

Preguntas 1er Parcial Ingeniería y Calidad de Software

Unidad 1: INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE

1.1-Introducción a la Ingeniería de Software

1) ¿Qué es la crisis del software?

Son los problemas que aparecen en el desarrollo del software al desarrollar, mantener y atender la demanda de nuevas aplicaciones. Insatisfacción del cliente, planificación y estimaciones imprecisas, calidad, falta de tiempo para recoger datos históricos, baja productividad, dificultad de mantenimiento, etc. Se produjo entre 1965 y 1985.

2) ¿Cuáles son los mitos del desarrollo del software? Mencione un ejemplo de cada uno

Creencias acerca del software y de los procesos empleados para construirlo:

- De la Gestión
 - Los estándares y procedimientos escritos le da al equipo todo lo que necesita (cuando no los hay, se depende de costumbres)
 - El hardware moderno es suficiente para el desarrollo de software de calidad, más allá de las herramientas de software
 - Se puede recuperar el tiempo perdido agregando programadores
- Del Cliente
 - Los cambios en requisitos no tienen gran impacto porque el software es flexible (el impacto depende del momento en que se introduzca el cambio)
- De los Desarrolladores
 - El trabajo termina cuando se escribió el programa y quedó funcionando (60% - 80% del esfuerzo es realizado después de la primera entrega para ser testeado)
 - No hay forma de comprobar la calidad hasta tener el programa ejecutándose
 - Solo se entrega al finalizar el proyecto el programa funcionando

3) Nombre las 3 características del software

- Se desarrolla, no se fabrica: la calidad se obtiene mediante un buen diseño, y los costos se encuentran en la ingeniería
- No se estropea: se deteriora lentamente debido a cambios, y su mantenimiento es más complejo que el del hardware
- Se construye a medida: no se ensamblan componentes existentes, aunque sí se busca la reutilización.

4) Defina: Ingeniería de Software

Disciplina de la ingeniería que comprende todos los aspectos de la producción de software. Abarca desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de este.

Es el tratamiento sistemático de todas las fases del ciclo de vida del software.

5) ¿Cuál es el objetivo de la Ing. de Software basado en la Tecnología multicapa? ¿Y cuál es el fundamento de la Ingeniería de Software?

El objetivo de la Ingeniería de Software es lograr productos de software de calidad mediante un proceso apoyado por métodos y herramientas.

El fundamento de la Ingeniería de Software es la capa del proceso: este define un marco de trabajo para áreas clave (que forman la base del control de gestión de proyectos) y establece un contexto en el cual se aplican métodos técnicos, se producen resultados, se establecen hitos, se asegura la calidad y se gestiona el cambio adecuadamente.

6) Nombre los procesos de Ingeniería de Software y en qué consisten

- Organizacionales: relacionados con los objetivos de la organización y elaboran procesos, productos o activos que ayuden a alcanzar estos objetivos
- Transversales: se apoyan en la contratación, el desarrollo y la entrega de software al cliente y contribuyen a que su uso sea correcto
- De Gestión de Proyectos: relacionados con el establecimiento del proyecto y la coordinación y gestión de los recursos necesarios
- Desarrollo y mantenimiento: de sistemas de software, así como la documentación para usuarios o equipos de mantenimiento
- Soporte: pueden ser empleados por otros procesos de cualquier categoría en cualquier punto del ciclo de vida del software

7) Indique cómo se dividen los procesos de Soporte

- Desarrollo de la documentación
- Gestión de la configuración del software
- Aseguramiento de la calidad
- Resolución de problemas
- Revisiones de software

8) ¿Cuales problemas se identifican en los requerimientos?

- Los usuarios no saben lo que quieren
- Un sistema tiene muchos usuarios y ninguno tiene una visión de conjunto
- Los usuarios no saben cómo hacer más eficiente la operación en su conjunto
- Los usuarios no saben qué partes de su trabajo pueden transformarse en software
- Los usuarios no saben detallar lo que saben en forma precisa

9) ¿Cuáles son las leyes & Lemas de los requerimientos y que indican?

- Ley de Ziv: los requerimientos nunca se entienden completamente
- Ley de Humphrey: los usuarios no saben realmente el software que quieren hasta que lo ven funcionando
- Lema de Wagner: un sistema interactivo nunca puede ser especificado ni testeado por completo

10) ¿Qué tipos de requerimientos hay según la función de calidad? Indique como se ligan con satisfacción del cliente

- Normales: objetivos y metas establecidas. Su presencia satisface al cliente
- Esperados: implícitos. Su ausencia causa mucha insatisfacción
- Emocionantes o Innovadores: superan las expectativas. Su presencia causa mucha satisfacción

11) Nombre qué técnicas de validación de requerimientos pueden usarse

- Revisiones de requerimientos
- Construcción de prototipos
- Generación de casos de prueba
- Análisis de consistencia automática

12) ¿Qué son los requerimientos funcionales, no funcionales y los del dominio? ¿Qué son las restricciones del diseño? (ver cuadro)

- Funcionales: funciones que el sistema será capaz de realizar
- No funcionales: restricciones de los servicios o funciones ofrecidos por el sistema, o del proceso de desarrollo
- Del dominio: provienen del dominio de aplicación del sistema y reflejan sus características. Pueden ser funcionales o no funcionales

Las restricciones del diseño son condicionantes existentes para el diseño, sin anticiparlo.

13) Cuadro de incumplimiento en los requerimientos funcionales y no funcionales: explíquelos

Los requisitos no funcionales establecen restricciones de cómo los requisitos funcionales deben ser implementados, y estos indican lo que el producto debe hacer. Si faltan funcionales, se degrada el sistema, pero si faltan no funcionales, se lo inutiliza. Los no funcionales suelen ser más críticos que los funcionales particulares.

14) En qué se diferencian los “requisitos de usuario” DRU de los “requisitos del software” ERS?

| Requisitos | DRU | ERS |
|------------|--|--|
| Detalles | Pocos | Muchos |
| Para quién | Punto de vista del cliente | Para el equipo de desarrollo |
| Contenido | Descripción del problema y metas esperadas | Responde “¿Qué características debe poseer un sistema que nos permita alcanzar los objetivos, y evitar los problemas, expuestos en los DRU?” |

15) Defina: proceso de Ingeniería de Software (gráfica)

Proceso iterativo e incremental que abarca el ciclo de vida del software, de la definición al desarrollo y al mantenimiento, integrando modificaciones y adaptaciones.

16) ¿Qué debe especificar un proceso de Ingeniería de Software?

