notación satisfacer los principios de análisis operacional. Creamos modelos que representan el contenido y flujo de la información (datos y control); partimos el sistema funcionalmente, y según los distintos comportamientos, establecemos la esencia de lo que se debe construir. El análisis estructurado no es un método sencillo aplicado siempre de la misma forma por todos los que lo usan. Más bien, es una amalgama que ha evolucionado durante los últimos 30 años.

OBJETIVO GENERAL

Obtener el conocimiento detallado del sistema de información actual y de todos los requerimientos y necesidades de información adicionales que permitan modelar un proyecto de desarrollo de software.

OBJETIVOS ESPECÍFICOS

- Identificar los pasos que se llevan en la etapa de análisis de un sistema de información
- Diferenciar y aplicar técnicas de modelamiento de procesos.
- Determinar el funcionamiento interno de los procesos y los mecanismos de comunicación de los mismos.
- Establecer base para la creación del diseño
- Definir conjunto de requisitos a validar cuando se construye el software
- Realizar una reevaluación de los requerimientos iniciales, con base en un conocimiento más detallado del sistema, que permita hacer estimativos más reales, llegando a determinar factibilidades, alternativas y cronogramas más

precisos, además de confirmar o aclarar completamente el alcance planteado anteriormente.

- Describir lo que requiere el cliente y servir de puente de comunicación entre la parte administrativa y la parte técnica.
- Traducir el lenguaje del usuario a un lenguaje de sistemas, que permita su desarrollo posterior.
- Emitir una formulación específica de los modelos lógicos, que representen lo que va a ser el nuevo sistema, con base en los requerimientos planteados por el usuario.
- Encontrar lo que la organización requiere que se haga antes de comenzar a imaginarse cómo hacerlo.

CONTENIDO

1. DEFINICIÓN Y CARACTERÍSTICAS

Es la transformación disciplinada de los requerimientos de información de un sistema o área, en una ESPECIFICACIÓN FUNCIONAL, expresada en términos lógicos y usando metodologías estándares.

Es el proceso de determinar QUE se necesita hacer, antes de decidir COMO debe hacerse. Es el acto del descubrimiento.

Responde a las preguntas:

¿QUÉ ES LO QUE HACE EL SISTEMA?
¿QUÉ HARÁ EL NUEVO SISTEMA?

Se caracteriza por:

- Utilización de gráficas para representar el sistema, evitando al máximo, descripciones narrativas.
- Enfatiza en el flujo de información y no en los flujos de control de un sistema.
- Presenta una documentación extensa de cada tarea a realizar.

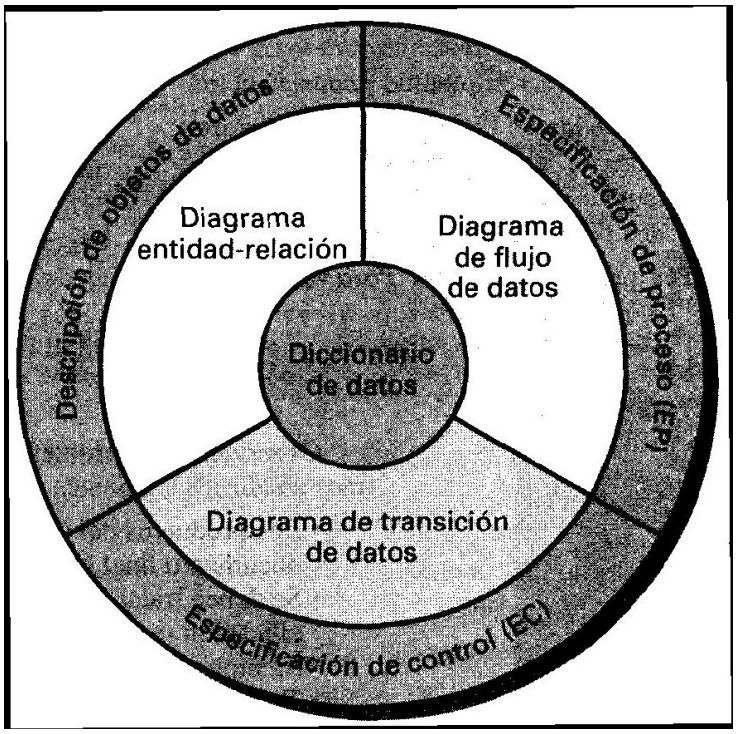
2. EL MODELADO DE ANÁLISIS

El modelo de análisis es realmente un conjunto de modelos, es la primera representación técnica de un sistema. Con los años se han propuesto muchos métodos para el modelado del análisis. Sin embargo, ahora dos tendencias dominan el panorama del modelado del análisis. El primero, análisis estructurado, es un método de modelado clásico y el otro enfoque, es el análisis orientado a objetos.

El análisis estructurado es una actividad de construcción de modelos. Mediante una notación que satisfaga los principios de análisis operacional creamos modelos que representan el contenido y flujo de la información (datos y control); partimos el sistema funcionalmente, y según los distintos comportamientos establecemos la esencia de lo que se debe construir. El análisis estructurado no es un método sencillo aplicado siempre de la misma forma por todos los que lo usan. Más bien, es una amalgama que ha evolucionado durante los últimos 30 años.

3. ELEMENTOS DEL MODELADO DE ANÁLISIS

El modelo de análisis debe lograr tres objetivos primarios: (1) describir lo que requiere el cliente, (2) establecer una base para la creación de un diseño de software, y (3) definir un conjunto de requisitos que se pueda validar una vez que se construye el software. Para lograr estos objetivos, el modelo de análisis extraído durante el análisis estructurado toma la forma ilustrada en la Figura



En el centro del modelo se encuentra el diccionario de datos -un almacén que contiene definiciones de todos los objetos de datos consumidos y producidos por el software-. Tres diagramas diferentes rodean el núcleo. El diagrama de entidad-relación (DER) representa las relaciones entre los objetos de datos. El DER es la notación que se usa para realizar la actividad de modelado de datos. Los atributos de cada objeto de datos señalados en el DER se pueden describir mediante una descripción de objetos de datos.

El diagrama de flujo de datos (DFD) sirve para dos propósitos: (1) proporcionar una indicación de cómo se transforman los datos a medida que se avanza en el sistema, y (2) representar las funciones (y subfunciones) que transforman el flujo de datos. El DFD proporciona información adicional que se usa durante el análisis del dominio de información y sirve como base para el modelado de función. En una especificación de proceso (EP) se encuentra una descripción de cada función presentada en el DFD.

El diagrama de transición de estados (DTE) indica cómo se comporta el sistema como consecuencia de sucesos externos. Para lograr esto, el DTE representa los diferentes modos de comportamiento (llamados estados) del sistema y la manera en que se hacen las transiciones de estado a estado. El DTE sirve como la base del modelado de comportamiento. Dentro de la especificación de control (EC) se encuentra más información sobre los aspectos de control del software.

Los componentes del modelado de análisis son:

DER: Notación utilizada para realizar la actividad del modelado de datos.

DFD: Indica cómo se transforman los datos a medida que se mueven en el sistema, y representa las funciones (y subfunciones) que transforman el flujo de datos.
DTE: diagrama de transición de estados
Diccionario de datos: Depósito que contiene definiciones de todos los objetos del modelo.
Miniespecificaciones: son la descripción detallada de los procesos o transformaciones de un sistema de información.

3.1 MODELADO DE DATOS - DER

El modelado de datos responde a una serie de preguntas específicas importantes para cualquier aplicación de procesamiento de datos

¿Cuáles son los objetos o entidades de datos que se van a procesar?
¿Cómo se componen y que atributos los describen?
¿Cuál es la relación entre los objetos y los procesos que los transforman?

Definir todos los datos que se introducen, almacenan, se transforman y se producen en la aplicación.

El modelado de datos y el DER proporcionan una notación concisa para examinar datos en el contexto de una aplicación.

Puede utilizarse:
- Como parte del modelo de análisis
- Diseño de la base de datos
- Soporte a otros métodos de análisis de requisitos

Conceptos Básicos

En esta sección se definen de manera formal los conceptos básicos del Modelo entidad/relación: entidad, relación y atributo.

Entidades

Una entidad es una cosa u objeto significativo (real o imaginario) acerca del cual se requiere conocer o almacenar información.

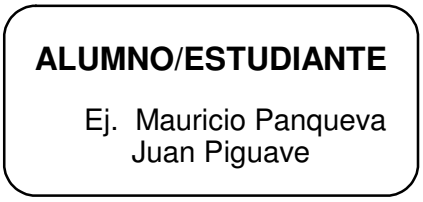
Las entidades se representan diagramáticamente mediante cajas de bordes redondeados, dentro de las cuales se coloca el nombre de la entidad. Este nombre es siempre singular y se presenta en mayúsculas.

El tamaño y disposición de la caja dentro del diagrama son arbitrarios, requiriéndose que posean suficiente espacio para colocar el nombre de la entidad (sin abreviaturas) y dispuestas para hacer el diagrama entidad/relación legible.

Así, es común alargar las cajas para permitir que las líneas de relación se conecten a ellas sin cruzarse o curvarse, con lo que el diagrama parecería una telaraña.

El nombre de la entidad debe representar la clase de objeto tratado, no una instancia. Así, los nombres Albert Einstein o Stephen Hawking no pueden nombrar una entidad; la entidad es CIENTÍFICO y los dos anteriores son instancias de esa entidad.

Es posible que haya sinónimos válidos de uso exactamente equivalente en el dominio de aplicación analizado. Se debe escoger un nombre primario; los sinónimos pueden mostrarse en mayúsculas, precedidos de una barra inclinada / (slash). Los ejemplos pueden mostrarse en mayúsculas y minúsculas.

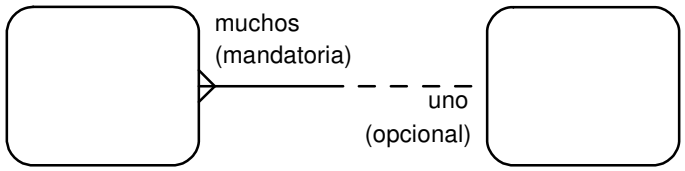


Toda cosa u objeto debe representarse exactamente mediante una entidad. Esto es, las entidades son mutuamente excluyentes en todos los casos.

Toda entidad debe ser identificable sin ambigüedad. Esto es, toda instancia de una entidad debe ser identificable de forma separada y distinta de todas las demás instancias de la misma entidad

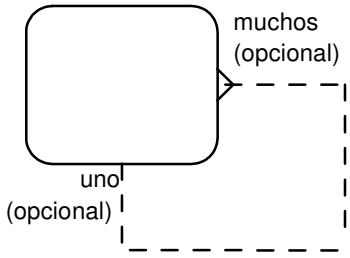
Relaciones

Una relación es una asociación nombrable, significativa y estable entre dos entidades.
Una relación es binaria, en el sentido de que corresponde siempre a la asociación entre exactamente dos entidades o de una entidad consigo misma.
Toda relación tiene dos extremos, para cada uno de los cuales existen:
Una leyenda.
Un grado o cardinalidad (uno o muchos).
Una opcionalidad (opcional o mandatoria).
Una relación se representa mediante una línea que conecta las cajas correspondientes a las dos entidades o que conecte recursivamente a una caja consigo misma.



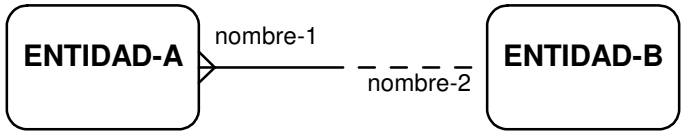
La relación más común es la de grado 1:N, mandatoria en el extremo N y opcional en el extremo 1.

Una relación recursiva es una jerarquía definida sobre una misma entidad, como se muestra en el diagrama.



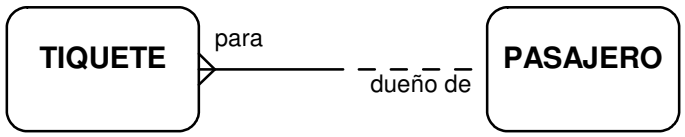
Este diagrama podría corresponder, por ejemplo, a la jerarquía de cuentas de un plan contable.

El nombre de cada relación se coloca en minúscula junto al extremo apropiado, como se muestra.



Cuando el extremo de la relación es mandatoria se emplea el verbo debe antes del nombre de la relación. Para relaciones opcionales, se emplea el verbo puede. Así, el diagrama anterior se lee de izquierda a derecha como:

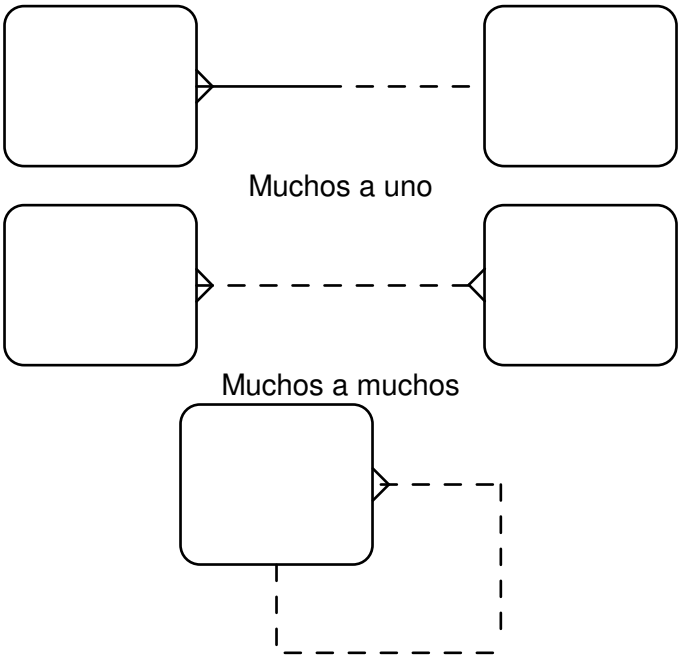
Cada ENTIDAD-A debe ser nombre-1 uno y sólo un ENTIDAD-B
y leído de derecha a izquierda:
Cada ENTIDAD-B puede ser nombre-2 uno o más ENTIDAD-A
Esto puede parecer incomprensible hasta que se lea un ejemplo real.



Cada TIQUETE debe ser para uno y sólo un PASAJERO y
Cada PASAJERO puede ser dueño de uno o más TIQUETES

El plural del nombre de la entidad se emplea cuando el grado es muchos. El grado de muchos se lee como uno o más. El grado uno se lee como uno y sólo uno. Al elaborar diagramas entidad/relación se logra mayor exactitud al colocar el extremo abierto (muchos) en el lado izquierdo o superior. Adicionalmente, el uso de los verbos ser y estar provee nombres de la relación mas significativos y útiles.

No todas las posibles combinaciones de opcionalidad y grado son válidas o frecuentes.

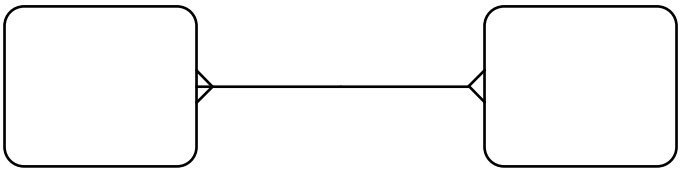


Muchos a uno

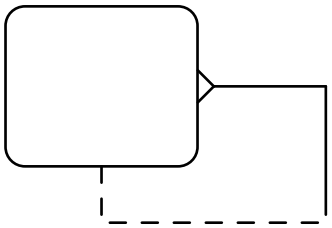
Muchos a muchos

Jerarquía recursiva de muchos a uno

Las relaciones anteriores son muy comunes. Las relaciones siguientes son inválidas, pues representan condiciones imposibles.



Relación Mandatoria de muchos a muchos



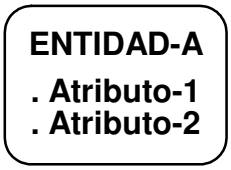
Jerarquía infinita

Atributos

Un atributo es cualquier detalle que sirve para: Identificar, Describir, Cualificar, Clasificar, Expresar el estado de una entidad.

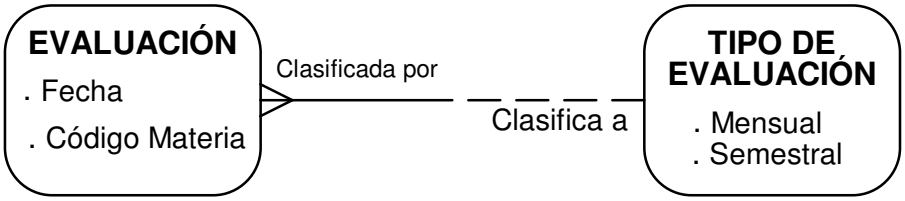
Un atributo puede ser texto, un número, una figura y así sucesivamente, según se requiera. En procesamiento de datos se tiende a utilizar únicamente texto y números, pero es razonable incluir otros tipos de datos tales como gráficos (por ejemplo, fotografías) o sonido.

Para representar un atributo, se escribe su nombre en minúscula y singular, opcionalmente acompañado de un ejemplo de su valor.



En un diagrama entidad/relación no es necesario incluir todos los atributos de cada entidad, pero añadir uno o dos durante el periodo inicial del Modelo es muy benéfico. En particular, es útil para distinguir entidades tipo de entidades instancia.

