Guía de preguntas para final

1. Describa el modelo de procesos incremental y diferéncielo del modelo iterativo.

   En el desarrollo en fases de desarrolla el sistema de tal manera que puede ser entregado en piezas. Esto implica que existen dos sistemas funcionando en paralelo: el sistema operacional y el sistema en desarrollo. Se desarrollan versiones.

   - Incremental: el sistema es particionado en subsistemas de acuerdo con su funcionalidad. Cada entrega agrega un subsistema.
     En un proceso de desarrollo incremental, los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuales menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema.
     Una vez que los incrementos del sistema se han identificado, los requerimientos para los servicios que se van a entregar en el primer incremento se define en detalles, y éste se desarrolla.
     Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio. Esto significa que tienen una entrega temprana de parte de la funcionalidad del sistema. Pueden experimentar con el sistema, lo cual les ayuda a clarificar sus requerimientos para los incrementos posteriores y para las últimas versiones del incremento actual.
   - Iterativo: el ciclo de vida iterativo se basa en la amplificación y refinamiento sucesivos de un sistema software mediante múltiples iteraciones, con retroalimentación cíclica. El sistema se incrementa a lo largo del tiempo, iteración tras iteración. La salida de una iteración no es un prototipo experimental o desechable, más bien es un subconjunto con calidad de producción del sistema final. Cada iteración conlleva la elección de un pequeño conjunto de requisitos que permite rápidamente diseñar, implementar y probar.

     Beneficios

     - Reducción temprana de riesgos.
     - Progreso visible desde las primeras etapas.
     - Una temprana retroalimentación, refinando el sistema a las necesidades reales de los usuarios.
     - Gestión de la complejidad: el equipo no se ve abrumado por la «parálisis del análisis».
     - El conocimiento adquirido en un ciclo se puede utilizar metódicamente para mejorar el propio proceso de desarrollo.

2. Elija 2 modelos de proceso, explíquelos y diferéncielos.

3. ¿Por qué cambian los requerimientos?

   Los requerimientos cambian y esto persiste a lo largo de la vida del sistema. Los cambios ocurren por:

   - Porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas (En sistemas grandes hay una comunidad diversa de usuarios).
   - Porque los clientes y los usuarios son distintos.
   - Porque cambió el problema que se estaba resolviendo.
   - Porque los usuarios cambiaron su forma de pensar o sus percepciones.
   - Porque cambió el ambiente de negocios (mercado, etc.).

4. Describa los requerimientos funcionales y no funcionales.

   Requerimientos funcionales

- Describen una interacción entre el sistema y su ambiente. Cómo debe comportarse el sistema ante determinado estímulo.
- Describen lo que el sistema debe hacer, o incluso cómo NO debe comportarse.
- Describen con detalle la funcionalidad del mismo.
- Son independientes de la implementación de la solución.
- Se pueden expresar de distintas formas.

Requerimientos no funcionales

- Describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema.

Dentro de los requerimientos *no funcionales* podemos encontrar:

- Requerimientos del producto: especifican el comportamiento del producto (usabilidad, eficiencia, rendimiento, espacio, fiabilidad, portabilidad).
- Requerimientos organizacionales: se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (entrega, implementación, estándares).
- Requerimientos externos: interoperabilidad, legales, privacidad, seguridad, éticos.

5. ¿Qué conocimientos debe tener un ingeniero de software?

   El ingeniero debe conocer las tecnologías y productos de sistemas operativos, lenguajes de programación, bases de datos, bibliotecas, etc.

   Además también conocer técnicas de administración de proyectos como planificación, análisis de riesgos, control de calidad, seguimiento de proyectos, etc.

   Ya que la ingeniería de software se desarrolla en un marco económico, social y legal, deberá haber cierta ***responsabilidad profesional y ética*** y no usar su capacidad y habilidades deshonestamente o deshonrando la profesión.

6. Definir qué es una especificación de requisitos de software (SRS). Indicar algunos 5 criterios a cumplir para una buena SRS.
7. Describa el modelo espiral.

   El modelo espiral o Boehm combina las actividades de desarrollo con la gestión del riesgo.

