PRIMER PARCIAL SISTEMAS DE INFORMACION
TEORÍA GENERAL DE SISTEMAS

1. **¿Qué se entiende por sistema?**
   Un sistema es un conjunto de partes o elementos organizados y relacionados que interactúan entre sí para lograr un objetivo. Los sistemas reciben entrada de datos, energía o materia del ambiente y proveen salida de información, energía o materia.
   Un sistema puede ser físico o concreto (una computadora, un televisor, un humano) o puede ser abstracto o conceptual (un software).

2. **Explique las características de los sistemas**
   - *Elementos:* La interacción e interdependencia de sus elementos.
   - *Objetivos:* Para la permanencia del sistema éste busca definir un sentido de unidad y propósito. La persecución de un objetivo o finalidad. Los sistemas deben considerarse como un todo compuesto de partes que a su vez pueden ser subsistemas.
   - *Entradas:* Son los ingresos del sistema que pueden ser recursos materiales, recursos humanos o información. Constituyen la fuerza de arranque que suministra al sistema de sus necesidades operativas.
   - *Proceso:* Es lo que transforma una entrada en salida (máquina, un individuo, una computadora, un producto químico, una tarea realizada por un miembro de la organización, etc.).
   - *Salidas:* Son los resultados que se obtienen de procesar las entradas, (productos, servicios e información). Son el resultado del funcionamiento del sistema, o, alternativamente el proceso para el cual existe el sistema.

3. **¿Cómo considera a la organización como sistema?**
   La organización se la considera como los siguientes:
   - La organización se debe enfocar como un sistema que se caracteriza por todas las propiedades esenciales a cualquier sistema social.
   - La organización debe ser abordada como un sistema funcionalmente diferenciado de un sistema social mayor.
   - La organización debe ser analizada como un tipo especial de sistema social, organizada en torno de la primacía de interés por la consecución de determinado tipo de meta sistemática.
   - Las características de la organización deben ser definidas por la especie de situación en que necesita operar, consistente en la relación entre ella y los otros subsistemas, componentes del sistema mayor del cual parte. Tal como si fuera un sociedad.

4. **Explique los subsistemas que forman a la organización como sistema**
   a. Subsistema psicosocial: Está compuesto por individuos y grupos en interacción. Dicho subsistema está formado por la conducta individual y la motivación, las relaciones del status y del papel, dinámica de grupos y los sistemas de influencia.
   b. Subsistema técnico: Se refiere a los conocimientos necesarios para el desarrollo de tareas, incluyendo las técnicas usadas para la transformación de insumos en productos.
   c. Subsistema administrativo: Relaciona a la organización con su medio y establece los objetivos, desarrolla planes de integración, estrategia y operación, mediante el diseño de la estructura y el establecimiento de los procesos de control.

5. **¿Qué se entiende por límites o fronteras de los sistemas?**

   La frontera de un sistema define qué es lo que pertenece al sistema y qué es lo que no. Lo que no pertenece al sistema puede ser parte de su suprasistema o directamente no ser parte.

   Establecer el límite de un sistema puede ser sencillo cuando hay límites físicos reales y se tiene bien en claro cuál es el objetivo del sistema a estudiar. Por ejemplo, el sistema digestivo humano incluye solo los órganos que procesan la comida. En cambio los límites son más difíciles de establecer cuando no es claro el objetivo o se trata de un sistema lógico o conceptual.

   Las fronteras de los sistemas también nos permiten establecer jerarquías entre subsistemas, sistemas y supersistemas.

6. **¿Cuáles son los elementos de un sistema?**

   a. Corriente de entrada: Se conoce que los sistemas abiertos para que puedan funcionar, deben importar a través de su corriente de entrada, ciertos recursos del medio, tales como insumos o energía, que permiten su funcionamiento y manutención.
   b. Proceso de conversión: Es el fenómeno que produce los cambios, en los mecanismos de conversión de entradas en salidas. Caracteriza de los sistemas y se define como la totalidad de los elementos, empeñados en la producción de un resultado. El proceso se presenta generalmente como la caja negra, en ella entran insumos y de ella salen elementos diferentes que son los productos, se pueden hacer ciertas deducciones a partir de las observaciones controladas.
   c. Corriente de salida: Equivale a la exportación que hace en el medio. Es la finalidad para la cual se reunieron elementos y relaciones del sistema. Los resultados de un sistema son las salidas. Estos deben ser congruentes (coherentes) con el objetivo del sistema. Los resultados son finales (concluyentes), mientras que los resultados del subsistema son intermedios.
   d. La comunicación de retroalimentación como elemento de control: Recordemos que todo sistema tiene algún propósito y la conducta que desarrolla, una vez que dispone de la energía suficiente prevista por sus corrientes de entrada, tiende a alcanzar ese propósito u objetivo. Así la comunicación de retroalimentaciones la información que indica como lo está haciendo el sistema en la búsqueda de su objetivo, y que introducido nuevamente al sistema con el fin de que se lleven a cabo las correcciones necesarias para lograr el objetivo.

7. **¿En qué se fundamenta la teoría general de sistemas?**

   La teoría general se sistemas se fundamenta en tres premisas básicas:

   - Los sistemas existen dentro de sistemas.
   - Los sistemas son abiertos.
   - Las funciones de un sistema dependen de su estructura.

8. **¿Qué se entiende por datos?**

   Corresponden a hechos que pueden procesarse para producir una información exacta y relevante.

9. **¿Qué se entiende por información?**

   Datos que se procesan y operan en una computadora para darle valor y significado.

10. Explique las categorías de la información

a. Estratégica.
   - Información estratégica es un instrumento de cambio.
   - Enfocada a la planeación a largo plazo
   - Orientada a la alta administración.

b. Táctica.
   - Información de control administrativo
   - Es un tipo de información compartida.
   - Tiene una utilidad a corto plazo.

c. Operacional.
   - Información rutinaria.
   - Muestra la operación diaria.
   - Tiene una utilidad a muy corto plazo.

11. ¿Cuáles son los atributos de la información?

- *Componentes:* Partes inidentificables del sistema.
- *Atributos:* Son las características que influyen en la operación del sistema.

Entre los términos de los componentes y atributos de los sistemas tenemos que definir lo que es la estructura de sistemas; la cual no es más que la forma o manera como se relacionan los componentes y atributos para obtener un fin común.

Ahora bien; existen diferentes relaciones que son dadas según a componentes y atributos las cuales definiré a continuación:

- *Relaciones disfuncional:* Son organizaciones donde todas las partes no funcionan o tienen conflictos.
- *Relaciones parasitarias:* Es cuando los sistemas se aprovechan de otros.
- *Relaciones simbióticas:* Cuando ambos sistemas se benefician.
- *Relaciones sinérgicas:* Es cuando los sistemas se esfuerzan entre sí para obtener beneficios comunes.
- *Relaciones optimizadas:* Son aquellos sistemas donde hay intercambios de recursos y objetivos entre los sub-sistemas manteniendo en equilibrio dinámico para optimizar la salida del sistema.

12. Definir conceptualmente y con ejemplos de sistemas de información

a. Conglomerado

Un conglomerado, se supone sin sinergia, es decir, que la suma de sus partes es igual al todo.

Podemos definir al conglomerado como un conjunto de objetos, de los cuales abstraemos ciertas características, es decir, eliminamos aquellos factores ajenos al estudio y luego observamos es comportamiento de las variables que nos interesan.

Será un conglomerado si las posibles relaciones que entre ellos se desarrollan no afectan la conducta de cada una de las partes.

b. Supersistema

Un supersistema es el sistema que integra a los sistemas desde el punto de vista de pertenencia.

En otras palabras, es un sistema mayor que contiene los sistemas menores.

c. **Subsistema**
En general, podemos señalar que cada una de las partes que encierra un sistema puede ser considerada como subsistema, es decir, un conjunto de partes en interrelaciones que se encuentra estructuralmente y funcionalmente, dentro de un sistema mayor, y que posee sus propias características. Así los subsistemas son sistemas más pequeños dentro de sistemas mayores.

d. **Complejidad**
Un sistema tiende a ser más complejo cuando tanto las interacciones y la variedad aumentan. Nótese que no se hace referencia al número de partes o subsistemas, sino al número de las interacciones posibles.

13. **¿Qué es entropía? Mencione tres ejemplos**
El cambio de estados más ordenados u organizados a estados menos ordenados y organizados, es una cantidad definida y medible, denominada 'Entropía'. Es el factor que explica el hecho de que mientras la energía total contenida en un sistema cerrado permanece constante, con el incremento de la entropía, esa energía puede ser utilizada cada vez menos.