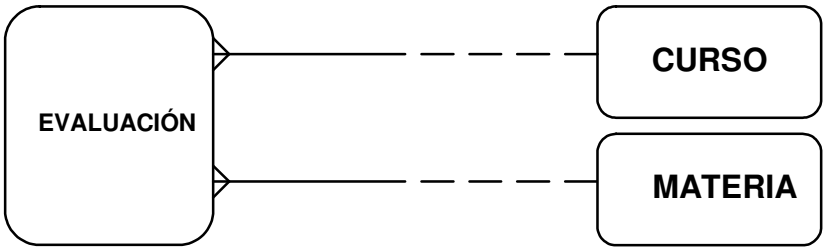
En el ejemplo de la siguiente figura, los atributos resultan necesarios para distinguir las dos entidades.



En esta caso, puede haber sólo cuatro o cinco tipos de evaluación, de acuerdo con su periodicidad, pero puede haber cientos de evaluaciones a lo largo del año escolar.

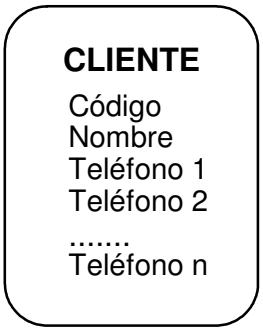
Los atributos deben describir únicamente a las entidades con las que están asociadas. Esto puede parecer obvio, pero es el error más común asociado con la identificación de atributos. Por ejemplo, ¿es 'nombre de materia' un atributo de evaluación o de materia propiamente dicho? Obviamente, es un atributo de materia, pero en el mundo real lo vemos replicado en muchos contextos, incluyendo la evaluación.

¿Por qué? En un sistema manual, usar el nombre de materia es una forma muy conveniente de representar una relación. Cuando se encuentran estas situaciones, se debe trazar una línea de relación, creando una nueva entidad si es necesario, como se ilustra en la figura.

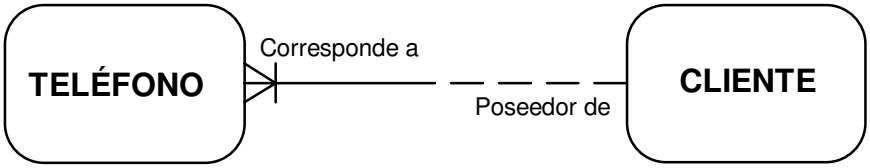


Como guía, una entidad sólo requiere ser descrita por entre dos y ocho atributos. Si se encuentran más de ocho, probablemente hay entidades o relaciones ocultas en el modelo. No se debe emplear el nombre de la entidad como parte del nombre del atributo, pues resulta redundante, ya que un atributo describe únicamente a una entidad.

Una entidad sólo puede tener un valor para cada atributo en un momento dado. Si resulta esencial tener múltiples valores, es necesario crear una nueva entidad para almacenarlos y una relación de muchos a uno con la entidad original, como se ilustra.



Siguiendo la regla anterior, tendríamos el diagrama de la figura siguiente.



Esta regla corresponde a la llamada primera forma normal.

El nombre de un atributo debe escribirse siempre en singular y minúscula. Un nombre plural coincide, generalmente, con el problema de repetición mencionado anteriormente.

Así mismo, la repetición de atributos puede revelar la existencia de entidades faltantes en el modelo.

Un atributo se transforma en una entidad cuando tiene significado completo en sí mismo, con relaciones y atributos propios.

Este es el caso del atributo Tipo Evaluación en la entidad EVALUACIÓN, que tiene como atributo propio la periodicidad en número de meses y debe, por tanto, ser tratado como una entidad en sí misma.

Identificador Único

Toda entidad debe poder ser identificada con unicidad mediante uno de sus atributos o una combinación de los mismos, denominado clave primaria o identificador único. Por tanto, siempre resulta necesario determinar qué atributo o atributos sirven para identificar una entidad.

Así mismo, todos los atributos de una entidad deben depender únicamente del valor de la clave primaria.

Si hay atributos que dependen sólo de parte de la clave primaria, deben ser removidos y trasladados a una nueva entidad. Esta operación conduce a la llamada segunda forma normal. Como ejemplo de ello tenemos la pregunta:

¿Depende el nombre de la materia del número de evaluación?

Obviamente no, el nombre de la materia no cambia cuando cambia el número de la evaluación y, por tanto, corresponde a una entidad separada, en este caso, MATERIA.

Atributos opcionales y mandatarios

Un atributo puede tener valor sólo durante una parte del tiempo o ser desconocido. Esta clase de atributos se denomina opcional y se representa mediante un pequeño círculo antepuesto al nombre del atributo.

El valor de un atributo que debe ser siempre conocido se representa mediante un asterisco (*) antepuesto al nombre del atributo.

Nótese que esto se aplica a los atributos cuyo valor debe ser siempre conocido, invalidando instancias de la entidad en que no se halla valor para estos.

Representación de atributos

Toda entidad debe ser identificable en forma única, de manera que cada instancia de la misma se pueda diferenciar de modo separado de las demás instancias de la misma entidad.

Los atributos empleados para diferenciar instancias conforman el identificador (o clave primaria) de la entidad.

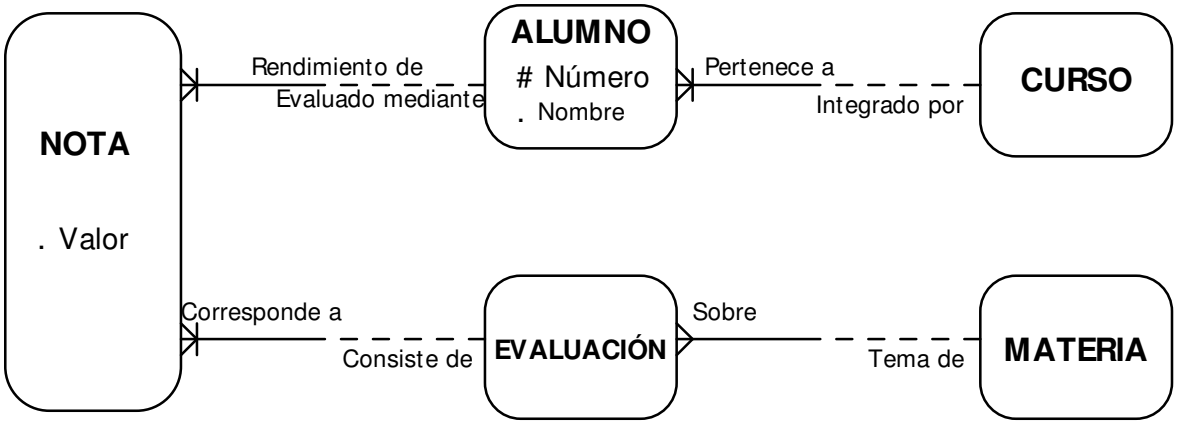
El identificador único puede ser un atributo, combinación de atributos o combinación de atributos y relaciones.

Nótese que una misma entidad puede tener más de una forma de identificación única.

Así, por ejemplo, un empleado puede identificarse tanto con su número de cédula como con un código interno asignado por la organización.

A los atributos que pueden emplearse alternativamente como identificadores de una entidad se les denomina claves candidatas.

El método básico de representar el identificador único en un diagrama entidad relación consiste en anteceder del signo # a cada atributo que contribuya al identificador único y en colocar una barra cruzada en la línea de la(s) relación(es) que participa(n) del identificador.



En este diagrama, para identificar a un alumno, se requiere el código de curso y el número de lista, en tanto que, para identificar la nota, se requiere el identificador de alumno unido con el identificador de evaluación.

Extensión al caso de estudio.

En nuestro ejemplo hemos apreciado las construcciones básicas empleadas para la elaboración de un modelo entidad relación, así como los conceptos básicos del Modelo.

En esta sección extenderemos el ejemplo para los colegios de bachillerato, examinando también algunas construcciones de agregación.

Planteamiento.

En bachillerato, un mismo profesor puede dictar una o más materias en varios cursos o paralelos, dándose también que en distintos cursos se estudien distintas materias.

Entre las entidades más relevantes de este dominio de aplicación tenemos:

CURSO	Cada uno de los grados anuales de estudio.
PARALELO	Cada uno de los grupos correspondientes a cada curso, identificados generalmente con las letras A, B, C, etc.
AULA	Espacio físico donde se aloja cada paralelo.

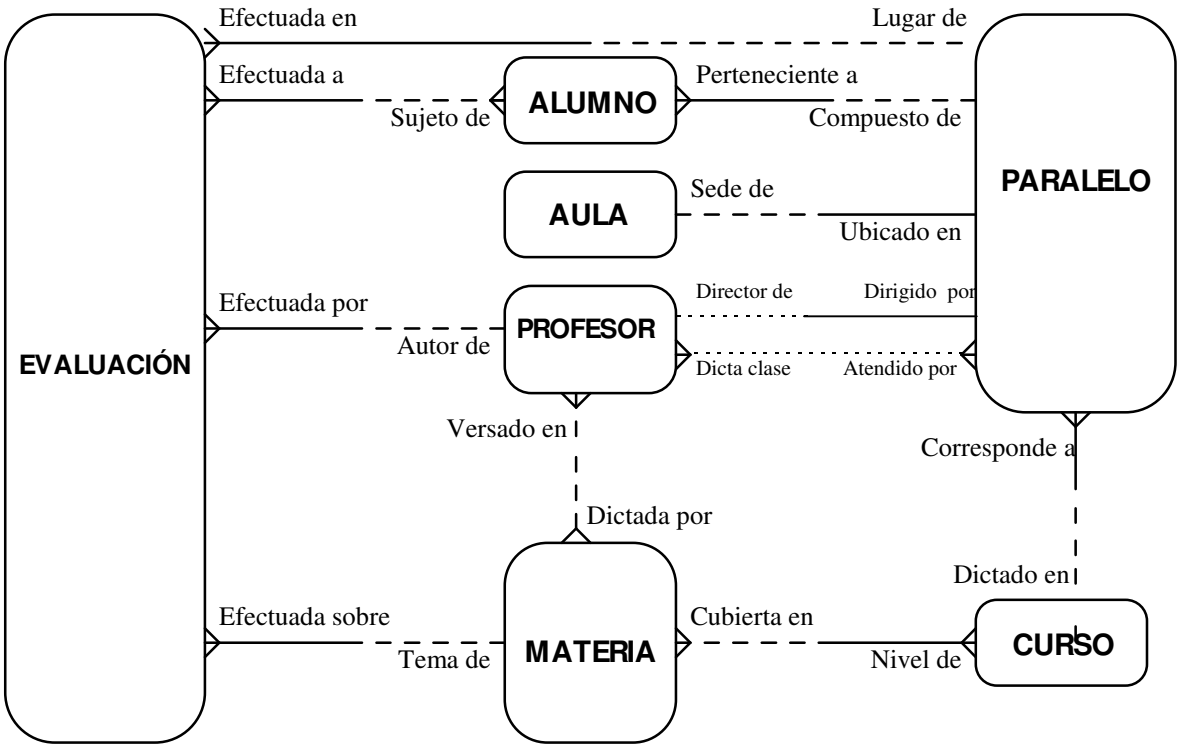
PROFESOR	Instructor encargado de dictar las materias correspondientes a cada curso en cada paralelo.
MATERIA	Asignatura objeto de estudio en los distintos cursos.
ALUMNO	O estudiante, asignado a cada paralelo.
EVALUACIÓN	Examen en el que se revisa el rendimiento de cada alumno en cada materia.

Entre estas entidades podríamos identificar, a priori, las siguientes relaciones:

CURSO/PARALELO	1:N Un mismo curso puede tener varios paralelos (nombrados A, B, C, etc.), en tanto que un paralelo debe corresponder a uno y sólo un curso.
PARALELO/AULA	1:1 Un mismo paralelo debe estar situado en una y sólo un aula, en tanto que un aula puede estar asignada a uno y sólo un paralelo.
PARALELO/ALUMNO	1:N Un paralelo debe tener uno o más alumnos, en tanto que un alumno debe pertenecer a uno y sólo un paralelo.
PROFESOR/PARALELO	1:1 Un profesor puede ser director de uno y sólo un paralelo, en tanto un que un paralelo debe tener uno y sólo un director de grupo.
PROFESOR/PARALELO	M:N Un profesor puede dictar clase en uno o más paralelos, en tanto que en un paralelo pueden dictar clase uno o más profesores.
PROFESOR/MATERIA	M:N Un profesor puede dictar una o más materias, en tanto que una misma materia puede ser dictada por uno o más profesores.
CURSO/MATERIA	M:N En un curso se debe dictar una o más materias, en tanto una misma materia puede ser dictada en uno o más cursos.
MATERIA/EVALUACIÓN	1:N Una misma materia puede ser objeto de una o más evaluaciones, en tanto una evaluación debe corresponder a una y sólo una materia.
PARALELO/EVALUACIÓN	1:N En un paralelo se pueden efectuar una o más evaluaciones, en tanto que una evaluación debe efectuarse en uno y sólo un paralelo.
PROFESOR/EVALUACIÓN	1:N Un profesor puede efectuar una o más evaluaciones, en tanto que cada evaluación debe haber sido efectuada por uno y sólo un profesor.
ALUMNO/EVALUACIÓN	M:N Un alumno puede presentar una o más evaluaciones, en tanto que una evaluación debe llevarse a cabo para uno o más alumnos.

¡Atención! Las relaciones mencionadas constituyen únicamente un punto de partida en el análisis y resultan de nuestra comprensión intuitiva del dominio de aplicación, siendo necesario refinarlas para llegar a un modelo completo y consistente.

El diagrama entidad/relación correspondiente a las entidades y relaciones identificadas es:



Examinando algunas de estas relaciones encontramos casos especiales dignos de Discusión.

Diferentes relaciones definidas sobre las mismas entidades.

Nótese que entre las entidades PROFESOR y PARALELO existen dos relaciones de diferente significado:

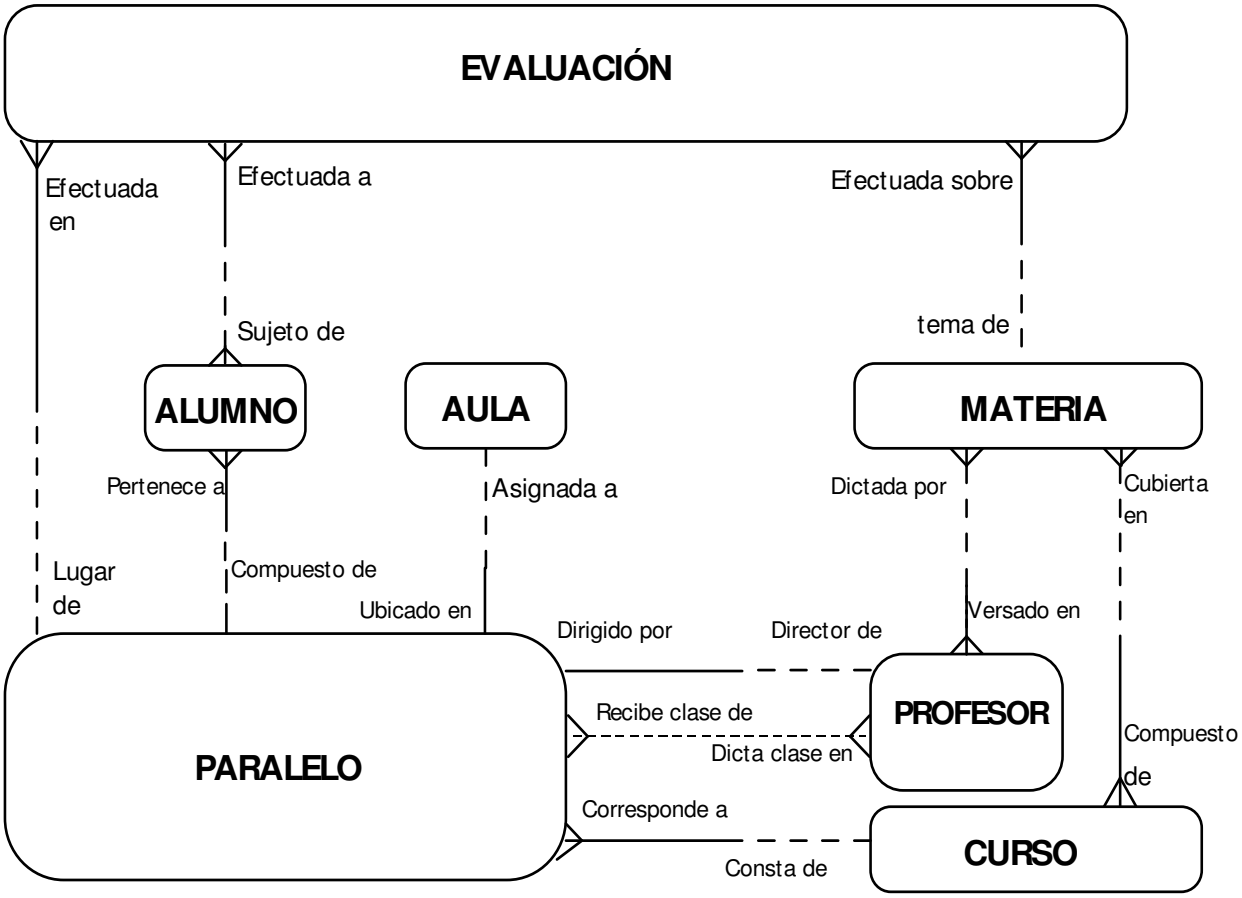
- Ser director de Grupo.
- Dictar clase en.

