Instrucciones

- Cada pregunta múltiple opción respondida correctamente tiene un valor de 2,445 puntos.
- Cada pregunta múltiple opción respondida incorrectamente resta 0,814 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 una hora y treinta minutos.
- El puntaje total del parcial es de 44 puntos.

1. Seleccione la afirmación correcta:

   a) El incremento, en los últimos años, de la complejidad de los sistemas de software no es un factor fundamental en el éxito o el fracaso de los proyectos de software.
   b) No hay métodos o procesos (generales -que apliquen a cualquier proyecto-) en la ingeniería de software debido a que, entre otras cosas, diferentes tipos de software pueden requerir diferentes enfoques de desarrollo de software.
   c) (b) y la ingeniería de software, como disciplina, incluye en sus tópicos temas éticos vinculados al desarrollo de software.
   d) Los costos y el cronograma de un proyecto de software no son tópicos de la ingeniería de software como disciplina, sino que son tópicos de la economía digital.

2. Seleccione la afirmación correcta:

   a) Los procesos de desarrollo de software actuales (o modernos) normalmente definen roles a llevar adelante durante la ejecución de los proyectos.
   b) a) y por ejemplo, Scrum tiene definido el rol de Scrum Master, entre otros roles.
   c) b) y Scrum propone revisar en cada *Sprint* la forma en la cual el equipo de desarrollo trabajó durante el *Sprint* y reflexionar sobre cómo se podría haber trabajado mejor. Esto se realiza en conjunto con el cliente, la gerencia de nivel medio y quienes firmaron los contratos para realizar el proyecto.
   d) Los procesos ágiles se centran en la arquitectura de software. En las primeras iteraciones se define y en las siguientes la arquitectura es refinada. Esto, por ejemplo, en Scrum se llama refinamiento sucesivo y en XP se llama ajustar la metáfora (arquitectura) por iteración.

3. Una empresa es contratada para desarrollar un cierto producto de software. Luego de trabajar en recolectar al menos una idea general de los requisitos, los líderes del proyecto de desarrollo de software estiman el tamaño del proyecto y el esfuerzo. Dado el tamaño del equipo de desarrollo estiman que el proyecto para la construcción del producto de software pedido llevará alrededor de un año y medio de trabajo.

   Seleccione la afirmación correcta:

   a) Dado que el trabajo va a durar un año y medio, se debería utilizar un proceso tradicional de desarrollo de software en lugar de uno ágil ya que los planes serán de mucho valor.
   b) Dado que el trabajo va a durar un año y medio, se debería utilizar un método ágil, ya que al durar ese tiempo van a haber muchos cambios en los requisitos durante el desarrollo.
   c) Los datos que nos da esta pregunta sobre el proyecto (que se estimó el tamaño y esfuerzo, y que dado el tamaño del equipo de desarrollo va a llevar un año y medio la construcción) son insuficientes para poder indicar qué proceso de desarrollo utilizar.
   d) Con esta realidad planteada, el siguiente paso es desarrollar una arquitectura en capas buscando flexibilidad debido a la extensión prolongada en el tiempo del proyecto (aproximadamente un año y medio).

4. Seleccione la afirmación correcta:

   a) Los procesos iterativos e incrementales logran liberar incrementos (versiones) al cliente al menos una vez por mes.
   b) Los roles, en la especificación de un proceso de desarrollo, normalmente definen las habilidades y las responsabilidades que deben poder llevar adelante las personas de los equipos de desarrollo. Buscan definir quién (con qué rol) hace qué (actividades, tareas, etc.).
   c) b) y el desarrollo incremental propone que cada incremento alcanzado (que surge de cada iteración) sea entregado al cliente para que pueda comenzar a utilizar el incremento construido.
   d) Los métodos ágiles se basan fuertemente en el concepto de velocidad. Este concepto indica que el desarrollo de software no debe ser postergado o suspendido por actividades externas al equipo; es decir, mantener la velocidad siempre. Se busca registrar esta velocidad para luego poder planificar las iteraciones o *Sprints*.