   - Trata de mejorar los ciclos de vida clásicos y prototipos.
   - Incorpora objetivos de calidad y gestión de riesgos
   - Elimina errores y alternativas no atractivas al comienzo
   - Permite iteraciones, vuelta atrás y finalizaciones rápidas
   - Cada ciclo empieza identificando:
   - Los objetivos de la porción correspondiente
   - Las alternativas
   - Restricciones
   - Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente

8. Describa los modelos de calidad presentados en la materia.
9. ¿Qué significa que una entrevista sea abierta?

   Que una entrevista sea abierta o no estructurada significa que el encuestador lleva a un tema en general, no hay preparación de preguntas específicas y inicia con preguntas que no dependen del contexto.

10. Describir los roles de SCRUM.
    - **Product owner**: conoce y marca las prioridades del proyecto o producto.
    - **Scrum master:** es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema que pueda aparecer. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas.
    - **Scrum team:** son las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el P.O.
    - **Usuarios/ clientes:** son los beneficiarios finales del producto, y son quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades.
    - **Product backlog:** lista maestra que contiene toda la funcionalidad deseada en el producto. La característica más importante es que la funcionalidad tiene un orden de prioridad.
    - **Sprint backlog:** lista que contiene toda la funcionalidad que el equipo se comprometió a desarrollar durante un sprint determinado.
    - **Burndown chart:** muestra un acumulativo de trabajo hecho.
11. ¿Por qué eligirías C.U como técnica de especificación de requerimientos?

    Elegiría C.U por los siguientes beneficios:
    - Al ser muy corta, representa requisitos del modelo de negocio que pueden implementarse rápidamente.
    - Necesitan poco mantenimiento.
    - Mantienen una relación cercada con el cliente.
    - Permite dividir los proyectos en pequeñas entregas.
    - Permite estimar fácilmente el esfuerzo de desarrollo.
    - Ideal para proyectos con requisitos volátiles o poco claros.
12. El sistema tiene un módulo de planificación de actividades que dada su dificultad se decidió utilizar un prototipo desechable. ¿Qué otro tipo de prototipo existe?

    El otro tipo principal de prototipo, en contraste con el prototipo desechable, es el prototipo evolutivo).

    Es aquel que no se descarta; en su lugar, se desarrolla progresivamente y se refina a través de múltiples ciclos de retroalimentación con el cliente hasta que se convierte en el sistema final operativo.
13. Describa las ventajas de C.U.
    - Herramienta para capturar requerimientos funcionales.
    - Descompone el alcance del sistema en piezas más manejables.
    - Medio de comunicación con los usuarios.
    - Utiliza lenguaje común y fácil de entender por las partes.
    - Permite estimar el alcance del proyecto y el esfuerzo a realizar.
    - Define una línea base para la definición de los planes de prueba.
    - Define una línea base para toda la documentación del sistema.
    - Proporciona una herramienta para el seguimiento de requisitos.
14. Describa la metodología Scrum. ¿Cuáles son sus principios?

    Scrum es un proceso al que se aplican, de manera regular, un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto.

Estas prácticas se apoyan unas a otras y su elección tiene origen en un estudio de la manera de trabajar en equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto.

Tiene sus principios como

- **Eliminar el desperdicio:** no generar artefactos ni perder el tiempo haciendo cosas que no le suman valor al cliente.
- **Construir la calidad con el producto:** la idea es inyectar la calidad directamente en el código desde el inicio.
- **Crear conocimiento:** en la práctica no se puede tener el conocimiento antes de empezar el desarrollo.
- **Diferir las decisiones:** tomar las decisiones en el momento adecuado ya que a medida que va pasando el tiempo se obtiene más información.
- **Entregar rápido:** debe ser una de las ventajas competitivas.
- **Respetar a las personas:** la gente trabaja mejor cuando se encuentra en un ambiente que la motive y se sienta respetada.
- **Optimizar el todo:** optimizar todo el proceso, ya que el proceso es una unidad, y para lograr tener éxito y avanzar, hay que tratarlo como tal.

15. ¿Cuáles son las características del software?

    El ***software*** son instrucciones, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

    Se caracteriza por ser un elemento lógica el cual se desarrolla y no se desgasta.

    - Elemento lógico: es intangible y existe como información, a diferencia del *hardware* que es un elemento físico.
    - Se desarrolla: se crea mediante procesos de ingeniería, no se "fabrica" en el sentido tradicional.
    - No se desgasta: no se estropea con el uso.

