Final Preguntas Inge 2

1. **¿Qué es y para qué sirve un indicador?**

- **Definición:** Un indicador es un elemento de información que se construye combinando distintas métricas.
- **Para qué sirve:** Su función principal es establecer una "pauta" o estándar de qué se considera una medida correcta para un atributo específico. Esto le permite al gestor del proyecto realizar ajustes en el producto, el proceso o el proyecto basándose en datos concretos proporcionados por el indicador.
- **Explicación fácil:** Imagina que las métricas son los "ingredientes" (como cuántos errores hay o cuánto tiempo se tardó) y el indicador es la "receta terminada" que te dice si el plato salió bien o si le falta sal (ajuste).

Ubicación: Página 18 del PDF de contenido.

2. **¿Cuáles son los requerimientos no funcionales que se ven afectados por la arquitectura?**

La arquitectura tiene un impacto directo y crítico en los siguientes requerimientos no funcionales:

- **Rendimiento:** Se optimiza agrupando operaciones críticas en pocos subsistemas para evitar comunicaciones excesivas que lo ralenticen.
- **Seguridad:** Se logra mediante arquitecturas en capas, protegiendo los datos más críticos en las capas más internas.
- **Protección:** Se busca centralizar las operaciones de protección en un solo subsistema para reducir costos y facilitar la validación.
- **Disponibilidad:** Se asegura diseñando componentes redundantes que puedan reemplazarse sin detener el sistema (tolerancia a fallos).
- **Mantenibilidad:** Requiere componentes pequeños (granularidad fina) y autocontenidos que sean fáciles de cambiar.

Ubicación: Página 66 del PDF de contenido.

3. ¿Qué es un riesgo? ¿Cuáles son los ítems de más alto riesgo según Boehm?

- **Definición de riesgo:** Es un evento no deseado que, de ocurrir, trae consecuencias negativas para el proyecto. Se caracteriza por tener **incertidumbre** (la probabilidad de que pase) y **pérdida** (el impacto o daño que causaría).
- **Ítems de alto riesgo según Boehm:** Si bien el PDF menciona que Boehm sugiere supervisar los **10 riesgos más altos** de la lista (ordenados por impacto y probabilidad), también destaca factores de riesgo clave como:
    - Falta de compromiso de los usuarios finales.
    - No entender por completo los requisitos.
    - Ámbito o requisitos del proyecto inestables.
    - Falta de habilidades adecuadas en el equipo.

Ubicación: Páginas 51, 52 y 55 del PDF de contenido

4. ¿Cuáles son las consideraciones que se deben tener en cuenta a la hora de diseñar una interfaz?

El diseño de la interfaz de usuario (UI) debe centrarse en que la comunicación entre el hombre y la máquina sea eficiente y fácil de aprender. Las principales consideraciones son:

- **Adaptación al usuario:** La tecnología debe adaptarse a las personas, no al revés. Se debe considerar la diversidad de los usuarios (edad, nivel de estudios, experiencia técnica).
- **Prevención de errores:** La interfaz debe ser lo suficientemente clara para evitar que el usuario se equivoque, ya que una UI compleja genera rechazo.
- **Inclusividad y Eficiencia:** Debe permitir un acceso rápido al contenido sin sacrificar la comprensión, integrando diversas tecnologías (sonido, video, etc.) y hardware (teclado, ratón, pantallas táctiles).

  Las 3 Reglas de Theo Mandel:

  a. Dar el control al usuario.
  b. Reducir la carga de memoria del usuario (no obligarlo a recordar muchas cosas).
  c. Lograr que la interfaz sea consistente (que siempre funcione igual).

  Ubicación: Páginas 12, 86, 87 y 95 del PDF de contenido.

5. ¿Qué es la gestión de configuración? Defina línea base y ejemplifique

- **Gestión de Configuración del Software (GCS):** Es un proceso para identificar, controlar y rastrear los cambios en los elementos de un software (código, documentos, datos) a lo largo de su vida útil.

- **Línea Base (Baseline):** Es un punto de referencia en el desarrollo del software que ha sido revisado y acordado formalmente. Una vez establecida, cualquier cambio sobre ella debe seguir un proceso formal de control.
- **Ejemplo:** Una "Especificación de Requisitos" aprobada por el cliente es una línea base. Si a mitad del proyecto el cliente quiere agregar un botón nuevo, no se cambia el documento así nomás; se debe pedir un cambio formal porque ya se había "cerrado" esa versión base para empezar a programar.

Ubicación: Páginas 7 y 139 del PDF de contenido

6. Rejuvenecimiento.

- **Concepto:** Es una parte del mantenimiento que busca mejorar la calidad global de un sistema que ya existe para evitar que se degrade o falle por su "envejecimiento".

Sirve para alargar la vida útil del sistema.

Tipos de Rejuvenecimiento:

1. **Re-Documentación:** Analizar el código para generar manuales o explicaciones de cómo funciona.

2. **Re-Estructuración:** Cambiar el código internamente para que sea más ordenado y fácil de manejar, sin cambiar lo que hace.
3. **Ingeniería Inversa:** A partir del código fuente, intentar recrear los documentos de diseño que se perdieron o nunca se hicieron.
4. **Re-Ingeniería:** Es lo más completo; se usa ingeniería inversa para entender el sistema y luego se crea un código nuevo mejor estructurado.

Ubicación: Páginas 7, 144 y 145 del PDF de contenido

7. **Calcule el camino crítico**

Tarea	Duración	Dependencia
A	30 min	-
B	10 min	A
C	120 min	B
D	15 min	A
E	80 min	A
F	20 min	A
G	45 min	C, I
H	25 min	G
I	60 min	D
J	25 min	B, I
K	50 min	D
L	30 min	D
M	5 min	J, K, L
N	100 min	E
O	10 min	F
P	10 min	C, H, M, N, O

8. Tipos de planificación organizacional vistos.

- Existen tres tipos principales de estructuras organizativas para los equipos de software:

1. **Descentralizado Democrático (DD):** No tiene un jefe permanente. Las decisiones se toman por consenso y la comunicación es horizontal entre todos. Es bueno para problemas complejos y a largo plazo.
2. **Descentralizado Controlado (DC):** Tiene un líder definido que coordina tareas y líderes secundarios para subtareas. La resolución de problemas es grupal, pero la implementación se divide. La comunicación es tanto horizontal como vertical.

3. **Centralizado Controlado (CC):** Tiene un jefe que toma las decisiones y coordina todo. La comunicación es vertical (del jefe hacia abajo). Es eficaz para tareas simples o urgentes.

Ubicación: Página 50.

