- Cada pregunta **múltiple opción** contestada correctamente tiene un valor de 3,34 puntos. Cada pregunta incorrecta de la múltiple opción resta 1,11 puntos.
- **El puntaje total del examen es 100 puntos y se aprueba con 60 o más puntos.**

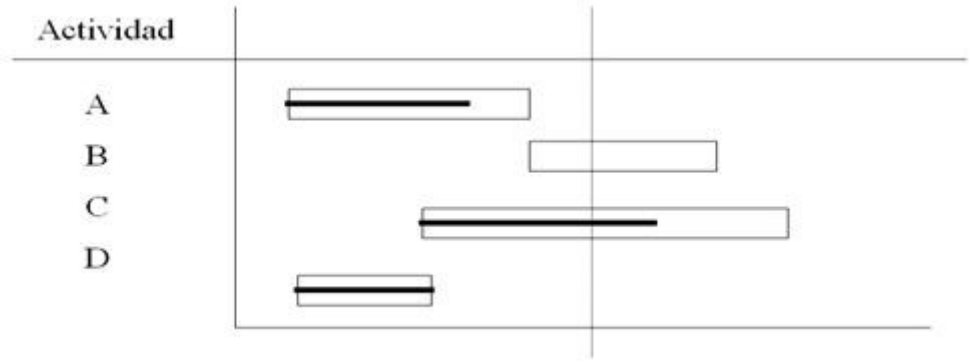
Múltiple Opción

1. La normas de calidad 25000 son de suma utilidad para:

   a) Evaluar los procesos de desarrollo de software
   b) Especificar los requisitos de calidad de un producto de software
   c) b) y sus métricas y su evaluación
   d) Todas las anteriores

---

2. Dado el siguiente diagrama de Gantt



y sabiendo que las actividades D y C están en el camino crítico, se puede saber que el proyecto está:
   a) atrasado.
   b) adelantado
   c) en tiempo
   d) es imposible saberlo

---

3. Sobre el diseño de un sistema:

   a) El objetivo es encontrar el diseño correcto
   b) (a) y el diseño debe ser verificable.
   c) Se debe tener en cuenta el principio de "abstracción".
   d) Ninguna de las anteriores

---

4. Sobre métricas de software:

   a) Uno de los objetivo sería utilizarlas para evaluar la calidad de un software
   b) a) y detectar áreas de mejora para poder tomar decisiones al respecto
   c) a) y siempre se toman medidas directas y objetivas sobre los productos de software
   d) Existe una relación inequívoca entre las métricas estáticas y la calidad del producto

---

5. La estructura de un área de proceso del modelo CMMI presenta:

   a) Metas específicas y prácticas asociadas
   b) Metas genéricas y prácticas asociadas
   c) a) y b) y tanto las metas como las prácticas asociadas son requeridas o sea obligatorias
   d) a) y b) y las metas son requeridas u obligatorias, mientras que las prácticas asociadas son esperadas o deseadas

6. Para definir la arquitectura de un sistema:

a) Basta con conocer todos los requisitos funcionales del sistema
b) Es sumamente relevante conocer los requisitos no funcionales, ya que estos pautan aspectos relevantes de la arquitectura.
c) Es independiente de los requisitos dado que todo sistema se puede llevar a un modelo en capas y a partir del mismo definir las funcionalidades
d) Ninguna de las anteriores.

7. Con respecto a los requisitos

a) Es necesario gestionar los cambios de los requisitos a partir de que son relevados.
b) Los requisitos de software son importantes unicamente al inicio del desarrollo de un sistema.
c) Entre las técnicas de obtención de requisitos se tienen: Casos de Uso, Tormenta de ideas, Revisiones y Encuestas.
d) Un requisito es verificable si se puede demostrar formalmente.

8. Respecto a los procesos de software

a) Los procesos de software comprenden: actividades, roles, responsabilidades
b) a) y la organización y disciplina de esas actividades
c) b) y los procesos tienen como objetivo mejorar la calidad del software construido
d) c) y la mejora de la calidad puede impactar en una mejora de los costos en los que se incurre al construirlo.

9. Los requisitos no funcionales:

a) Describen propiedades que el sistema debe tener o restricciones que debe respetar
b) (a) y se pueden dividir en tres tipos: requisitos del producto, requisitos de la organización y requisitos externos
c) (b) y un ejemplo de requisito externo es el rendimiento.
d) (c) y en lo posible se deberán escribir de manera cuantitativa para que puedan ponerse objetivamente a prueba.

10. Sobre los prototipos

a) Los mock-ups son prototipos verticales, esto significa que implementan un porción funcional de la aplicación.
b) (a) y permiten resolver incertidumbres sobre la factibilidad de la arquitectura propuesta u otros riesgos técnicos.
c) Los mocks-ups son prototipos horizontales, esto significa que se enfocan en porciones de la interfaz de usuario
d) (c) y permiten explorar comportamientos específicos del producto.