No es necesario dictar clase en un paralelo para dirigirlo, de la misma forma que es posible dictar clase en un paralelo y no ser director del mismo.

La existencia de varias relaciones diferentes definidas sobre las mismas entidades es bastante común y no se debe desestimar su posible ocurrencia en un modelo entidad/relación.

Reemplazo de relaciones 1:1 por relaciones 1:N.

En algunos casos, puede presentarse que la relación ser director de sea 1:N; esto es, que un mismo profesor pueda ser (simultáneamente) director de varios paralelos. En tal caso, la forma 1:N siempre podrá expresar también una relación 1:1, pudiendo, por tanto, substituirla.

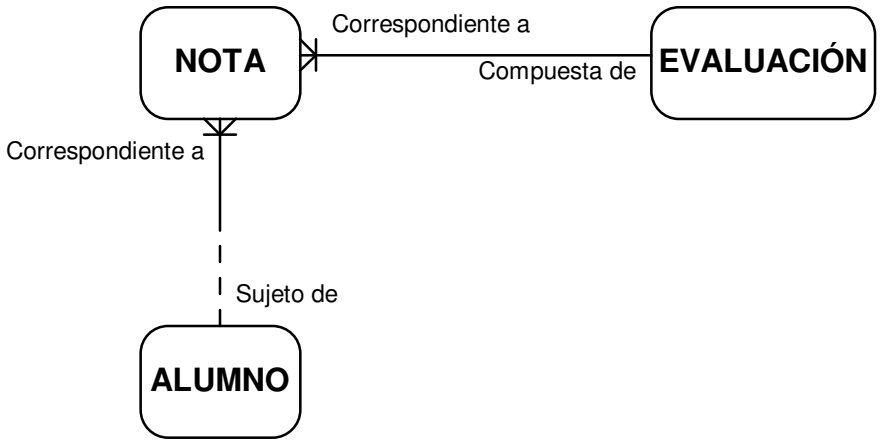


Entidades de intersección.

En la variedad de Modelo entidad/relación estudiada en este documento se trata a las relaciones M:N como entidades de intersección; esto es, para toda relación M:N identificada en el modelo se crea una nueva entidad que "intersecta" a las entidades participantes en la relación M:N y cuyo identificador único (clave primaria) se forma mediante la combinación de las claves primarias de dichas entidades.

La operación de considerar una relación M:N como entidad en sí misma se conoce como agregación.

Aplicando este procedimiento de resolución a la relación M:N existente entre ALUMNO y EVALUACIÓN, se crea una nueva entidad (NOTA) que intersecta a las dos anteriores, como se aprecia en el diagrama.



Examinando las relaciones:

CURSO/MATERIA	Pensum.
PROFESOR/MATERIA	Estar capacitado para dictar.
PROFESOR/PARALELO	Dictar clase en.

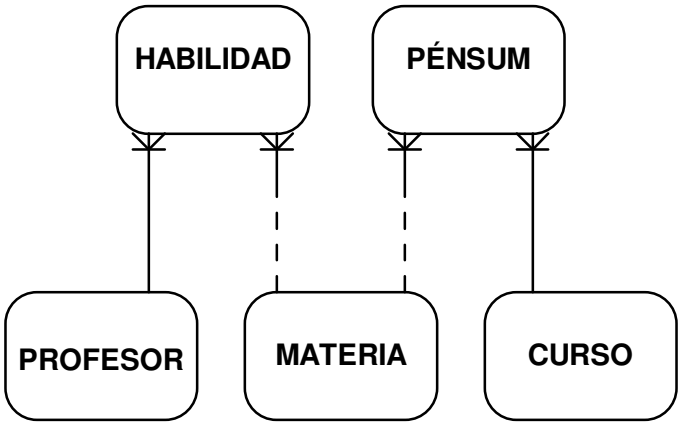
Encontramos que NO son relaciones independientes; esto es, no cualquier profesor puede dictar cualquier materia en cualquier paralelo.

De hecho, un profesor sólo puede dictar, en un paralelo dado, las materias que este capacitado para dictar en cada curso.

Así mismo, en un paralelo dado, únicamente se pueden estudiar aquellas materias definidas para el curso correspondiente al paralelo.

Esta dependencia corresponde a una forma de relación conocida como relación ternaria; esto es, una relación definida simultáneamente sobre tres entidades y que no siempre se puede descomponer en tres relaciones binarias independientes sin pérdida de información.

Aplicando la técnica de resolución a las relaciones M:N CURSO/MATERIA y PROFESOR/MATERIA, se define las entidades de intersección PENSUM y HABILIDAD, respectivamente.



Nótese en esta figura el uso de las barras perpendiculares trazadas sobre las líneas de relación.

Una barras perpendicular a la línea de relación indica que la entidad lado N construye su clave primaria involucrando la clave primaria de la entidad lado 1.

Como se aprecia en el diagrama, este es siempre el caso con las entidades de intersección.

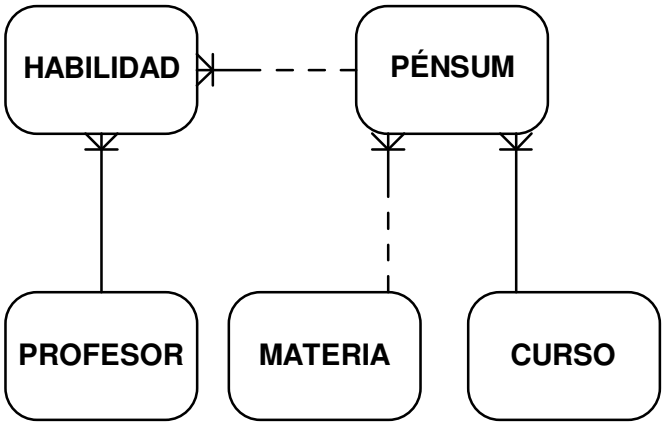
Este tipo de relación se conoce también como relación de dependencia y a la entidad de intersección se le denomina débil respecto a la(s) entidad(es) lado 1.

El modelo elaborado hasta ahora, sin embargo, contiene aún un defecto fundamental.

Un profesor habilitado para dictar matemáticas en 3o (álgebra) no necesariamente está habilitado para dictarlas en 6o (cálculo).

Es decir, no existe una relación directa entre PROFESOR y MATERIA, sino entre PROFESOR y PÉNSUM (esto es, simultáneamente con MATERIA y CURSO).

Reemplazando la relación entre HABILIDAD y MATERIA por la relación HABILIDAD y PENSUM, tenemos el diagrama siguiente:



En este diagrama reflejamos el hecho de que un PROFESOR puede estar habilitado para dictar la MATERIA matemáticas en los CURSOS 3o y 4o (álgebra), pero no en los CURSOS 5o y 6o (trigonometría y cálculo).

De esta manera, un profesor no se relaciona de forma directa con la entidad MATERIA en general, sino con la intersección PENSUM existente entre MATERIA y CURSO.

Esta técnica permite resolver relaciones ternarias en relaciones binarias sin pérdida de información.\

La construcción de "promover" una relación M:N a una entidad y definir nuevas relaciones sobre dicha entidad se denomina agregación y representa una de las herramientas conceptuales más poderosas del Modelo entidad/relación.

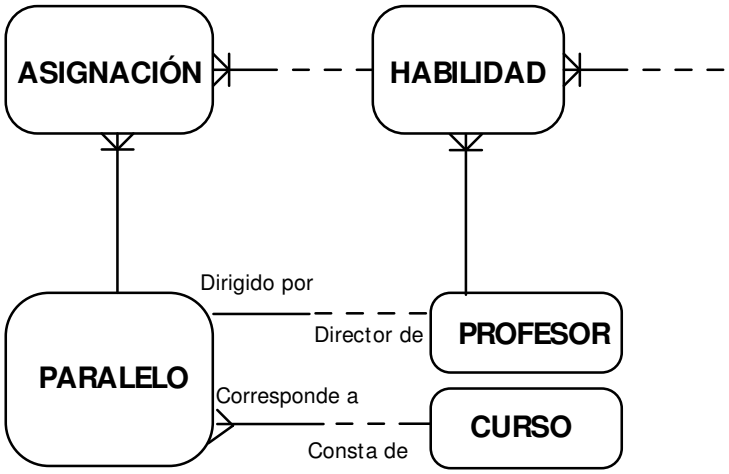
Aplicando mecánicamente esta transformación a la relación M:N existente entre PROFESOR y PARALELO (dictar clase en), se podría definir una entidad de intersección, denominada ASIGNACIÓN, que resolvería la relación.

Esta entidad de intersección, sin embargo, no se puede definir directamente entre PROFESOR y PARALELO, pues permitiría asociar cualquier PROFESOR con cualquier PARALELO.

Un análisis más cuidadoso revela que sólo es posible asignar un PROFESOR a un PARALELO si el PROFESOR está habilitado par dictar la MATERIA dada en el CURSO correspondiente al PARALELO.

Debido a ello, ASIGNACIÓN debe intersectar las entidades PARALELO y HABILIDAD, en vez de PARALELO y PROFESOR.

Es de notar que (aunque el modelo no lo determina explícitamente), el CURSO correspondiente al PENSUM asociado con la HABILIDAD del PROFESOR debe ser el mismo CURSO correspondiente al PARALELO con el cual se asocia la HABILIDAD.



Esta construcción se denomina clausura transitiva de la relación.

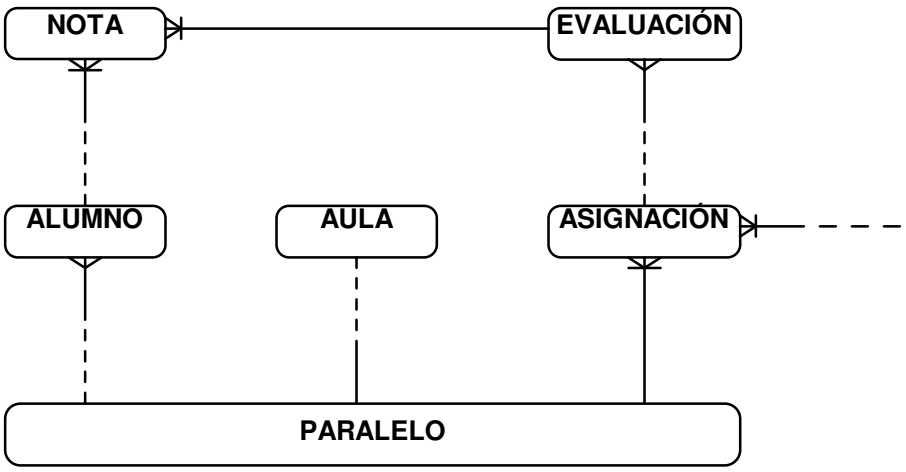
Por último, las relaciones directas

MATERIA/EVALUACIÓN	Ser tema de
PROFESOR/EVALUACIÓN	Ser autor de
PARALELO/EVALUACIÓN	Ser lugar de

No son tampoco independientes entre sí.

De hecho, sólo el PROFESOR que haya sido asignado para dictar MATERIAS en un PARALELO particular (mediante la entidad de intersección ASIGNACIÓN) puede originar EVALUACIONES.

Estas relaciones, por tanto, pueden reemplazarse por una única relación entre ASIGNACIÓN y EVALUACIÓN como se ilustra en el diagrama.



Es de notar que el paralelo correspondiente a la EVALUACIÓN (a través de la entidad ASIGNACIÓN) debe corresponder al PARALELO de los ALUMNOS participantes en la EVALUACIÓN.

Este es, nuevamente, el caso de una clausura transitiva de la relación.

Modelo final.

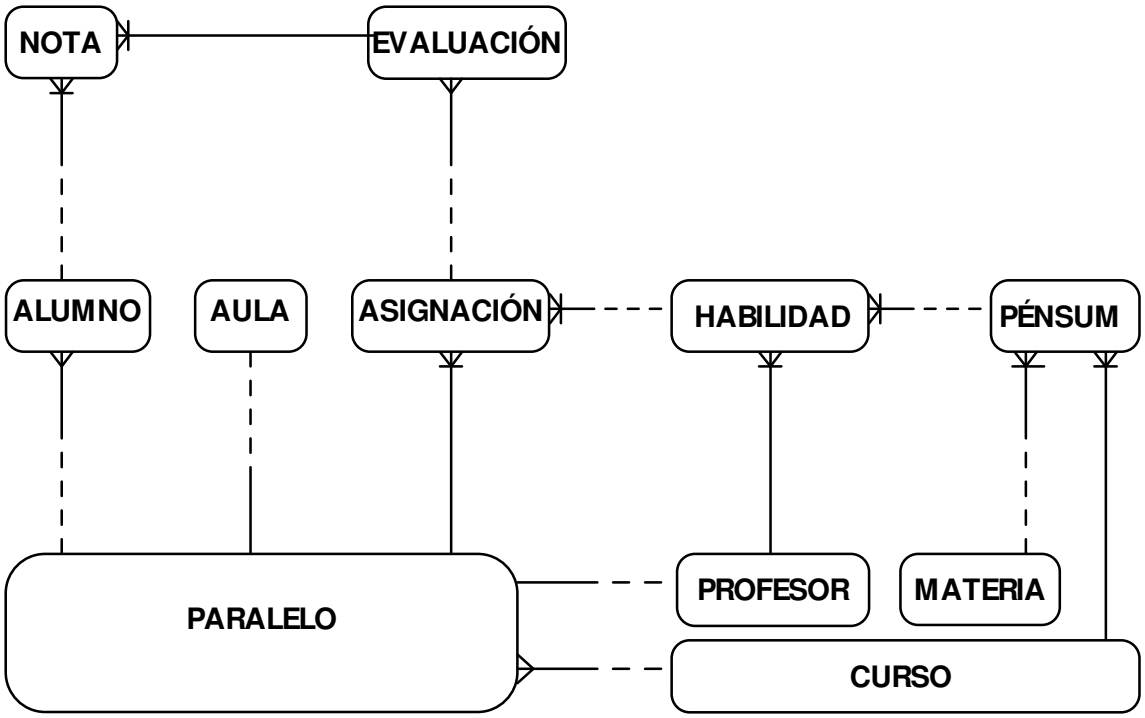
Después de estas transformaciones, resultantes de un análisis más detallado, tenemos como lista de entidades:

CURSO	Cada uno de los grados anuales de estudio.
PARALELO	Cada uno de los grupos correspondientes a cada curso, identificados generalmente con las letras A, B, C, etc..
AULA	Espacio físico donde se aloja cada paralelo.
PROFESOR	Instructor encargado de dictar las materias correspondientes a cada curso en cada paralelo.
MATERIA	Asignatura objeto de estudio en los distintos cursos.
ALUMNO	O estudiante, asignado a cada paralelo.
PÉNSUM	Materias cubiertas por el programa de un curso particular.
HABILIDAD	Materias susceptibles de ser dictadas por un profesor en un curso particular.
ASIGNACIÓN	Asociación entre un profesor, una materia en un curso particular y un paralelo dado.
EVALUACIÓN	Examen en el que se revisa el rendimiento de cada alumno en cada materia.
NOTA	Rendimiento obtenido por un alumno dado en una evaluación particular.

La lista de relaciones identificadas es:

CURSO/PARALELO	1:N Un mismo curso puede tener varios paralelos (nombrados A, B, C, etc.), en tanto que un paralelo debe corresponder a uno y sólo un curso.
PARALELO/AULA	1:1 Un mismo paralelo debe estar situado en una y sólo un aula, en tanto que un aula puede estar asignada a uno y sólo un paralelo.
PARALELO/ALUMNO	1:N Un paralelo puede tener uno o más alumnos, en tanto que un alumno debe pertenecer a uno y sólo un paralelo.
CURSO/PENSUM	1:N con dependencia. Un curso debe estar presente en varias líneas de pensum, en tanto que cada línea de pensum debe referenciar a uno y sólo un curso.
MATERIA/PENSUM	1:N con dependencia. Una materia puede estar presente en varias líneas de pensum, en tanto que cada línea de pensum debe referenciar a una y sólo una materia.
PENSUM/HABILIDAD	1:N con dependencia. Una línea de pensum (curso, materia) puede estar presente en varias líneas de habilidad, en tanto que una línea de habilidad debe corresponder a una y sólo una línea de pensum.
PROFESOR/HABILIDAD	1:N con dependencia. Un profesor debe estar asociado a varias líneas de habilidad, en tanto que una línea de habilidad debe estar asociada con uno y sólo un profesor.
PROFESOR/PARALELO	1:1. Un profesor puede estar asociado a una y sólo una línea en paralelo, en tanto que una línea de paralelo debe estar asociada con una y sola una línea en profesor.
HABILIDAD/ASIGNACIÓN	1:N con dependencia. Una línea de habilidad (profesor, materia, curso) puede estar asociada con varias asignaciones, en tanto que cada línea de asignación debe estar asociada con una y sólo una habilidad.
PARALELO/ASIGNACIÓN	1:N con dependencia. Un paralelo debe estar asociado con una o más líneas de asignación, en tanto que una línea de asignación debe referenciar a uno y sólo un paralelo.
ASIGNACIÓN/EVALUACIÓN	1:N Una misma asignación (profesor, materia, curso, paralelo) puede ser objeto de una o más evaluaciones, en tanto que una evaluación debe corresponder a una y sólo una asignación.
EVALUACIÓN/NOTA	1:N con dependencia. Una misma evaluación debe tener una o más notas, en tanto que una nota debe corresponder a una y sólo una evaluación.
ALUMNO/NOTA	1:N con dependencia. un alumno puede tener una o más notas, en tanto que una nota debe corresponder a uno y sólo un alumno.