9. Característica de todos los riesgos.

- Todos los riesgos comparten dos características fundamentales:
1. **Incertidumbre:** El riesgo puede ocurrir o no; es decir, hay una probabilidad (nunca del 100%, porque si no sería un problema real y no un riesgo).
2. **Pérdida:** Si el riesgo se convierte en realidad, siempre traerá consecuencias negativas o pérdidas para el proyecto.

Ubicación: Página 51.

10. Definición de recuperabilidad (principios de diseño).

En el contexto del diseño (específicamente de interfaces y usabilidad), la recuperabilidad se refiere a la capacidad del sistema para permitir al usuario corregir sus errores.

- **Explicación fácil:** Si te equivocas al hacer clic o borrar algo, el sistema debe ofrecerte una forma de "deshacer" esa acción o recuperarte del fallo sin perder tu trabajo. Un diseño es recuperable si es tolerante a los errores del usuario.

Ubicación: Páginas 98 (mencionado como tolerancia a errores en usabilidad) y 112 (como tipo de defecto).

11. Diferencia entre modelo repositorio y modelo cliente-servidor.

- Modelo Repositorio: Todos los subsistemas comparten una base de datos central única. Los datos se gestionan en un solo lugar y los módulos acceden a él. Es eficiente para compartir muchos datos, pero si el repositorio falla, todo el sistema cae.
- Modelo Cliente-Servidor: El sistema se divide en proveedores de servicios (servidores) y consumidores (clientes). Los datos y funciones están distribuidos. Es más flexible y escalable, pero la comunicación depende de la red.
- Diferencia clave: En el repositorio todo gira en torno a los datos centrales; en cliente-servidor, todo gira en torno a los servicios distribuidos.

Ubicación: Páginas 68 y 69.

12. Describa el problema de las 4 P.

- Se refiere a los 4 elementos fundamentales que deben gestionarse en un proyecto de software para que sea exitoso:
1. **Personal (People):** Es el factor más importante. Se debe gestionar a las personas, sus roles y habilidades.

2. **Producto (Product):** Se debe definir claramente qué se va a construir (alcance y objetivos) antes de empezar.
3. **Proceso (Process):** Se debe elegir un marco de trabajo (metodología) adecuado para que el equipo pueda cumplir con el plan.
4. **Proyecto (Project):** Se debe planificar y controlar el trabajo para evitar el caos y asegurar la calidad.

Ubicacion: pagina 16

13. ¿Cuáles son los elementos claves de la gestión de proyectos?

- Para asegurar el éxito de un proyecto en tiempo, costo y calidad, se deben integrar los siguientes elementos:

1. Métricas.
2. Estimaciones.
3. Planificación temporal (calendario).
4. Planificación organizativa (equipo).
5. Análisis de riesgos.
6. Seguimiento y control.

Ubicación: Página 17.

14. Describa el modelo MOI.

- Es un modelo de liderazgo para mejorar la gestión del equipo de software. Sus siglas significan:

- **M (Motivación):** La habilidad del líder para incentivar al personal técnico a que dé su máximo rendimiento.
- **O (Organización):** La capacidad para estructurar procesos que permitan transformar el trabajo en un producto final.
- **I (Innovación):** La capacidad de fomentar nuevas ideas y creatividad en el equipo.

Ubicación: Página 48.

15. ¿Qué define el diseño arquitectónico? Describa los tipos de organización del sistema.

- **Definición:** El diseño arquitectónico define la relación entre los componentes estructurales principales del software. Es como el plano general de la casa antes de diseñar cada habitación.

Tipos de organización:

- **Repositorio:** Datos centrales compartidos.
- **Cliente-Servidor:** Servicios distribuidos.
- **Capas (Abstract Machine):** El sistema se organiza en capas jerárquicas, donde cada capa presta servicio a la superior.

Ubicación: Páginas 58, 67, 68, 70.

16. Enumere y describa los principios de diseño.

Para lograr un buen diseño se deben seguir principios como:

- Modularidad: Dividir el sistema en partes pequeñas (módulos) independientes.
- Cohesión: Que cada módulo haga una sola cosa y la haga bien.
- Acoplamiento: Que los módulos dependan lo menos posible unos de otros.
- Abstracción: Ocultar los detalles complejos y mostrar solo lo necesario.
- Refinamiento: Ir de lo general a lo detallado paso a paso.

Ubicación: Páginas 62, 63, 64.

17. ¿Qué es un proyecto?

Es un esfuerzo planificado y controlado para crear un producto de software. Implica comprender los factores de éxito, gestionar riesgos, definir objetivos claros, asignar recursos y monitorear el progreso para cumplir con los plazos y la calidad.

Ubicacion pagina 16.

18. En planificación temporal, ¿Qué tareas se deben realizar para hacer el seguimiento y control de proyecto?

- Para controlar que el proyecto no se desvíe del plan (tiempo y costo), se deben realizar tareas como:
    - Monitorear el progreso regularmente comparando lo real vs. lo planificado.
    - Gestionar el alcance para evitar cambios descontrolados.
    - Analizar riesgos y tomar acciones correctivas si hay retrasos.
    - Utilizar métricas de estado y uso del tiempo.

Ubicación: Páginas 16, 17, 19.

19. ¿Cuáles son los tipos (áreas) de diseño de software? Describa.

El diseño se divide en cuatro áreas principales:

1. **Diseño de Datos:** Transforma los objetos de datos en estructuras de datos (como bases de datos).
2. **Diseño Arquitectónico:** Define la estructura general y la relación entre los componentes principales.
3. **Diseño de Interfaz:** Describe cómo se comunica el software con el usuario y con otros sistemas.
4. **Diseño de Componentes (o Procedimental):** Transforma la arquitectura en una descripción detallada de cómo funciona cada módulo por dentro.

Ubicación: Páginas 57, 58, 59.

20.¿Qué es verificación? ¿Qué es validación? De todos los tipos de pruebas vistos, ¿Cuál dirías que se está verificando y cuál que se está validando?

- **Verificación:** ¿Estamos construyendo el producto correctamente? Se enfoca en si el software cumple con las especificaciones técnicas y de diseño. (Ej: Pruebas unitarias, de integración, de caja blanca).
- **Validación:** ¿Estamos construyendo el producto correcto? Se enfoca en si el software cumple con lo que el cliente realmente necesita y quiere. (Ej: Pruebas de aceptación, de caja negra, validación con el cliente).

Ubicación: Página 120.

21. ¿Qué es el rejuvenecimiento del software? Describa los tipos.

Es mejorar la calidad de un sistema existente para alargar su vida útil. Tipos:

- **Re-Documentación:** Crear documentación nueva del código para entenderlo mejor.
- **Re-Estructuración:** Mejorar la estructura interna del código sin cambiar su función.
- **Ingeniería Inversa:** Analizar el código para recuperar el diseño original.
- **Re-Ingeniería:** Rediseñar y reescribir partes del sistema para modernizarlo.