16. Enuncie las técnicas de validación de requerimientos que conoce.

    Las validaciones de requerimientos pueden ser

    - Formal: El equipo de desarrollo guía al cliente, explicándole las implicaciones de cada requerimiento. Es conveniente realizarlas después de la revisión informal.
    - Informal: los desarrolladores deben tratar los requerimientos con tantos stakeholders como sea posible.
    - Construcción de prototipos: permite a clientes y desarrolladores examinar aspectos del sistema propuesto para decidir si es adecuado para el producto terminado.
    - Generación de casos de pruebas: la creación de casos de prueba ayuda a confirmar la verificabilidad de los requerimientos.

17. ¿Cuáles son los objetivos de las metodologías ágiles?

    Los objetivos de las metodologías ágiles son

    - Producir software de alta calidad con un costo efectivo y en el tiempo apropiado.
    - Esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.
    - Ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

18. ¿Cuáles son los componentes de calidad según la visión holísticas?

Se centran en los **atributos** que un producto debe poseer para satisfacer las necesidades explícitas e implícitas del usuario.

Estos atributos de calidad, que también se consideran **requerimientos no funcionales**, son los siguientes:

- Funcionalidad: el grado en que el software proporciona funciones que satisfacen necesidades.
- Confiabilidad: la capacidad del software de mantener su nivel de rendimiento bajo condiciones especificadas.
- Eficiencia: el rendimiento relativo a la cantidad de recursos utilizados (tiempo de respuesta, uso de memoria, etc).
- Facilidad de uso: la facilidad con la que los usuarios pueden aprender a usar el software y operarlo.
- Mantenibilidad: la facilidad con la que el software puede ser modificado (corregir defectos, adaptarse a nuevos entornos, mejorar el rendimiento).
- Portabilidad: la capacidad del software de ser transferido de un entrono de hardware o software a otro.
- Seguridad: la capacidad de proteger la información y los datos para que personas o sistemas no autorizados no puedan leerlos o modificarlos.
- Compatibilidad: la capacidad de dos o más sistemas o componentes para intercambiar información y utilizar las mismas funciones.

19. ¿Cuántos puntos de vista genéricos conoce?

    - **Interactuadores**: personas o sistemas que interactúan directamente son el sistema y que pueden influir en los requerimientos del sistema de algún modo.
    - **Indirecto**: stakeholders que no usan el sistema pero influyen en los requerimientos.
    - **Dominio**: características y restricciones del dominio que influyen en los requerimientos.

20. ¿Qué es un requerimiento? Describa requerimiento funcional y no funcional. El requerimiento “El sistema debe tener un tiempo de respuesta de menor a 20 seg.” ¿De qué tipo es?

    Un requerimiento es una característica del sistema o una descripción de algo que es capaz de hacer para satisfacer el propósito del sistema.

    Los *requerimientos funcionales* describen la interacción entre el sistema y su ambiente. Estos requerimientos definen lo que el sistema debe hacer o no.

    Qué describen: El comportamiento del sistema ante un determinado estímulo. En esencia, describen la funcionalidad del *software*. **Ejemplo:** "El sistema debe permitir al usuario registrar una nueva cuenta."

    Los *requerimientos no funcionales d*escriben una restricción sobre el sistema que limita las elecciones en la construcción de una solución al problema. Se centran en cómo debe comportarse el sistema. **Qué describen:** Restricciones de diseño, aspectos de rendimiento, calidad y atributos de la arquitectura del *software*.
    El requerimiento propuesto resulta ser uno no funcional ya que es un atributo de rendimiento.

21. Describa diferencias entre el modelo en cascada y el modelo en V.
    Ambos se basan en un flujo de fases, pero se diferencian fundamentalmente en el manejo de la verificación y validación.
    La diferencia clave es que el modelo en cascada es estrictamente lineal, mientras que el modelo en V integra las pruebas de forma paralela a las fases de desarrollo.
    En el modelo en cascada las etapas se representan cayendo en cascada, y cada una debe completarse antes de que comience la siguiente.
    Las pruebas y la verificación se concentran en las etapas finales del ciclo, lo cual es su principal debilidad.
    La eliminación de fallas puede ser extremadamente difícil y costosa durante las últimas etapas de prueba del sistema.
    El Modelo en V (Verification and Validation Model) es una extensión del modelo en cascada que mejora la gestión de riesgos al hacer explícitos los vínculos entre las fases de desarrollo y las fases de prueba.

La planificación de pruebas comienza al inicio. Cada fase de desarrollo tiene una fase de prueba directamente asociada donde hay un énfasis en la verificación de los entregables de cada fase y los errores se detectan antes.