11. Sobre el modelo del proceso:

a) En el desarrollo incremental el costo de implementar cambios en los requisitos por lo general se reduce en relación a los mismos cambios bajo un modelo en cascada.
b) En el desarrollo con prototipos , estos siempre son desechados luego que se usan para demostrar conceptos.
c) En el desarrollo en fases con liberaciones parciales, realizar la liberación luego de cada fase no es obligatorio.
d) En el desarrollo utilizando SCRUM no se realizan entregas parciales del producto final.

12. Indique la opción más adecuada.

a) La priorización de requisitos es muy importante. Permite entre otras cosas entender mejor las necesidades de los clientes y la importancia de cada requisito.
b) Para priorizar los requisitos se debe tener en cuenta las necesidades de los clientes, las relaciones de precedencias de los requisitos y el costo de satisfacer cada requisito.
c) Es posible que el proceso de priorización termine con todos los requisitos con la misma prioridad. Si se pueden desarrollar todos con las restricciones existentes entonces el proceso de priorización está bien.
d) Dos de las anteriores son correctas.

13. Respecto a la liberación y mantenimiento:

a) La estrategia de liberación paulatina resulta siempre más adecuada que el big-bang
b) Siempre que se suplanta un sistema por otro es necesario realizar una conversión de datos
c) Muchas veces la liberación de una nueva versión de una aplicación (que incluye por ejemplo un cambio en una funcionalidad) requiere de una liberación paulatina
d) Ninguna es correcta.

---

14. Dadas las siguientes afirmaciones sobre Arquitectura de Software

I. Las propiedades de un sistema, como seguridad, performance y disponibilidad, están influidas por la arquitectura de software utilizada.
II. El diseño arquitectónico es un proceso creativo el cual varía según el sistema que se va a desarrollar para cubrir los requerimientos funcionales y no funcionales.
III. Capas Jerárquicas y Descomposición Orientada a Objetos son ejemplos de patrones de arquitecturas.
IV. Un cliente fino, en una arquitectura cliente-servidor, es responsable de ejecutar el software de presentación pero no del procesamiento de la información ni de la gestión de los datos.

Seleccione la opción correcta:
a) Solo las afirmaciones I), II) y III) son correctas.
b) Solo la afirmación IV) es correcta.
c) Solo las afirmaciones I) y III) son correctas.
d) Todas las afirmaciones son correctas.

---

15. Indique la opción más adecuada.

a) Un caso de uso describe una secuencia de interacciones entre un sistema y un actor externo que resultan en un resultado de valor para el actor.
b) (a) y son dependientes del método de diseño y del lenguaje que se utilice para la implementación.
c) (a) y los casos de uso no tienen porque desarrollarse por completo de una vez.
d) Todas las anteriores son correctas.

---

16. La configuración del software:
a) Comprende los elementos que componen toda la información generada durante del proceso de ingeniería de software.
b) (a) y un cambio sobre uno de estos elementos debe ser aceptado por el encargado de la gestión de la configuración con el fin de controlar los cambios .
c) Si un ítem de configuración se encuentra dentro de la línea base podremos decir que el mismo es estable y que ya no debe ser modificado.
d) Todas las anteriores.

---

17. Indique la opción más adecuada.

a) Las gestión de la configuración implica: gestión del cambio, gestión de versiones, armado del sistema y gestión de las inversiones
b) (a) y su propósito es evitar perder tiempo en modificar versiones incorrectas del sistema, entregar la versión incorrecta a los clientes o perder el código fuente de una versión del sistema o de un componente.
c) Los beneficios y costos de la gestión de la configuración están relacionados al alcance y formalismo. El alcance corresponde al número de funcionalidades que se van a construir del sistema.
d) Ninguna de las anteriores es correcta.

---

18. Dadas las siguientes afirmaciones respecto a las cualidades de software:

a) La correctitud es una propiedad matemática que establece la equivalencia entre el comportamiento del software y los casos de prueba.
b) La productividad refiere al proceso de producción de software y esta relacionada con la eficiencia del mismo.
c) Normalmente la confiabilidad se define en términos del comportamiento estadístico: la probabilidad de que el software opere como es esperado en un intervalo de tiempo especificado, por ende la confiabilidad es una cualidad absoluta.
d) b) y un programa es robusto si se comporta en forma razonable aún en circunstancias que no fueron anticipadas

en la especificación de requerimientos.

19. En un equipo jerárquico la responsabilidad de asignar y coordinar las tareas de los integrantes de cada equipo de desarrollo corresponde:

a) únicamente a los jefes de cada equipo.
b) exclusivamente al director del proyecto.
c) a los miembros del equipo por igual.
d) al gerente funcional de cada área.

