

EXAMEN DE TITULO

Contenido del Resumen

Malla de informática y telecomunicaciones 2018.....	5
Cursos de Software y Gestión	6
Tics III.....	6
Proyectos.....	6
Acta de constitución:.....	6
Creación Estructura Desglose de Trabajo (EDT):.....	7
Valor Ganado.....	8
Ruta critica.....	8
Curva S.....	8
Matriz de riesgo.....	8
Ingeniería en software	12
Metodologías tradicionales.....	12
Metodologías agiles	15
BPMN.....	15
Especificación de requisitos.	19
Requisitos Funcionales.....	20
Requisitos no Funcionales.....	20
Calidad del producto	22
UML.....	25
Modelo Conceptual.....	26
Diagrama de casos de uso.....	30
Formato de casos de uso.....	32
Diagrama de interacción	36
Diagrama secuencia	36
Diagrama de colaboración	39
Diagrama de clases.....	40
Patrones de diseño.....	43
Transición de estados.....	44
Diagrama componentes	45
Diagrama de despliegue.....	47

Diagrama de Actividades o diagrama de flujo	48
Arquitectura de sistemas	49
Atributos de calidad de la arquitectura	49
Patrones de Arquitectura	50
Evaluación de proyectos	55
Flujo de caja.....	55
Evaluación Social	56
Ejercicios.....	57
Cursos de Redes	58
Redes de datos	58
Modelo OSI y TCP/IP:	58
Frame Ethernet:	60
Protocolo ARP (Address Resolución Protocol)	60
Spinning Tree Protocol.....	62
Vlans:	63
IPV4:	65
La capa de Red.....	69
Protocolos de ruteo.....	70
Ruteo Estático o manual	71
Ruteo dinámico	71
Algoritmos de ruteo	71
RIPv2.....	72
EIGRP	72
OSPF (Open Shortest path first)	73
Protocolos de Routing interior y exterior (BGP)	73
Capa de Transporte	74
Cursos de telecomunicaciones.....	76
Comunicaciones Digitales	76
Ejercicios.....	77
Cursos de ciencias de la computación	78

Estructura de datos y algoritmos	78
Ordenes de complejidad	78
Estructuras fundamentales	78
Listas.....	78
Tablas de Hash (key,value).....	78
Grafos.....	79
Algoritmos de grafos	79
Algoritmo de Prim	80
Algoritmo de kruskal	80
Algoritmo de Rutas de costo mínimo.....	81
Floyd-Warshall.....	82
Dijkstra:	83
Arboles	84
Algoritmos de Ordenamiento	86
Recorridos y búsqueda	86
Bases de datos.....	87
Características de una BD.....	87
Modelamiento de las bases de datos	88
Algoritmo de transformación de E-R a Relacional.....	92
Normalización de una BD	93
Structured Query Language	95
Sistemas Operativos.....	96
¿Qué es un sistema operativo?.....	96
Procesos	97
Treads.....	101
Interrupciones	102
Cambios de contexto y PCB.....	102
Scheduling.....	105
Algorimos de Procesos	106
Procesos Threads	109
Condiciones de carrera.....	110
Semáforos	111

Variables de condición	111
Monitor	112
Fabulas - Modelos de problemas	112
Deadlocks	113

Malla de informática y telecomunicaciones 2018

udp Escuela de Informática
y Telecomunicaciones
FACULTAD DE INGENIERÍA Y CIENCIAS

Facultad de Ingeniería
Escuela de Informática Y Telecomunicaciones
Ingeniería Civil en Informática y Telecomunicaciones

Plan de Estudios N°3.2 (2018)

Sigla	Créditos
Nombre	
Idnum	Prerreq

Semestre 1	Semestre 2	Semestre 3	Semestre 4	Semestre 5	Semestre 6	Semestre 7	Semestre 8	Semestre 9	Semestre 10	Semestre 11
5 5 31 31	5 10 30 61	5 15 31 92	6 21 35 127	6 27 34 161	6 33 35 196	5 38 29 225	5 43 29 254	5 48 30 284	5 53 31 315	1 54 30 345
CBM-1000 7 Álgebra y Geometría 1 -	CBM-1002 6 Álgebra Lineal 6 1	CBM-1005 6 Ecuaciones Diferenciales 11 6, 7	CIT-2204 6 Probabilidad y Estadística 16 7	CIT-2750 6 Optimización 22 6, 12	CII-2000 6 Introducción a la Economía 28 12	CII-1000 6 Contabilidad y Costos 33 2	CIT-2203 6 Gestión Organizacional 38 5	CIT-33XX 6 Electivo Profesional 43 SC	CIT-33XX 6 Electivo Profesional 48 SC	
CBM-1001 7 Cálculo I 2 -	CBM-1003 6 Cálculo II 7 2	CBM-1006 6 Cálculo III 12 7	CBM-2000 6 Métodos Numéricos 17 6, 7	CIT-2106 6 Electrónica y Electrotecnia 23 18	CIT-2202 6 Modelos Estoc. y Simul. 29 15,16,22	CIT-2005 6 Ingeniería de Software 34 24, 25	CIT-2004 6 Arquitectura de sistemas 39 34	CIT-34XX 6 Electivo Profesional 44 SC	CIT-34XX 6 Electivo Profesional 49 SC	
CBQ-1000 6 Química 3 -	CBF-1000 7 Mecánica 8 2	CBF-1001 7 Calor y Ondas 13 7, 8	CBF-1002 7 Electricidad y Magnetismo 18 11, 12	CIT-2200 6 Proyecto en TICs I 24 15, 19	CIT-2101 6 Señales y Sistemas 30 16, 23	CIT-2102 6 Comunicaciones Digitales 35 30	CIT-2105 6 Criptografía y Seguridad en Redes 40 31	CIT-34XX 6 Electivo Profesional 45 SC	CIT-34XX 6 Electivo Profesional 50 SC	
CIT-1000 6 Programación 4 -	CIT-1010 6 Programación Avanzada 9 4	CIT-2000 6 Estructura de Datos 14 9	CIT-2001 6 Diseño y Análisis de Algoritmos 19 14	CIT-2002 6 Bases de Datos 25 19	CIT-2003 6 Sistemas Operativos 31 15, 25	FIC-1003 5 Derecho en Ingeniería 36 180cr	CIT-2201 6 Proyecto en TICs II 41 24,34,35	CIT-3200 6 Evaluación de proyectos TIC 46 33, 41	CIT-3201 7 Proyecto en TICs III 51 46	
FIC-1000 5 Comunicación para la Ingeniería 5 -	Minor/CFG 5 10 SC	CIT-2100 6 Redes de Datos 15 9	Minor/CFG 5 20 SC	Minor/CFG 5 26 SC	CIT-2103 6 Sistemas Digitales 32 23	CIT-2104 6 Arquitectura de Computadores 37 32	Minor/CFG 5 42 SC	CIT-33XX 6 Electivo Profesional 47 SC	CIT-33XX 6 Electivo Profesional 52 SC	
		CIG-1012 5 Inglés I 21 -		CIG-1013 5 Inglés II 27 21	CIG-1014 5 Inglés III 53 27			Práctica I	Práctica II	

- Formación en Ciencias Básicas
- Formación Transversal
- Formación en Ciencias de la Ingeniería
- Formación en Ingeniería Aplicada
- Prácticas y actividad de Titulación

Líneas de profundización en cursos electivos

Electivos 33XX: área Informática (4 cursos)
Electivos 34XX: área Telecomunicaciones (4 cursos)

SC: Según los pre-requisitos del curso en cuestión

Actividad de titulación Ing. Civil en Informática y Telecomunicaciones (30CR)

Egreso y Titulación

Cursos de Software y Gestión

Tics III

Proyectos

¿Qué es un proyecto?

Un proyecto es un esfuerzo temporal para llevar a cabo un producto, con recursos y tiempos de inicio y fin determinados. Se termina cuando se logran los objetivos establecidos. También se puede poner fin al proyecto si el cliente desea terminar el proyecto.

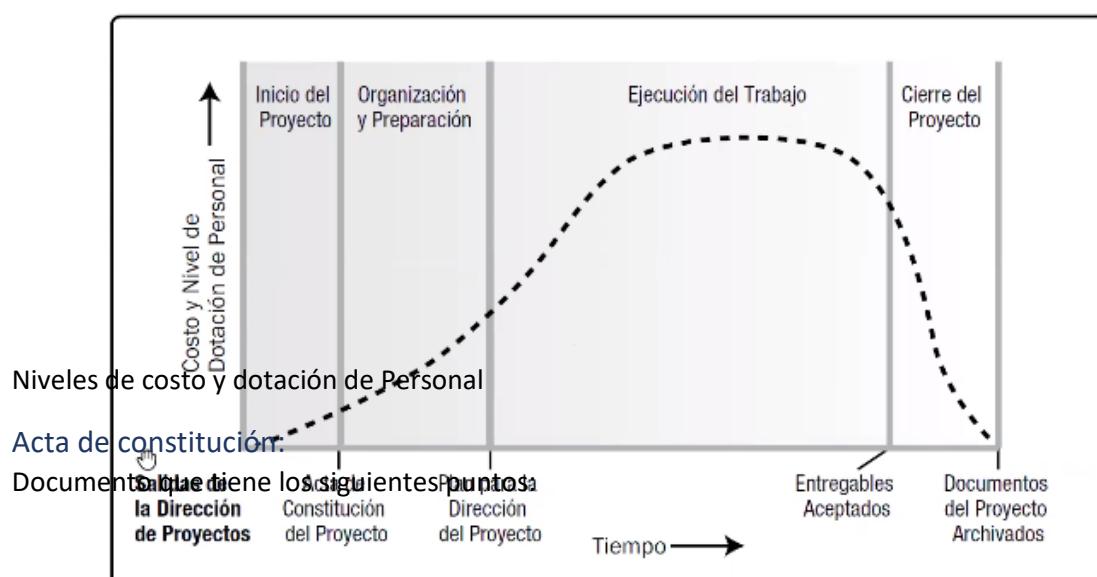
Esto se podría incluir en el análisis de riesgos.

Portafolio de proyectos se refiere a empresas muy grandes que tienen una cantidad muy grande de proyectos que pueden o no pertenecer a un programa, estos se generan a partir de la gestión de una PMO. Las fases de proyecto son 5.

- Inicio
- Planificación
- Ejecución
- Monitoreo y control
- Cierre

Director de proyectos o jefe de proyectos

Se encarga para liderar al equipo responsable de alcanzar los objetivos del proyecto y concretar el proyecto en el tiempo estimado, puede asignar métricas de monitoreo del proyecto, desempeño.



- El propósito o la justificación del proyecto,
- Los objetivos medibles del proyecto y los criterios de éxito asociados,
- Los requisitos de alto nivel,
- Los supuestos y las restricciones,
- La descripción de alto nivel del proyecto y sus límites,
- Los riesgos de alto nivel,
- El resumen del cronograma de hitos,
- El resumen del presupuesto,
- La lista de interesados,
- Los requisitos de aprobación del proyecto (es decir, en qué consiste el éxito del proyecto, quién decide si el proyecto tiene éxito y quién firma la aprobación del proyecto),
- El director del proyecto asignado, su responsabilidad y su nivel de autoridad
- El nombre y el nivel de autoridad del patrocinador o de quienes autorizan el acta de constitución del proyecto.

Gestión del alcance del proyecto:

Definir el objetivo del proyecto, Planificación del alcance, recopilar los requisitos, definir el criterio de éxito, por lo general es el cumplimiento de los requisitos funcionales y no funcionales junto con la satisfacción del cliente. La validación de este alcance se define a través del cumplimiento de los hitos del proyecto a partir de la firma del acta de proyecto.

Ej Hitos: Diseño, Esp de requisitos.

Creación Estructura Desglose de Trabajo (EDT):

¿Qué es EDT?

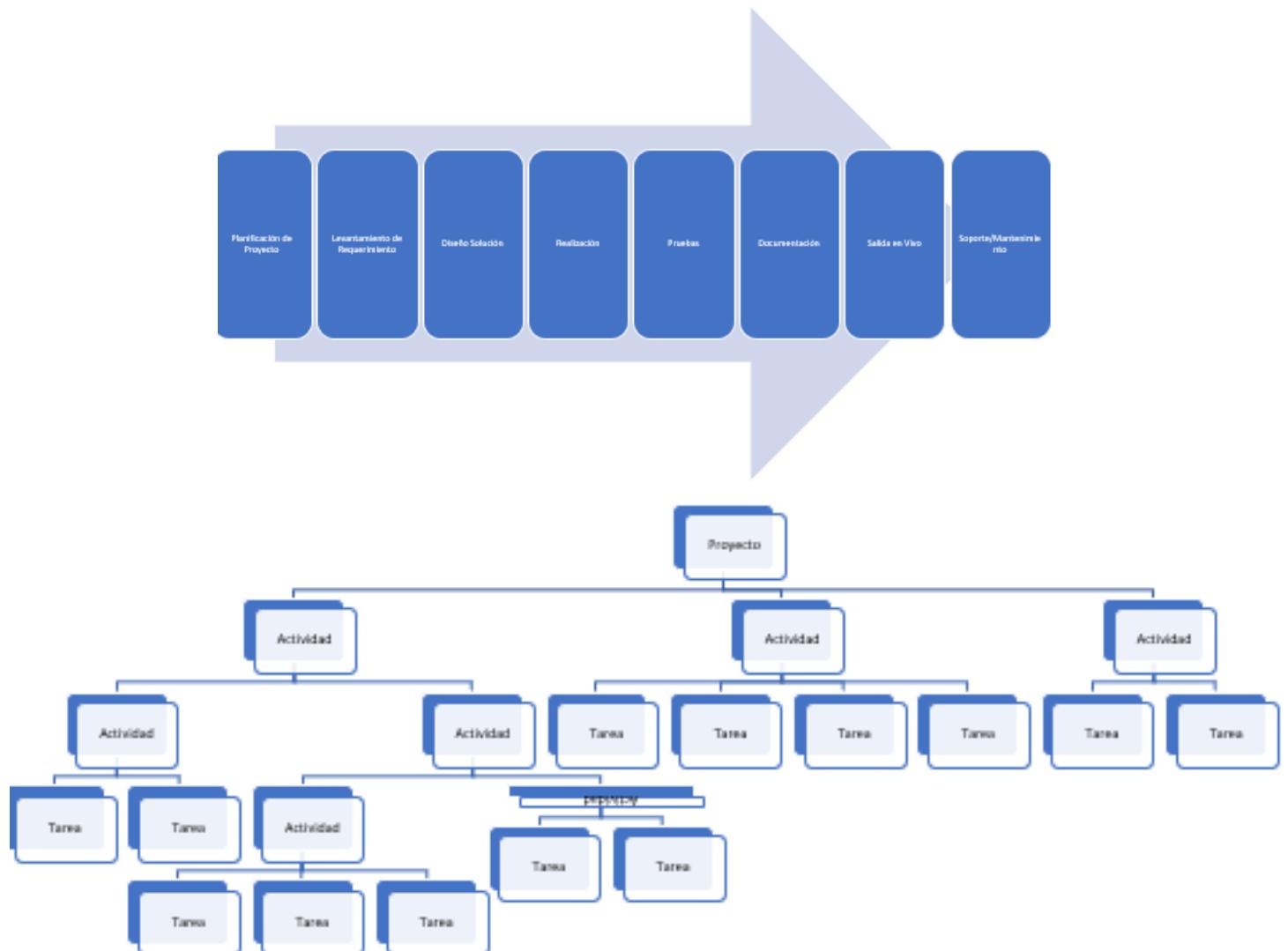
- Es una representación gráfica del proyecto de forma descriptiva que descompone de forma jerárquica el entregable final del proyecto
- Se logra organizando el proyecto en actividades y tareas en distintos niveles hasta lograr el nivel de detalle adecuado para planear y controlar el proyecto

¿Cómo construir un EDT?

1. Se debe iniciar identificando la meta o el objetivo general
2. Se definen los entregables principales del producto
3. Se procede a descomponer cada entregable en las actividades que se deben realizar para terminarla

4. A su vez, cada actividad debe dividirse en las tareas requeridas para su culminación
5. Se refina y pule el EDT hasta que se esté conforme con la identificación de actividades y el nivel de detalle.

Ejemplo:



Valor Ganado

Ruta critica

Curva S

Matriz de riesgo

Tipos:

- Técnicos: Mala esp. Requisitos, Tecnología no aprendida, desempeño

- Externo: Empresas externas, proveedores, mercado, tiempo cliente, clima.
- De la Organización: priorización del proy. , recursos humanos, Se enfermo un ql.
- Dirección de Proyectos: Estimación de tiempos, Planificación, Control.

Condiciones Definidas para las Escalas de Impacto de un Riesgo sobre los Principales Objetivos del Proyecto (Sólo se muestran ejemplos para impactos negativos)					
Objetivo del Proyecto	Se muestran escalas relativas o numéricas				
	Muy bajo /0,05	Bajo /0,10	Moderado /0,20	Alto /0,40	Muy alto /0,80
Costo	Aumento del costo insignificante	Aumento del costo < 10%	Aumento del costo del 10 - 20%	Aumento del costo del 20 - 40%	Aumento del costo > 40%
Tiempo	Aumento del tiempo insignificante	Aumento del tiempo < 5%	Aumento del tiempo del 5 - 10%	Aumento del tiempo del 10 - 20%	Aumento del tiempo > 20%
Alcance	Disminución del alcance apenas perceptible	Áreas secundarias del alcance afectadas	Áreas principales del alcance afectadas	Reducción del alcance inaceptable para el patrocinador	El elemento final del proyecto es efectivamente inservible
Calidad	Degradoación de la calidad apenas perceptible	Sólo se ven afectadas las aplicaciones muy exigentes	La reducción de la calidad requiere la aprobación del patrocinador	Reducción de la calidad inaceptable para el patrocinador	El elemento final del proyecto es efectivamente inservible

Esta tabla muestra ejemplos de definiciones del impacto de los riesgos para cuatro objetivos diferentes del proyecto. Deben adaptarse al proyecto individual y a los umbrales de riesgo de la organización durante el proceso de Planificación de la Gestión de los Riesgos. De forma similar, pueden desarrollarse definiciones del impacto para las oportunidades.

Ejercicio:

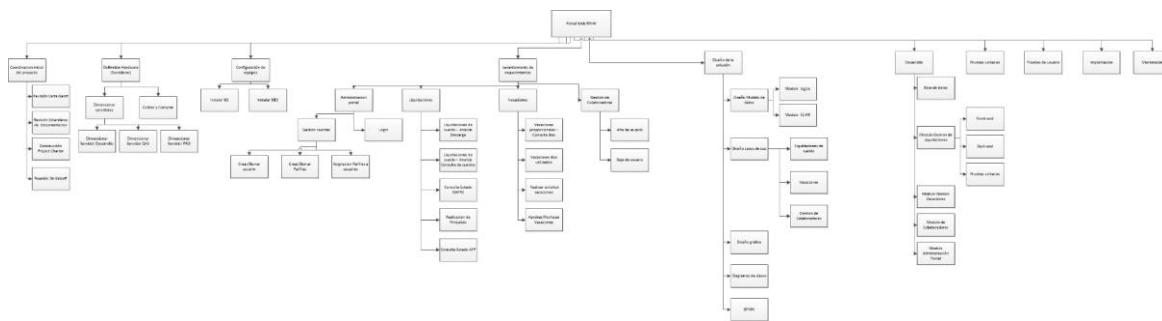
El subgerente de desarrollo de una empresa multinacional se encuentra en proceso de licitación para llevar a cabo la implementación de un portal empresa que permita a los usuarios de la compañía acceder a su información relacionada con Recursos Humanos. Los requerimientos que el cliente desea implementar en su portal de auto consultan son los siguientes:

- Consultar y descargar liquidaciones de sueldo
 - La descarga puede realizarse por una liquidación de sueldo puntual o intervalo de periodo
 - Consultar días de vacaciones proporcionales disponibles
 - Consultar días de vacaciones utilizados
 - Descargar comprobantes de vacaciones utilizados
 - Realización de solicitud de vacaciones
 - Aprobación o rechazo de vacaciones
 - Consultar estado de pago AFP
 - Consultar estado de pago ISAPRE
 - Capacidad para dar de alta y baja a empleados
 - Realización de finiquitos

*Considere en su planificación que el desarrollo es desde cero, por tanto, debe incluir todas las actividades para un desarrollo de esta clase.

*el cliente no posee infraestructura, debe considerar esto en su análisis

Resultado:



Ingeniería en software

¿Qué es la Ingeniería de Software?

- Es una disciplina formada por un conjunto de herramientas, métodos y técnicas que se utilizan en el desarrollo de software.

- Busca que el desarrollo de proyectos sea efectivo (calidad) y eficientes (costo).
- Permite guiar de manera sistemática todo el ciclo de vida del software.

Metodologías tradicionales

Modelo Cascada:

Proceso de desarrollo secuencial, también denominado desarrollo lineal o clásico.



- + Muy utilizado para adaptaciones o mejoras de sistemas existentes.
- + Útil cuando los requisitos son fijos
- + Útil cuando no se tiene una disponibilidad completa del cliente
 - Es raro que los proyectos sigan este flujo lineal
 - Es difícil que el cliente sepa de manera explícita todos los requisitos
 - El cliente debe tener paciencia para la entrega.

A este modelo se pueden aplicar múltiples cambios en su estructura según el proyecto

Modelo incremental:

Se refiere a ciclos de cascadas para llegar a un punto que esté todo terminado

El primer incremento corresponde al producto esencial, el plan se va modificando al entregar cada incremento para cubrir de mejor manera las necesidades del cliente.

Se entrega un producto operacional al finalizar cada iteración

+Util cuando no se tiene todo el personal necesario para el desarrollo de un proyecto

Modelos Evolutivos

Surge cuando los requisitos van cambiando a medida que se desarrolla el sistema.

De esta forma los modelos evolutivos son iteraciones

Modelo basado en prototipos:

Este paradigma se basa en la producción de una versión operacional del software, para que un conjunto de requisitos limitado (excluyéndose parte de la funcionalidad)

Prototipo Desechable: se construye luego de la fase de especificaciones de los requerimientos.

Prototipo Evolutivo: es iteracional y en cada iteración surge un refinamiento del prototipo

-Se puede adicionar funcionalidad intrascendente bajo presiones de uso normal.

-Posibilidad de pensar que es en si una especificación (solo es una parte)

-Puede demostrar funcionalidad que no es posible bajo apremios reales.

-El usuario puede creer que tiene frente a si el sistema completo y operable.

-Posibilidad de subestimar el proyecto, ya que las salidas están pronto disponibles.

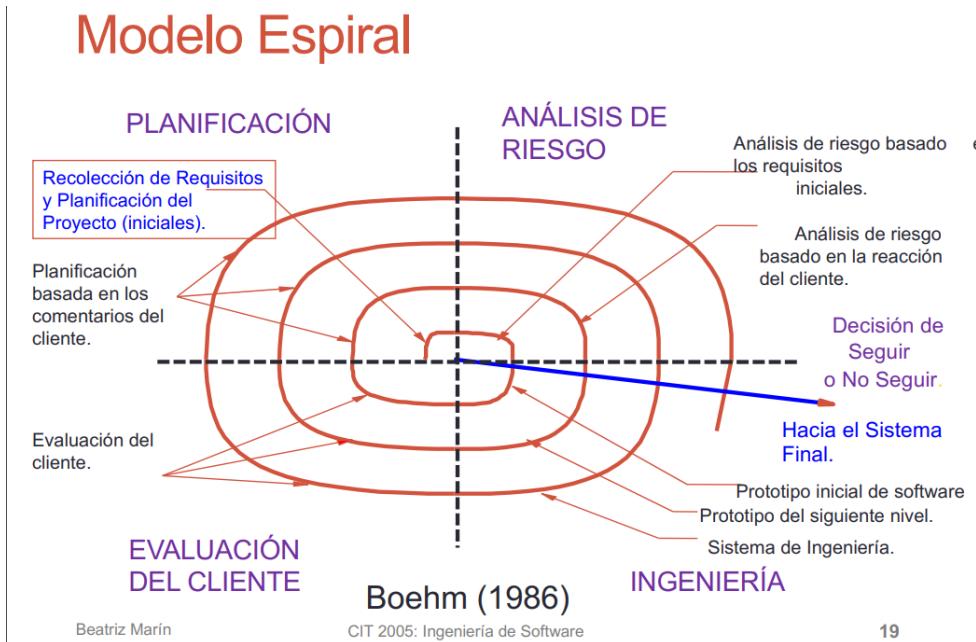
+Los usuarios tienen la posibilidad de interactuar con el prototipo y realimentar a los desarrolladores.

+ Para los usuarios es confortable enfrentarse a un prototipo que a una especificación

+Decrece el esfuerzo de desarrollo entre un 40% y un 70%.Un diseño rápido es posible cuando los requisitos son claros.

Modelo Espiral

Sus principales actividades son Planificación, Análisis de riesgo, Ingeniería, Evaluación del cliente.



- Criticidad del análisis de riesgo
- Difícil de “Vender” como algo controlable
- +Demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto.
- +Permite la utilización de la creación de prototipos como un mecanismo de reducción de riesgo.
- +Permite utilizar el enfoque de creación de prototipos en cualquier etapa de la evolución de del producto.

Metodologías agiles

Para el uso en general de las metodologías agiles se necesita una gran participación del cliente para la resolución rápida de problemas o obtención de información crítica, lo que es eficiente en cuanto al desarrollo del proyecto, pero no siempre va a estar disponible el cliente para esto, por esto, queda a criterio del grupo de trabajo y comodidad el uso de estas metodologías.

RUP • MDD • XP • COTS • TDD • SCRUM • Lean • DevOps

BPMN

Se utiliza para representar procesos de negocios o operaciones en formato workflow, tiene un formato fácil de entender por los stakeholders y personas no relacionadas con el mundo del software. Incluso si estos flujos son bastante complejos.

Se representan 5 tipos de elementos básicos.

- 1) Objetos de flujo
- 2) Objetos de datos
- 3) Objetos de conexión
- 4) Swimlines (carriles)
- 5) Artefactos

Objetos de flujo

Eventos: Ocurren dentro de un proceso de negocio y afectan al flujo del proceso. Normalmente, son gatillados por un trigger o impactan el resultado del sistema hay 3 eventos: inicial, intermedio y final.



