Respuestas a Cuestionario

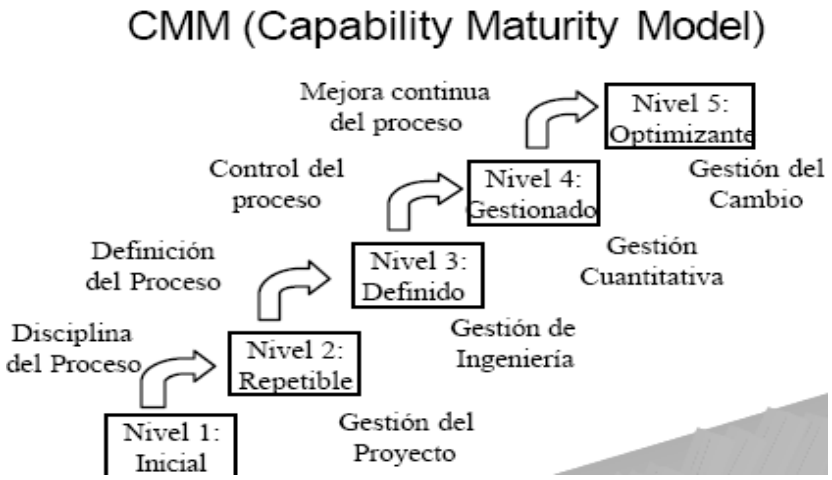
1. Cuales son las capas de la ingeniería del software

Herramientas: facilitan el trabajo
Métodos: como hacer posible la estrategia (proceso mas lenguaje).
Modelo de Proceso: solo procesos genéricos, estrategias ciclos de vida
Enfoque de calidad: siempre se aplica al principio y es la base de la ing. de Software

2. Para las métricas de punto de función describa los parámetros de medición

Parámetros	Descripción
N° de Entradas por el usuario	Las entradas, formularios, ventanas por donde el usuario deberá ingresar datos para interactuar con el sistema.
N° de salidas da das al usuario	Es la cantidad de salidas, como ser reportes, consultas donde el sistema devuelve datos al usuario.
N° de preguntas hechas al usuario	Es la cantidad de preguntas hechas al usuario, es decir mensajes, diálogos donde el sistema necesita que el usuario tome algún tipo de decisión para continuar con el procesamiento de datos.
N° de archivos	Es la cantidad de archivos con los que el sistema interactúa ya sea generándolos o manipulándolos.
N° de interfases externas	Es la cantidad de interfases externas con las que deberá interactuar el sistema, como por ejemplo impresora, lector de códigos de barras, al software u otro sistema con el cual necesita una conexión

3. Como clasifica el CMM a los desarrolladores de software según la madurez en la aplicación de la Ingeniería del Software



4. Si le dan la responsabilidad para la realización de las estimaciones de costo, tiempo y esfuerzo para un proyecto de software, cual seria el proceso que seguiría.

Lo primero es definir el proyecto, es decir definir los objetivos y el alcance del proyecto, mas el ámbito del proyecto, luego tener la lista de los requisitos del software y sus funciones principales; una vez se tengan estos datos se procede a la búsqueda de datos históricos (métricas) de proyectos similares para usarlos como base para la estimación, por lo menos se deberán encontrar 3 métricas de proyectos(optimista, esperado, pesimista), luego aplicar los métodos de estimaciones conocidos(LDC, PF, Cocomol y II) y por ultimo hacer una conciliación de los datos lanzados por las estimaciones, para así obtener una estimación de costo tiempo y esfuerzo para el proyecto a desarrollar.

5. En la gestión de proyectos a que se denomina las 4"p" (explique)

Personal: Es el factor humano y es fundamental para el éxito del proyecto.
Producto: Es lo que se desea construir o desarrollar.
Proceso: Es el conjunto de actividades a seguir para crear un producto.
Proyecto: Es el elemento organizativo de gestión, el proyecto construye el Producto (Patrón organizativo).

6. Describa la estrategia RAD
Este ciclo de vida como su nombre lo dice es para el desarrollo rápido de aplicaciones (entre 60 a 90 días). Consiste en descomponer el sistema en varios subsistemas independientes y para cada subsistema se le asigna un equipo de desarrollo de SW, donde los equipos trabajan de manera paralela, por lo general utilizan técnicas de 4GL.
Ventajas: Rápido, Sw Robusto.
Desventajas: Caro, debe haber bastante coordinación y gestión de comunicación.
El modelo DAD es recomendable utilizarlo cuando:

Cada proceso se puede particionar en módulos. Ej.: Sistema de Generación y almacenamiento de Documentos, módulo de generación de plantillas.
Existen funciones y componentes de apoyo para la implementación. Ej.: Sistema de registros, existen componentes para registrar datos específicos.
Tiempo para diseño es pequeño. Ej.: Se cuenta con el apoyo y recursos para diseñar en un corto tiempo.

8. Explicar el método de estimación basado en LDC
LCD.- para la estimación de líneas de código se debe primero identificar las funciones principales de SW, luego se deberá buscar 3 métricas de proyectos parecidos, donde una debe ser optimista, otra la esperada y la siguiente la mas probable. Una vez hecho esto se calcula las KLDC (Opt. + 4esp + pes)/6 para cada función luego se anotan los valores esperados de costo por línea y líneas/mes para poder calcular el costo y esfuerzo para el proyecto.
KLDC * Costo línea = Costo
KLDC/Líneas Mes = Personas Mes.

9. Bajo que criterios se debería decidir utilizar una determinada estrategia de desarrollo de software.
Para elegir una determinada estrategia de SW, se deberá tener encuenta los siguientes criterios.

1) Madurez de los desarrolladores
2) Conocimiento de los requisitos del SW
3) Tiempo de Desarrollo en proyectos similares
4) Madurez del Cliente
5) Grado de interacción posible entre el cliente y el desarrollador
6) Tiempo para desarrollar el SW.

10. Que haría durante el proceso de desarrollo para cumplir con los factores de calidad del ISO
Corrección: hacer seguimiento y control de los requisitos durante todo el desarrollo de SW para asegurar que se esta haciendo lo que se pidió.

Fiabilidad: Implementar mecanismos de seguridad, que permitan recuperar los datos en caso de fallas, implementar componentes redundantes para que reemplacen a otros componentes en caso de fallas, validación de datos.

Eficiencia: Numero reducido de subsistemas, con poco comunicación entre ellos tal vez sea necesario implantar en una sola capa para ahorro de recurso, seguimiento de estándares.

Usabilidad: seguimiento de estándares de diseño y de interfase, creación de manuales, realizar diseño de la arquitectura.
Mantenabilidad: Diseñar una arquitectura de tres capas, seguir estándares de codificación manual de diseño, descomposición en subsistemas y módulos.
Portabilidad: implementar en un lenguaje que no dependa de la plataforma, seguimiento de estándares, utilizar componentes ampliamente conocidos.