20. Construir un prototipo:

a) consiste en construir una parte o un aspecto del producto para, por ejemplo, evaluar la factibilidad
b) (a) y disminuir riesgos relacionados a la implementación de os requerimientos
c) (b) exige un esfuerzo adicional que sólo se justifica si el prototipo puede evolucionar al producto final
d) consiste en una entrega funcional al cliente para experimentar su utilización

21. Dadas las siguientes afirmaciones respecto a estilos arquitectónicos, indique la opción correcta:

a) Si el estilo arquitectónico seleccionado para un sistema, es el de cliente servidor, se puede afirmar que los requerimientos no funcionales se cumplen.
b) En general, luego de que un sistema es liberado, es menos costoso modificar la arquitectura.
c) En el estilo arquitectónico cliente-servidor, sólo es necesario que los clientes conozcan a los servidores, los clientes no tienen porque conocerse entre si ni los servidores conocer a los clientes.
d) En el estilo tubos y filtros, los filtros se conocen entre sí, esto facilita la reutilización de los filtros.

22. A la hora de estimar costos de un proyecto, es posible aplicar la técnica de análisis de reservas y disponer reservas o provisiones para contingencias que son:

a) colchones para las actividades del camino crítico
b) la parte del presupuesto destinada a cubrir las respuestas de mitigación o contingencia planificadas para los «conocidos-desconocidos» susceptibles de afectar al proyecto.
c) cantidades específicas del presupuesto que se reservan para cubrir trabajo no previsto dentro del alcance del proyecto ni identificado como riesgo que pueda surgir (las variables «desconocidas-desconocidas»).
d) Cantidades que no se incluyen en la línea base de costos.

23. Acerca del mantenimiento de software puede decirse que:
a) los defectos más costosos de corregir durante el mantenimiento son los errores introducidos al codificar.
b) según el libro del curso existen tres tipos de mantenimiento de software: Corrección de defectos (donde los errores de requerimientos son los más difíciles de corregir), Adaptación al ambiente (puede cambiar el hardware o la plataforma), Nuevas funcionalidades o modificación de funcionalidades existentes (aparecen necesidades de la organización que deben considerarse en el software)
c) el costo del mantenimiento de software es siempre considerablemente menor que el costo de desarrollo.
d) debido a que el equipo que desarrolló el software siempre es el mismo que lo mantiene, es poco costoso agregar funcionalidades luego de que el software está operativo.

24. Un sistema legado (sistemas “legacy”)
a) Es un sistema construido hace varios años, normalmente con tecnologías hoy obsoletas
b) Es un sistema ya obsoleto
c) Es cualquier sistema de software no construido por el desarrollador que lo mantiene
d) Es un sistema que al dia de hoy carece de importancia para la organización donde se encuentra funcionando

25. Respecto a las estrategias de integración:

a) Big bang tiene como ventajas que se pueden paralelizar la prueba de varios módulos y que no son necesarios drivers ni stubs.
b) Bottom-Up establece que para verificar un módulo todos los módulos que son invocados por el módulo en cuestión deben estar verificados, esta estrategia requiere de drivers pero no de stubs.
c) b) y la estrategia Top-Down permite hacer demostraciones tempranas del producto.
d) c) y la estrategia sandwich tiene como objetivo probar primero los módulos con menor cantidad de líneas de código.

---

26. El criterio recomendado para detener las pruebas de verificación de software es:
a) Cuando todos los casos de prueba ejecutan sin error
b) Cuando el tiempo destinado a realizar las mismas se acabó
c) Cuando el cliente nos lo solicite
d) Ninguna de las anteriores

---

27. Durante las pruebas de integración se busca:
a) Encontrar defectos en las interfaces entre los módulos integrados
b) a) y dependiendo de la estrategia de integración escogida será necesario implementar drivers y/o stubs
c) La validación del cliente
d) Probar los casos de uso principales

---

28. En la ecuación base para los modelos algorítmicos para la estimación de esfuerzo $E = (a + b\ S^c)\ m(X)$, ¿qué variable refleja la deseconomía de escala ?
a) a
b) b
c) c
d) X

---

29. Un plan de verificación y validación:
a) Deberá describir el alcance que tendrá el proceso de pruebas respecto del sistema a ser construido, listando los requisitos funcionales y no funcionales que estarán dentro (y fuera) del plan.
b) (a) y a su vez deberá describir los niveles y estrategias de prueba que se van a realizar en el sistema.
c) (b) y un detalle muy importante del plan es describir los recursos necesarios para llevar adelante el plan de pruebas, los cuales incluyen: recursos humanos, infraestructura, software, hardware y herramientas.
d) b) y a la hora de realizar la planificación de cada prueba se deberá tener en cuenta la disponibilidad del cliente ya que deberá verificar los documentos más relevantes, como es el caso del de requisitos.

---

30. Las técnicas de verificación dinámicas:
I) Sirven tanto para verificar como para validar
II) Son las mas efectivas para lograr detectar defectos de forma temprana
III) Sirven para probar otras características además de funcionalidad
IV) Sirven para verificar tanto software como documentos (requerimientos, diseño, etc.)

a) Todas las afirmaciones son verdaderas
b) I y IV son verdaderas
c) I, II, y III son verdaderas
d) I y III son verdaderas
