38. Quines lo conforman y cuales son las actividades del grupo de SQA.
R.- Las actividades del SQA son:

- Define Estándares, Métodos y Herramientas
- Revisiones y Verificaciones
- Ajuste a los estándares establecidos
- Seguimiento a los cambios
- Mediciones y Auditorias
- Registro y realización de informes

39. Cual es la diferencia entre el estándar de calidad del ISO y el estándar de calidad del IEEE
R.- **El estándar IEEE esta:** Para la IEEE la calidad de software es el grado en que un sistema, componente o proceso cumple con los requerimientos especificados y con las necesidades o expectativas del cliente o usuario.

- Orientado al desarrollo de software critico
- No es recomendado para software que ya empezó a ser desarrollado
- Apoya la preparación y definición de SQAPs.

El estándar ISO se basa en el producto:

- Corrección adecuación, precisión, interoperabilidad,
- Fiabilidad recuperabilidad, conformidad
- Eficiencia comportamiento en el tiempo, utilización de recursos, conformidad
- Usabilidad facilidad para comprenderlo, aprenderlo, aprenderlo, operarlo, grado en que resulta atractivo, conformidad
- Mantenibilidad facilidad para ser analizado, cambiado, probado, estabilidad, conformidad
- Portabilidad facilidad para adaptarlo, instalarlo, capacidad de coexistir, reemplazar, conformidad

40. Es siempre una buena idea la modularización durante el desarrollo de un software
R.- **Concepto.-**

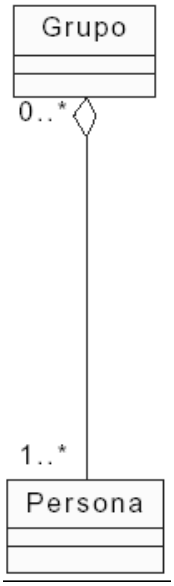
- Componentes identificables y tratables por separado
- Permite a un programa ser manejable intelectualmente
- Criterios que permiten evaluar un método de diseño con respecto a su capacidad de definir un sistema modular eficaz.

La modularización es buena en el diseño cuando se trata de sistemas medianos.

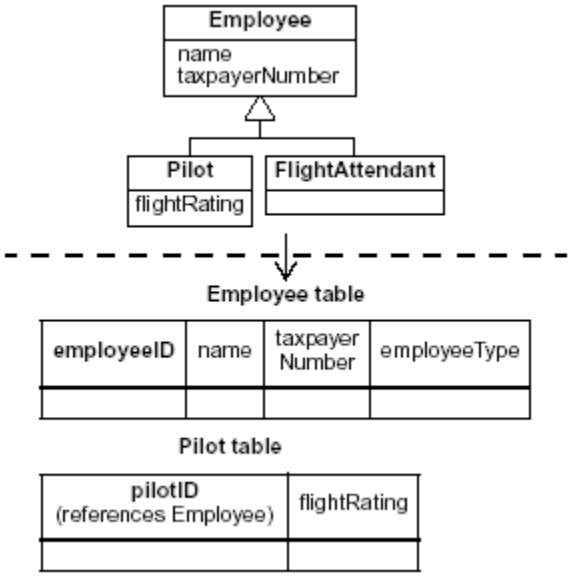
41. En el diseño de software a que se denomina factorización
R.-
42. Como evaluamos si un diseño modular es efectivo
R.- Cuando:

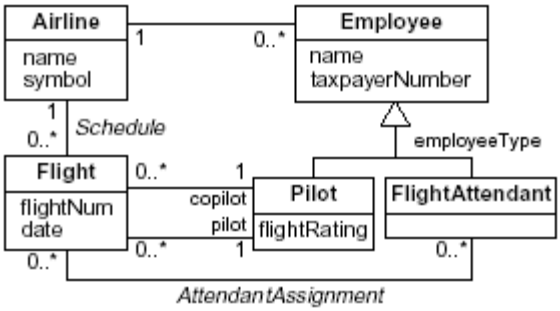
- Hay Capacidad de descomposición modular
- Hay Capacidad de empleo de componentes modulares (reutilización)
- Hay Capacidad de comprensión modular (entender un módulo sin referencias a otros, o con las menos posibles)
- Hay Continuidad modular (cambios en módulos y con poco impacto)
- Hay Protección modular