11. Es lo mismo calidad del software que SQA

No.
Calidad de Sw: grado con el cual un sistema cumple con los requerimientos especificados, y es el resultado de prácticas profesionales y el cumplimiento de estándares. (Producto)

SQA: Define estándares, métodos y herramientas, realiza revisiones y verificaciones, seguimientos auditorias es decir SQA se aplica al proceso.

12. Cuales son y que dicen los estándares para asegurar la calidad del software

IEEE STD 730: La guía IEEE 730.1 explica y clarifica los contenidos de un Plan de Aseguramiento de La Calidad del Software (o Software Quality Assurance Planning, en adelante SQAP)
Que satisfaga los requerimientos de IEEE Std 730-1989. Además explica que esta norma la guía IEEE 730.1 presenta el consenso de la comunidad de desarrolladores y personas dedicadas al mantenimiento del software, con sus conocimientos y experiencia en
Generar, implementar, evaluar y modificar un Plan de Aseguramiento de la Calidad del Software. Este SQAP debe describir los planes y actividades del personal encargado de asegurar la calidad de un producto software. El personal de Aseguramiento de calidad del Software observa el proceso de desarrollo informando de las deficiencias observadas en los procedimientos y en los productos resultado de ellos.

7. Explicar la diferencia entre el COCOMO básico y el COMOMO II

COCOMO II se dirige a las siguientes tres fases del ciclo de vida en espiral: desarrollo de Aplicaciones, diseño anticipado y Post-Arquitectura.

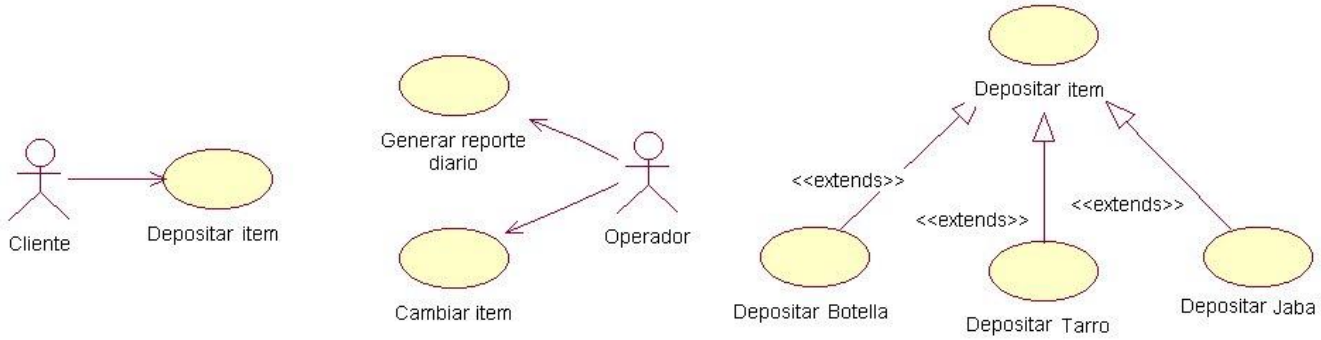
- Los tres modos del exponente se han reemplazado por cinco factores de escala.
- Se han añadido los siguientes drivers de coste a COCOMO II: DOCU, RUSE, PVOL, PEXP, LTEX, PCON y SITE.
- Se han eliminado los siguientes drivers de coste del modelo original: VIRT, TURN, VEXP, LEXP, Y MODP.

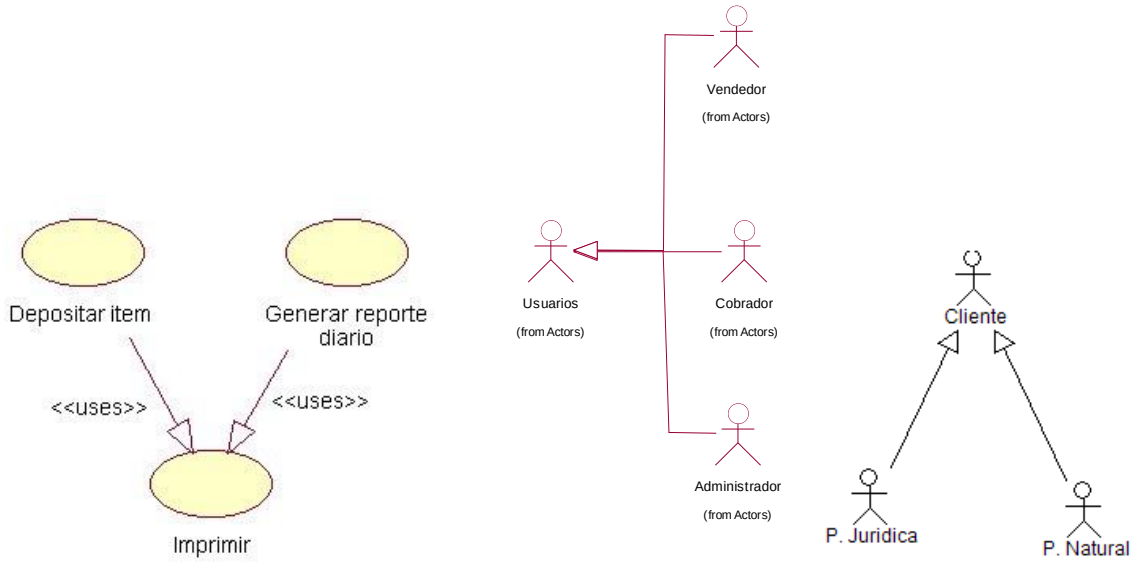
Los valores de aquellos drivers de coste que se han conservado del modelo original fueron modificados considerablemente para reflejar las calibraciones actualizadas.

Las otras diferencias principales se refieren a efectos relacionados con el tamaño, que incluyen reutilización, reingeniería y cambios en efectos de escala.