- La secuencia de actividades a realizar por el equipo de desarrollo
- Los productos que deben crearse
- La asignación de tareas a cada miembro del equipo
- Los criterios para controlar el proceso

17) Nombre 5 características de los productos de software

- Mantenibles
- Confiabilidad
- Utilización Adecuada
- Eficiencia
- Entendible
- Escalable
- Visible
- Aceptable
- Soportable
- Seguridad
- Robusto
- Rápido

18) ¿Cuáles son los retos de la Ingeniería de Software? Explíquelos brevemente

- De lo heredado: mantener y actualizar el software, evitando costos excesivos y manteniendo el negocio en funcionamiento
- De la heterogeneidad: desarrollar técnicas para construir un software confiable y flexible
- De la entrega: reducir tiempos de entrega sin comprometer la calidad

19) ¿En qué consiste en la Interfaz Usuaria: calidad percibida, calidad externa y calidad en el uso?

- Calidad percibida: características de un producto que aportan gran satisfacción a usuarios específicos
- Calidad externa: instante en que un producto satisface las necesidades establecidas e implícitas al ser utilizado en condiciones específicas
- Calidad en el uso: efectividad, eficiencia y satisfacción con los cuales ciertos usuarios pueden alcanzar ciertas metas en entornos concretos

20) Defina la usabilidad e indique a qué hace referencia

Es la forma en que se alcanzan los objetivos con efectividad, eficiencia y satisfacción.

Instante en que un producto puede ser utilizado por usuarios específicos para alcanzar metas específicas con efectividad, eficiencia y satisfacción en un contexto de uso concreto.

Hace referencia a la rapidez y facilidad con que las personas llevan a cabo sus tareas propias a través del uso del producto.

21) ¿Cuál es la importancia de la usabilidad y que toma en cuenta?

Importancia:

- Reducción de costos de producción
- Reducción de costos de mantenimiento y apoyo
- Reducción de costos de uso
- Mejora en la calidad del producto

Toma en cuenta:

- Rapidez y facilidad de hacer las cosas
- Aproximación al usuario
- Conocimiento del contexto de uso
- Satisfacer las necesidades del usuario
- Los usuarios son los que determinan si el producto es fácil de usar

22) ¿A qué hace referencia la usabilidad y en qué puntos descansa?

Hace referencia a la rapidez y facilidad con que las personas llevan a cabo sus tareas propias a través del uso del producto. Descansa sobre:

- Aproximación al usuario
- Conocimiento del contexto de uso
- Satisfacer las necesidades del usuario
- Los usuarios son los que determinan si el producto es fácil de usar

23) Cuándo se considera la usabilidad?

Debe ser considerada a lo largo de todo el proyecto, más allá del método de desarrollo del sistema.

24) ¿Qué es y para que fines se usa la guía SWEBOOK en Ingeniería de Software?

El Software Engineering Body Of Knowledge es una guía completa y actualizada que identifica las partes del cuerpo de conocimientos sobre los cuales hay un consenso generalizado de la ingeniería de software, las organiza en áreas del conocimiento y crea los medios necesarios para acceder a ellas.

25) ¿Cuáles son las ventajas de las GUI? (Somerville págs. 327 a 347)

- Son fáciles de aprender y utilizar
- Pantallas múltiples: es posible ir de una tarea a otra sin perder de vista la información generada por la primera
- Es posible interactuar rápidamente

26) Nombre los principios de diseño de interfaces usuarias (Somerville)

- Familiaridad del usuario
- Consistencia
- Mínima sorpresa
- Recuperabilidad
- Guía al usuario
- Diversidad de usuarios

27) ¿Cuáles son los 5 estilos primarios de interacción con el usuario según Shneiderman? (Somerville)

- Manipulación directa (de los objetos en pantalla)
- Selección de menús
- Llenado de formularios
- Lenguaje de comandos
- Lenguaje natural

28) Cuales son los 5 lineamientos claves para la utilización efectiva del color en las interfaces de usuario? (Somerville)

- Limitar el número de colores utilizados y ser conservador al utilizarlos
- Utilizar un cambio de color para mostrar un cambio en el estado del sistema
- Utilizar el código de colores para apoyar la tarea que los usuarios están tratando de llevar a cabo
- Utilizar el código de colores en una forma consciente y consistente
- Ser cuidadoso al utilizar pares de colores

29) Que factores de diseño influyen en la redacción de mensajes? (Somerville)

- Contexto
- Experiencia
- Nivel de habilidad
- Estilo
- Cultura

30) ¿En qué consiste la evaluación de la interfaz? ¿Qué comprende? (Somerville)

Es el proceso de valorar la forma en que se utiliza una interfaz y verificar que cumple sus requerimientos.
Técnicas:

- Diseñar y realizar experimentos estadísticos con los usuarios típicos en laboratorios contruidos especialmente con equipos de supervisión
- Cuestionarios que recolectan información de la opinión de los usuarios
- Observación de los usuarios
- Instantáneas de videos del uso típico del sistema

- Inclusión de código que recolecte información de los recursos utilizados y los errores

1.2-Disciplinas de la Ingeniería de Software


31) Identifique los objetivos a tener en cuenta cuando queremos implementar u optar por un servicio de Ingeniería de Software

- Diseñar programas adaptados a las necesidades y exigencias de los clientes
- Desarrollar, desplegar, implementar y mantener los sistemas
- Solucionar problemas de programación
- Estar presente en todas las fases del ciclo de vida del producto
- Estimar y contabilizar los costos, y evaluar el tiempo de desarrollo
- Realizar el seguimiento del presupuesto y cumplir los plazos de entrega
- Liderar equipos de trabajo de desarrollo de software
- Estructurar la elaboración de evidencias que comprueben el funcionamiento del programa
- Diseñar, construir, administrar y mantener las bases de datos
- Liderar y orientar a los programadores durante el desarrollo
- Incluir procesos de calidad, calculando métricas e indicadores y verificando la calidad del software producido
- Estructurar e inspeccionar el trabajo del equipo

32) Nombre las disciplinas de software usadas por un ingeniero de software

- Modelado de Negocios
- Requerimientos
- Análisis y Diseño
- Pruebas
- Administración y Configuración del Cambio
- Administración de Proyectos
- Ambientes
- Implementación

33) Nombre los siete principios de la Ingeniería de Software y analice por qué existe cada principio (el para qué o por qué)

- La razón que exista todo, para no trabajar sin sentido
- Mantenerlo sencillo (KISS), para que sea fácil de usar y de mantener
- Mantener la visión, para no desarrollar algo que no es necesario
- Otros consumirán lo que usted produce, porque hay que ser considerado con los demás
- Abrazar al futuro, para no hacer software que quede obsoleto rápidamente
- Planear por anticipado la reutilización, para asegurar la mantenibilidad pero sin llevarlo al extremo
- Pensar , para tener en claro lo que se debe hacer

34) Nombre y explique las reglas vinculadas al “Código limpio” (90/90, Navaja de Oakham, YAGNI, DRY, ley de Demeter, optimización prematura, etc.)