Actividades: Puede ser un proceso de negocio, un proceso secundario o una tarea.



Entradas (Gateways): Representan decisiones o bifurcaciones.

Objetos de datos

Datos: Proveen información a un proceso o representan la información generada por un proceso.



Objetos de Conexión

Flujo de Secuencia: Se utiliza para mostrar el orden de los eventos



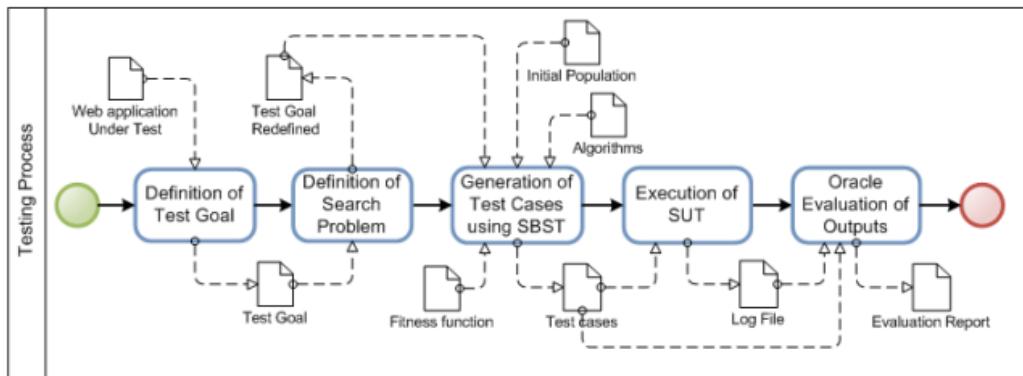
Flujo de mensajes: Se utiliza para mostrar el flujo de los mensajes entre los distintos eventos.



Flujo de Asociación: Se utiliza para asociar artefactos con objetos de flujo.

Swimlanes

Piscinas: Identifican los participantes dentro de un flujo de trabajo.



Carriles: Indican quien realiza que dentro de una piscina.



Artefactos

Se utilizan para mostrar las entradas y salidas de las actividades en los procesos.

Grupo: Se utiliza para clarificar la documentación o análisis.



Anotaciones: Se utilizan para incluir información adicional

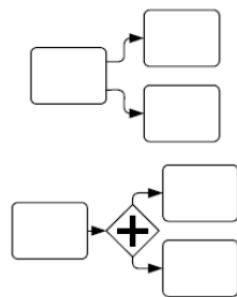


Elementos Comunes

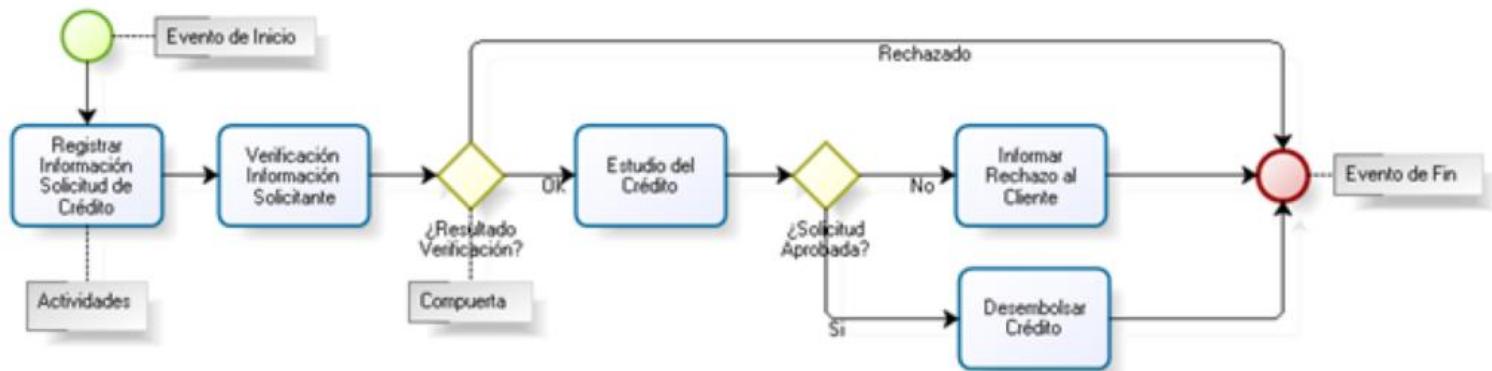
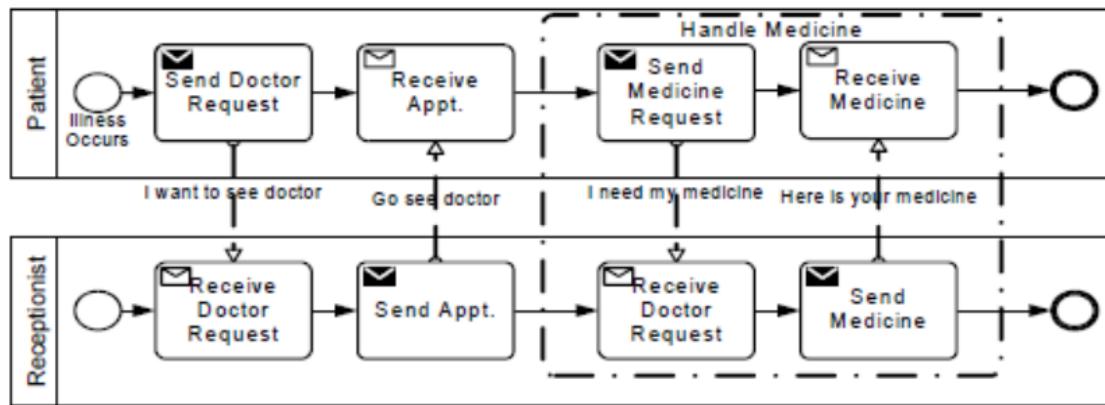
→ **Mensaje**: Describen una comunicación entre dos participantes de un proceso.



→ **Bifurcación**: Se utiliza para dividir la salida de una actividad.



Ejemplos



Especificación de requisitos.

Es un proceso sistemático de definición, comprensión, análisis y documentación de los requisitos.

Los requisitos definen los servicios que el sistema debe proporcionar a los usuarios, y las restricciones y condiciones de uso de estos.

La especificación debe separar la funcionalidad de implementación.

La especificación necesita usar un lenguaje de especificación orientado a procesos.

La especificación debe ser un modelo cognitivo

La especificación debe ser localizada y débilmente acoplada.

Muchos de los problemas de utilización son a partir de estas etapas.

Es difícil introducir cambios en los requisitos una vez que estos han sido consensuados entre clientes y desarrollador.

Hay una falta de entendimiento entre clientes y aquellos que desarrollan el sistema.

El documento de especificación de requisitos tiene que tener la siguiente estructura:

- Introducción
- Descripción general
- Requisitos específicos
- Apéndices
- Glosario

Requisitos Funcionales

Definen las funcionalidades del sistema

Id	Descripción	Entrada	Salida	Categoría	Prioridad
				<i>Evidente/Oculta</i>	<i>Baja/media/Alta</i>
ID	Función	Entrada	Salida	Prioridad	Categoría
RF1.1	Registrar la venta de una lista de artículos comprados	Lista de artículos comprados	Mensaje que indique que la operación se ha realizado exitosamente	Alta	Evidente
RF1.2	Registrar las ventas completadas	Venta completada	Venta registrada en BD	Alta	Oculta
RF1.3	Mostrar la descripción y precio de un producto	Identificador del producto	Se despliega la descripción y precio del producto	Media	Evidente
...					

Requisitos no Funcionales

Definen características del rendimiento y de la implementación, pueden ser atributos de calidad que puede tener el software.

Característica	Sub-Característica
Funcionalidad	Adecuación, Corrección, Interoperabilidad, Seguridad, Conformidad
Fiabilidad	Madurez, Tolerancia a fallos, Recuperabilidad, Conformidad
Usabilidad	Comprensibilidad, Aprendibilidad, Operabilidad, Atractividad, Conformidad
Eficiencia	Comportamiento temporal, Utilización de recursos, Conformidad
Mantenibilidad	Analizabilidad, Cambiabilidad, Estabilidad, Facilidad de prueba, Conformidad
Portabilidad	Adaptabilidad, Instalabilidad, Coexistencia, Reemplazabilidad, Conformidad

Id	Característica	Sub-Característica	Detalle y limitación
RNF1	Eficiencia	Comportamiento Temporal	La descripción y el precio de un producto aparecerá en menos de 5 segundos
RNF2	Portabilidad	Adaptabilidad	El sistema operativo debe ser Windows 10 y Ubuntu

Calidad del producto

Característica Sub-Característica

Funcionalidad Adecuación, Corrección, Interoperabilidad, Seguridad, Conformidad.

Fiabilidad Madurez, Tolerancia a fallos, Recuperabilidad, Conformidad.

Usabilidad Comprensibilidad, Aprendibilidad, Operabilidad, Atractividad, Conformidad.

Eficiencia Comportamiento temporal, Utilización de recursos, Conformidad.

Mantenibilidad Analizabilidad, Cambiabilidad, Estabilidad, Facilidad de prueba, Conformidad.

Portabilidad Adaptabilidad, Instalabilidad, Coexistencia.

Calidad del producto - Funcionalidad:

* Adecuación: Capacidad del producto software para proporcionar en conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

* Exactitud: Capacidad del producto software para proporcionar los resultados eo efectos correctos o acordados, con el grado necesario de precisión.

* Interoperabilidad: Capacidad del producto software para interactuar con uno o más sistemas especificados.

* Seguridad de acceso: Capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados.

* Cumplimiento funcional: Capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

Calidad del producto - Fiabilidad:

* Madurez: Capacidad del producto software para evitar fallar como resultado de fallos en el software.

* Tolerancia a fallos: Capacidad del software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces especificados.

* Capacidad de recuperación: Capacidad del producto software para re-establecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.

* Cumplimiento de la fiabilidad: Capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con la fiabilidad.

Calidad del producto - Usabilidad:

- * Capacidad para ser entendido: Capacidad del producto de software que permite al usuario entender si el software es adecuado y cómo puede ser usado para unas tareas o condiciones de uso particulares.
- * Capacidad para ser aprendido: Capacidad del producto de software que permite al usuario aprender sobre su aplicación.
- * Capacidad para ser operado: Capacidad del producto de software que permite al usuario operarlo y controlarlo.
- * Capacidad de atracción: Capacidad del producto de software para ser atractivo al usuario.
- * Cumplimiento de la usabilidad: Capacidad del producto de software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

Calidad del producto - Eficiencia:

- * Comportamiento temporal: Capacidad del producto software para proporcionar tiempos de resuelta, tiempos de proceso y potencia apropiadas, bajo condiciones determinadas.
- * Utilización de recursos: Capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.
- * Cumplimiento de la eficiencia: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia.

Calidad del producto - Mantenibilidad:

- * Capacidad para ser analizado: Es la capacidad del producto software para serle diagnosticadas deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas.
- * Capacidad para ser cambiado: Capacidad del producto software que permite que una determinada modificación sea implementada.

- * Estabilidad: Capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software.
- * Capacidad para ser probado: Capacidad del producto software que permite que el software modificado sea validado.
- * Cumplimiento de la mantenibilidad: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

Calidad del producto - Portabilidad:

- * Adaptabilidad: Capacidad del producto de software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismo destinos de quesos proporcionados para este propósito por el propio software considerado.
- * Instalabilidad: Capacidad del producto de software para ser instalado en un entorno especificado.
- * Coexistencia: Capacidad del producto de software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.
- * Capacidad para reemplazar: Capacidad del producto de software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.
- * Cumplimiento de la portabilidad: Capacidad del producto de software para adherirse a normas o convenciones relaciones con la portabilidad.

Calidad del producto - Calidad en Uso:

- * Efectividad: Capacidad del producto de software para permitir a los usuarios alcanzar objetivos especificados con exactitud y completitud, en un contexto de uso especificado.
- * Productividad: Capacidad del producto de software para permitir a los usuarios gastar una cantidad adecuada de recursos con relación a la efectividad alcanzada, en un contexto de uso especificado.
- * Seguridad física: Capacidad del producto de software para alcanzar niveles aceptables del riesgo de hacer daño a personas, al negocio, al software, a las propiedades o al medio ambiente en un contexto de uso especificado.
- * Satisfacción: Capacidad del producto de software para satisfacer a los usuarios en un contexto de uso especificado.

UML

Unified Modeling Language, Su objetivo es describir cualquier tipo de sistema en términos de diagramas orientados a objetos.

- Diagramas de secuencia
- Casos de uso
- Diagrama de clases
- Diagrama de objetos
- Diagrama de componentes,
- Diagrama de Distribución
- Diagrama de Actividad
- Diagrama de estados
- Diagrama de colaboración.

Funciones: son aquellas que el sistema se supone que tiene que hacer. Estas deben ser identificadas y anotadas en grupos lógicos y cohesivos. Por ejemplo, el sistema deberá autorizar pagos con tarjetas de crédito.

Atributos: son cualidades no-funcionales del sistema, por ejemplo, la facilidad de uso, que a menudo se confunden con las funciones del sistema.

- Funciones Evidentes: Obligatorias, el usuario está consciente que está realizando.
- Funciones ocultas: Obligatorias, pero no son visibles por el usuario.
- Funciones superfluas: Opcionales, su agregación no afecta significativamente el costo u otras funciones.

Ejemplos de atributos:

- Tiempo de respuesta
- Tolerancia a fallos
- Facilidad de Uso

Modelo Conceptual

El modelo conceptual es una descripción de componentes de software, es una representación de conceptos de dominio del problema en el mundo real.

El modelo conceptual muestra:

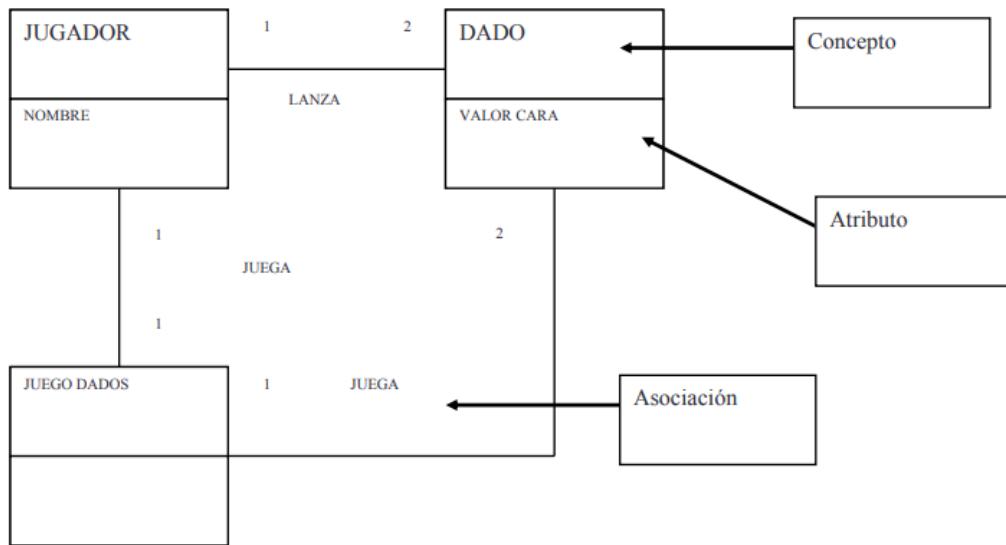
- Conceptos
- Asociaciones entre conceptos
- Atributos

Los artefactos de software, como una ventana o una base de datos, no forman parte del modelo conceptual, salvo que el dominio a modelar se refiera a conceptos de software, por ejemplo, un modelo de interfaces graficas.

Representa la vista **estática** del dominio por tanto **no se define ninguna operación**.

En términos informales el **concepto es una idea, cosa u objeto**.

Ejemplo:



Como identificar conceptos:

Mediante lista de categorías de conceptos

Realice una lista de conceptos candidatos siguiendo la siguiente lista, que contiene categorías comunes usualmente utilizadas.

Categoría de Concepto	Ejemplo
Objeto Físico o Tangible	Caja, Avión, Producto
Especificación, diseño o descripción de cosas Lugares	Especificación de producto, Plan de vuelo Tienda, Aeropuerto, Oficina
Transacción	Venta, Compra, Reserva
Líneas de ítems de Transacción	Línea de Productos de Pedido a Bodega
Roles de Personas	Cajero, Bodeguero
Contenedores de otras cosas	Tienda, Avión, Bodega
Cosas en Contenedores	Producto, Pasajero, Caja
Otros Computadores o Sistemas electromecánicos externos a nuestro sistema	Sistema de Autorización de Tarjeta de Crédito Control de Trafico Aéreo
Conceptos Abstractos	Hambre
Organizaciones	Departamento de Ventas
Eventos	Venta, Despegue, Aterrizaje
Procesos (a menudo no representado como un concepto, pero puede ser)	Venta de producto
Catálogos	Catálogo de Productos Catálogo de Partes
Registros de Finanzas, Trabajos, contratos, materias legales	Contratos de Empleo
Instrumentos Financieros y Servicios	Línea de Crédito
Manuales, Libros	Manual de reparaciones

Mediante identificación entre frases:

Identificar sustantivos en las descripciones textuales de los casos de uso y considerarlos como candidatos a conceptos o atributos.

ERROR

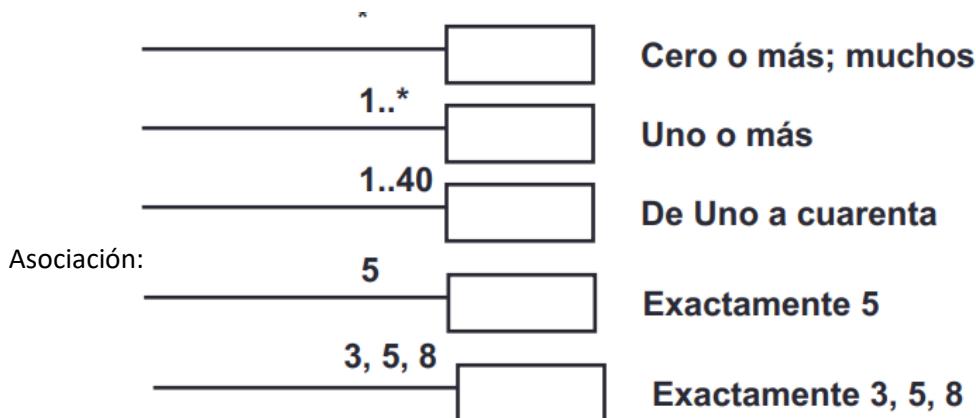
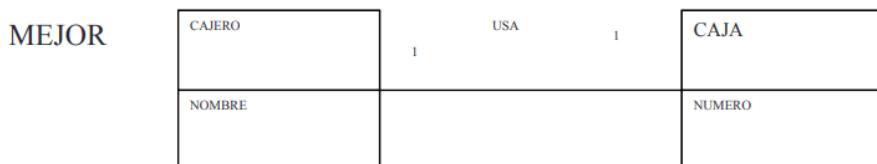
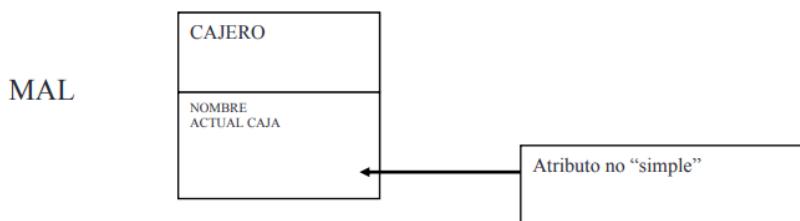
“Si pensamos en un concepto X como un numero o texto en mundo real, X probablemente sea un atributo”

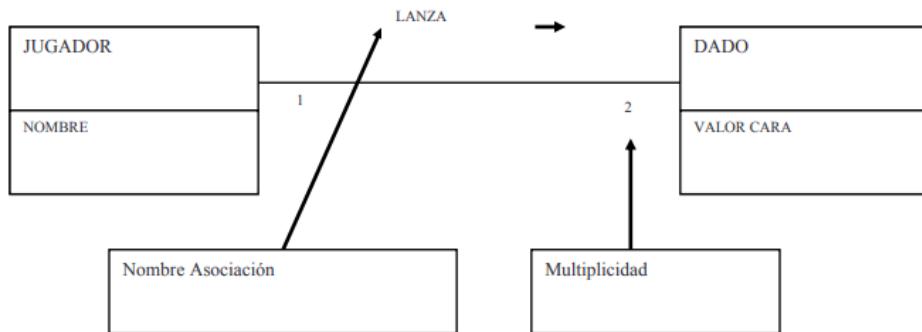
Atributos

Un atributo es un valor lógico de un datos de un objeto.

Deberán ser incluidos en el modelo todos aquellos por los cuales los requerimientos surgen o implican la necesidad de recordar información.

Los atributos en un modelo conceptual deben ser preferentemente atributos simples o valores de datos puros.





La flecha indicada en el nombre (opcional) indica la dirección en la que se debe leer el nombre de la asociación. No indica dirección de visibilidad o navegación. De omitirse la flecha la convención de lectura es de izquierda a derecha o de arriba abajo.

Es posible tener múltiples asociaciones entre 2 conceptos o entre un mismo objeto (común)

Identificar asociaciones:

Categoría	Ejemplo
A es parte física de B	Ala-Avión
A es parte lógica de B	Línea de Ítem de venta- venta
A está físicamente contenida en B	Pasajero-Avión
A está lógicamente contenida en B	Vuelo-Plan de Vuelo
A es una descripción para B	Plan de Vuelo – Vuelo
A es una línea de Ítem de una transacción o reporte de B	Línea de Ítem de venta – Venta
A es conocido /grabado /registrado /reportado /capturado en B	Venta- Caja
A es un miembro de B	Cajero – Tienda
A es una subunidad organizacional de B	Departamento – Tienda
A usa o administra B	Cajero –Caja
A se comunica con B	Cliente-Cajero
A se relaciona con una transacción de B	Pasajero-Pasaje
A es una transacción relacionada con otra transacción de B	Reserva- Cancelación

Diagrama de casos de uso

Permite describir los procesos del dominio.

Un caso de uso es un documento narrativo que describe la secuencia de eventos de un actor (agente externo) que utiliza un sistema para completar un proceso.

Los casos de uso son historias o casos de utilización de un sistema, que normalmente abarca varios pasos.

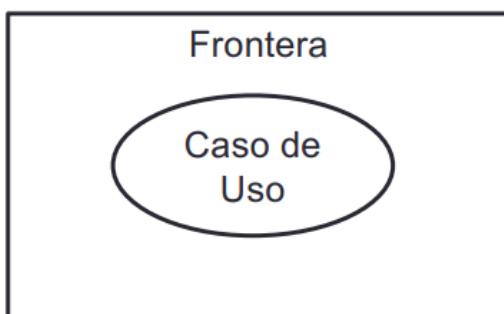


Notación en UML para un actor



Comprar productos

Notación en UML para caso de uso



- Un actor es una entidad involucrada en el sistema, que de alguna manera participa en la historia del caso de uso.
- Estimula el sistema con eventos de entrada o recibe algo de él.
- Los actores están representados por el papel (rol que desempeñan)
- Se indica la relación entre el actor y caso de uso mediante una línea.

Frontera:

- Es el límite físico y/o lógico para el caso de uso
- Representa la frontera del sistema modelado
- Los actores pueden ser internos o externos a la frontera del caso de uso
- Una frontera puede encerrar a mas de un caso de uso.

Identificación:

Un método de identificación se basa en actores:

1. Se identifican los **actores** relacionados con un sistema o empresa.
2. Para cada actor se identifican los **procesos** que inician o en los cuales participan.

Otro método de identificación se basa en eventos:

1. Se identifican los **eventos** externos a los que un sistema ha de responder.
2. Se relacionan los **eventos** con los **actores** y con los **casos de uso**.

Clasificación:

Los casos de uso deberían clasificarse en **primarios, secundarios y opcionales** para asignarles prioridad de desarrollo.

Primarios: Los casos de uso primarios representan los procesos más importantes, como Comprar productos

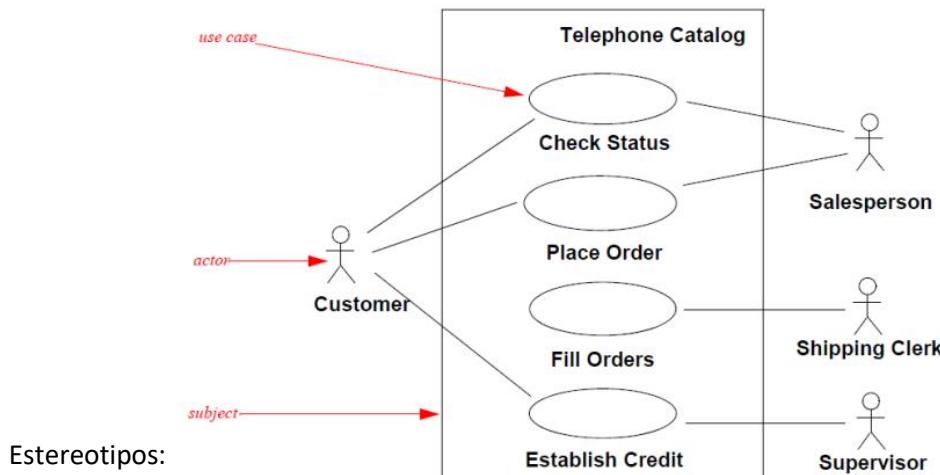
Secundarios: Los casos secundarios de uso representan procesos menores o raros; por ejemplo, Solicitud de surtir un nuevo producto

Opcionales: Los casos opcionales de uso representan procesos que pueden no abordarse

Al momento del desarrollo los casos de uso deben ordenarse según su clasificación y se deben abordar según su orden de prioridad. Si el caso de uso es muy complejo, se puede abordar su forma simplificada.

Un diagrama de casos de uso explica gráficamente un conjunto de casos de uso de un sistema, los actores y la relación entre ellos.

Los **casos de uso** se muestran en **ovalos**, los **actores** son las figuras estilizadas y las **relaciones** son **líneas**



Estereotipos:

<< include >> : Se recomienda utilizar cuando se tiene un **conjunto de características** que son similares en más de un caso de uso.

De esta forma lo mas adecuado es derivar un caso de uso separado con las características similares, con el fin de evitar copiar y pegar y hacer referencia a el desde el caso de uso original.