El diagrama completo para el modelo en estudio es:



Identificación de entidades, relaciones y atributos.

En el ejemplo anterior, hemos efectuado un análisis de entidades y relaciones, sin abordar aún la identificación de atributos.

Como puede apreciarse, el Modelo entidad/relación no es una técnica de aplicación mecánica que provea, mediante una secuencia estricta y simple de actividades, el modelo preciso.

Excepto en los casos más simples, siempre resulta necesario revisar varias veces el modelo resultante del análisis para capturar con la mayor exactitud posible las propiedades del dominio modelado.

Esta tarea involucra muy de cerca al usuario final, como único árbitro definitivo de las dudas e inconsistencias encontradas, así como para la eliminación de relaciones irrelevantes.

Así, en el ejemplo anterior, no se contemplaron relaciones carentes de importancia para el dominio de aplicación tales como:

ALUMNO/MATERIA Qué relación existe?. Quizás que el alumno dado es muy aficionado a la materia. Es esta relación relevante?. Podría serlo si el sistema incluyera datos de tipo vocacional, pero no si se limita a calificaciones.

PROFESOR/ALUMNO Que relación existe?. Quizás que el alumno y el profesor son buenos amigos. Es esta relación relevante?.

La inclusión de relaciones redundantes es uno de los errores más comunes en el Modelo. Este es el caso con las dos relaciones anteriormente citadas. Un analista principiante podría sentirse tentado a incluir estas relaciones para responder a preguntas tales como:

- Cuál es la nota promedio de un ALUMNO dado para una MATERIA dada?.
- Qué profesores dictan clase sobre cada MATERIA a qué ALUMNOS?.

Nótese, sin embargo, que para responder estas preguntas basta con recorrer las relaciones ya definidas en el orden apropiado, siendo por tanto, redundante establecer relaciones directas entre las entidades involucradas.

Así, dado un ALUMNO, se puede enlazar todas sus NOTAS, seleccionando para cada una únicamente aquellas correspondientes a una MATERIA dada y así calcular la nota promedio.

Igualmente, dado un PROFESOR, se puede enlazar todas sus HABILIDADES, enlazando a continuación sus ASIGNACIONES para determinar el PARALELO, a partir del cual se pueden enlazar todos los ALUMNOS.

Podría alegarse, quizás, que las primeras relaciones citadas en la sección anterior (aquellas identificadas a priori) eran suficientes para construir un sistema automatizado, sin entrar en las sutilezas de qué profesor puede dictar qué materia o qué materias se cubren en qué cursos.

Lo esencial en un sistema automático, sin embargo, no es sólo la habilidad de almacenar los datos propios del dominio de aplicación , sino también (y muy especialmente) garantizar su consistencia.

A este respecto podría decirse que uno de los objetivos centrales del Modelo es modelar en la estructura de datos (NO en los programas de aplicación) la mayor cantidad posible de propiedades dinámicas del dominio de aplicación.

Así, se garantiza que el modelo de datos permite controlar inconsistencias que, de otra manera, deberían ser chequeadas por los programas de aplicación. Por supuesto, esto redunda en una reducción de la complejidad y dificultad en el mantenimiento de las aplicaciones.

Extendiendo este concepto, es fácil observar que una gran cantidad de lógica procedimental necesaria en los programas de aplicación puede obviarse mediante la simple inclusión de entidades y relaciones de carácter abstracto en el modelo de datos.

Pasos del Modo.

Una vez dotados de la comprensión básica de los conceptos y construcciones del Modelo entidad/relación podemos planear la secuencia de pasos necesaria para elaborar un modelo completo. Estos pasos son:

1. Identificación entidades.
2. Identificación relaciones.
3. Elaborar el diagrama entidad/relación.
4. Identificar atributos.
5. Refinar el modelo, comparándolo contra los requerimientos establecidos para el dominio de aplicación.
6. Transformar el modelo entidad/relación a un diseño lógico implementable.

Por ejemplo, a un esquema de tablas, vistas e índices de una base de datos relacional.

Identificación de entidades.

La identificación de entidades es el paso central del proceso de Modelaje. Una simplificación útil de los diversos tipos de entidad es:

Personas naturales o jurídicas:	Tales como ALUMNO, PROFESOR, CLIENTE, EMPLEADO, PROVEEDOR, etc..
Objetos	Tangibles o no, tales como ARTÍCULO, CUENTA CORRIENTE, CUENTA CONTABLE, etc..
Transacciones	O eventos, tales como EVALUACIÓN, COMPRA, DESPACHO, CONSIGNACIÓN, TRANSFERENCIA, etc..
Lugares	Tales como AULA, CIUDAD, BODEGA, etc..
Conceptos	O abstracciones, tales como TIPO DE EVALUACIÓN, TIPO DE CLIENTE, TIPO DE TRANSACCIÓN, UNIDAD DE MEDIDA, etc..

En términos generales, se presenta que sólo las personas pueden originar transacciones y que la acción de estas recae siempre sobre un objeto dado.

Por otra parte, las personas (y objetos) suelen residir (o almacenarse) en lugares.

Así mismo, los conceptos permiten controlar y limitar qué tipo de persona puede llevar a cabo qué tipo de transacción y qué tipo de transacción se puede aplicar sobre cada tipo de objeto.

Identificación de relaciones.

En un sistema comercial típico (transaccional) una buena guía para iniciar al análisis es preguntarse:

- Qué transacción ocurre en el modelo?. Transacciones: VENTA, DESPACHO, TRANSFERENCIA, ASIENTO CONTABLE, etc..
- Quién origina cada transacción?. Personas: CLIENTE, PROFESOR, PROVEEDOR, etc..
- Sobre qué objeto recae la acción de la transacción?. Objetos: ARTÍCULO, CUENTA CORRIENTE, CUENTA CONTABLE, etc..
- De qué lugares provienen las personas, en qué lugares se almacenan o procesan los objetos?. Lugares: BODEGA para ARTÍCULO, CIUDAD para PROVEEDOR, AULA para PARALELO, etc..
- De qué depende la naturaleza de las transacciones, personas u objetos?. Conceptos: UNIDAD para ARTÍCULO, TIPO DE TRANSACCIÓN para TRANSACCIÓN BANCARIA, CLASE DE CLIENTE para CLIENTE, etc.

Elaboración del diagrama entidad/relación.

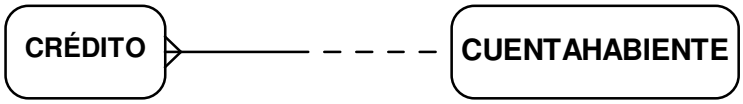
La importancia de los diagramas entidad/relación no debe ser, de ninguna manera, desestimada, como (desafortunadamente) ocurre con la documentación en los sistemas convencionales.

El diagrama entidad/relación no es únicamente un medio de documentación o revisión rápida. Es, ante todo, un instrumento de comunicación con el usuario.

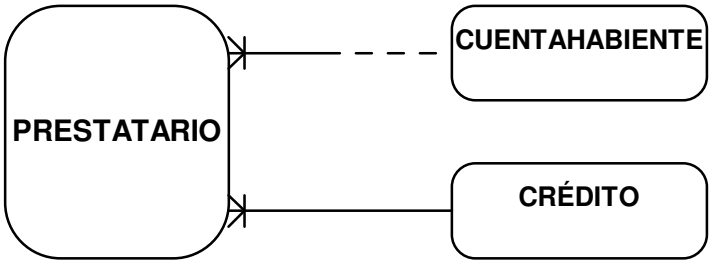
Una experiencia de años en el Modelo entidad/relación muestra que los usuarios finales aprenden con mucha facilidad a interpretar diagramas entidad/relación, validando la completitud y consistencia del mismo, dirimiendo diferencias e identificando formas más adecuadas de modelar.

Veamos algunos ejemplos de observaciones hechas por usuarios a modelos entidad/relación sometidos a su revisión:

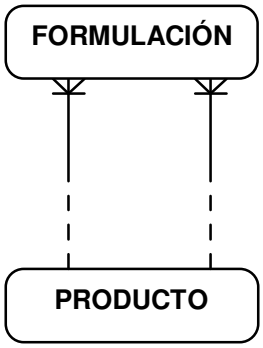
1. "Sí entiendo correctamente lo que dice el diagrama, creo que es un error considerar que un crédito pueda ser conferido a uno y sólo un cuenta-habiente. Es perfectamente posible que dos o más clientes del banco apliquen para un mismo crédito, en cuyo caso se hacen igualmente solidarios por la deuda".



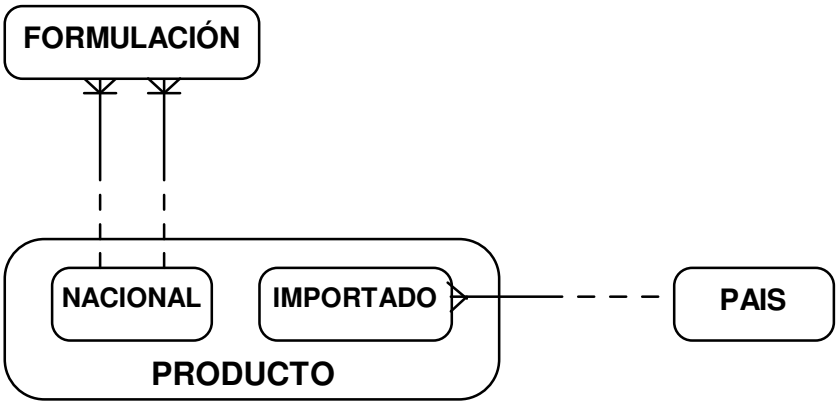
Una relación inicialmente identificada como 1:N resulta ser M:N y requiere, por tanto, de una nueva entidad (de intersección)



2. "No sé si me equivoco, pero no creo que todos los productos tengan una formulación. Nosotros vendemos también compuestos importados para los cuales ni conocemos ni nos interesa la formulación".



Una relación debe resulta ser una relación puede y se introduce una nueva entidad (producto importado) con atributos y relaciones propias.



Valide siempre sus diagramas entidad/relación con el usuario final.

Identificación de atributos.

La identificación de atributos es el paso siguiente después de la elaboración (y aprobación) del diagrama entidad/relación.

Es posible que durante este proceso se identifiquen nuevas entidades y relaciones, sutilmente ocultas en el modelo original.

No se desanime. Esa es la naturaleza progresiva del Modelo entidad/relación.

Identificador único.

El primero de los atributos que debe ser identificado para cada entidad es el identificador o clave primaria.

Toda entidad debe tener una clave primaria. En los raros casos en que el Modelo no evidencie un atributo (o combinación de atributos) que conformen un identificador único, es lícito "inventar" uno utilizando, por ejemplo, números consecutivos, como en los documentos.

Número de atributos**.**

Un chequeo de calidad importante es verificar que toda entidad tenga, por lo menos, dos atributos: La clave primaria y un descriptor.

Este criterio no se aplica necesariamente a las entidades de intersección, que pueden consistir, únicamente, de las claves primarias de las entidades intersectadas.

Caso de estudio. La identificación de atributos par nuestro caso de estudio sería:

CURSO
- # número
- nombre

PARALELO
- # letra

AULA
- # número
- capacidad (en número de alumnos)

PROFESOR
- # número
- nombre
- sexo

ALUMNO
- # número
- nombre
- sexo

MATERIA
- # código
- nombre
- área (sociales, técnicas, humanidades, etc.)

PENSUM
- Intensidad (en número de horas por semana)

HABILIDAD
- Calificación (excelente, bueno, aceptable)

ASIGNACIÓN
- fecha_inicial
- fecha_final

EVALUACIÓN
- # número
- tipo de evaluación (mensual, semestral, final)
- fecha

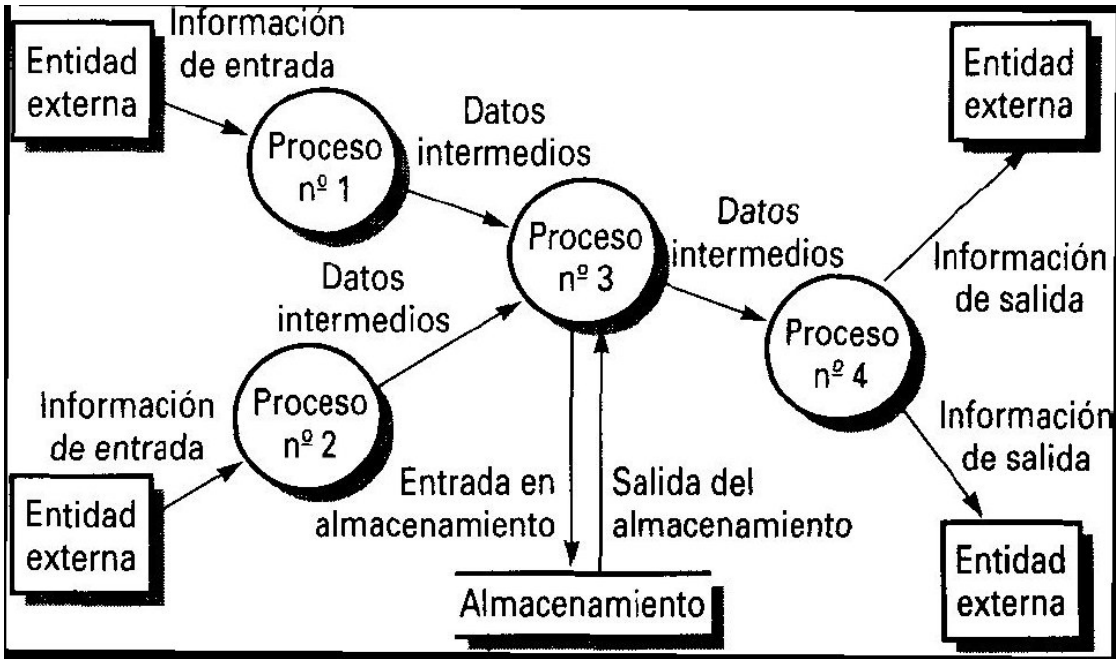
NOTA
- calificación (por ejemplo, de 1 a 10)

3.2 MODELADO FUNCIONAL Y FLUJO DE INFORMACIÓN- DFD

La información se transforma a medida que fluye por un sistema basado en computadora. El sistema acepta entradas en una gran variedad de formas; aplica elementos de hardware, software y humanos para transformar la entrada en salida, y produce salida en una gran variedad de formas. La entrada puede ser una señal de control transmitida por un controlador, una serie de números escritos por un enlace de una red o un archivo voluminoso de datos recuperado de un almacenamiento secundario.

La transformación puede ser, desde una sencilla comparación lógica, hasta un complejo algoritmo numérico o un mecanismo de reglas de inferencia de un sistema experto. La salida puede ser el encendido de un diodo de emisión de luz (LED) o un informe de 200 páginas.

Efectivamente, podemos crear un modelo de flujo para cualquier sistema de computadora, independientemente del tamaño y de la complejidad.



Ejemplo de un Modelo del flujo de información.

El análisis estructurado es una técnica del modelado del flujo y del contenido de la información. Tal como muestra el ejemplo anterior, el sistema basado en computadora se representa como una transformación de información.

Se utiliza un rectángulo para representar una entidad externa, esto es, un elemento del sistema (por ejemplo, un elemento hardware, una persona, otro programa) u otro sistema que produce información para ser transformada por el software, o recibe información producida por el software. Un círculo (también llamado burbuja) representa un proceso o transformación que es aplicado a los datos (o al control) y los modifica. Una flecha representa uno o más elementos de datos (objetos de dato). Toda flecha en un diagrama de flujos de datos debe estar etiquetada. Las líneas en paralelo representan un almacenamiento -información almacenada que es utilizada por el software-. La simplicidad de la notación del DFD es una razón por la que las técnicas del análisis estructurado son ampliamente utilizadas.

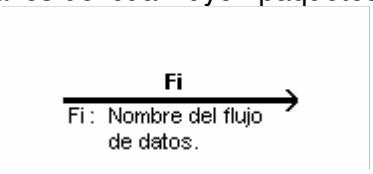
Es conveniente hacer notar que el diagrama no proporciona ninguna indicación explícita de la secuencia de procesamiento o de una condición lógica. El procedimiento o secuencia puede estar implícitamente en el diagrama, pues los detalles lógicos son generalmente retrasados hasta el diseño del software. Es importante no confundir un DFD con el diagrama de flujo.

El DIAGRAMA DE FLUJO DE DATOS : Ayuda a dividir los requerimientos y documentar su particionamiento en forma gráfica.

Permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "conductos" y "tanques de almacenamiento" de datos

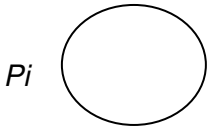
ELEMENTOS QUE LO COMPONEN :

- Flujo de datos : conducto a través del cual fluyen paquetes de información.