Ubicación: Páginas 144, 145.

22. La empresa de pastas HOGAREÑAS S.A desea hacer un estudio del tiempo de demora en hacer una lasaña, para esto ha identificado las siguientes actividades en minutos. Encontrar el camino crítico, calculando tiempos tempranos y tiempos tardíos. Calcular el tiempo final.

ACTIVIDADES		PREDECESOR	DURACIÓN
A	Comprar queso		30
B	Rayar queso	A	5
C	Batir huevos		2
D	Mezclar huevos y queso	C, B	3
E	Picar cebollas y hongos		7
F	Cocinar la salsa de tomate	E	25
G	Hervir agua		15
H	Hervir la pasta de lasaña	G, F	10
I	Enjuagar la pasta de lasaña	H	2
J	Unir los ingredientes	I, F, D, B	10
K	Precalentar horno		15
L	Hornear la lasaña	J, K	30

23. ¿Cómo se clasifican los riesgos?

Se pueden clasificar en tres categorías generales:

1. **Riesgos del Proyecto:** Amenazan el plan del proyecto (presupuesto, calendario, recursos).
2. **Riesgos Técnicos:** Amenazan la calidad y la implementación técnica (tecnología nueva, dificultad de diseño).
3. **Riesgos del Negocio:** Amenazan la viabilidad del producto en el mercado (nadie lo quiere, pierde dinero).

Ubicación: Página 51 (implícito en la descripción general de riesgos).

24. ¿Cómo puede ser la organización del sistema? Diseño arquitectónico)

.Esta pregunta es similar a la 15. La organización puede ser:

- Centralizada (Repositorio).
- Distribuida (Cliente-Servidor).
- Jerárquica (Capas).

Ubicación: Página 67.

25. Explique el PCMM (modelo de capacitación y motivación del personal).

Nota importante: En el PDF proporcionado *no* aparece explícitamente el término "PCMM" (People Capability Maturity Model). Sin embargo, el documento trata este tema bajo el "Modelo MOI de Liderazgo" (página 48), que se enfoca en Motivación, Organización e Innovación para gestionar el capital humano, que es el activo más valioso.

26. Enumere y describa las técnicas de estimación que conozca.

Sirven para predecir tiempo, costos y recursos:

- **Juicio Experto:** Consultar a personas con experiencia para que den su opinión.
- **Técnica Delphi:** Un grupo de expertos estima de forma anónima y luego discuten las diferencias hasta llegar a un consenso.
- **Planning Poker:** Técnica ágil donde el equipo usa cartas con números (serie de Fibonacci) para estimar el esfuerzo de las tareas en conjunto.
- **Puntos de Función / LDC:** Estimaciones basadas en fórmulas matemáticas usando el tamaño o funcionalidad del software.

Ubicación: Páginas 33, 34.

27. Defina y diferencia pruebas de caja blanca y de caja negra. De un ejemplo de cada una.

- Caja Blanca (Cristal): Se conoce el código interno. Se prueban los caminos lógicos, bucles y condiciones internas del programa.

    - *Ejemplo:* Probar que un `if` en el código funcione tanto para el "verdadero" como para el "falso".

- Caja Negra (Opaca): No se mira el código. Se prueba la funcionalidad basándose en qué entra y qué debe salir según los requisitos.
    - *Ejemplo:* Ingresar un usuario y contraseña válidos en un login y verificar que el sistema permita el acceso, sin importar cómo lo haga por dentro.

Ubicación: Páginas 113, 118.

28. ¿Qué consideraciones se deberían tener en cuenta a la hora de desarrollar una interfaz de usuario?

Se debe priorizar la **experiencia del usuario (UX)**:

- Dar el control al usuario (que pueda deshacer acciones).
- Reducir la carga de memoria (que no tenga que recordar cosas de una pantalla a otra).
- Hacer la interfaz consistente (que los botones y menús sean siempre iguales).
- Adaptarse a la diversidad de usuarios (principiantes y expertos).

Ubicación: Página 86, 95.

29. Defina métricas del proceso y del producto/objetivas y subjetivas. ¿Qué son las métricas post mortem y cuáles las tempranas?

- **Métricas del Proceso:** Miden cómo trabajamos (eficiencia, calidad del flujo de trabajo) para mejorar a largo plazo.
- **Métricas del Producto:** Miden las características del software en sí (calidad, fiabilidad, complejidad) en un momento dado.
- **Métricas Post Mortem:** Se analizan *después* de terminar el proyecto para aprender de los errores y éxitos (lecciones aprendidas).
- **Métricas Tempranas:** Se usan al *inicio* (durante el diseño o análisis) para predecir problemas antes de programar.

Ubicación: Páginas 19, 20, 22.

30.

. **Desarrolle un PERT de las siguientes actividades**

Descripción	Tarea	Precedencia	Duración esperada en días
Realizar entrevistas	A	Ninguna	4
Aplicar cuestionarios	B	A	4
Leer informes de la organización	C	Ninguna	8
Analizar el flujo de datos	D	B, C	3
Introducir prototipos	E	B, C	8
Observar las reacciones hacia el prototipo	F	E	5
Realizar el análisis de costo/ beneficio	G	D	8
preparar la propuesta	H	F, G	2
Presentar la propuesta	I	H	2

31. Enumere las actividades del análisis de riesgo.

1. **Identificación:** Hacer una lista de todos los posibles riesgos (brainstorming).
2. **Análisis/Estimación:** Determinar la probabilidad de que ocurran y el impacto (daño) que causarían.
3. **Planificación (Gestión):** Decidir qué hacer si ocurren (mitigar, evitar, aceptar).
4. **Supervisión (Control):** Vigilar los riesgos durante todo el proyecto.

Ubicación: Página 53, 54.

32. Describa los fundamentos del diseño. ¿Qué es la cohesión funcional? Describa los grados de cohesión.

- Fundamentos: Modularidad, abstracción, refinamiento.
- Cohesión: Medida de cuán relacionadas están las tareas *dentro* de un solo módulo. Se busca que sea alta.
- Cohesión Funcional (La mejor): El módulo hace una sola cosa específica y nada más.
- Grados de Cohesión (de peor a mejor):
    - Coincidental (tareas sin relación).
    - Lógica (tareas similares, ej. "todas las entradas").
    - Temporal (se ejecutan al mismo tiempo).
    - Procedimental (parte de una secuencia).
    - Comunicacional (usan los mismos datos).

    - Secuencial (la salida de uno es entrada de otro).
    - Funcional (una sola tarea definida).

Ubicación: Página 63.

33. Enumere las técnicas de validación que conozca.

La validación asegura que el software cumple con los requisitos del cliente.