	COCOMO 81	COCOMO II
ESTRUCTURA DEL MODELO	Un modelo único que asume que se ha comenzado con unos requisitos asignados para el software	Tres modelos que asumen que se progresa a lo largo de un desarrollo de tipo espiral para consolidar los requisitos y la arquitectura, y reducir el riesgo.
FORMA MATEMATICA DE LA ECUACIÓN DEL ESFUERZO	Esfuerzo = A (q) (SIZE)^{Exponente}	Esfuerzo = A (q) (SIZE)^{Exponente}
EXPONENTE	Constante fija seleccionada como una función de modo: ▪ Orgánico = 1.05 ▪ Semi-libre = 1.12 ▪ Rígido = 1.20	Variable establecida en función de una medida de cinco factores de escala: ▪ PREC Precedencia ▪ FLEX Flexibilidad de desarrollo ▪ RESL Resolución de Arquitectura / Riesgos ▪ TEAM Cohesión del equipo ▪ PMAT Madurez del proceso
MEDIDA	Líneas de código fuente (con extensiones para puntos de función)	Puntos objeto, Puntos de función ó líneas de código fuente.
DRIVERS DE COSTE (c_i)	15 drivers, cada uno de los cuales debe ser estimado: ▪ RELY Fiabilidad ▪ DATA Tamaño Base de datos ▪ CPLX Complejidad ▪ TIME Restricción tiempo de ejecución ▪ STOR Restricción de almacenamiento principal ▪ VIRT Volatilidad máquina virtual ▪ TURN Tiempo de respuesta ▪ ACAP Capacidad del analista ▪ PCAP Capacidad programador ▪ AEXP Experiencia aplicaciones ▪ VEXP Experiencia máquina virtual ▪ LEXP Experiencia lenguaje ▪ TOOL Uso de herramientas software ▪ MODP Uso de Técnicas modernas de programación ▪ SCED Planificación requerida	17 drivers, cada uno de los cuales debe ser estimado: ▪ RELY Fiabilidad ▪ DATA Tamaño Base de datos ▪ CPLX Complejidad ▪ RUSE Reutilización requerida ▪ DOCU Documentación ▪ TIME Restricción tiempo de ejecución ▪ STOR Restricción de almacenamiento principal ▪ PVOL Volatilidad plataforma ▪ ACAP Capacidad del analista ▪ PCAP Capacidad programador ▪ AEXP Experiencia aplicaciones ▪ PEXP Experiencia plataforma ▪ LTEX Experiencia lenguaje y herramienta ▪ PCON Continuidad del personal ▪ TOOL Uso de herramientas software ▪ SITE Desarrollo Multi-lugar ▪ SCED Planificación requerida
OTRAS DIFERENCIAS DEL MODELO	Modelo basado en: ▪ Fórmula de reutilización lineal ▪ Asunción de requisitos razonablemente estables	Tiene muchas otras mejoras que incluyen: ▪ Fórmula de reutilización No-lineal ▪ Modelo de reutilización que considera esfuerzo necesario para entender y asimilar ▪ Medidas de rotura que se usan para abordar la volatilidad de requisitos ▪ Características de autocalibración

13. Hacer dos ejemplos de cada una de las relaciones posibles en un diagrama de casos de uso.





.

14. Como sugiere el PUDS distribuir el esfuerzo y el tiempo en una planificación de un proyecto de software

Tiempo:

Proyecto	Inicio	Elaboración	Construcción	Transición
Típicos	10%	30%	50%	10%
Complejos	20%	33%	40%	7%

Esfuerzo: 40% análisis y diseño 20 % implementación 40% pruebas

15. Cuando y como se desarrollar un modelo de negocio y un modelo de dominio
Modelo de Negocio: Es más indicado para proyectos con flujo de información frecuente y con muchos actores como ser: Sw de gestión se utilizan los diagramas de Casos de uso y actividades organizado en calles de UML

Modelo de Dominio: Se aplica a proyectos de Sw de entretenimiento, educativo, etc. Donde no existen flujos de información se utiliza el diagrama de clases de UML

19. Explicar el proceso de cada flujo de trabajo del PUDS
Captura de Requisitos: Primero se identifica actores y casos de uso, luego se hace una priorización de casos de uso, se detallan los casos de uso y se realiza el diagrama de casos de uso y por ultimo se realiza un prototipo de interfaces para el usuario.

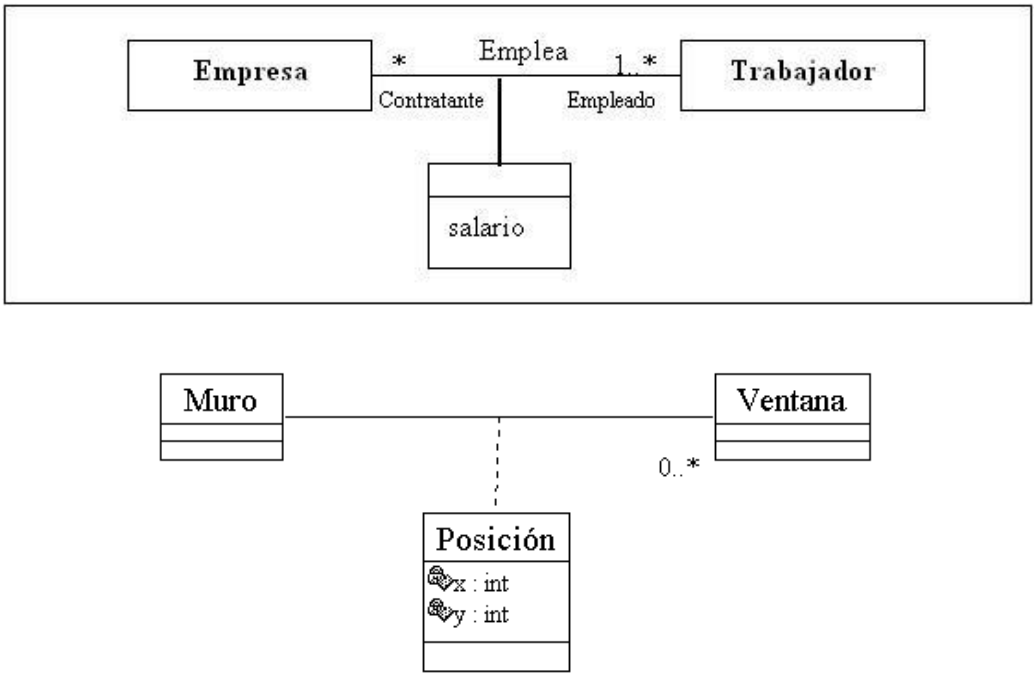
Análisis: Se realizar el análisis de la arquitectura, organizando los caso de uso en paquetes, luego se analiza los casos de uso (Diagrama de colaboración) luego se analizan las clases y por ultimo se analizan los paquetes (dependencias entre paquetes)

Diseño: se realiza el diseño de la arquitectura (Diagrama de Paquetes y despliegue) luego se diseña los casos de uso (Diagrama de iteración) luego las clases (diagrame de clases) y se diseña por ultimo los subsistemas.

Implementación: El arquitecto realiza la implementación de la arquitectura, luego se integra el sistema, se implementa un subsistema, se implementa una clase y se realizan las pruebas de unidad.