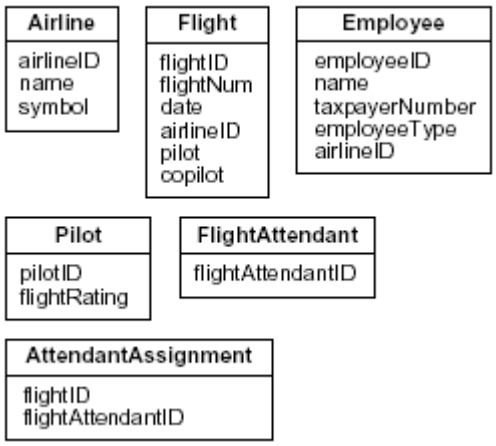
43. Haga un ejemplo donde se vea un diseño con problemas de acoplamiento y cohesión y otro ejemplo libre de esos problemas
R.-
44. Como representa el diseño modular con UML (haga un ejemplo)
R.-
45. Para un desarrollo de software siguiendo el método OO que ciclo de vida utilizaría y por que
R.-
46. Haga un ejemplo de clases utilizando agregación y realice el mapeo a BD relacionales
R.-



47. Con UML como modelaría una estructura de generalización / especialización y luego haga el mapeo a una BD Relacional (haga un ejemplo)

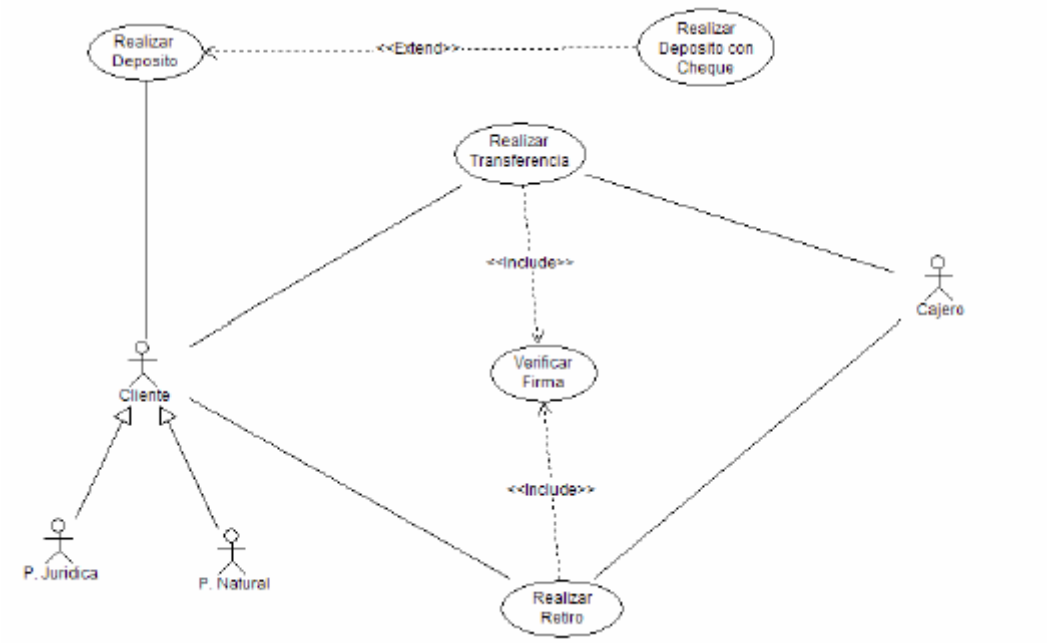






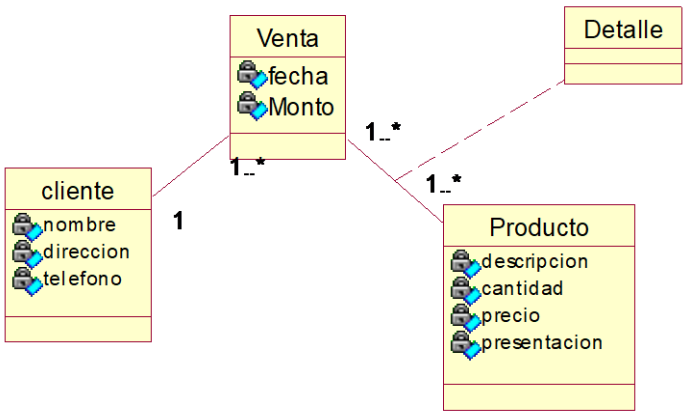
48. En un diagrama de casos uso muestre el uso de generalización / especialización

