**Preguntas Verdadero O Falso Promocion Ingenieria De Software 1 2024.**

**Los metodos discretos de recopilacion de informacion no son suficientes por si solos y deben completarse con otros metodos. (V o F) Justificar.**

**Verdadero.**

Los métodos discretos de recopilación de información, como encuestas con respuestas cerradas, conteos o mediciones puntuales, pueden proporcionar datos estructurados y cuantificables, pero por sí solos pueden ser insuficientes para capturar la complejidad de un problema.

Para una comprensión más completa, es necesario complementarlos con otros métodos, como técnicas cualitativas (entrevistas abiertas, observación participante, análisis de documentos) o métodos continuos (monitoreo en tiempo real, series temporales). Esto permite obtener tanto precisión en los datos como una mejor interpretación del contexto en el que se generan.

**La elicitacion de requerimientos es una es una actividad principalmente tecnica, con pocos desafios sociales. (V o F) Justificar.**

**Falso.**

La **elicitación de requerimientos** no es solo una actividad técnica, sino que también implica importantes desafíos sociales. Aunque se basa en técnicas estructuradas como entrevistas, encuestas y análisis de documentos, gran parte del proceso depende de la comunicación efectiva entre desarrolladores y usuarios.

Algunos de los desafíos sociales incluyen:

* **Dificultades de comunicación**: Los clientes pueden no expresar claramente lo que necesitan o pueden tener expectativas poco realistas.
* **Diferencias en la terminología**: Los usuarios pueden describir sus necesidades en términos no técnicos, lo que requiere interpretación por parte del ingeniero de software.
* **Conflictos entre partes interesadas**: Diferentes usuarios pueden tener necesidades o prioridades contradictorias.
* **Resistencia al cambio**: Algunas personas pueden no estar dispuestas a modificar sus procesos actuales o aceptar nuevas soluciones.

Por estas razones, la elicitación de requerimientos combina habilidades técnicas con habilidades interpersonales, como la negociación, la empatía y la gestión de expectativas.

**Un requerimiento funcional describe una restriccion especifica sobre como se debe construir el sistema, mientras que los no funcionales detallan que debe hacer el sistema. (V o F) Justificar.**

**Falso.**

La definición está invertida:

* **Los requerimientos funcionales** describen **qué debe hacer el sistema**, es decir, las funciones, características y comportamientos esperados. Ejemplo: "El sistema debe permitir a los usuarios registrar una cuenta con su correo electrónico y contraseña".
* **Los requerimientos no funcionales** especifican **cómo debe construirse el sistema**, estableciendo restricciones y criterios de calidad como rendimiento, seguridad, usabilidad y disponibilidad. Ejemplo: "El sistema debe responder a cada solicitud en menos de 2 segundos".

Ambos tipos de requerimientos son esenciales para garantizar que el software cumpla con las expectativas tanto en funcionalidad como en calidad.

**El modelo de diagramas de casos de uso facilita la participacion activa de los usuarios en la definicon de los requerimientos . (V o F) Justificar.**

**Verdadero.**

El modelo de **diagramas de casos de uso** facilita la participación activa de los usuarios en la definición de los requerimientos porque presenta la funcionalidad del sistema de manera visual y sencilla, sin necesidad de conocimientos técnicos avanzados.

Algunas razones que justifican esto son:

* **Lenguaje comprensible**: Representa las interacciones entre los actores (usuarios o sistemas externos) y el sistema de forma intuitiva.
* **Facilita la comunicación**: Ayuda a que los usuarios y desarrolladores tengan una comprensión común sobre lo que debe hacer el sistema.
* **Permite detectar omisiones o errores**: Al visualizar los casos de uso, los usuarios pueden identificar funcionalidades faltantes o mal definidas.
* **Fomenta la validación temprana**: Los usuarios pueden confirmar si el sistema cubrirá sus necesidades antes de que se avance en el desarrollo.

Por estas razones, los diagramas de casos de uso son una herramienta clave en la **ingeniería de requisitos**, ayudando a mejorar la precisión y claridad de la especificación del sistema.

**Un caso de uso puede ser considerado un requerimiento, auqnue no tenga un escenario asociado .(V o F) Justificar.**

**Falso.**

Un **caso de uso** sin un **escenario asociado** no puede considerarse un requerimiento completo, ya que no proporciona suficiente información sobre el comportamiento esperado del sistema.