- 90/90 (principio de Pareto): el primero 90% del código ocupa el 90% del tiempo de desarrollo. El 10% restante del código ocupa el otro 90% del tiempo de desarrollo.
- Navaja de Oakham: siempre pensar antes de añadir algo.
- YAGNI (No lo vas a necesitar): agregar solo aquello que es necesario en un principio, y extender la funcionalidad luego solo si es necesario.
- Diseño grande por adelantado: pensar con detalle suficiente la arquitectura antes de empezar a desarrollar.

- DRY (No te repitas): para más sencillo apoyo y modificación posterior.
- Ley de Demeter (LoD): no hablar con extraños; dividir las responsabilidades y encapsular la lógica dentro de clases, métodos o estructuras.
- Evitar la Optimización Prematura: hacer las cosas sencillas en un primer momento, y optimizar luego si fuera necesario.
- Principio del Menor Asombro: código intuitivo y obvio (menos wtf's/min).
- Inversión de Control (IoC): para bajo acoplamiento y alta modularidad
- Composición sobre Herencia: para lograr polimorfismo y reutilización
- Encapsular lo que varía: aislar puntos de inestabilidad mediante abstracciones
- 4 reglas de diseño simple: los tests pasan, se expresa intención, no hay duplicidades, mínimo número de elementos
- Regla de los Boy Scouts: si se encuentra código que se puede mejorar, se lo debe mejorar
- Último momento responsable: diferir las decisiones hasta el último momento posible para minimizar costos por cambios de requisitos
- Patrones de la GoF
- Patrones SOLID

35) Explique en que consiste SOLID, qué son y para qué sirven estos principios

Grupo de principios de diseño orientado a objetos para lograr software de calidad. Son:

- SRP (Single Responsibility): una sola razón para que cambie una clase
- OCP (Open/Closed): abierto para extensión, cerrado para modificación
- LSP (Liskov Substitution): la clase heredada debe complementar al comportamiento de la base
- ISP (Interface Segregation): muchas interfaces específicas son mejores que una general
- DIP (Dependency Inversion): el programador debe trabajar a nivel de la interfaz y no de la implementación, haciendo que tanto clases de alto nivel como de bajo nivel dependan de abstracciones

36) Nombre cuatro malas prácticas de desarrollo y mantenimiento

- Planificación y estimaciones imprecisas
- No se recopilan datos de proyectos pasados
- Se invierte más dinero en mantenimiento que en formación de los ingenieros
- No se documenta lo suficiente
- Se pasa directamente a la codificación
- Procesos software improvisados
- No se siguen rigurosamente las especificaciones
- No se hace planificación de riesgos
- Se resuelven crisis inmediatas apagando el fuego
- Se sacrifica funcionalidad y calidad para cumplir plazos
- No se realizan pruebas, verificaciones o revisiones

37) ¿Por qué es complejo el software? Mencione los motivos

- La complejidad del dominio del problema
- La dificultad de controlar el proceso de desarrollo
- Los problemas para caracterizar sistemas discretos

38) Qué se puede proponer para mejorar el desarrollo de software?

- Aplicar métodos, técnicas y herramientas de desarrollo
- Adoptar estándares de desarrollo

- Utilizar la experiencia acumulada
- Documentación
- Aplicar estándares, lecciones aprendidas y buenas prácticas

39) En que consiste el Desarrollo Distribuido o Global de Software (DGS)? ¿Qué escenarios puede tener? (ver cuadro)

| Escenario | Misma organización | Diferente organización |
|-------------|---|---|
| Mismo lugar | Desarrollo co-localizado | Desarrollo co-localizado con subcontratación |
| Mismo país | Desarrollo distribuido (DSD) | Desarrollo distribuido con subcontratación (DSD) |
| Otro país | Deslocalización Desarrollo Global (GSD) | Subcontratación deslocalizada Desarrollo Global (GSD) |

40) ¿Cuáles son los desafíos del Desarrollo Global de Software (DGS)?

- Diferencias culturales y prácticas de negocio del cliente
- Participación apropiada de usuarios y personal de campo
- Consciencia de trabajo local y comunicación informal
- Relación de confianza laboral
- Gestión de conflictos y discusiones abiertas de interés
- Entendimiento común de los requisitos
- Reuniones efectivas
- Demoras

41) ¿Qué es el outsourcing y porqué se recurre a él? ¿Cuál es el punto clave?

Es un sistema de contratación en el que una empresa recurre a otra para realizar tareas especializadas. La razón más importante para recurrir a él es la de delegar la responsabilidad y preocupación de la administración operacional. El punto clave es evitar tener un único proveedor, para no depender de él.

42) ¿Cuáles son los dos tipos diferentes de outsourcing?

- Total: se transfieren las infraestructuras y el personal al proveedor, y este se hace cargo de todas las fases de implementación y asume el riesgo de la propiedad de los recursos.
- Selectivo: el cliente puede cubrir ciertas responsabilidades, para minimizar riesgos respecto al outsourcing total.

43) Indique de qué se tratan los modelos de delivery global (Ver cuadro comparativo, su término, tener en cuenta la descripción y ubicación geográfica)

- Onsite: en las instalaciones del cliente
- Offsite: fuera de las instalaciones del cliente (mismo o distinto país)
- Onshore: empresas del mismo país
- Nearshore: empresas de países vecinos o con zonas horarias similares
- Offshore: empresas en países diferentes
- Híbrido: gestión local (onshore), contratación nearshore u offshore

Unidad 2: PROCESO DE INGENIERÍA DE SOFTWARE Y GESTIÓN ÁGIL

Unidad 2.1: Introducción a la gestión ágil de proyectos de desarrollo de software

44) Defina: ¿qué es agilidad? ¿Qué colaboración requiere la agilidad?

Es la habilidad de responder de forma versátil al cambio y evolución de los requisitos y a las circunstancias del entorno para maximizar los beneficios. Requiere colaboración interna (en el equipo) y externa (con el cliente).