14. **¿Qué diferencia existe entre recursividad y conglomerado?**
Un conglomerado es un conjunto de objetos desprovisto de sinergia y la recursividad es el hecho de que un sistema sinergético este compuesto de partes con características que son a su vez sinergéticas.

15. **¿Qué es la sinergia?**
Cuando la suma de las partes es diferente del todo; cuando un objeto cumple con este principio o requisito decimos que posee o existe sinergia.
Un objeto posee sinergia cuando el examen de una o alguna de sus partes (incluso a cada una de sus partes) en forma aislada, no puede explicar o predecir la conducta del otro.

16. **Describir que es energía entrante, energía saliente, incremento de la información. Dar dos ejemplos de cada uno**
Para que los sistemas abiertos puedan funcionar, deben importar ciertos recursos del medio, para comprender todos estos insumos usamos el término de 'energía'. Por lo tanto, los sistemas a través de su corriente de entrada, reciben la energía necesaria para su funcionamiento y mantención.
La energía saliente equivale a la exportación que el sistema hace al medio. Generalmente no existe una sino varias corrientes de salida. Podemos dividir estas corrientes de salida como positivas y negativas para el medio y el entorno, entendiéndose aquí por medio todos aquellos otros sistemas (o supersistemas) que utilizan de una forma u otra la energía que exporta ese sistema.
En general, la energía que importa el sistema del medio tiende a comportarse de acuerdo con la ley de la conservación, que dice que la cantidad de energía que permanece en un sistema es igual a la suma de la energía importada, menos la suma de la energía exportada, sin embargo, la información no responde a esta ley de conservación. Efectivamente, el sistema importa información desde su medio a través de sus centros receptores y canales de comunicación, pero no podemos decir que la cantidad d información que se mantiene dentro de un sistema es igual a la suma de las informaciones

que entran menos la suma de las informaciones que salen, ya que en este caso, la información se comporta de acuerdo a lo que se ha denominado 'ley de los incrementos' que dice que la cantidad de la información que permanece en el sistema es igual a la información que existe más la que entra, es decir, hay una agregación neta en la entrada y la salida no elimina información del sistema.

17. Explique la diferencia entre sistemas naturales y sistemas hechos por el hombre

Un sistema natural tiene una lógica integral, funcional y conjunta, es una lógica de carácter ecológico. Este sistema ecológico tiene por objetivo conformar, equilibrar un sistema de supervivencia natural; es un sistema de vida y de evolución funcionando en y con el entorno físico.

Un sistema hecho por el hombre es una variable dependiente de un sistema social

18. Explique los niveles de sistemas

- *Primer nivel:* Estructuras estáticas (ejemplo: el modelo de los electrones dentro del átomo).
- *Segundo nivel:* Sistemas dinámicos simples (ejemplo: el sistema solar).
- *Tercer nivel:* Sistemas cibernéticos o de control (ejemplo: el termostato).
- *Cuarto nivel:* Los sistemas abiertos (ejemplo: las células).
- *Quinto nivel:* Genético social (ejemplo: las plantas).
- *Sexto nivel:* Animal.
- *Séptimo nivel:* El hombre.
- *Octavo nivel:* Las estructuras sociales (ejemplo: una empresa).
- *Noveno nivel:* Los sistemas trascendentes (ejemplo: lo absoluto).

19. Describir por grupo la clasificación de los sistemas

En cuanto a su constitución, los sistemas pueden ser físicos o abstractos:

a. Sistemas físicos o concretos: Están compuestos por equipos, por maquinaria y por objetos y cosas reales. Pueden ser descritos en términos cuantitativos de desempeño.
b. Sistemas abstractos: Están compuestos por conceptos, planes, hipótesis e ideas. Aquí, los símbolos representan atributos y objetos, que muchas veces sólo existen en el pensamiento de las personas.

En cuanto a su naturaleza, los sistemas pueden ser cerrados o abiertos:

a. Sistemas cerrados: Son los sistemas que no presentan intercambio con el medio ambiente que los rodea, pues son herméticos a cualquier influencia ambiental. No reciben ninguna influencia del ambiente, y por otro lado tampoco influencian al ambiente. No reciben ningún recurso externo.
b. Sistemas abiertos: Son los sistemas que presentan relaciones de intercambio con el ambiente, a través de entradas y salidas. Intercambian materia y energía regularmente con el medio ambiente.

20. Defina retroalimentación

La retroalimentación, es un mecanismo de control de los sistemas dinámicos por el cual una cierta proporción de la señal de salida se redirige a la entrada, y así regula su comportamiento.

21. Definir conceptualmente y con ejemplos

a. Enfoque reduccionista

Este enfoque estudia un fenómeno complejo a través del análisis de sus elementos o partes componentes. En este enfoque se trata de explicar que las ciencias o sistemas para su mejor entendimiento divididos a un grado tan elemental, separados de tal modo que facilitaran su estudio a un nivel tan especializado.

Como ejemplo podemos citar la biología, divididos por ejemplo en citobiología, microbiología o la virología, que son ciencias más especializadas de la biología.

Este enfoque busca desmenuzar tanto como se pueda, lo que se este estudiando.

El enfoque reduccionista busca estudiar un fenómeno complejo, reduciéndolo al estudio de sus unidades constitutivas de modo que podamos explicar el fenómeno complejo a través del estudio individual de uno de sus constituyentes.

El enfoque antagónico a este es de la generalización o totalitario, que busca entender al sistema o fenómeno complejo como un todo único.

En muchos casos este enfoque es rechazado porque al extraer, al menos de manera parcial, un objeto o situación particular del contexto que lo comprende y con el que interactúa puede que no se logre comprender la situación en su totalidad.

Este enfoque ha permitido el crecimiento de muchas ciencias y que ha permitido el estudio de un fenómeno complejo a través del análisis de sus elementos o partes componentes.

Pero existen fenómenos que solo pueden ser explicados tomando en cuenta el todo que los comprende y del que forman parte a través de su interacción.

El enfoque de sistemas pretende integrar las partes hasta alcanzar una totalidad lógica o de una independencia o autonomía relativa con respecto a la totalidad mayor de la cual también forma parte. No solo es necesario definir la totalidad sino también sus partes constituyentes y las interacciones de estas.

b. Enfoque estructuralista

El Enfoque de Sistemas, es una forma ordenada de evaluar una necesidad humana de índole compleja y consiste en observar la situación desde todos los ángulos y determinar:

- Los elementos distinguidos en el problema.
- La relación de causa y efecto que existe entre ellos.
- Las funciones específicas que cumplen en cada caso.
- Los intercambios que se requerirán entre los recursos una vez que se definan.

El enfoque de sistemas concibe la organización como un sistema unido y dirigido de partes interrelacionadas que tienen un propósito y está compuesto por partes que se interaccionan. Plantea que la actividad de un segmento de la organización afecta en diferentes grados la actividad de todos sus segmentos.

Uno de sus supuestos básicos del enfoque de sistemas es que las organizaciones no son autosuficientes, intercambian recursos con el ambiente externo definido, éste como todos los elementos extraños a la organización que son relevantes para sus operaciones.

Considera que la organización institucional, es un sistema que se conforma por subsistemas donde se sigue un proceso de transformación hasta obtener un resultado, el cual debe estar en constante retroalimentación; todo ello a través de la interacción de las partes que se consideran como subsistemas, donde cada departamento o servicio coopera e interactúa con funciones y actividades específicas que conllevan al logro de un objetivo general dentro de la institución.

Destaca la esencia dinámica y las interrelaciones de las organizaciones y el quehacer administrativo, ofreciendo un marco que permita planificar las acciones y en la mayoría de los casos adelantarnos a las consecuencias inmediatas, mediatas o inesperadas cuando se presentan.

Con el enfoque de sistemas los directores de las instituciones pueden conservar con más facilidad el equilibrio entre las necesidades de los distintos servicios que conforman la organización y los requerimientos de esta en su conjunto. La comunicación no sólo es entre empleados y departamentos, sino también y con frecuencia, con representantes de otras organizaciones.

PARADIGMAS DEL DESARROLLO DE SOFTWARE

1. **Describa detalladamente el paradigma Recursivo Paralelo**
   - **Definición y Características**

     El modelo recursivo paralelo funciona de la siguiente forma:
     - Realizar los análisis suficientes para aislar las clases del problema y las conexiones más importantes.
     - Realizar un pequeño diseño para determinar si las clases y conexiones pueden ser implementadas de manera práctica.
     - Extraer objetos reutilizables de una biblioteca para construir un prototipo previo.
     - Conducir algunas pruebas para descubrir errores en el prototipo.
     - Obtener realimentación del cliente sobre el prototipo.
     - Modificar el modelo de análisis basándose en lo que se ha aprendido del prototipo, de la realización del diseño y de la realimentación obtenida del cliente.
     - Refinar el diseño para acomodar sus cambios.
     - Construir objetos especiales (no disponibles en la biblioteca)
     - Realizar pruebas para descubrir errores en el prototipo.