Se hace referencia a otro caso de uso que se ocupa en este, por lo general ese caso lo hace el sistema

<< extend >> : Se recomienda utilizar cuando un caso de uso es **similar a otro** (características), pero **hace algo más**. Es decir, cuando se describa una variación en un comportamiento normal.

Se debe poner el comportamiento normal en un caso de uso y el comportamiento inusual en otro caso de uso. Dibujar todas las variaciones como extensiones.

Aclaración de definiciones:

Ambos estereotipos hay que sacar fuera el comportamiento común de la mayoría de los casos de uso a un caso de uso simple que es usado por otros casos de uso.

En el caso del <<extend>>, los actores tienen una relación con el caso de uso que esta extendido. Se asume que el actor podrá trabajar con el caso de uso base y con todas las extensiones.

En cambio, con <<include>>, a menudo no hay actores asociados con el caso de uso común.

Formato de casos de uso

Los casos de uso de alto nivel describen un proceso de manera breve. Son útiles en niveles bajos de análisis para la definición de procesos.

Los casos de uso expandidos describen un proceso mas a fondo que el de alto nivel, su principal diferencia es la sección “curso normal de los eventos”, que describe los eventos del proceso paso a paso.

Caso de Uso de Alto nivel:

Formato de un Caso de uso de Alto Nivel

Caso de uso: <Nombre>

Actores: <actor1>,, <actor n>

Tipo: <tipo del caso>, se asume es tipo esencial, sólo se especifica Primario, secundario u opcional.

Descripción: <una breve descripción del proceso>

Ejemplo de un Caso de uso de Alto Nivel

- Ejemplo: comprar artículos en una tienda cuando se emplea una terminal en el punto de venta.

Caso de uso: Comprar Productos

Actores: Cliente, Cajero

Tipo: Primario

Descripción:

Un cliente llega a la caja registradora con los artículos que comprará. El Cajero registra los artículos y cobra el importe. Al terminar la operación, el Cliente se marcha con los artículos comprados.

Caso de uso Expandido:

Formato de un Caso de uso Expandido

Caso de uso: <Nombre>

Actores: <actor1>, ..., <actor n>

Propósito: <propósito>

Tipo: <tipo del caso>, se asume es tipo esencial, solo se especifica Primario, secundario u opcional.

Resumen: <una breve descripción del proceso>

Referencias cruzadas: <casos o funciones del sistema

Formato de un Caso de uso Expandido

• **Curso Normal de los Eventos**

Acción del actor	Respuesta del sistema
1º Acción 1	1º Respuesta 1
.	.
.	.
Nº Acción n	Nº Respuesta n

• Cursos Alternativos:

- <Línea del curso que posee alternativa>: < Acción alternativa>
 -
 - **Sección:** <Nombre Sección Alternativa>

Ejemplo de un Caso de uso Expandido

Caso de Uso:	Registrar cursos
Actores:	Estudiante (Iniciador), sistema de facturación
Propósito:	Capturar la inscripción de un curso y sus posibles modificaciones
Resumen:	El estudiante entra en el sistema y revisa la lista de cursos que tiene inscritos en el semestre actual. Puede agregar y borrar los cursos de la lista, revisar e imprimir la lista de cursos. Al terminar, se almacena la inscripción de los cursos.
Tipo:	Primario y esencial
Referencias cruzadas:	R1.1, R1.3, R1.4, R1.5, R1.9, R1.10, R2.1

FACULTAD DE INGENIERIA

Ejemplo de un Caso de uso Expandido

Sección principal	Acción de los actores	Respuesta del sistema
Flujo normal de eventos	1. Este CU empieza cuando el estudiante se conecta al sistema e introduce su password.	2. El sistema verifica su password y verifica que el estudiante no tenga deudas. El sistema muestra la lista de cursos inscritos en el semestre actual.
	3. El estudiante elige la actividad que desea realizar: Si la actividad es Agregar se hace el subflujo S-1 “Agregar curso” Si la actividad es Borrar se hace el subflujo S-2 “Borrar curso” Si la actividad es Imprimir se hace el subflujo S-3 “Imprimir inscripción” Si la actividad es Salir el CU termina.	

Ejemplo de un Caso de uso Expandido

S-1: Agregar curso.	Acción de los actores	Respuesta del sistema
Flujo normal de eventos	1. El Estudiante selecciona el curso que desea tomar.	2. El sistema muestra la lista de cursos que el estudiante puede tomar: sigla, nombre, paralelo, horario, profesor, número de créditos.
		3. El sistema verifica que el estudiante posee los prerequisitos para la inscripción del curso. El sistema enlaza al estudiante con el paralelo del curso.

Ejemplo de un Caso de uso Expandido

Flujos alternativos	Línea 2: El identificador del estudiante no es válido. El usuario puede intentarlo de nuevo o salir. Línea 2: El estudiante presenta deudas. La única opción disponible es Salir.
	S1 - Línea 3: No hay cupo para la inscripción. Se pide seleccionar otro paralelo u otro curso. S1 - Línea 3: No posee prerequisitos, se pide seleccionar otro curso. S1 - Línea 3: No se puede crear el enlace estudiante - curso. Se guarda la información para crear el enlace posteriormente.

Diagrama de interacción

Los diagramas de interacción muestran cómo se comunican los objetos en una interacción

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones

Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para representar cierto comportamiento.

Habitualmente, un diagrama de interacción capta el comportamiento de un solo caso de uso.

Hay dos tipos de diagramas de interacción Diagramas de secuencia y de colaboración.

Diagrama secuencia

Muestra la secuencia cronológica de mensajes entre objetos durante un escenario concreto.

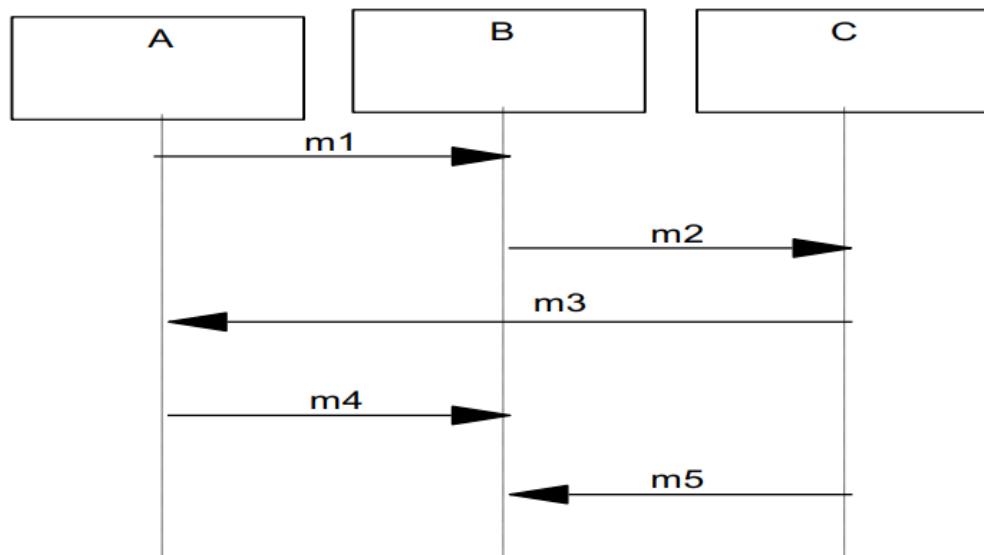
Cada objeto tiene una **Línea vertical** denominada línea de vida del objeto. Esta línea representa la vida del objeto durante la interacción.

Cada mensaje se representa mediante una flecha entre las líneas de vida de dos objetos.

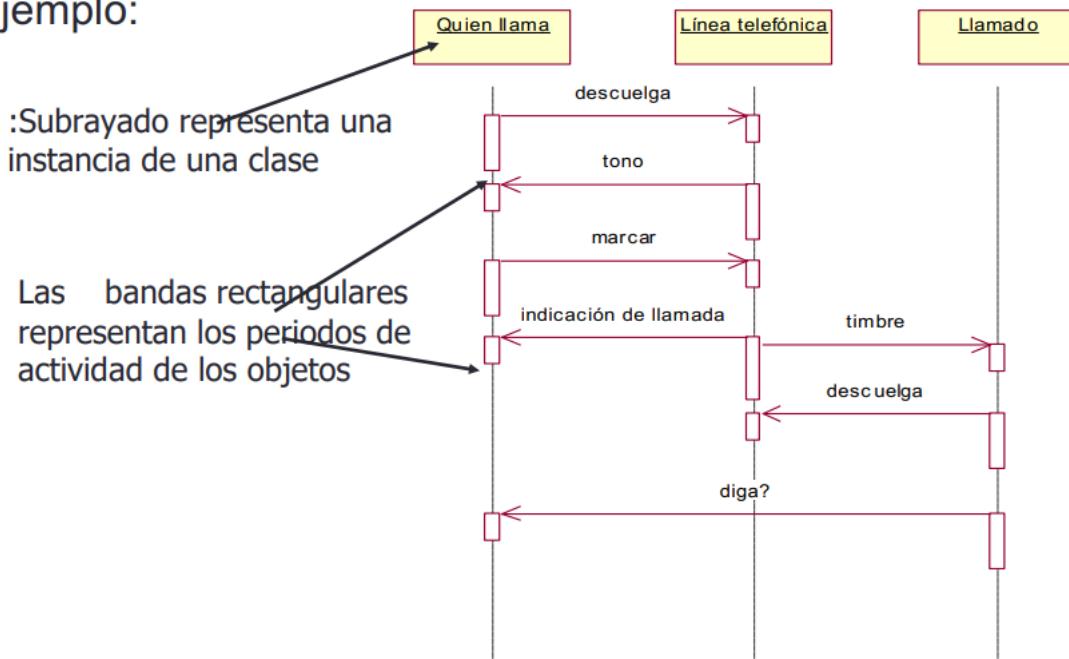
El tiempo transcurre de arriba abajo.

Cuando existe una demora entre el envío y la atención se puede indicar usando la línea oblicua.

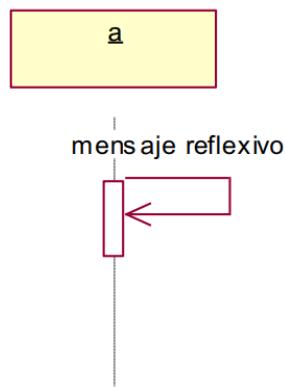
Ejemplo Notación:



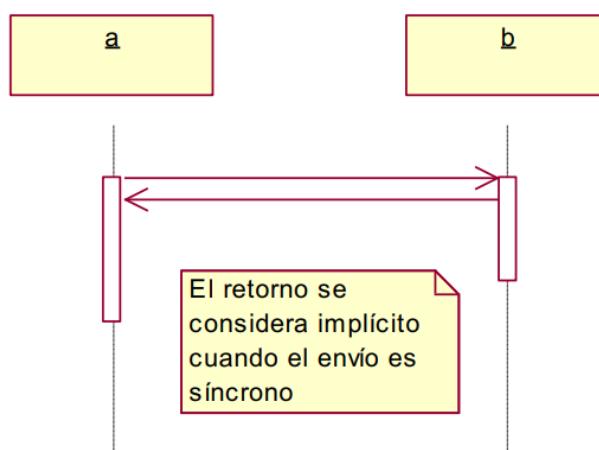
▪ Ejemplo:



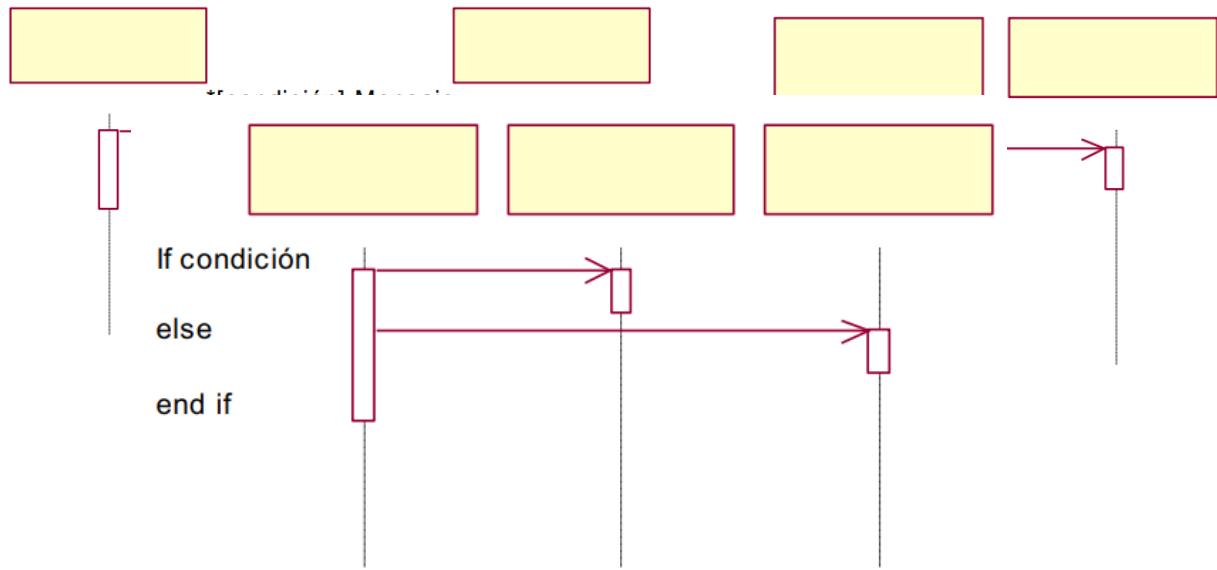
Un objeto puede enviarse a sí mismo un mensaje. Este mensaje (auto delegación) se indica con una flecha que vuelve al objeto.



Los diagramas pueden incluir un regreso, el cual indica el regreso de un mensaje, no un nuevo mensaje. Normalmente, no es necesario indicar el retorno del control.



El mensaje puede representar interacciones. Existe un marcador de iinteraccion, que muestra que un mensaje se envia muchas veces varios objetos receptores. Ej. Para cada linea de pedido.



En el mensaje ser puede poner una condición que indica cuando se envía el mensaje. Ej.: si hay existencia.

Diagrama de colaboración

El contexto de una interacción comprende los argumentos, las variables locales creadas en ejecución y los enlaces entre objetos que participan en la interacción.

La colaboración se realiza mediante el intercambio de mensajes. Un mensaje desencadena **una acción** en el objeto destinatario.

Son útiles en la fase exploratoria **para identificar objetos y métodos**. Se ocupa para un caso de uso específico. Ejemplo:

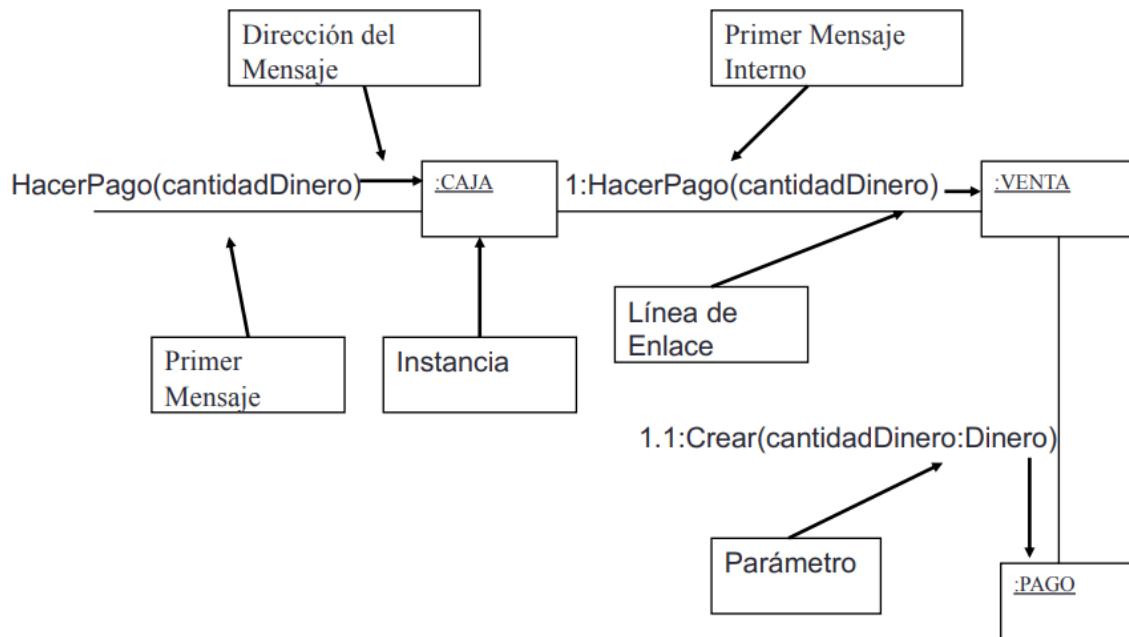


Diagrama de clases

El diagrama de clases ilustra las especificaciones para las clases de software y las interfaces en la aplicación.

Típicamente este compuesto por:

- Clases, Asociaciones y atributos
- Interfaces, con sus operaciones constantes
- Métodos
- Información de los tipos de atributos
- Navegabilidad
- Dependencia

Para definir las clases debemos hacernos las siguientes preguntas:

¿Cómo se conectan los objetos a otros objetos?

¿Cuáles son los métodos de la clase?

Para el diseño:

Se debe **inspeccionar el diagrama de interacción**, los cuales sugieren las conexiones necesarias entre **objetos y los métodos** que cada clase de software debe definir.

Un diagrama de clases requiere de los diagramas de interacción, donde el diseñador identificara las clases de software que participan en la solución además de los métodos de estas y el modelo conceptual, el que aportara los detalles a las definiciones de las clases.

Como hacerlo:

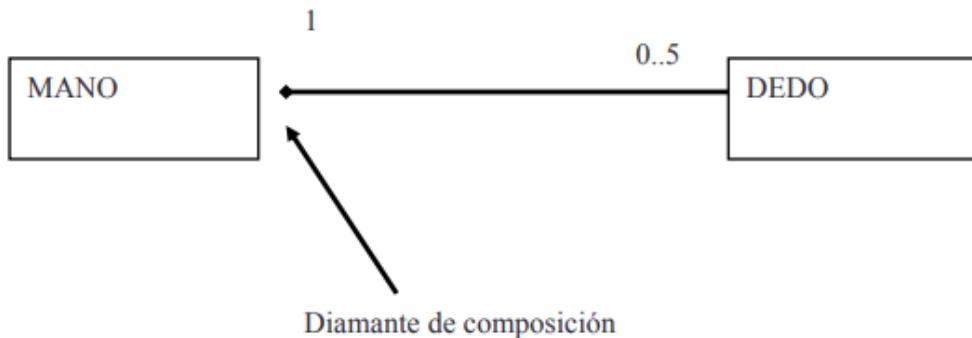
1. Identifique todas las clases participantes en la solución del software (Obtén^galas analizando los diagramas **de interacción y el modelo conceptual**)
2. Dibújelos en un diagrama de clases.
3. Duplique los atributos de los conceptos asociados en el modelo conceptual.
4. Agregue los nombres de los métodos analizando los diagramas de interacción.
5. Agregue la información de tipos para los atributos y métodos.
6. Agregue las asociaciones necesarias para soportar los requerimientos de visibilidad de los atributos.
7. Agregue las flechas de navegabilidad a las asociaciones para indicar la dirección de la visibilidad de los atributos.

Asociaciones:

- La asociación expresa una conexión semántica bidireccional entre clases.
- Una asociación es una abstracción de la relación existente en los enlaces entre los objetos.

Agregaciones y Composiciones:

La agregación es un tipo de asociación utilizada para modelar relaciones de “Partes de un todo” El todo es llamado compuesto y las partes no poseen denominación.

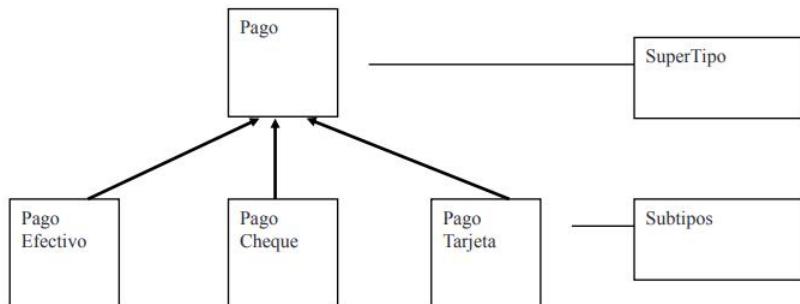


Generalizaciones:

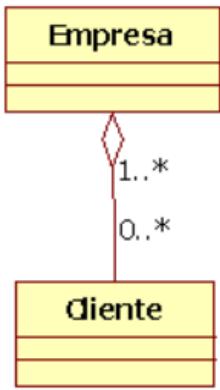
Algunos conceptos muy similares; conceptos tales como: Pago en efectivo, pago con cheque, pago con tarjeta de crédito.

En estos casos es posible organizarlos en un tipo jerárquico llamado generalización, en el cual un supertipo representa un concepto más general y los subtipos especificaciones de este. Ejemplo:

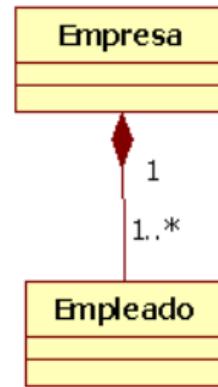
- Todos los miembros de un subtipo son miembros de su supertipo.



Agregación y composición



Agregación



Composición

Luego de hacer el modelo de clases:

¿El modelo realizado satisface los requisitos?

¿El Modelo es Correcto?

Tipos de atributos:

public (+): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.

private (-): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden utilizar).

protected (#): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accedido por métodos de la clase además de las subclases que se deriven (ver herencia).

Patrones de diseño

Patrones de Diseño

- Los patrones de diseño nombran, explican y evalúan un diseño recurrente en un sistema orientado a objetos.
- Proponen una forma para **reutilizar** la experiencia de los diseñadores, clasificando y describiendo **formas de solucionar problemas** que ocurren de manera frecuente en el desarrollo de software.

Transición de estados

Describe el ciclo de vida completo de un objeto, con los estados validos que puede tener y las acciones que pueden ocurrir durante su vida.

- Estado inicial



- Estado intermedio

Pagado

- Estado Final



- Transición

[precondition] event / [postcondition]



Ejemplo

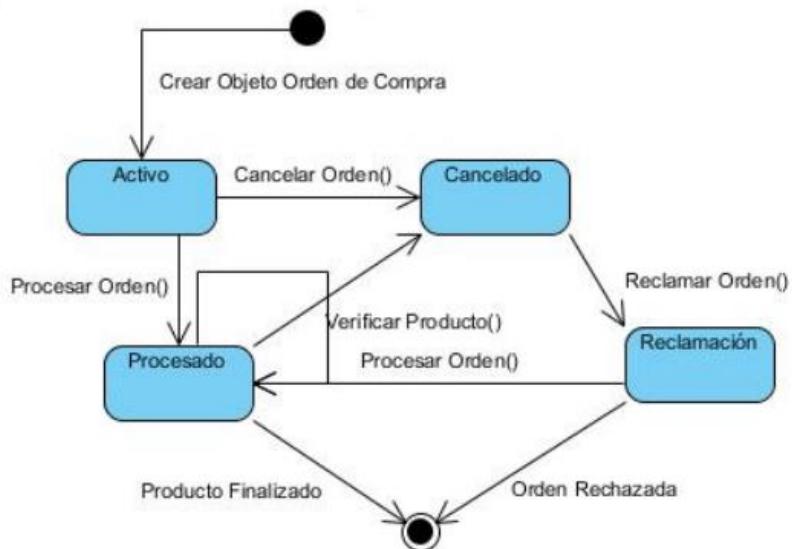
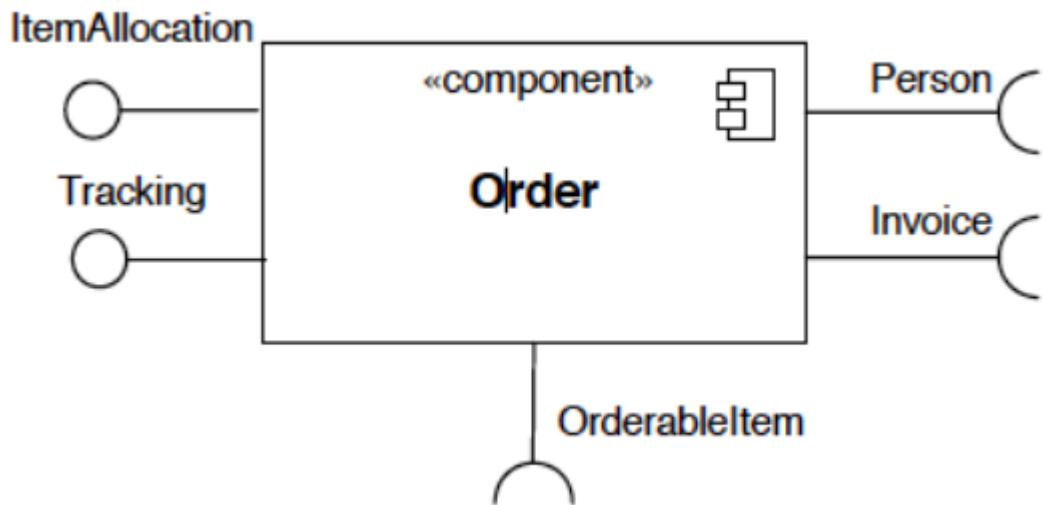


Diagrama componentes

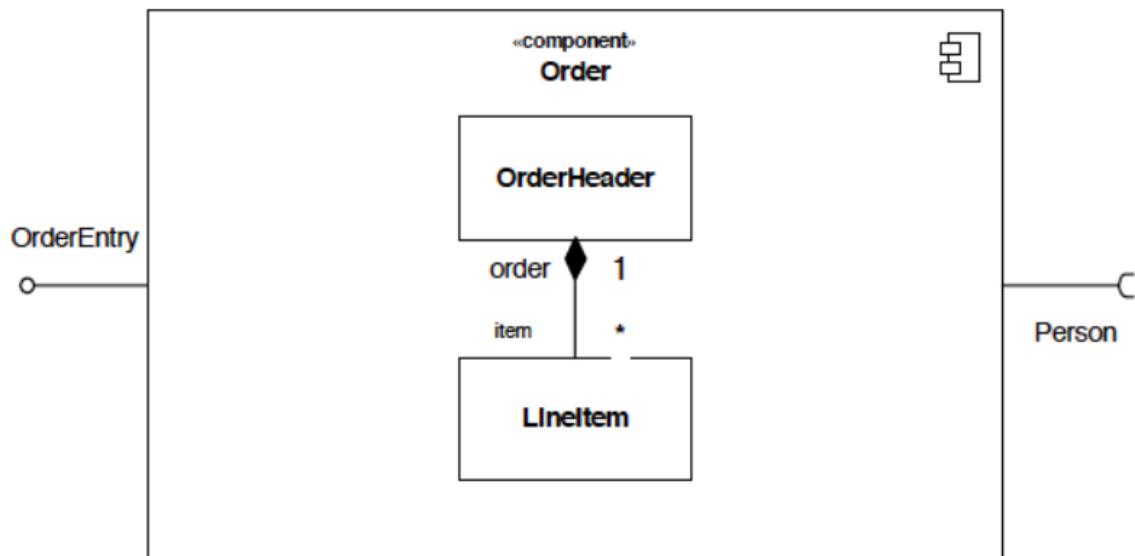
Son utilizados para modelar la arquitectura de un sistema.

