1 Define los siguientes conceptos:

1.1 Ingeniería del software.

La *ingeniería del software* es una disciplina que se enfoca en el desarrollo sistemático y eficiente de software, aplicando principios de ingeniería para diseñar, construir, probar y mantener sistemas de software de calidad que cumplen con los requisitos y expectativas de los usuarios.

1.2 Metodologías clásicas.

Las *metodologías clásicas* en el desarrollo de software son enfoques tradicionales que se caracterizan por ser secuenciales y tener una planificación detallada al inicio del proyecto, donde cada fase depende de la finalización de la anterior. Lo cual implica una gran rigidez frente a cambios. Esto impulsó la evolución hacia enfoques más flexibles y colaborativos, las metodologías ágiles.

1.3 Metodologías ágiles.

Las *metodologías ágiles* son enfoques flexibles y colaborativos para el desarrollo de software. Se centran en la entrega continua de software funcional, la adaptabilidad a cambios en los requisitos y la colaboración abierta y efectiva entre equipos.

1.4 Ciclo de vida del software.

El *ciclo de vida del software* representa el conjunto de fases y procesos que un sistema de software atraviesa desde su concepción hasta su obsolescencia. La elección del modelo de ciclo tiene un gran impacto en el desarrollo de software, adaptándose este generalmente a las necesidades del proyecto y las preferencias del equipo.

2 Ejercicios de autoevaluación:

2.1 ¿Qué es el software para sistemas de información?

El *software para sistemas de información* se refiere a los programas y aplicaciones informáticas diseñados para gestionar y procesar datos en un entorno específico, con el objetivo de facilitar el funcionamiento y la toma de decisiones en una organización.

2.2 ¿Cuáles son las características inherentes al software que lo diferencian otros productos industriales?

Las principales características que diferencian al software de otros productos industriales son la intangibilidad, la maleabilidad y la facilidad de replicación.

2.3 ¿Por qué el desarrollo de software se organiza en forma de proyectos?

El desarrollo de software se organiza en forma de proyectos para facilitar su creación, gestionando eficientemente los recursos y riesgos, estableciendo objetivos claros y garantizando la entrega de un producto de software de alta calidad en un plazo determinado.

2.4 ¿Qué relación existe entre las tareas, los roles y los artefactos en un proceso de desarrollo?

En un proceso de desarrollo de software, las tareas son las actividades específicas que se realizan, los roles son las responsabilidades asignadas a las personas y los artefactos son los productos generados. Estos elementos están

interrelacionados, ya que ciertos roles llevan a cabo tareas para producir artefactos clave.

2.5 Indicar cuatro tareas relacionadas con la actividad de gestión del proyecto.

1. Planificación del proyecto.
2. Seguimiento y control del progreso.
3. Gestión de riesgos.
4. Coordinación y comunicación con el equipo.

2.6 ¿Cuáles son las etapas del ciclo de vida en cascada?

Las etapas del ciclo de vida en cascada son: el análisis de requisitos, el diseño, la implementación, las pruebas y el mantenimiento.

2.7 ¿Qué es y para qué sirve MÉTRICA?

MÉTRICA es una metodología de desarrollo de software que sirve para gestionar proyectos en el ámbito de la administración pública de España.

2.8 ¿Cuáles son las prácticas fundamentales del proceso unificado?

Las prácticas fundamentales del proceso unificado son la modelización visual, la gestión de cambios, la verificación continua y la gestión de componentes reutilizables.

2.9 Indicar las ventajas e inconvenientes de la reutilización de software.

Ventajas:

- Ahorro de tiempo y recursos.
- Mayor consistencia y calidad.
- Facilita la actualización y mantenimiento.
- Fomenta buenas prácticas de diseño.
- Acelera el desarrollo.

Inconvenientes:

- Problemas de incompatibilidad.
- Dependencia externa.
- Necesidad de documentación detallada.
- Complejidad de integración.
- Limitación de personalización.
- Posible sobrecarga de funcionalidades.

2.10 ¿Qué son las herramientas CASE y para qué se utilizan?

Las herramientas CASE (Computer-Aided Software Engineering) son software que ayuda en el desarrollo de software, proporcionando soporte para actividades como el modelado, el diseño, la implementación y la prueba de software. Se utilizan para aumentar su eficiencia y calidad.

2.11 ¿Qué es el diagrama de Gantt y para qué se utiliza?

El diagrama de Gantt es una herramienta de gestión de proyectos que visualiza las tareas programadas en el tiempo. Se utiliza para planificar, programar y seguir el progreso de un proyecto, mostrando las relaciones entre las tareas y la duración de cada una.

2.12 ¿Qué es el UML (El leguaje Unificado de Modelado), para qué sirve? ¿Y cuáles son los 13 diagramas de UML?

UML es un lenguaje estándar de modelado visual utilizado en el desarrollo de software. Sirve para visualizar, especificar, construir y documentar los artefactos de un sistema. Los 13 diagramas de UML son:

- Los diagramas de casos de uso.
- Los diagramas de clases.
- Los diagramas de objetos.
- Los diagramas de secuencia.
- Los diagramas de colaboración.
- Los diagramas de estado.
- Los diagramas de actividad.
- Los diagramas de componentes.
- Los diagramas de despliegue.
- Los diagramas de estructura compuesta.
- Los diagramas de tiempo.
- Los diagramas de comunicación.
- Los diagramas de paquetes.