- Procesos o transformaciones: Es el lugar de donde la información sufre cambios, como consecuencia de cálculos u operaciones matemáticas o lógicas.

Representación Pi : Nombre del Proceso



- Almacenamiento : Lugar donde se acumula la información dentro del sistema, con el propósito de utilizarse en procesos posteriores.

Representación

Ai: Nombre del Almacenamiento

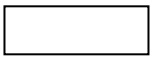
Ai _____

- Terminal : Persona u organización que reside fuera del contexto del sistema mismo y que origina y recibe la información del sistema. Hay dos tipos :

- Fuente : Persona o departamento donde se origina la información de entrada.

- Sumidero o destino : Persona u organización hacia donde va dirigida la información generada por el sistema.

Representación Ti : Nomb.de la Terminal

Ti 

COMO CONSTRUIR UN D.F.D

1. Definir flujos de entrada y salida del sistema, además de las fuentes y sumideros, de donde y hacia donde se dirige la información.

2. Identificar los procesos del D.F.D.

3. Identificar los almacenamientos necesarios.

4. Identificar todos los flujos de información internos
5. Definir con detalle el nombre de los flujos de información, de los procesos y de los almacenamientos.

6. Revisar el D.F.D. planteado, observando que sea consistente, completo y claro. Este proceso se repite hasta tener un D.F.D muy perfeccionado.

7. Expandir en otros D.F.D. los procesos que requieran mayor detalle en su definición.

NIVELES DE UN D.F.D

1. Diagrama de Contexto (Nivel 0): Representa el conjunto de entradas y salidas del sistema; sirve para dar una visión global del medio ambiente y está conformado por un solo proceso.

2. Diagrama de Flujo de Datos General (Nivel 1): Representa la expansión del único proceso del diagrama de contexto. Descubre la conformación general del sistema de información propuesto.

3. Nivel 2 : Resulta de expandir los procesos del DFD general en otros DFD

4. Nivel 3 ….Nivel n : Resultan de la expansión sucesiva del proceso del nivel 2.

Esta debe hacerse hasta que el sistema quede lo suficientemente claro y no necesariamente todos los procesos se expanden al mismo nivel.

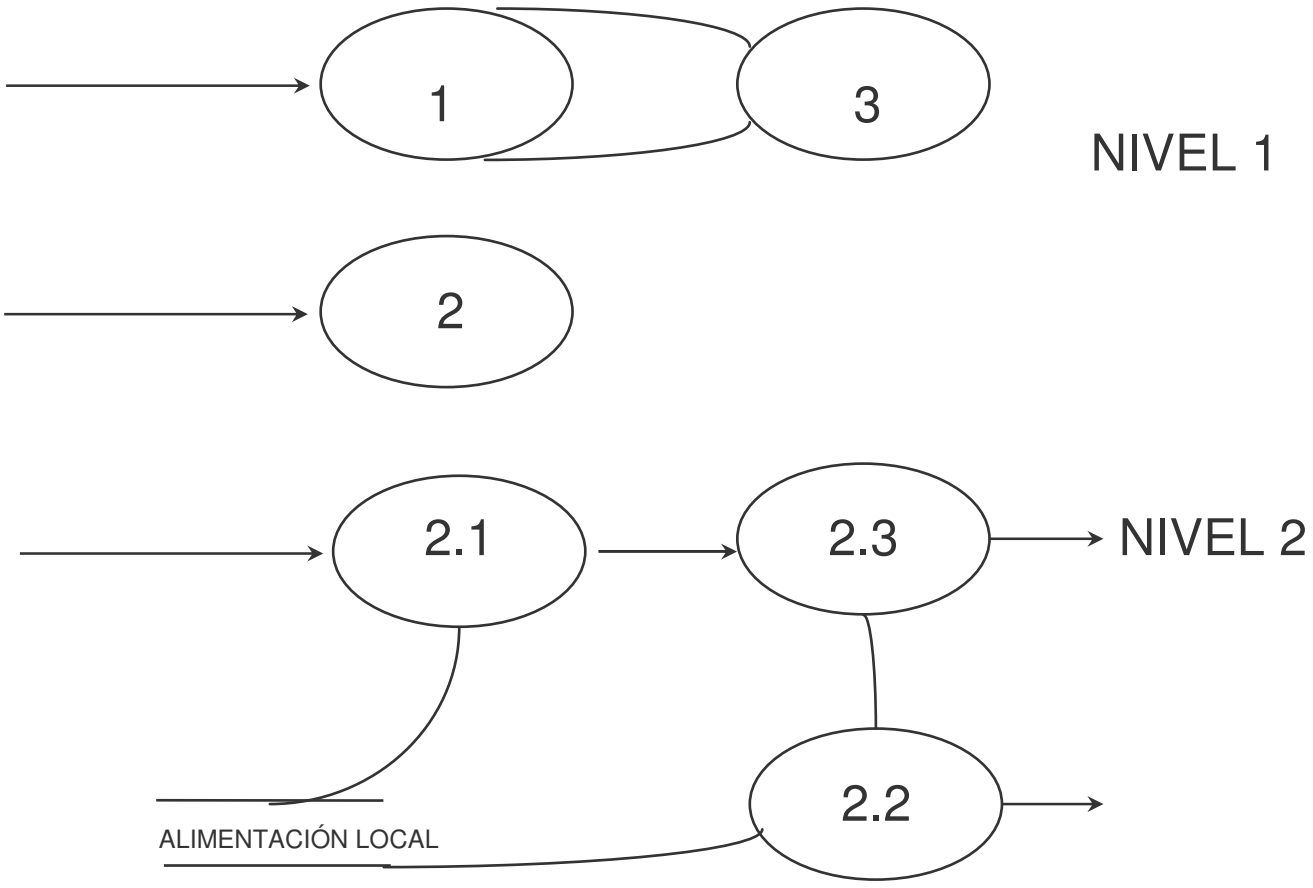
CONCEPTOS ADICIONALES:

1. BALANCEO: Los flujos de entrada y salida de un DFD, deben ser equivalentes a los flujos de E/S del proceso del nivel superior, a partir del cual se hizo la expansión.

2. CONVERSIONES NUMÉRICAS: Cada proceso recibe el nombre del proceso padre, un punto y otro número único.
El proceso del diagrama de contexto se enumera con cero.

3. ALMACENAMIENTOS LOCALES: Van apareciendo en un nivel local del DFD y sólo será utilizado por ese nivel.

4. EXPANSIÓN DE LA PARTICIÓN: Es el número de procesos en que se expande un proceso padre. Esta expansión no debe ser mayor a 7 procesos.



3.3 ESPECIFICACIÓN DE CONTROL

La especificación de control (EC) representa el comportamiento del sistema (al nivel al que se ha hecho referencia) de dos formas diferentes. La EC contiene un diagrama de

transición de estados (DTE) que es una especificación secuencial del comportamiento. También puede contener una tabla de activación de procesos (TAP) -una especificación combinatoria del comportamiento.

La especificación del proceso (EP)

Se usa para describir todos los procesos del modelo de flujo que aparecen en el nivel final de refinamiento. El contenido de la especificación de procesamiento puede incluir una narrativa textual, una descripción en lenguaje de diseño de programas (LDP) del algoritmo del proceso, ecuaciones matemáticas, tablas, diagramas o gráficos. Al proporcionar una EP que acompañe cada burbuja del modelo de flujo, el ingeniero del software crea una «mini-especificación» que sirve como primer paso para la creación de la Especificación de Requisitos del Software y constituye una guía para el diseño de la componente de programa que implementará el proceso.

Suceso de Entada	
Suceso de sensor	0 0 0 0 1 0
Indicador de parpadeo	0 0 1 1 0 0
Interruptor de comienzo parada	0 1 0 0 0 0
Estado de la acción de visualización	
Terminada	0 0 0 1 0 0
En marcha	0 0 1 0 0 1
Exceso de tiempo	0 0 0 0 0 1
Salida	
Señal de alarma	0 0 0 0 1 0
Activación de procesos	
Monitorizar y controlar el sistema	0 1 0 0 1 1
Activar/desactivar el sistema	0 1 0 0 0 0
Visualizar mensajes y estado	1 0 1 1 1 1
Interactuar con el usuario	1 0 0 1 0 1

3.4 DICCIONARIO DE DATOS

Gramática formal para describir el contenido de los objetos definidos en el análisis estructurado.

- Nombre del elemento
- Dónde y cómo se usa (qué procesos lo usan y cómo)
- Descripción del contenido
- Información adicional.

El propósito del diccionario es definir sin ambigüedad los elementos de datos.

Cuando el diccionario crece rápidamente, se recomienda el uso de una herramienta case.

Define todos los términos utilizados en el diagrama de flujo de datos . Se aplican para los siguientes elementos

- Fuentes y sumideros, en caso de que su nombre no sea claro. Ejemplo Inspección: Lugar donde se realiza una revisión exhaustiva de los documentos de un contrato.

- Procesos o transformaciones
- Almacenamientos: consiste en definir la estructura de los campos que va a contener el almacenamiento. Ejemplo: Empleados = Código del empleado + nombre+ datos personales+cargo+dependencia+fecha de ingreso

- Flujo de Datos : Consiste en definir la composición que fluye por un canal. Ejemplo: Asientos contables= Fecha+ No.documentos +cuenta+deter db/cre+valor+descripción

- Componente de Datos : Es un conjunto de datos que hace parte de otro flujo de datos Ejemplo : Datos personales =dirección+teléfono+estado civil+profesión+número de hijos

Este componente forma parte del flujo de datos :

Registro de Empleados.

- Elementos de Datos : Es un flujo de datos que no se puede descomponer en un flujo de datos de más bajo nivel. Ejemplo: Edad, cédula, estado civil etc.

EL DICCIONARIO DE DATOS SE PUEDE DEFINIR POR :

- ORDEN ALFABÉTICO
- POR TIPO DE ELEMENTO (almacenamiento, flujo, etc.)
- POR NIVELES DEL DFD

LA IMPORTANCIA DEL DICCIONARIO DE DATOS RADICA EN QUE LOS TÉRMINOS SEAN ENCONTRADOS FÁCILMENTE PARA CONSULTA.

3.5 MINIESPECIFICACIONES

Es la descripción detallada de los procesos o transformaciones de un Sistema de Información.

Aspectos a tener en cuenta :

- Definir miniespecificaciones para cada proceso

  - En la miniespecificación se describe el cálculo y la transformación que se hace sobre la información, evaluándose el contenido de esta.
  - La miniespecificación detalla qué hace el proceso independiente de cómo lo hará.

Formas de escribir las miniespecificaciones.

- Español estructurado
- Convenciones (mientras que, hasta que, solo si etc.

ACTIVIDADES

Consultar y desarrollar el siguiente taller :

Tema 01: Análisis

1. Defina en que consiste la etapa del análisis.
2. Qué actividades se desarrollan durante la etapa del análisis.
3. Qué objetivos persigue el análisis.
4. En que consiste la metodología de análisis estructurado y que técnicas se utilizan.

Tema 02: Diagramas de flujos de datos DFD

1. Defina DFD o diagrama de flujo de datos.
2. Qué notación usan los analistas para elaborar un DFD y explique cada uno. Grafique y de ejemplo.
3. Cuáles son los pasos que se siguen para elaborar un DFD.
4. Defina Diagrama de contexto.
5. En que consiste el balanceo de un DFD.
6. En que consiste el nivelamiento de un DFD.
7. Que es una función primitiva.
8. Hasta donde se debe expandir un DFD.
9. Qué errores no se deben cometer al realizar un DFD.

Tema 03: Diccionario de datos

1. Defina diccionario de datos DD.
2. A que elementos del DFD se le aplica el DD y de un ejemplo de cada uno.
3. Qué notación utilizan los analistas para elaborar el DD. Explique cada una y de ejemplo.
4. Dentro de los flujos de datos explique:
   a. Qué es un elemento de datos.
   b. Qué es un componente de datos.

5. ¿Qué es un sinónimo dentro del DD y cuando ocurre?
6. Explique usando un ejemplo como elaboraría un DD para:
    a. Fuente.
    b. Flujo de datos.
    c. Elemento de datos.
    d. Componente de datos.
    e. Almacenamiento.
    f. Proceso

Tema 04: Miniespecificaciones

1. Defina lo que es una miniespecificación.
2. Qué tipos de miniespecificaciones se usan para describir un proceso, defina cada una y acompáñela de un ejemplo.
3. En una tabla de decisión, que son las condiciones, acciones y reglas, y como es la representación gráfica de este.
4. Dos ventajas y desventajas de las tablas de decisión.

Tema 05: Modelo Entidad Relación (Diagrama de estructura de datos)

1. Defina Modelo Entidad Relación (MER).
2. Qué es una entidad de datos y como se representa.
3. Qué es una relación y como se representa.
4. Dentro de la metodología relacional de B de D defina:
    a. Tabla.
    b. Atributos.
    c. Columnas o campos.
    d. Tuplas o registros.
    e. Archivo.
    f. Clave.
    g. Clave principal.
    h. Clave foránea.
    i. Dependencia funcional.
5. .Dentro de las relaciones en que consiste:
    a. La cardinalidad.
    b. El orden.
6. Cuáles son los tipos de diagramas E-R explíquelos y de un ejemplo gráfico de cada uno.
7. Cómo se debe destruir una relación(M:M).
8. Porqué se utiliza una matriz de relaciones.
9. Defina normalización.
10. Defina los pasos para normalizar los almacenamientos.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad por parte del profesor.
- Exposición magistral de los conceptos inherentes a la ingeniería de software, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Desarrollo de plenarias y exposiciones.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en Internet

RECURSOS

- Humanos: profesor y alumnos.
- Institucionales: Salón de clases.
- Materiales: Módulo de Ingeniería del Software

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación del taller sobre la unidad
- Revisión del informe de análisis

CRITERIOS DE EVALUACIÓN

Marque F o V según el corresponda

___ El análisis de requerimientos del nuevo sistema busca conocer detalladamente los requerimientos de información del usuario

___ El análisis estructurado es el uso de los DFD, DD, DSD y las Miniespecificaciones, para generar el documento objetivo llamado especificaciones estructuradas

___ Terminal es una persona u organización que reside fuera del contexto del sistema mismo y que origina y recibe la información del sistema

___ El análisis estructurado se hace bajo descripciones narrativas.

___ BALANCEO son los flujos de entrada y salida de un DFD, que deben ser equivalentes a los flujos de E/S del proceso del nivel superior, a partir del cual se hizo la expansión

UNIDAD 2

DISEÑO DEL SISTEMA

INTRODUCCIÓN

El proceso de Diseño del Sistema de Información (DSI) es la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información.

A partir de dicha información, se generan todas las especificaciones de construcción relativas al propio sistema, así como la descripción técnica del plan de pruebas, la definición de los requisitos de implantación y el diseño de los procedimientos de migración y carga inicial, éstos últimos cuando proceda.

Llamado también diseño lógico, consiste en bosquejar los elementos constitutivos del software como interfaz con el usuario, base de datos con sus tablas, consultas y reportes, además de los datos que se ingresarán, los que serán calculados y los que se almacenarán. El diseño del sistema produce los detalles que establecen la forma en que el sistema cumplirá con los requerimientos identificados durante la fase del análisis. Es el proceso de planeación de un nuevo sistema dentro de la empresa para reemplazar o completar el existente.

JUSTIFICACIÓN

La importancia del diseño del software se puede describir con una sola palabra "calidad". El diseño es el lugar en donde se fomentará la calidad en la ingeniería del software. El diseño proporciona las representaciones del software que se pueden evaluar en cuanto a calidad. El diseño es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado. El diseño del software sirve como fundamento para todos los pasos siguientes del soporte del software y de la ingeniería del software.

Sin un diseño, corremos el riesgo de construir un sistema inestable, un sistema que fallará cuando se lleven a cabo cambios; un sistema que puede resultar difícil de comprobar; y un sistema cuya calidad no puede evaluarse hasta muy avanzado el proceso, sin tiempo suficiente y con mucho dinero gastado en él

Los ingenieros del software se encargan de desarrollar el software o supervisar el desarrollo por parte de los analistas programadores, los cuales deben hacer parte del equipo de software. Además se debe producir la documentación del mismo.

OBJETIVO GENERAL

El objetivo del diseño es producir un modelo o representación de una entidad que se será construida a posteriori, descubriendo la estructura lógica detallada que de solución al sistema planteado.

Servir de enlace y comunicación entre las especificaciones detalladas del sistema (análisis) y el ambiente con sus posibilidades específicas de programación e implementación.

OBJETIVOS ESPECÍFICOS

- Identificar los pasos que se llevan en la etapa de diseño de un sistema de información.
- Conceptualizar los principios claves del diseño.
- Definir la arquitectura de software multicapa.
- Crear los medios de comunicación entre usuarios y las máquinas.
- Convertir el diseño de datos, interfaces y arquitectura en un sistema de información operacional.
- Diseñar el modelo lógico, la interfaz gráfica de usuario y los componentes de un sistema de información
- Garantizar la adecuada calidad del sistema.
- Lograr el desarrollo de características como : Confiabilidad, eficiencia, facilidad de manejo, portabilidad, economía y consistencia.
- Desarrollar la forma como los módulos de la estructura del sistema realizaran su trabajo (Diseño Detallado).
- Definir con todo detalle el diseño de los archivos, identificando su tipo, medios de acceso, claves alternas y demás elementos.
- Definir concretamente el diseño de las entradas y salidas del sistema (documentos fuente, pantallazos y reportes).
- Definir como será la interacción con el usuario y la operación del sistema.
- Asegurar que el sistema brinde apoyo a las actividades de la empresa para las que fue desarrollado.
- Garantizar que todas la necesidades y requerimientos plasmados en el análisis sean incluidos en las etapas posteriores.