- **Pruebas de Aceptación:** El usuario prueba el sistema final.
- **Pruebas Alfa:** El usuario prueba en el entorno del desarrollador.
- **Pruebas Beta:** El usuario prueba en su propio entorno real sin el desarrollador presente.
- **Revisión con el cliente:** Mostrar prototipos o versiones preliminares.

Ubicación: Página 131 (Pruebas Alfa/Beta), 120.

34. ¿Qué tipos de mantenimiento conoce? Defina ingeniería inversa y reingeniería.

La validación asegura que el software cumple con los requisitos del cliente.

- **Pruebas de Aceptación:** El usuario prueba el sistema final.
- **Pruebas Alfa:** El usuario prueba en el entorno del desarrollador.
- **Pruebas Beta:** El usuario prueba en su propio entorno real sin el desarrollador presente.
- **Revisión con el cliente:** Mostrar prototipos o versiones preliminares.
- **Ubicación:** Página 131 (Pruebas Alfa/Beta), 120.

35. describa el concepto de cohesión y sus grados.

- **Fundamentos:** Modularidad, abstracción, refinamiento
- **Cohesión:** Medida de cuán relacionadas están las tareas *dentro* de un solo módulo. Se busca que sea **alta**
- **Cohesión Funcional (La mejor):** El módulo hace una sola cosa específica y nada más.
- **Grados de Cohesión (de peor a mejor):**
    - Coincidental (tareas sin relación).
    - Lógica (tareas similares, ej. "todas las entradas").
    - Temporal (se ejecutan al mismo tiempo).
    - Procedimental (parte de una secuencia).
    - Comunicacional (usan los mismos datos).
    - Secuencial (la salida de uno es entrada de otro).
    - Funcional (una sola tarea definida).

Ubicación: Página 63.

36. Defina las estrategias de integración.

Son las formas de unir los módulos para probarlos juntos:

- **Big Bang:** Unir todo de golpe (no recomendado, difícil encontrar errores).
- **Descendente (Top-Down):** Empezar por el módulo principal e ir bajando. Se usan "stubs" (módulos falsos) para simular los de abajo.
- **Ascendente (Bottom-Up):** Empezar por los módulos de más abajo (los más detallados) e ir subiendo. Se usan "drivers" (controladores) para probarlos.

Ubicación: Páginas 125, 126.

37. ¿Qué es la barrera del mantenimiento?

Es el punto crítico económico y técnico donde ya no conviene seguir manteniendo un sistema viejo porque es demasiado costoso, riesgoso o difícil. En este punto, es más rentable tirar el sistema viejo y construir uno nuevo desde cero que seguir arreglándolo.

- Ubicación: Página 141.

Finales Ingeniería de Software II

Una empresa de fletes de La Plata ha hecho convenios con otras empresas más pequeñas en el resto del país para optimizar los viajes y lograr mejores beneficios. Para eso necesita gestionar la información de los viajes de manera óptima. En una primera etapa piensan trabajar con 20 empresas con proyección a ir creciendo cada semestre. Los requerimientos se presentan muy variantes (cambiantes) tanto funcionales como no funcionales. Se solicita que la interfaz sea intuitiva y pensada para evitar errores y poder informar claramente cómo actuar en cada caso.

[(1) Mencione y describa dos principios de Nielsen que apunten al feedback y la prevención de errores que son dos de los problemas detectados en la interface].

Rta: Pagina 94
Teniendo en cuenta que la interfaz debe ser intuitiva y evitar errores, los dos principios (heurísticas) clave son:

1. Visibilidad del estado del sistema (Feedback): El sistema debe mantener siempre informado al usuario de lo que está ocurriendo mediante retroalimentación adecuada en un tiempo razonable.
    - Ejemplo en el caso: Si el operador asigna un viaje, el sistema debe mostrar un mensaje verde que diga "Viaje asignado con éxito" o un ícono de "cargando" mientras busca la ubicación, para que no crea que se trabó.
2. Prevención de errores: Es mejor diseñar un sistema que prevenga el error antes que uno que solo dé buenos mensajes de error.

- Ejemplo en el caso: En el formulario de "solicitar flete", en lugar de dejar que el usuario escriba la fecha manualmente (y pueda poner una fecha pasada por error), poner un calendario desplegable que tenga bloqueados los días anteriores a hoy.

Tiene que soportar una fuerte demanda de acceso y con un rendimiento óptimo las 24 horas del día. Para ello deciden hacer un sitio donde presentar sus servicios y que la gente pueda solicitarlos sin importar donde se encuentren. El software por desarrollar parece ser de tamaño medio, pero se torna complejo por la organización de rutas de reparto y geolocalización de cada entrega que solicitaron para ser vistas en el sistema, esto va a hacer que el equipo esté mucho tiempo trabajando junto abordando nuevas tareas y capacitaciones para desarrollar el producto. Deciden solicitar el desarrollo a SoftIT una empresa con amplia experiencia en Ingeniería de Software. En la empresa SoftIT aceptaron el trabajo, y, dada las características del desarrollo decidieron Identificar y definir los elementos en el sistema, para controlar sus cambios a lo largo de todo el desarrollo.
[(2) a) ¿A qué estaría haciendo referencia? b) ¿Cuáles serían los elementos que controlar? Dé al menos 2 ejemplos.] *Leandro, líder del proyecto piensa en la mejor constitución del equipo que lo desarrollará.*

Rta: Pagina 138 y 139
a) Hace referencia a la Gestión de Configuración del Software (GCS). Es el conjunto de actividades diseñadas para controlar los cambios, identificando los productos de trabajo que es probable que cambien y auditando que las modificaciones se hagan correctamente.

b) Elementos a controlar: Se llaman Items de Configuración de Software (ICS) o Configuration Items. Son la unidad básica de lo que se va a controlar.

Ejemplo 1: El Documento de Especificación de Requisitos (SRS), ya que el texto dice que los requerimientos son "muy variantes", este documento cambiará mucho.

Ejemplo 2: El Código Fuente de los módulos de geolocalización.

[(3) Enumere al menos 3 factores a considerar cuando se planifica una estructura de equipo]. *Bruno (socio de Leandro), identifica los riesgos a cubrir. Le preocupa la decisión y luego de ordenarlos trazó una línea de corte.*

Rta: Página: 50.
Respuesta: Leandro debe considerar la naturaleza del proyecto para armar el equipo. Tres factores clave son:

1. La dificultad del problema: El texto menciona que hay "geolocalización" y "rutas de reparto", lo cual añade complejidad técnica.
2. El tamaño del programa: Se menciona que el software es de "tamaño medio", lo que influye en cuánta gente se necesita.
3. La estabilidad de los requisitos: El caso dice explícitamente que los requisitos son "muy variantes/cambiantes", lo que suele requerir equipos más flexibles o ágiles. (Otros factores válidos: tiempo que el equipo permanecerá unido, grado de modularización).