3 Define las metodologías Ágiles, cita las más importantes y explica en qué consisten con sus ventajas e inconvenientes.

Las metodologías ágiles son enfoques de desarrollo de software que se centran en la flexibilidad, adaptabilidad y entrega incremental de productos. Estas metodologías buscan mejorar la colaboración entre equipos de desarrollo y clientes, así como responder de manera efectiva a los cambios en los requisitos del proyecto. Las más importantes son:

Kanban, que es un método visual para gestionar el trabajo y optimizar el flujo de trabajo. Se centra en la visualización de las tareas y limita el trabajo en progreso.

- ***Ventajas:*** Mayor visibilidad, flexibilidad en la gestión del flujo de trabajo y fácil adaptación a cambios.
- ***Inconvenientes:*** Menos estructurado que Scrum, lo que puede llevar a una menor previsibilidad.

Extreme Programming (XP), que es una metodología centrada en las mejores prácticas de desarrollo, como la programación en pareja, las pruebas unitarias frecuentes y las entregas constantes.

- ***Ventajas:*** Mejora la calidad del código, mayor adaptabilidad y retroalimentación continua.
- ***Inconvenientes:*** Puede ser intensivo en recursos y puede resultar desafiante implementar todas las prácticas en algunos entornos.

Lean Software Development (LSD), la cual está inspirada en los principios del *lean manufacturing,* el cual busca minimizar el desperdicio, optimizar el flujo de trabajo y entregar valor continuamente.

- ***Ventajas:*** Eficiencia en la entrega, reducción de costos y enfoque en la satisfacción del cliente.
- ***Inconvenientes:*** Puede requerir un cambio cultural significativo y puede resultar desafiante en proyectos muy complejos.

Crystal, la cual fue desarrollada por *Alistair Cockburn* y es en realidad una familia de metodologías ágiles que se adaptan a diferentes tamaños y complejidades de proyectos.

- ***Ventajas:*** Flexibilidad según el tamaño del proyecto, énfasis en la comunicación y adaptabilidad.
- ***Inconvenientes:*** Requiere experiencia para elegir y aplicar la variante adecuada.

Dynamic Systems Development Method (DSDM), que se centra en la entrega rápida de sistemas, con un enfoque específico en la colaboración entre equipos y la comunicación efectiva.

- ***Ventajas:*** Rápida adaptación a cambios, entrega temprana y enfoque en la calidad.
- ***Inconvenientes:*** Puede requerir una mayor inversión de tiempo y recursos.

4 SCRUM: *(Visiona el siguiente vídeo para ayudarte a contestar este apartado: https://www.youtube.com/watch?v=sLexw-z13Fo)*

4.1 ¿Qué es Scrum?

Scrum es un marco de trabajo ágil que se utiliza para gestionar y desarrollar productos complejos. Se centra en la flexibilidad y la colaboración para responder a los cambios de manera efectiva durante el proceso de desarrollo.

4.2 ¿Scrum es una metodología Ágil o/y un Framework? (Justifica la respuesta)

Scrum es un marco de trabajo ágil, no una metodología. La principal diferencia radica en que una metodología proporciona un conjunto de reglas y procesos más rígidos, mientras que un marco de trabajo, como Scrum, ofrece directrices y roles flexibles que se pueden adaptar según las necesidades del proyecto.

4.3 ¿Qué es un Rol?, ¿para qué sirve? ¿Cuáles son los roles detallados en Scrum?

Un rol en Scrum es una función que cumple una persona dentro del equipo y sirve para organizar y facilitar la ejecución de los ciclos de desarrollo. Los roles principales en Scrum son:

- El ***Product Owner***, que se encarga de definir los objetivos del producto, ayudando con ello a la comunicación cliente-desarrollador.
- El ***Scrum Master***, que facilita el proceso de desarrollo y elimina obstáculos del mismo. Básicamente hace *coaching* sobre Scrum al Equipo de Desarrollo
- El ***Development Team***, que realiza el trabajo real y está formado por toda la gente que desarrolla el proyecto en sí.

4.4 ¿Qué es el backlog?

El backlog en Scrum es una lista ordenada de todas las características, funcionalidades y tareas necesarias para el producto. Puede dividirse en Backlog del Producto, que contiene elementos de trabajo futuros, y Backlog del Sprint, que contiene elementos seleccionados para un sprint específico.

4.5 ¿Qué es el sprint?

Un sprint en Scrum es un período de tiempo fijo, generalmente de 2 a 4 semanas, durante el cual se lleva a cabo el desarrollo de un conjunto de elementos del backlog. Al final de cada sprint, se entrega un incremento potencialmente entregable del producto.

4.6 ¿Cuáles son las características de Scrum?

Las características clave de Scrum son:

- La adaptabilidad a las circunstancias.
- La comunicación y colaboración constantes.
- La entrega incremental de partes.

4.7 Ventajas e inconvenientes de Scrum

Ventajas de su implementación:

- Flexibilidad y adaptabilidad
- Entregas Incrementales y mejora continua
- Transparencia y colaboración activa

Inconvenientes a la hora de implementarlo:

- Resistencia al cambio de paradigma
- Comprensión incompleta de roles y responsabilidades
- Falta de colaboración y comunicación entre miembros del equipo
- Dificultad de estimación y planificación de las actividades