**Justificación:**

* **Un caso de uso describe una funcionalidad general**, pero sin un escenario, no especifica los pasos exactos ni las condiciones bajo las cuales ocurre la interacción.
* **Los escenarios detallan las secuencias de eventos** que muestran cómo los actores interactúan con el sistema en diferentes situaciones (casos normales, excepcionales, alternativos).
* **Sin un escenario, el caso de uso es ambiguo**, lo que dificulta su implementación y validación.

Por ejemplo, un caso de uso llamado **"Realizar pago"** sin un escenario no especifica si el pago se hace con tarjeta, transferencia o efectivo, ni qué sucede si la transacción falla.

Por lo tanto, un caso de uso necesita al menos un escenario para ser considerado un requerimiento claro y útil en el desarrollo del software.

**El disparo de una transicion en una red de petri puede generar un numero ilimitado de tokens en los lugares de salida .(V o F) Justificar.**

**Falso.**

En una **Red de Petri**, el disparo de una transición genera un número de **tokens finito y bien definido** en los lugares de salida, determinado por la función de peso de los arcos de salida.

**Justificación:**

* Cada transición tiene **arcos de salida** con pesos específicos que determinan cuántos tokens se agregan a los lugares de destino.
* No existe una regla en la teoría de Redes de Petri que permita la generación de un número **ilimitado** de tokens en un solo disparo.
* Aunque algunas variantes como las **Redes de Petri estocásticas o coloreadas** pueden introducir cierta flexibilidad en la distribución de tokens, el número de tokens generados sigue estando **delimitado por la estructura y reglas de la red**.

Por lo tanto, el disparo de una transición **siempre genera una cantidad finita de tokens**, definida por los pesos de los arcos en el modelo.

**Una tabla de decision puede incluir consiciones no atomicas si son esenciales para el problema que se modela. (V o F) Justificar.**

**Verdadero.**

Una **tabla de decisión** puede incluir **condiciones no atómicas** si son esenciales para representar correctamente el problema que se modela.

**Justificación:**

* **Las condiciones atómicas** son aquellas que no pueden descomponerse más, como "El usuario es mayor de edad (Sí/No)".
* **Las condiciones no atómicas** pueden involucrar expresiones más complejas, como "El usuario es mayor de edad Y tiene cuenta activa".
* **En algunos problemas, es más eficiente y claro representar condiciones compuestas** en lugar de desglosarlas en múltiples condiciones atómicas.

Sin embargo, se recomienda mantener la tabla de decisión lo más simple posible. Si una condición no atómica puede dividirse en elementos más básicos sin perder claridad, es preferible hacerlo para mejorar la legibilidad y evitar redundancias.

**En la nivelacion DFD (Diagrama de flujo de datos), las entradas y salidas de un proceso padre deben mantenerse en los diagramas hijos correspondientes .(V o F) Justificar.**

**Verdadero.**

En la **nivelación de Diagramas de Flujo de Datos (DFD)**, las **entradas y salidas de un proceso padre deben mantenerse en los diagramas hijos correspondientes** para garantizar la coherencia y continuidad del modelo.

**Justificación:**

* **Consistencia del sistema:** Cada nivel de descomposición debe reflejar correctamente los datos que entran y salen del proceso en el nivel superior.
* **Trazabilidad:** Mantener las mismas entradas y salidas permite rastrear cómo fluye la información a través de los distintos niveles del DFD.
* **Evita pérdida o creación arbitraria de datos:** Si un diagrama hijo introduce nuevas entradas o salidas que no aparecen en el padre, o si elimina alguna existente, se rompe la integridad del modelo.

Por lo tanto, **las entradas y salidas del proceso padre deben conservarse en los diagramas hijos** para que la estructura del sistema sea clara y consistente en todos los niveles de detalle.

**El modelo en espiral combina actividades de desarrollo con la gestion de riesgos, permitiendo iteraciones y ajustes rapidos. (V o F) Justificar.**

**Verdadero.**

El **modelo en espiral** combina actividades de **desarrollo** con la **gestión de riesgos**, permitiendo iteraciones y ajustes rápidos a lo largo del ciclo de vida del software.