45) Nombre 3 enfoques ligeros y 3 de los más completos (diapositiva del paraguas Agile, de los más usados)

Ligeros:

- SCRUM
- Lean
- Kanban
- XP
- CI
- FDD (Feature Driven Development)
- TDD (Test Driven Development)
- Crystal Clear

Completos:

- DSDM (Dynamic Systems Developing Method)
- AgilePM (Agile Project Management)
- AUP (Agile Unified Process)
- OpenUP (Open Unified Process)
- SAlFe (Scaled Agile Framework)
- DAD (Disciplined Agile Delivery)
- Sappient\Approach
- PMI-ACP

46) ¿Por qué razones surge ágil?

- Entorno cambiante
- Dependencia de procesos rígidos
- Otras:
 - Intangibilidad del software
 - Falta de visibilidad del producto terminado
 - Insatisfacción del usuario
 - Soluciones que no son a medida
 - Costosas fases
 - Errores introducidos en fases tempranas
 - Falta de flexibilidad ante los cambios
 - Proceso de desarrollo encorsetado por documentos firmados
 - Desarrollo más lento
 - Poco entendimiento del negocio

47) Manifiesto por el desarrollo ágil de SOFTWARE: nombre sus valores ágiles

- Individuos e Interacciones sobre Procesos y Herramientas
- Software Funcionando sobre Documentación Exhaustiva
- Colaboración con el Cliente sobre Negociación de Cambios
- Respuesta ante el Cambio sobre Seguimiento de un Plan

48) Dentro de los principios del Manifiesto Ágil, explique el concepto de feedback en ágil (Principios nros.1 y 6)

Feedback es la respuesta obtenida en la comunicación dentro del equipo o con el cliente, que indica qué se está haciendo bien y qué se puede mejorar.

49) Dentro del principio número 9, ¿cuál es la fórmula para mejorar la agilidad?

Atención continua + excelencia técnica + buen diseño = mejora de la agilidad

50) ¿Cuáles son sus elementos claves?

- Poca documentación
- Simplicidad (los valores)
- Análisis como una actividad constante
- Diseño evolutivo
- Integraciones
- Testeos diarios

51) Cuadro comparativo entre Gestión Tradicional vs Ágil

| Tradicional | Ágil |
|--|--|
| Basada en normas provenientes de estándares | Basada en experiencias de proyectos anteriores |
| Equipo por especialización | Equipo multidisciplinario |
| El cliente interactúa con el equipo de desarrollo mediante reuniones | El cliente es parte del equipo de desarrollo |
| Fases | Solapamiento |
| Requisitos detallados | Visión del producto |
| Cierta resistencia a los cambios | Adaptación a los cambios |
| Seguimiento del plan | Seguimiento de backlogs |
| Grupos grandes | Grupos pequeños |
| Muchos artefactos y roles | Pocos artefactos y roles |

52) Ágil vs. Tradicional: ¿cuándo es bueno elegir cada uno?

Es bueno elegir tradicional cuando se consigue desarrollar de forma repetible los productos especificados en el tiempo y con los costes previstos, y es buen elegir ágil cuando se consigue entregar de forma repetible un valor innovador. En tradicional, se busca un acuerdo previamente negociado; mientras que en ágil, el acuerdo es iterativo y adaptable a los cambios.

53) ¿Cuáles son las premisas de Gestión Ágil y Tradicional? Indique cuál es predictiva y cual adaptativa.

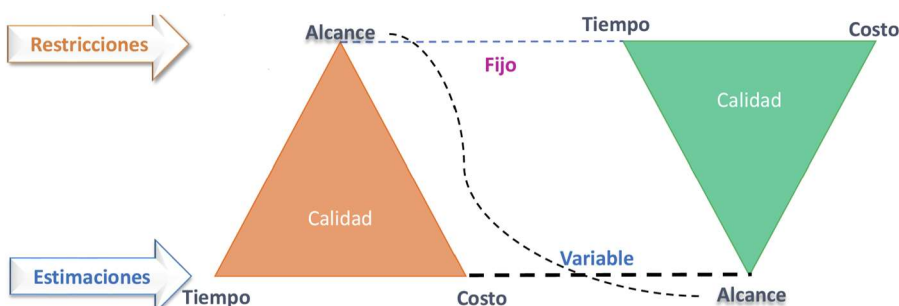
Tradicional (predictiva):

- El plan crea estimaciones de costos/cronogramas
- Todos los proyectos mantienen características y comportamientos regulares
- El objetivo es lograr el producto previsto en el tiempo planificado sin desbordar los costes estimados

Ágil (adaptativa):

- La visión crea estimaciones de las funcionalidades
- No hay una forma única y válida para gestionar cualquier tipo de proyecto
- El objetivo es el valor para el producto; no la funcionalidad, fechas y costes

54) Explique la conversión del triángulo de restricciones de Tradicional a Ágil (armar gráfica)



Un proyecto tradicional se encuentra dirigido al plan: el tiempo y el costo se calculan según el alcance. Para convertirlo a ágil, se debe establecer el tiempo y el costo y, a partir de ellos, ver el alcance; se encuentra dirigido al valor.

55) Nombre por los menos 2 ventajas y 2 desventajas de un framework de gestión tradicional vs. uno ágil

| Framework | Ventajas | Desventajas |
|-----------|--|--|
| Waterfall | <ul style="list-style-type: none"> • Acuerdo temprano con el cliente • Medición sencilla del progreso • Participación de miembros en otros proyectos según la fase activa • El diseño se completa al principio del CVDS • El software puede ser diseñado completamente y con más cuidado | <ul style="list-style-type: none"> • Suele fallar la eficacia de los requisitos • Posibilidad de insatisfacción del cliente con el producto entregado |
| Ágil | <ul style="list-style-type: none"> • Alto nivel de participación del cliente • Sentido de pertenencia del cliente • Oportunidades frecuentes del cliente para ver el trabajo, tomar decisiones y solicitar cambios • Se puede producir una versión básica de software operativo, que puede ser completada luego • Desarrollo centrado en el usuario | <ul style="list-style-type: none"> • Clientes sin tiempo o interés en participar • Es mejor cuando los miembros del equipo están completamente dedicados al proyecto • Posibilidad de no completar elementos establecidos para la entrega en el plazo asignado • Posible frecuente refactorización |

56) ¿Cómo elegimos un framework?

Considerando el marco de trabajo:

- Tamaño de la empresa
- Estructura del equipo
- Recursos disponibles
- Necesidades de las partes interesadas
- Estructura/Tamaño de la cartera de productos
- Prioridades para el cliente en el desarrollo del proyecto

57) ¿Qué tipo de modelo es Scrum?

Es un modelo ágil o de referencia centrado en prácticas de gestión.

58) Defina: empowerment y feedback constante

Empowerment: otorgar autonomía para tomar decisiones al equipo de desarrollo, generando un clima de sinergia grupal.