     Este enfoque continua hasta que el prototipo evoluciona hacia una aplicación en producción.
   - **Ventajas**

     El progreso se produce iterativamente. Lo que hace diferente al modelo recursivo/paralelo es el reconocimiento de que:
     - El modelo de análisis y diseño para sistemas orientado a objetos no puede realizarse a un nivel uniforme de abstracción.
     - El análisis y diseño pueden aplicarse a componentes independientes del sistema de manera concurrente.

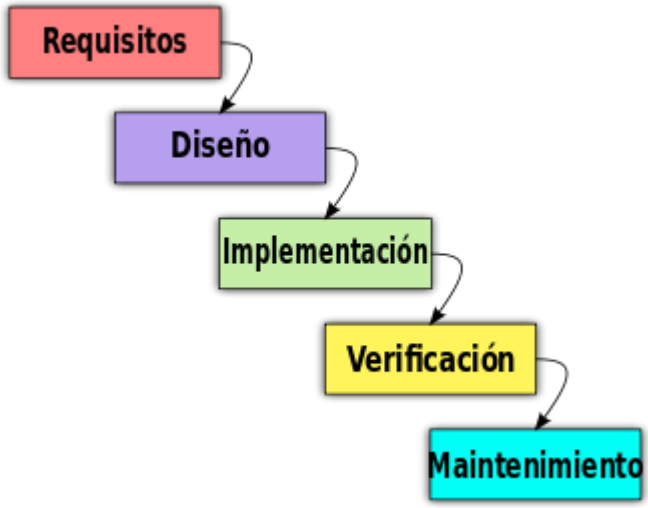
2. **Describa detalladamente el paradigma Cascada**

   Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

   Un ejemplo de una metodología de desarrollo en cascada es:
   1. Análisis de requisitos.
   2. Diseño del Sistema.
   3. Diseño del Programa.
   4. Codificación.
   5. Pruebas.
   6. Implantación.
   7. Mantenimiento.

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. La palabra cascada sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

- **Representación gráfica**



- **Ventajas**

  Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

- **Desventajas**

  En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.

  El proceso de creación del software tarda mucho tiempo ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera. Esto es la base para que funcione bien.

  Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.

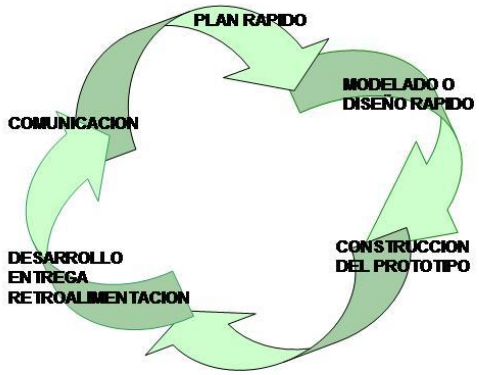
3. Describa detalladamente el paradigma prototipo

El Modelo de prototipos, en Ingeniería de software, pertenece a los modelos de desarrollo evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos.

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo. Sus etapas son:

- Plan rápido
- Modelado, diseño rápido
- Construcción del Prototipo
- Desarrollo, entrega y retroalimentación
- Comunicación

- **Representación gráfica**



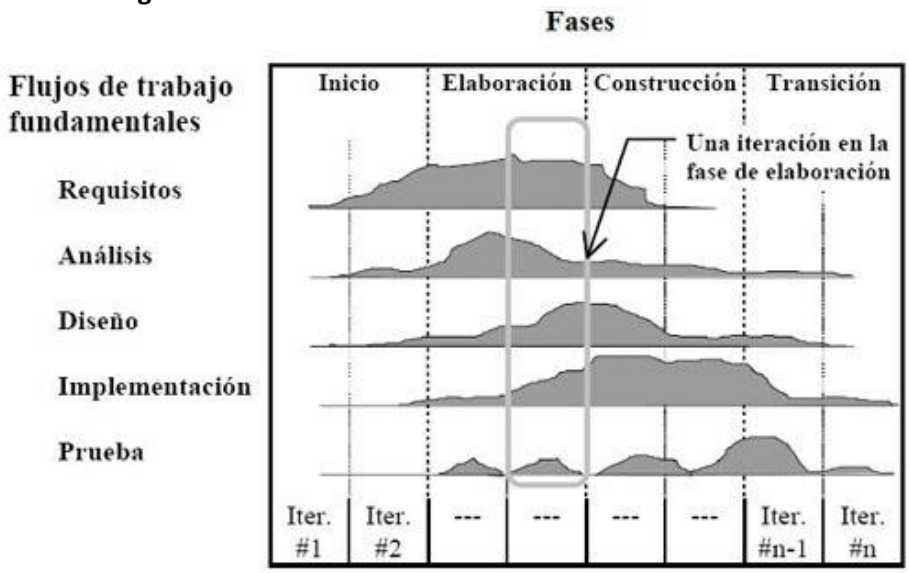
- **Ventajas**
  Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
  También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.
- **Desventajas**
  El usuario tiende a crearse unas expectativas cuando ve el prototipo de cara al sistema final. A causa de la intención de crear un prototipo de forma rápida, se suelen desatender aspectos importantes, tales como la calidad y el mantenimiento a largo plazo, lo que obliga en la mayor parte de los casos a reconstruirlo una vez que el prototipo ha cumplido su función. Es frecuente que el usuario se muestre reacio a ello y pida que sobre ese prototipo se construya el sistema final, lo que lo convertiría en un prototipo evolutivo, pero partiendo de un estado poco recomendado.

4. Describa detalladamente el paradigma PUDS

- **Definición y Características**
  El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio sólo consta de varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.
  Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen

incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

- **Representación gráfica**



- **Ventajas**

  Las iteraciones y construcciones proporcionan reducir la proporción de las tareas, grupos de trabajo y permiten un constante control de riesgos y realimentaciones.

  Permite un lenguaje común como lo es UML, convirtiendo al desarrollo de software una disciplina de ingeniería en vez de meramente escribir código.

  Al emplear las tecnologías de componentes, se puede emplear la reutilización, reduciendo el tiempo y los costes de desarrollo.

- **Desventajas**

  Todo el proceso como tal, se encuentra muy ligado al método, lo que puede ocasionar inconvenientes al tratar de combinar con otras metodologías.

  A pesar de su desarrollo, el método aún no incluye una metodología totalmente explícita para el control de las actividades de gestión.

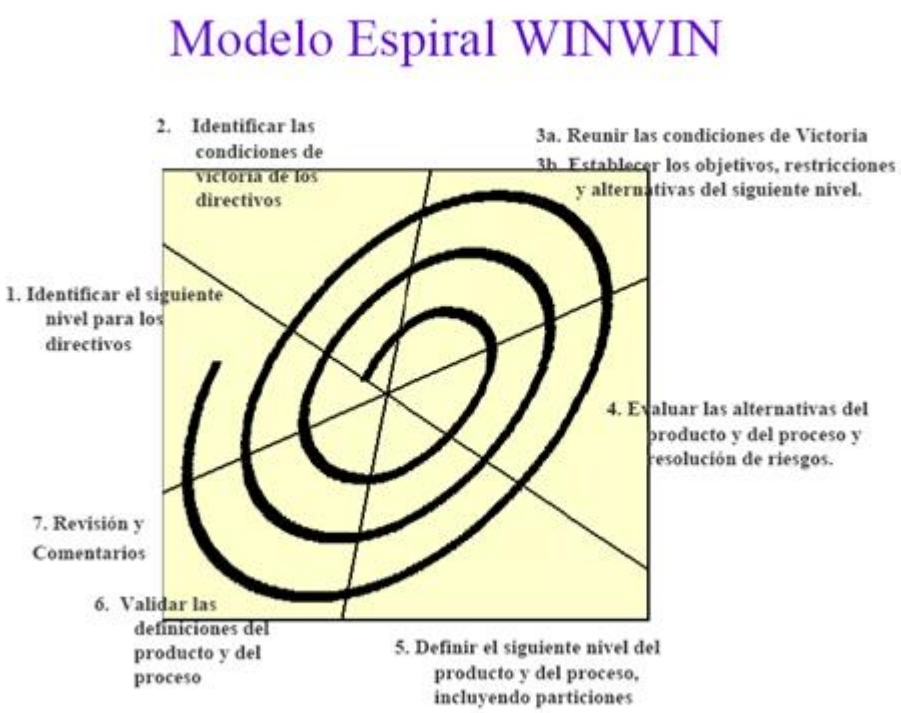
5. Describa detalladamente el paradigma Win Win

- **Definición y Características**

  El modelo en espiral WIN WIN de Boehm, define un conjunto de actividades de negociación al principio de cada paso alrededor de la espiral. Más que una simple actividad de comunicación con el cliente se definen las siguientes actividades:

  - Identificación del sistema o subsistemas clave de los directivos.
  - Determinación de las condiciones de victoria de los directivos.
  - Negociación de las condiciones de victoria de los directivos para reunirlas en un conjunto de condiciones para todos los afectados (incluyendo el equipo del proyecto de software).El modelo en espiral WIN WIN introduce tres hitos en el proceso, llamados puntos de fijación que ayudan a establecer la completitud de un ciclo alrededor del espiral y proporcionan hitos de decisión antes de continuar el proyecto de software. El modelo espiral de Win Win acentúa ciclos de la elaboración concurrente.