CONTENIDO

1. DEFINICIÓN

Es la transformación de las especificaciones funcionales de un sistema, en un modelo que defina con suficiente detalle "como" se va a lograr su programación e implementación física.

El diseño es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en los requisitos del cliente y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un diseño bueno. En el contexto de la ingeniería de software, el diseño se centra en cuatro áreas importantes de interés: datos, arquitectura, interfases y componentes.

2. CARACTERÍSTICAS

El diseño de corresponde a una representación abstracta del sistema, que plantea una solución para ser luego implementada.
El diseño se preocupa fundamentalmente por la forma del sistema, definiendo con todo detalle como se obtendrá el modelo planteado

Para definir la obtención , se deben desarrollar las siguientes actividades :

- Abstracción. Es la capacidad de generalizar los aspectos de un problema, de tal manera que sea posible asignar prioridades y establecer jerarquía.
- Operacionalización. Convertir la estructura jerárquica en algo realizable y funcional.
- Elaboración. Lograr el suficiente detalle y participación de lo establecido en la estructura (contrario a la abstracción).
- Verificación. Es comprobar que se cumple con lo especificado en el diseño.
- El desarrollo del diseño está sujeto a las posibilidades de programación e implementación existentes en la organización

UN BUEN DISEÑO DEBE SER:

- Completo. Abarque todos los requerimientos.
- Consistentes: Bien definidas todas su interfaces.
- Claro. Fácil de traducir a un lenguaje de programación.
- Mantenible. Fácil de modificar.
- Práctico. Fácil de realizar
- Evaluable. Fácil de revisar y mejorar sucesivamente.

3. PASOS DEL DISEÑO

El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño de software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño del software es la primera de las tres actividades técnicas -diseño, generación de código y pruebas- que se requieren para construir y verificar el software. Cada actividad transforma la información de manera que dé lugar por último a un software de computadora validado.

Cada uno de los elementos del modelo de análisis proporciona la información necesaria para crear los cuatro modelos de diseño que se requieren para una especificación completa de diseño.

Los requisitos del software, manifestados por los modelos de datos funcionales y de comportamiento, alimentan la tarea del diseño. Mediante uno de los muchos métodos de diseño la tarea se produce mediante un diseño de datos, un diseño arquitectónico, un diseño de interfaz y un diseño de componentes.

3.1 DISEÑO DE DATOS

El diseño de datos transforma el modelo del dominio de información que se crea durante el análisis en las estructuras de datos que se necesitarán para implementar el software.

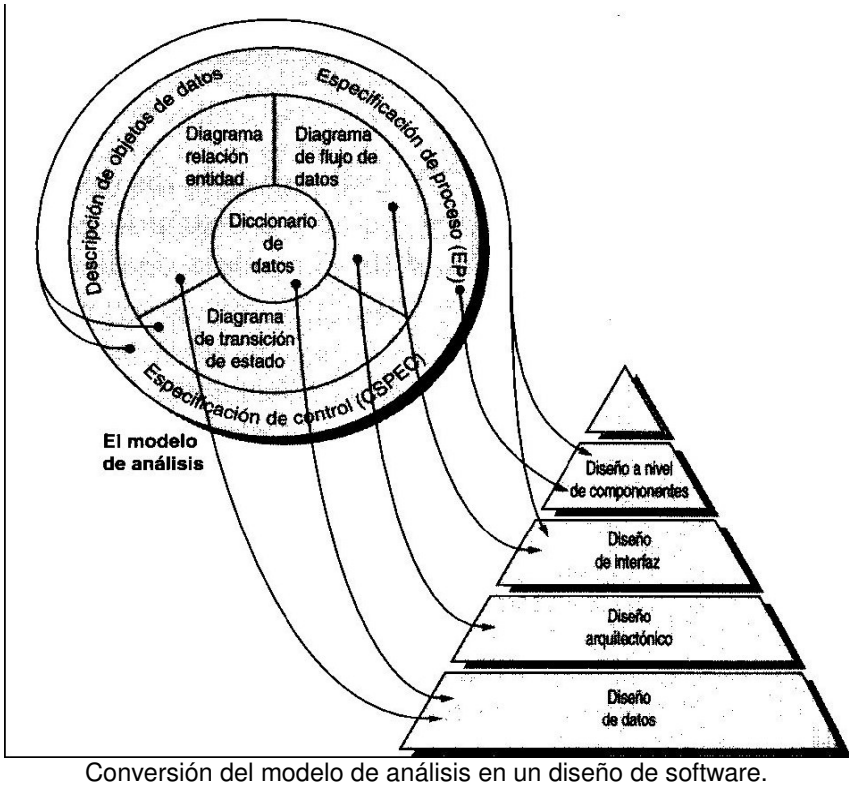
Los objetos de datos y las relaciones definidas en el diagrama relación entidad y el contenido de datos detallado que se representa en el diccionario de datos proporcionan la base de la actividad del diseño de datos. Es posible que parte del diseño de datos tenga lugar junto con el diseño de la arquitectura del software.

A medida que se van diseñando cada uno de los componentes del software, van apareciendo más detalles de diseño.

3.2 DISEÑO ARQUITECTÓNICO

El diseño arquitectónico define la relación entre los elementos estructurales principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos.

La representación del diseño arquitectónico -el marco de trabajo de un sistema basado en computadora- puede derivarse de la especificación del sistema, del modelo de análisis y de la interacción del subsistema definido dentro del modelo de análisis.



Conversión del modelo de análisis en un diseño de software.

3.3 DISEÑO DE LA INTERFAZ

El diseño de la interfaz describe la manera de comunicarse el software dentro de sí mismo, con sistemas que interoperan dentro de él y con las personas que lo utilizan. Una interfaz implica un flujo de información (por ejemplo, datos y/o control) y un tipo específico

de comportamiento. Por tanto, los diagramas de flujo de control y de datos proporcionan gran parte de la información que se requiere para el diseño de la interfaz.

3.4 DISEÑO A NIVEL DE COMPONENTES

El diseño a nivel de componentes transforma los elementos estructurales de la arquitectura del software en una descripción procedimental de los componentes del software. La información que se obtiene de EP, EC y de DTE sirve como base para el diseño de los componentes.

ACTIVIDAD…

Razone las causas por las que se descomponen los DFDs (Diagramas de Flujo de Datos). ¿Tendría sentido aplicar la descomposición en otras notaciones de análisis como DTEs (Diagramas de Transición de Estados) o DERs (Diagramas Entidad-Relación)?

Solución .: Para mejorar la legibilidad del DFD de un sistema complejo, facilitar la construcción del DFD, permitir el desarrollo en paralelo del DFD e incluso la reutilización parcial del DFD, éste se divide en diferentes niveles de abstracción. Así, el DFD de contexto da la visión más abstracta del sistema: cuales son los agentes externos que interactúan con el sistema y que datos le suministran o solicitan. El DFD de contexto se concreta en el DFD de nivel 1 y, a su vez, cada proceso complejo de dicho DFD se concreta o "explota" en otros DFDs. Así sucesivamente, hasta alcanzar procesos elementales. Por las razones antes mencionadas, tiene sentido aplicar esta estrategia, que en el diseño y la codificación suele denominarse "refinamiento progresivo", en la representación de sistemas complejos con cualquier otra notación de análisis. Por ejemplo: – Un DTE podría descomponerse "explotando" sus estados. – Un DER podría descomponerse "explotando" sus entidades.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad a cargo del profesor.
- Exposición magistral de los conceptos inherentes a la fase de diseño, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en Internet

RECURSOS

- Humanos: Profesor y alumnos
- Institucionales: Salón de clase
- Materiales: Texto guía

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación de un mapa conceptual elaborado.
- Revisión del informe de la fase de diseño de un proyecto de software

CRITERIOS DE EVALUACIÓN

- ¿Cuáles son los pasos que se siguen en la fase de diseño ?
- ¿Qué es un interfaz gráfica de usuario (GUI)?
- ¿Qué es el diseño global y el diseño detallado?
- ¿Qué es el diseño de los datos?
- ¿Qué es el diseño a nivel de componentes?

UNIDAD 3

IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

INTRODUCCIÓN

La finalidad de la etapa de implementación y prueba del sistema es poner en marcha el Sistema desarrollado y someterlo a un proceso de evaluación.

Durante la puesta en marcha, el equipo de Proyecto debe acompañar al usuario para finalmente, hacerle la entrega y que comience su Operación.

Esta es la etapa que generalmente lleva mayor tiempo en el cronograma (Creación), porque ese tiempo que se gasta en la implementación se afecta directamente sobre el tiempo invertido en depurar correctamente las etapas anteriores.

Se compone de las siguientes actividades : Codificación, Documentación interna, Pruebas

En la fase de prueba, el software se emplea de manera experimental, para asegurarse de que el software no tenga fallas, que se haya desarrollado de acuerdo a los requisitos del usuario y de acuerdo a los estándares de calidad.

JUSTIFICACIÓN

La fase de implementación es fundamental porque requiere del compromiso de la alta gerencia. La probabilidad de éxito en la implementación de cualquier proyecto está directamente relacionada con la posición organizacional del patrocinador de más alta jerarquía. Por esta razón, se recomienda siempre asegurar el compromiso abierto de la alta gerencia para apoyar la implementación de un proyecto.

Cuando los altos niveles de la organización están directamente comprometidos con ello, existen mayores probabilidades de éxito. A medida que declina el más alto nivel de apoyo a un proyecto, sus probabilidades de éxito declinan más rápido.

El compromiso de la alta gerencia significa algo más que aprobación. Supone participación en forma periódica para asegurar que los objetivos del proyecto se están alcanzando y que su filosofía e intenciones se reflejan en forma adecuada.

OBJETIVO GENERAL

Poner en marcha un sistema de información y comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica

OBJETIVOS ESPECÍFICOS

- Verificar el correcto funcionamiento de las interfaces existentes entre los distintos componentes y subsistemas, conforme a las verificaciones establecidas para el nivel de pruebas de integración
- Diferenciar cada uno de los niveles de pruebas que se le aplican a un sistema de información
- Diseñar un plan de pruebas para un sistema de información
- Aplicar las pruebas necesarias a un sistema de información

CONTENIDO

1. CONCEPTOS GENERALES

Los errores descubiertos en la etapa de implementación son costosos:

En el Análisis implica reinicio del proceso, atraso del cronograma considerable, reunión adicional con el cliente, inseguridad con el proyecto

En el Diseño implica inicio del proceso de una etapa media, atraso en el cronograma (a veces se incluye análisis de nuevo), Capacitación técnica adicional

La implementación requiere:

- Refinamiento
- Conocimiento técnico
- Habilidades de programación
- Manejo de herramientas
- Tecnología (Hardware y Software)
- Estándares
- Disciplina
- Depuración y manejo de errores
- Conocimiento de la expectativa del cliente

La Codificación requiere

- Refinar el nivel de abstracción al máximo.
- Traducir el diseño al lenguaje de programación.
- Creación de unidades que servirán para ir mando el producto.

Y paralelo a la codificación viene la documentación interna que es dotar al código fuente de legibilidad para facilitar el mantenimiento y esto necesita escoger el lenguaje de programación, y tener en cuenta aspectos del lenguaje que afectan la confiabilidad y mantenimiento del sistema, como también el estilo de programación.

Aspectos generales del Lenguaje de programación

- Lenguaje de alto nivel.
- Que permita el uso de nombres significativos.
- Manejo fácil de módulos y funciones.
- Estructuras de control y decisión adecuadas.
- Uso de declaraciones de variables, constantes y otros tipos de datos.
- Manipulación de errores.

Criterios pragmáticos

- Tamaño del proyecto.
- Conocimiento del lenguaje por parte del grupo.
- Requerimiento uso lenguaje específico.
- Disponibilidad de compiladores para el sistema computador/sistema operativo.
- Software portable.
- Tipo de aplicación a desarrollar.
- Criterios de gestión
- Proyección de vida del lenguaje
- Soporte técnico
- Costo de licencias de operación
- Características de robustez
- Restricciones legales

Ciertas estructuras o instrucciones facilitan la realización de programas confiables y mantenibles: Módulos, Declaraciones de variables, Tipos de datos definidos por el usuario, Estructuras de datos complejas (registros, punteros), Alcance de variables, Manejo de excepciones, etc.

Estilos de codificación :

Metodología (Top-down y bottom-up)

Prácticas a favor de: buena documentación, utilizar nombres significativos y descriptivos de variables y constantes, adecuado uso de comentarios (documentación interna del código), formato de edición o estructura (usar sangrado y si el lenguaje lo permite, escribir en mayúsculas palabras reservadas del lenguaje para destacarlas de las del usuario).

Debe buscarse ante todo **SIMPLICIDAD** y **LEGIBILIDAD**. Para ello debe tenerse en cuenta que:

Entre más corto sea el código este será más simple y fácil de entender.

Es mejor minimizar el número de decisiones en un módulo para reducir su complejidad.

La creación de procedimientos y funciones que describan en lenguaje semi-natural el proceso.

Factores que afectan la calidad del código fuente :

- Uso de técnicas de codificación estructurada.
- Buen estilo de codificación.
- Buena selección de estructuras de datos locales.
- Comentarios internos bien escritos.
- Formato de código fuente consistente y legible.
- Utilización de identificadores significativos.

Dentro de las técnicas de codificación estructurada hay estándares para la estructura del programa:

- No usar instrucciones de flujo aleatorio (GO TO).
- Usar únicamente flujo de control descendente.
- Producir constructores de una entrada y una salida
-

Consideraciones prácticas para escoger el lenguaje de programación

- Requerimientos de quien contrató el desarrollo del software.
- Conocimiento del Lenguaje por parte de los programadores.
- Lenguajes usados en proyectos anteriores o concurrentes.
- Disponibilidad y calidad del compilador del Lenguaje.
- Disponibilidad de herramientas de soporte para desarrollo de Software.
- Portabilidad.

En tiempo de implementación vienen las pruebas que son una fase fundamental en el proceso de desarrollo de software.

2. DEFINICIÓN DE PRUEBAS

Las pruebas se definen como el proceso de ejercitar o evaluar el sistema, por medios manuales o automáticos, para verificar que satisface los requerimientos o, para identificar diferencias entre los resultados esperados y los que produce el sistema.

La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

Una prueba tiene éxito si descubre un error no detectado hasta entonces.

También podemos decir que es el proceso mediante el cual se establecen la existencia de errores. La prueba puede demostrar existencia de errores, NO LA AUSENCIA DE ELLOS

Recordando el objetivo de las pruebas, se diseñan pruebas que tengan la mayor probabilidad de encontrar el mayor número de no conformidades con la mínima cantidad de esfuerzo y tiempo, para lo cual existen fundamentalmente dos enfoques, que al combinarlos permite lograr mayor efectividad:

- Pruebas de Caja Negra o Enfoque Funcional, hace referencia a pruebas que se llevan a cabo a través de la interfaz gráfica del software (GUI). O sea, pretenden

demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

- Pruebas de Caja Blanca, son aquellas pruebas que se basan en los caminos lógicos del software, la estructura interna y la implementación del software en pruebas, proponiendo casos que ejerciten conjuntos específicos de condiciones y/o ciclos.

Para cada enfoque, existen diversas técnicas que enriquecen las posibilidades de los probadores "para hacer más con menos".

En esta actividad se realiza la especificación de detalle del plan de pruebas del sistema de información para cada uno de los niveles de prueba establecidos en el proceso Análisis:

- Pruebas unitarias.
- Pruebas de integración.
- Pruebas del sistema.

Para ello se toma como referencia el plan de pruebas, que recoge los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para planificar paso a paso las actividades de prueba. También puede ser una referencia el plan de integración del sistema de información propuesto, opcionalmente, en la tarea de definición de Componentes y Subsistemas de Construcción.

El catálogo de requisitos, el catálogo de excepciones y el diseño detallado del sistema de información, permiten la definición de las verificaciones que deben realizarse en cada nivel de prueba para comprobar que el sistema responde a los requisitos planteados. La asociación de las distintas verificaciones o componentes, grupos de componentes y subsistemas, o al sistema de información completo, determina las distintas verificaciones de cada nivel de prueba establecido.

Las pruebas unitarias comprenden las verificaciones asociadas a cada componente del sistema de información. Su realización tiene como objetivo verificar la funcionalidad y estructura de cada componente individual.

Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes, generalmente reflejados en la definición de subsistemas de construcción o en el plan de integración del sistema de información. Tienen por objetivo verificar el correcto ensamblaje entre los distintos componentes.

Las pruebas del sistema, de implantación y de aceptación corresponden a verificaciones asociadas al sistema de información, y reflejan distintos propósitos en cada tipo de prueba:

– Las pruebas del sistema son pruebas de integración del sistema de información completo. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen.