Feedback constante: para el crecimiento incremental y desarrollo adaptativo de la programación, y una mejora constante en la forma de trabajo del equipo.

Unidad 2.2: Scrum: ceremonias, roles y artefactos

59) Defina Scrum. ¿Qué permite y porqué se dice que es un framework? (diapositivas más adelante)

Scrum es un framework que permite organizar a un equipo para que logre cierto ritmo sostenible y cíclico de trabajo a través de múltiples iteraciones en el transcurso del ciclo de vida del proyecto, de manera en que el equipo se sienta cómodo entregando productos parciales (incrementos) y de valor al cliente.

Es un framework porque solo indica qué deben hacer los equipos para trabajar empíricamente, pero no cómo.

60) Nombre cuáles son los valores o principios que son base de toda la actividad en Scrum (ver gráfica)

- Transparencia o franqueza
- Respeto
- Coraje
- Compromiso
- Enfoque o Focus

61) Explique el proceso Scrum (gráfica diapositiva 14), indicando los roles, ceremonias o eventos, artefactos y como es su flujo/proceso. Identifique cada uno de ellos (usar el resumen)

El proceso comienza con el PO definiendo US para el Product Backlog. En el Sprint Planning, el PO, SM y Scrum Team analizan entre las US que cumplen la DoR, cuáles pasarán al Sprint Backlog. A partir de este, el Scrum Team desarrolla el sprint, teniendo Daily Scrum con el SM donde verifican el progreso y solucionan inconvenientes.

Una vez que finaliza el sprint, se debería tener un incremento que cumpla la DoD. Allí, se realiza un Sprint Review con el PO, el SM, el Scrum Team y los Stakeholders, mostrando lo que se hizo durante el sprint. Luego, se realiza una Sprint Retrospective con el PO, el SM y el Scrum Team, que lleva a un Backlog Refinement para corregir y mejorar las US que corresponda. Luego, comienza el Sprint Planning para el próximo sprint.

62) ¿Qué es un MVP?

Un MVP (Producto Mínimo Viable) es un producto con las características esenciales para satisfacer a los clientes.

63) Según la gráfica de la diapositiva 21, ¿qué comprende un “Sprint cero”?

Comprende la elaboración de documentos iniciales (contrato, MVP, Kick Off), la capacitación metodológica (KT/Knowledge Transfer, DoD, DoR, DoT), el roadmap (alcance y visión), el mapeo de impacto, el mapeo de US, la definición de los ambientes (Development, Testing, Staging, Production) y la definición de la arquitectura.

64) Los roles scrum, ¿qué buscan? (gráfica con tres círculos donde buscan el balance)

- PO: busca que se haga lo correcto, priorizando las US del backlog
- SM: busca que se haga rápido y siguiendo la metodología Scrum
- Scrum Team: busca hacer bien el incremento, con calidad y buenas prácticas, sin dejar deuda técnica.

65) ¿Qué es la deuda técnica? De 3 ejemplos de las formas.

Es la suma de todas las cosas que se deberían haber hecho anteriormente para tener un software decente y de calidad. Es el costo del trabajo adicional causado por la elección de la solución más rápida en lugar de la más efectiva.

Formas: definición inicial del proyecto insuficiente, infringir buenas prácticas de la arquitectura, estructura del código, patrones de diseño o estándares, algoritmos y funciones difíciles de leer y entender, falta de unit tests, alto acoplamiento, alta complejidad del código, falta de code reviews, refactorizaciones atrasadas,

bugs no resueltos en varios sprints, pocos tests de aceptación, duplicación de código, falta de builds y deploys automatizados, falta de colaboración entre equipos, mala aplicación de Scrum, etc.

66) ¿Qué tipos de deuda técnica hay (S. McConnell)? ¿De qué forma se puede gestionar?

Tipos:

- Intencional: decisión consciente de optimizar el presente en lugar del futuro
- No Intencionada/Accidental: error inevitable

Formas de gestión:

- Mantener una lista dentro de un sistema de seguimiento: determinando tareas necesarias, esfuerzo y un cronograma estimado para saldar la deuda
- Mantener una lista como parte de un trabajo pendiente del producto de Scrum: tratando cada deuda como una US

67) ¿Cuáles objetivos se agregan en Scrum 2020?

- Menos prescriptivo, lenguaje más simple y eliminación de terminología específica del software
- No más preguntas en la Daily Scrum: se entrega valor derivado del objetivo del producto: product goal y sprint goal.
- Responsabilidades reemplaza a funciones o roles
- Artefactos reemplaza a componentes

68) Ver el objetivo y responsabilidades de los roles (cuadro)

| Responsable | Objetivo | Responsabilidades |
|---------------|---|--|
| Developers | <ul style="list-style-type: none"> • Generar el incremento usable | <ul style="list-style-type: none"> • Crear el Sprint Backlog • Velar por la calidad (DoD) • Adaptarse cada día para alcanzar el Spring Goal • Ser responsables y profesionales |
| Product Owner | <ul style="list-style-type: none"> • Maximizar el valor del Scrum Team • Entregar un backlog priorizado para el Sprint • Mantener el Product Backlog | <ul style="list-style-type: none"> • Desarrollar y comunicar explícitamente el Product Goal • Aclarar los PBI (Product Backlog Items) • Mantener al PB transparente y visible • Disponer el PB ordenado |
| Scrum Master | <ul style="list-style-type: none"> • Que el Scrum Team y la organización entiendan Scrum • Hacer efectivo al Scrum Team • Ser un líder servicial | <p>Al Scrum Team:</p> <ul style="list-style-type: none"> • Enseñar a ser autogestionado y multifuncional • Acompañar para crear productos de alto valor • Eliminar impedimentos <p>Al PO:</p> <ul style="list-style-type: none"> • Enseñar a definir el Product Goal y mantener el PB • Hacer ver la importancia de PBI claros • Hacer entender que planificamos en un entorno complejo con empirismo • Facilitar colaboración con Stakeholders <p>A la organización:</p> <ul style="list-style-type: none"> • Liderar y organizar la implantación de Scrum • Planificar y asesorar la implantación de Scrum • Ayudar a entender a los Stakeholders como trabaja el empirismo • Eliminar barreras entre Stakeholders y Scrum Team |

69) En Scrum, nombre los resultados si mantenemos los valores en cada persona que trabaja en el equipo que desarrolla el proyecto (reviews, backlog saneado, equipo de alto rendimiento, etc.)