Más estructurado que el cascada, pero sigue siendo rígido para sistemas con requisitos cambiantes.

22. ¿Qué es el estudio de factibilidad? Describa.

    Factibilidad es lo mismo que viabilidad.

23. Describa la técnica de especificación dinámica de casos de uso.

    Los casos de uso son procesos de modelado de las funcionalidades del sistema en término de los eventos que interactúan entre los usuarios y el sistema.

    Tienen sus orígenes en el modelado orientado a objetos pero su eficiencia en modelado de requerimientos hizo que se independice de la técnica de diseño utilizada, siendo aplicable a cualquier metodología de desarrollo.

    El uso de C.U facilita y alienta la participación de los usuarios.

    El **proceso de modelado** es el siguiente

    1. Identificar a los actores: se buscarán los potenciales en documentación, minutas de reuniones o documentos de requerimientos. El objetivo es saber quién o qué proporciona las entradas al sistema, sus salidas, interfaces requeridas, la información.

       Se nombran mediante sustantivos o frase sustantiva.

    2. Identificar los CU para los requerimientos: las principales preguntas a responder es cuáles son las tareas del actor, información necesaria, qué información proporciona el actor del sistema, necesita el sistema informar eventos o cambios y viceversa.
    3. Construir el diagrama.
    4. Realizar los escenarios.

    Cuenta con **componentes** los cuales son los siguientes

    - Diagrama de casos de uso: ilustra las interacciones entre el sistema y los actores. Representa un objetivo individual del sistema y describe la secuencia de actividades y de interacciones para alcanzarlo. Para que el CU sea considerado un requerimiento debe estar acompañado de su respectivo escenario.
    - Escenarios: descripción de la interacción entre el actor y el sistema para realizar la funcionalidad.
    - Actores: inicia una actividad en el sistema. Representa a un usuario que interactúa. Puede ser una persona, sistema externo o dispositivo externo que dispare un evento.
    - Relaciones
        - Asociaciones: relación entre un actor y un CU en el que interactúan entre sí.
        - Extensiones: un CU extiende la funcionalidad de otro CU. Un CU puede tener muchos CU extensiones pero los CU extensiones sólo son iniciados por un CU.
        - Uso o inclusión: reduce la redundancia entre dos o más CU al combinar los pasos comunes de los CU.
        - Dependencia: relación entre CU que indica que un CU no puede realizarse hasta que se haya realizado otro CU.
        - Herencia: relación entre actores donde un actor hereda las funcionalidad de uno o varios actores.

24. Explique los objetivos de la especificación de requerimientos.

    Los *objetivos* son

    - Permitir que los desarrolladores expliquen cómo han entendido lo que el cliente pretende del sistema
    - Indicar a los diseñadores qué funcionalidad y características va a tener el sistema resultante.

- Indicar al equipo de pruebas qué demostraciones llevar a cabo para convencer al cliente de que el sistema que se le entrega es lo que había pedido.

25. Realice una comparación entre modelo de ciclo de vida clásico (cascada) y desarrollo basado en componentes.

    El ciclo de vida clásico o cascada es un modelo tradicional y el desarrollo basado en componentes es uno de los paradigmas de proceso de software, diferenciándose principalmente en su enfoque de inicio y su flujo de trabajo.

    - Modelo en cascada: fue introducido en la era de control (70s), donde las etapas caen en cascadas por lo tanto cada etapa de desarrollo debe completarse antes de que comience la siguiente.

      Su simplicidad hace que sea fácil de explicar pero la desventaja es que no se tiene un resultado concreto hasta que el proceso esté terminado además de que las fallas más triviales se encuentran al comienzo del período de prueba y las más **graves al final**.
      Finalmente, resulta poco realista congelar fases por errores.

    - Desarrollo basado en componentes: es uno de los modelos o paradigmas de proceso de software. Esta técnica supone que las partes ya existen y el proceso de desarrollo se enfoca en integrar estas partes preexistentes.

26. DTE y Redes de Petri son notaciones para describir distintos aspectos de un sistema de software ¿Cuáles son estos aspectos? Razone la contestación proponiendo un ejemplo sencillo de aplicación de cada notación.

    Los DTE describen el sistema en su aspecto dinámico , es decir, en función de los cambios que ocurren a lo largo del tiempo.