[(4) a) Defina qué es la línea de corte.
b) Teniendo en cuenta la decisión del inciso anterior, enumere 3 riesgos de este proyecto, cada uno con su estrategia]. *Leandro sabe de la importancia de tener todo debidamente probado con lo cual sugiere diferentes técnicas de caja blanca y caja negra, y va realizando las pruebas de integración.*

Rta: Página: 55.

a. Línea de corte: Es una técnica (usada en la tabla de Boehm) donde se ordenan los riesgos según su probabilidad e impacto. Se traza una línea horizontal; los riesgos que quedan por encima de la línea son los que se deben gestionar activamente (con planes de contingencia), y los que quedan por debajo solo se supervisan por si suben de categoría.
b. 3 Riesgos del proyecto y sus estrategias:

1. Riesgo: Alta inestabilidad de requisitos (el cliente cambia de idea a cada rato).
    a. Estrategia: Mitigación. Usar un modelo incremental y hacer prototipos rápidos de la interfaz para congelar requisitos por etapas.

2. Riesgo: Falta de experiencia técnica en geolocalización (se menciona que necesitan "capacitaciones").
    a. Estrategia: Mitigación. Contratar una capacitación externa inmediata o asignar tiempo extra en el cronograma para la curva de aprendizaje (Riesgo de personal/técnico).

3. Riesgo: Problemas de rendimiento con la "fuerte demanda 24hs".
    a. Estrategia: Supervisión. Realizar pruebas de carga y estrés tempranas para vigilar si el servidor aguanta la concurrencia proyectada.

[(6) Indique qué integración utilizaría, porqué y describa brevemente la misma]. La empresa de fletes decide comprar un nuevo vehículo para sumar a la flota:

Rta: Página: 126 y 127.

Recomendaría la Integración Sándwich (o Híbrida).

Porque el caso menciona que la interfaz es crítica ("intuitiva", "feedback") y los requisitos cambian mucho: esto pide una integración Descendente (Top-Down) para probar la interfaz y el flujo principal cuanto antes.

Pero también hay módulos complejos de bajo nivel como la "geolocalización" y "rutas": esto pide una integración Ascendente (Bottom-Up) para asegurar que los cálculos complejos funcionen bien antes de unirlos.

Descripción breve: La integración Sándwich combina ambas. Se prueban los niveles superiores (interfaz) con "stubs" (módulos falsos) y simultáneamente los niveles inferiores (geolocalización) con "drivers" (controladores), encontrándose en el medio del sistema.

	Tarea	Precedida por	Duración
A	Hacer un estudio de factibilidad	-	3
B	Lista de los modelos posibles	A	1
C	Investigar todos los modelos posibles	B	3
D	Entrevista con el mecanismo	C	1
E	Obtener precios en las agencias	C	2
F	Compilar los datos adecuados	B,E	1
G	Escoger los tres modelos mejores	F	2
H	Conducción de prueba con los 3 modelos	G	3
I	Pedir datos de garantía y de financiamiento	G	3
J	Escoger un vehículo	H,I	1
K	Comprar en automóvil	F,G,J	2

(7)Calcular las fechas más tempranas, más tarifas, realizar el grafico y encontrar el camino critico] Video donde lo explica: 8_Clase 5 Planificación Temporal - 2022

1. Describe 4 criterios de diseño de software.

(Página 59) Para crear un buen diseño de software, se deben seguir ciertos criterios técnicos fundamentales:

1. **Modularidad:** El sistema debe dividirse en partes más pequeñas (subsistemas o componentes) que sean independientes y tengan una función clara. Esto hace que sea más fácil de entender y arreglar.
2. **Interfaces simplificadas:** Las conexiones entre los módulos deben ser lo más simples posible. Si un módulo necesita hablar con otro, debe hacerlo de forma clara y sin enredos complejos.
3. **Independencia funcional:** Cada componente debe encargarse de una sola cosa específica. Si un módulo hace de todo, es un mal diseño.

4. **Representaciones múltiples:** El diseño debe poder verse desde distintos ángulos (como un plano de datos, un plano de estructura, un plano de interfaz) para entenderlo por completo.

2. Menciona los paradigmas de equipo y describe 1.

(Página 50) El texto presenta tres estructuras u organigramas de equipo genéricos (paradigmas de organización):

1. Democrático Descentralizado (DD)
2. Descentralizado Controlado (DC)
3. Centralizado Controlado (CC)

- Descripción del Centralizado Controlado (CC):
    - Cómo funciona: Hay un jefe absoluto que toma todas las decisiones y coordina todo el trabajo. La comunicación es vertical (del jefe hacia abajo).
    - Cuándo es útil: Es ideal para resolver problemas sencillos o tareas que requieren mucha rapidez, ya que no se pierde tiempo discutiendo decisiones.

3. Menciona 4 principios de Nielsen y describe 1.
(Páginas 99 y 100) Los principios de usabilidad de Jacob Nielsen para mejorar la interfaz de usuario son:

1. Diálogo simple y natural.
2. Lenguaje de usuario (hablar como la persona, no como la máquina).
3. Minimizar el uso de la memoria del usuario.
4. Consistencia.
5. Feedback (Retroalimentación).

- Descripción de "Feedback":
    - Qué es: El sistema siempre debe informar al usuario sobre lo que está pasando.
    - Ejemplo: Si el usuario hace clic en "Guardar", el sistema debe mostrar un mensaje de "Guardando..." o un ícono de carga. Si no muestra nada, el usuario pensará que se trabó y volverá a hacer clic, generando errores.

4. 2 ventajas y 2 desventajas del patrón repositorio.

(Páginas 68 y 69) Este patrón organiza el sistema alrededor de una base de datos central donde todos los subsistemas guardan y leen información.

- **Ventajas:**
    1. **Eficiencia en datos grandes:** Es excelente para compartir grandes volúmenes de información sin tener que estar pasándolos de un subsistema a otro.
    2. **Centralización del control:** Es más fácil hacer copias de seguridad (backup) y gestionar la seguridad porque todo está en un solo lugar.
- **Desventajas:**

1. **Punto único de fallo:** Si el repositorio central se rompe, todo el sistema deja de funcionar.
2. **Dificultad de evolución:** Si querés cambiar la estructura de la base de datos, tenés que actualizar todos los subsistemas que la usan, lo cual es muy costoso y difícil.