- Reviews del trabajo: presentación de la situación
- Backlog saneado: con priorización actualizada y listo para el siguiente sprint
- Equipo de alto rendimiento
- Sprints cortos
- Time-Box

70) Explique las características de un equipo ágil (autoorganizado, autónomo, multidisciplinario, cross-funcional, autogestionado o empowerment, autogestionado.)

- Autónomo y multidisciplinario: no se debe depender de otros, para evitar esperas y retrasos
- Autoorganización: el equipo decide cómo convertir los PBIs en soluciones que funcionan
- Cross-funcional: el equipo tiene todas las habilidades necesarias para crear el incremento
- No hay títulos: todos son desarrolladores
- No hay sub-equipos
- Comprometidos con el Sprint Goal y la calidad
- Autogestionado (empowerment): se potencia el trabajo en equipo, se motiva al equipo y se consigue el compromiso de sus miembros; dándoles capacidad de toma de decisiones en el Sprint Planning, organizando el Sprint Review y participando activamente en la Retrospectiva.

71) Scrum: cuadro de Eventos, nombre y explique cada evento o ceremonia y sus objetivos (reutilizar para las preguntas que siguen)

72) Defina qué es un sprint y cuánto puede durar

73) Indique la definición de backlog

74) ¿Qué preguntas se responden dentro de la Sprint Planning? (gráfica de qué y cómo)

75) ¿Cómo hacemos un Sprint Planning? Ver las gráficas: de 3 pasos (objetivos, Sprint backlog y Plan) y la de los roles (PO, equipo y SM)

76) ¿Cuál es el único compromiso que asume el equipo?

77) ¿Qué preguntas se responden en una Daily, en qué se centra su dirección y cuál es su propósito?

78) ¿Para qué sirve una Sprint Review o demo?

79) ¿Qué es una agenda o guión de una Review?

80) ¿Qué es una retrospectiva? ¿Cuál es su propósito?

| Evento | Descripción |
|-----------------|---|
| Sprint | Periodo de tiempo durante el que se desarrolla un incremento de funcionalidad. Duración máxima: 30 días. Todos duran lo mismo. Solo lo puede abortar el SM con acuerdo de PO si deciden que no es viable. Se realiza la parte del backlog decidida por el Scrum Team. Backlog: listado con los requisitos del sistema, con su contenido, priorización y disponibilidad. |
| Sprint Planning | Objetivo: planificar el trabajo del sprint. Preguntas: ¿por qué?, ¿qué? (el Sprint Goal) y ¿cómo? (el Sprint Backlog). El Scrum Team decide cuánto entra en el sprint. Se forma el Sprint Backlog. Pasos: <ol style="list-style-type: none"> 1. Definir el objetivo del sprint 2. Crear el Sprint Backlog 3. Elaborar un plan Roles: |

| | |
|----------------------|--|
| | <ul style="list-style-type: none"> • PO: gestiona el PB, está disponible para el equipo y colabora con los Stakeholders • Scrum Team: construye la solución, vela por la calidad y se autoorganiza • SM: Ayuda a eliminar impedimentos, forma y acompaña en prácticas ágiles, es facilitador y líder servicial <p>El único compromiso que asume el equipo es el Sprint Goal, no en elementos específicos del backlog.</p> |
| Daily Scrum | <p>Objetivo: permitir inspeccionar y adaptar el Sprint Goal.</p> <p>Duración: 15 minutos.</p> <p>Su dirección se centra en:</p> <ul style="list-style-type: none"> • Identificación y eliminación de impedimentos en el camino de los desarrolladores • Revisión de prioridades <p>Se pregunta:</p> <ul style="list-style-type: none"> • ¿Qué hicimos ayer? • ¿Qué vamos a hacer hoy? • ¿Qué nos está bloqueando? |
| Sprint Review / Demo | <p>Sirve para inspeccionar el resultado del sprint y generar adaptaciones. El Scrum Team lo presenta a los Stakeholders, y juntos analizan el avance hacia el Product Goal.</p> <p>La agenda o guión es una lista con los issues a presentar (done) y los que no se llegaron a presentar con su estado basado en el Sprint Backlog. Ayuda a ordenar la Demo.</p> |
| Sprint Retrospective | <p>Reunión con una duración fija realizada al final de cada sprint para que el equipo reflexione sobre su progreso, darse retroalimentación y mejorar.</p> <p>Objetivo: mejorar la calidad y efectividad del equipo.</p> <p>Se inspecciona a las personas y sus interacciones, procesos y herramientas.</p> <p>Se analiza lo que falló y lo que salió bien, y se identifica acciones de mejora a implementar.</p> |

81) PRACTICA: ARMADO Y DESARROLLO DE USER STORIES Y SUS CRITERIOS DE ACEPTACION APLICANDO LAS REGLAS DE ESCRITURA VISTAS EN LA TEORIA & PRACTICA, y TAMBIEN EL DESARROLLO DE STORY MAPPING SEGÚN CASO PLANTEADO (Para el Global integrador, su Recuperatorio y Finales)

82) Defina una User Story y qué representa. ¿Cómo es su estructura? ¿Cuál es la jerarquía que tiene en un backlog? ¿Indique qué es una Épica y un Storyboard (PRACTICA)

En cada US, se debe proporcionar un identificador y una descripción (como rol quiero X para motivo). Se le debe colocar una estimación (en SP, mediante Scrum Poker por ejemplo), la conversación con aspectos definidos en otros tiempos con otra documentación, los criterios de aceptación (Cuando, Entonces/Espero), Epic Link y US relacionadas.

Se obtienen al dividir una Épica, es decir, una US con una estimación demasiado grande (8, 13 SP o más).

Un Storyboard es una representación visual derivada de Kanban. Tienen al menos tres swimlanes: columnas que representan el progreso del trabajo en un sprint

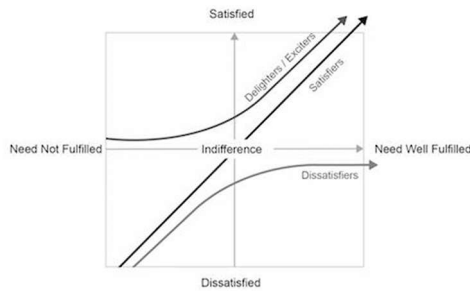
83) Explique en que consiste INVEST (PRACTICA)

Son las características para definir una buena US:

- Independant: pueden completarse en cualquier orden
- Negotiable: cocreadas por los desarrolladores y los clientes
- Valuable: para los clientes o usuarios
- Estimable: los programadores pueden encontrar una estimación razonable
- Small: construibles en poco tiempo (días/persona)
- Testable: se deben poder escribir pruebas para verificarlas

84) Explique las técnicas usadas para priorizar las US (MoSCoW, pares, Kano y 100 puntos) PRACTICA

- MoSCoW: Must, Should, Could, Won't
- Pares: mediante la comparación entre US
- Kano: según cuánta satisfacción o disatisfacción provocará en los clientes el cumplimiento o incumplimiento, se clasifican en satisfactores, disatisfactores y excitantes



- 100 puntos: los participantes distribuyen 100 puntos entre las US, teniendo mayor prioridad las que más puntos sumen

85) Explique las dependencias entre US - PRACTICA

Hay dependencias obligatorias y discrecionales, internas y externas, y combinaciones entre estas.