    El aspecto clave que describe es:

    - Comportamiento por estados: El sistema se considera en un estado particular hasta que un estímulo lo obliga a cambiar a un nuevo estado.
    - Secuencialidad: Se centran en el flujo de control de un objeto individual o de un sistema sencillo que no maneja concurrencia compleja.

    Un ejemplo básico con un DTE es un semáforo donde

    - Estados: rojo, verde y amarillo.
    - Transiciones: tiempo o un evento que fuerza el cambio de un estado a otro.

    Aplicación:

    1. Estado inicial: el semáforo está en rojo.
    2. Transición 1: pasa un tiempo determinado y el semáforo cambia de rojo a verde.
    3. Transición 2: pasa un tiempo determinado y el semáforo cambia de verde a amarillo.
    4. Transición 3: pasa un tiempo determinado y el semáforo cambia de amarillo a rojo.

    Las Redes de Petri fueron inventadas con el objeto de especificar sistemas de tiempo real y son necesarias para representar aspectos de *concurrencia.*

    El aspecto clave que describe es:

    - **Concurrencia:** la **ejecución simultánea** de componentes de programación, llamadas o procesos. Las tareas se realizan en paralelo, aunque en un orden impredecible.
    - **Sincronización:** permiten que varios procesos colaboren y **compartan información y recursos** de manera controlada para asegurar la integridad.

    Un ejemplo simple para una red de Petri es una impresora donde

    - Sitios: tienen tokens los cuales indican si la impresora está libre o no.
    - Transiciones: representan eventos o acciones como “Se inicia la impresión”.

- Tokens: manejan la **coordinación de eventos y estados** y controlan la ejecución.

En conclusión, tanto los **Diagramas de Transición de Estados (DTE)** como las **Redes de Petri** son notaciones utilizadas para describir el **aspecto dinámico** de un sistema de software, modelando cómo evoluciona a través de diferentes **estados** o condiciones.

La **diferencia clave** radica en el alcance de su modelado:

- **DTE:** Está diseñado para especificar el **comportamiento secuencial** de una entidad simple. Modela el ciclo de vida de un objeto o un proceso individual, mostrando cómo un **estímulo** obliga a un cambio de estado.
- **Redes de Petri:** Fueron creadas para especificar **sistemas de tiempo real**. Su principal valor es su capacidad para representar y analizar fenómenos de **concurrencia** (procesos que se ejecutan simultáneamente) y la **sincronización** necesaria entre ellos para el uso compartido de recursos.

27. ¿Qué es un prototipo? ¿Cuáles son los proyectos candidatos a prototipar?

    Un prototipo es un producto parcialmente desarrollado que permite que clientes y desarrolladores examinen algunos aspectos del sistema propuesto y decidan si éste es adecuado para el producto terminado.

    Dado que el propósito fundamental del prototipo es la verificación de requerimientos, los proyectos candidatos ideales son aquellos que presentan una o más de las siguientes características:

    1. Requerimientos inciertos o ambiguos: proyectos donde el cliente no puede articular claramente sus necesidades o donde la visión inicial es vaga.
    2. Alta interfaz de usuario: sistemas donde la interacción del usuario y la funcionalidad de la interfaz son críticas. El prototipo permite que el cliente experimente la interfaz para asegurar que es intuitiva y funcional, verificando así los requisitos de usabilidad.
    3. Tecnología nueva o riesgosa: proyectos que utilizan tecnologías no probadas o donde existe riesgo técnico. El prototipo se convierte en una prueba de concepto para examinar la viabilidad técnica de un aspecto particular del sistema.
    4. Sistema donde los errores son costosos de corregir: la verificación es necesaria para evitar correcciones costosas en etapas posteriores. Por lo tanto, los proyectos con alta criticidad o donde una falla en la comprensión inicial de los requisitos puede generar grandes pérdidas, son candidatos principales para prototipar.

    En conclusión, cualquier proyecto donde exista una incertidumbre significativa en la comprensión de lo que se debe construir, o donde la interacción con el sistema es clave y debe ser validada por el cliente de forma temprana, es un candidato fuerte para el uso de prototipos.

28. ¿Cuáles son las técnicas de elicitación que conoce? Describa la técnica de muestreo de la documentación, formularios y datos existentes.