R.-



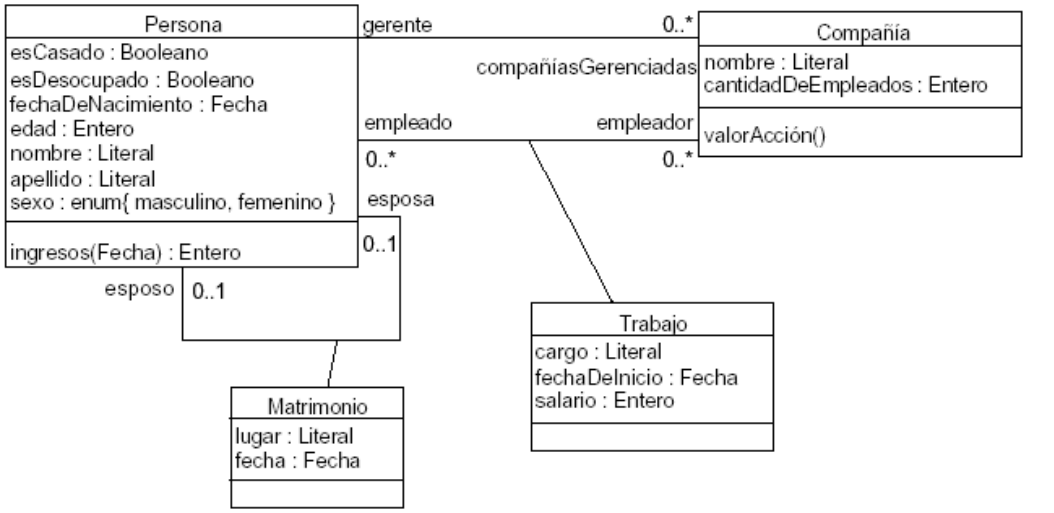
49. Para el caso de uso Retirar dinero de una cuenta corriente de un banco haga el diagrama de secuencia correspondiente.
R.-

50. Realice un diagrama de clases para el caso de uso realizar venta de productos en una Importadora
R.-



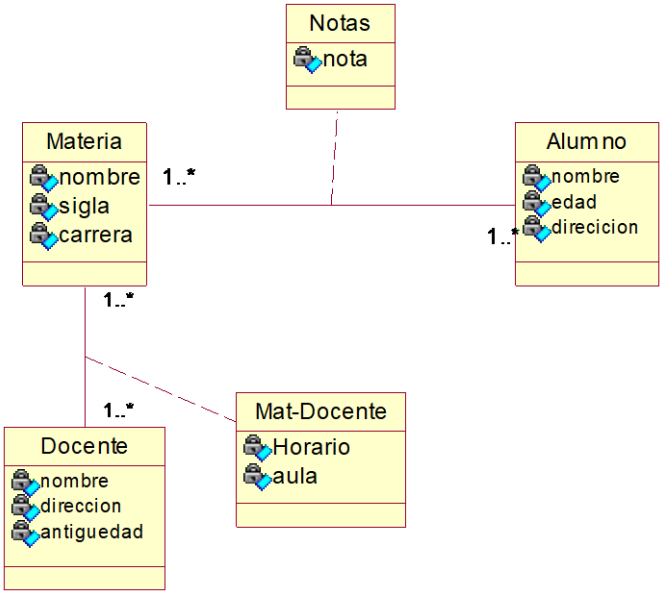
51. Para el caso de uso Depositar dinero de una cuenta corriente de un banco haga el diagrama de secuencia correspondiente.
R.-

52. Haga un ejemplo de un diagrama de clases que contenga una asociación recursiva y que genere una clase asociación. Luego realice el mapeo a un BD relacional
R.-



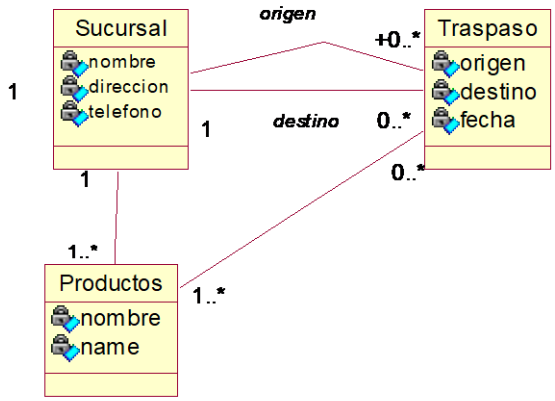
53. Haga un diagrama de clases para el registro de notas de alumnos, considere que un alumno lleva mas de una materia, y que una misma materia puede ser dictada por mas de un docente.