**Justificación:**

* **Iteraciones continuas:** El modelo en espiral se basa en ciclos repetitivos (iteraciones), donde cada fase mejora y amplía la versión anterior del software.
* **Gestión de riesgos:** En cada iteración, se identifican y analizan los riesgos, lo que permite tomar decisiones informadas y mitigar posibles problemas antes de avanzar a la siguiente fase.
* **Flexibilidad y ajustes rápidos:** Debido a su naturaleza iterativa, el modelo permite realizar cambios en los requisitos o en el diseño en función del feedback y los resultados obtenidos en cada ciclo.
* **Combina enfoques:** Integra elementos del modelo en cascada (estructura organizada) y de metodologías ágiles (adaptabilidad y mejora continua).

Por estas razones, el modelo en espiral es ideal para proyectos grandes y complejos donde la incertidumbre es alta y la gestión de riesgos es clave.

**En los sitemas de informacion, la calidad depende unicamnete del software y no de factores como los datos o los procesos organizacionales. (V o F) Justificar.**

**Falso.**

La calidad de un **sistema de información** no depende únicamente del **software**, sino que también está influenciada por otros factores clave como **los datos** y **los procesos organizacionales**.

**Justificación:**

* **Datos:** La calidad de los datos es fundamental. Un sistema de información puede tener un software bien diseñado, pero si los datos son incorrectos, incompletos o desactualizados, el sistema proporcionará resultados erróneos o ineficaces.
* **Procesos organizacionales:** Los procesos internos de una organización afectan cómo se utilizan y gestionan los sistemas de información. Si los procesos son ineficientes o mal definidos, incluso el mejor software no podrá operar de manera efectiva.
* **Interacción entre los factores:** La calidad de un sistema de información se logra cuando se gestionan de manera efectiva tanto los datos, como los procesos y el software. Estos factores están interrelacionados y son esenciales para asegurar que el sistema cumpla su propósito de manera eficiente y confiable.

Por lo tanto, la calidad de un sistema de información es un resultado de la interacción entre varios factores, no solo del software.

**La normal ISO/IEC 25010 establece un marco para evaluar la capacidad y madurez de los procesos de una organizacion. (V o F) Justificar.**

**Falso.**

La norma **ISO/IEC 25010** no se enfoca en evaluar la **capacidad y madurez de los procesos de una organización**, sino en definir un **modelo de calidad del software**. Esta norma establece un conjunto de **características y subcaracterísticas** que se utilizan para evaluar la calidad del software, como:

* **Funcionalidad**
* **Fiabilidad**
* **Usabilidad**
* **Eficiencia**
* **Mantenibilidad**
* **Portabilidad**

Su propósito principal es proporcionar un marco para evaluar la **calidad del software** desde diferentes perspectivas, pero no se centra en la madurez de los procesos organizacionales.

Por otro lado, la **ISO/IEC 15504 (SPICE)** o el **CMMI (Capability Maturity Model Integration)** son los marcos normativos que están más orientados a la evaluación de la **capacidad y madurez de los procesos** de una organización.

**En el ciclo de mejora continua basado en ISO 9001, el modelo PDCA (Plan-Do-Check-Act) se utiliza para implementar y verificar mejoras en procesos de desarrollo. (V o F) Justificar.**

**Verdadero.**

El **modelo PDCA (Plan-Do-Check-Act)** es una metodología clave dentro del ciclo de mejora continua y se utiliza en la norma **ISO 9001** para implementar y verificar mejoras en los procesos de desarrollo y en la gestión de la calidad de manera general.

**Justificación:**

* **Plan (Planificar):** En esta fase, se identifican las áreas de mejora y se planifican las acciones a tomar. Se establecen objetivos y se diseñan planes para lograr mejoras en los procesos.
* **Do (Hacer):** Aquí se implementan las acciones planificadas. Es la fase de ejecución de las mejoras propuestas.
* **Check (Verificar):** Se revisan los resultados de las acciones implementadas. Se evalúa si los procesos están alcanzando los objetivos establecidos y se realizan análisis para determinar si hay desviaciones.
* **Act (Actuar):** Basado en los resultados de la fase de verificación, se toman acciones para ajustar, corregir o estandarizar los procesos según sea necesario, cerrando el ciclo o comenzando una nueva iteración de mejora.