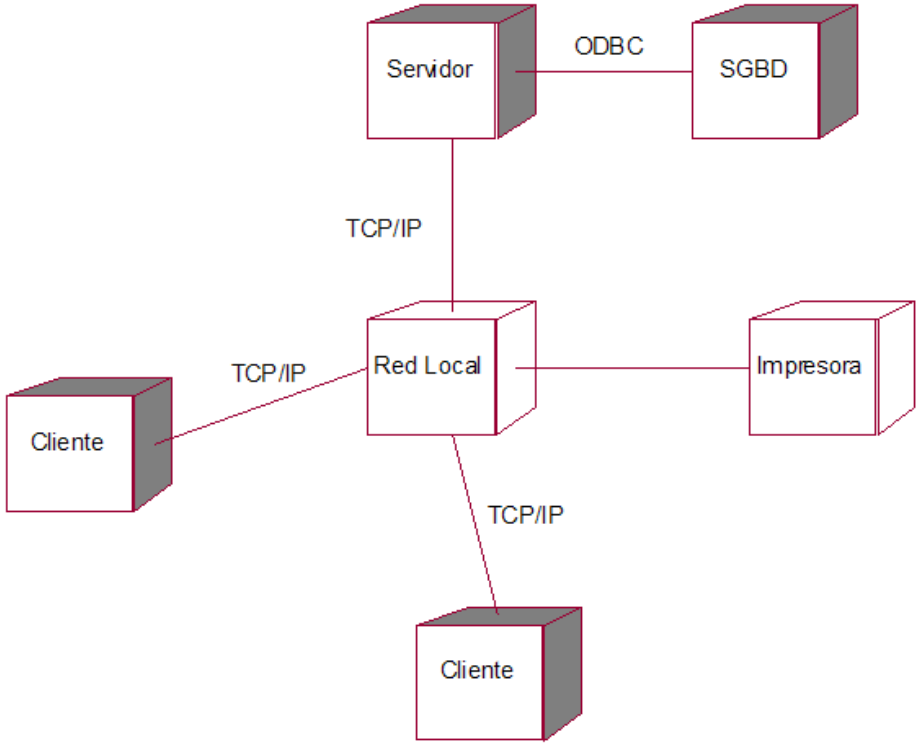
Pruebas el ing. de pruebas planifícalas las pruebas y las diseña el ing. de componentes implementa las pruebas y se realizan pruebas de sistema y pruebas de integración y por ultimo se evalúan las pruebas.

16. Cuando se aparece una clase asociación, realice dos ejemplos



17. Cual es el propósito del diagrama de despliegue. Haga un ejemplo

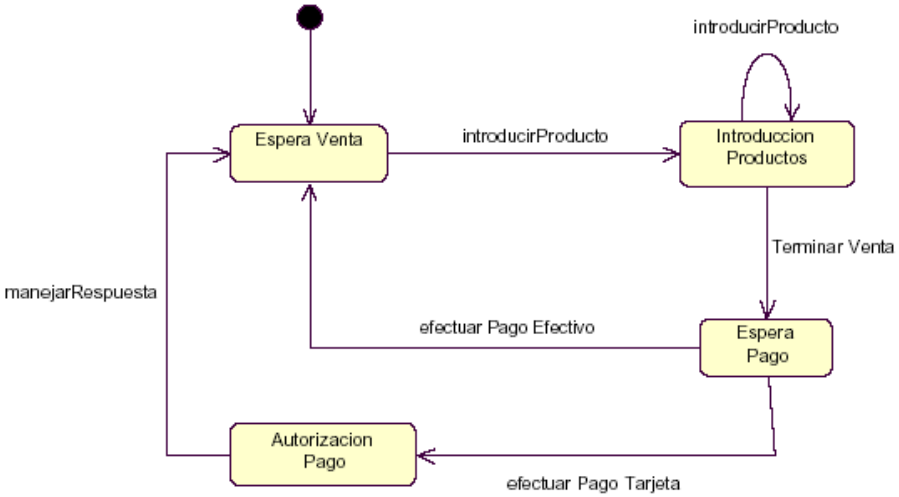
Su función es la de modelar la topología del H, es decir una visión estática de la arquitectura, donde se muestra la configuración de Nodos de procesamiento en tiempo de ejecución.



18. Cual es el propósito del diagrama de estados. Haga un ejemplo

Modelar el comportamiento por eventos, muestra el flujo de control entre estados, un evento es la especificaron de acontecimientos en tiempo y espacio.

Modela aspectos dinámicos del sistema

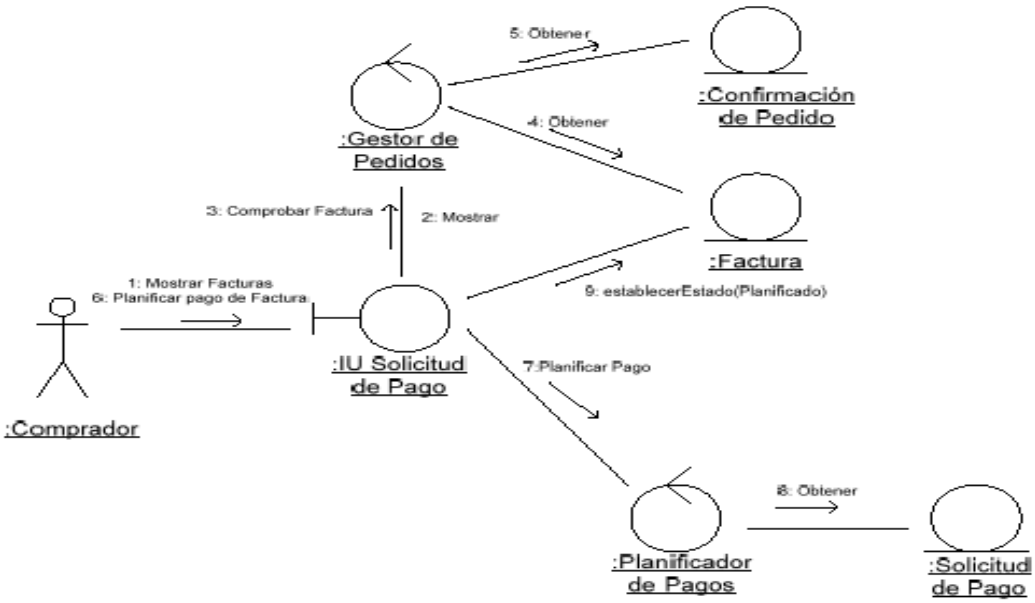


20. Explicar el modelo de diseño según la ingeniería del software

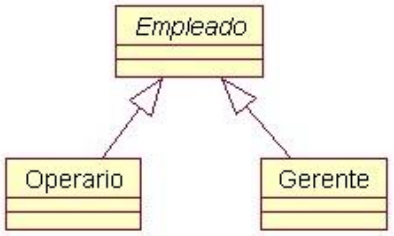
Está centrado en el como 3 paso concretos:
Diseño de SW: se traduce los requisitos a un conjunto de representaciones graficas que describen la estructura de Datos (BD), arquitectura.
Codificación: se Traduce el diseño a un lenguaje artificial.

Pruebas de SW: el SW debe ser probado para describir defectos que puedan existir.