86) Explique que son los criterios de aceptación y sus formatos/Gherkin (PRACTICA)

Son las condiciones que un producto de software debe satisfacer para ser aceptado por un usuario o cliente. Formatos:

- Comprobar [criterios]
- Demostrar [comportamiento esperado]
- Verificar que cuando [rol] hace [acción] consigue [resultado/comportamiento esperado]
- Dado que [contexto/escenario] y adicionalmente [contexto] cuando [evento/comportamiento] entonces [resultado/comportamiento esperado]
- Gherkin:
 - Escenario [número] [título]
 - Dado que [contexto] y adicionalmente [contexto]
 - Cuando [evento]
 - Entonces [resultado/comportamiento esperado]

87) Explique cuál es el método para medir la calidad de un criterio de aceptación (SMART) (PRACTICA)

- Specific
- Measurable
- Achievable
- Relevant
- Time-boxed

88) Indique cuales son las reglas de escritura para los criterios de aceptación (PRACTICA)

- Evitar "no"
- Usar voz activa
- Evitar pronombres
- Evitar conjunciones
- Evitar absolutos inalcanzables

89) Explique en que consiste el user story mapping. ¿De qué se compone le mapa? (PRACTICA)

En US Mapping, se establece el Goal y, a partir de ello, se obtienen actividades, épicas y, finalmente, US. Las US se ubican todas debajo de la épica de la que surgen, y se grafican líneas horizontales dividiéndolas según el sprint en que se realizarían.

90) ¿Cuál es la función de los artefactos en Scrum? ¿Cuál es el compromiso que contienen? (gráfica)

Su función es aumentar la transparencia, para inspeccionar y adaptar en los eventos. Cada uno tiene un compromiso para asegurar la transparencia.

91) Indique según el cuadro: definición de cada uno de los artefactos y el compromiso que tienen.

| Artefacto | Descripción |
|-----------------|--|
| Product Backlog | Lista ordenada, fuente de trabajo para todo el Scrum Team. Compromiso: Product Goal: estado futuro del producto, que sirve como guía al Scrum Team para sus planificaciones. |
| Sprint Backlog | Contiene el Sprint Goal, los PBIs seleccionados y el plan para lograrlo. Debe tener el detalle suficiente (DoR) para ser desarrollado. Compromiso: Sprint Goal. Debe haber muchas maneras de conseguirlo, de modo que se pueda renegociar con el PO las tareas en caso de no poder alcanzarlo. |
| Incremento | Debe ser un paso hacia el Product Goal, y debe cumplir con la DoD. Puede haber muchos en un mismo Sprint, y se los presenta juntos en la Sprint Review. Compromiso: Definition of Done (DoD): descripción formal del estado del incremento cuando alcanza las medidas de calidad requeridas para el producto. |

92) ¿Que representan los Story Points? ¿Sobre qué tratan?

Los SP representan todo lo que pueda afectar el esfuerzo necesario para completar un ítem. Tratan sobre incertidumbre, complejidad o riesgo.

93) ¿Cómo sabes cómo vamos al estimar?

- Midiendo la velocidad: trabajo completado en un intervalo de tiempo (suele ser un sprint)
- Según el valor aportado

94) Defina qué es el refinamiento y en qué nos ayuda

Reorganización del PB, nos ayuda a dejar listos (DoR) los PBIs y permite tener las conversaciones necesarias para no retrasar al equipo y la entrega de valor.

95) ¿Qué formas hay de estimar? (cuadro)

- Talla de camiseta
- Estimación Mágica
- Planning Poker

96) ¿Qué es y qué diferencia hay entre un PBI y una US?

| US | PBI |
|---|---|
| Explicación general e informal de una función de software escrita desde la perspectiva del usuario final o el cliente | Elemento de alto nivel que describe una funcionalidad general que pueden incluir US y otros elementos |
| Enfoque: centradas en el valor para el cliente | Enfoque: centradas en una amplia variedad de elementos de trabajo |
| Perspectiva: del usuario final | Contenido: especificaciones, solicitudes de nuevas funciones, correcciones de errores, requisitos de cambio |
| Unidad de trabajo: la más pequeña del desarrollo ágil | Priorización: basada en el valor que aporta y en la complejidad |

| | |
|---|-------------------|
| Colaboración: la fomenta con las partes interesadas no técnicas | Inclusión técnica |
|---|-------------------|

97) ¿Qué es el Slicing? ¿Cómo se hace? ¿Qué ventajas tiene?

Es descomponer el trabajo en ítems más pequeños, que ayuden al negocio y al equipo a tener una mejor priorización y estimación al momento de planificar. Se lo puede hacer vertical, horizontal o con el método SPIDR. Ventajas:

- Entrega de valor con más frecuencia
- Planificación más fácil
- Feedback frecuente del negocio
- Mejor sincronización entre personas y equipos
- Mejores prioridades
- Menos riesgo
- Integraciones menos complicadas
- Entendimiento compartido

98) Indique qué es el DoR y el DoD. ¿Qué debería de incluir la definición de DONE?

DoR: técnica para definir las características que debe cumplir un ítem en el sprint

DoD: entendimiento compartido de “completado” para el producto. Es genérico para el incremento, debe ser visible y universalmente entendido, formado con el consenso entre PO, desarrolladores y Testing. Es el denominador común de la calidad del producto, y se suelen relacionar con requerimientos no funcionales.

99) ¿Qué no hay que hacer en Scrum?

- Dejar de respetar el framework
- Ser poco específico
- No comunicarse con el PO
- Dejar de actualizar los tableros/herramientas que gestionan información
- No estar disponible para que se comuniquen conmigo
- No disponer de un PO
- Contar con un SM asignado a varios clientes/proyectos
- Asignar por % a varios proyectos a los miembros del Scrum Team (con excepciones)
- No aplicar buenas prácticas
- Sacrificar la calidad del software

2.3-Desarrollo y Métricas Ágiles

100) ¿Cómo se eligen las métricas en un proyecto ágil?