Un componente representa una parte modular del sistema, que encapsula el estado y el comportamiento de un conjunto de clases.

Un componente se comunica con otros componentes mediante dependencias e interfaces. Ej. Un componente con 2 interfaces entregadas y 3 interfaces requeridas.



Representación interna de un componente:



Dependencias entre componentes:

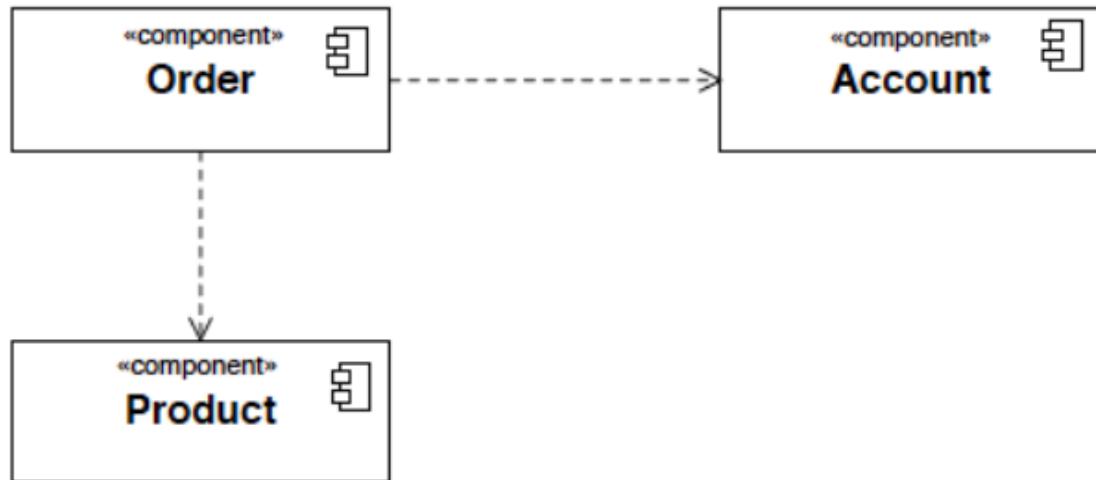


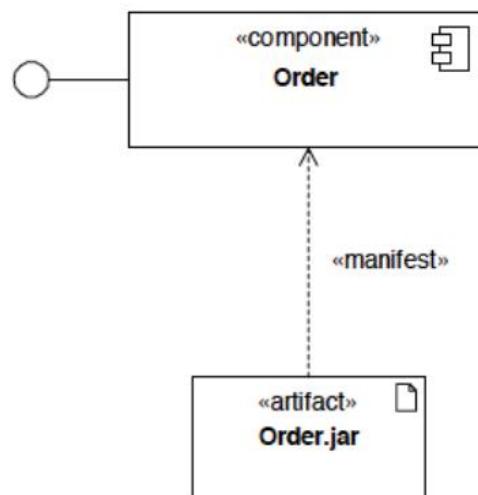
Diagrama de despliegue

Se utiliza para representar como serán distribuidos los componentes de la arquitectura de software.

Se deben especificar los artefactos, que son piezas de información físicas relacionadas a los componentes de la arquitectura

Además, no es necesario especificar los nodos físicos que conforman la arquitectura.

Notación para un artefacto:



Notación para un nodo:

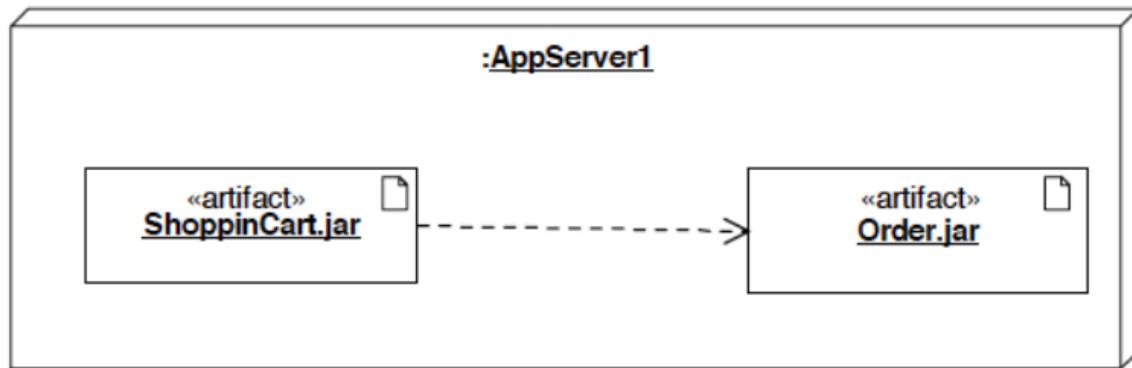


Diagrama de Actividades o diagrama de flujo

El diagrama de actividades es considerado de comportamiento porque describe lo que es importante para la comunicación clara y concisa. Demuestra lógica en el algoritmo o proceso el cual se quiere describir.

Componentes:

- **Acción:** Un paso en la actividad en el que los usuarios o el software realizan una tarea dada. En Lucidchart, las acciones se representan a través de rectángulos con aristas redondeadas.
- **Nodo de decisión:** Una rama condicional en el flujo que se representa con un diamante. Incluye una sola entrada y dos o más salidas.
- **Flujos de control:** Otro nombre para los conectores que muestran el flujo entre pasos en el diagrama.
- **Nodo inicial:** Simboliza el inicio de la actividad. El nodo inicial se representa con un círculo negro.
- **Nodo terminal:** Representa el paso final en la actividad. El nodo terminal se representa por medio de un círculo negro de contorno blanco.

<https://www.lucidchart.com/pages/es/tutorial-diagrama-de-actividades-uml>

Arquitectura de sistemas

Atributos de calidad de la arquitectura

- **Disponibilidad:** Tiempo o intervalo de tiempo cuando el sistema debe estar disponible ej métrica: 99%, 99,99999%-> bancos etc. del tiempo.
- **Interoperabilidad:** Integración con otros sistemas, por ejemplo, agregar un sistema a un software ya creado. Métrica: % de información intercambiada correctamente.
- **Modificabilidad:** sistema ya operando que necesita un cambio Métrica: cantidad, tamaño, complejidad, esfuerzo, dinero.
- **Desempeño:** Latencia, deadlines de procesamiento, miss-rate.
- **Seguridad:** % de módulos comprometidos a un ataque, % de datos comprometidos.
- **Comprobabilidad:** facilidad con que el software manifiesta los fallos en el testeo. Métricas: esfuerzo para encontrar fallas, probabilidad de fallas en el sig test, validación vs verificación.
- **Usabilidad:** facilidad para el usuario en realizar una tarea y el soporte que provee el sistema. Métricas: Tiempo de ejecución tarea x el usr, cantidad de errores, numero de tareas terminadas, satisfacción del usuario.
- **Variabilidad:** forma especial de modificabilidad, asociado a líneas de producto.
- **Portabilidad:** cantidad de equipos, SO o navegadores.
- **Distribución del desarrollo:** Globalización del desarrollo teletrabajo.
- **Escalabilidad:** Horizontal-vertical /cantidad de servidores-más RAM a uno en específico.
- Capacidad de despliegue: a como un ejecutable llega a la plataforma (ancho de banda, integración) ej. Actualizaciones de Windows, parches etc.
- Movilidad: se refiere al movimiento y alcance de una plataforma en cuanto a tamaño, tipo de display, tipo de dispositivo de entrada, uso de batería.
- Seguimiento: largo de colas, tiempo promedio de procesamiento, y salud de componentes, analitics.

Todos estos atributos de calidad cuestan dinero en infraestructura o en trabajo para asegurar la calidad del sistema.

Patrones de Arquitectura

Patrón de capas: Esta arquitectura se distingue por separar la lógica de negocios con la lógica de diseño, separar la capa de datos de la capa de diseño. Ej.: podría ser una capa de presentación, servicios, persistencia, base de datos, etc.

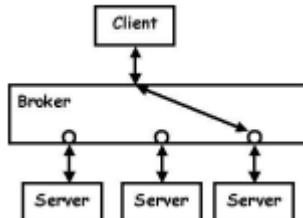
+Desarrollo se realiza por capas

+Se trabaja en niveles y el cambio solo afecta a la capa en cuestión.

El más utilizado es el de **Tres capas** anteriormente mencionado (presentación, negocio, datos).



Broker (Intermediario): Esta arquitectura es usada para estructuras distribuidas de software, para múltiples componentes que interactúan entre ellos, el “bróker” se encarga de mediar las consultas de un cliente y el servidor.



+ Manejo de fallas

+ No se necesita saber de dónde se sacan los datos.

+ Encapsula el monitoreo de bajo nivel (memoria, cantidad de servidores, etc)

+ Elasticidad

- único punto de falla

- Difícil testeo

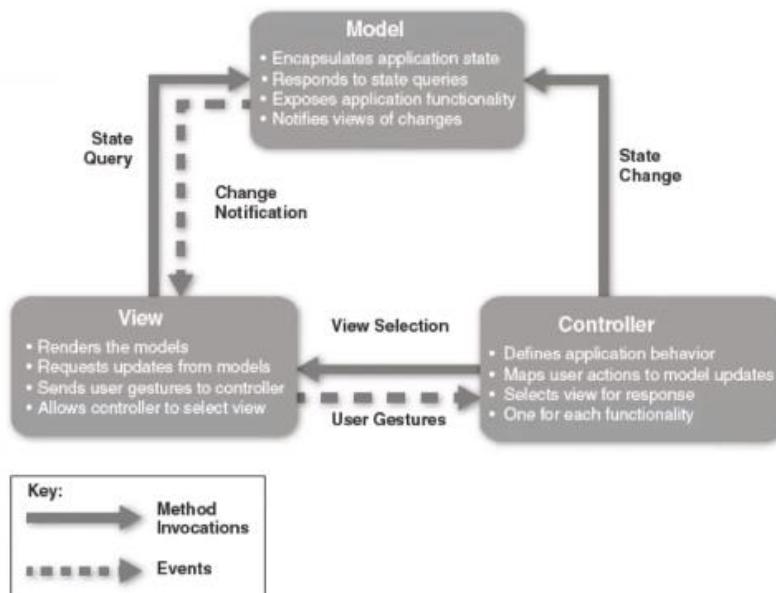
Modelo Vista controlador (MVC):

Mantiene la interfaz de usuario aparte de los controladores o la funcionalidad y los datos.

Modelo -> Datos de la aplicación

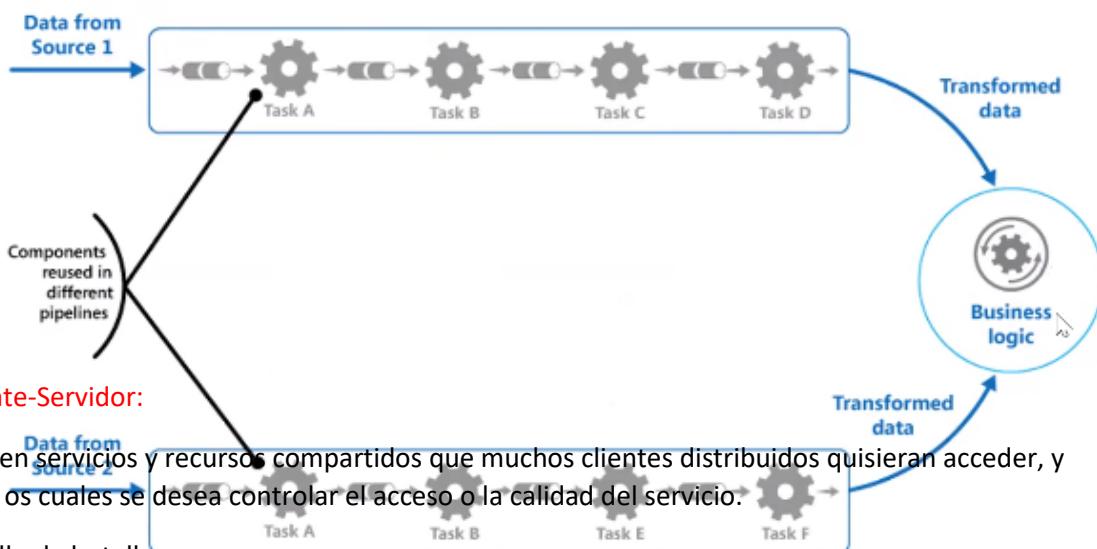
Controlador -> Mediador entre la vista y el modelo. Administra todo.

Vista -> Despliegue de datos e interacción del usuario



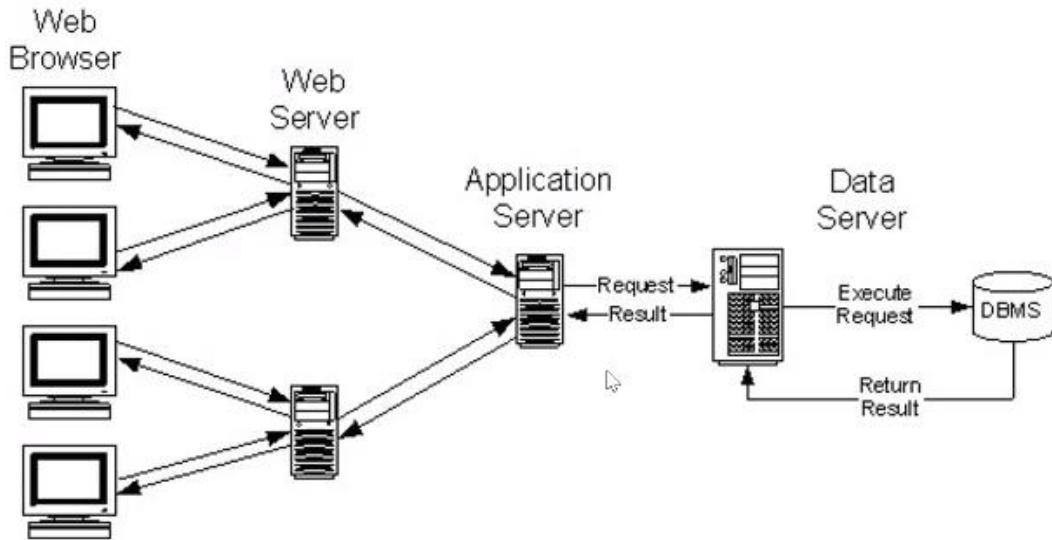
Pipe and Filters (Filtros y tuberías):

Se aplica cuando los datos de entrada se deben transformar excesivamente en datos de salida mediante una serie de operaciones (filtros) así los manda por diferentes tuberías donde se le aplican otros filtros. Etc. Se ocupan principalmente en redes agrícolas y sensores para separar sus datos..



-Punto único de fallas

-Poca escalabilidad



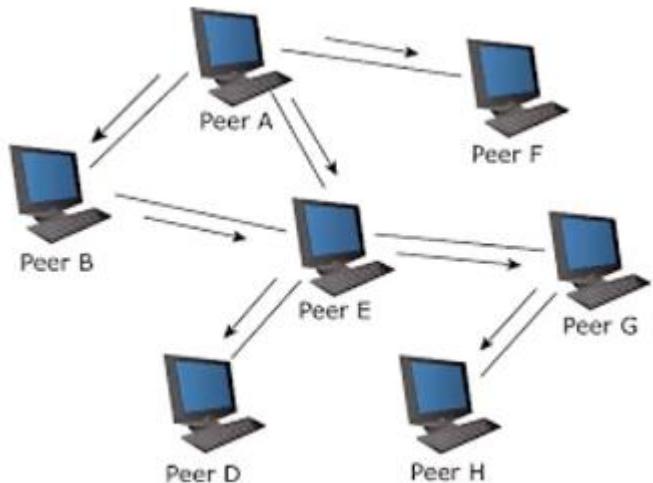
Peer to Peer: Sistema distribuido, una comunidad de hosts colaboran para sostener un servicio, administrando recursos. por lo que cada nodo de la red es un cliente y servidor al mismo tiempo.

+Escalabilidad

+Sin punto único de falla

+Sin servidor central

- Entrada-Salida de la red
- Seguridad
- Consistencia
- Tamaño



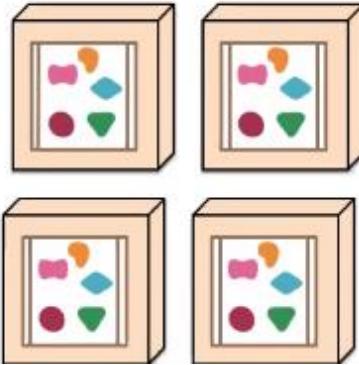
SOA: Existen proveedores de servicios que ofrecen servicios a consumidores de servicios. Los consumidores requieren entender y usar los servicios sin conocer detalle de la implementación. Su implementación más común es con Apis ya sea SOAP o Rest.

Microservicios: muy similar a SOA, pero su principal diferencia es que los servicios se autorreplican bajo demanda

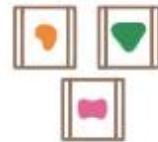
A monolithic application puts all its functionality into a single process...



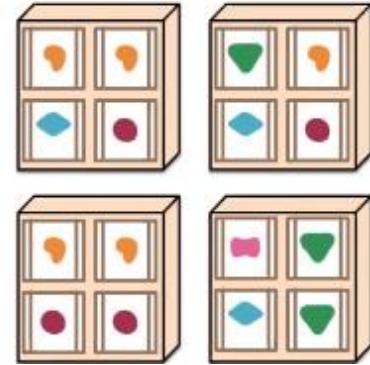
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



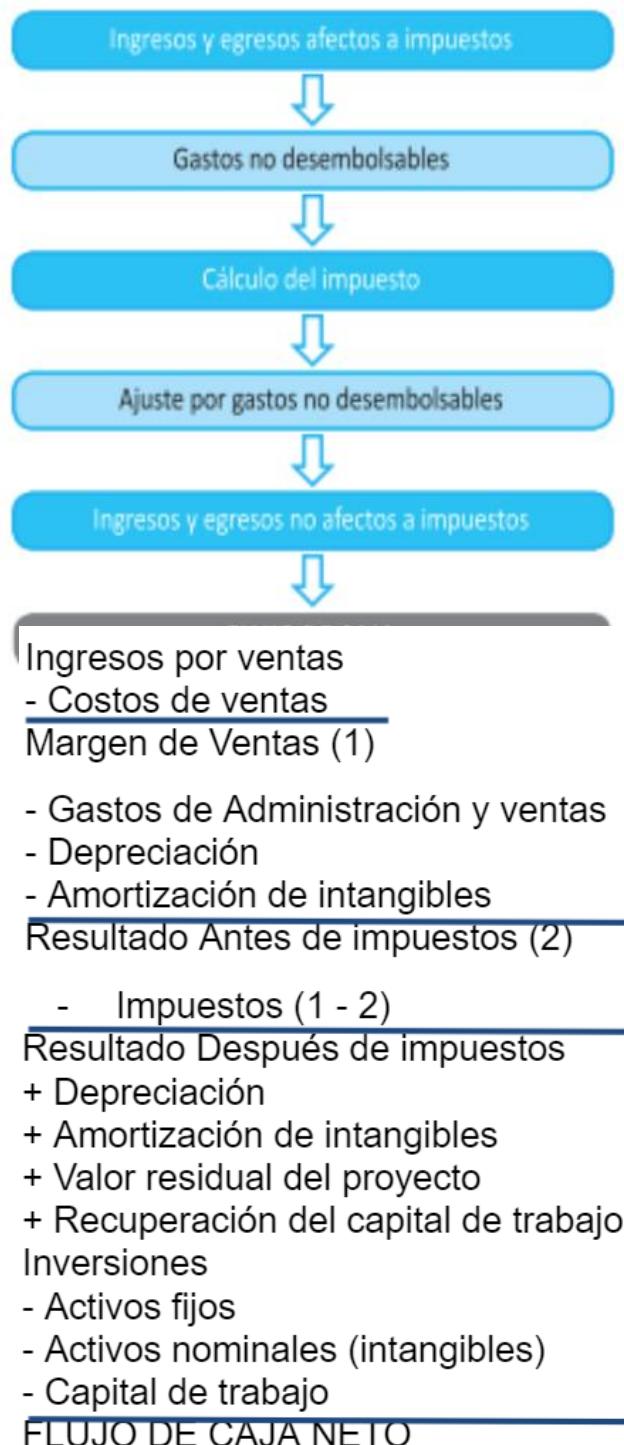
Correspondencia entre Tácticas y Patrones

Pattern	Modifiability								
	Increase Cohesion		Reduce Coupling				Defer Binding Time		
	Increase Semantic Coherence	Abstract Common Services	Encapsulate	Use a Wrapper	Restrict Comm. Paths	Use an Intermediary	Raise the Abstraction Level	Use Runtime Registration	Use Startup-Time Binding
Layered	X	X	X		X	X	X		
Pipes and Filters	X		X		X	X			X
Blackboard	X	X			X	X	X	X	X
Broker	X	X	X		X	X	X	X	
Model View Controller	X		X			X			X
Presentation Abstraction Control	X		X			X	X		
Microkernel	X	X	X		X	X			
Reflection	X		X						

Evaluación de proyectos

Flujo de caja

Un flujo de caja es una estructura de varias columnas que representan los momentos en que se generan los costos y beneficios de un proyecto (generalmente anual)



Evaluación Social

Ejercicios

Cursos de Redes

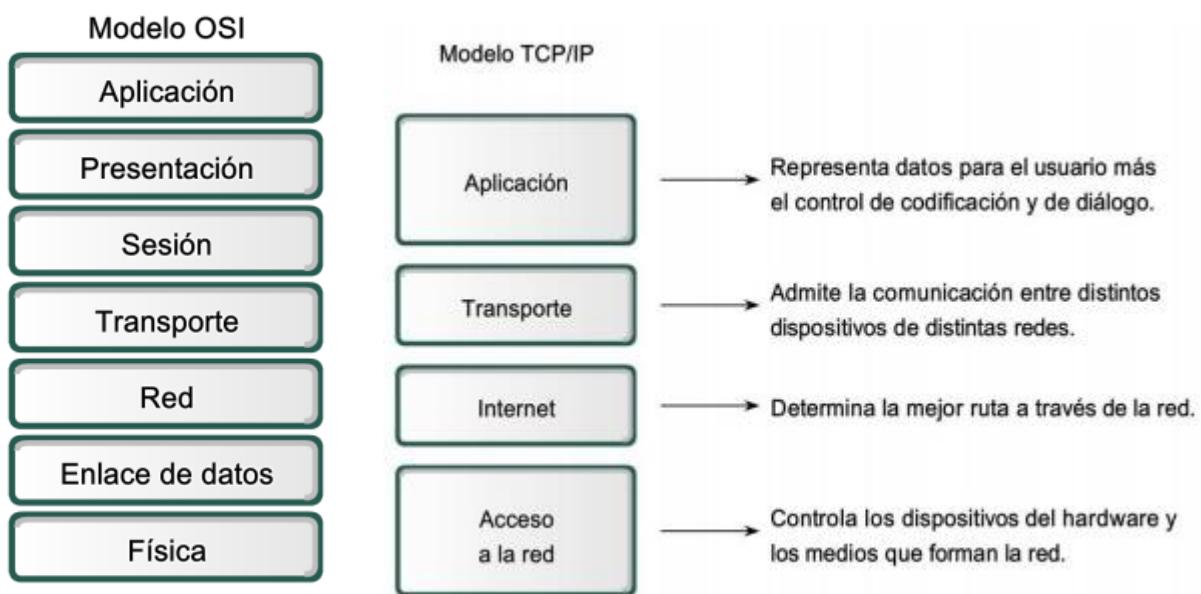
Redes de datos

Modelo OSI y TCP/IP:

Osi esta Definido en 7 capas

+ Divide la comunicación en capas simplifica el estándar

+Impide que los cambios en una capa afecten a otra.