- **Representación gráfica**



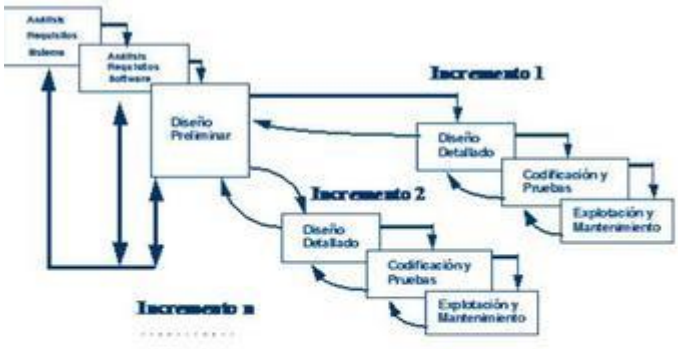
- **Ventajas**
  El impacto que tiene esta técnica entre otras es que con ella se puede realizar un desarrollo de Software más rápidamente de acuerdo a las voluntades de cooperación de los stakeholders (poseedores de apuestas).
  Se aplica en el análisis de requerimientos, calendario, características, etc. Además sirve para separar las personas del problema, enfocarse en los intereses.
- **Desventajas**
  La dificultad de esta técnica ocurre cuando entre los stakeholders ocurre Win-Lose esto conlleva a Lose-Lose (perder y perder).

6. Describa detalladamente el paradigma incremental

- **Definición y Características**
  En este modelo se desarrolla el sistema para satisfacer un subconjunto de requisitos especificados y en posteriores versiones se incrementa el sistema con nuevas funcionalidades que satisfagan más requisitos.
  - Combina elementos del modelo de cascada con la filosofía interactiva de construcción de prototipos
  - Cada secuencia lineal produce un producto operacional con cada incremento de la misma forma que progresa el tiempo en el calendario
  - El primer incremento es a menudo el núcleo
  - Como un resultado de evaluación y/o utilización se desarrolla un plan para el incremento siguiente, este proceso se repite hasta llegar al producto completo
  - Este modelo es particularmente útil cuando la dotación de personal no es suficiente para una implementación completa
  - Los primeros incrementos se pueden implementar con menos recursos

- Si es muy riesgoso desarrollar el sistema completo de una sola vez, entonces debería considerar este modelo.

- **Representación gráfica**



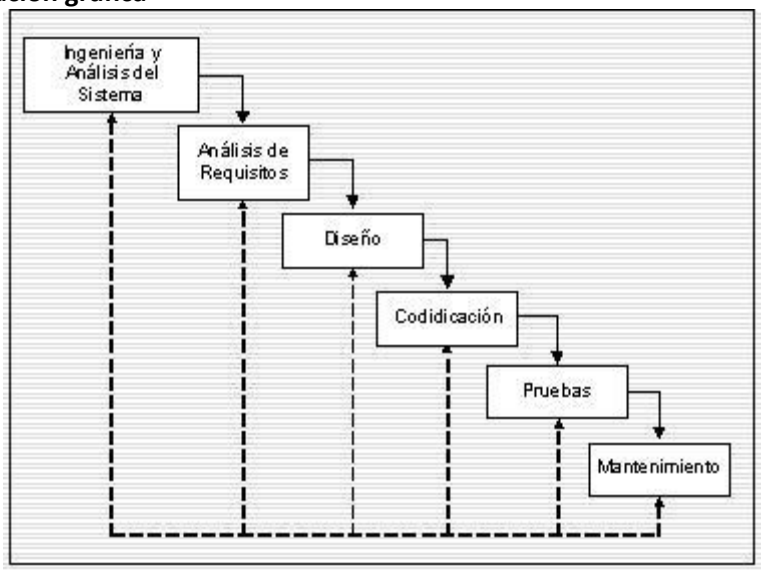
- **Ventajas**
  - Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
  - Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
  - Si un error importante es realizado, sólo la última iteración necesita ser descartada y utilizar el incremento previo.
- **Desventajas**
  - Se presupone que todos los requisitos se han definido al inicio.
  - Se requiere de una experiencia importante para definir los incrementos de forma de distribuir en ellos las tareas en forma proporcional
  - Si el sistema a desarrollar es de gran magnitud y se cuenta con un único grupo para construirlo se corre el riesgo que el desarrollo se prolongue demasiado en tiempo

7. Describa detalladamente el paradigma secuencial

- **Definición y Características**
  - Definición del Problema: Esta fase está dada por el enunciado del problema, el cual requiere una definición clara y precisa. Es importante que se conozca lo que se desea realizar, mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa.
  - Análisis del Problema: En la etapa de análisis es importante establecer los límites de la solución que se va a entregar y acotar al máximo la funcionalidad, tratar de cubrir demasiado es uno de los más grandes errores que se cometen al momento de comenzar a analizar un problema.
  - Diseño de un algoritmo: Al momento de comenzar a diseñar un algoritmo es importante cumplir ciertas características.
    - Finito: El algoritmo debe finalizar su ejecución en un número finito de pasos.
    - Definido: Los pasos que ejecutará el algoritmo deben estar bien definidos y no permitir dobles interpretaciones.

    - Entradas: Un algoritmo debe tener entrada de datos, ya sea desde alguna ubicación de memoria o ingresada por el usuario.
    - Salidas: Un algoritmo debe entregar un resultado, producto del proceso ejecutado después del ingreso de datos.
- Codificación: Es la operación de traspasar la solución del problema que se puede encontrar en un algoritmo gráfico o no gráfico a un lenguaje de programación de alto nivel que sea reconocido por un compilador o interprete y transforme el código fuente en un software o programa.
- Prueba y Depuración: La etapa de pruebas de un algoritmo tienen como objetivo verificar si el algoritmo sobrevive a las situaciones más inusuales, esto se logra verificando si existe un control sobre los datos que ingresan a nuestro algoritmo, por otra parte la etapa de depuración de un algoritmo incluye modificar el algoritmo, para que logre sobrevivir a los eventos más inusuales.
- Documentación: La documentación puede estar escrita en variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas.
- Mantenimiento: La etapa de mantenimiento puede ser ejecutada por errores encontrados en el programa o por mejoras que se deseen realizar, generalmente es llevada a cabo, después de haber finalizado el programa, tiene un gran nivel de dificultad y es importante estudiar y planificar todos los cambios que se desean realizar, ya que pueden afectar a otras áreas del programa.

- **Representación gráfica**



- **Ventajas**
    - Suministra una plantilla en la que pueden colocarse los métodos para cada una de las fases
    - Pasos similares a los pasos genéricos.
    - Modelo procedimental más utilizado.
- **Desventajas**
    - Gran énfasis en la producción de documentos completamente elaborados, producto de las fases de análisis y especificación de requerimientos y de diseño.
    - No muy aplicable a productos de software altamente interactivos.

- Es difícil tener todos los requerimientos, bien definidos al principio, como lo requiere el modelo y además presenta dificultades para acomodar posibles incertidumbres existentes al comienzo de los proyectos.
- Los productos de software raramente siguen el flujo secuencial que propone el modelo. Siempre hay iteraciones y se crean problemas en la aplicación del paradigma.
- Un error importante no detectado al principio puede ser desastroso.
- Se requiere mucha paciencia por parte del cliente, porque solo hasta las etapas finales del desarrollo podrá tener una versión operativa del producto.

8. Describa detalladamente el paradigma frameWork

- **Definición y Características**

  En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

  Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio.

- **Representación gráfica**

- **Ventajas**

  Las principales ventajas de la utilización de un framework son:

  1. El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.

  2. La reutilización de componentes software al por mayor. Los frameworks son los paradigmas de la reutilización.

  3. El uso y la programación de componentes que siguen una política de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

- **Desventajas**

  1. La dependencia del código fuente de una aplicación con respecto al framework. Si se desea cambiar de framework, la mayor parte del código debe reescribirse.