5. Seleccione la afirmación correcta:

   a) El modelo CMMI, por sus características, puede ser utilizado para la mejora de procesos de desarrollo de software (incluso procesos ágiles).
   b) Las actividades de un proceso de desarrollo de software, de una empresa particular, deberían reflejar la mejor forma (que la empresa conoce) para desarrollar sus productos. Por eso, los procesos de las empresas deben seguirse de forma disciplinada y no realizar cambios durante el transcurso un proyecto.
   c) Los prototipos de software sirven mucho para la etapa de puesta en producción (liberación) del software. Esto es porque con los mismos se pueden hacer demostraciones a los usuarios de cómo deben usar el sistema que se está instalando. En etapas tempranas los prototipos no son de utilidad ya que los usuarios tienden a pensar que el producto ya está pronto y debería ser disponibilizado, causando discusiones y riesgos de relacionamiento cliente-usuario-empresa de desarrollo que serán difíciles de resolver.
   d) El desarrollo con *Scrum* debe ser siempre con la práctica “cliente en el lugar”. De otras forma, no se está haciendo Scrum sino una variante de Scrum.

6. En general, el estilo y la estructura de la arquitectura:

   a) deberían depender de los requisitos no funcionales del sistema, como ser rendimiento, seguridad, protección, disponibilidad y mantenibilidad, entre otros.
   b) deberían depender de los requisitos no funcionales del sistema, así como de aspectos de presupuesto, tecnología y tendencias del mercado, siendo este último el más importante de todos.
   c) deberían depender únicamente de los requisitos funcionales del sistema, ya que los requisitos no funcionales no deberían afectar fuertemente la arquitectura.
   d) no deberían depender de los requisitos del sistema (funcionales o no funcionales), ya que cualquier estilo y estructura de arquitectura podría ser adecuada para un sistema.

7. Los principios de diseño son nociones clave que brindan la base para el diseño de software. En este contexto:

   I. Abstracción refiere a "una vista de un objeto que se enfoca en la información relevante para un propósito particular e ignora el resto de la información".
   II. Acoplamiento refiere a "una medida de la fuerza de asociación de los elementos dentro de un módulo".
   III. Cohesión refiere a "una medida de la interdependencia entre módulos en un programa informático".
   IV. Encapsulación y ocultamiento de información refiere a “agrupar y empaquetar los detalles internos de una abstracción y hacer que esos detalles sean inaccesibles para las entidades externas”.

   a) Sólo las afirmaciones II y III son correctas
   b) Sólo las afirmaciones I y IV son correctas
   c) Sólo las afirmaciones I y II son correctas
   d) Sólo la afirmación I es correcta

8. Según Sommerville, el Diseño es:

   a) realizar en forma creativa diagramas de componentes para validar con el cliente.
   b) una actividad creativa donde se identifican los componentes del software y sus relaciones, con base en los requerimientos de un cliente.
   c) analizar los requisitos funcionales y realizar en forma creativa un diagrama de componentes y de ser necesario un diagrama de despliegue.
   d) analizar los requisitos funcionales y no funcionales y decidir cuando es necesario o no realizar en forma creativa un diagrama de despliegue.

9. En el proceso de diseño de la arquitectura:

   a) en un proceso ágil, usualmente resulta exitoso el desarrollo incremental de arquitecturas.
   b) la disponibilidad o seguridad del sistema no son relevantes.
   c) la salida consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación.
   d) no se incluye la actividad de documentar la arquitectura considerando distintas vistas, como puede ser el modelo de vistas 4+1 (Krutchen).