Capa Física: Define las especificaciones

- **Eléctricas:** (niveles de voltaje, temporización de cambios de voltaje, velocidad de datos físicos, distancias de transmisión máximas).
- **Mecánicas:** (conectores físicos, configuración de los pines, longitud de cables, etc)
- **Procedimiento y funcionales:** para activar, mantener y desactivar el enlace físico entre sistemas finales

Capa de enlace:

- proporciona tránsito de datos confiable a través de un enlace físico
- Se ocupa del direccionamiento físico mac , la topología de red, el acceso a la red, la notificación de errores, entrega ordenada de frames y control de flujo

Capa de red: Proporciona conectividad y selección de ruta entre 2 sistemas de host que pueden estar ubicados en redes geográficamente distintas

Capa de transporte:

- La capa de transporte segmenta los datos originados en el host emisor y los reensambla en una corriente o flujo de datos (streaming) dentro del sistema del host receptor
- Responsable de la confiabilidad del transporte entre dos hosts
- Al proporcionar un servicio confiable, se utilizan dispositivos de detección y recuperación de errores de transporte

Capa de Sesión

- La capa de sesión establece, administra y finaliza las sesiones entre dos hosts que se están comunicando
- Sincroniza el diálogo entre las capas de presentación de los dos hosts y administra su intercambio de datos

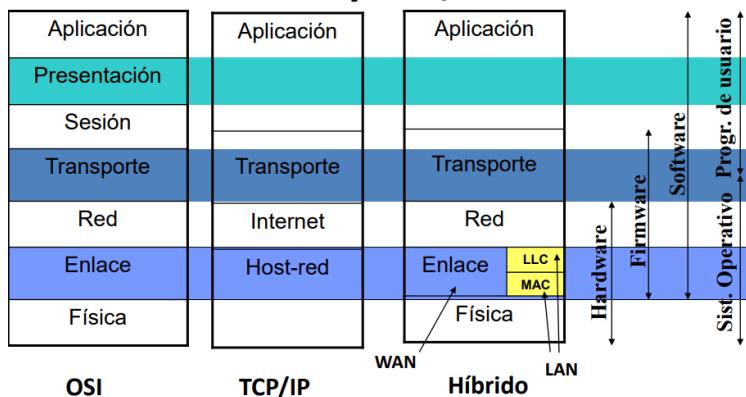
Capa de presentación:

- Convierte los datos de la red al formato requerido por la aplicación

Capa de Aplicación:

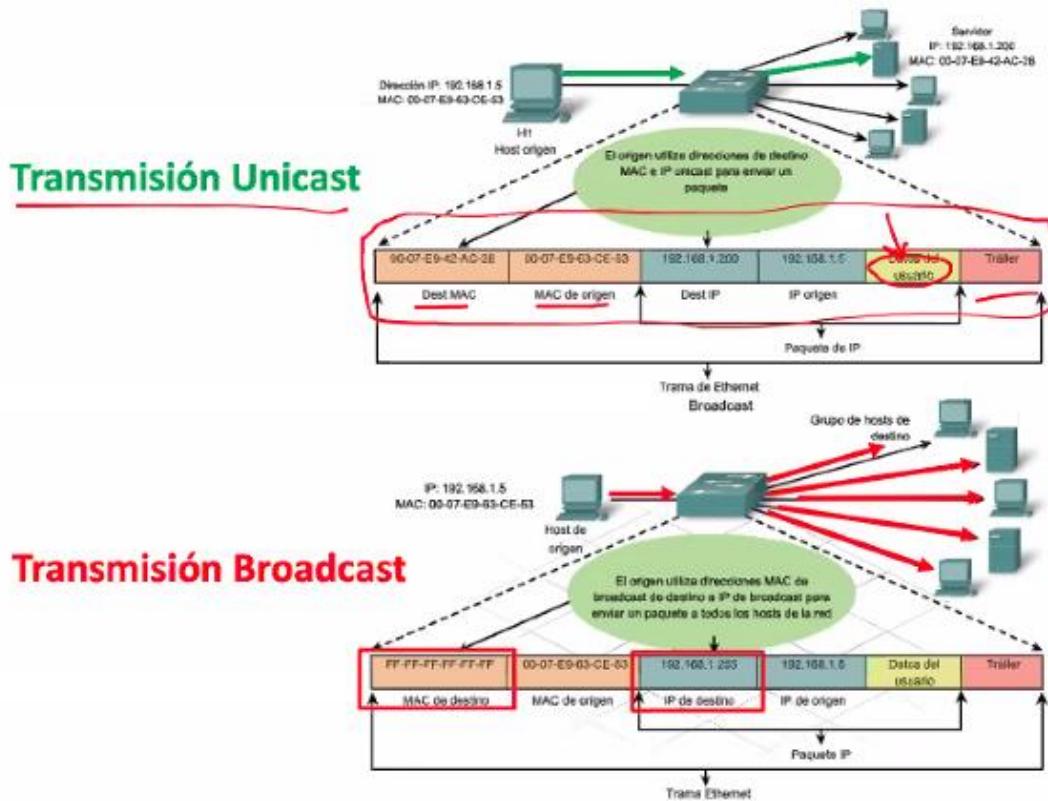
- Es la interfaz que ve el usuario final
- Muestra la información recibida
- En ella residen las aplicaciones
- Envía los datos de usuario a la aplicación de destino usando los servicios de las capas inferiores

Comparación Modelos de referencia OSI y TCP/IP



Frame Ethernet:

Mac origen-Macdestino-IPorigen-IPdestino-Puertos.



Dominio broadcast: Donde un paquete broadcast puede llegar, puede ser hasta un router ya que lo descarta, un Switch lo reenvía a todos los equipos de la red.

Dominio de colisión: se produce donde haya cables Half-duplex que se puedan compartir a múltiples equipos sin un switch por lo que generaría ruido en ambos sentidos.

Protocolo ARP (Address Resolution Protocol)

Es un protocolo para resolver la MAC del destino cuando tengo la IP de destino. A través de un ARP request a todos (broadcast) Quien tiene esta IP y al recibir la respuesta se obtiene la ARP replay dándole la MAC que le faltaba para enviar el primer frame. Esta MAC se guarda en una tabla ARP para enviar otros frames.

1. Mensaje ARP Request

Encabezado				Mensaje ARP
Encabezado MAC		Encabezado IP		¿Cual es tu dirección MAC?
MAC Destino	MAC Origen	IP Destino	IP Origen	
FF:FF:FF:FF:FF:FF	01:00:D1:B5:D4:F1	200.59.4.5	200.59.4.1	
Dirección MAC Broadcast		Dirección IP consultada		

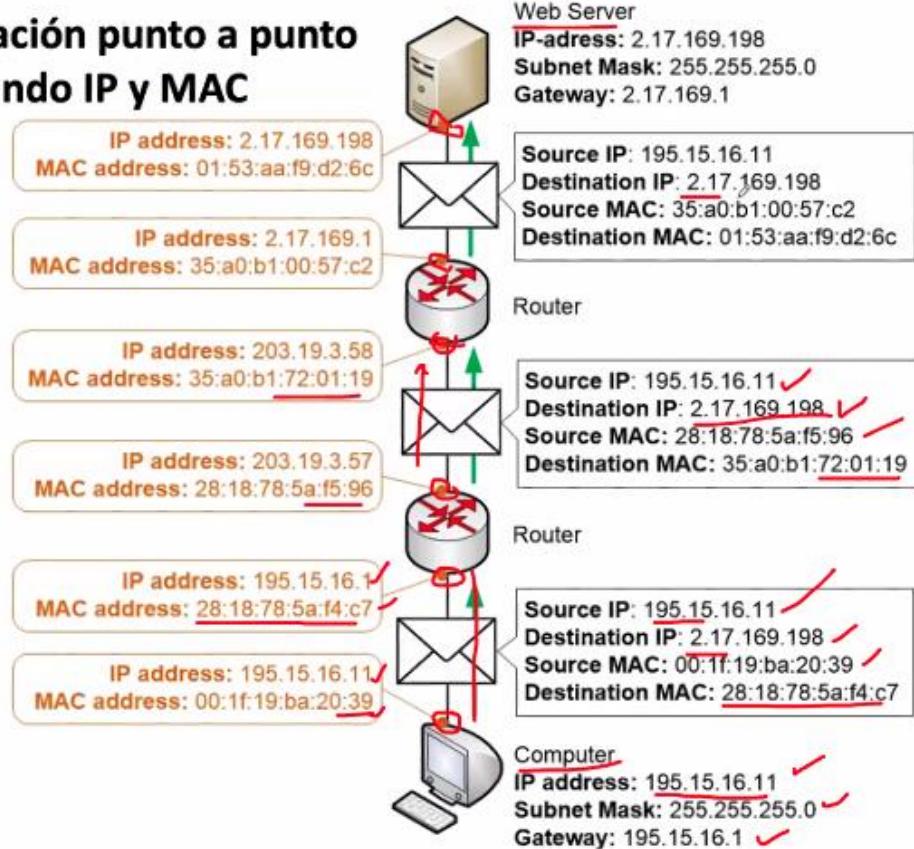
2. Mensaje ARP Reply

Encabezado				Mensaje ARP
Encabezado MAC		Encabezado IP		¿Cual es tu dirección MAC?
MAC Destino	MAC Origen	IP Destino	IP Origen	
01:00:D1:B5:D4:F1	F1:01:E1:B5:F4:14	200.59.4.1	200.59.4.5	

Conexión con un equipo o servidor que no esté en la red:

Se mantiene la IP origen y destino y las macs cambian por todos los pasos haciendo el protocolo ARP en cada una de las redes que se llegue.

Comunicación punto a punto usando IP y MAC



Redes Redundantes:

- Estructura de red jerárquica (Acceso, distribución y núcleo)
- Alta disponibilidad
- Redundancia (enlaces y equipos)
 - Mayor Costo
 - Mayor complejidad
 - Mayor disponibilidad

Spinning Tree Protocol

- Evita loops desactivando interfaces de los Switches
- Evita tormenta broadcast

- Evita inconsistencia tablas de switch
- Todos los Switches tienen contacto con el Root switch

Root switch: se determina a través del número de prioridad el menor es el root, si es el mismo se hace a través de las MAC de cada switch. Prioridad > MAC

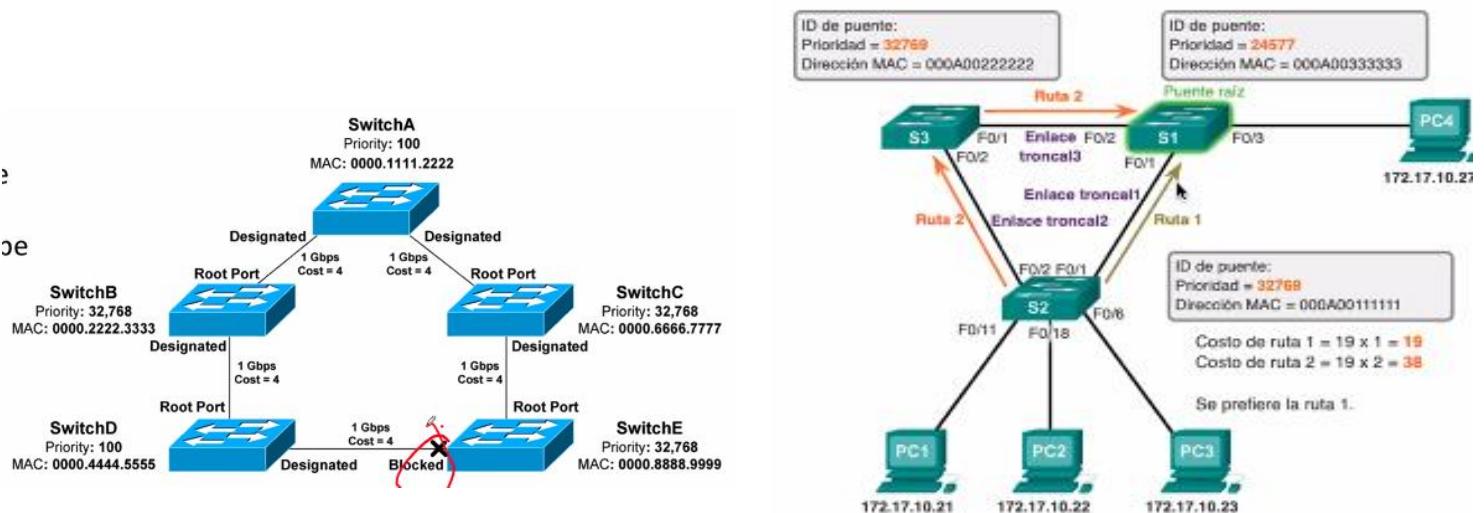
Prioridad de enlaces:

Velocidad de enlace	Costo (especificación IEEE revisada)	Costo (especificación IEEE anterior)
10 Gb/s	2	1
1 Gb/s	4	1
100 Mb/s	19	10
10 Mb/s	100	100

Puerto raíz (PR): puerto en dirección al puerto raíz

Puertos designados (PD): están habilitados

Puertos alternativos (PA): bloqueado



Vlans:

- es una partición lógica de una red de capa 2.
- Uno de los objetivos de una Vlan es agrupar dispositivos conectados a un switch en dominio de broadcast lógicos.
- Las Vlan permiten agrupar usuarios en dominio broadcast común, independientemente de su ubicación física en la red.
- Se pueden crear varias particiones para que coexistan varias VLAN.

- Las Vlan se aislan mutuamente y los paquetes pueden pasar entre ellas solamente mediante un Router o Switch de capa 3.
- Las ventajas de las Vlan son las siguientes:
 - Seguridad
 - Segmentación
 - Flexibilidad
 - Ahorro de dinero en Switch
 - Mejor rendimiento
 - reducción de dominios broadcast

Puertos troncales: puertos que tienen configurado el paso de varias Vlans

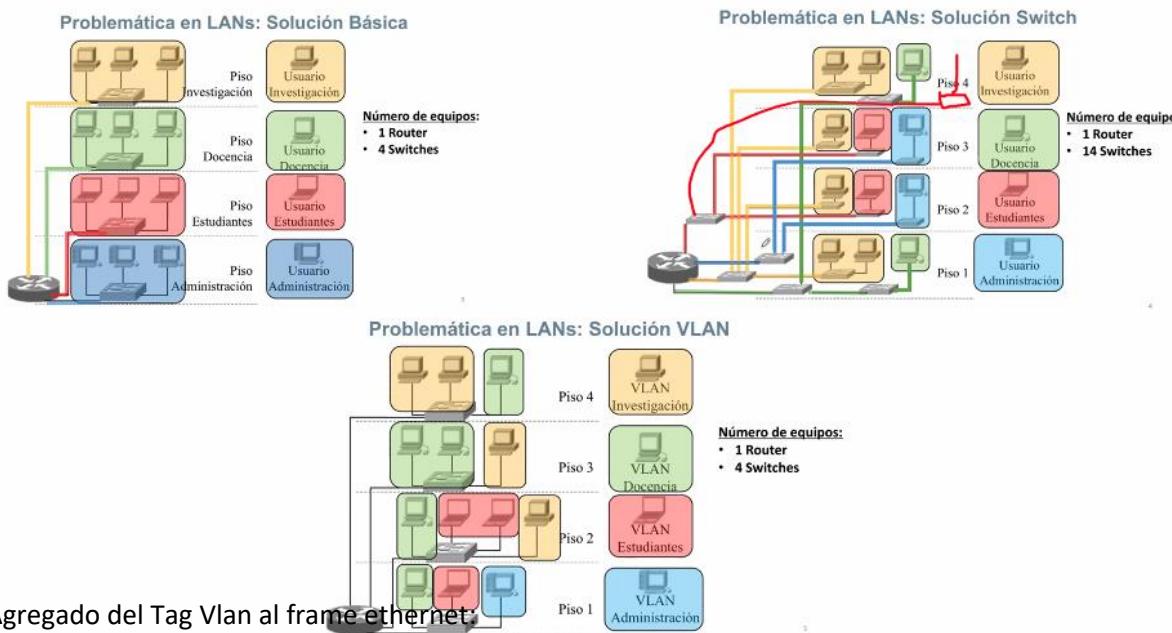
Tipos:

VLAN tipo 1 (Vlan basada en puerto)

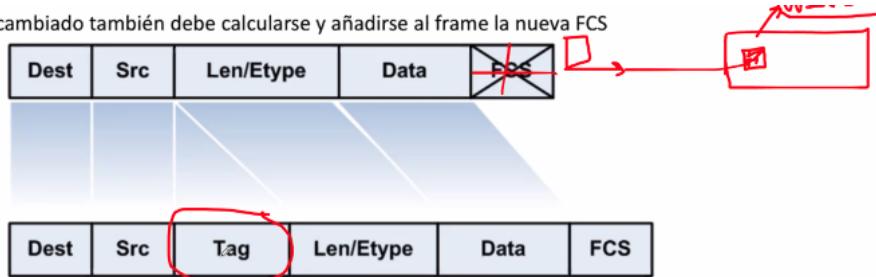
VLAN tipo 2 (Vlan basada en dirección MAC)

VLAN tipo 3:

- Vlan basada en dirección de red
- Vlan basada en protocolo



- Como el frame ha cambiado también debe calcularse y añadirse al frame la nueva FCS



- La etiqueta de 4 bytes se inserta en el frame después del campo origen y está compuesta por dos secciones de dos bytes cada una
 - TPID: ID del protocolo de etiquetas (p.e. 0x8100 para frames ethernet)
 - TCI: el campo con información de control de etiquetas. Los 3 primeros bits se conocen como bits de prioridad, indican la prioridad del frame por razones de QoS

14

Comunicación inter Vlans:

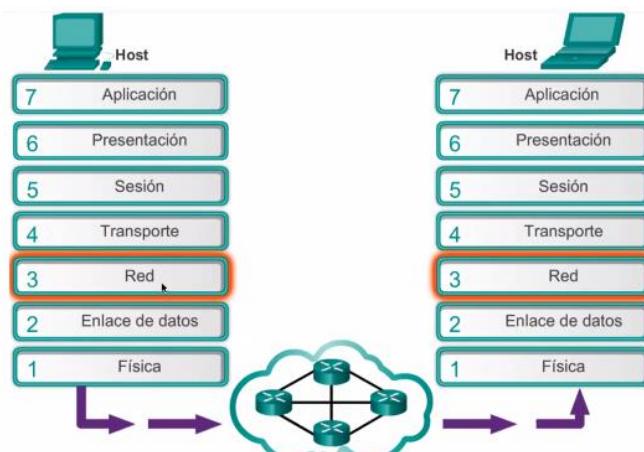
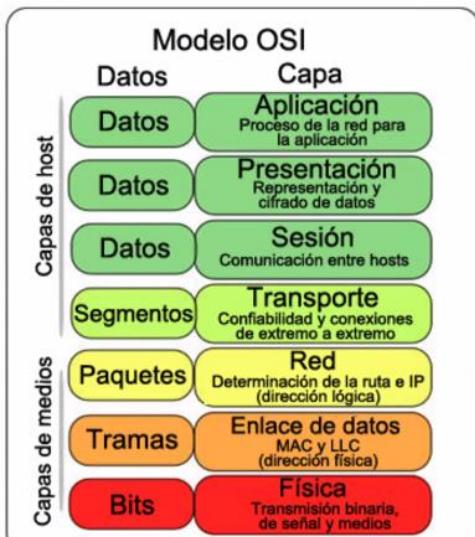
Routing inter Vlans:

- Múltiples enlaces:
- Interfaces virtuales:

IPV4:

Al trabajar TCP-IP juntos garantiza que los segmentos lleguen al destino, garantizando la conectividad.

Capa de red



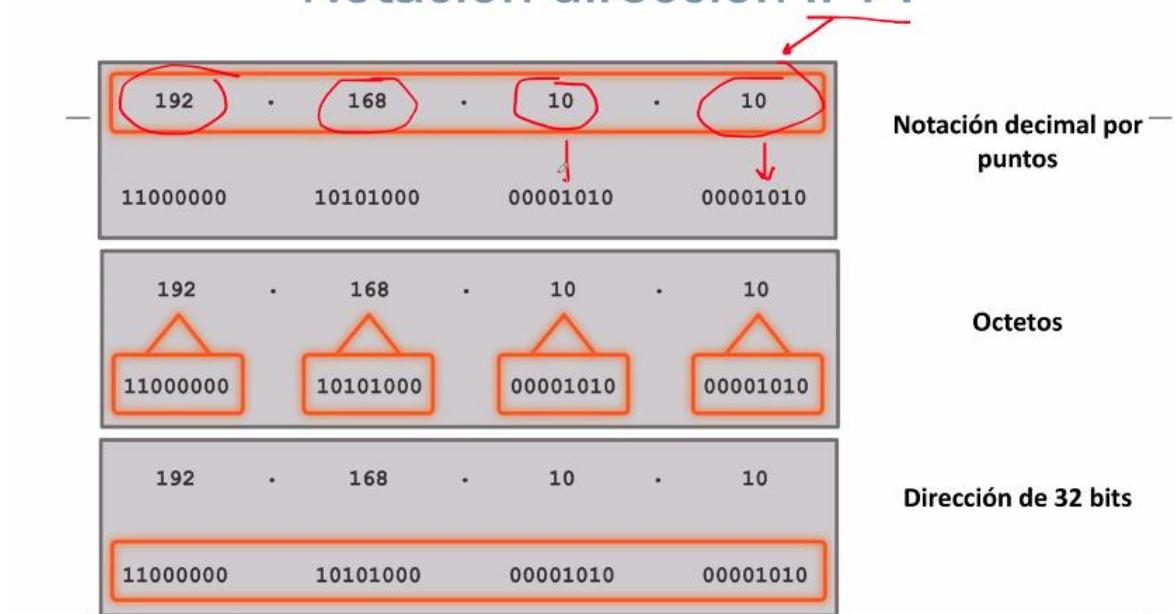
Los protocolos de capa de red reenvían las PDU de la capa de transporte entre hosts.

Procesos de transporte de extremo a extremo

- Direccionamiento de dispositivos finales
- Encapsulación
- Enrutamiento
- Desencapsulación

Notación dirección IPV4:

Notación dirección IPv4



Conversión binario - decimal

Raíz-Base	2	2	2	2	2	2	2	2
Exponente	7	6	5	4	3	2	1	0
Valores de bits de octeto	128	64	32	16	8	4	2	1
Dirección binaria	1	1	0	0	0	0	0	0
Valores de bits binarios	128	64	0	0	0	0	0	0

Direcciones IPv4 con clase o classful

Clase de dirección	Rango del 1er octeto (decimal)	Bits del primer octeto (los bits verdes no cambian)	Red (N) y Host (H) partes de la dirección	Máscara de subred predeterminada (decimal y binaria)	Formato prefijo	¿Cuántas redes?. ¿Cuántos hosts por cada red?
A	1-127**	00000000-01111111	N.H.H.H	255.0.0.0 /8		
B	128-191	10000000-10111111	N.N.H.H	255.255.0.0 /16		
C	192-223	11000000-11011111	N.N.N.H	255.255.255.0 /24		
D	224-239	11100000-11101111	No disponible (multicast)			
E	240-255	11110000-11111111	No disponible (experimental)			

Direcciones IP públicas y privadas:

Direcciones IP reservadas y privadas (RFC 1918)

Red o rango	Uso
127.0.0.0	Reservado (fin clase A)
128.0.0.0	Reservado (ppio. clase B)
191.255.0.0	Reservado (fin clase B)
192.0.0.0	Reservado (ppio. clase C)
224.0.0.0	Reservado (ppio. clase D)
240.0.0.0 – 255.255.255.254	Reservado (clase E)
→ 10.0.0.0	Privado clase A
→ 172.16.0.0 – 172.31.0.0	Privado clase B
→ 192.168.0.0 – 192.168.255.0	Privado clase C

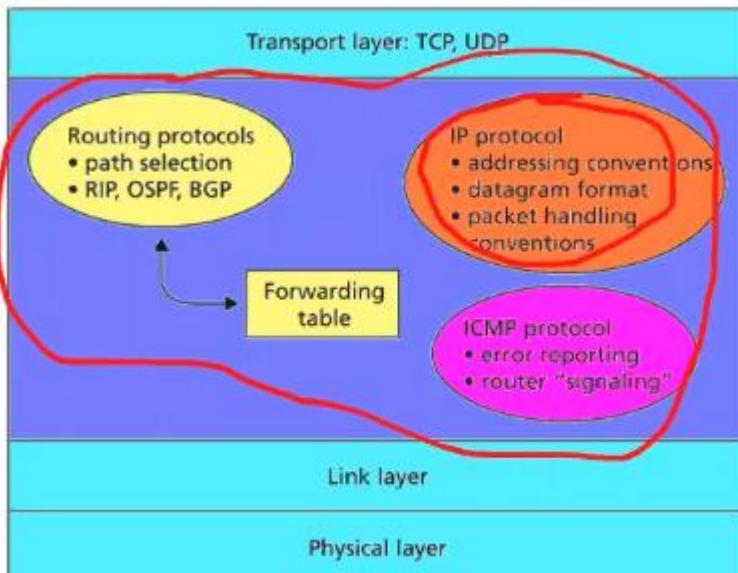
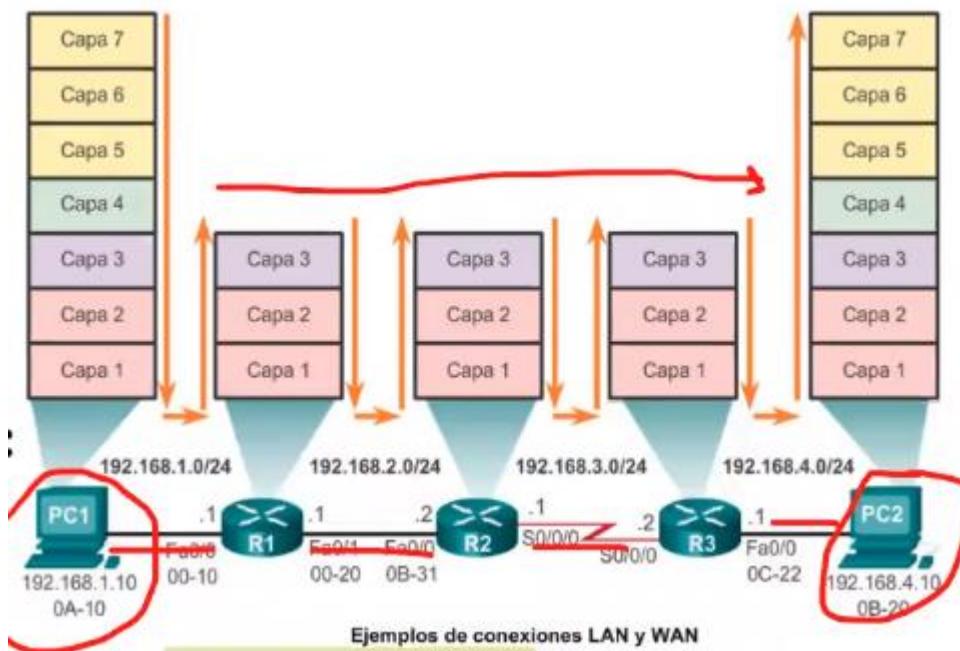
Comunicaciones IP pública – Privada (NAT):

Se hace un mapeo interno del Router a través de los puertos identificando todos los equipos al momento de la salida de un paquete a la WAN, por lo que cuando el paquete vuelva será identificado el destino privado por el puerto.

NAT dinámico:

NAT estático:

La capa de Red



Protocolos de ruteo

Los protocolos de ruteo sirven para direccionar un paquete a su destino, esto se hace a través de la información contenida en las tablas de ruteo de cada router, proveyendo a los router donde direccionar el próximo salto e ir formando una cadena de direccionamiento punto a punto para llegar a su destino.