  2. La demanda de grandes cantidades de recursos computacionales debido a que la característica de reutilización de los frameworks tiende a generalizar la funcionalidad

de los componentes. El resultado es que se incluyen características que están "de más", provocando una sobrecarga de recursos que se hace más grande en cuanto más amplio es el campo de reutilización.

9. Describa detalladamente el paradigma de técnicas de cuarta generación

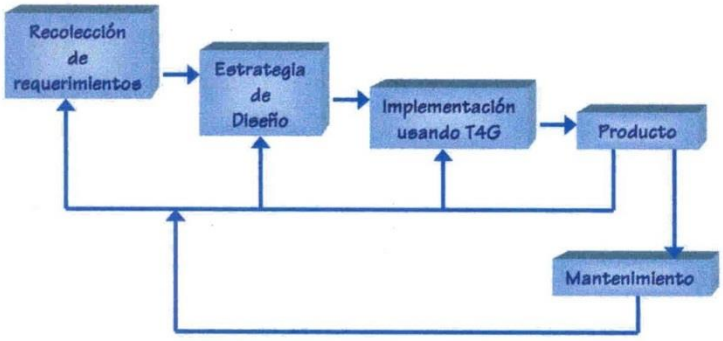
- **Definición y Características**

  Las técnicas de cuarta generación son un conjunto muy diverso de métodos y herramientas que tienen por objeto el de facilitar el desarrollo del software, facilitan al que desarrolla el software la propiedad de especificar algunas características del mismo a alto nivel, más tarde, la herramienta genera automáticamente el código fuente a partir de esta especificación.

  Los tipos más comunes de generadores de código curen uno o varios de los siguientes aspectos:

  - Acceso a base de datos: utilizando lenguajes de consulta de alto nivel.
  - Generadores de códigos: a partir de una especificación de los requisitos se genera automáticamente toda la aplicación.
  - Generación de pantallas: permitiendo diseñar la pantalla dibujándola directamente, incluyendo además el control del cursor y la gestión de los errores de los datos de entrada.
  - Gestión de entornos gráficos.
  - Generación de informes.

- **Representación gráfica**



- **Ventajas**

  La evolución de los lenguajes tiende cada vez más a alejarnos de la maquina o hardware, creando una mayor abstracción de los problemas a resolver, esto es beneficioso pues genera un ahorro significativo de recursos como el tiempo que es tan valioso actualmente.

  Los Lenguajes de Cuarta Generación tienden a ser muy compatibles entre sus mismas evoluciones lo que nos permite crear aplicaciones con la confianza de que el trabajo realizado no será desechado más adelante.

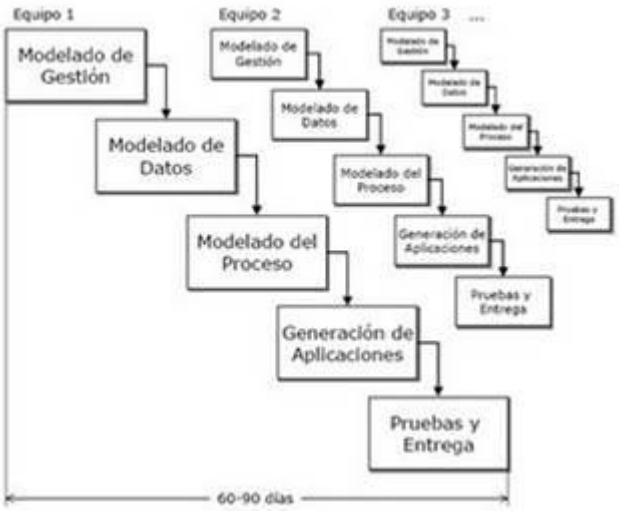
- **Desventajas**

El código fuente que produce es ineficiente, al estar generado automáticamente no pueden hacer uso de los trucos habituales para aumentar el rendimiento, que se basan en el buen conocimiento de cada caso en particular.
Sólo son aplicables al software de gestión, la mayoría de las herramientas de cuarta generación están orientadas a la generación a partir de grandes bases de datos, pero últimamente están surgiendo herramientas que generan esquemas de códigos para aplicaciones de ingeniería y de tiempo real.

10. Describa detalladamente el paradigma DRA

- **Definición y Características**

  El desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo de software, que implica el desarrollo iterativo y la construcción de prototipos. El desarrollo rápido de aplicaciones es un término originalmente utilizado para describir un proceso de desarrollo de software introducido por James Martin en 1991.
  Principios básicos:
  - Objetivo clave es para un rápido desarrollo y entrega de una alta calidad en un sistema de relativamente bajo coste de inversión.
  - Intenta reducir los riesgos inherentes del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.
  - Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos (en cualquier etapa de desarrollo), promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Hace especial hincapié en el cumplimiento de la necesidad comercial, mientras que la ingeniería tecnológica o la excelencia son de menor importancia.
  - Control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite.
  - En general incluye Joint application development (JAD), donde los usuarios están intensamente participando en el diseño del sistema, ya sea a través de la creación de consenso estructurado en talleres, o por vía electrónica.
  - La participación activa de los usuarios es imprescindible.
  - Iterativamente realiza la producción de software, en lugar de enfocarse en un prototipo.
  - Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.
- **Representación gráfica**



- **Ventajas**
  1. Comprar puede ahorrar dinero en comparación con construir.
  2. Los entregables pueden ser fácilmente trasladados a otra plataforma.
  3. El desarrollo se realiza a un nivel de abstracción mayor.
  4. Visibilidad temprana.
  5. Mayor flexibilidad.
  6. Menor codificación manual.
  7. Mayor involucramiento de los usuarios.
  8. Posiblemente menos fallas.
- **Desventajas**
  1. Comprar puede ser más caro que construir.
  2. Costo de herramientas integradas y equipo necesario.
  3. Progreso más difícil de medir.
  4. Menos eficiente.
  5. Menor precisión científica.
  6. Riesgo de revertirse a las prácticas sin control de antaño.
  7. Más fallas (por síndrome de "codificar a lo bestia").
  8. Prototipos pueden no escalar, un problema mayúsculo.

11. ¿Qué diferencia existe entre método, modelo y paradigma?

Paradigma. Es una conceptualización más concreta de una idea o teoría subyacente, contiene definiciones, constructos e interrelaciones entre dichos constructos.
Modelo. Es una representación más detallada de la realidad. Los modelos pertenecen de manera específica a fenómenos particulares y diferentes modelos pueden representar al mismo fenómeno desde distintos puntos de vista paradigmáticos.
Método. Constituye un camino para alcanzar los objetivos estipulados en un plan, es un camino para llegar a un fin.

12. ¿Cuál es el propósito de los ciclos del desarrollo de vida?

El propósito de este programa es definir las distintas fases intermedias que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

Estos programas se originan en el hecho de que es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación. El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados.

PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE

1. **¿Cuáles son las características del PUDS?**
   - *Dirigido por casos de uso:* Los casos de uso no sólo inician el proceso de prueba sino que le proporcionan un hilo conductor. Dirigido por casos d uso quiere decir que el proceso de desarrollo sigue un hilo, avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan, y los casos de usos finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba.
   - *Centrado en la arquitectura:* El concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. Sin embargo, también se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el software, los bloques de construcción reutilizables de que se dispone, consideraciones de implantación, sistemas heredados, y requisitos no funcionales. La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. Debido a lo que es significativo depende en arte d una valoración, que a su vez se adquiere con la experiencia, el valor de una arquitectura depende de las personas que se hayan responsabilizado de su creación.
   - *Iterativo e incremental:* El desarrollo de un producto de software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas, esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son mini-proyectos. Los desarrolladores basan la selección de lo que se implementará en una iteración en dos factores. En primer lugar, la iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora. En segundo lugar, la iteración trata los riesgos más importantes. Las iteraciones sucesivas se construyen sobre los artefactos de desarrollo tal como quedaron al final de la última iteración. Al ser miniproyectos comienzan con los casos de uso y continúan a través del trabajo de desarrollo subsiguiente, que termina convirtiendo en código ejecutable los casos de uso que se desarrollaban en la iteración.

2. **Describa las 4P del PUDS**
   - *Persona:* Los principales autores de un proyecto software son los arquitectos, desarrolladores, ingenieros de prueba, y el personal de gestión que les da soporte, además d los usuarios, clientes y otros interesados. Las personas son realmente seres humanos, a diferencia del término abstracto trabajadores.