    - Métodos discretos: son menos perturbadores que otras formas de averiguar requerimientos. Se consideran insuficientes para recopilar información cuando se utilizan por sí solos, por lo que deben utilizarse junto con uno o varios de los métodos.
        - Muestreo de la documentación, formularios y datos existentes.
        - Investigación y visitas al lugar.
        - Observación del ambiente de trabajo.
    - Métodos Interactivos: Son utilizados para obtener los requerimientos de los miembros de la organización. Tienen en común la necesidad de **hablar y escuchar** para comprender a las personas.
        - Cuestionarios.
        - Entrevistas.
        - Planeación conjunta de requerimientos (JRP).

- Braninstorming.

Muestreo de la documentación, formularios y datos existentes: consiste en **observar** la información contenida en los documentos, formularios y datos ya existentes dentro de la organización algunos ejemplos serán organigramas, notas, minutas, registros contables, documentación de proyectos anteriores.

29. Describa las técnicas de especificación dinámicas y las estáticas. Enumere sus diferencias y de ejemplos.

    - Técnica de especificación estática: se describe el sistema a través de las entidades u objetos, sus atributos y relaciones con otros. No describe cómo las relaciones cambian con el tiempo.
    - Técnica de especificación dinámica: se considera un sistema en función de los cambios que ocurren a lo largo del tiempo. Considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado. Algunos ejemplos son Diagramas de Transición de Estados (DTE), redes de Petri , tablas de decisión.

30. Defina modelo de proceso. Describa los modelos prescriptivos y los descriptivos.

    Un modelo de proceso de software es una representación simplificada de un proceso de software que presenta una visión del mismo. Funciona como un marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.

    - **Modelos prescriptivos:** prescriben un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de la ingeniería de software, tareas, aseguramiento de la calidad y mecanismos de control.

    Cada modelo de proceso prescribe también un “flujo de trabajo”, es decir de qué forma los elementos del proceso se interrelacionan entre sí.

    - **Modelos descriptivos:** estos modelos se centran en la descripción de la forma en que el proceso se realiza en la realidad dentro de una organización o proyecto.

    Ambos deberían ser iguales.

31. La concesionaria de automóviles Autocard quiere optimizar sus procesos, para ello realizan un análisis sobre la conveniencia de compra de software genérico o personalizado. Describa las diferencias, ventajas y desventajas de cada una.

    - Software genérico: consiste en sistemas aislados producidos por organizaciones desarrolladoras de software y que se venden en un mercado abierto. Es un producto para un mercado amplio. Puede ser utilizado como una base que luego se adapta según lo necesario para el cliente.
      Al ser un producto ya existente, tiene una disponibilidad inmediata o muy rápida. Por venderse a muchos clientes, su costo inicial suele ser menor.
      Por otro lado, la desventaja es que la empresa podría tener que adaptar el software para que se ajuste a sus procesos específicos de optimización.
    - Software personalizado: el software personalizado se refiere a los **sistemas requeridos por un cliente particular**, es decir, que se desarrollan específicamente para la empresa.

      La diferencia principal con el software genérico es que este es un sistema desarrollado para un cliente particular y no se vende en el mercado abierto.

      Lo positivo es que tiene un ajuste perfecto a los requerimientos y procesos de optimización de Autocard, ya que se construye a medida.
      Pero lo negativo es que requiere mayor tiempo y costo de desarrollo inicial. Existe el riesgo de gastar tiempo y dinero construyendo el sistema erróneo si los requerimientos iniciales no se definen correctamente.

32. En IceSoft arman múltiples entrevistas para los distintos stakeholders para capturar los requerimientos. ¿Cómo procedería cuando los requerimientos de los stakeholders están en conflicto?

El conflicto entre requerimientos de distintos *stakeholders* es un problema de consistencia o coherencia del modelo de requerimientos. Como las entrevistas con múltiples *stakeholders* realizadas por IceSoft han revelado inconsistencias o ambigüedades en el modelo de requerimientos (por ejemplo, el departamento de ventas quiere una característica que la gerencia considera inviable) se debe aplicar alguna de las técnicas de verificación como la revisión formal de los requerimientos.

Este método implica la participación de todos los *stakeholders* relevantes (los que están en conflicto) para que revisen el documento de requerimientos. Este proceso permite la detección temprana de inconsistencias y ambigüedades.

33. ¿Qué es un requisito? Describa funcional y no funcional.

    Un requisito es una necesidad o una condición que el software debe cumplir. La especificación de requerimientos es el documento que contiene el conjunto de todos los requerimientos del sistema, tanto funcionales como no funcionales.