– Las pruebas de implantación incluyen las verificaciones necesarias para asegurar que el sistema funcionará correctamente en el entorno de operación al responder satisfactoriamente a los requisitos de rendimiento, seguridad y operación, y coexistencia con el resto de los sistemas de la instalación, y conseguir la aceptación del sistema por parte del usuario de operación.

– Las pruebas de aceptación van dirigidas a validar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catálogo de requisitos y en los criterios de aceptación del sistema de información, y conseguir la aceptación final del sistema por parte del usuario.

Las pruebas unitarias, de integración y del sistema se llevan a cabo en el proceso Construcción del Sistema de Información (CSI), mientras que las pruebas de implantación y aceptación se realizan en el proceso Implantación y Aceptación del Sistema (IAS).

3. PRINCIPIOS DE LAS PRUEBAS

- A las pruebas se les debe poder hacer un seguimiento hasta los requisitos del cliente.
- Las pruebas deben planificarse mucho antes de que empiecen.
- Las pruebas deben empezar por lo pequeño y progresar hacia lo grande.
- Algunas veces no son posibles pruebas exhaustivas.
- Para ser más efectivas deben conducirse por un equipo independiente

4. RECOMENDACIONES

- Use siempre datos de entrada bien definidos para los que se conozcan los resultados correctos que deben obtenerse.

- Detecte primero los defectos obvios (usando datos de prueba muy simples) y luego sí realice pruebas más complejas.

- Cuando modifique algo mientras prueba realice un solo cambio cada vez y utilice los mismos datos con los que detectó el defecto.

- Pruebe el programa para verificar si detecta entradas incorrectas

5. ETAPAS INVOLUCRADAS EN TODAS LAS PRUEBAS

- Seleccionar qué es lo que debe medir la prueba, es decir, cuál es su objetivo, para qué exactamente se hace la prueba.

- Decidir cómo se va a realizar la prueba, es decir, qué clase de prueba se va a utilizar para medir la calidad escogida y qué clase de elementos de prueba se deben usar.

- Desarrollar los casos de prueba. Un caso de prueba es un conjunto de datos o situaciones de prueba que se utilizarán para ejecutar la unidad que se prueba o para revelar algo sobre el atributo de calidad que se está midiendo.

- Determinar cuáles deberían ser los resultados esperados o correctos de los casos de prueba y crear el documento con los casos y sus resultados esperados, denominado oráculo de prueba, antes de realizar la prueba.

- Ejecutar los casos de prueba.

- Comparar los resultados de la prueba con los resultados esperados. Cualquier discrepancia entre ellos significa un error. Típicamente el error está en el sistema o unidad probada, pero también puede ser generado por algún aspecto del mismo proceso de prueba o en el oráculo de prueba.

- Deben ser completos (mayor probabilidad de encontrar errores).

- Comprobación

6. ESTRATEGIAS DE PRUEBAS

- Casos de prueba + pasos de la prueba
  - Planificación de las pruebas
  - Diseño de los casos de prueba
  - Ejecución
  - Agrupación y evaluación de los datos resultantes.

- La prueba comienza a nivel de módulo y trabaja hacia fuera (integración del sistema).
- Según el momento puede aplicarse una técnica diferente de prueba

- El responsable de la prueba debe ser en lo posible un grupo independiente al desarrollador.

- Debe incluirse la depuración además de la prueba

7. NIVELES DE PRUEBAS

7.1 PRUEBA DE UNIDAD

En tiempo de implementación vienen las pruebas de unidad de función específica donde a medida que se construye una unidad se va probando para descartar errores de nivel bajo a medida que se sube en el mapa de construcción. Se debe construir una prueba, por cada diferente dirección de flujo dentro de la unidad.

Se debe hacer la creación de listas de chequeo que reúnan las características relevantes a probar para un tipo de unidad determinado.

Los ítems como estándares de presentación y de programación pueden incluirse.

Debe manejar utilitarios para agilizar los procesos de prueba de unidad.

Se deben probar en orden ascendente para garantiza mayor eficiencia en pruebas de integración con unidades mayores.

En éste proceso de verificación sobre la menor unidad de software: módulo, se repasan los caminos de control importantes y siempre está orientada a caja blanca. Incluye pruebas sobre

- Interfaz del módulo
- Estructuras de datos locales
- Condiciones de límite
- Caminos independientes
- Caminos de manejo de errores

Se considera adjunto a la codificación.

Se desarrollan casos de prueba con resultados esperados y cuando un módulo no es independiente, debe probarse con otro modulo que lo controle o con módulos subordinados que verifiquen la interfaz entre ambos. Las pruebas de unidad se simplifican si el módulo es altamente cohesivo.

7.2 PRUEBA DE INTEGRACIÓN

El objetivo de esta tarea es verificar el correcto funcionamiento de las interfaces existentes entre los distintos componentes y subsistemas, conforme a las verificaciones establecidas para el nivel de pruebas de integración.

Para cada verificación establecida, se realizan las pruebas con los casos de pruebas asociados, efectuando el correspondiente análisis e informe de los resultados de cada verificación, y generando un registro conforme a los criterios establecidos en el plan de pruebas
La prueba la interacción entre módulos, es una técnica sistemática para construir la estructura de un programa al mismo tiempo en que se buscan errores de interacción.

Método BIG-BANG: Todos los módulos se codifican por separado y se integran todos al tiempo, sin hacer prueba unitaria, causando efectos desastrosos.

Métodos de desarrollo incremental: Algunas partes del sistema se codifican, se hacen pruebas unitarias y se integran antes que otras porciones del sistema. El sistema completo se crea en pasos incrementales y no con un solo esfuerzo monumental.

- El Top-Down (descendente).
- El Bottom-Up (ascendente).
- El threads (hilos).

7.3 PRUEBA DE VALIDACIÓN

Se realiza una vez ya se ha hecho la prueba del software como paquete de módulos que interactúan.

Funcionamiento de acuerdo a las expectativas del cliente (expresadas en el documento de requerimientos).

Pruebas de caja negra que demuestren conformidad con los requisitos (funcionales, de rendimiento, documentación).

7.4 PRUEBA DE SISTEMA

El objetivo de las pruebas del sistema es comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

En la realización de estas pruebas es importante comprobar la cobertura de los requisitos, dado que su incumplimiento puede comprometer la aceptación del sistema por el equipo de operación responsable de realizar las pruebas de implantación del sistema, que se llevarán a cabo en el proceso Implantación y Aceptación del Sistema.

Este tipo de pruebas están constituida por una serie de pruebas diferentes cuyo propósito es ejercitar las operaciones soportadas en el sistema.

Se trabajan para verificar que se ha integrado adecuadamente todos los elementos del sistema.

- Pruebas de recuperación
- Pruebas de seguridad
- Prueba de resistencia
- Prueba de rendimiento

8. PLAN DE PRUEBAS

Funcionalidad: Descripción de funcionalidad a probar (incluyendo los módulos necesarios)

Caso de prueba: Caracterización de los datos de entrada, identifica cada caso de prueba.
Ambiente: Prerrequisitos de información o de procesos para la prueba.
Instrucciones: Si se requiere (según complejidad)
Datos esperados.

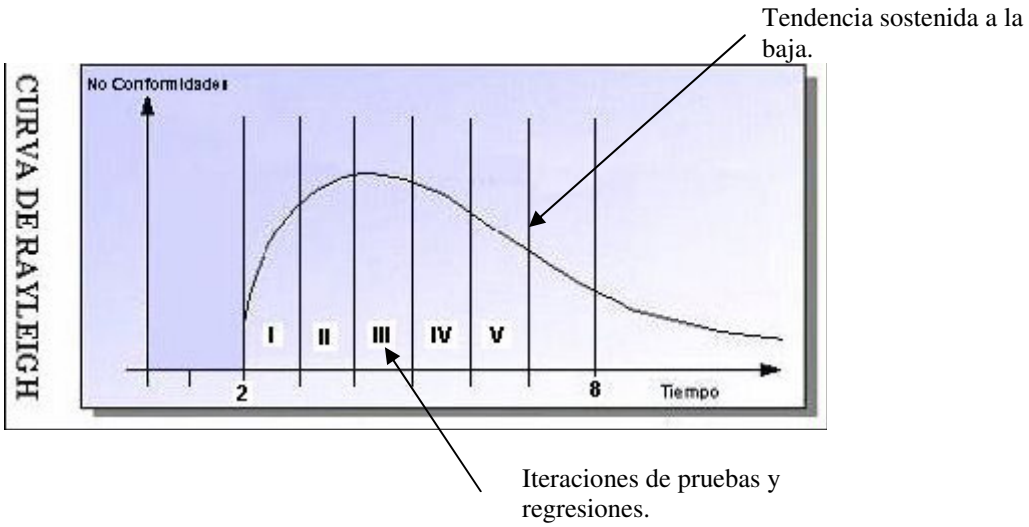
Registros de resultados de la prueba.

En la ejecución de ésta etapa, los responsables de la evaluación preparan las condiciones del ambiente y los datos que se usarán para ejecutar las pruebas hasta obtener un 'ambiente de pruebas controlado' sobre el cual se ejecutan los requerimientos de prueba.

Con base en la experiencia de los proyectos desarrollados, se propone que la etapa de ejecución de cada proceso de pruebas sea dividida en al menos 5 iteraciones de pruebas con sus respectivas regresiones, es decir, 5 ciclos de ejecución de los requerimientos de prueba o subconjuntos de requerimientos agrupados por algún criterio válido como por ejemplo la funcionalidad, naturaleza, grado de estabilidad.

Los resultados obtenidos en cada iteración de pruebas, en términos de horas hombre (HH) invertidas y cantidad de no conformidades por tipo son registrados en el documento entregable de pruebas, donde adicionalmente se presentan hallazgos generales relevantes y recomendaciones.

Es responsabilidad de los probadores ejecutar la mayor cantidad posible de iteraciones y regresiones de prueba hasta poder demostrar cuantitativamente que la cantidad de hallazgos en cada iteración presentan una tendencia sostenida a la baja que permita concluir que se está logrando la estabilidad funcional del producto de software.



ACTIVIDAD…

- Consultar en Internet cuál es la metodología para diseñar un plan de pruebas.
- Visitar empresas desarrolladoras de software y consultar como hacen su plan de pruebas.
- Diseñar un plan de pruebas para su proyecto de desarrollo de software.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad a cargo del profesor.
- Exposición magistral de los conceptos inherentes a la fase de pruebas, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en Internet

RECURSOS

- Humanos: Profesor y alumnos
- Institucionales: Salón de clase
- Materiales: Texto guía

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Elaborar un mapa conceptual sobre pruebas de software
- Evaluar el diseño de un plan de pruebas

CRITERIOS DE EVALUACIÓN

1. Cuales de los siguientes asertos son correctos
   a. Las pruebas son el único método para poder verificar y validar el software
   b. Verificar y validar son dos conceptos equivalentes
   c. Verificar una aplicación consiste en comprobar si satisface los requisitos marcados por el usuario
   d. Validar una aplicación consiste en comprobar si satisface los requisitos marcados por el usuario

2. Fallos, defectos y errores

   a. Un fallo suele ser la consecuencia de un defecto
   b. Un defecto suele ser el origen de un error
   c. Un fallo puede ser el origen de un error
   d. Un error suele ser la consecuencia de un fallo

3 Cuales de las siguientes afirmaciones son correctas

   a. Las pruebas requieren del orden del 40% del esfuerzo de desarrollo
   b. Las pruebas se centran en chequear si el software no hace lo que debe
   c. El mejor probador es el propio programador que ha desarrollado el programa, ya que lo conoce “a conciencia”
   d. Cuantos menos defectos se encuentren por unidad de tiempo, mayor es la eficiencia de las pruebas

4 Cuales de las siguientes afirmaciones son correctas

   a. El enfoque de caja blanca también se denomina enfoque funcional.
   b. El enfoque de caja negra consiste en probar todas las posibles funciones de entrada-salida del programa.
   c. Existen los siguientes tipos o enfoques de pruebas: caja blanca, caja negra, aleatorias y deterministas.
   d. Ninguna de las anteriores.

UNIDAD 4

IMPLANTACIÓN Y EVALUACIÓN

INTRODUCCIÓN

Es el proceso de instalar el software en el equipo, entrenar a los usuarios y construir los archivos y bases de datos necesarios para utilizarlo. Una vez hecho esto, se lleva a cabo la evaluación para identificar puntos débiles y fuertes y poder evaluar la operatividad del software, el impacto en la organización y la opinión de los administradores.

JUSTIFICACIÓN

Cada organización es única, tiene su propia combinación exclusiva de hombres, recursos económicos, máquinas, materiales y métodos. No solamente son diferentes los componentes individuales de la organización, sino también el grado de evolución de su sistema de información para la administración. Esta singularidad hace necesario que cada organización desarrolle sus propias especificaciones de su sistema de información para la administración, mediante una evaluación sistemática de su propio ambiente externo e interno y de su punto de vista, de acuerdo con sus propias necesidades únicas.

Por eso esta fase de implantación y evaluación es fundamental porque garantiza el éxito de un proyecto del desarrollo de software.

OBJETIVO GENERAL

Dar una visión global del proceso de implantación de un sistema de información, prestando especial atención a las repercusiones que el desarrollo de cada una de las fases tiene sobre la fase de post – implantación del sistema.

Mostar el impacto que tiene la implantación y evaluación de un sistema de información en las estructuras de la empresa y las acciones que se han de emprender para adaptar las estructuras al nuevo sistema de modo que la fase de post - implantación cumpla con los objetivos que se la definan.

OBJETIVOS ESPECÍFICOS

Definición de los objetivos de la fase de formación de los usuarios.

Detallar las acciones que se han de seguir para capacitar correctamente a los usuarios.

Resaltar la importancia que una correcta formación de los usuarios tiene en el desarrollo de la fase de implantación.

Detallar las diferentes opciones de conversión y arranque de un sistema de información.

Resaltar las repercusiones que el arranque del sistema tiene en el posterior funcionamiento del sistema.

Definir las opciones existentes actualmente para el mantenimiento de los sistemas de información tras el arranque del sistema.

Se presta atención a las implicaciones que cada una de las opciones tiene en la estructura de los departamentos de informática de las empresas, así como en las ventajas y desventajas que tienen de cara al futuro las distintas opciones.

CONTENIDO

1. CONCEPTOS GENERALES

La parte más compleja de un proyecto de implantación de software es aquella relacionada con la gente que lo va a utilizar; con sus necesidades, sus expectativas... y su dolor. Si cuando se está implantando, los costos o el dolor actual superan los beneficios esperados, hay que revisar el proyecto.

La tendencia de la tecnología de información orientada a los negocios es minimizar la complejidad tecnológica de cara al usuario, y dejarla, tipo caja negra, para el uso exclusivo de los expertos que desarrollaron el paquete. De esta forma se evitan las complicaciones de tener que intervenirlo/modificarlo distorsionando su integridad.

Los enfoques tradicionales de implantación de paquetes (o de cualquier tecnología de aplicación de software) tienden a trivializar la participación, durante la etapa de proyecto, de quien finalmente será usuario de la tecnología. Y esto es así, pues fueron basados en la premisa de que el paquete debía adaptarse a la empresa, colocando mayor énfasis en lo tecnológico e ignorando la mejora de los procesos y funcionalidades del negocio, beneficiario final del proyecto.

Los nuevos enfoques asumen que la empresa debe adaptarse al paquete, pues se supone que la tecnología está suficientemente depurada e incluye plantillas con procesos que ya incorporan mejores prácticas de negocio; de alguna manera se está comprando reingeniería, permitiendo que el mayor peso se coloque en el proceso de negocio y en la funcionalidad.

Por eso, uno de los errores más frecuentes que se sigue cometiendo es considerar una implantación como netamente tecnológica, cuando en realidad el contenido de procesos y relaciones humanas es sustancialmente más fuerte y de mayor riesgo/impacto. Ya sea que la reingeniería venga con el paquete, ya sea que haya que desarrollarla.

La empresa que no considere que el verdadero factor crítico de éxito en una implantación está en la gente, está condenada a que los proyectos no le salgan bien, ni en tiempo ni en costo; y ni aun en calidad.

En todo caso, invito a que se haga una encuesta a nivel de usuarios de paquetes implantados en los últimos tres o cuatro años, y estoy seguro de que los resultados serán de mayor insatisfacción en aquellos casos en los que se dejo la parte humana (gerencia del cambio) para después, o no se considero en ningún momento.

En cuanto a los niveles de reingeniería que traen incorporados los paquetes, cabe mencionar que para nuestra región latinoamericana, hay que hacer trabajos de adaptación importantes, pues no toda empresa está bajo ISO-9000, ni cuenta con los niveles de tecnología y disciplina que se supone deben tener; además de no contar (a veces por no poder pagarlo) con el personal certificado para consolidar el éxito de la implantación.

Uno puede separar los componentes de un paquete, de cara a quien lo va a utilizar, en:

-El componente de datos.

-El de la aplicación en sí mismo, y

-El de las pantallas que se ven (front end).

Y en este último, que es el que utiliza la gente, debe estar el gran esfuerzo al momento de implantar, pues en la parte tecnológica de la aplicación en sí misma, usualmente hay tal nivel de parametrización de trabajo previo del proveedor del software, que al momento de implantar solo se requieren comandos para adaptar el ambiente de procesamiento y comunicaciones; el "solo se requieren" del párrafo anterior no pretende minimizar de ninguna manera la importancia de este componente.