- *Proyecto:* Elemento organizativo a través del cual se gestiona el desarrollo del software. El resultado de un proyecto es una versión de un producto.
- *Proceso:* Un proceso de ingeniería de software es una definición del conjunto completo de actividades necesarias para transformar los requisitos de usuario en un producto. Un proceso es una plantilla para crear proyectos.
- *Producto:* Artefactos que se crean durante la vida del proyecto como los modelos, código fuente, ejecutables, y documentación.

3. **Describa los flujos de trabajo del PUDS**
    - Requisitos
    - Análisis
    - Diseño
    - Implementación
    - Prueba

4. **Explicar cada una de las fases del PUDS**
    - Inicio
    - Elaboración
    - Construcción
    - Transición

DIAGRAMAS EN UML

1. **UML es un modelo o una metodología. Justificar**
   Metodología.

2. **¿Cuál es el propósito del vocabulario de UML?**
   El propósito general del vocabulario UML fue crear un lenguaje de modelado estándar donde todo el implicado en el proceso de modelado pueda entender claramente las ideas de sus compañeros de trabajo.

3. **¿Qué son las relaciones?**
   Una relación es una conexión entre elementos. En el modelado orientado a objetos, las tres relaciones más importantes son las dependencias, las generalizaciones y las asociaciones. Gráficamente, una relación se representa como una línea, usándose diferentes tipos de líneas para diferenciar los tipos de relaciones.

4. **¿Qué son los diagramas?**
   Un diagrama es una presentación gráfica de un conjunto de elementos, que la mayoría de las veces se dibuja como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se utilizan para visualizar un sistema desde diferentes perspectivas. Como ningún sistema complejo puede ser comprendido completamente desde una única perspectiva, UML define varios diagramas que permiten centrarse en diferentes aspectos del sistema independientemente.
   Los buenos diagramas hacen comprensible y accesible el sistema. La elección del conjunto adecuado de diagramas para modelar un sistema obliga a plantearse las cuestiones apropiadas sobre el sistema y ayuda a clarificar las implicaciones de las decisiones.

5. **Describir los elementos estructurales y de comportamiento de UML y cuáles son los diagramas que están asociados a ellos**

   Los diagramas estructurales de UML existen para visualizar, especificar, construir y documentar los aspectos estáticos de un sistema y se organizan en líneas generales alrededor de los principales grupos de elementos que aparecen al modelar un sistema.

   Los diagramas de comportamiento de UML se emplean para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema y se organizan en líneas generales alrededor de las formas principales en que se puede modelar la dinámica de un sistema.

6. **Defina los diagramas de clases**

   Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones.

   Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Esto incluye, principalmente modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados: Los diagramas de componentes y los diagramas de despliegue.

   Lo que distingue a un diagrama de clases de los otros tipos de diagramas es que normalmente contienen los siguientes elementos:

   - Clases
   - Interfaces
   - Relaciones de dependencia, generalización y asociación.

   Los diagramas de clases pueden contener notas y restricciones.

   Los diagramas de clases también pueden contener paquetes o subsistemas, los cuales se usan para agrupar los elementos de un modelo en partes más grandes.

   Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Esta vista soporta principalmente los requisitos funcionales de un sistema, los servicios que el sistema debe proporcionar a sus usuarios finales.

   Cuando se modela la vista de diseño estática de un sistema, normalmente se utilizarán los diagramas de clases de una de éstas formas:

   - Para modelar el vocabulario de un sistema
   - Para modelar colaboraciones simples
   - Para modelar un esquema lógico de base de datos

7. **Defina los diagramas de componentes**

   Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase – usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema.

   Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de

software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Debido a que los diagramas de componentes son más parecidos a los diagramas de casos de usos, éstos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema.

En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema.

Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

8. Defina los diagramas de estructura compuesta

Un diagrama de estructura compuesta es un tipo de diagrama de estructura estática en el Lenguaje de Modelado Unificado (UML), que muestra la estructura interna de una clase y las colaboraciones que esta estructura hace posibles. Esto puede incluir partes internas, puertas mediante las cuales, las partes interactúan con cada una de las otras o mediante las cuales, instancias de la clase interactúan con las partes y con el mundo exterior, y conectores entre partes o puertas. Una estructura compuesta es un conjunto de elementos interconectados que colaboran en tiempo de ejecución para lograr algún propósito. Cada elemento tiene algún rol definido en la colaboración.

Las entidades de estructura compuesta claves identificadas en la especificación UML 2.0 son:

- *Parte*

  Una parte es un elemento que representa un conjunto de una o más instancias que pertenecen a una instancia del clasificador contenida. Por ejemplo, si una instancia de diagrama se apropia de un conjunto de elementos gráficos, luego los elementos gráficos se pueden representar como partes, si fuera útil hacer eso para modelar algún tipo de relación entre ellos. Tener en cuenta que una parte se puede quitar de sus padres antes de que el padre se elimine, para que la parte no se elimine al mismo tiempo.
- *Puerto*

  Un Puerto es un elemento escrito que representa una parte visible externa de una instancia del clasificador contenido. Los puertos definen la interacción entre un clasificador y su entorno. Un Puerto puede aparecer en el límite de la parte contenida, una clase o una estructura compuesta. Un Puerto puede especificar los servicios que un clasificador provee así como también los servicios que este requiere de su entorno.
- *Interfaces*

  Una interfaz es similar a una clase pero con un número de restricciones. Todas las operaciones de la interfaz son públicas y abstractas, y no proveen ninguna implementación predeterminada. Todos los atributos de la interfaz deben ser constantes. Sin embargo, mientras que una clase puede solo heredar de una sola super-clase, puede implementar interfaces múltiples.
- Delegar

  Un conector delegar se usa para definir los trabajos internos de los puertos e interfaces externas del componente. Un conector delegar se muestra como una flecha con un estereotipo «delegar». Esto conecta un contrato externo de un componente como se

muestra por sus puertos a la realización interna del comportamiento de la parte del componente.

- Colaboración

  Una colaboración define un conjunto de roles co-operativos usados colectivamente para ilustrar una funcionalidad especifica. Una colaboración debería solo mostrar los roles y los atributos requeridos para lograr sus tareas o funciones definidas. Aislar los roles primarios es un ejercicio de simplificar la estructura y clasificar el comportamiento, y también provee para poder re- usarlo. Un elemento colaboración a menudo implementa un patrón.

9. Defina los diagramas de objetos

Los diagramas de objetos modelan las instancias de los elementos existentes en los diagramas de clases. Un diagrama de objetos muestra un conjunto de d objetos y sus relaciones en un momento concreto.

Los diagramas de objetos se utilizan para modelar la vista de diseño estática o la vista de procesos estática de un sistema, pero desde la perspectiva de instancias reales o prototípicas. Esto conlleva el modelado de una instantánea del sistema en un momento concreto y la representación de un conjunto de objetos, sus estados y sus relaciones.

Lo que distingue al diagrama de objetos es que este contiene:

- Objetos
- Enlaces

10. Defina los diagramas de artefactos

Los diagramas de artefactos son uno de los dos tipos de diagramas que aparecen cuando se modelan los aspectos físicos de los sistemas orientados a objetos. Un diagrama de artefactos muestra la organización y las dependencias entre un conjunto de artefactos.

Los diagramas de artefactos se utilizan para modelar la vista de implementación estática de un sistema. Esto implica modelar las cosas físicas que residen en un nodo, como ejecutables, bibliotecas, tablas, archivos y documentos. Los diagramas de artefactos son fundamentalmente diagramas de clases que se centran en los artefactos de un sistema.

Normalmente, los diagramas de artefactos contienen:

- Artefactos
- Relaciones de dependencia, generalización, asociación y realización.

Los diagramas de artefactos se utilizan para modelar la vista de implementación estática de un sistema. Esta vista se ocupa principalmente de la gestión de configuraciones de las partes de un sistema, formada por artefactos que pueden ensamblarse de varias formas para producir un sistema ejecutable.

Cuando se modela la vista de implementación estática de un sistema, normalmente se utilizarán los diagramas de artefactos de una de las maneras siguientes:

- Para modelar código fuente
- Para modelar versiones ejecutables
- Para modelar bases de datos físicas
- Para modelar sistemas adaptables

11. Defina los diagramas de despliegue

Los diagramas de despliegue son uno de los dos tipos de diagramas que aparecen cuando so modelan los aspectos físicos de los sistemas orientados a objetos. Un diagrama de

despliegue muestra la configuración de los nodos que participan en la ejecución y de los artefactos que residen en ellos.

Los diagramas de despliegue se utilizan para modelar la vista de despliegue estática de un sistema. La mayoría de las veces, esto implica modelar la topología del hardware sobre el que se ejecuta el sistema. Los diagramas de despliegue son fundamentalmente diagramas de clases que se ocupan de modelar los nodos de un sistema.