R.-



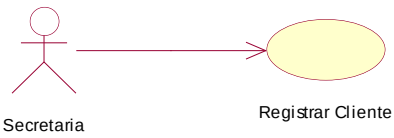
54. Haga un diagrama de clases para el registro de traspaso de productos entre sucursales de una importadora

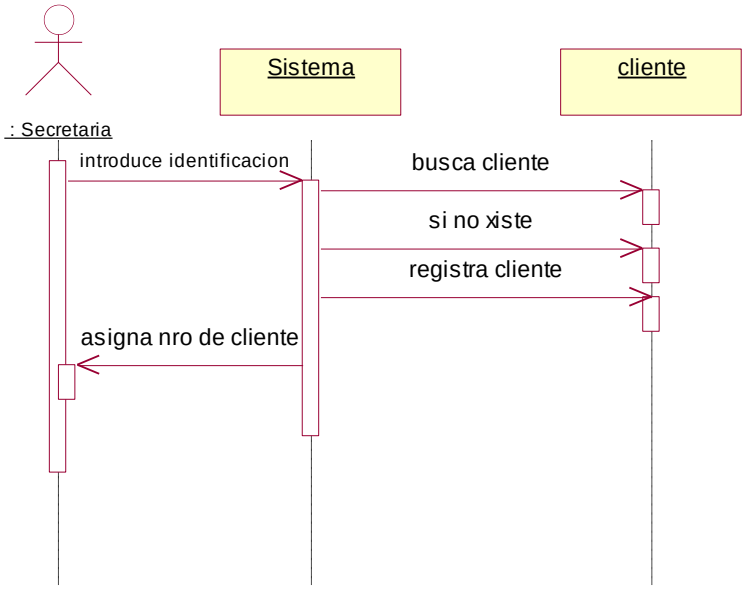
R.-



55. El registro de clientes en una importadora lo realiza la secretaria de dicha importadora, haga su diagrama de casos de uso y su respectivo diagrama de secuencia