21. Haga un ejemplo de un diagrama de colaboración para un determinado caso de uso, Utilizando clases del análisis del PUDS



22. De ejemplos de clase base y clase hoja en UML



Clase Base

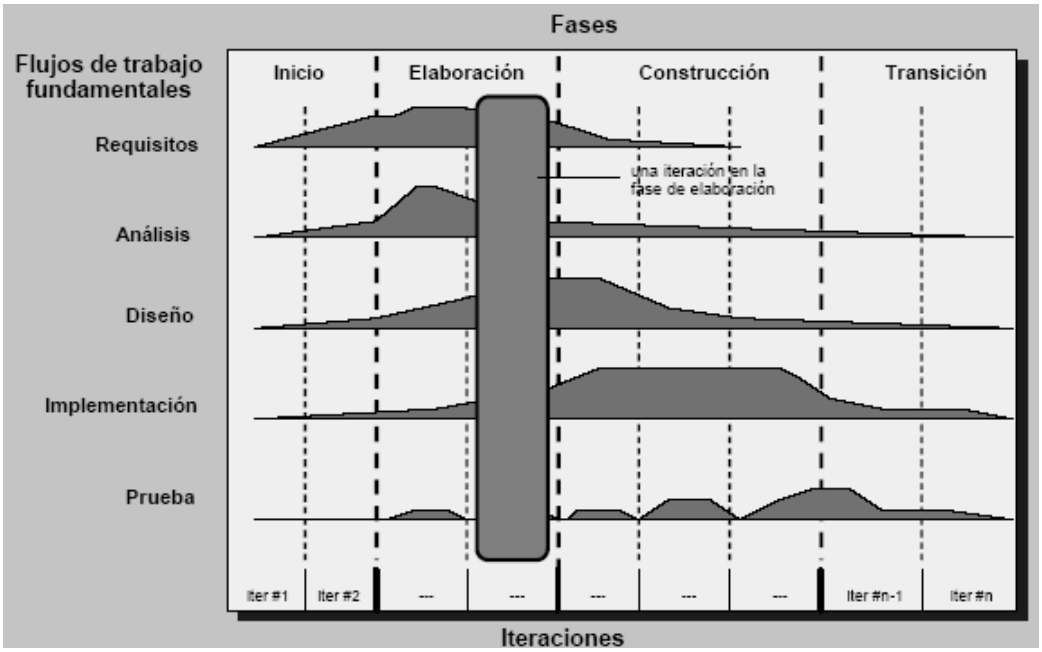
Clases Hoja

23. En un diagrama de actividad de UML a que se denomina "estado de acción" y "estado De actividad"
Estado de Acción: Son atómicos puede ocurrir eventos pero no se interrumpe la ejecución de un estado de acción, la ejecución de un estado de acción con lleva un tiempo insignificante, un estado de acción representa la ejecución de una acción del sistema, Ej. Evalúa un expresión, pasar mensajes a objetos, crear o destruir un objeto.
Un estado de Acción es un estado de actividad que no se puede descomponer.

Estado Actividad: se puede descomponer, es decir se puede representar su actividad con otros diagramas de actividad los estados de actividad no son atómicos, pueden ser interrumpidos, llevan un cierto tiempo en completarse.
No existe diferencia en la notación entre estos dos estados.

24. Explicar la tecnología estratificada (multicapa, visión de la Ingeniería de Software).
Ver la pregunta 1

25. Representar gráficamente y describir completamente el PUDS incluyendo objetivos de cada fase



Fase	Objetivo
Inicio	Definir el alcance del proyecto
Elaboración	Planificar el proyecto elaborar una arquitectura base
Construcción	Construir el Sistema
Transición	Transición a los usuarios

PUDS: esta dirigido por caso de uso, es centrado en la arquitectura y es iterativo e incremental.
Casos de uso determina lo que hace el sistema
Arquitectura: determina la forma del sistema

26. Para las métricas orientadas al tamaño y orientadas a la función diseñe una base de datos que permita almacenar datos históricos de proyectos ya desarrollados utilizando dichas métricas.
Esta en tu cuaderno

27. Suponga que es el gestor de un proyecto de desarrollo de un SGBD con interfaces graficas y le piden que elabore el la tabla de contenido del plan de proyecto correspondiente.

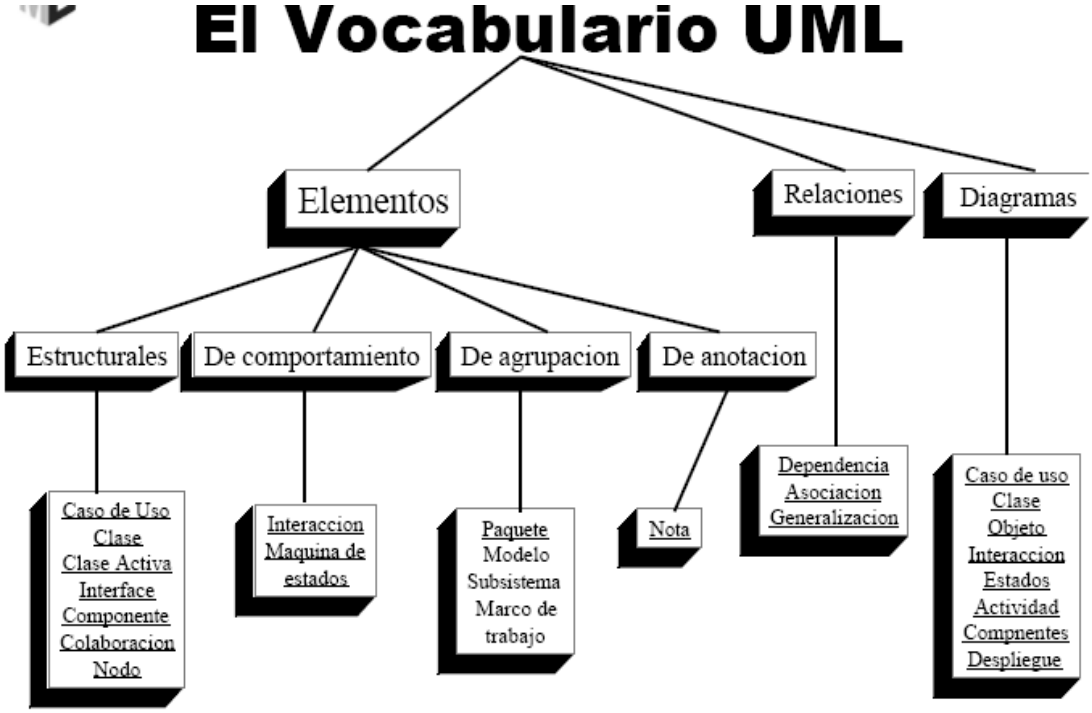
Contenido de un Documento de Planeación de Proyectos de Software

1. Introducción
   - 1.1 Propósito del documento
   - 1.2 Identificación del problema
   - 1.3 Objetivos Generales
   - 1.4 Funciones principales el proyecto
   - 1.5 Limitantes técnicas y administrativas
2. Estimación de recursos
   - 2.1 Datos históricos utilizados para estimar
   - 2.2 Técnicas de estimulación
   - 2.3 Estimaciones
   - 2.3.1 Esfuerzo

2.3.2 Costos
3. Cronograma
3.1 Diagrama PERT
3.2 Diagrama de Gantt
3.3 Tabla de recursos
4. Recursos del proyecto
4.1 Personal Involucrado
4.2 Contratos externos
4.3 Hardware
4.4 Software
4.5 Recursos especiales
5. Organización del equipo de trabajo
5.1 Estructura del equipo
5.2 Reportes de Gestión.
5.3 Mecanismos de control