5. Cuándo usar juicio experto.

(Página 33) El juicio experto es una técnica de estimación que se basa en consultar a personas con experiencia. Se debe usar cuando:

- **No hay datos históricos:** Es un proyecto nuevo y no tenemos registros anteriores para comparar.
- **Proyectos de alto impacto:** Cuando hay mucho en juego y se necesita la validación de alguien que sepa mucho del tema antes de avanzar.
- **Nuevos dominios:** Cuando el equipo técnico nunca trabajó con esa tecnología o negocio específico y necesita la guía de un especialista.

1. ¿Qué es y para qué sirve un indicador?
Página 18
Definición: Un indicador es un elemento de información que se construye combinando distintas métricas.
Para qué sirve: Sirve para evaluar el estado actual de un proceso o proyecto y compararlo con un objetivo predefinido. Básicamente, te da una "señal" de si vas bien o mal. Permite a los gestores tomar decisiones informadas para ajustar el producto, el proceso o el proyecto.
Ejemplo: Imaginate que las "métricas" son los ingredientes (cantidad de errores encontrados, horas trabajadas), y el "indicador" es la receta final que te dice "La calidad está bajando" o "Estamos atrasados".

2. ¿Cuáles son los requerimientos no funcionales que se ven afectados por la arquitectura?
Página 66
La arquitectura del software tiene un impacto directo sobre la calidad del sistema, afectando principalmente a estos requerimientos no funcionales:

- Rendimiento: Si la arquitectura tiene demasiadas comunicaciones entre componentes, el sistema puede volverse lento. Se optimiza usando componentes de grano grueso (más grandes y menos numerosos).
- Seguridad: Una arquitectura en capas permite poner los activos más críticos en las capas internas para protegerlos mejor.
- Protección (Safety): Se suele usar un solo subsistema crítico para monitorear la seguridad, reduciendo costos de validación.
- Disponibilidad: Se logra con componentes redundantes (duplicados) para que si uno falla, otro tome su lugar sin detener el sistema.
- Mantenibilidad: Se favorece usando componentes pequeños y autocontenidos que sean fáciles de cambiar sin romper todo lo demás.

3. ¿Qué es un riesgo? ¿Cuáles son los ítems de más alto riesgo según Bohem?

Página 51, 52 y 55

- Definición de Riesgo: Es un evento no deseado que tiene dos características clave: Incertidumbre (puede pasar o no) y Pérdida (si pasa, trae consecuencias negativas).
- Ítems de alto riesgo (Top 10 de Boehm): Boehm sugiere monitorear los riesgos más graves. Algunos de los principales factores que menciona el documento (pág. 52) son:

1. Falta de compromiso de los usuarios finales.
2. No entender completamente los requisitos.
3. Alcance o requisitos del proyecto inestables (cambian mucho).
4. Falta de habilidades adecuadas en el equipo.
5. Expectativas no realistas.

4. ¿Cuáles son las consideraciones que se deben tener en cuenta a la hora de diseñar una interfaz?

Página 86, 87 y 95

El objetivo principal es crear una comunicación eficiente entre el humano y la máquina. Las consideraciones clave son:

- Adaptación al usuario: La tecnología debe adaptarse a la persona, no al revés. Hay que considerar quién va a usar el sistema (edad, conocimientos técnicos).
- Prevención de errores: Diseñar para que el usuario no se equivoque, ya que una interfaz confusa genera rechazo.
- Las 3 Reglas de Oro de Theo Mandel:
    1. Dar el control al usuario: Que pueda deshacer acciones y sentir que maneja el sistema.
    2. Reducir la carga de memoria: No obligar al usuario a recordar cosas de una pantalla a otra.
    3. Hacer la interfaz consistente: Que los botones y menús siempre funcionen igual en todas partes.
- Inclusividad: Permitir el uso de diversos periféricos (teclado, mouse, pantallas táctiles).

5. ¿Qué es la Gestión de Configuración? Definir línea base. Ejemplifique

Página 7 y 139

- Gestión de Configuración del Software (GCS): Es el conjunto de actividades para identificar, controlar y rastrear los cambios en el software a lo largo de su vida. Asegura que sepamos qué versión estamos usando y qué cambios se hicieron.
- Línea Base (Baseline): Es un punto de referencia aprobado formalmente. Una vez que se establece una línea base (por ejemplo, "Versión 1.0 de Requisitos"), cualquier cambio futuro debe pasar por un proceso formal de aprobación. Es como poner un "seguro" a lo que ya está hecho para poder seguir avanzando sobre terreno firme.
- Ejemplo: Imaginate que terminás de escribir el "Plan del Proyecto" y el cliente lo firma. Ese documento firmado es ahora una Línea Base. Si la semana que viene

querés cambiar una fecha, no podés simplemente borrarla; tenés que hacer una solicitud de cambio formal porque ya estaba aprobado.

6. Rejuvenecimiento.
Página 144 y 145

- Concepto: Es el proceso de mejorar un sistema existente para alargar su vida útil y evitar que se degrade. Es como hacerle un "lifting" al software.
- Tipos de Rejuvenecimiento:

1. Re-Documentación: Crear documentación nueva porque la vieja se perdió o no existe. Ayuda a entender qué hace el código.
2. Re-Estructuración: Ordenar el código internamente (hacerlo más limpio) sin cambiar lo que hace por fuera.
3. Ingeniería Inversa: Analizar el código para intentar recuperar el diseño original (ir de atrás para adelante).
4. Re-Ingeniería: Es lo más completo. Se analiza el sistema viejo, se rediseña y se vuelve a construir una versión mejorada.

7. Calcule el camino crítico.

1)Tipos de planificación organizacional vistos en la materia.
Página: 50.
Respuesta: Existen tres estructuras genéricas de equipo:

- **Democrático Descentralizado (DD):** No hay jefe permanente, decisiones por consenso. Ideal para problemas difíciles.
- **Descentralizado Controlado (DC):** Hay un líder y sublíderes. Resolución de problemas grupal, pero implementación dividida. Ideal para proyectos grandes y complejos.
- **Centralizado Controlado (CC):** Un líder toma todas las decisiones. Ideal para tareas simples y rápidas.

2)Características de todos los riesgos

Página: 51.
Respuesta: Todo riesgo tiene dos características fundamentales:

- **Incertidumbre:** El riesgo puede ocurrir o no (probabilidad < 100%).
- **Pérdida:** Si el riesgo se convierte en realidad, traerá consecuencias negativas (daño o costo).

3)Definir Recuperabilidad (principio de diseño)

Página: 112 (referido en pruebas de recuperación).
Respuesta: Aunque el PDF lo menciona principalmente en la etapa de pruebas ("Prueba de recuperación tras fallos"), como principio de diseño se refiere a la capacidad del sistema

para reponerse de fallos y restablecer el funcionamiento correcto y los datos afectados en un tiempo aceptable.