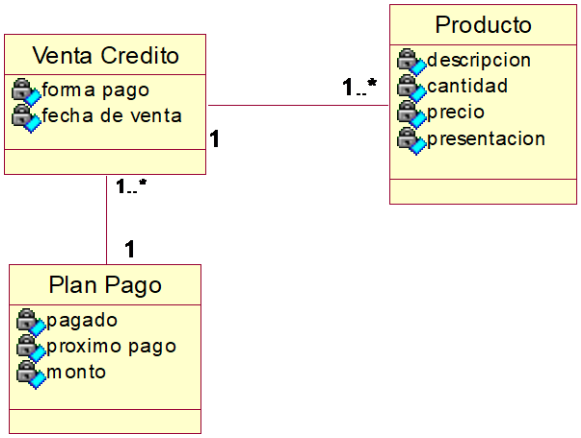
R.-





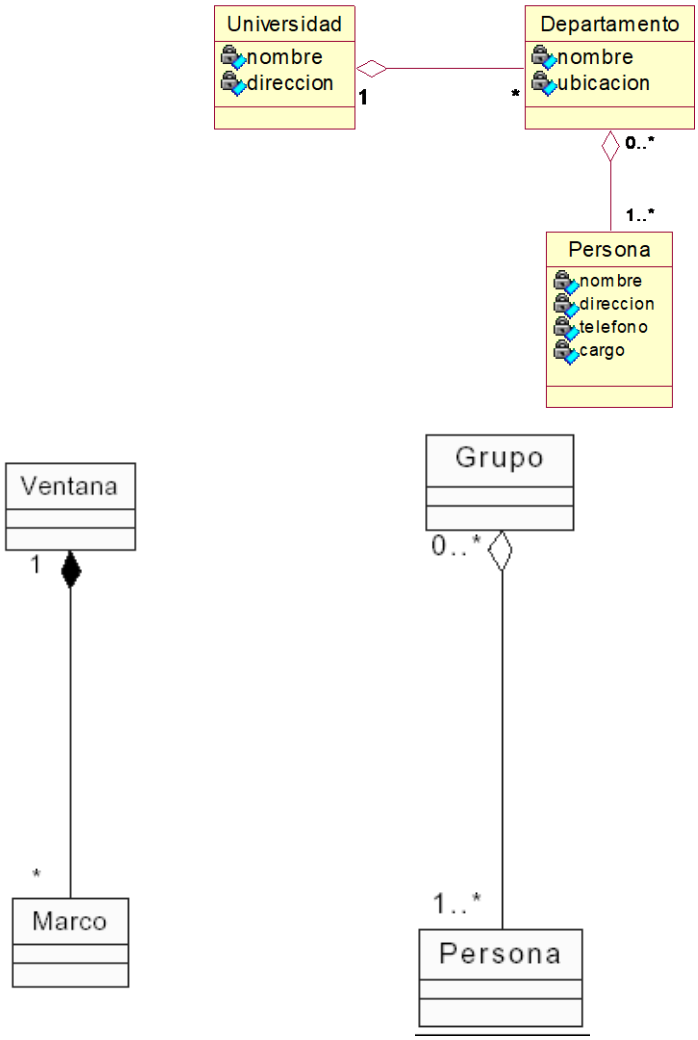
56. Realice el diagrama de clases para registrar una venta de productos al crédito

R.-



57. Realice ejemplos utilizando agregación y composición entre clases

R.-



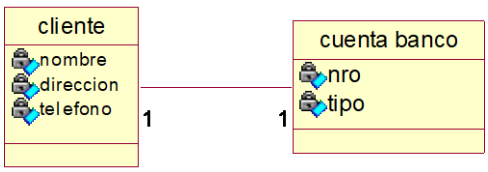
58. Bajo que criterios decide usar los estereotipo <<uses>> y <<extiende>> en un diagrama de casos de uso

R.- <<uses>> .- Cuando un Caso de uso necesita de la funcionalidad de otro.

<<extiende>>.- Cuando la realización de un caso de uso puede usar o no otro.

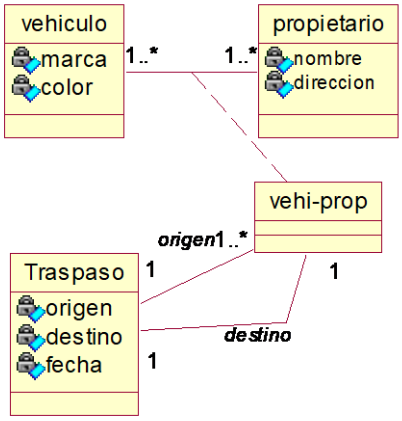
59. Haga ejemplos de asociaciones (1 – 1) , (0 – 1), (1 a muchos), (0 a muchos) y (muchos a muchos) usando clases

R.-



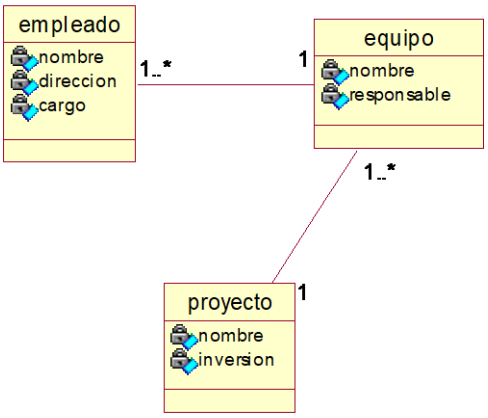
60. Realice el diagrama de clases para registrar el traspaso de vehículos de un propietario hacia otro, se debería poder saber quienes fueron los d ueños del vehículo.

R.-



61. Realice el diagrama de clases para una constructora que quiere registrar la asignación e empleados a equipos de trabajo para los diferentes proyectos de construcción, para cada equipo un trabajador es el responsable.