    Tipos de requisitos:

    1. Requerimientos funcionales: describen las **funciones** que el sistema deberá ofrecer a sus usuarios, se centra en el **comportamiento** del sistema.
    2. Requerimientos no funcionales: describe las restricciones y cualidades del sistema, centrándose en el cómo el sistema debe realizar sus funciones o en cuán bueno es el sistema.

       Estos incluyen:

    - **Restricciones de diseño:** limitaciones en la forma en que el sistema puede ser construido.
    - **Restricciones de proceso:** requerimientos sobre las herramientas, normas y estándares que deben utilizarse en el proceso de desarrollo.
    - **Requerimientos de calidad:** Aspectos como el rendimiento, la seguridad, la usabilidad o la fiabilidad del sistema.

34. ¿Cuáles son las responsabilidades profesionales y éticas de un ingeniero de software?

    - Confidencialidad: respetar la confidencialidad de sus empleados y clientes.
    - Competencia: no falsificar el nivel de competencia y aceptar responsabilidades fuera de su capacidad.
    - Derecho de la propiedad intelectual: conocer las leyes vigentes sobre las patentes y copyright.
    - Uso inapropiado de computadoras: no usar habilidades técnicas para actividades inapropiadas

35. Describa el modelo esencial.

    El modelo esencial o modelo del dominio del problema es el primer modelo que se genera en el proceso de desarrollo de software. Su objetivo principal es describir el dominio del problema y los requerimientos del usuario de una forma independiente de la implementación técnica que se le dará.

    Cuenta con las siguientes características:

    - Independencia de la tecnología: el modelo esencial debe ser completamente abstracto. Debe describir qué hará el sistema para resolver el problema, sin especificar cómo se implementará (es decir, sin hacer referencia a hardware, software o tecnología específica).
    - Foco en el dominio: Se centra en los elementos del dominio del problema, capturando los requisitos funcionales y no funcionales desde la perspectiva del usuario.
    - Contenido: El modelo esencial contiene el conjunto de todos los requerimientos del sistema, tanto funcionales como no funcionales.
    - Uso: Este modelo se utiliza para la verificación de requerimientos, asegurando que el modelo propuesto concuerde con las intenciones del usuario, lo cual es fundamental para evitar correcciones costosas en etapas posteriores del desarrollo.

36. ¿Cuál es la principal fuente de errores en la definición y análisis de requerimientos? ¿Por qué?
    La principal fuente de errores en la definición y análisis de requerimientos es la comunicación entre los *stakeholders* (clientes, usuarios) y el equipo de desarrollo.

    La razón es que si el proceso de comunicación falla, se genera una **falla en el entendimiento** del problema, los problemas se comunicación pueden ser

    - Dificultad para expresar claramente las necesidades.
    - No ser conscientes de sus propias necesidades.
    - No entender cómo la tecnología puede ayudar.
    - Miedo a parecer incompetentes por ignorancia tecnológica.
    - No tomar decisiones por no poder prever las consecuencias, no entender las alternativas o no tener una visión global.
    - Cultura y vocabulario diferentes.
    - Intereses distintos en el sistema a desarrollar.
    - Medios de comunicación inadecuados.
    - Conflictos personales o políticos.
    - Limitaciones cognitivas
        - No conocer el dominio del problema.
        - Hacer suposiciones sobre el dominio del problema.
        - Hacer suposiciones sobre aspectos tecnológicos.
        - Hacer simplificaciones excesivas.
    - Conducta humana
        - Conficlos y ambigüedades en los roles de los participantes.
        - Pasividad de cliente, usuarios o ingenieros de requisitos.
        - Temor a que el nuevo sistema lo deje sin trabajo.
    - Técnicos
        - Complejidad del dominio del problema.
        - Complejidad de los requisitos.
        - Múltiples fuentes de requisitos.
        - Fuentes de información poco claras.

37. ¿Qué tipo de requerimientos son?

    a. Los documentos a entregar deberán ajustarse a la reglamentación de la empresa.

       No funcional.

    b. El sistema debe administrar las ventas por internet.

       Funcional.

38. Defina qué son los stakeholders. De 4 ejemplos.

    Los *stakeholders* (partes interesadas) son los individuos u organizaciones que tienen un interés activo en el sistema que se está desarrollando, o que son afectados por él. Estos individuos son la fuente de los requerimientos y son cruciales para el proceso de Verificación de requerimientos (por ejemplo, en las Revisiones Formales), ya que son quienes deben asegurar que el modelo sea correcto contra sus intenciones.