Para ahorrar tiempo de direccionamiento se pueden ocupar varias métricas para el cálculo de la mejor ruta entre redes, estas métricas pueden ser:

- Capacidad del canal (BW)
- Retardo (Delay)
- Carga (Load)
- Confiabilidad o tasa de errores (Reliability)
- Número de Saltos (hop count)
- MTU

Si un router tiene 2 o más rutas con la misma métrica usa balanceo de carga entre ellas, enviando un porcentaje similar por ambas rutas.

La obtención del camino óptimo se debe hacer constantemente debido a que la red está constantemente en cambios.

1. PC1 envía un paquete a PC2



3. R2 reenvía el paquete a R3



2. R1 reenvía el paquete a PC2



4. R3 reenvía el paquete a PC2



Ruteo Estático o manual

Las decisiones se toman en base a conocimiento del administrador de red, es un proceso bastante tedioso debido a que se tiene que ir router por router configurando las redes a las que se desea llegar y la ip donde mandarla (sig. salto), no es escalable.

Enrutamiento estático

```
ip route 172.16.1.0 255.255.255.0 200.33.146.2  
ip route 200.33.147.0 255.255.255.252 200.33.146.2  
ip route 10.1.50.0 255.255.255.0 200.33.146.2
```

2021-07-14 19:05:26

Enrutamiento estático ruta predeterminada

```
ip route 0.0.0.0 0.0.0.0 200.33.147.1
```

Ruteo dinámico

Deciden la ruta optima obtenida en tiempo real en base a mensajes entre routers. Requieren un protocolo de ruteo, sirve para redes grandes debido a que solo se debe activar el protocolo de ruteo.

Algoritmos de ruteo

Vector Distancia (Bellman-Ford):

- Cada router conoce la distancia de sus vecinos directamente conectados
- El router envia una lista de actualizaciones de las redes que alcanza y su distancia
- Si todos los routers actualizan sus redes alcanzables con la info recibida habrá una convergencia en la tabla de ruteo.
- Ocupa menos tráfico.

Estado de Enlace (Dijkstra):

- Cada router conoce la distancia a sus vecinos directamente conectados.
- La información de distancia. Conocida como estado de enlace (**link state**) y es enviada por broadcast a todos los routers de la red.
- Todos los routers construyen la topología de la red
- Cada router calcula el camino mas corto a todas las redes alcanzables de manera independiente a su tabla de routing.
- Tiempo mas corto de convergencia (pero mas trafico usado en un principio).

Si se tienen 2 tipos de protocolos en 2 enlaces diferentes no se pueden comparar métricas, por lo que se define el concepto de **Distancia Administrativa** la cual se define según protocolo EIGRP=90 , RIP=120 , OSPF= 110

RIPv2

- Vector distancia ósea Ocupa Bellman-Ford
- Se utiliza para redes pequeñas (5-10) como máximo 15 saltos si son más es inalcanzable
- Intercambia tabla de routing a sus vecinos cada 30 s
- Rip v2 soporta subredes
- Soporta balanceo de carga de hasta 6 rutas (4 por defecto)

Gateway of last resort is not set					
I	192.168.4.0/24	[120/1]	via 192.168.2.2,	00:00:02,	Serial0/0/0
I	192.168.5.0/24	[120/2]	via 192.168.2.2,	00:00:02,	Serial0/0/0
C	192.168.1.0/24	is directly connected, FastEthernet0/0			
C	192.168.2.0/24	is directly connected, Serial0/0/0			
I	192.168.3.0/24	[120/1]	via 192.168.2.2,	00:00:02,	Serial0/0/0

EIGRP

- Usa vector distancia ósea ocupa Bellman-ford
- Protocolo propietario de Cisco
- Mezcla de métricas (BW,Delay, Reliability, Loading , MTU)
- Incrementa la escalabilidad de la red 100 saltos por defecto y 255 configurable
- Balanceo de carga hasta 6 rutas que pueden o no tener un mismo costo.(Para BW o redundancia)
- Cada 5 segundos se envían un saludo y luego se comparten las tablas

K Defaults	Metric Formula
$K_1 = 1$ $K_2 = 0$ $K_3 = 1$ $K_4 = 0$ $K_5 = 0$	$256 * (K_1 * bw + \frac{K_2 * bw}{256 - load} + K_3 * delay) * \frac{K_5}{rej + K_4}$ <p> $\cdot bw = 10^7 / \text{minimum path bandwidth in kbps}$ $\cdot delay = \text{interface delay in } \mu\text{secs} / 10$ </p>

$$([10\ 000\ 000/\text{ancho de banda}] + [\sum \text{de retrasos}/10]) * 256 = \text{Métrica}$$

Ethernet	1.000
Fast Ethernet	100
Gigabit Ethernet	10
Token Ring 16 M	630
FDDI	100
T1 (serial predeterminada)	20 000
DS0 (64kb/s)	20 000
1024 kb/s	20 000
56 kb/s	20 000

OSPF (Open Shortest path first)

- Estado del enlace ósea ocupa Dijkstra
- Tiene 2 niveles Jerarquicos (areas)
- Area 0 o backbone (obligatoria)
- Areas adicionales (opcionales). OSPF multiareas
- Soporta máscaras de tamaño variable
- Metricas más complejas
- Múltiples rutas
- Muy escalables debido a la topología
- Se sobrecarga el área 0 ()



Costo = ancho de banda de referencia / ancho de banda de la interfaz
 (el ancho de banda de referencia predeterminado es 10^8)
Costo = 100 000 000 bps / ancho de banda de la interfaz en bps

Tipo de interfaz	Ancho de banda de referencia en bps	Ancho de banda predeterminado en bps	Costo
10 Gigabit Ethernet 10 Gbps	100,000,000	\div 10,000,000,000	1
Gigabit Ethernet 1 Gbps	100,000,000	\div 1,000,000,000	1
Fast Ethernet 100 Mbps	100,000,000	\div 100,000,000	1
Ethernet 10 Mbps	100,000,000	\div 10,000,000	10
Serial 1.544 Mbps	100,000,000	\div 1,544,000	64
Serial 128 kbps	100,000,000	\div 128,000	781
Serial 64 kbps	100,000,000	\div 64,000	1562

El mismo costo debido al ancho de banda de referencia

Costo acumulado

Protocolos de Routing interior y exterior (BGP)

Interior (IGP):

SE utiliza para el rputing de un Sistema autónomo, que pueden estar configurados con RIP, OSPF, EIGRP e ISIS

Exterior (EGP):

Se utilizan como protocolos de ruteo inter sistema autónomo, comunicando SA que pueden tener configurados diferentes protocolos de ruteo.

Capa de Transporte

La capa de transporte es responsable de establecer una sesión de comunicación temporal entre 2 aplicaciones y de transmitir datos entre ellas. TCP/IP utiliza dos protocolos para lograrlo.

- Protocolo de control de trasmisión (TCP) : Segmentos
 - Confiable
 - Acuse de recibo ACK
 - Reenvio de datos
 - Entrega los datos en un orden concreto
- Protocolo de datagramas de usuario (UDP): Datagramas
 - Rápido
 - Menor carga
 - No requiere acuses de recibo
 - No reenvía datos perdidos

Responsabilidades:

Rastreo de comunicación individual entre aplicaciones en los hosts de origen

División de los datos en segmentos para su administración y reunificación de los datos segmentados en streams de datos de aplicación en el destino.

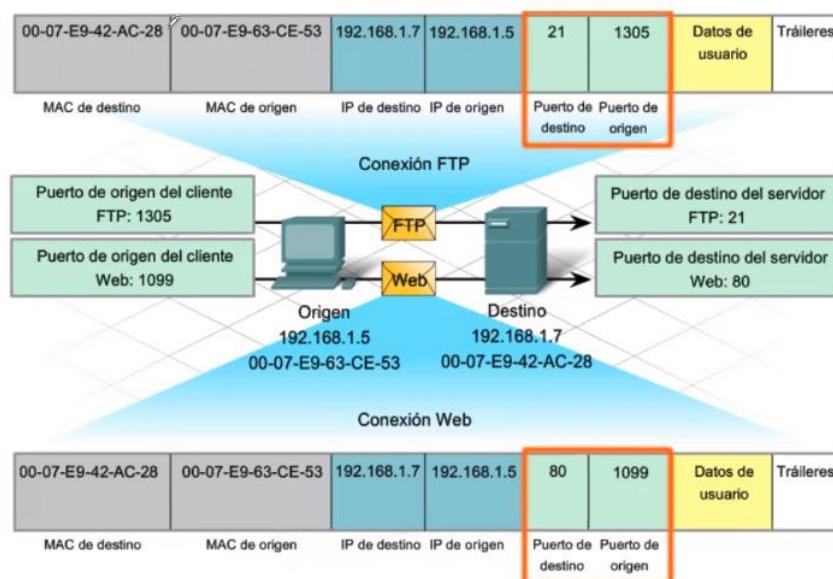
Identificación de la aplicación correspondiente para cada stream de comunicación.

La capa de aplicación se comunica con la de transporte a través de los puertos, estos puertos se dividen en 3 rangos:

0-1023 : conocidos servicios de forma estática

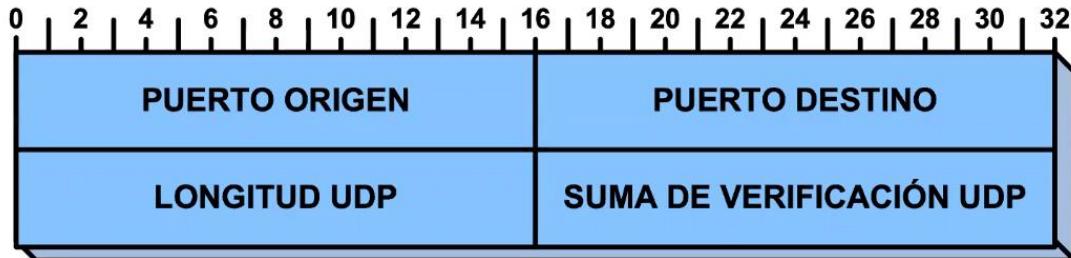
1024-49151: registrados. Se utilizan para múltiples propósitos.

49152-65535: Puertos dinámicos y privados, no se les debe asociar servicios.

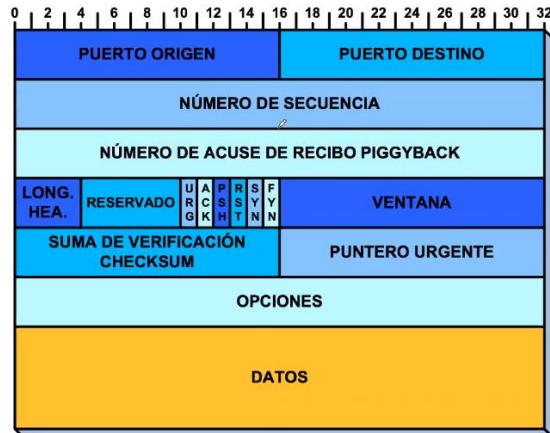


Cabecera UDP:

- **Puertos de Origen y Destino:** identifican los extremos de la conexión.
- **Longitud:** número total de bytes en el datagrama UDP, incluyendo la cabecera y los datos
- **Checksum o suma de verificación UDP:** es opcional y se rellena con ceros cuando no se quiere usar.



Cabecera TCP:



Cabecera del Protocolo de control de transmisión (TCP)

- Segmento de tamaño mínimo **20 Bytes**
- Tamaño máximo de datos **65.495 Bytes**
- Puerto de origen y destino (16 bits):** Puerto de la máquina origen. Al igual que el puerto destino es necesario para identificar la conexión actual
- Número de secuencia (32 bits):** Indica el número de secuencia del primer byte que transporta el segmento.
- Número de acuse de recibo (32 bits):** Indica el número de secuencia del siguiente byte que se espera recibir. Con este campo se indica al otro extremo de la conexión que los bytes anteriores se han recibido correctamente

- **Ventana (16 bits):** Número de bytes que el emisor del segmento está dispuesto a aceptar por parte del destino
- **Suma de verificación (16 bits):** Suma de comprobación de errores del segmento actual. Para su cálculo se utiliza una pseudocabecera que también incluye las direcciones IP origen y destino
- **Puntero de urgencia (16 bits):** Se utiliza cuando se están enviando datos urgentes que tienen preferencia sobre todos los demás e indica el siguiente byte del campo Datos que sigue a los datos urgentes. Esto le permite al destino identificar donde terminan los datos urgentes. Nótese que un mismo segmento puede contener tanto datos urgentes (al principio) como normales (después de los urgentes)
- **Opciones (variable):** Si está presente únicamente se define una opción: el tamaño máximo de segmento que será aceptado
- **Datos:** Información que envía la aplicación

Se pueden dar problemas de flujo con TCP debido al tamaño del buffer de recepción o el tamaño del canal, por lo que se debe adaptar a través de partida lenta, probando datos de a poco para estabilizar el flujo de forma que se adapte a las condiciones.

Cursos de telecomunicaciones

Comunicaciones Digitales

Ejercicios

Cursos de ciencias de la computación

Estructura de datos y algoritmos

Ordenes de complejidad

Los órdenes de complejidad hacen referencia al costo computacional del algoritmo, están dispuestos en 3 medidas O , Ω , Θ .

Estas medidas se refieren a $f(n)$ donde n representa el tamaño de algún elemento del problema a resolver mediante el algoritmo.

- Superior asintótica (O): Se utiliza para encontrar la función del peor caso.
- Cota inferior asintótica (Ω): Se utiliza para calcular el mejor caso para los algoritmos
- Cota ajustada asintótica (Θ): Es la función promedio de un algoritmo.

Estructuras fundamentales

Listas

- Ligada o enlazada: contiene el dato y la información del sucesor, pero no del antecesor, el último miembro de la lista apunta a null.
- Dblemente Ligada: Contiene el dato y la información del sucesor y del antecesor, el último apunta a null y el primero(root) apunta también.
 - Circular: Es una lista doblemente enlazada, pero el root apunta al último y el último apunta al root.
 - Indexada (Array): Es un arreglo porque cada miembro tiene un índice.
- Hash (lista asociativa):
- Pila -> LIFO (Last in First Out)
- Cola -> FIFO (First in First Out)

Operaciones

Lista: prepend(x), append(x), remove(x), head(), tail(), get(i)

Pila: push(x), pop(), peek()

Cola: enqueue(x), dequeue(x)

Tipos especiales:

Lista: circular
Cola: priorizada (por algún parametro), deque (doubled ended queue)

Tablas de Hash (key,value)

-Son diccionarios asociativos respecto de una llave y un dato vinculado.

-La asociación viene dada por una función de hash que recibe como parámetro el dato y entrega una llave (Hash (value)= key)

- Open addressing (direcciónamiento abierto a cada valor)
- Chaining (entrada de valores puede tener una lista ligada de valores para una misma key)

Grafos

-Es una estructura de datos que surge de la conexión (aristas) entre vértices. Tiene varios usos y además tiene una cantidad enorme de algoritmos relacionados. Se denota como $G(V,E)$.

- De hecho hay toda una teoría que se crea en torno a ello “Teoría de grafos”

Elementos del grafo:

vértices: se refiere a aquellos puntos o nodos del grafo. Pueden contener datos o no.

Aristas: son conexiones que comunican a un par de vértices. Pueden tener valores asociados (como costos) o no. Asimismo, las aristas pueden tener direccionalidad.

Se pueden representar con:



Aplicaciones de los grafos

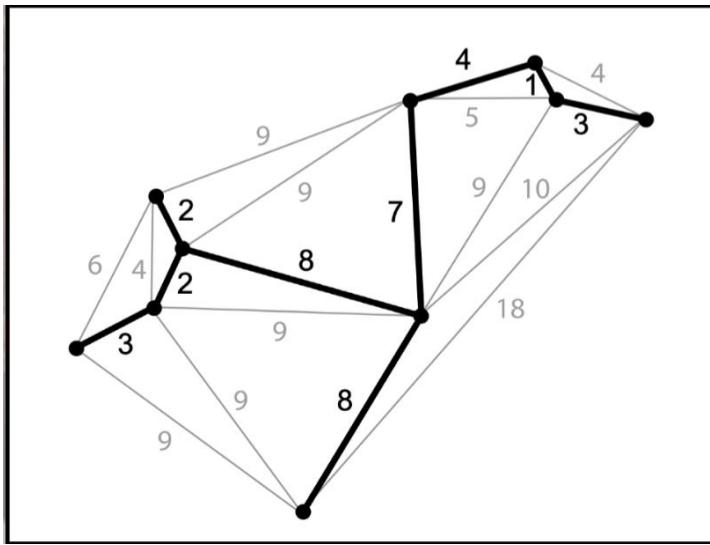
Se pueden utilizar como mapas para establecer localizaciones y rutas que las conectan (esto se puede)

Algoritmos de grafos

Árbol de cobertura mínimo

Un árbol de cobertura está formado por los vértices de un grafo y un subconjunto de sus aristas. Es una configuración acíclica que permite transitar de cualquier vértice a cualquier otro en el grafo.

Se debe notar que estos arboles se construyen sobre grafos no dirigidos y que tienen costos asociados a sus aristas. Asimismo, se tratan de grafos conexos.



Adicionalmente, al tratarse de un árbol de cobertura mínimo, denota la configuración que cumple y es de menor costo.

Algoritmo de Prim

A partir de un grafo (G):

- Se selecciona un vértice V1 al azar
- Incluirlo en el conjunto de vértices visitados (ST)
- Mientras se hayan visitado todos los vértices en ST con uno no-visitado (V2)
- Elegir la arista de menor costo e incluir a V2 en ST.

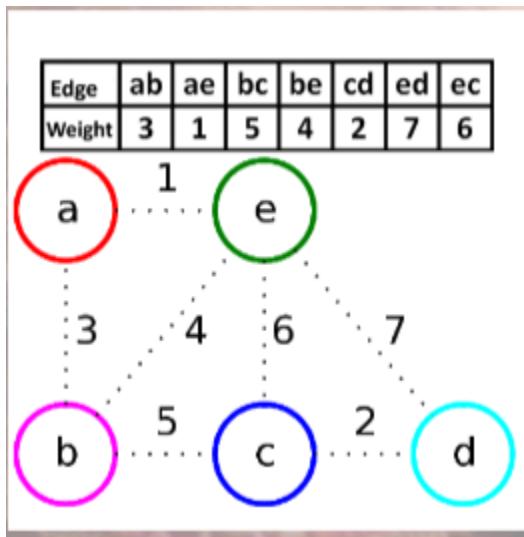
GIF: <https://enacademic.com/dic.nsf/enwiki/34313>

Algoritmo de kruskal

A partir de un grafo G:

- Ordenar el conjunto de las aristas E por costo
- Seleccionar e incluir la arista de menor costo en el conjunto (e1) de aristas presentes (ST) y registrar los vértices alcanzados
- Mientras no se haya alcanzado todos los vértices en V:
 - Seleccionar la siguiente arista de menor costo que no forme un ciclo (e2) con las aristas en ST

- Agregar e2 a ST.



GIF: <https://sahebg.github.io/computersceince/minimum-spanning-tree-Kruskal-algorithm-c-program-example/>

Algoritmo de Rutas de costo mínimo

Se trata de un algoritmo se vean como elementos de transporte y cada vértice es una localidad y la arista es el costo involucrado en desplazarse entre los vértices asociados

Existen 2 tipos de algoritmos de costo mínimo que se utilizan para los cálculos de rutas optimas:

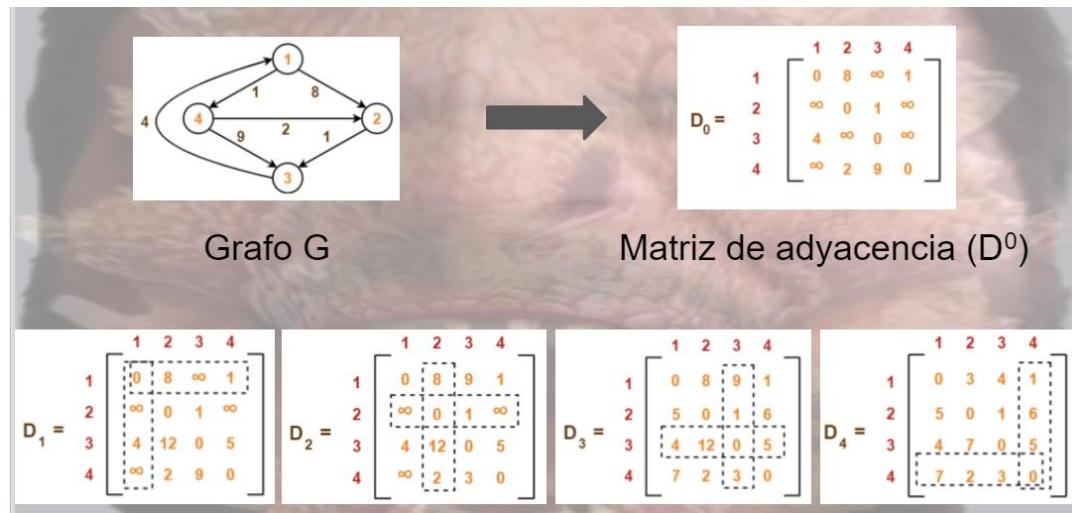
Floyd-Warshall

- Se trata de algoritmo de programación dinámica que calcula los costos de las rutas optimas entre todos los vértices del grafo. (all-pairs shortest path).
- Trabaja sobre grafos dirigidos
- Obtienen finalmente los costos de todas las mejores rutas entre cualquier par de vértices del grafo.

A partir de un grafo G:

- Definir una matriz D^0 que corresponde a la matriz de adyacencia de G. Si no existe una ruta directa entre 2 vértices, colocar la posición en cuestión como ∞ .
- Para todos los vértices V_k , construir una nueva matriz D^k , donde todos los elementos de esta nueva matriz se definen de acuerdo a la siguiente expresión:

$$Dk(i,j) = \min(Dk + 1(i,j), Dk - 1(i,k) + Dk - 1(i,k))$$



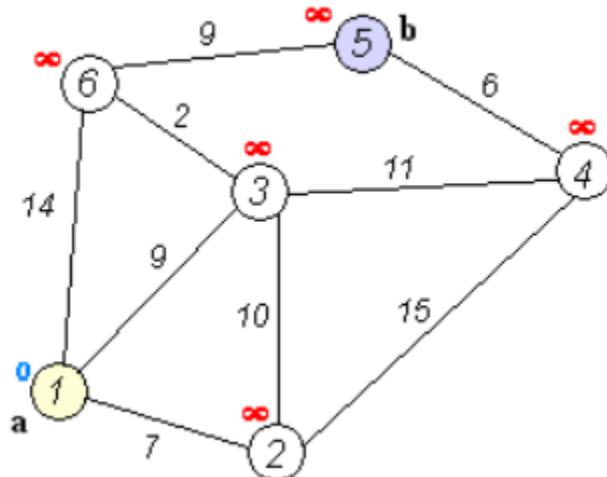
Fuente: <https://www.gatevidyalay.com/floyd-warshall-algorithm-shortest-path-algorithm/>

Dijkstra:

- Calcula los costos de todas las rutas mínimas desde un vértice de origen a todos los otros vértices del grafo G.
- No admite aristas de costo negativo.
- Trabaja sobre grafos dirigidos

A partir de un grafo G:

- Se define la matriz de adyacencia al igual que en el algoritmo de Floyd-Warshall (D^0).
- Se crea un vector de costos C de cardinalidad igual a V. Se inicializa este vector con todos los valores en ∞ .
- Se crea un vector (initialmente vacío) de vértices visitados por W.
- Se selecciona el vértice de origen O y se coloca $C(O)=0$
- Mientras no hayan visitado todos los vértices:
 - Seleccionar el vértice S que no este en W de menor costo ($C(S)$) a algún vértice W.
 - Para cada vértice N vecino de S, $C(N)=\min(C(N), C(S) + D^0(S,N))$.
 - Agregar S a W



Fuente: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

Arboles

ABB (árbol binario de búsqueda)

AVL (árbol binario que se balancea solo en la inserción):

- Son arboles binarios que se auto balancean.
- El concepto balanceo se da en relación con la altura, el balanceo se da por la diferencia de altura entre el subárbol izquierdo y el derecho, esta diferencia tiene que ser máximo 1.
- Sabemos que los árboles binarios tienen la característica de lograr operaciones en tiempo O ($\log n$), si n es el numero de elementos de un árbol.

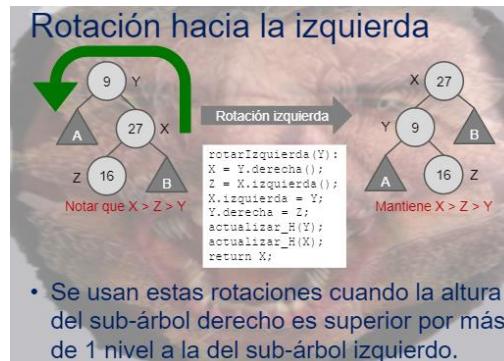
Sin embargo, esa complejidad no es garantizada, pues en casos malos podría a ser O(n), para mantener la complejidad O ($\log n$) el árbol debe estar balanceado.

Este balanceo se hace a partir de operaciones llamadas rotaciones los cuales son movimientos de los nodos existen 2 tipos.

- Rotación hacia la derecha:



- Rotación hacia la izquierda:



Para casos más complejos se pueden combinar ambas rotaciones para lograr el balanceo

Rotaciones dobles:



Inserción: Es igual al árbol binario de búsqueda, pero usa un rebalanceo sobre la rama que se coloco el nuevo nodo.

Borrado: Para borrar el nodo primero se necesita buscarlo y encontrarlo:

- Si es una hoja, no se reemplaza.
- Si tiene un solo hijo, se reemplaza por ese hijo.
- Si tiene 2 hijos, se reemplaza por el descendiente derecho de mas a la izquierda y recursivamente se borra.
- Finalmente se balancea hacia arriba para preservar el AVL.