4)Diferencias entre modelo repositorio y modelo cliente servidor

Páginas: 68 y 70.
Respuesta:

- Repositorio: Centraliza todos los datos en un solo lugar compartido. Los subsistemas interactúan a través de esta base de datos central. Es eficiente para datos grandes pero si falla el centro, falla todo.
- Cliente-Servidor: Distribuye la carga. Hay proveedores de servicios (servidores) y consumidores (clientes). Es más fácil de escalar (agregar más servidores) y distribuir en una red.

5)Qué es GCS. Línea Base, ejemplifique

Páginas: 137 y 139.
Respuesta:

- GCS (Gestión de Configuración del Software): Es el proceso para identificar, controlar y gestionar los cambios en el software a lo largo de su vida.
- Línea Base: Es un punto de referencia aprobado formalmente (una "foto" del proyecto en un momento dado).
- Ejemplo: La "Especificación de Requisitos" firmada por el cliente. A partir de esa firma, cualquier cambio requiere un trámite formal.

6)Describa rejuvenecimiento

Página: 145.
Respuesta: Es el proceso de mejorar un sistema existente para alargar su vida útil y calidad. Incluye técnicas como:

- Re-documentación: Crear manuales nuevos del código.
- Re-estructuración: Ordenar el código internamente.
- Ingeniería Inversa: Analizar el código para recuperar el diseño.
- Re-ingeniería: Rediseñar y reconstruir el sistema.

1) Describa el problema de las “4 p”.

Página: 16.
El problema es que muchas veces los gestores se enfocan demasiado en cuestiones técnicas y descuidan la gestión equilibrada de los cuatro elementos clave: Personal (la gente), Producto (qué se hace), Proceso (cómo se hace) y Proyecto (la planificación). El descuido del personal es la causa más común de fracaso.

2)¿Cuáles son los elementos claves de la gestión de proyectos?
Página: 16.
Respuesta: Son las 4 P mencionadas arriba: Personal, Producto, Proceso y Proyecto.

3)Describa el modelo MOI.
Página: 48.
Respuesta: Es un modelo de liderazgo para gestionar equipos:

- M (Motivación): Incentivar al personal para que rinda al máximo.
- O (Organización): Estructurar el equipo y los procesos para lograr el producto.
- I (Innovación/Ideas): Fomentar la creatividad y nuevas soluciones.

4)¿Qué define el diseño arquitectónico? Describa los tipos de Organización del Sistema.
Páginas: 58 y 67.
Respuesta: Define la relación entre los principales elementos estructurales del software. Los tipos de organización son:

- Repositorio (datos centralizados).
- Cliente-Servidor (servicios distribuidos).
- Máquina Abstracta o Capas (organización jerárquica donde cada capa sirve a la superior).

5)Enumere y describa los principios de Diseño.

Página: 59.
Respuesta:

- Modularidad: Dividir en partes pequeñas e independientes.
- Abstracción: Ocultar detalles complejos.
- Refinamiento: Ir de lo general a lo detallado.
- Cohesión: Que cada módulo haga una sola cosa.
- Acoplamiento: Que los módulos dependan poco entre sí.

6)Defina y describa GCS.

Páginas: 137 y 139.
Respuesta:

- GCS (Gestión de Configuración del Software): Es el proceso para identificar, controlar y gestionar los cambios en el software a lo largo de su vida.
- Línea Base: Es un punto de referencia aprobado formalmente (una "foto" del proyecto en un momento dado).
- Ejemplo: La "Especificación de Requisitos" firmada por el cliente. A partir de esa firma, cualquier cambio requiere un trámite formal.

7)Ejercicio de PERT: Hallar tiempos tempranos - tardíos - caminos críticos y duración total del proyecto .
Páginas: 40-42.
Método: Para hallar los tiempos, debés dibujar la red de tareas.

- Tiempos Tempranos: Se calculan de inicio a fin (sumando duraciones).
- Tiempos Tardíos: Se calculan de fin a inicio (restando duraciones).
- Camino Crítico: Es la ruta donde el Tiempo Temprano es igual al Tiempo Tardío (holgura = 0). Determina la duración total del proyecto.

1. ¿Qué es un proyecto? Describa el problema de las 4 “P”.
Página: 16.
Respuesta: Un proyecto es un esfuerzo planificado y controlado para crear un producto, gestionando recursos y riesgos. (Para las 4P, ver respuesta 1 del Bloque 2).

2. En Planificación temporal, ¿Qué tareas se deben realizar para hacer el seguimiento y control del proyecto?

Páginas: 17 y 19.
Respuesta: Se debe monitorear el cronograma regularmente comparando la fecha real vs. planificada, gestionar el alcance para evitar cambios descontrolados, y analizar riesgos. Se usan métricas de "Estado" y "Uso del tiempo"

3. ¿Cuáles son los tipos (áreas) de Diseño de Software? Describa.

Página: 58.
Respuesta:
- Diseño de Datos: Estructuras de datos y bases de datos.
- Diseño Arquitectónico: Estructura general del sistema.
- Diseño de Interfaz: Comunicación usuario-sistema.
- Diseño de Componentes: Detalle procedimental de cada módulo.

4. ¿Qué es Verificación? ¿Qué es Validación? De todos los tipos de pruebas vistos, ¿Cuál dirías que se está verificando y cual que se está validando al usar la prueba?

Página: 120.
Respuesta:
- Verificación: ¿Estamos construyendo el producto correctamente? (Cumplir especificaciones técnicas).
- Validación: ¿Estamos construyendo el producto correcto? (Cumplir necesidades del cliente).
- En pruebas: Una prueba de Caja Blanca suele verificar (lógica interna), y una de Caja Negra suele validar (requisitos funcionales).

5. Describa el Proceso de GCS.

Página: 143 (Gestión de cambios).
Respuesta: Implica: 1) Identificar los ítems a controlar, 2) Controlar las versiones, 3) Controlar los cambios (solicitud, evaluación, aprobación), 4) Auditorías de configuración, 5) Informes de estado.

6. ¿Qué es el rejuvenecimiento del software? Describa los tipos.

- contestada mas arriba

7. La empresa de pastas HOGAREÑAS S.A. desea hacer un estudio del tiempo que demora en hacer una lasaña, para esto ha identificado las siguientes actividades (en minutos). Encontrar el camino crítico, calculando tiempos tempranos y tiempos tardíos. Calcular el tiempo final.

1. ¿Qué es un riesgo? ¿Cómo se clasifican?

Página: 51.
Respuesta: (Definición ver Bloque 1, punto 2). Se clasifican en:

- Riesgos del Proyecto: Afectan la planificación (costos, plazos).
- Riesgos Técnicos: Afectan la calidad y diseño del software.
- Riesgos del Negocio: Afectan la viabilidad comercial del producto.