10. Patrones arquitectónicos:

    a) El patrón MVC (modelo vista controlador) es usualmente utilizado cuando existen múltiples formas de ver e interactuar con los datos. El usuario interactúa directamente con el modelo para seleccionar la vista que sea más apropiada.
    b) La arquitectura en capas organiza el sistema en capas con funcionalidades relacionadas a cada capa. Cada capa brinda servicios a la capa “de encima” y puede consumir servicios de cualquier capa que esté “por debajo” (incluso dos o tres niveles inferiores) así como de capas que estén un nivel “por encima”.
    c) En el patrón de repositorio todos los datos del sistema se gestionan en un repositorio central, accesible a todos los componentes del sistema. Los componentes pueden interactuar a través de este repositorio y directamente entre ellos.
    d) El patrón de tubería y filtro se aplica cuando el procesamiento de datos se organiza de forma que cada componente de procesamiento (filtro) sea discreto y realice un tipo de transformación de datos. Se suele utilizar en procesamientos batch.

11. Sobre la construcción de software, seleccione la afirmación correcta:

    a) La planificación de la construcción incluye, entre otras cosas, cómo van a ser construidas e integradas las componentes de software.
    b) (a) y la construcción es una etapa del desarrollo de software que normalmente no se realiza en la práctica profesional (es decir, en la industria de software) ya que se pasa directo a la programación luego del diseño.
    c) (b) y la construcción efectiva y eficiente que se ha logrado mostrar a través de los últimos años y que sigue vigente es mediante el uso del lenguaje de programación Java.
    d) La construcción y la programación son lo mismo. En particular, el SWEBOK define a la construcción de software como la actividad de programar.

12. Sobre la construcción de software según el SWEBOK, seleccione la afirmación correcta:

    a) El reuso es una actividad que comprende utilizar (reutilizar) software ya existente como parte del desarrollo de software que se está realizando.
    b) (a) y la definición de reuso no comprende el construir software que pueda ser luego reutilizado.
    c) El reuso no se da en la práctica industrial. Es un concepto teórico que, lamentablemente, no puede ser realizado en la industria debido a las complejidades intrínsecas del software.
    d) La construcción de software, en procesos de desarrollo de software en cascada, se realiza sin revisiones o inspecciones de código ya que no forman parte de la especificación del proceso en cascada.

13. Indique cuál de las siguientes proposiciones no es un requisito del software:

    a) El producto deberá mostrar un listado con todas las empresas, de las que desplegará nombre y RUT.
    b) La consulta de empresas no podrá demorar más de 10 segundos
    c) El producto deberá ser desarrollado en Java.
    d) Toda empresa que desee registrarse ante el organismo regulador deberá llenar una declaración jurada y presentarla en la oficina de recepción.

14. Respecto a la relación de inclusión entre casos de uso

    a) Se utiliza para describir escenarios alternativos complejos.
    b) Solo se pueden incluir casos de uso en el flujo principal de un caso de uso
    c) Si el caso de uso incluido termina de forma no exitosa, se continúa la ejecución del flujo principal en el punto siguiente del caso base.
    d) El caso de uso incluido no sabe quién lo incluye e incluso puede ser incluido por varios casos de uso, por lo que no puede ir a un punto específico del caso base.

15. Para especificar los requisitos para un sistema de negocio, sería mejor utilizar:

    a) Lenguaje natural, ya que es expresivo, intuitivo y universal, lo que significa que los requisitos pueden ser entendidos por los usuarios y los clientes.
    b) Una especificación estructurada, en la que los requisitos son escritos de manera estándar, para poder darle la estructura y el rigor que la especificación de este tipo de sistemas requiere.
    c) Una especificación en un lenguaje formal para reducir la ambigüedad propia del lenguaje natural, que pueda ser validada y aceptada como contrato del sistema a desarrollar.
    d) Un conjunto de modelos gráficos UML, ya que una imagen vale más que mil palabras y se ahorra, así, tener que agregar descripciones textuales.

16. Según Wiegers, es deseable que un requisito sea:

a) Verificable, es decir, que se establezca el cubrimiento de pruebas deseado para el producto.
b) Factible, es decir, que sea posible implementarlo dada las capacidades y limitaciones del sistema y su ambiente operacional.
c) Necesario, es decir, que fue solicitado por el cliente.
d) No ambiguo, es decir, que una misma persona no pueda tener dos interpretaciones distintas del requisito.