Los diagramas de despliegue normalmente contienen:

- Nodos
- Relaciones de dependencia y asociación

Los diagramas de despliegue también pueden contener artefactos, cada uno de los cuales debe de residir en algún nodo. Los diagramas de despliegue también pueden contener paquetes o subsistemas, los cuales se utilizan para agrupar elementos del modelo en bloques más grandes.

Los diagramas de despliegue se utilizan para modelar la vista de despliegue estática de un sistema. Esta vista abarca principalmente la distribución, la entrega y la instalación de las partes que configuran el sistema físico.

Cuando se modela la vista de despliegue estática de un sistema, normalmente se utilizan los diagramas de despliegue de una de las siguientes maneras:

- Para modelar sistemas embebidos
- Para modelar sistemas cliente/servidor
- Para modelar sistemas completamente distribuidos

12. Detallar el diagrama de casos de uso

Los diagramas de casos de uso son uno de los tipos de diagramas UML que se utilizan para modelar los aspectos dinámicos de un sistema. Los diagramas de casos de uso son importantes para modelar el comportamiento de un sistema, un subsistema o una clase. Cada uno muestra un conjunto de casos de uso, actores y sus relaciones.

Los diagramas de casos de uso se emplean para modelar la vista de casos de uso de un sistema. La mayoría de las veces, esto implica modelar el contexto de sistema, subsistema o clase, o el modelado de los requisitos de comportamiento de esos elementos.

Los diagramas de casos de uso son importantes para visualizar, especificar y documentar el comportamiento de un elemento. Estos diagramas facilitan que los sistemas, subsistemas y clases sean abordables y comprensibles, al presentar una vista externa de cómo pueden utilizarse estos elementos en un contexto dado. Los diagramas de casos de uso también son importantes para probar sistemas ejecutables a través de ingeniería directa y para comprender sistemas ejecutables a través de ingeniería inversa.

Normalmente, un diagrama de casos de uso se distingue del resto de los diagramas porque contiene:

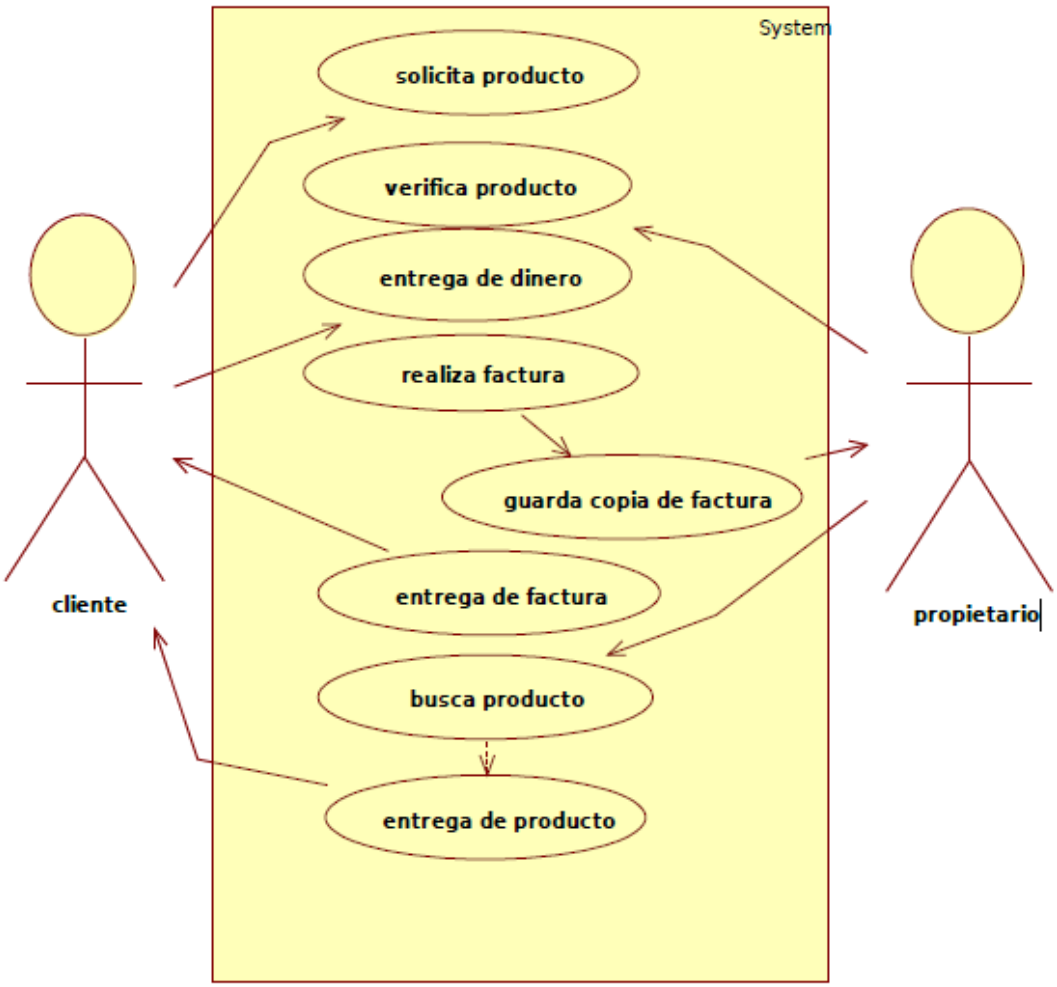
- Sujetos
- Casos de uso
- Actores
- Relaciones de dependencia, generalización y asociación

En los diagramas de casos de uso, el sujeto se representa como un rectángulo que contiene un conjunto de elipses que son los casos de uso. El nombre del sujeto se coloca dentro del rectángulo. Los actores se muestran como monigotes fuera del rectángulo, con el nombre debajo. Las líneas conectan los iconos de los actores con las elipses de los casos de uso con los que se comunican. Las relaciones entre los casos de uso (como la extensión y la inclusión) se dibujan dentro del rectángulo.

Los diagramas de casos de uso se emplean para modelar la vista de casos de uso de un sujeto, como un sistema. Esta vista abarca principalmente el comportamiento externo del sujeto (los servicios visibles externamente que el sujeto proporciona en el contexto de su entorno).
Cuando se modela la vista de casos de uso estática de un sujeto, normalmente se emplean los diagramas de casos de uso de una de las dos formas siguientes:

- Para modelar el contexto de un sujeto
- Para modelar los requisitos de un sujeto

Ejemplo. Diagrama de casos de uso para modelar el comportamiento de las ventas en una tienda.



13. Defina los diagramas de interacción

14. Defina los diagramas de secuencia

Los diagramas de secuencia se utilizan para modelar los aspectos dinámicos de los sistemas. Un diagrama de secuencia muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Un

diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes.
Un diagrama de secuencia destaca la ordenación temporal de los mensajes.
Los diagramas de secuencia contienen:

- Roles u objetos
- Comunicaciones o enlaces
- Mensajes

Los diagramas de secuencia tienen dos características que los distinguen de los diagramas de comunicación:

- La línea de vida
- El foco de control

Una secuencia de mensajes está bien para mostrar una secuencia sencilla y lineal, pero a menudo necesitamos mostrar condicionales y bucles. A veces queremos mostrar la ejecución concurrente de varias secuencias. Este tipo de control puede mostrarse mediante operadores de control estructurados en los diagramas de secuencia.
Los siguientes tipos de controles son los más habituales:

- Ejecución opcional
- Ejecución condicional
- Ejecución paralela
- Ejecución en bucle (iterativa)

15. Defina los diagramas de comunicación

Los diagramas de secuencia se utilizan para modelar los aspectos dinámicos de los sistemas. Un diagrama de secuencia muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Un diagrama de comunicación es un diagrama de interacción que destaca la organización estructural de los objetos que envían y reciben mensajes.
Un diagrama de comunicación destaca la organización de los objetos que participan en una interacción.