R.-



62. Taxonomia de las herramientas CASE

R.-

63. Cual es el ciclo de vida de las herramientas CASE

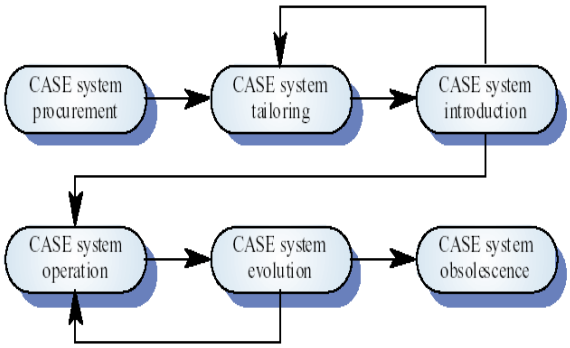
R.-

1. **Procuración**
   - Existen compañías y métodos estándar
   - Existencia de hardware futuro
   - La clase de aplicaciones a ser desarrolladas
   - Seguridad
2. **Adaptación**
   - Instalación
   - Definición del modelo de procesos
   - Herramientas de integración
   - Documentación
3. **Introducción**
   - Puede requerir adaptarse a practicas de trabajo comunes

- Migración de proyectos hacia los sistemas CASE

4. **Operación**
   - Puede requerir adaptarse a practicas de trabajo comunes
   - Migración de proyectos hacia los sistemas CASE
5. **Evolución**
   - Conforme el sistema se usa, surgen nuevos requerimientos
   - Una evolución del presupuesto debe estar disponible o el medio ambiente será progresivamente menos útil con el tiempo
   - La compatibilidad a futuro debe mantenerse

6.- **Obsolescencia**
   - En alguna etapa, el medio ambiente caerá en desuso y tendrá que reemplazase
   - El reemplazo de un medio ambiente debe planearse y colocarse en período de tiempo específico
   - Los proyectos soportados en forma corriente deben trasladarse a un nuevo ambiente , antes de que el soporte se caiga



64. Cuales son los conceptos esenciales del Diseño

R.-

1. **abstracción.-** permite especificar procedimientos y datos suprimiendo detalles de bajo nivel (Procedimental, De datos, De control).
2. **refinamiento**.- la arquitectura de un programa se desarrolla refinando sucesivamente niveles de detalle procedimental. Se desarrolla una jerarquía descomponiendo una abstracción procedimental para, paso a paso, llegar a los enunciados del lenguaje de programación**.** El refinamiento ayuda a revelar detalles de bajo nivel a medida que progresa el diseño.
3. **modularidad.-**
   - Componentes identificables y tratables por separado
   - Permite a un programa ser manejable intelectualmente
   - Criterios que permiten evaluar un método de diseño con respecto a su capacidad de definir un sistema modular eficaz.
4. **arquitectura**
5. **jerarquía de control**
6. **partición estructural**
7. **estructura de datos**
8. **ocultación de información**

65. Que es la arquitectura de software

R.- La arquitectura de software es una frase ambigua, ya que se le asigna diferentes significados, dependiendo del escenario y de los interlocutores.

Puede representar al proceso (se utiliza como sinónimo del “proceso de diseño de la arquitectura”).

Puede representar al producto. Aquí tiene 2 acepciones:

- Representa el aspecto estético y funcional de un producto de software
- Representa la estructura interna (o macroestructura) sobre la cual se apoya el software

66. Cual es el proceso del diseño de la arquitectura de software

R.- El proceso involucra las siguientes actividades:

- **Estructuración del Sistema.-** El sistema se descompone en varios subsistemas principales y se identifican las comunicaciones entre estos subsistemas.
- **Modelos de Control.-** Un modelo de control establece las relaciones entre las diferentes partes de un sistema.
- **Descomposición Modular.´-** Los subsistemas identificados son descompuestos en módulos.

Por lo general estas tres actividades están entrelazadas, más que llevarse a cabo en forma secuencial.
La entrada a este proceso son los requisitos del sistema.
La salida es el documento de diseño arquitectónico.
Hay más de un camino para ir desde la entrada hasta la salida.
aunque siempre hay algunos que son más convenientes que otros.

67. Modelo Cliente/Servidor

R.-

- modelo de sistemas distribuido que muestra cómo datos y procesamiento se distribuyen a lo largo de varios procesadores.
- Sus componentes:
  - **conjunto de servidores** independientes que ofrecen servicios a otros subsistemas(servidores de impresión, servidores de administración de archivos, servidores de compilación)
  - **conjunto de clientes**.- Invocan los servicios ofrecidos por los servidores
    Existen varias instancias de un programa cliente que se ejecuta de forma concurrente. Tienen que conocer los nombres de los servidores disponibles y los servicios que suministran, pero los servidores no conocen a los clientes.
  - Una red que permite a los clientes acceder a los servicios.
- No existe una relación 1:1 entre procesos y procesadores: un computador servidor puede ejecutar varios procesos servidores

68. Modelo repositorio

R.-

- arquitectura en la que todos los datos se ubican en una base de datos central a la que acceden todos los subsistemas
- útil en sistemas que utilizan grandes cantidades de datos, generados por un subsistema y utilizados por otro
- Ejm. sistemas de información corporativa, sistemas CAD y CASE, sistemas CAD y CASE.