28. Para la pregunta cuatro se le piden que haga las estimaciones de tiempo, costo y esfuerzo. Explique paso a paso el proceso que seguiría para cumplir con este pedido.
Ver la pregunta 4

29. Represente gráficamente el vocabulario de UML



30. Describa el proceso de captura de requisitos según el PUDS
Ver pregunta 19

31. Realice dos ejemplos de cada tipo de relación utilizando casos de uso
Ver pregunta 13

32. (10) Explique los modelos de negocio y modelo de dominio
Ver pregunta 15

33. Como determina el tamaño y la complejidad de un proyecto de software
Aplicando estimaciones.

34. Para que tipos de proyectos de software es más apropiado utilizar el método Estructurado y para que proyectos es mas apropiado el OO, proponga ejemplos
Ni idea

35. Como determina la probabilidad y como calcula el impacto para los riesgos
Probabilidad = se establece una escala, luego se define las consecuencia del riesgo con esto se estima el impacto, la determinación de la probabilidad de ocurrencia de cada evento se realiza viendo el riesgo de proyectos anteriores y la experiencia del equipo de desarrollo.
Calculo del Impacto: se toma encuenta tres factores:
La naturaleza del riesgo.- que indica los problemas probables que aparecerán
Alcance del riesgo.- Combina la severidad con su distribución.

Temporizacion del riesgo.- considera cuando y por cuanto tiempo se deja sentir el impacto.

36. Para la planificación temporal en función a que define las actividades a seguir durante el desarrollo
Ver pregunta 20

37. Mencione cuales son los puntos que contiene un plan de proyecto de software
Ver pregunta 27

38. Quienes lo conforman y cuales son las actividades del grupo de SQA.
Actividades: Definir estándares, métodos y herramientas, realizar revisiones y verificaciones ajustes a estándares establecidos, seguimiento de cambios mediciones y auditorias, registro y realización de informes.

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

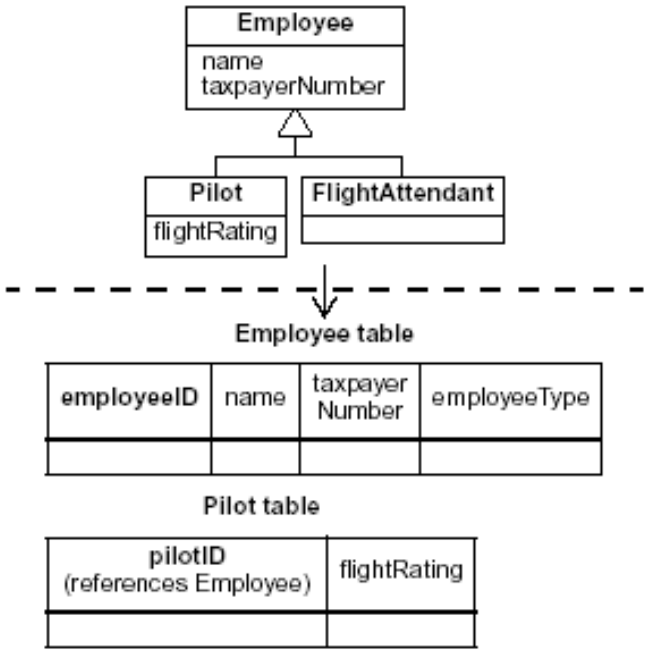
43. Haga un ejemplo donde se vea un diseño con problemas de acoplamiento y cohesión y otro ejemplo libre de esos problemas
R.-

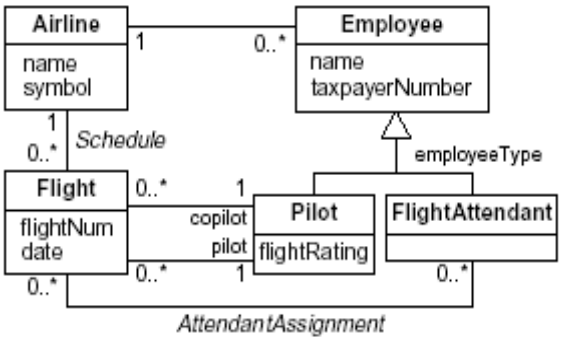
44. Como representa el diseño modular con UML (haga un ejemplo)
R.-

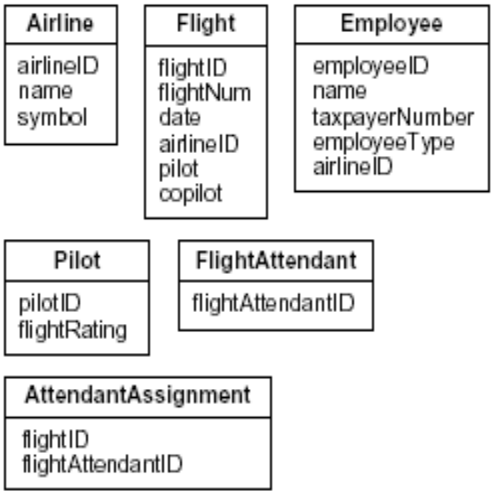
45. Para un desarrollo de software siguiendo el método OO que ciclo de vida utilizaría y por que
R.-

46. Haga un ejemplo de clases utilizando agregación y realice el mapeo a BD relacionales
R.-

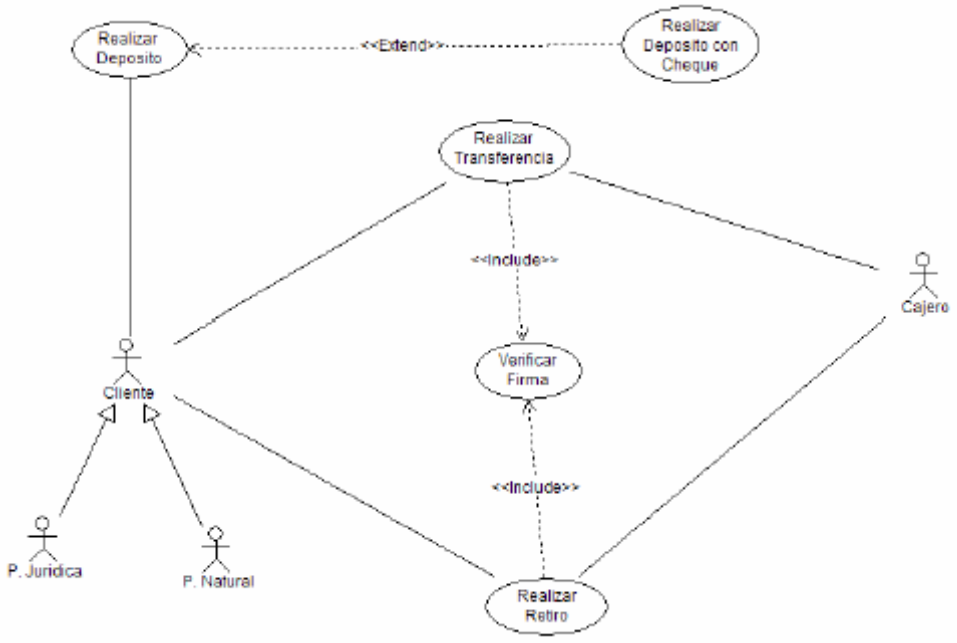
47. Con UML como modelaría una estructura de generalización / especialización y luego haga el mapeo a una BD Relacional (haga un ejemplo)