Los diagramas de comunicación contienen:

- Roles u objetos
- Comunicaciones o enlaces
- Mensajes

Los diagramas de comunicación tienen dos características que los distinguen de los diagramas de secuencia:

- El camino
- El número de secuencia

Cuando se modelan los aspectos dinámicos de un sistema, normalmente se utilizan los diagramas de interacción de dos formas:

- Para modelar flujos de control por ordenación temporal
- Para modela flujos de control por organización

16. Defina los diagramas de estados

Los diagramas de estados son uno de los cinco tipos de diagramas de UML que se utilizan para modelar los aspectos dinámicos de un sistema. Un diagrama de estados muestra una máquina de estados. Tanto los diagramas de actividades como los diagramas de estados son útiles para modelar la vida de un objeto. Sin embargo, mientras que un diagrama de

actividades muestra el flujo de control entre actividades a través de varios objetos, un diagrama de objetos muestra el flujo de control entre estados dentro de un único objeto.
Los diagramas de estados se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto supone el modelado del comportamiento de objetos reactivos. Un objeto reactivo es aquel para el que la mejor forma de caracterizar su comportamiento es señalar cuál es su respuesta a los eventos lanzados desde fuera de su contexto. Un objeto reactivo tiene un ciclo de vida bien definido, cuyo comportamiento se ve afectado por su pasado. Los diagramas de estados pueden asociarse a las clases, los casos de uso, o a sistemas completos para visualizar, especificar, construir y documentar la dinámica de un objeto individual.
Normalmente, los diagramas de estados contienen:

- Estados simples y compuestos
- Transiciones, incluyendo eventos y acciones

Los diagramas de estados se utilizan para modelar los aspectos dinámicos de un sistema. Estos aspectos dinámicos pueden involucrar el comportamiento dirigido por eventos de cualquier tipo de objetos en cualquier vista de la arquitectura de un sistema, incluyendo las clases, interfaces, componentes y nodos.

17. Detallar el diagrama de actividades

Los diagramas de actividades son uno de los cinco tipos de diagramas UML que se utilizan para el modelado de los aspectos dinámicos de los sistemas. Un diagrama de actividades es fundamentalmente un diagrama de flujo que muestra el flujo de control entre actividades. Al contrario que un diagrama de flujo clásico, un diagrama de actividad muestra tanto la concurrencia como las bifurcaciones del control.
Los diagramas de actividades se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar los pasos secuenciales (y posiblemente concurrentes) de un proceso computacional. Con in diagrama de actividades también se puede modelar el flujo de valores entre los distintos pasos. Los diagramas de actividades pueden servir para visualizar, especificar, construir y documentar la dinámica de una sociedad de objetos, o pueden emplearse para modelar el flujo de control de una operación. Mientras que los diagramas de interacción destacan el flujo de control entre los distintos pasos. Una actividad es una ejecución estructurada en curso de un comportamiento. La ejecución de una actividad se descompone finalmente en la ejecución de varias acciones individuales, cada una de las cuales puede cambiar el estado del sistema o puede comunicar mensajes.

Los diagramas de actividades se distinguen de los otros tipos de diagramas al contener:

- Acciones
  En el flujo de control modelado por un diagrama de actividades suceden cosas. Por ejemplo, se podría evaluar una expresión que estableciera el valor de un atributo o que devolviera algún valor. También se podría invocar una operación sobre un objeto, enviar una señal a un objeto o incluso crear o destruir un objeto. Estas computaciones ejecutables y atómicas se llaman acciones.
- Nodos de actividad
  Un nodo de actividad es una unidad organizativa dentro de una actividad. Rn general, los nodos de actividad son agrupaciones anidadas de acciones o de otros nodos de actividad. Además, los nodos de actividad tienen una subestructura visible; en general, se considera que se necesita un cierto tiempo para que se completen.

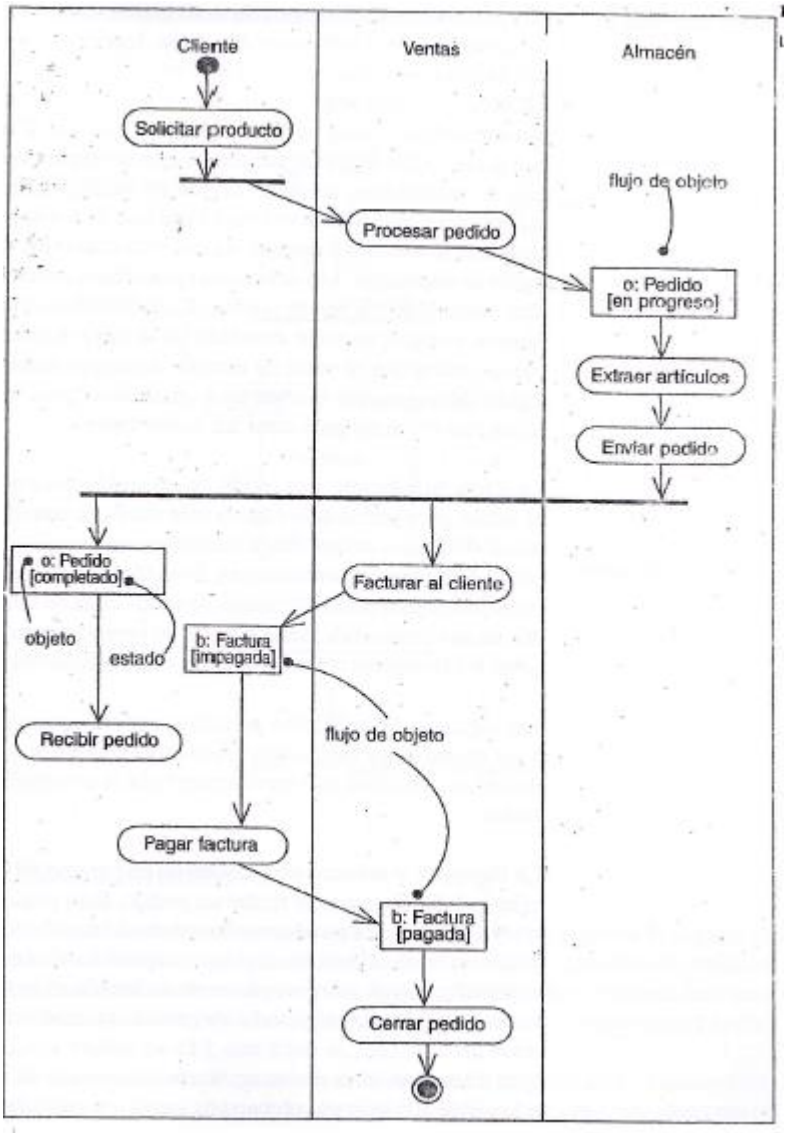
- Flujos
- Objetos valor

Los diagramas de actividades se utilizan para modelar los aspectos dinámicos de un sistema. Estos aspectos dinámicos pueden involucrar la actividad de cualquier tipo de abstracción en cualquier vista de la arquitectura de un sistema, incluyendo clases, interfaces, componentes y nodos.

Cuando se modelan los aspectos dinámicos de un sistema, normalmente se utilizan los diagramas de actividades de dos formas:

- Para modelar un flujo de trabajo
- Para modelar una operación

Ejemplo. Diagrama de actividades para modelar el comportamiento de las ventas en una tienda.



18. Defina los diagramas de paquetes

Los paquetes se utilizan para organizar los elementos de modelado en artes mayores que se pueden manipular como un grupo. La visibilidad de estos elementos puede controlarse para que algunos sean visibles fuera del paquete mientras que otros permanecen ocultos. Los paquetes también se pueden emplear para presentar diferentes vistas de la arquitectura del sistema.

Los paquetes bien diseñados agrupan elementos cercanos semánticamente y que suelen cambiar juntos. Por tanto los paquetes bien estructurados son cohesivos y poco acoplados, y el acceso a su contenido está muy controlado.

19. Ishikawa

El método de Ishikawa consiste en realizar e identificar el problema principal con sus categorías y hacer una lluvia de ideas para buscar causa-efecto de cada problema.

a. *Identificar el problema*
   - Especificar describiendo los procesos que involucran el sistema para idntificar el área del proyecto
   - Representar mediante un gráfico
     a. *Identificar lista de problemas*
        Hacer una lista de problemas de todos los procesos del sistema, cada proceso comienza especificando la falencia, insuficiencia, deficiencia, o algún otro criterio que indique el problema.

     b. *Depurar problema*
        De la lista de problemas ya identificados, seleccionar los problemas que forman parte del caso de estudio del proyecto, los demás problemas son depurados o eliminados.
        Volver a numerar los problemas para su codificación.

     c. *Identificar propietario de problemas*
        Hacer una lista de propietarios de problemas, las personas que son afectadas cada vez que ocurre un problema
        - Elaborar una matriz de problemas y propietarios de problemas
          Realizar consultas por cada problema con los propietarios, si son afectados marcar identificando la intersección.

     d. *Análisis de los problemas*
        Elaborar un grafo de problemas y las aristas son dirigidas hacia otros problemas que visualizan causa-efecto.

b. *Identificar principales categorías*
   Especificar y describir por categoría la causa-efecto que se realizan o involucran cada proceso.

c. *Identificar las causas*

d. *Analizar y discutir el diagrama*
   a. *Conclusión*
   b. *Alternativas de cambio*
   c. *Del conjunto de alternativas. ¿Cuál o cuáles son las más necesarias o recomendables para que los procesos funcionen adecuadamente?*