Este ciclo **iterativo** permite realizar ajustes continuos en los procesos para mejorar la eficiencia y calidad a lo largo del tiempo, lo que lo convierte en un pilar del sistema de gestión de calidad basado en ISO 9001.

**El software es el conjunto de los programas de computo, procedimientos, reglas, documentación y datos asociados orman parte de las operaciones de un sistema computacional. (V o F) justificar.**

**Verdadero.**

El **software** no se limita solo a los programas de cómputo, sino que también incluye **procedimientos, reglas, documentación y datos asociados**, todos los cuales son fundamentales para el funcionamiento de un sistema computacional.

**Justificación:**

* **Programas de cómputo:** Son el conjunto de instrucciones ejecutables que permiten realizar diversas tareas en el sistema.
* **Procedimientos y reglas:** Establecen cómo deben ejecutarse los programas y cómo interactúan con otros componentes del sistema.
* **Documentación:** Incluye manuales de usuario, guías técnicas y especificaciones que facilitan la comprensión, uso y mantenimiento del software.
* **Datos asociados:** Son la información con la que opera el software, como configuraciones, bases de datos y archivos utilizados en los procesos computacionales.

Dado que todos estos elementos son esenciales para el correcto funcionamiento de un sistema computacional, la afirmación es **verdadera**.

**Los puntos de vista se pueden utilizar como una forma de clasificar los stakeholders. (V o F) Justificar.**

**Verdadero.**

Los **puntos de vista** pueden utilizarse como una forma de clasificar a los **stakeholders** (partes interesadas) en un proyecto, ya que permiten organizar a los involucrados según sus intereses, preocupaciones y responsabilidades dentro del sistema.

**Justificación:**

* **Diferentes stakeholders tienen diferentes intereses** en un sistema. Por ejemplo, un usuario final se preocupa por la usabilidad, mientras que un gerente se enfoca en costos y tiempos de desarrollo.
* **Los puntos de vista ayudan a estructurar la comunicación** con cada grupo de stakeholders, facilitando el análisis de requerimientos desde múltiples perspectivas.
* **Clasificar stakeholders por puntos de vista mejora la toma de decisiones**, ya que permite identificar y abordar las necesidades específicas de cada grupo de manera organizada.

Por ejemplo, algunos puntos de vista comunes en la ingeniería de software incluyen:

* **Operacional:** Perspectiva de los usuarios que interactúan con el sistema.
* **Gerencial:** Enfocado en costos, tiempos y viabilidad del proyecto.
* **Técnico:** Considera arquitectos, desarrolladores e ingenieros que diseñan y construyen el software.

Dado que los puntos de vista ayudan a agrupar a los stakeholders en función de su relación con el sistema, la afirmación es **verdadera**.

**Los requerimientos funcionales describen una interaccion entre el sistema y su ambiente . (V o F) Justificar.**

**Verdadero.**

Los **requerimientos funcionales** describen **qué debe hacer el sistema**, incluyendo las interacciones entre el sistema y su entorno (usuarios, otros sistemas, dispositivos, etc.).

**Justificación:**

* **Definen las funcionalidades del sistema**, es decir, los servicios o comportamientos esperados en respuesta a determinadas entradas o eventos del ambiente.
* **Involucran interacciones con actores externos**, como usuarios que ingresan datos, sistemas que intercambian información o dispositivos que envían señales.
* **Ejemplos de requerimientos funcionales:**
  + "El sistema debe permitir a los usuarios autenticarse con nombre de usuario y contraseña."
  + "El sistema debe enviar un correo de confirmación después de que el usuario complete una compra."
  + "El sistema debe registrar cada transacción en la base de datos."

Dado que los requerimientos funcionales especifican cómo el sistema responde a su entorno, la afirmación es **verdadera**.

**Un requerimiento es consistente cuando no necesita ampliarse . (V o F) Justificar.**

**Falso.**

Un **requerimiento es consistente** cuando **no entra en conflicto con otros requerimientos del sistema**, no cuando "no necesita ampliarse".