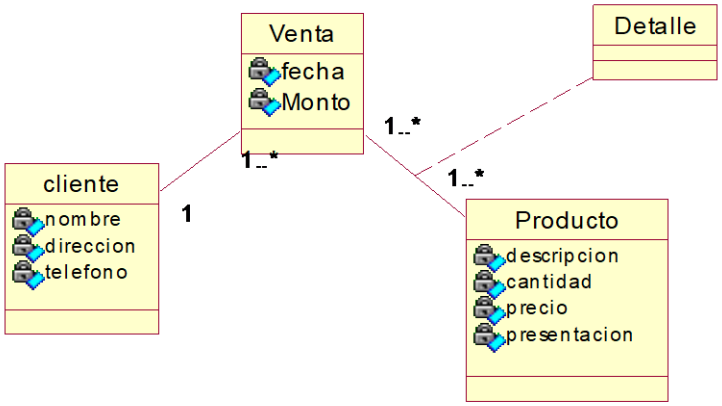


48. En un diagrama de casos uso muestre el uso de generalización / especialización



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

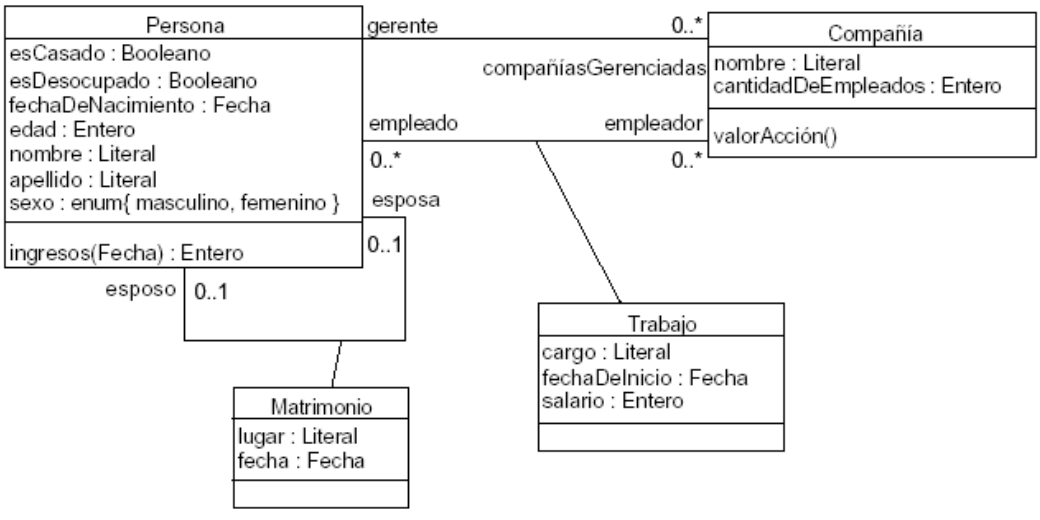
50. Realice un diagrama de clases para el caso de uso realizar venta de productos en una Importadora



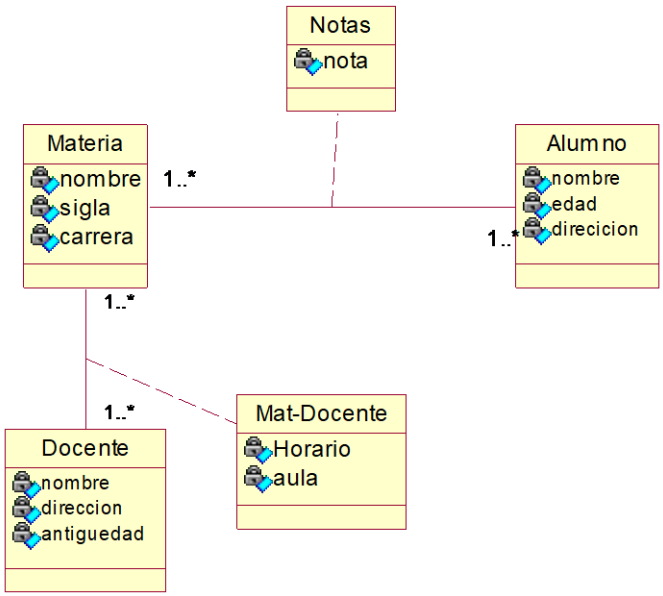
51. Para el caso de uso Depositar dinero de una cuenta corriente de un banco haga el diagrama de secuencia correspondiente.

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

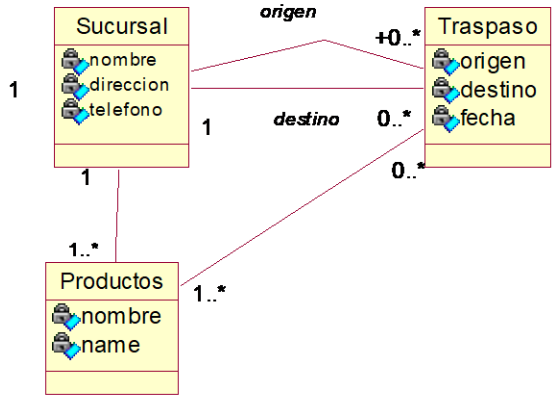
R.-



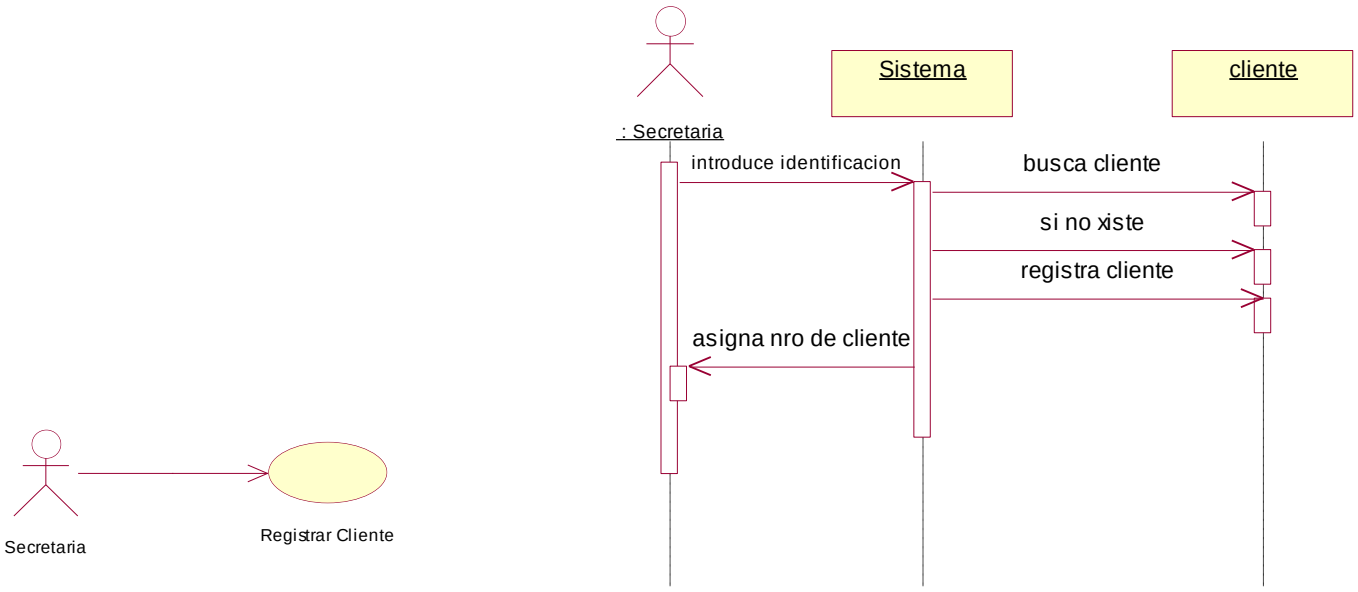
53. Haga un diagrama de clases para el registro de notas de alumnos, considere que un alumno lleva mas de una materia, y que una misma materia puede ser dictada por mas de un docente.
R.-