Búsqueda: se realiza de la misma forma que el árbol binario de búsqueda:

Árbol Heap:

- Árbol binario aplanado a un arreglo
- Los hijos del elemento i son elementos $2i+1$ y $2i+2$ (relación entre los valores)
- Max-heap: nodo padre mayor que sus 2 hijos, se cumple en sub-arboles
- Min-heap: nodo padre es menor que sus 2 hijos, se cumple para sub-arboles
- Es una implementación eficiente de colas priorizadas

Algoritmos de Ordenamiento

Algoritmo	Detalle	O	Θ	Ω
BubbleSort	Por cada	$O(n^2)$	$\Theta(n^2)$	$\Omega(n^2)$
InsertSort	dad	$O(n^2)$	$O(n^2)$	$\Omega(n)$
QuickSort	SE ocupa un	$O(n^2)$	$\Theta(n \log(n))$	$\Omega(n \log(n))$
MergeSort	Corta en 2 hasta que quede lo mas pequeño y hace Merge para ordenar.	$O(n \log(n))$	$\Theta(n \log(n))$	$\Omega(n \log(n))$
HeapSort	dad	$O(n \log(n))$	$\Theta(n \log(n))$	$\Omega(n \log(n))$

Recorridos y búsqueda

Breadth find system BFS: búsqueda en anchura

Depth find system DFS: búsqueda en profundidad

Parecida al backtracking en un árbol

Se puede quedar en un cliclo

Backtracking: concepto general según el que vuelvo a la última opción posible para explorar una nueva alternativa de solución.

Bases de datos

características de una BD

¿Qué se entiende por una BD?

Una BD es un conjunto de datos relacionados el cual es agrupado o estructurado de algún modo. Este conjunto se crea, almacena, actualiza y/o consulta para registrar y/o inferir información.

¿En qué contexto existen las BD?

Hoy en día, las BD son fundamentales en casi cualquier ámbito. Son un medio de almacenamiento masificado en la industria. Pueden usarse para muchas cosas, desde almacenar datos para un servidor de juegos en la casa, pasando por un registro de ventas de alguna empresa hasta apoyo a la toma de decisiones.

¿De qué se componen las BD?

Las BD en sí son sólo datos ordenados y almacenados. No obstante, generalmente cuando nos hacemos esta pregunta generalmente nos referimos al sistema como un todo (SABD: Sistema Administrador de Bases de Datos o SGBD: Sistema de Gestión de Bases de Datos). Al referirnos al SABD, tenemos entonces varios tipos de lenguajes además de los datos en sí.

¿Qué propiedades tienen las BD?

Las BD pueden contener distintos volúmenes de información, de acuerdo con su necesidad. Además, como veremos en el curso, las BD pueden tener distintos grados de normalización, lo que influye directamente en el desempeño y rigurosidad del modelo presentado. Además, las BD deben ser consistentes.

Se pueden resumir las propiedades en ACID:

- Atomicidad: Que no se interrumpa una operación en curso.
- Consistencia: Que un int sea siempre un int no pueda ser string
- Aislamiento: Los datos no dependen de nadie mas, se definen x si solos
- Durabilidad: Persistencia que no cambien en el tiempo x si solos

¿Cómo uso una BD?

Una BD tiene tantos usos distintos que casi no se pueden enumerar. No obstante, el manejo de la BD se basa esencialmente en la creación, actualización y consulta de ella.

Esto se logra por medio de un lenguaje con el cual tendremos contacto este semestre, SQL. En esto nos concentraremos mucho este semestre, interacción con BD.

Ahora tenemos más o menos una idea de lo que es una BD.

Como mencionamos anteriormente, una BD se puede consultar o actualizar, pero además se puede administrar y efectuar operaciones más complejas sobre ella.

Hay diversas formas de interactuar con una BD. Esto va a depender esencialmente de qué tipo de usuario seamos.

Modelo de BD jerárquicas

Son aquellas BD que están organizadas como un árbol invertido, de ésta forma jerarquizando la información. Tienen el problema de no poder representar fácilmente redundancia de datos.

Modelo de BD de red

Es un modelo de BD que de cierto modo extiende el modelo jerárquico en el hecho que un nodo hijo pueda tener varios nodos padres. El gran problema de este tipo de BD es que es muy difícil de administrar.

Modelo de BD relacional

Es el modelo de BD más corriente hoy en día. Consiste en el almacenamiento de información de forma ordenada y relacionada entre sí. Trataremos este modelo en profundidad en este curso.

Modelamiento de las bases de datos

¿Qué ocurriría si después de implementada la BD nos damos cuenta que nos faltan datos? ¿O si duplicamos información? (De forma indeseada)

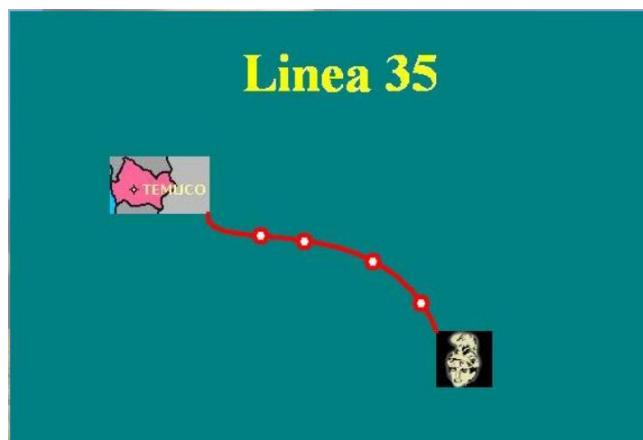
Es por eso recomendable que desde un inicio se siga un proceso o metodología precisa para la confección de una BD.

¿Qué entendemos por modelar?

Modelar se refiere a plasmar el problema específico que necesitamos resolver en términos abstractos, de modo que éstos sean representables y luego manejables. En el caso de una BD, se puede hablar de modelar el problema como una representación de los requerimientos de información del mismo.

Por ejemplo, deseo manejar información de la línea 35 del Pretonsporte la cual está en construcción y que une Temuco con Pretoria.

En un primer tiempo, probablemente me interesaría, por ejemplo, manejar información acerca de los horarios y recorridos del tren.



Ahora bien, supongamos que por órdenes del Gerente General de Pretonsporte, el señor Pretoriano, el Estratega, se requiere mantener información de los torniquetes y cajas en las estaciones.

Los datos involucrados cambian sustancialmente, así como el modelo que soporta estos datos.

Quizás si manejamos esa información “al lote” se nos puede pasar algún dato importante, y que luego repercutirá en falta de información o insuficiencia de la misma al momento de consultar la BD.

Es por eso que se requiere de un proceso estructurado y detallado, que nos ayude a describir en términos abstractos la información del problema a resolver.

Lo que debemos hacer normalmente es pensar primero en qué elementos están involucrados en el problema. Generalmente vemos que es lo que se tiene para resolver el problema.

Así entonces, surgen los conceptos de Entidad y Relación (o vínculo).

Entidad: Representación particular de recursos de datos, o respuesta a un recurso de servicio que puede estar incluido en un mensaje de petición o respuesta. Una entidad consiste en "meta-información" en forma de cabeceras de entidad, y el contenido en forma de cuerpo de entidad.

De la definición mostrada entonces, podemos inferir que una entidad es todo objeto o concepto que interactúa o participa directamente en el problema.

Además, esas entidades tienen particularidades, propiedades o atributos que nos interesa almacenar.

Entonces, estas “entidades” deben relacionarse de alguna forma. Para ese efecto definimos **los vínculos o relaciones**.

- Los vínculos o relaciones son asociaciones existentes entre las entidades presentes en el problema.
- Estas relaciones generalmente representan asociaciones explícitas o visibles en el contexto del problema.
- Cada relación tiene un nombre, cardinalidad, aridad y puede tener también atributos.
- Estos vínculos van a constituir los enlaces entre tablas más adelante.
- En resumen, su primer paso es encontrar las entidades y relaciones que existen en el problema que se plantea.

Aridad: Cuantas entidades relacionan los vínculos.

Cordialidad: La mayor cantidad de las dos entidades que se relacionan.

Con respecto al último punto, es importante destacar que el modelo E-R es sólo el primer paso. En esta sesión, analizaremos el siguiente paso, el cual viene a ser la transformación del modelo E-R a un modelo relacional (de tablas).

¿Qué es una clave?

Por clave, se entiende un conjunto de atributos con ciertas propiedades. Hay tipos específicos de claves que deben ser definidas.

¿Qué tipos de claves?

Superclave: es un conjunto de atributos que definen una relación de forma única.

Clave candidata: es aquella superclave de una relación en la que no tiene ningún subconjunto que sea superclave de la misma relación. Eso implica que (por definición de superclave)

- 1) dentro de la relación no existen dos tuplas con la misma clave candidata
- 2) no es posible eliminar ningún atributo de la clave candidata sin afectar 1).

De las claves definidas en la transparencia anterior surgen dos tipos de claves esenciales para el modelo relacional:

Clave primaria: Es aquella clave candidata que se elige para identificar a las tuplas en una tabla de forma única (ésta no puede tener valores nulos en ninguno de sus atributos).

Clave externa o foránea : Es aquella clave que tiene valores que coinciden con valores de la clave primaria de otra relación.

Existe un tercer tipo de clave pero que no es tan relevante:

Clave alternativa: Son aquellas claves candidatas que no se eligieron clave primaria.

Es importante además destacar que el concepto de clave solo existe en el modelo relacional.

Algoritmo de transformación de E-R a Relacional

Ahora que tenemos el concepto de clave, estamos preparados para plantear una metodología o algoritmo de transformación de modelo E-R en modelo relacional.

El modelo relacional se refiere a una representación lógica de los datos como tablas o relaciones.

Paso 1:

Escriba una relación (tabla) por cada entidad que haya encontrado en el modelo E-R. Cada relación (tabla) lleva el nombre de la entidad y como campos o atributos de la relación (tabla) se transcriben los atributos de la entidad.

Paso 2:

Elija una clave candidata para cada relación (tabla) generada a partir de cada entidad. Esa será la clave primaria inicial de la relación (tabla).

Paso 3:

Para cada vínculo 1-1, agregue los atributos de la clave primaria de una de las entidades asociadas a la otra como clave externa. Se recomienda que se elija aquella entidad con mayor participación como la que “dona” su clave primaria. Si no hay participación completa, entonces se crea una nueva relación (tabla) que tiene por atributos la concatenación de claves primarias de las entidades involucradas.

Paso 4:

Para cada vínculo 1-n, agregue los atributos de la clave primaria de la entidad del lado 1 del vínculo a la relación de la entidad del lado n del vínculo como clave externa. Si no hay participación completa del lado n del vínculo, entonces se crea una nueva relación (tabla) que tiene por atributos la concatenación de claves primarias de las entidades involucradas.

Paso 5:

Para cada vínculo m-n, cree una nueva relación (tabla) con el nombre del vínculo y que tenga como atributos la concatenación de las claves primarias de ambas entidades involucradas. Esa concatenación será la clave primaria de la relación (tabla) y cada una de los conjuntos originales será clave externa hacia las relaciones (tablas) de las entidades que conforman el vínculo.

Paso 6:

Para cada vínculo n-ario, cree una nueva relación (tabla) con el nombre del vínculo y que tenga como atributos la concatenación de las claves primarias de todas las entidades involucradas. Esa concatenación será la clave primaria de la relación (tabla) y cada una de los conjuntos originales será clave externa hacia las relaciones (tablas) de las entidades que conforman el vínculo.

Nota:

Para todo vínculo en el modelo, se recomienda que si éste tiene un atributo inherente (asociado al vínculo en sí), se cree una nueva relación (tabla) en la cual sus atributos contengan a ese atributo como campo de la relación.

Una vez efectuado éste algoritmo de transformación, se obtiene el tan esperado modelo relacional (o de tablas).

Normalización de una BD

La normalización apunta a sostener una mayor optimización de los modelos lográndolo a través de ciertas condiciones, mayor rigurosidad y cambios de esquema del modelo relacional planteado inicialmente.

Dentro de los principales problemas que encontramos:

Redundancia:

Problemas para modificar:

Problemas en el ingreso:

Problemas en la eliminación:

Uno de los contras de la normalización es que al crear mas tablas incrementa la cantidad de Joins que hacer, esto empeora la eficiencia por esto hay que ver si vale la pena normalizar o no, y hasta que forma normal hacerlo.

Formas Normales

Primera Forma normal o 1NF:

El requisito para que el esquema este en 1NF es que ninguna de sus relaciones tiene un atributo multivaluado, es decir que todas las relaciones tienen atributos formados por un solo valor.

Segunda Forma normal o 2NF:

Para que el esquema este en segunda forma normal tiene que estar en primera forma normal y además cumple que todos los atributos no primos (Que no pertenecen a alguna clave candidata) entregan información de la clave completa.

Para transformar las relaciones a 2NF se tiene que crear nuevas relaciones de tal forma que la llave esté relacionada solo con los atributos que entreguen información referente a esa llave.

Tercera Forma normal o 3NF:

Para que este el esquema en 3NF debe estar en 2NF y además cumplir que todos los atributos no primos (que no pertenecen a una llave candidata) entregan información de la clave completa y no de otros atributos.

Dependencias Funcionales:

Definen asociaciones entre atributos de una relación

Los elementos que existen en dependencias funcionales se denominan descriptores y definen subconjuntos del conjunto total de los atributos.

Estas asociaciones constan de un implicante y un implicado.

Sean X e Y descriptores. Se dice que Y depende funcionalmente de X o que X determina o implica a Y, que se representa por $X \rightarrow Y$, si solo si, cada valor de X tiene asociado en todo momento un único valor de Y.

Por ejemplo: RUT \rightarrow nombre

Significa que para dos tuplas con el mismo RUT tendrían también el mismo nombre.

Transitivas:

1NF

No pueden existir atributos multivaluados en el esquema.

2NF

Está en 1NF

Todo atributo no-primo tiene dependencia funcional completa respecto de cada una de las claves candidatas.

3NF

Está en 2NF

Ningún atributo no-primo depende de forma transitiva de ninguna clave candidata

Structured Query Language

Lenguaje de consulta

Sirve para ingresar, crear o consultar datos.

Sintaxis

Select

Drop

Insert

Alter table

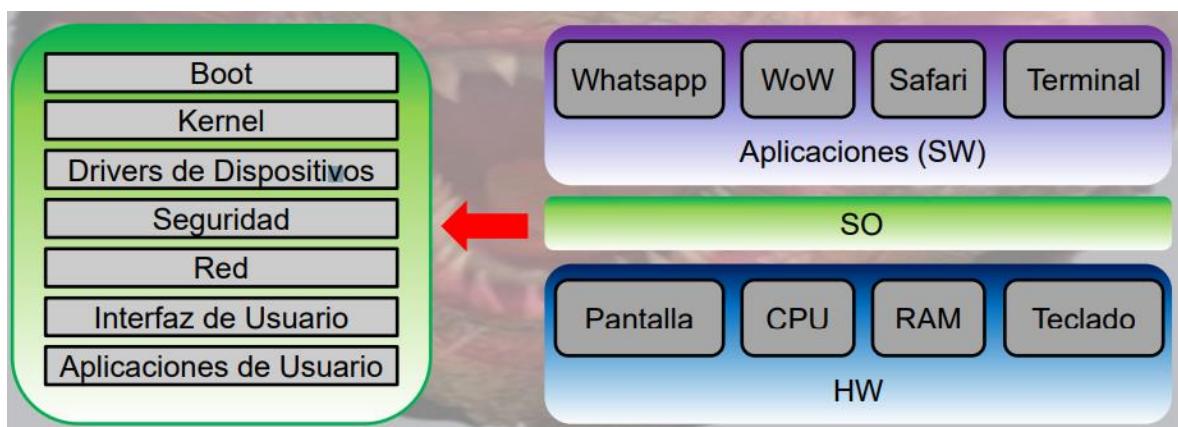
Delete

Sistemas Operativos

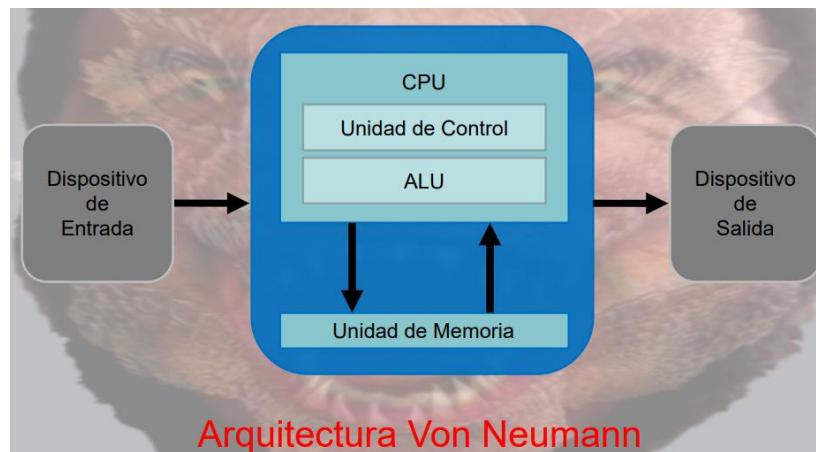
¿Qué es un sistema operativo?

Un SO gestiona los recursos, concretamente:

- Administración de la CPU, memoria, dispositivos de I/O, archivos, red y la adaptación de todos ellos.
- Establece prioridades entre tareas
- Hace compartición de tiempo entre los diversos SW presentes en el sistema.
- Administración de seguridad e integridad.
- Provee de independencia de HW
- Ofrece una interfaz de usuario hacia los recursos.



SO = Kernel + Programas de Sistema



Procesos

Recordemos que un proceso es la abstracción de un programa en ejecución (junto con todos los recursos). Dicho programa abarca cierto espacio en la memoria, donde tiene la capacidad para leer y escribir datos y además contiene a su stack de ejecución.

Este concepto es central relacionado con los SO, puesto que el objetivo de un computador es el de operar sobre datos y entregar resultados/efectuar acciones.

Existe un paralelismo que se hace evidente cuando hay HW dedicado a la operación múltiple de procesos secuenciales. Por lo que también hay que repartir/administrar recursos paralelamente, por lo que el SO es clave.



– Datos del programa y estado de actividad (II)



Los procesos pueden estar en diferentes estados los cuales son:

Categorías de procesos:

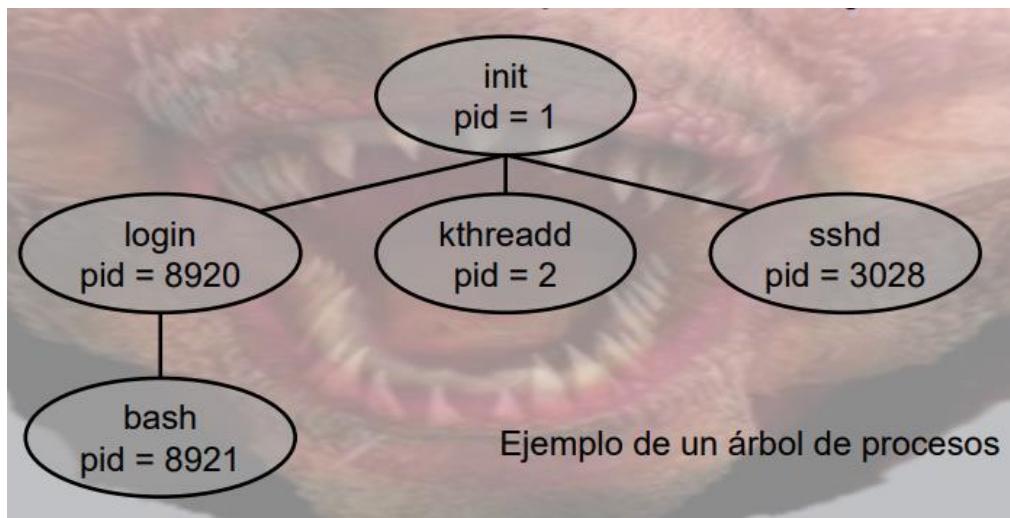
- Asociados a la CPU
- Asociados a I/O
- Procesos independientes
- Procesos cooperativos

Threads

Es una tarea específica parte del proceso completo, son las tareas más livianas del proceso.

- Los threads típicamente están asociados a una tarea específica.
- Están asociados a un proceso, de modo que un proceso puede efectuar múltiples tareas.
- La información de cada thread se almacena en el PCB.

Los procesos se vinculan en forma de un árbol. Cada proceso tiene un padre y puede tener 0 o más procesos hijos.



Para ejecutar varios programas al mismo tiempo se utiliza el Scheduling (de corto plazo) de procesos, de modo de optimizar el uso de la CPU y ordenar la ejecución de las tareas.

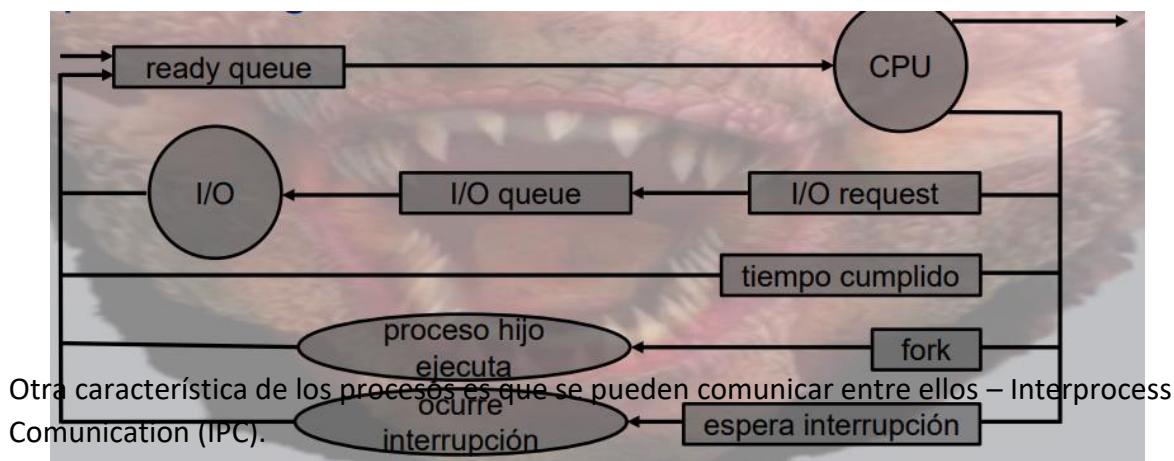
Para esto intervienen varias colas de procesos, las más importantes son:

- Job queue (procesos que entran al sistema)
- Ready queue (procesos listos y en RAM)

Los estados de los procesos se rigen de acuerdo con el grafo mostrado previamente

Sus tareas quedan muy vinculadas a su clasificación (I/O, CPU).

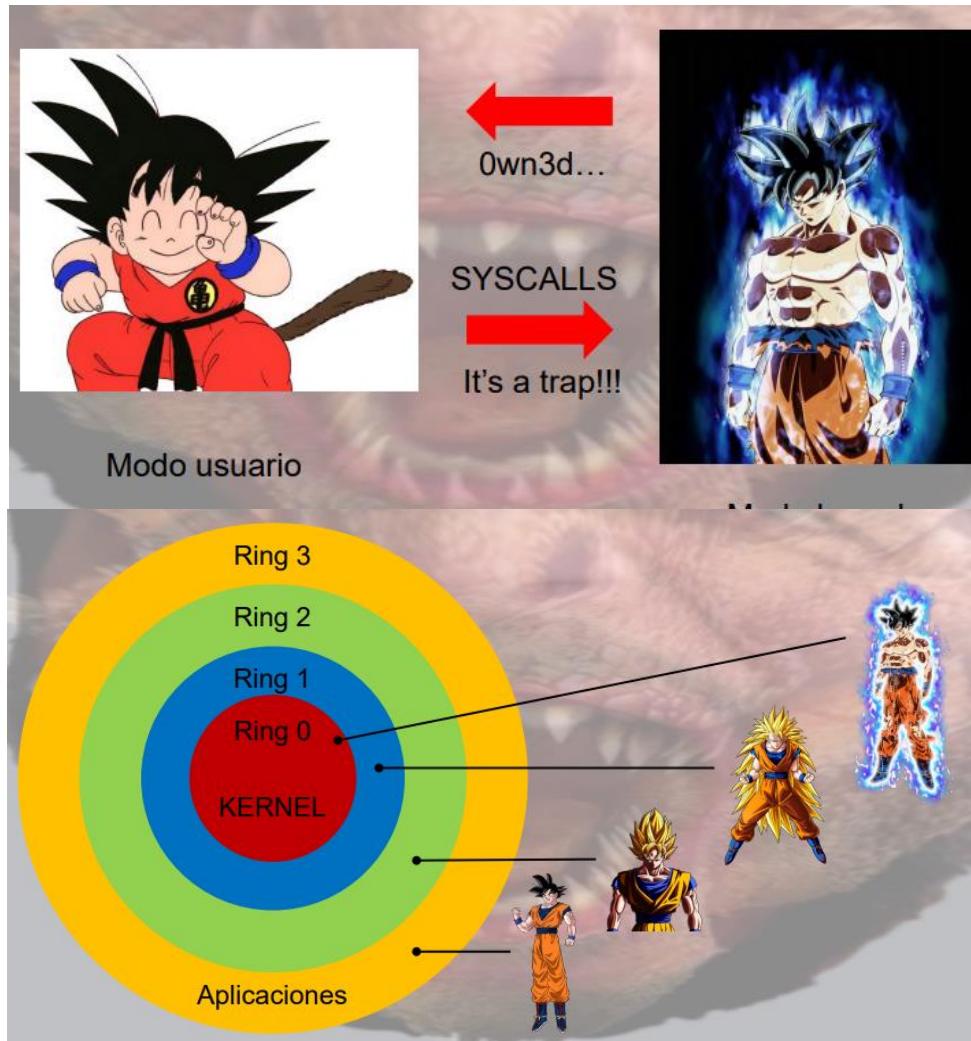
La dinámica de ejecución de los procesos en CPU sigue típicamente la lógica descrita por el programa.



Estas comunicaciones se llevan a cabo típicamente de dos maneras fundamentales:

- Paso de mensajes (Similar a los Zoquetes de redes - pipes)
- Memoria compartida

Los programas funcionan en modo dual.



Pipes:

Los pipes son canales de comunicación entre procesos. Consideran múltiples variables (unidireccional/bidireccional) full/half duplex, requisito de “parentesco” entre procesos comunicados, intramaquina o intermaquina.

Existen dos clases de pipes:

- Pipes comunes (anónimos)
- Pipes con nombre

El Inter Process Communication es necesaria para que los procesos puedan señalar resultados u operaciones requeridas entre ellos.