ATENCIÓN: ESTA PREGUNTA NO ESTÁ BIEN FORMULADA.

Una de estas propiedades deseables es que el requisito sea no ambiguo. Esto significa, lisa y llanamente, que no tenga dos interpretaciones distintas, tanto sea de una sola persona o de varias.

Wiegers distingue esos dos tipos de ambigüedad en dos lugares:

Unambiguous
Natural language is prone to two types of ambiguity. One type I can spot myself, when I can think of more than one way to interpret a given requirement. The other type of ambiguity is harder to catch. That's when different people read the requirement and come up with different interpretations of it. The requirement makes sense to each of them but means something different to each of them.

Ambiguous requirements
One symptom of ambiguity in requirements is that a reader can interpret a requirement statement in several ways (Lawrence 1996). Another sign is that multiple readers of a requirement arrive at different understandings of what it means.

El sentido en que pensamos esta pregunta, entonces, siguiendo esta distinción, es que la opción d) define no ambiguo solo en el aspecto de que una misma persona no tenga dos interpretaciones distintas, pero esta definición sería incompleta, porque podría darse el caso de que cada uno de los que lo leyeran lo interpretaran de una única manera, pero estas interpretaciones fueran distintas entre sí, con lo que el requisito sería ambiguo. Por lo tanto, la opción sería incorrecta. No hay error en el concepto. Pero sí se presenta un problema en la formulación, que surge por haber usado el verbo poder. De haber puesto: «d) No ambiguo, es decir, que una misma persona no tenga dos interpretaciones distintas del requisito», la opción sería falsa porque no alcanzaría como definición de no ambigüedad por lo antedicho.
Pero, como plantea Felipe, se puede entender que decir que una persona no puede tener más de una interpretación del requisito implica necesariamente que no pueda haber más de una interpretación distinta, porque, decir que no es posible que una persona tenga más de una interpretación implicaría que solo hay una interpretación posible, y, por ende, nadie (ni uno ni varios) puede interpretarlo de dos maneras.

Por tanto, la formulación da lugar a que se consideren dos opciones como correctas, por lo que se darán los puntos a quienes contestaron tanto b) como d), o a quienes no hayan respondido, en atención a que pueden haber entendido que había dos respuestas correctas.

17. Respecto al nivel de detalle con que deben escribirse las historias de usuario:

    a) Una historia de usuario se escribe en alto nivel y los detalles los obtendrán el equipo de desarrollo y el de verificación de la conversación con el cliente y en ningún caso será necesario registrarlos por escrito.
    b) Una historia debe ser escrita con todos los detalles necesarios para poder ser implementada y verificada adecuadamente.
    c) Una historia se escribe en alto nivel y luego los detalles necesarios para implementarla y probarla se especifican en los criterios de aceptación.
    d) No es posible que los desarrolladores produzcan una estimación razonable del esfuerzo necesario para la implementación de una historia a menos que cuenten con criterios de aceptación detallados de ella.

18. De acuerdo a la realidad planteada en el laboratorio 1, una adecuada selección de técnicas de obtención de requisitos debería contemplar:

    a) Que sean capaces de identificar requisitos de los diferentes usuarios del sistema: funcionarios MTOP, autoridades MTOP, administradores sistema MTOP, responsables de empresas de transporte, choferes y ciudadanos. Los cuales tendrán un fuerte impacto en cómo organizar y modelar esas funcionalidades, teniendo en cuenta la interacción del sistema con cada uno de estos usuarios.
    b) Que sean capaces de identificar requisitos de comunicación con interfaces externas, requisitos no funcionales y restricciones del proyecto. Los cuales van a tener un fuerte impacto en los patrones de arquitectura a elegir para la solución.
    c) Que sean capaces de identificar requisitos de los diferentes usuarios “disponibles” del sistema. Para los ciudadanos resulta imposible relevar sus requisitos porque son millones.
    d) Dos opciones entre la a), la b) y la c) son correctas.