2. ¿Cómo puede ser la organización del sistema? (diseño arquitectónico).

contestada mas arriba

3. ¿En qué se diferencian la verificación a la validación? Enumere las pruebas de integración que conozca.

contestada mas arriba
Pruebas de Integración:

- **Descendente (Top-Down):** De arriba hacia abajo (usa stubs).
- **Ascendente (Bottom-Up):** De abajo hacia arriba (usa drivers).
- **Sándwich:** Combina ambas.
- **Big Bang:** Todo junto (no recomendada).

4. ¿Qué tipos de diseño de software conoce?
contestada mas arriba

5. Como es el proceso de GCS.
contestada mas arriba

6. Dada la siguiente tabla cuál o cuáles son el/los camino/s críticos? ¿Cuándo terminará el proyecto? Si llueve durante el periodo de excavación, ¿cuánto tiempo se retrasa el proyecto?

7. Explique el P-CMM (modelo de capacitación y motivación del personal)
Página: 48.

El PDF no utiliza explícitamente las siglas "P-CMM" (People Capability Maturity Model), pero cubre este concepto bajo la Planificación Organizativa y el modelo MOI (Motivación, Organización, Innovación), destacando que el personal es el activo más importante y se debe gestionar su capacidad y motivación para el éxito del proyecto.

1. Enumere y describa las técnicas de estimación que conozca.
Página: 33-34.

- Juicio Experto: Consultar a especialistas.
- Delphi: Expertos estiman anónimamente y discuten hasta consensuar.
- Planning Poker: Estimación ágil con cartas (Fibonacci) en equipo.
- Puntos de Función / LDC: Basadas en fórmulas matemáticas sobre el tamaño del software.

2. Que es la gestión de configuración. Defina línea de base. Ejemplifique.
contestada mas arriba

3. Describa el rejuvenecimiento.
contestada mas arriba

4. Una empresa de software (bla bla bla) lleva un proyecto con requisitos muy estrictos (bla bla) pero no tiene el personal necesario para cierto proyecto. Analice la siguiente situación: Es imposible encontrar el personal con las habilidades adecuadas. a) Analice el riesgo. b) Planifique (Estrategias de tratamientos de riesgos).

Caso de Riesgo (Falta de personal).

- Página: 53-54.
- a) Análisis: Es un Riesgo del Proyecto (recursos) con impacto "Serio" o "Catastrófico" y probabilidad alta (ya dice que es "imposible" encontrar).
- b) Estrategia (Planificación):
  - Mitigación: Capacitar al personal actual (aunque tome tiempo) o contratar consultores externos temporales.
  - Evitación: Reducir el alcance del proyecto para que no requiera esas habilidades específicas tan complejas.

5. Qué consideraciones se deberían tener en cuenta a la hora de desarrollar una interfaz de usuario.

Páginas: 86 y 95.
Respuesta:

- Conocer al usuario (perfil).
- Consistencia (mismos colores y botones).
- Feedback (informar qué pasa).
- Minimizar carga de memoria (no obligar a recordar).
- Prevención de errores.

6. Diferencie Modelo de Repositorio y Modelo Cliente/Servidor.
contestada mas arriba

7. Un diagrama de Pert (Fácil).

1. ¿Qué es la planificación?¿Cuáles son los objetivos de la planificación temporal y organizacional?.

Página: 16-17.
Respuesta: Es el proceso de definir objetivos, recursos y tareas.

- Objetivo Temporal: Controlar el tiempo y cumplir plazos.
- Objetivo Organizacional: Gestionar el equipo humano y roles.

2. ¿Qué es el diseño? Defina acoplamiento y cohesión y sus diferentes grados.

Página: 57, 63, 64.
Respuesta: El diseño es la representación técnica del software antes de construirlo.

- Cohesión: Qué tan relacionadas están las tareas dentro de un módulo. (Se busca Alta). Grados: Funcional (mejor), Secuencial, Comunicacional, Procedimental, Temporal, Lógica, Coincidental (peor).
- Acoplamiento: Qué tanto depende un módulo de otro. (Se busca Bajo).

3. ¿Qué es la GCS?
contestada mas arriba

4. ¿Qué son los riesgos?¿Cómo se clasifican?. Ejemplifique.
contestada mas arriba
Ejemplo: Riesgo Técnico -> "La tecnología de base de datos nueva no soporta la cantidad de usuarios esperada".

5. Defina estimación y los tipos de estimación que conoce.

Página: 33.
Respuesta: Es predecir valores futuros (tiempo, costo, esfuerzo) con datos incompletos.
Tipos: Juicio Experto, Basada en Algoritmos (COCOMO), Basada en Analogías.

6. ¿Qué es el mantenimiento? Defina el rejuvenecimiento.
Página: 141.
Respuesta:

- Mantenimiento: Es la etapa final del ciclo de vida donde se corrigen errores, se adapta o mejora el software después de entregado.
- (Rejuvenecimiento: cpntestada mas arriba)

7. Calcular camino crítico (PERT + CMP)

Nº Afirmación V/F

1. Los elementos de la configuración del software son solo los programas y los datos.
2. Una línea base es un concepto de GCS que nos ayuda a controlar los cambios.
3. El análisis del riesgo no implica dejar de lado la gestión de riesgo, ya que podría llevar a obtener una deuda técnica.
4. El concepto de evaluación de un riesgo no implica que siguiendo esta estrategia, la probabilidad de que el riesgo se materialice sea cero.
5. Una tarea en la planificación temporal se describe por 3 elementos: precursor, duración y fecha de entrega.
6. La estrategia de desarrollo en cascada es un proceso secuencial que se hace en etapas.
7. La investigación de usuario es una de las fases del desarrollo de la experiencia de usuario (UX).
8. La mantenibilidad y el rendimiento impactan en el diseño de la arquitectura de software.
9. Las métricas GQM son base de datos que se usan en el ejemplo de diseño de la arquitectura de software en capas.
10. Las métricas estándar se relacionan con las características de calidad del software.
11. No es recomendable usar una métrica que “mida” cierto objetivo.
12. Si se realiza una prueba y el software ha fallado significa que no hace lo que especifican los requerimientos.
13. Un defecto por omisión en el software resulta cuando algún aspecto del código es incorrecto.
14. La prueba de caja negra también denominada prueba de comportamiento, se centra en los requisitos funcionales del software analizando las entradas y salidas.
15. La complejidad del software es una métrica que proporciona una medición cuantitativa del comportamiento de la calidad del software.
16. Las técnicas de estrategias de pruebas son parte de la Verificación y Validación incluidas en el aseguramiento de la calidad del software.
17. Las pruebas de regresión de software solo se pueden hacer en forma manual.