**Justificación:**

* **Consistencia significa ausencia de contradicciones.** Un requerimiento es consistente si no contradice otros requerimientos funcionales o no funcionales del sistema.
* **La necesidad de ampliación no implica inconsistencia.** Un requerimiento puede ser claro pero aún requerir detalles adicionales sin que esto signifique que sea inconsistente.
* **Ejemplo de inconsistencia:**
  + Requerimiento 1: "El sistema debe permitir a los usuarios cambiar su contraseña sin requerir confirmación por correo electrónico."
  + Requerimiento 2: "El sistema debe enviar un correo de confirmación antes de que un usuario pueda cambiar su contraseña."
  + Estos dos requerimientos se contradicen, lo que significa que el sistema no puede cumplir ambos simultáneamente.

Por lo tanto, **un requerimiento es consistente cuando no presenta contradicciones dentro del conjunto de requerimientos del sistema, no cuando no necesita ampliarse**.

**Una máquina de estado finito tiene uno o varios finales . (V o F) Justificar.**

**Verdadero.**

Una **Máquina de Estado Finito (FSM, por sus siglas en inglés)** puede tener **uno o varios estados finales**, dependiendo de su diseño y propósito.

**Justificación:**

* **FSM con un solo estado final:** Algunos autómatas terminan su ejecución en un único estado final.
* **FSM con múltiples estados finales:** Existen máquinas de estado que pueden finalizar en diferentes estados dependiendo de las transiciones y las entradas recibidas.
* **Ejemplo:**
  + Un **autómata que valida cadenas** puede tener múltiples estados de aceptación, dependiendo de la estructura de la cadena procesada.
  + Un **sistema de control de usuario** puede tener estados finales como "Sesión cerrada" o "Cuenta bloqueada", según la interacción del usuario.

Dado que una Máquina de Estado Finito **puede tener uno o más estados finales**, la afirmación es **verdadera**.

**En una red de Petri no se pueden representar tareas que se realizan en paralelo . (V o F) Justificar.**

**Falso.**

En una **Red de Petri**, **sí se pueden representar tareas que se realizan en paralelo**, ya que este modelo matemático es especialmente útil para modelar **procesos concurrentes y sincronización**.

**Justificación:**

* **Capacidad de modelar concurrencia:** Una Red de Petri permite representar **eventos que pueden ocurrir simultáneamente**, lo que la hace adecuada para modelar **tareas en paralelo**.
* **Distribución de tokens:** Si una transición tiene **múltiples lugares de salida**, puede generar tokens en diferentes lugares al mismo tiempo, representando **ejecución paralela**.
* **Ejemplo:**
  + Un sistema de procesamiento de pedidos donde una orden puede **ser empaquetada y facturada al mismo tiempo**.
  + Un sistema operativo donde varios procesos pueden **ejecutarse en paralelo**, sincronizándose cuando sea necesario.

Dado que las Redes de Petri **permiten modelar paralelismo y concurrencia**, la afirmación es **falsa**.

**La simplicidad del modelo en espiral hace que sea facil de explicarlo a los clientes . (V o F) Justificar.**

**Falso.**

El **modelo en espiral** no es simple, sino que es un modelo complejo que combina el desarrollo iterativo con la gestión de riesgos, lo que puede hacer que su explicación a los clientes no técnicos sea difícil.

**Justificación:**

* **Estructura compleja:** El modelo en espiral tiene múltiples ciclos de planificación, análisis de riesgos, desarrollo y evaluación, lo que lo hace más difícil de comprender que modelos más lineales como el **modelo en cascada**.
* **Enfoque en la gestión de riesgos:** Incluye una fase específica para la identificación y evaluación de riesgos, lo cual es esencial para los desarrolladores y gestores, pero puede resultar complicado de explicar a clientes sin experiencia técnica.
* **Dificultad para visualizar el progreso:** A diferencia de modelos más simples como **Cascada o Incremental**, en el modelo en espiral el avance del proyecto no sigue un camino lineal claro, lo que puede confundir a los clientes.

Por estas razones, la afirmación es **falsa**, ya que la estructura iterativa y basada en riesgos del **modelo en espiral** lo hace difícil de explicar a clientes sin conocimientos técnicos.

**Uno de los valores de las metodologías agiles es tener la documentación completas . (V o F) Justificar.**

**Falso.**

Las **metodologías ágiles** no priorizan tener una documentación completa, sino que enfatizan la comunicación directa y la entrega de software funcional.