69. Arquitectura tres capas

R.-

- **Capa de presentación**: se encarga de mostrar la información e interactuar con el usuario.
- **Capa de procesamiento** de la aplicación: implementa la lógica de la aplicación.
- **Capa de administración de datos:** se refiere a todas las operaciones de la base de datos.

70. Explicar modelado de control del proceso de diseño de la arquitectura de software

R.-

1. **Control centralizado.-**
   - un subsistema tiene la responsabilidad de controlar el sistema y administrar la ejecución de otros subsistemas
   - dos clases, según se ejecuten secuencialmente o en paralelo
     1. **modelo de llamada- retorno (**ejecución secuencial):
        a. el control se inicia en la parte superior de una jerarquía y por medio de llamadas a subrutinas pasa a los niveles del árbol
        b. no es un modelo estructural, por lo que no es necesario, por ejemplo, que la Rutina 1.1. forme parte de la Rutina 1
        c. sólo se aplica a sistemas secuenciales
        d. utilizado por lenguajes de programación como Ada, Pascal y C, aunque también en lenguajes OO.
        e. ventaja: es relativamente sencillo analizar los flujos de control y conocer cómo responderá el sistema a cierto tipo de entradas
        f. inconveniente: las excepciones a operaciones normales son complicadas de gestionar

     2.- modelo del administrador:
        a. se aplica a los modelos concurrentes
        b. un componente del sistema se designa como administrador y controla el inicio, detención y coordinación del sistema según las variables de estado del sistema. Verifica si otros procesos han producido información para procesar o si ha que pasarles información para el procesamiento.
        c. un proceso es un subsistema o módulo que se ejecuta en paralelo con otros procesos
        d. utilizado en sistemas de tiempo real “suaves” (con restricciones de tiempo no muy estrictas)

   2.- control dirigido por eventos

- se rigen por eventos generados en el exterior
- diferentes tipos de sistemas dirigidos por eventos
    - ? hojas de cálculo (el valor cambiante de las celdas provoca que otras se modifiquen)
    - sistemas de producción basados en reglas (por ejemplo, de Inteligencia Artificial) en los que una condición que se convierte en verdadera provoca que se dispare una acción
    - objetos activos, en los que el cambio de valor de un atributo del objeto dispara algunas acciones
- dos tipos de modelos principales
    - **modelos de transmisión :** los subsistemas registran un interés en eventos específicos y cuando ocurren el control se transfiere al subsistema que puede manejar el evento
    - **modelos dirigidos por interrupciones** especialmente útiles para sistemas de tiempo real que necesitan manejar rápidamente eventos generados en el exterior

71. Patrones del diseño

R.- Son una identificación de mecanismos o colaboraciones genéricas. Utilización de clases abstractas.

72. Que es reutilización del software

R.-

- "Reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo" (Sametinger)
- La reutilización es un enfoque de desarrollo [de software] que trata de maximizar el uso recurrente de componentes de software existentes " (Sommerville).
- “La reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes ” (Sodhi)

73. Cuales son los activos del software

R.- Son productos de software diseñados expresamente para ser reutilizados en el desarrollo de muchas aplicaciones :

Un componente de software
Una especificación de requerimientos
Un modelo o especificación de diseño
Un algoritmo
Un patrón de diseño
Una arquitectura de dominio
Un esquema de base de datos
Una especificación de prueba
La documentación de un sistema
Un plan

- La reutilización de software va más allá de la reutilización de componentes de software. Involucra uso de otros elementos, denominados activos de software , tales como:
  algoritmos, diseños, arquitecturas de software, planes, documentación y especificaciones de requerimientos
- La reutilización de software es un proceso de la Ingeniería de Software que involucra el uso recurrente de activos de software en: la especificación, el análisis , el diseño, la implementación y las pruebas de una aplicación o sistema de software.