El primer elemento, el de los datos, si bien no tiene complejidad de implantación, se convierte en una actividad importante, por aquello de la confiabilidad y representatividad de la información histórica, la cual repercutirá en la calidad de la información para tomar decisiones, fin último de todo el esfuerzo. Debería preverse desde el principio del proyecto de tal manera que nunca pase por la ruta crítica de la implantación.

Desde siempre se ha planteado la dicotomía entre implantación gradual y aquella llamada "big bang" o implantación rápida. Ambas tienen sus ventajas y desventajas, las cuales surgirán del ambiente propio de cada empresa. Se puede comparar con una cirugía masiva para un ser humano. Y la pregunta es si ese cuerpo en particular será capaz de resistir ese tipo de intervención. O si por el contrario debe hacerse en forma escalonada, de manera tal que en un cierto horizonte haya un resultado exitoso.

En cuanto a la organización del proyecto, es recomendable que el líder de la implantación sea alguien proveniente de las áreas de negocio y con un alto nivel de prestigio y credibilidad. Una manera de darse cuenta de si la persona elegida es la correcta, es el nivel de quejas de sus jefes por haberlo asignado al proyecto siendo una persona tan valiosa. Si no hay quejas, algo no está bien.

Hay que evitar las implantaciones de laboratorio; esas en las cuales la gente de sistemas se encierra con un par de usuarios, lo hacen todo, y después lo muestran ya terminado. Hay que hacer participar, en forma comprometida, desde el principio a quienes usarán el sistema. Si en una empresa, es el departamento de sistemas o informática quien está liderando la implantación, recomiendo revisar el proyecto y asegurarse un rol preponderante para la gente del negocio.

La experiencia me ha enseñado que no hay dos empresas iguales, ni hay dos zonas geográficas iguales dentro de una misma empresa, de cara a una implantación de software. Y no hay tampoco dos departamentos iguales. Por consiguiente, los enfoques de implantación tampoco pueden ser iguales. La linealidad del enfoque de implantación, podrá parecer mas barata al principio, pero resultará mas cara al final. Hay que mantener un balance adecuado para que la inversión agregue valor en vez de problemas y frustraciones.

En términos generales, recomiendo tener en cuenta los siguientes aspectos:

Construir un plan de trabajo que permita "visualizar" el producto final. El nivel de detalle y la coherencia y secuencia de las actividades deben ser "fáciles" de interpretar por el lector del plan. Si uno debe explicar que "cuando digo esto, quiero decir lo otro", seguramente hay un problema de "visualización" en el plan.

No durar más de cinco (5) meses implantando. Después de ese tiempo, comienza un desgaste tal, que debe ser identificado, compensado e incluido formalmente en el plan. Una manera de manejar proyectos largos/grandes, es separarlos en proyectos menores que comprendan áreas de impacto y resultados claramente diferenciadas.

Mantener la relación costo actual/beneficio futuro en un punto controlable que evite que el proyecto se convierta en un problema.

Desarrollar un análisis de riesgo que identifique las fuentes, los conceptos, sus impactos y las acciones necesarias para compensarlo. El riesgo tecnológico debe ser una consideración importante.

Uno debe poder pararse al final del proyecto, mirar para atrás en el tiempo y poder decir que fue exitoso. La otra opción, parados en el mismo sitio, es dar explicaciones de por qué aún no está terminado, por qué nadie lo usa, por qué todos lo critican, por qué salio mal, etcétera, etcétera.

Afortunadamente, implantar un paquete ya no es lo que solía ser; pues ahora lo verdaderamente importante es que quien lo vaya a utilizar participe activamente y se sienta "dueño" del proyecto.

2. LA IMPLANTACIÓN

Es el proceso de verificar e instalar nuevo equipo, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarla.

Cada estrategia de implantación tiene sus méritos de acuerdo con la situación que se considere dentro de la empresa. Sin importar cuál sea la estrategia utilizada, los encargados de desarrollar el sistema procuran que el uso inicial del sistema se encuentre libre de problemas.

Los sistemas de información deben mantenerse siempre al día, la implantación es un proceso de constante evolución.

3. LA EVALUACIÓN

La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes. La evaluación ocurre a lo largo de cualquiera de las siguientes dimensiones:

- Evaluación operacional

Valoración de la forma en que funciona el sistema, incluyendo su facilidad de uso, tiempo de respuesta, lo adecuado de los formatos de información, confiabilidad global y nivel de utilización.

- Impacto organizacional

Identificación y medición de los beneficios para la organización en áreas como finanzas (costos, ingresos y ganancias), eficiencia operacional e impacto competitivo.

- Opinión de los administradores

Evaluación de las actitudes de directivos y administradores dentro de la organización así como de los usuarios finales.

- Desempeño del desarrollo

La evaluación del proceso de desarrollo de acuerdo con criterios tales como tiempo y esfuerzo de desarrollo, concuerdan con presupuestos y estándares, y otros criterios de administración de proyectos.

Cuando la evaluación de sistema se conduce en forma adecuada proporciona mucha información que puede ayudar a mejorar la efectividad de los esfuerzos desarrollo de aplicaciones subsecuentes

3.1 ¿ESTAMOS CONSTRUYENDO SOFTWARE SIN DEFECTOS?: ESTADO ACTUAL DE LA PRÁCTICA

Todos, alguna vez, hemos sufrido algún error informático, ya sea una factura indebidamente cargada o la destrucción del trabajo de todo un día, por culpa de un fallo misterioso en el software. Tales problemas nacen de la complejidad del software. La extrema dificultad para construir sistemas software multiplica la probabilidad de que persistan errores aún después de haberse finalizado y entregado el sistema, manifestándose cuando éste es utilizado por el cliente.

La construcción de un sistema software tiene como objetivo satisfacer una necesidad planteada por un cliente. ¿Cómo puede saberse si el producto construido se corresponde exactamente con lo que el cliente deseaba? y ¿Cómo se puede estar seguro de que el producto que ha construido va a funcionar correctamente?

Desgraciadamente, nuestra capacidad para medir la fiabilidad del software es muy inferior a lo que sería necesario. Sería deseable que los informáticos pudieran demostrar matemáticamente la corrección de sus programas, al estilo de los otros ingenieros. Los otros ingenieros recurren a análisis matemáticos para predecir cuál será el comportamiento de sus creaciones en el mundo real. Esa predicción permite descubrir defectos antes de que el producto esté operativo. Por desdicha, las matemáticas tradicionales, aptas para la descripción de sistemas físicos (los tipos de sistemas tratados por las otras ingenierías), no son aplicables al universo sintético binario de un programa de ordenador.

Es la matemática discreta, una especialidad mucho menos madura, y casi no estudiada hasta la aparición de las computadoras, la que gobierna el campo de los sistemas software.

Dada la imposibilidad de aplicar métodos matemáticos rigurosos, el modo que tienen los informáticos para respaldar la confianza de los programas es la verificación empírica. La fiabilidad de los programas irá creciendo a lo largo de este proceso. Se hacen funcionar los programas, observando directamente su comportamiento y depurándolos cada vez que aparece una deficiencia una vez el sistema a construir ha sido terminado. Sin embargo, este modo de actuar no proporciona una solución definitiva debida, principalmente, a dos razones:

a. Si descubrimos en el código errores muy graves que afectan a productos anteriores (requisitos, diseño,…) debemos volver atrás en el desarrollo. Sin embargo, estamos ya al final del proyecto (en la etapa de codificación), ya se ha gastado casi la totalidad del tiempo y del presupuesto. ¿Qué hacer? ¿Entregamos tarde el sistema y repetimos el desarrollo? ¿Le pedimos al cliente un aumento del presupuesto?

b. Por otra parte, la comprobación que empírica no sirve para garantizar que no hay errores en el software, puesto que ello depende, por un lado, de la porción del programa que se esté ejecutando en el momento de esta comprobación, y por otro, de las entradas que se le hayan proporcionado al código. Por lo tanto, pueden existir errores en otras partes del programa que no se ejecuten en ese momento o con otras entradas que no se hayan usado en la prueba.

Por lo tanto, lo recomendable es que producto software vaya siendo evaluado a medida que se va construyendo. Como veremos posteriormente, se hace necesario llevar cabo, en paralelo al proceso de desarrollo, un proceso de evaluación o comprobación de los distintos productos o modelos que se van generando, en el que participarán desarrolladores y clientes.

No obstante, la calidad que se puede obtener mediante este procedimiento artesanal es bastante baja.

De ahí que, a pesar de haber sido ensayados rigurosa y sistemáticamente, la mayoría de los programas grandes contengan todavía defectos cuando son entregados. Ello se debe

a la complejidad del software. Un programa de apenas unos centenares de líneas de código puede contener decenas de decisiones, lo que permite millares de rutas de ejecución alternativas, resultando materialmente imposible el ensayo exhaustivo de todas las posibles rutas alternativas. Para alcanzar una confianza de no más de 10 fallos por hora tendría que ejecutarse un programa durante muchísimos múltiplos de 10 horas, esto es, durante muchos múltiplos de 100.000 años.

Así pues, la ambición de conseguir programas perfectos sigue siendo una cima inaccesible. Existe, hoy por hoy, la imposibilidad práctica de conseguir software totalmente libre de defectos y, debemos, por tanto, aceptar las actuales limitaciones que padece la construcción de sistemas software. De hecho, algunos autores sugieren que, dada la entidad no física del software, los defectos en los programas son inherentes a su naturaleza.

3.2 CONTROL DE CALIDAD DEL SOFTWARE

El interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades.

Como primera aproximación es importante diferenciar entre la calidad del PRODUCTO software y la calidad del PROCESO de desarrollo. Las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

También es importante destacar que la calidad de un producto software debe ser considerada en todos sus estados de evolución a medida que avanza el desarrollo de acuerdo al ciclo de vida seleccionado para su construcción (especificaciones, diseño, código, etc.). No basta con tener en cuenta la calidad del producto una vez finalizado, cuando los problemas de mala calidad ya no tienen solución o la solución es muy costosa.

Los principales problemas a los que se enfrenta el desarrollo de software a la hora de tratar la calidad de un producto software son la definición de calidad y su comprobación:

Con respecto a la definición de la calidad del software: ¿Es realmente posible encontrar un conjunto de propiedades en un producto software que nos den una indicación de su calidad? Para dar respuesta a estas preguntas aparecen los Modelos de Calidad. En los Modelos de Calidad, la misma se define de forma jerárquica. Resuelven la complejidad mediante la descomposición. La calidad es un concepto que se deriva de un conjunto de sub-conceptos.

En el caso de la calidad del software, el término es difícil de definir. Con el fin de concretizar a qué nos referimos con calidad de un sistema software, se subdivide en atributos:

- Funcionalidad – Habilidad del software para realizar el trabajo deseado.
- Fiabilidad – Habilidad del software para mantenerse operativo (funcionando).
- Eficiencia – Habilidad del software para responder a una petición de usuario con la velocidad apropiada.
- Usabilidad – Habilidad del software para satisfacer al usuario.
- Mantenibilidad – Habilidad del software para poder realizar cambios en él fácilmente y con una adecuada proporción cambio/costo.
- Portabilidad – Habilidad del software para operar en diferentes entornos informáticos.

En términos generales, se pueden distinguir dos tipos de evaluaciones durante el proceso de desarrollo: Verificaciones y Validaciones, éstas se definen como:

- Verificación: Proceso de determinar si los productos de una cierta fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior.
- Validación: Proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de las necesidades del cliente.

3.3 TÉCNICAS DE EVALUACIÓN DE SOFTWARE

Tanto para la realización de verificaciones como de validaciones se pueden utilizar distintos tipos de técnicas. En general, estas técnicas se agrupan en dos categorías:

Técnicas de Evaluación Estáticas: Buscan faltas sobre el sistema en reposo. Esto es, estudian los distintos modelos que componen el sistema software buscando posibles faltas en los mismos. Así pues, estas técnicas se pueden aplicar, tanto a requisitos como a modelos de análisis, diseño y código.

Técnicas de Evaluación Dinámicas: Generan entradas al sistema con el objetivo de detectar fallos, cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias entre la salida esperada y la salida real. La aplicación de técnicas dinámicas es también conocida como pruebas de software o testing y se aplican generalmente sobre código puesto que es, hoy por hoy, el único producto ejecutable del desarrollo.

Si bien es cierto, que en el caso de las técnicas estáticas, dado que detectan faltas su corrección es más directa. Mientras que las técnicas dinámicas, como se centran en los fallos su proceso de depuración asociado es mucho más complejo, puesto que se debe, primero, buscar la falta que provoca el fallo (lo cual, no es en absoluto inmediato como sabe cualquier programador) y posteriormente corregirlo.

4. EL MANTENIMIENTO

Este proceso tiene cambios asociados a:

- Corrección de errores
- Adaptaciones requeridas
- Mejoras

Generalmente vuelve a aplicar los pasos de análisis y desarrollo, pero en el contexto de software ya existente.

- Diseño
- Implementación
- Documentación del sistema
- Documentación de usuario

El mantenimiento reinicia el desarrollo en las fases de análisis.

Las actividades de análisis, durante el mantenimiento, implican la comprensión del alcance y efecto de una modificación deseada, además de las restricciones para hacer la modificación

Tipos de Mantenimiento

- Corrección: Para corregir los errores o defectos encontrados (no solamente por el cliente).
- Adaptación: Para acomodarlo a los cambios de su entorno.
- Mejora: Para ir más allá de los requisitos funcionales, atendiendo funciones adicionales.
- Prevención: Cambios a fin de que se pueda corregir, adaptar y mejorar fácilmente.

5. LA DOCUMENTACIÓN

La Documentación

- Del sistema (manual técnico, manual de proceso, documentos referentes al desarrollo).

- Del usuario (documentos relacionados a las funciones del sistema, orientado a las tareas y operaciones del usuario).
- Interna: O de código fuente. Comentarios contenidos en los programas.

Documentación del Usuario

- Una descripción funcional de lo que puede hacer el sistema.
- Cómo instalar el sistema y adecuarlo a otras configuraciones particulares del hardware.
- Un manual introductorio que explique, en forma sencilla cómo iniciarse en el sistema.
- Un manual de referencia que describa con detalle las ventajas del sistema, disponibles para el usuario y cómo se pueden usar.
- Una guía del operador (si se requiere operador del sistema), que explique cómo debe reaccionar éste ante situaciones surgidas mientras el sistema se encuentra en uso

Documentación del Código Fuente

- Comentarios de cabecera de programa o de módulo: Contiene información general sobre la creación o modificación del objeto.

- Comentarios de línea: Proveen información detallada sobre algún bloque corto de instrucciones contenidas en un módulo.

ACTIVIDAD…

Consultar en Internet que se debe observar en el momento de evaluar un software.

Consultar con empresas desarrolladoras de sistemas de información como hacen la evaluación de un producto de software.

ESTRATEGIAS METODOLÓGICAS

- Presentación de la unidad a cargo del profesor.
- Exposición magistral de los conceptos inherentes a la fase de Implantación y evaluación, utilizando diferentes medios didácticos para apoyar el proceso de aprendizaje.
- Resolución en grupo de talleres y cuestionarios.
- Lecturas guiadas.
- Consultas e investigaciones en Internet

RECURSOS

- Humanos: Profesor y alumnos
- Institucionales: Salón de clase
- Materiales: Texto guía

INDICADORES DE EVALUACIÓN

- Evaluación escrita sobre la unidad
- Evaluación del informe final de la implantación de un proyecto de desarrollo de software

CRITERIOS DE EVALUACIÓN

¿Cómo fue su proceso de implantación y evaluación para el proyecto de desarrollo de software?

BIBLIOGRAFÍA

www.wikipedia.org /wiki/Sistema - Definición dentro de contexto

www.itver.edu.mx/areas/deptos/sc/ - 1k - En caché - Páginas similares

FUENTE : PSL Productora de Software

Corporación Colombia Digital- Carrera 13 No.90-36, Of. 405 Tel (571)611-3059, Fax (571) 6113028. Bogotá, Colombia

www.Colombiadigital.net info@ColombiaDigital.net

http://www.degerencia.com/articulos.php?artid=186

www.conocimientosweb.net/dcmt/ficha1643.html - 15k - En caché - Páginas similares

www.intersoft.com.bo/portal/servicios/gestioncalidad.php - 12k - En caché - Páginas similares

www.cecam.sld.cu/pages/rcim/revista_1/articulos_pdf/r0100a01.pdf - Páginas similares

Fowler, Martin. UML Distilled. Addison -Wesley Longman Inc.1997

Sommerville, Ian. Ingeniería de Software. 6ª Edición. Addison Wesley. 2002

Pressman, Roger. Ingeniería del Software. 5ª Edición. Mc Graw Hill. 2002

Ruble, David. Análisis y Diseño Práctico de Sistemas Cliente/Servidor con GUI. Prentice - Hall, 1998.

Larman, Craig. UML y patrones. Introducción al análisis y diseño orientado a objetos. Ed. Prentice – Hall, 1999.

Rumbaugh James. Modelado y diseño orientado a objetos. Ed. Prentice – Hall, 1991.

Booch Grady, Rumbaugh James, Jacobson Ivar. El lenguaje Unificado de Modelado. Ed. Addison Wesley, 1999.

McCONNEL, Steve. Desarrollo y gestión de proyectos informáticos. McGraw Hill. España, 1997.