**Justificación:**

* **En el Manifiesto Ágil**, uno de los valores clave es:  
  **"Software funcionando sobre documentación extensiva"**.  
  Esto significa que, aunque la documentación es útil, no debe ser un obstáculo para el desarrollo y entrega de valor.
* **Las metodologías ágiles buscan flexibilidad**, por lo que se enfocan en la documentación esencial y no en documentación extensa y rígida como en modelos tradicionales.
* **Ejemplo:** En **Scrum**, los requerimientos suelen expresarse como **historias de usuario**, que son simples y directas, en lugar de documentos detallados.

Por lo tanto, la afirmación es **falsa**, ya que **las metodologías ágiles priorizan la entrega de software funcional sobre la documentación completa**.

**La calidad es la conformidad con los rquisitos explicitos e implicitos de un cliente . (V o F) Justificar.**

**Verdadero.**

La **calidad** en el desarrollo de software se define como la **conformidad con los requisitos explícitos e implícitos** del cliente, lo que significa que un producto de calidad debe cumplir tanto lo que el cliente solicita directamente como lo que se espera de manera implícita.

**Justificación:**

* **Requisitos explícitos:** Son los que el cliente especifica de manera directa, como funcionalidades, tiempos de respuesta o compatibilidad con ciertos dispositivos.
* **Requisitos implícitos:** Son aquellos que no siempre se mencionan, pero que se asumen como estándar en un producto de calidad, como la seguridad, usabilidad y mantenibilidad.
* **Ejemplo:**
  + Un cliente puede pedir explícitamente que un sistema permita gestionar pedidos en línea (requerimiento funcional).
  + Implícitamente, se espera que el sistema sea estable, seguro y fácil de usar, aunque el cliente no lo haya mencionado directamente.

Dado que la calidad implica satisfacer tanto los requisitos explícitos como los implícitos del cliente, la afirmación es **verdadera**.

**La calidad del producto obtenido y la calidad del proceso de desarrollo son independientes . (V o F) Justificar.**

**Falso.**

La **calidad del producto obtenido** y la **calidad del proceso de desarrollo** están **íntimamente relacionadas** y no son independientes.

**Justificación:**

* **Proceso de desarrollo de calidad:** Un **proceso bien gestionado y de alta calidad** tiene una gran probabilidad de generar productos de alta calidad. Esto se debe a que un proceso controlado y eficiente garantiza que los errores se detecten y se corrijan rápidamente, que se cumplan los requisitos del cliente, y que el trabajo se realice con buenas prácticas y estándares.
* **Calidad del producto:** Si el proceso de desarrollo es deficiente, es probable que el producto final tenga defectos, no cumpla con los requisitos o sea ineficiente.
* **Relación entre ambos:** Una mejora en el proceso de desarrollo generalmente **mejora la calidad del producto final**, ya que el control de calidad, las pruebas, y el manejo adecuado de los requisitos son esenciales en el desarrollo del software.

Por lo tanto, la calidad del producto y del proceso son **dependientes** y la afirmación es **falsa**.

**En la visión holística de la calidad Stylianou y Kumar, la calidad de la infraestructura esta relacionada con la calidad de la información . (V o F) Justificar.**

**Verdadero.**

En la visión **holística de la calidad** de Stylianou y Kumar, se considera que la **calidad de la infraestructura** está estrechamente relacionada con la **calidad de la información**.

**Justificación:**

* **Infraestructura de TI**: Esto se refiere a los componentes tecnológicos (hardware, software, redes, etc.) que permiten el almacenamiento, procesamiento y transmisión de información. Una infraestructura de calidad facilita el acceso a datos precisos, rápidos y seguros.
* **Calidad de la información**: Se refiere a la precisión, confiabilidad, relevancia y accesibilidad de los datos. Si la infraestructura tecnológica no es adecuada, la calidad de la información se ve afectada, ya que no será posible almacenar, procesar o transmitir los datos de manera efectiva.

En esta visión holística, se entiende que ambos elementos están interconectados: una infraestructura robusta y eficiente facilita una gestión óptima de la información, lo que a su vez mejora la calidad global del sistema o proceso.

Por lo tanto, la afirmación es **verdadera**, ya que **la calidad de la infraestructura influye directamente en la calidad de la información**.Principio del formulario

Final del formulario