Algunos ejemplos son

1. Clientes/usuarios: quienes interactúan directamente con el sistema para utilizar sus funciones y quienes expresan sus necesidades (la fuente de los requerimientos).
2. Desarrolladores: aunque están en el equipo de desarrollo, tienen un interés directo en los requerimientos para poder construir el sistema.
3. Equipo de pruebas: Tienen un interés en que los requerimientos sean testeables para poder realizar las demostraciones y convencer al cliente de que el sistema cumple lo pedido.
4. Diseñadores: tienen un interés en los requerimientos, ya que estos les indican qué funcionalidad y características va a tener el sistema resultante, sirviendo como guía para el diseño.

Agregar algunos ejemplos más.

39. Defina qué es un requerimiento no funcional. De 4 ejemplos.
    - El tiempo de respuesta de la búsqueda de clientes no debe superar los 2 segundos, incluso con 1 millón de registros. *Ejemplo de requerimiento de calidad.*
    - Todos los datos personales de los usuarios deberán ser cifrados utilizando el estándar AES-256.
    - El sistema debe desarrollarse utilizando la base de datos PostgreSQL y la interfaz de usuario debe ser accesible desde cualquier navegador moderno.
    - La documentación de desarrollo debe ajustarse a los requisitos de la norma ISO 9001.
40. Defina Técnicas de especificación estáticas. De 3 ejemplos.
    Las técnicas estáticas describen el sistema a través de las entidades u objetos, sus atributos y las relaciones que mantienen con otros elementos. Su principal limitación es que no describen cómo cambian esas relaciones o el estado del sistema con el paso del tiempo. Responden a la pregunta de qué es el sistema y qué datos maneja.

- Diagrama de Flujo de Datos (DFD): Esta herramienta visualiza el sistema como una red de procesos funcionales conectados por conductos y almacenamiento de datos. Aunque modela el flujo de la información, se enfoca en la estructura de los componentes y la forma en que los datos se mueven.
- Modelo Entidad-Relación (Implícito): Aunque no se nombra directamente, la descripción de las técnicas estáticas (entidades, atributos y relaciones) corresponde a este tipo de modelado, que define la estructura de los datos que maneja el sistema.
- Diccionario de Datos (Implícito): Al ser una técnica que documenta formalmente la estructura y el contenido de la información (entidades y atributos) en el sistema, se alinea con la descripción de las técnicas estáticas.

41. Explique qué es un Diagrama de Flujo de Datos (DFD).

    El DFD visualiza el sistema como una red de procesos funcionales conectados entre sí por flujos de información. Se utiliza para representar la estructura del sistema y cómo la información se mueve y se transforma dentro de él.

    Este diagrama se enfoca en los siguientes aspectos:

    - Procesos funcionales: descompone el sistema en las actividades o transformaciones que modifican o utilizan los datos.
    - Flujo de información: muestra cómo los datos fluyen desde una fuente externa, a través de los procesos funcionales, hasta llegar a su destino o a un lugar de almacenamiento.
    - Estructura: se considera una técnica estática porque describe la estructura de los componentes y la forma en que los datos se mueven a través de ellos, centrándose en qué datos maneja el sistema y no en su comportamiento temporal o concurrente.
42. Enuncie 4 principios de las Metodologías ágiles.
    - La mayor prioridad es satisfacer al cliente a través de fáciles y continuas entregas de software valuable.

- Los cambios de requerimientos son bienvenidos pero hace que el desarrollo sea tardío. Por lo tanto, los procesos ágiles capturan los cambios para que el cliente obtenga ventajas competitivas.
- Entregas frecuentes de software, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente.
- Usuarios y desarrolladores deben trabajar juntos durante todo el proyecto.
- Construir proyectos alrededor de motivaciones individuales.
- Darles el ambiente y el soporte que ellos necesitan y confiar el trabajo dado. El diálogo cara a cara es el método más eficiente y efectivo de intercambiar información entre el equipo de desarrolladores.
- El software que funciona es la medida clave del progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los stakeholders, desarrolladores y usuarios deberían ser capaces de mantener un paso constante indefinidamente.
- La atención continua a la excelencia técnica y buen diseño incrementa la agilidad.
- Simplicidad es esencial.
- las mejores arquitecturas, requerimientos y diseños surgen de la propia organización de los equipos.
- A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces refina y ajusta su comportamiento en consecuencia.