Pipes responden a un método de pasos de mensajes.

Notar que en los ejemplos mostrados hay una rutina fork() "Bifurcación"

Treads

Son procesos livianos destinados a efectuar tareas específicas. Tienen sus propios valores (entre ellos también una prioridad) y entorno.

Ese entorno se denomina contexto.

Contextualizando, ese contexto se refiere al contexto que contextualiza la operación del thread.

Los threads van asociados a un proceso que los inicia y a su vez, pueden tener atribuciones sobre otros threads del mismo proceso.

Así pues, el contexto tiene una cantidad mínima de información referente al thread específico. El resto de la información es compartida con el proceso padre.

Una particularidad del fork() es que el proceso padre genera al proceso hijo y espera a que ese termine de ejecutar.

Bajo UNIX el fork() es la única forma de crear procesos. Luego de ello se llama a execve() que le asigna el programa a ejecutar al proceso (como se inicia el primer proceso).

Bajo Windows, se hace uso de la syscall CreateProcess. Efectúa ambas acciones en una sola llamada.

De manera similar hay syscalls para terminar la ejecución de un proceso.

- Exit en UNIX
- ExitProcess en Windows.

Existen varios casos de la terminación de un proceso:

- Salida normal
- Salida error
- Error fatal

- Asesinato(kill,TerminateProcess)

Para lidiar con las transiciones es necesario presentar el concepto de interrupciones y de Scheduling (calanderización)

Elementos de control de procesos:

- Interrupciones
- Syscalls
- Scheduling
- Sincronización

Interrupciones

Son señales que indican una necesidad de detener un proceso en ejecución para dar lugar a otro requerimiento que necesita atención inmediata->Multitasquing, cambio de contexto.

Tipos:

- Interrupciones de HW
- Interrupciones de SW

Para poder manejar las interrupciones y en caso de que lleguen varias concurrentes, existe una cola de interrupciones. Estas interrupciones tienen también un “Nivel de prioridad”. Son manejados por un Interrupt Handler que se le denomina Interrupt Vector.

Las interrupciones de HW son generadas por dispositivos de I/O al terminar su operación -> IRQs

Las interrupciones de SW también se generan al finalizar la ejecución de programas o bien al requerir tareas por parte del SO.

Las interrupciones generan cambio de contexto.

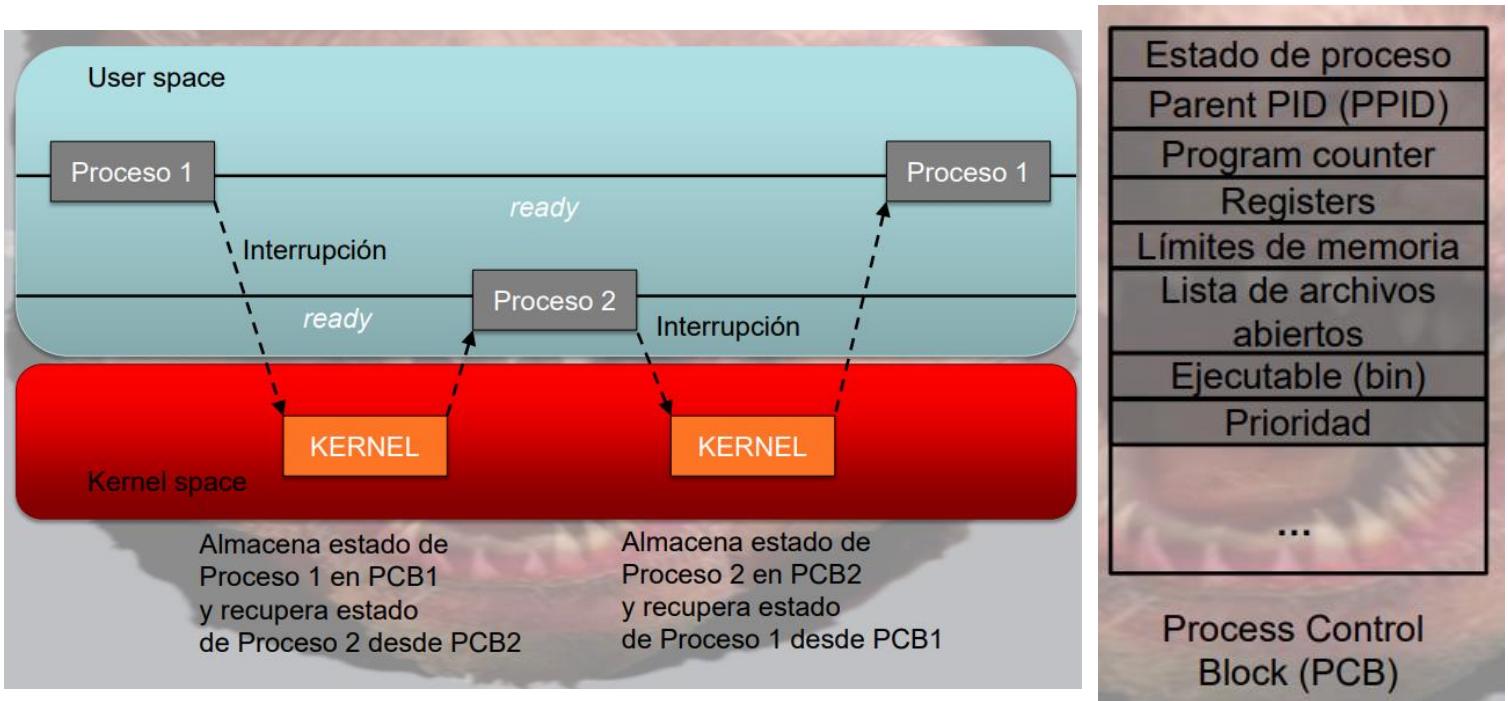
Cambios de contexto y PCB

Los cambios de contexto involucran almacenaje y carga de los datos desde/hacia el PCB.

Eso se debe a que los procesos que salen de la CPU deben guardar su estado para poder ser restaurado una vez vuelvan a ejecutar en CPU.

Cambio de contexto

PCB- Process Control Block



Las interrupciones, en cuanto a procesos concierne, muchas veces se relacionan con las Syscalls:

- Fork() es utilizada como bifurcación, separar o duplicar un proceso, al crearlo el port ID cambia y el proceso creado quedaría como hijo del creador.
- Exec() cuando es invocado, el programa especificado en el parámetro exec() va a reemplazar el proceso completo incluidos los threads.
- Wait() espera por la finalización de uno o mas procesos hijos, devuelve una tupla que contiene el pid del hijo y la indicación del exit status.
- Kill() envia una señal al proceso según su pid, según signal.
- Sleep()
- Killall()

Todas las señales: <https://mcsp.wartburg.edu/zelle/cs360/handouts/python-process.html>

Scheduling

Implementar multitasking o asignación de tiempo de CPU a múltiples procesos.

Tipos de scheduling

- Long term Scheduler: se refiere a la selección de los procesos que quedan en la Ready Queue y define su cantidad.
- Medium term scheduler: Modifica temporalmente el grado de multiprogramación y hace swapping (RAM -> HDD y HDD-> RAM)
- Short term scheduler: Manda procesos de la Ready Queue a CPU y efectua el cambio de contexto.

El scheduling es importante para organizar la multiprogramación. Ya que si hay mucho cambio de contexto pierde mucha eficiencia (Thrashing).

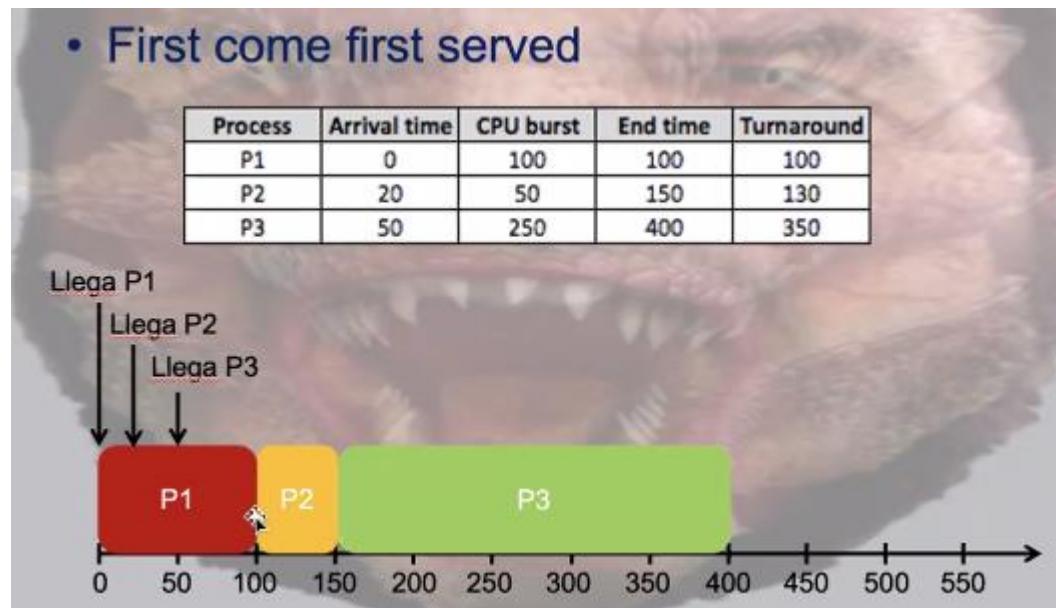
Hay varios algoritmos de Scheduling, se clasifican a su vez por las dimensiones según las cuales toman sus decisiones y/u operan.

- Scheduler expropiativo
- Scheduler no-expropiativo
- Batch scheduler
- Interactive scheduling
- Real-time scheduling

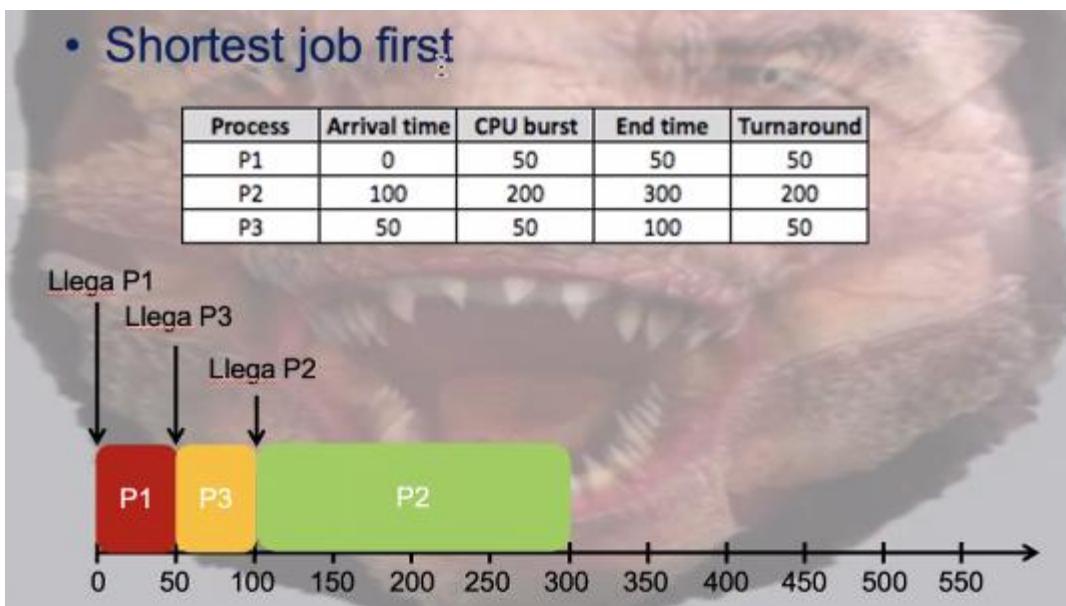
En todos estos casos se busca que el proceso tenga el uso de la CPU un tiempo razonable.

Algoritmos de Procesos

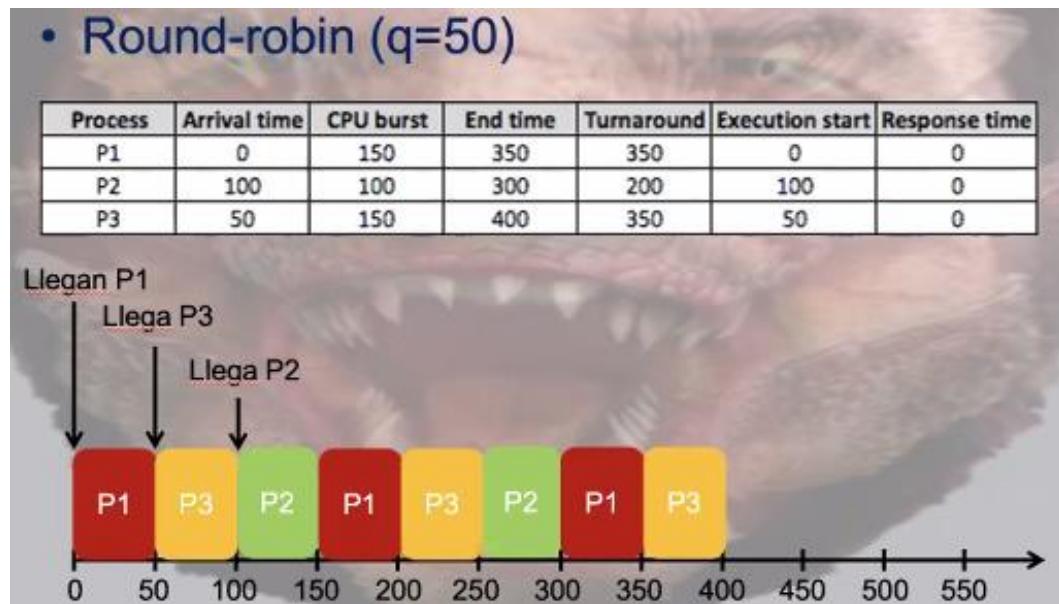
- First come first served: no expropiativo



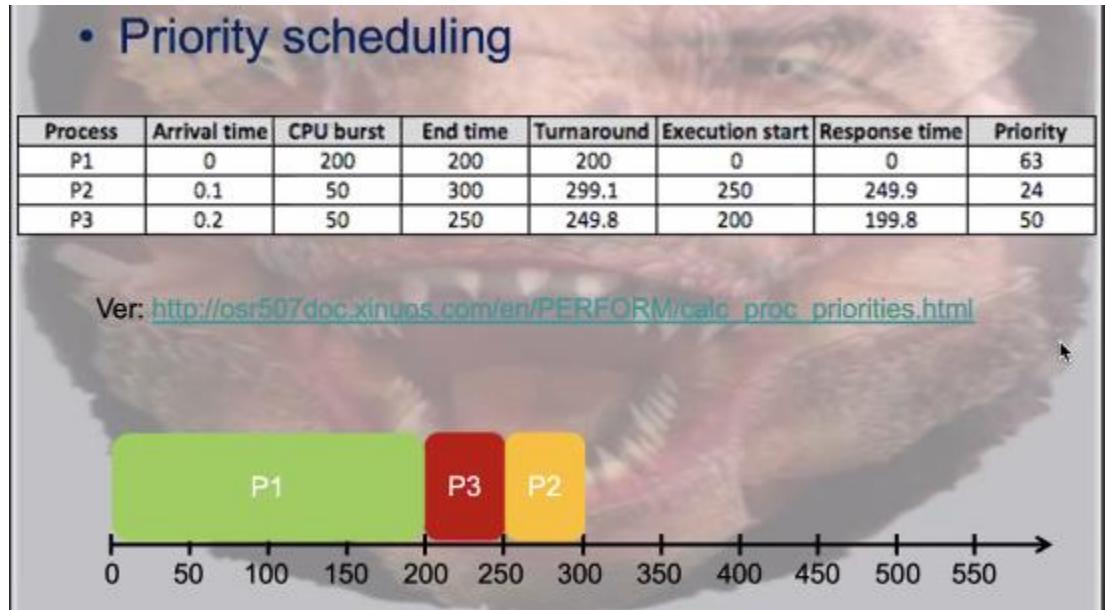
- Shortest Job first: no expropiativo



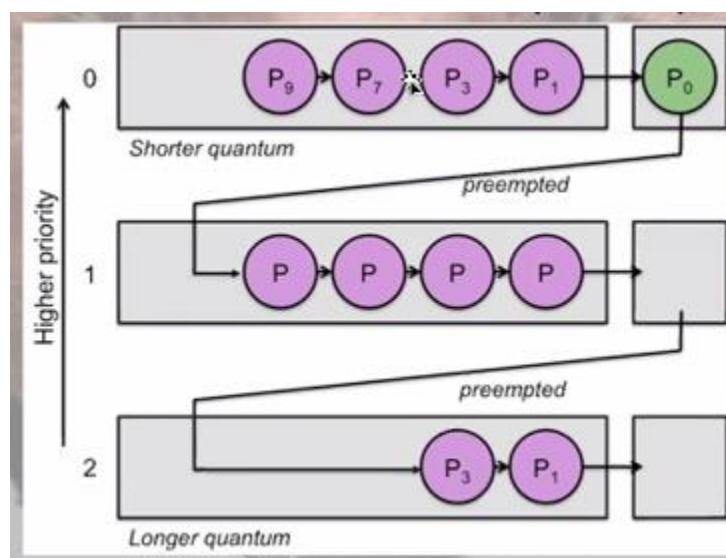
- Round Robin: expropiativo



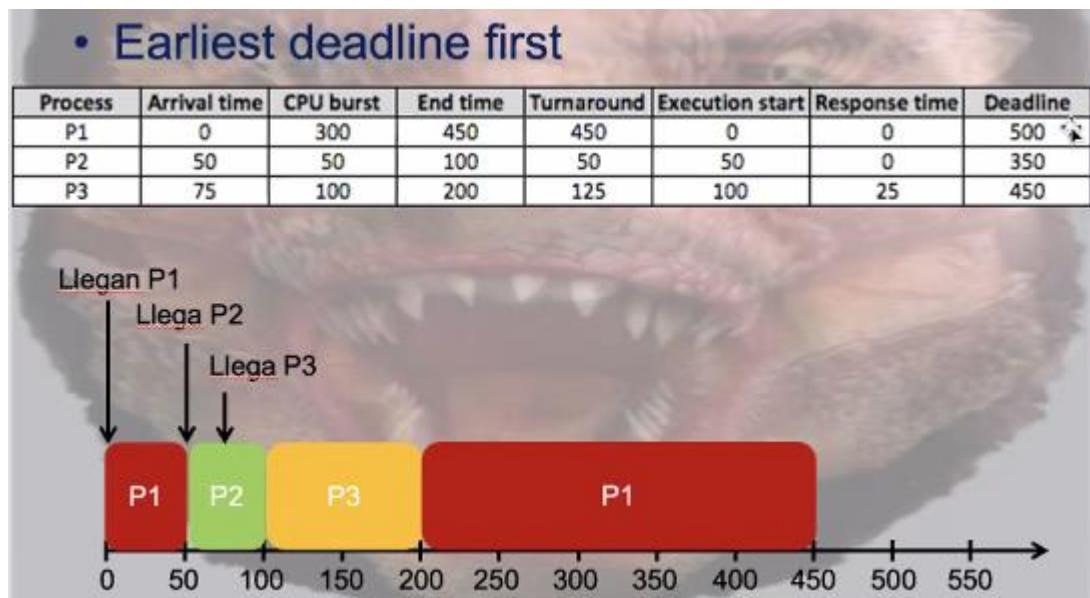
- Priority scheduling: expropiativo o no expopiativo



- Multilevel Feedback Queue (MLFQ): expropiativo



- Earliest deadline first: expropiativo



Procesos Threads

Threads son procesos livianos que, al igual que los procesos, requieren los datos del PCB para ejecutarse.

Diferencias entre threads y procesos:

- Un thread siempre está asociado a un proceso y efectúa una tarea específica. Cada proceso puede tener asociados uno o más threads.
- Las operaciones asociadas a los threads son menos costosas que la de los procesos.
- Procesos: Agrupan recursos y ejecución
- Threads: Unidades de ejecución.

Un thread comparte datos con su proceso padre, pero también tiene datos propios.

Específicamente:

Proceso	Thread
<ul style="list-style-type: none"> • Memoria y vars globales • Archivos • Procesos hijos • Alarmas pendientes y señales 	<ul style="list-style-type: none"> • Stack • Registros • Estado

Las operaciones típicas(en POSIX threads) que se usan son las siguientes:

- `thread_create()`
- `thread_exit()`
- `thread_join()`
- `thread_yield()>sched_yield()`: ceder el control

Condiciones de carrera

Representan una dependencia de las salidas de una operación en base al orden temporal de ejecución de las instrucciones internas sin control por parte del programador.

Sección critica (SC):

Se refiere a la región de memoria que trabaja el código. Requiere que únicamente un proceso o thread este trabajando sobre dicha región al mismo tiempo. Osea debe tener un acceso secuencial y único a dicha región.

Exclusión mutua: al menos un thread/proceso puede entrar a su SC. Si no hay threads/procesos en su SC y alguno(s) quiere(n) entrar, deben decidir cual en un tiempo acotado.

Ausencia de inanición: Si un thread/proceso quiere entrar a su SC, podría hacerlo después de un tiempo.

Opciones:

- Lock (Spinlock)
- Eliminar interrupciones
- Turn (Algoritmo de Peterson)

Locks de Mutex : Se abstrae a una variable que garantiza la atomicidad de su manejo y exclusión mutua de acceso a la SC

- Acquire

- Release

Semáforos

Es un mecanismo de sincronización inventados por Dijkstra, en vez de tratarse de un lock que restringe el acceso de un único thread a una SC, permite que accedan $N > 0$ procesos a la sección critica, se implementa mediante un contador que se incrementa/decrementa automáticamente.

Para un semáforo S, se implementan 2 funciones:

P(), wait()> intenta decrementar el valor del contador para simbolizar que ha entrado un thread a la SC.

V(), signal()> intenta incrementar el valor del contador. Representa una ubicación libre más para que otro thread pueda entrar a la sección critica.

Cuando el contador del semáforo llega a 0, no pueden entrar mas threads a la SC.

Variables de condición

Son mecanismos de sincronización en que la condición no viene de limitar la cantidad de threads que acceden a la SC, sino que se trata de condiciones arbitrarias.

- Wait() que SIEMPRE bloquea el thread.
- Signal() que despierta a un thread bloqueado en caso que ya lo haya

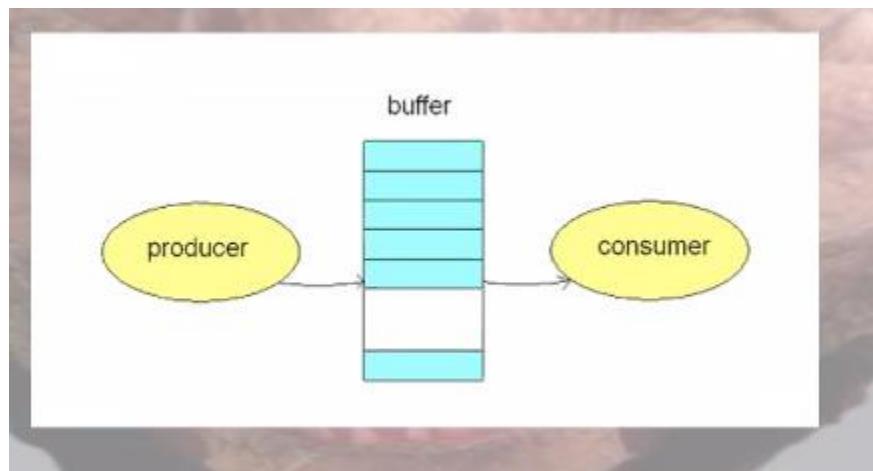
Monitor

Se trata de un objeto que contiene funcionalidades, y es resguardado por un lock en su totalidad al ser accedido por un thread.

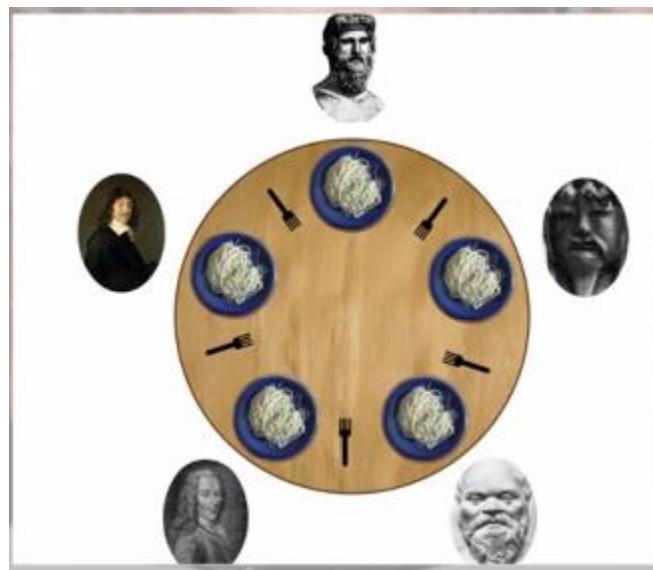
Cualquier operación del monitor queda bloqueada hasta que el thread que llama libere el lock

Fábulas - Modelos de problemas

Productor consumidor:



Filósofos comensales:



Escritor lector:



Deadlocks

Es una situación en la cual dos o mas threads quedan bloqueados para siempre esperandose entre ellos.

Ejemplo: Dos trenes que van en direcciones contrarias se encuentran en el camino que hay una sola vía.

Los deadlocks seueden dar en presencia de las siguientes condiciones(necesarias y simultaneas):

- Exclusión mutua
- Espera circular
- No expropiación

Para manejar los deadlocks, se tienen las siguientes estrategias:

Prevencion de deadlock (no dejar que el sistema llegue a ese estado.)

Detección y recuperación (dejar que ocurra y luego aplicar resolución y posteriormente prevención para evitar que ocurra)

Ignorar el problema (sorprendentemente, es la estrategia empleada por Windows y unix)

Swaps:

- Swap-out:
- Swap-in:

Procesos en memoria:

- Compactación(desfragmentación):
- First-fit: donde quepa
- Best-fit: analiza todo y ve la opción que deje menor espacio vacio en la ram

- Worst-fit: analiza y elige la opción que deje mayor espacio sin uso.

Fragmentacion:

-Interna

-Externa

Paginación:

Min:

RAndom:

List resenly used (LRU):

Relog: