Instrucciones

- Cada pregunta múltiple opción contestada correctamente tiene un valor de 3,34 puntos.
- Cada pregunta incorrecta de la múltiple opción resta 1,11 puntos.
- Ante dos opciones correctas en una pregunta, se debe seleccionar la opción más completa.
- La evaluación es de carácter individual y la duración es de 2 horas.
- **El puntaje total del examen es 100 puntos y se aprueba con 60 o más puntos.**

Múltiple Opción

1. ¿Cuál de las siguientes afirmaciones es correcta sobre la Ingeniería de Software?
   a) Se deben considerar restricciones financieras y organizacionales para proveer soluciones.
   b) (a) e implica asumir compromisos que tal vez no provean la mejor solución desde el punto de vista técnico.
   c) (b) y en estas circunstancias se acepta no adoptar un enfoque sistemático ni organizado.
   d) (b) y los métodos a aplicar son independientes de las circunstancias del proyecto.

La opción c) es incorrecta porque un proceso de ingeniería debería ser sistemático y organizado, independientemente de las circunstancias del proyecto. Además, la opción d) es incorrecta porque existen métodos más o menos apropiados según las circunstancias, por ejemplo, optando por un proceso más creativo y flexible a los cambios cuando se requiere una mezcla de habilidades de desarrollo y diseño gráfico.

2. Respecto a los modelos de proceso:
   a) Cuando los requisitos no son claros, no es una buena oportunidad para aplicar metodologías ágiles.
   b) Si el software a construir es crítico, se recomienda tener un mayor grado de formalismo en el proceso a seguir.
   c) Al trabajar con una metodología ágil se pone énfasis en un proceso de desarrollo que promueva las pruebas de integración por sobre la documentación.
   d) El Manifiesto por el Desarrollo Ágil de Software destaca el contar con resultados de valor por sobre las personas y sus interacciones.

Precisamente la falta de claridad en los requisitos es una recomendación para aplicar metodologías ágiles, puesto que estos se van a ir dilucidando en el correr del proyecto y no hay que hacer un compromiso inicial con el alcance ni un plan de gestión del proyecto basado en ellos (opción *a)* incorrecta).
Si el software es crítico, un mayor grado de formalismo permite p. ej. verificar propiedades en los requisitos (opción *b)* correcta).
Una metodología ágil no implica poner énfasis en las pruebas, ni que no se haga documentación. Las dos cosas no tienen relación (opción *c)* incorrecta).
El Manifiesto por el Desarrollo Ágil de Software destaca contar con ambas cosas.

3. Acerca de las metodologías ágiles puede decirse que:
   I. Es conveniente que el equipo sea autogestionado y con buena interacción entre sus integrantes
   II. No es tan necesario que cliente se involucre.
   III. Solo se puede aplicar si los requisitos cambian todo el tiempo.
   IV. No se realiza ninguna documentación, ya que la documentación es el código.
   V. No es necesario seguir estándares de codificación, ya que estos son propios de una metodología basada en planes.

   Seleccione la opción correcta:
   a) Solo la opción II es correcta
   b) Solo las opciones I, III y V son correctas
   c) Solo las opciones IV y V son correctas
   d) Solo la opción I es correcta

En las metodologías ágiles es imperativo contar con el involucramiento del cliente (II); se puede aplicar la metodología incluso si los requisitos son estables (III); por más que el énfasis no esté en la documentación, eso no implica que no se realice ninguna (IV); puede ser necesario seguir estándares de codificación, tanto si lo pide el cliente como si lo decide la organización o el equipo; los estándares de codificación no son ni propios ni exclusivos de las metodologías basadas en planes.

4. Con respecto al modelo de proceso MUM:
   I. Es un proceso de software iterativo e incremental guiado por un plan.
   II. Se divide en cuatro fases: Inicial, Elaboración, Construcción y Transición.
   III. Está centrado en la arquitectura.
   IV. En la fase de Construcción, el equipo se dedica exclusivamente a la implementación, ya habiendo culminado previamente (en las fases previas) de definir de forma completa los requisitos y el diseño.

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

En la fase de Construcción también se itera sobre requisitos y diseño, puesto que, como el modelo es iterativo e incremental, se va iterando en las distintas disciplinas sobre los requisitos o casos de uso a desarrollar en cada iteración.

5. Acerca de la documentación en un proceso ágil:
   a) Los procesos ágiles tienen como restricción que no exista documentación en ningún momento.
   b) Es importante documentar si el equipo que va a mantener el sistema es diferente al que lo está desarrollando.
   c) Toda la comunicación del equipo se basa en la documentación.
   d) No es necesario construir documentos para el cliente.

En cualquier proceso puede ser necesario documentar para dar soporte a la comunicación entre el equipo y el cliente o para proveer información para el mantenimiento posterior. En un proceso ágil, si bien se prefiere no basarse en documentación, no está excluida. En ese sentido, la comunicación dentro del equipo es mayoritariamente cara a cara.

6. En los proyectos de software el cambio es inevitable. Indique las posibles estrategias para abordar el cambio.
   a) No tomar decisiones de diseño hasta el final de un proyecto.
   b) Disponer de un comité de seguridad.
   c) Estimar todas las tareas de un proyecto en la etapa inicial y no permitir ninguna desviación.
   d) Por un lado, anticiparse al cambio. Por ejemplo, mediante prototipos. Por otro lado, generar tolerancia al cambio. Por ejemplo, mediante una arquitectura y diseño adecuado.

Una forma de hacer frente al cambio es disminuir el costo del retrabajo. Esto se puede lograr anticipando el cambio o siendo tolerante a él.

7. Respecto a las técnicas de obtención de requisitos. Seleccione la opción INCORRECTA:
   a) La técnica de Observación consume mucho tiempo, por lo que no son adecuados para todos los usuarios ni para todas las tareas.
   b) Las entrevistas son buenas para conseguir una comprensión global de qué quieren los stakeholders y cómo podrían interactuar con el sistema.
   c) La técnica de tormenta de ideas ayuda a la participación de los stakeholders. No está permitido criticar ni debatir.
   d) Los Focus groups (grupo focal) son reuniones estructuradas en las cuales un selecto grupo de interesados y expertos trabajan en conjunto para definir, crear, refinar y acordar documentos y modelos que representen los requisitos de usuario.

d) Esa es la definición de Workshops Los Focus Groups son grupos de usuarios que participan en una actividad de obtención de requisitos para generar contribuciones e ideas sobre los requisitos funcionales y de calidad de un producto.

8. Respecto a las dificultades que se pueden presentar en el proceso de ingeniería de requisitos. Seleccione la opción correcta:
   a) Diferentes stakeholders pueden tener conflictos con sus requisitos.
   b) Los requisitos cambian (o su prioridad) durante el proceso de análisis. Pueden surgir nuevos stakeholders y el entorno empresarial puede cambiar.
   c) Factores organizacionales y políticos pueden influir en los requisitos del sistema.
   d) Todas las opciones son correctas.

9. Respecto a los requisitos no funcionales. Seleccione la opción INCORRECTA:

a) Afectan fuertemente las propiedades y restricciones del sistema.
b) Especifican comportamientos del producto, por ejemplo requisitos de interoperabilidad o requisitos legislativos.
c) Son afectados por (por ejemplo) políticas de la organización en relación la obligatoriedad de utilización de cierto tipo de tecnologías para la construcción del producto.
d) Los requisitos no funcionales pueden ser más críticos que los requisitos funcionales. Si no se cumplen el sistema puede resultar inútil.

b) Los ejemplos presentados no son correctos (son ejemplos de requisitos externos), ejemplos de comportamiento del producto son: velocidad de ejecución, confiabilidad

10. En relación a los requisitos**:**

Si tengo el caso de uso (CU) Realizar Videollamada, considerando que para hacer una videollamada tengo que estar autenticado en la aplicación:

I. El CU Autenticarme en la aplicación podría modelarse como una inclusión del CU Realizar Videollamada.
II. El CU Autenticarme en la aplicación es una extensión de CU Realizar Videollamada.
III. El CU Compartir Pantalla podría modelarse como una extensión del CU Realizar Videollamada.
IV. El CU Realizar Videollamada lo inicia la aplicación.
V. El CU Registrar Usuario podría modelarse como una extensión del CU Autenticarme en la aplicación.

Seleccione la opción correcta:

a) Solo hay 2 afirmaciones correctas
b) Solo hay 3 afirmaciones correctas
c) Solo hay 4 afirmaciones correctas
d) Las 5 afirmaciones son correctas

Afirmaciones I), III) y V) son correctas

II) Dado que es necesario que se cumpla, la autenticación debería modelarse como una precondición o inclusión. IV) La aplicación nunca inicia el CU x def.

11. Respecto a Historias de Usuario, de las opciones de enunciado de HUs, seleccione la opción correcta:
a) COMO usuario QUIERO visualizar la lista de visitas PARA poder gestionar la llegada de pacientes y de salas de espera.
b) COMO enfermera QUIERO visualizar la lista de visitas PARA poder gestionar la llegada de pacientes y de salas de espera.
c) COMO usuario QUIERO ser capaz de mostrar la lista de visitas.
d) COMO usuario QUIERO ser capaz de mostrar la lista de visitas PARA trabajar mejor.

a) dice “usuario” (no se especifica un rol), c) dice “usuario” y le falta el PARA, d) el PARA no aporta valor al negocio.

12. Respecto al diseño del sistema. Seleccione la opción INCORRECTA:
a) Consiste en dos actividades: diseño arquitectónico y diseño detallado.
b) El diseño arquitectónico especifica cada componente con detalles suficientes para facilitar su construcción.
c) Un sistema se considera desacoplado si consiste de componentes que se pueden implementar separadamente y el cambio en un componente tiene mínimos impactos en otros componentes.
d) Cuanto mayor cohesión más fácil de comprender y modificar un módulo.

b) La definición es de diseño detallado. Diseño arquitectónico de software (a veces llamado diseño de alto nivel): desarrolla la estructura y la organización de alto nivel del software e identifica los diversos componentes.

13. Cuando se está construyendo la arquitectura de un sistema se deben tomar decisiones para las que se tienen en cuenta algunos de los siguientes puntos:
a) Seguridad, resiliencia, acoplamiento y apertura.
b) Seguridad, performance, disponibilidad y mantenibilidad.
c) Independencias, abstracción, singularidad.
d) Dependencias, camino crítico y performance.

Estos son algunos de los puntos relevantes cuando se construye la arquitectura. El resto de las opciones contienen términos que no aplican directamente o son incorrectos.

14. En el diseño de software son posibles **beneficios** del Reuso los siguientes:
    a) Reducción de riesgos en el proyecto, aceleración del desarrollo y mayor confianza en el código construido.
    b) Prescindir de personas especializadas a la vez que se debe probar durante mayor tiempo el software a reusar.
    c) Disminuir los costos de mantenimiento ya que el desarrollo de software reutilizable no requiere mayor esfuerzo.
    d) Evitar las pruebas de integración porque el componente a reutilizar ya ha sido probado.

Para el reuso sí son importante las personas especializadas. El software ha sido ya probado y testeado en producción. El costo de mantenimiento aumenta, por lo que no es un beneficio. Desarrollar software reutilizable es costoso. No es necesario reescribir componentes pero si sucede no es un beneficio.

15. Respecto al patrón arquitectónico Componentes Distribuidos. Seleccione la opción INCORRECTA:
    a) Es una arquitectura abierta que permite agregar nuevos recursos según sea necesario.
    b) El sistema es flexible y escalable.
    c) Cada entidad distribuible puede ser un cliente o un servidor.
    d) Los sistemas de componentes distribuidos dependen del middleware, que entre otras cosas, gestiona las interacciones de componentes.

c) No hay distinción entre clientes y servidores

16. ¿Cuál de las siguientes afirmaciones es correcta sobre consideraciones prácticas de la construcción de software?
    a) La construcción no está exenta de actividades de diseño idénticas a las de las etapas de diseño.
    b) Involucra dos formas de testing: unitario y de integración, que no los realiza quien escribe el código.
    c) La construcción podría involucrar solo el uso de un lenguaje de configuración para realizar una instalación personalizada.
    d) La integración de los componentes construidos se realiza una vez que todos los componentes a liberar están listos.

Salvo la opción correcta, el resto de las opciones son incorrectas porque: si bien hay actividades similares, están enfocadas en el diseño a pequeña escala y no, por ejemplo, en el diseño arquitectural (opción a); por el contrario, el testing unitario y de integración son realizados por quien escribe el código (opción b); la integración podría ser también incremental y no por fases como fue descrito (opción d).

17. Respecto a los criterios de entrada y de salida de los casos de prueba:
    a) Para los criterios de entrada se debe tener en cuenta que el entorno de pruebas esté listo.
    b) (a) y que el software a ser probado esté instalado en el entorno de pruebas.
    c) Un criterio de salida típico podría ser alcanzar un cierto grado de cubrimiento del código.
    d) Todas las opciones son correctas.

Véase la sección 6.2.4 Test Entry and Exit Criteria del libro para más información.

18. Seleccione la opción INCORRECTA, referida al esfuerzo dedicado a las pruebas:
    a) Es preciso que las pruebas sean exhaustivas e incluyan todos los posibles valores de entradas y sus combinaciones con diferentes precondiciones.
    b) El esfuerzo dedicado a las pruebas debe ser controlado y tomar en cuenta el riesgo y las prioridades.
    c) Los casos de prueba deberían incluir una muestra razonable del uso que se le dará al software.
    d) Para encontrar más defectos se deben agregar casos de prueba que prueben partes del software que aún no han sido probadas o combinaciones de valores de entrada que no han sido previamente usadas.

Véanse las secciones 2.4 General Principles of Testing y 2.1.3 Software Quality para más información.

19. Las pruebas de software dinámicas pueden realizarse con diferentes propósitos. Seleccione la opción INCORRECTA:
    a) Para mostrar el correcto cumplimiento de los requisitos
    b) Para provocar fallas
    c) Para medir la calidad del producto
    d) Para detectar malas prácticas de programación

Para detectar malas prácticas de programación se pueden realizar análisis estáticos del producto, como, por ejemplo, revisiones, pero estas no son pruebas dinámicas, sino estáticas (opción d). Las opciones a), b) y c) son propósitos de las pruebas de software en general y se pueden realizar mediante pruebas dinámicas. Véase la sección 2.1.2 Testing Terms para más información.

20. Dentro de la etapa de análisis y diseño del proceso de pruebas,
    a) se deben crear casos de prueba para examinar el comportamiento especificado del objeto de la prueba.
    b) no es necesario incluir casos de prueba de error, porque es difícil crear las precondiciones necesarias para ejecutarlos.
    c) no es posible crear casos de prueba para entradas o condiciones inesperadas, porque no se ha especificado cuáles serían estas situaciones ni qué manejo se debe hacer de ellas.
    d) no se puede preparar la infraestructura y el ambiente para correr el programa que se va a probar, porque aún no se sabe dónde y cómo se ejecutarán esas pruebas.

La actividad más propia de esta etapa es el diseño de los casos de prueba para verificar el comportamiento del producto de software (opción *a)* correcta). Sí es necesario incluir casos de prueba de error, por más que sea difícil crear las precondiciones necesarias para ejecutarlos (opción *b*) incorrecta). Precisamente, para las entradas o condiciones inesperada (que, por inesperadas no se han especificado) es necesario crear un manejo de excepciones genérico que las capture (opción *c)* incorrecta). En esta etapa es cuando se debe definir y preparar la infraestructura y el ambiente para correr las pruebas (opción *d)* incorrecta).

21. A la hora de definir qué pruebas hacer:
    a) Si la interacción entre componentes es importante en el producto, las pruebas de integración pueden sustituir por entero a las pruebas unitarias.
    b) Lo razonable es alcanzar el mayor cubrimiento posible.
    c) No tiene sentido probar partes del código con posibles entradas con las que ese código nunca se va a invocar, porque nunca se va a producir una falla en el contexto de la aplicación.
    d) Tiene sentido probar partes del código con posibles entradas con las que ese código nunca se va a invocar en la aplicación, para no dejar deuda técnica que se exprese en una falla si luego el código se modifica.

Si bien las pruebas de integración pueden encontrar defectos también a nivel unitario, las pruebas que verifican la interacción no son suficientes como para sustituir a todas las pruebas unitarias. Habrá pedazos de código que necesiten una prueba unitaria (opción *a)* incorrecta). Lo razonable es hacer un testeo razonable según el tipo de aplicación de que se trate, y sus prioridades y riegos. No siempre procurar el mayor cubrimiento posible es razonable (opción *b)* incorrecta). Ciertamente uno podría obviar probar ciertos caminos que no se recorrerán de acuerdo a cómo se realizan las invocaciones en el producto de software desarrollado, pero esto podría dejar sin detectar algún defecto del módulo y generar deuda técnica, y este defecto podría expresarse si se realizan modificaciones o se quiere reutilizar el código de ese módulo (opciones *c)* y d)).

22. El mantenimiento de software puede entenderse como la modificación de un sistema luego de que este comienza a ser utilizado. Entre los tipos de mantenimiento puede encontrarse:
    a) El mantenimiento que busca reparar fallas.
    b) El mantenimiento que tiene como objetivo adaptar el software a cambios normativos o regulatorios.
    c) El mantenimiento para agregar o modificar funcionalidades al sistema.
    d) Todas las opciones son correctas.

Lo que se expresa en la pregunta son las distintas definiciones de tipo de mantenimiento vistos en el curso.

23. Sobre las estrategias de adopción de software:
    a) La estrategia de "Big-bang", así como la "paulatina" generan niveles de riesgo similares debido a que su nivel de segmentación (por módulos, unidades, etc.) son similares.
    b) En una estrategia "paulatina" la segmentación por módulos permite obtener un software con mejores características de manteniblidad.
    c) La estrategia de "procesamiento en paralelo" permite planificar escenarios de entrenamiento y validación en operación
    d) Siempre es preferible utilizar estrategias híbridas ya que se podrán seleccionar las mejoras prácticas de estrategias individuales.

a) Las estrategias “Big-Bang” y “paulatina” presentan distintos niveles de riesgo, tanto desde una perspectiva de segmentación de la integración así cómo desde su gestión
b) La estrategia de adopción de software no tienen por qué estar ligada a la mantenibilidad del software. Esto ultimo se puede afectar principalemente por aspectos como su diseño o apego a estandares.
d) Si bien la estrategias hibridas pueden presentar ventajas, no es correcto que siempre será así ante cualquier tipo de software.

24. Los cambios urgentes pueden tener que ser implementados sin pasar por todas las etapas del proceso de ingeniería de software:

a) Por ejemplo, si nos encontramos ante una falla grave y de alto impacto, la prioridad es reparar el sistema para retomar la operación normalmente.
b) Ante cambios en reglas de negocio, no debería ser necesario realizar análisis de requisitos ni de impacto para lograr implementar el resultado en menor tiempo.
c) Los cambios urgentes no requieren de actividades de verificación y validación, si el cambio falla se vuelve a generar otro cambio urgente para volver a empezar.
d) Tanto (a) como (b) son correctas.

b) Cambios en las reglas de negocio pueden implicar impactos complejos en el software. Por lo tanto, podría ser necesario, al menos, realizar el análisis de impacto correspondiente para proceder a la implementación.
c) No es correcto prescindir siempre de actividades de V&V ante cambios urgentes, por ejemplo si estos presentan niveles de complejidad considerables.

25. ¿Cuál de las siguientes afirmaciones es correcta sobre la gestión de la configuración?
a) El software debería ser rearmado frecuentemente y probado inmediatamente luego de armar una nueva versión.
b) (a) pero no implica gestionar un sistema de software en evolución.
c) (a) pero no es necesario documentar la liberación dado que no se volverá a recrear en un futuro.
d) El cliente no debería involucrarse en la gestión de los cambios para no condicionar las decisiones.

Salvo la opción correcta, el resto de las opciones son incorrectas porque: implica efectivamente gestionar un software en evolución (opción b); cuando se produce una liberación se debe documentar para que pueda ser recreada en un futuro (opción c); en las metodologías ágiles el cliente se involucra activamente en este aspecto (opción d).

26. Si un elemento del *product backlog* cumple con los criterios INVEST, entonces:
a) el equipo no podrá comenzar a trabajar en él hasta que el *product owner* estime.
b) el equipo puede estimar el esfuerzo que llevará completarlo.
c) puede ser terminado en dos iteraciones como máximo.
d) quedan claramente definidas las dependencias de las historias.

La opción *a)* es incorrecta, porque un el primer criterio INVEST implica que se puede comenzar a trabajar en el elemento inmediatamente. La opción *c)* también es incorrecta, porque uno de los criterios INVEST es *short*, es decir, que el elemento pueda ser desarrollado en una iteración, no en dos. Y la opción *d)* es incorrecta porque el criterio de compleción del elemento es la *definition of done*, que no se requiere para comenzar a trabajar en la historia. Los criterios INVEST se pueden aplicar al *definition of ready*. Por eso, para poder comenzar a trabajar en una historia, se debe poder estimar el esfuerzo (opción *b*).

27. Dado un conflicto técnico sobre el que los integrantes del equipo no logran ponerse de acuerdo, ¿cuál de los siguientes planteos del director del proyecto corresponde a la técnica de consensuar/conciliar?
a) «Este tema está tardando más de lo esperado en resolverse y no han llegado a un acuerdo, vamos a retomarlo en la próxima reunión».
b) «Este tema está tardando más de lo esperado en resolverse y no han llegado a un acuerdo, vamos a ir por la opción presentada por Elisa».
c) «Todos tienen algo de razón, vamos a incorporar un poco de cada propuesta».
d) «Todos estamos de acuerdo en que tenemos un problema y que la causa es un error técnico».

La opción *a)* es un ejemplo de la técnica de retirarse/eludir. La opción *b)* es un ejemplo de la técnica de forzar/dirigir, porque el director impone su punto de vista. La opción *c)* es un ejemplo de la técnica de consensuar/conciliar, porque se busca una solución que aporte un cierto grado de satisfacción a todas las partes. La opción *d)* es un ejemplo de la técnica de suavizar/adaptarse, porque se resaltan los puntos en común, lo que mantiene un ambiente cordial, aunque no resuelve el problema.

28. Una estimación puede entenderse cómo «una proyección de la experiencia del pasado hacia el futuro, ajustando según las diferencias entre el pasado y el futuro».
a) Esta afirmación es falsa, ya que no considera cambios en los equipos de trabajo y esto afecta la estimación.
b) Esta afirmación es verdadera. Como parte de la «experiencia pasada» puede consultarse, por ejemplo, el registro de estimación de proyecto similares ya ejecutados.
c) Esta afirmación es verdadera. Como parte del "ajuste a futuro" podría considerarse la dedicación que cada programador tendrá en el proyecto como parte de la estimación de esfuerzo.
d) Tanto (b) como (c) son correctas.

El enunciado es verdadero; plantea una definición general de estimación. La opción c) es falsa ya que la dedicación de cada programador no afecta la estimación de esfuerzo, sino de duración.

29. La EDT es
    a) una descomposición jerárquica del alcance total del trabajo
    b) (a) que debe realizar el equipo del proyecto para cumplir con los objetivos del proyecto.
    c) una descomposición temporal del alcance total del trabajo
    d) (c), donde el total del trabajo correspondiente a los niveles inferiores debe corresponder al acumulado para los niveles superiores.

La definición correcta es la dada por *a)* y *b)*. La EDT no representa una descomposición temporal del proyecto.

30. Respecto a la gestión de riesgos:
    I) La gestión de riesgos finaliza una vez establecidas las estrategias para cada riesgo.
    II) Si un riesgo tiene alto impacto y baja probabilidad de ocurrencia, debe aceptarse.
    III) Los riesgos positivos, no es necesario gestionarlos.
    IV) Transferir un riesgo es bajar la probabilidad de ocurrencia y el impacto a 0.
    V) El grado de tolerancia a los riesgos es similar en todas las organizaciones.

    a) Solo 3 afirmaciones son correctas.
    b) Solo 2 afirmaciones son correctas.
    c) Solo 1 afirmación es correcta.
    d) Ninguna afirmación es correcta.

I) gestión de riesgos se mantiene durante todo el proyecto: II) no tiene por qué aceptar un riesgo con esas características, depende de la organización y el proyecto; III) los riesgos deben gestionarse; IV) transferir no siempre implica bajar a 0 el impacto, y V) depende de cada organización.