74. Que son CSR (Componentes de software reutilizable)

R.- Es un artefacto de software auto -contenido y claramente identificable que:

- ejecuta funciones específicas,
- tiene una interface clara a través de la cual se integra a otros sistemas,
- tiene una documentación apropiada y
- tiene un status de reuso definido

Sus características son:

- **Identificable.-** Debe tener una identificación clara y consistente que facilite su catalogación y acceso
- **Es autocontenido.-** Un componente no debe requerir la reutilización de otros componentes para cumplir su función
- **Es rastreable.-** Debe retener su identidad y ser rastreable durante el ciclo de desarrollo
- **Es reemplazable.-** Puede ser reemplazado por una nueva versión o por otro componente que proporcione los mismos servicios

- **Es accesible sólo a través de su interfaz.-** Es accedido a través de una interfaz claramente definida. La interfaz debe ser independiente de la implementación física(debe ocultar los detalles de su diseño interno)
- **Sus servicios son inmutables.-** Los servicios que ofrece un componente a través de su interfaz no deben variar. La implementación física de estos servicios pueden ser modificadas , pero no deben afectar la interfaz
- **Está documentado.-** Debe tener una documentación adecuada que facilite:
  la recuperación del componente desde el repositorio, la evaluación del componente, su adaptación al nuevo ambiente y su integración con otros componentes del sistema en que se reutiliza.
- **Es mantenido.-** Debe ser mantenido para facilitar un reuso sistemático:
  Su estado de reuso debe contener información acerca de:
  quien es el propietario del componente, quien lo mantiene, a quien acudir cuando surjan problemas y cual es el estado de la calidad del componente.

75. Que es Arquitectura de objetos distribuidos (CORBA)

R.-

- consiste en eliminar la distinción entre cliente y servidor y diseñar la arquitectura del sistema como una arquitectura de objetos distribuidos
- **los componentes fundamentales son:**
  - objetos que proveen una interfaz a un conjunto de servicios que suministran
  - otros objetos llaman a estos servicios sin ninguna distinción lógica entre un cliente (receptor de un servicio) y un servidor (proveedor de un servicio)
- **funcionamiento**
  - los objetos se distribuyen a lo largo de varios computadores sobre una red
  - se comunican a través de middleware (una especie de "bus de software" que provee un conjunto de servicios que permiten comunicación, agregación y destrucción de objetos del sistema
  - middleware: agente de solicitud de objetos (ORB, Object Request Broker) y provee una interfaz transparente entre objetos
- **ventajas**
  - permite retrasar las decisiones sobre dónde y cómo se deben suministrar los servicios pues los objetos proveedores de servicios se pueden ejecutar en cualquier nodo de la red
  - arquitectura abierta: permite agregar nuevos recursos si es necesario pues los estándares del ORB (p.ej., CORBA) se han desarrollado para permitir la comunicación y servicios entre objetos escritos en diferentes lenguajes
  - sistema flexible y escalable: se pueden crear diferentes instancias del sistema con el mismo servicio suministrado por objetos diferentes o por réplicas de objetos para hacer frente a diversas cargas del sistema
- **desventajas**
  - más complejas de diseñar que los sistemas cliente/servidor clásicos

76. Características del PUDS

R.-

1. **Dirigido por casos de uso.-**
   - Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactua con él
   - Casos de uso como el hilo conductor que orienta las actividades de desarrollo

2.- Centrado en la arquitectura.-

- Concepto similar a la arquitectura de un edificio.
  - Varios planos con diferentes aspectos del edificio .
  - Tener una imagen completa del edificio antes que comience la construcción
- Arquitectura en sofware
  - Diferentes vistas del sistema: estructural, funcional, dinámico, etc.
  - Plataforma en la que va a operar
  - Determina la forma del sistema
- Arquitectura: determina la forma del sistema
- Casos de uso: determinan la función del sistema

3.- Iterativo e incremental

- Descomposición de un proyecto grande en mini-proyectos
- Cada mini- proyecto es una iteración
- Las iteraciones deben estar controladas
- Cada iteración trata un conjunto de casos de uso
- Ventajas del enfoque iterativo:
  - Detección temprana de riesgos
  - Administración adecuada del cambio
  - Mayor grado de reutilización

- Mayor experiencia para el grupo de desarrollo

77. Ciclo de vida del PUDS

R.-