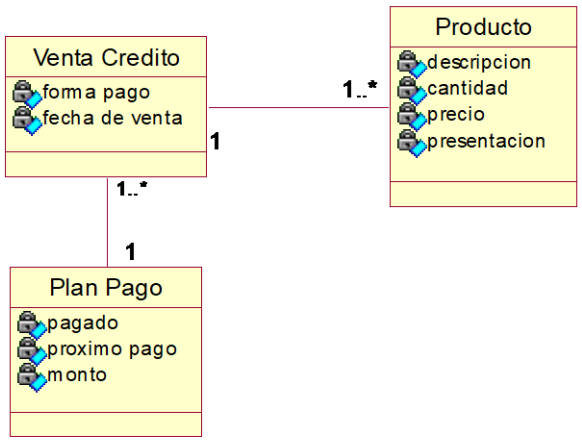
54. Haga un diagrama de clases para el registro de traspaso de productos entre sucursales de una importadora
R.-



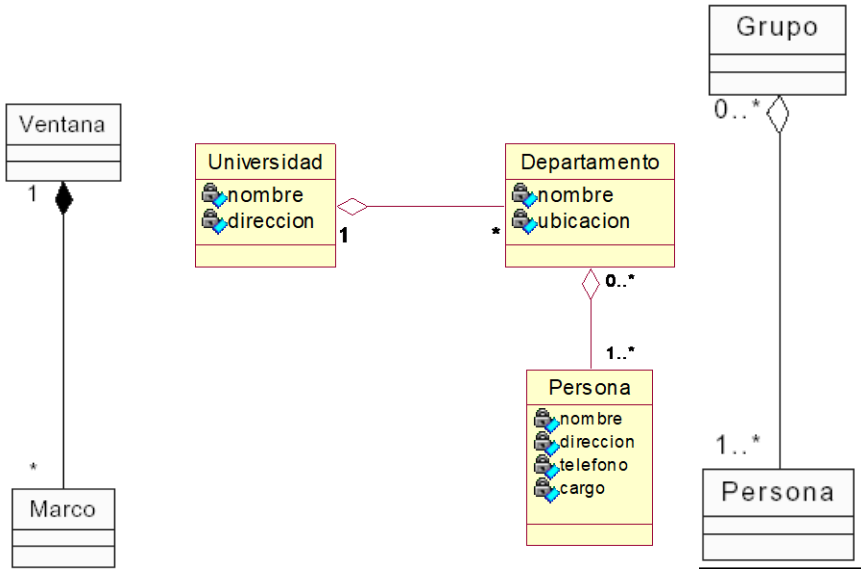
55. El registro de clientes en una importadora lo realiza la secretaria de dicha importadora, haga su diagrama de casos de uso y su respectivo diagrama de secuencia
R.-



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

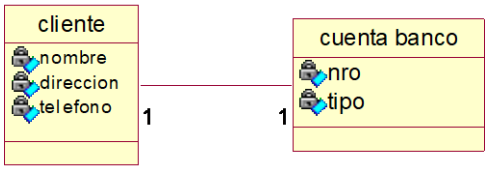


57. Realice ejemplos utilizando agregación y composición entre clases
R.-

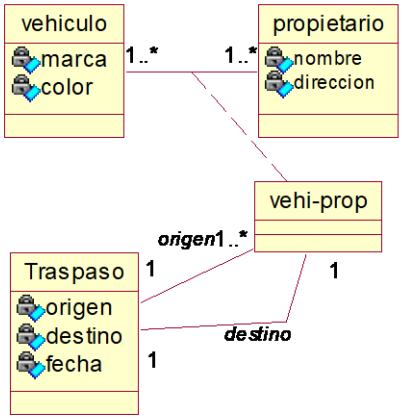


58. Bajo que criterios decide usar los estereotipo <<uses>> y <<extiende>> en un diagrama de casos de uso
R.- <<uses>> .- Cuando un Caso de uso necesita de la funcionalidad de otro.
<<extiende>>.- Cuando la realización de un caso de uso puede usar o no otro.

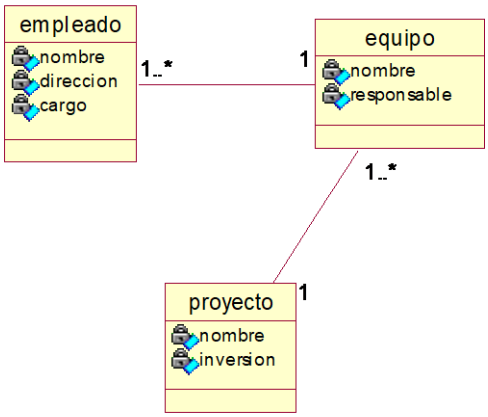
59. Haga ejemplos de asociaciones (1 – 1) , (0 – 1), (1 a muchos), (0 a muchos) y (muchos a muchos) usando clases
R.-



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.-

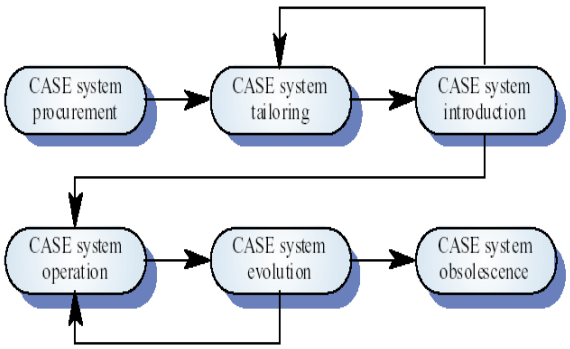


62. Taxonomia de las herramientas CASE

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.-