Se eligen según el proyecto y qué se necesita medir/reportar al manager del proyecto y cliente, que nos permita medir la calidad, el cumplimiento de objetivos, hallar errores y oportunidades de mejora y que se puedan realimentar y actualizar.

101) En Kanban, ¿qué significa WIP y qué se mide?

WIP significa “Trabajo en curso”, estableciendo un límite a cuándo puede haber a la vez, y mide el “lead time” (tiempo medio para completar un elemento), tratando de minimizarlo.

102) ¿Qué es XP?

Es un framework ágil. Significa “Programación Extrema”, y consiste en integrar las prácticas de métodos tradicionales resumiendo o utilizando lo más práctico y eficaz.

103) ¿Qué responsabilidades tiene un arquitecto?

Tiene la responsabilidad de investigar, evaluar y seleccionar las mejores alternativas tecnológicas para atender las necesidades específicas del negocio a un costo razonable.

104) Completar: Los arquitectos deben identificar e involucrar activamente a los interesados de modo de...

- Comprender las restricciones reales del sistema
- Administrar las expectativas de los interesados
- Negociar las prioridades del sistema
- Tomar decisiones de compromiso

105) ¿Cuáles son las tres etapas del proceso de arquitectura de software?

- Definir los requerimientos
- Diseño de la arquitectura
- Validación

106) ¿Qué cuestionamientos se deben hacer para precisar el funcionamiento final del software?

- Costo: ¿cuánto se está dispuesto a invertir en el desarrollo y mantenimiento?
- Duración del desarrollo
- Cantidad de usuarios
- Grado de aislamiento: ¿funciona de forma aislada o se integra con otros productos?

107) Nombre ejemplos de con qué trabaja un arquitecto (ver gráfica de colores)

- Patrones empresariales
- Patrones de arquitectura
- Estilos de arquitectura
- Principios de arquitectura
- Patrones de diseño
- Principios de diseño
- OOP
- Paradigmas de programación
- Código limpio

108) Nombre las responsabilidades que posee un Arquitecto de software

- Articular la visión arquitectónica
- Conceptuar y experimentar con diferentes alternativas tecnológicas
- Crear modelos, componentes y documentos de especificación de interfaces
- Validar la arquitectura contra los requerimientos y presunciones del impacto de la alternativa seleccionada sobre la estrategia tecnológica de la organización

109) Nombre las funciones de un Arquitecto de software. ¿Cuál es la fundamental del rol?

- Gestión de los requisitos no funcionales y definición de la Arquitectura de Software
- Selección de la tecnología (responsabilidad por el riesgo técnico)
- Mejora continua de la arquitectura
- Facilitador, líder y formador en aspectos de arquitectura (no solo para el área de desarrollo)
- Aseguramiento de la calidad (fundamental del rol, mediante CI)

110) Según el tamaño de un sistema, ¿qué tipos de arquitectos especializados pueden intervenir? ¿Cuál es su especialización?

- De soluciones: proporciona soluciones específicas e integrales para una serie de aplicaciones, un proveedor o un programa específico dentro de una organización
- De sistemas: diseña la arquitectura global de sistemas informáticos, involucrando hardware
- Enterprise: diseña la arquitectura para sistemas transversales dentro de una empresa
- Orientado a servicios (SOA): en específico para una tecnología
- DevOps: especializado en CI/CD

111) ¿Qué plantean y para qué sirven las métricas?

Plantean el para qué medimos y qué valor agregan. Sirven para apalancar la mejora continua, ayudando a las organizaciones a enfocar a las personas en lo más importante y en generar valor.

112) ¿Cómo se calcula el avance de un “proyecto” ágil?

Mediante el uso de métricas ágiles (que miden aspectos del proceso de desarrollo) en lugar de métricas empresariales. No sirven la gestión de avance por horas, pues no hay fecha de fin del proyecto.

113) ¿Cuáles son las métricas ágiles principales y su definición?

- Velocidad: cantidad de trabajo completado en un sprint
- Lead time: tiempo transcurrido desde que se solicita un trabajo hasta que se completa
- Cycle time: tiempo que se tarda en completar una tarea desde que se comenzó a trabajar en ella
- Burnup/Burndown charts: representaciones visuales del trabajo realizado frente al planificado
- Evolución de épicas y versiones: para realizar el seguimiento del progreso de cada sprint
- Gráficos de control: tiempo total que está en progreso hasta que se termina
- Diagramas de flujo acumulado: cantidad de trabajo acumulado en diferentes etapas durante un periodo determinado
- Throughput (rendimiento): cantidad de trabajo que se completa en un periodo de tiempo determinado
- Defect rate: cantidad de errores o defectos encontrados en el trabajo entregado
- Customer satisfaction: medida en que el cliente está satisfecho con el producto entregado
- Net promoter score: medida de la disposición de los clientes a recomendar el producto a otros
- Team happiness: satisfacción y compromiso del equipo con su trabajo y el proceso ágil

114) ¿Cómo usar métricas ágiles para optimizar la entrega y cuáles son los antipatrones ante los que hay que estar alerta?

- Sprint burndown chart: permite observar la cantidad de trabajo pendiente en un sprint y la velocidad del equipo. Antipatrones:
 - El equipo termina todos los sprints antes de tiempo
 - El equipo no cumple las previsiones en los sprints
 - La línea de evolución marca caídas pronunciadas
 - El PO añade o cambia el alcance en mitad del sprint
- Evolución de épicas y versiones: informan del flujo de trabajo dentro de la épica y la versión. Antipatrones:
 - Las previsiones de épicas o versiones no se actualizan a media que el equipo avanza
 - No se observa progreso tras varias iteraciones
 - El PO no entiende completamente el problema a solucionar
 - El alcance aumenta con mayor rapidez que la capacidad de atenderlo del equipo
 - El equipo no lanza versiones incrementales a lo largo del desarrollo de las épicas
- Velocidad: permite predecir la rapidez con que un equipo puede recorrer el backlog. Se ve en el burndown. Antipatrones:
 - Muchas variaciones en un largo periodo de tiempo

- Gráfico de control: se centra en el tiempo de ciclo, y permite medir los resultados de los cambios casi instantáneamente. Antipatrones:
 - Se debe buscar tendencias, porque puede ser muy variable al principio
 - Aumento de la duración del ciclo
 - Duración del ciclo heterogénea
- Diagrama de flujo acumulado: ayuda a entender cómo fluye el trabajo, identificar cuellos de botella, evaluar la eficiencia y velocidad de entrega. Antipatrones:
 - Las incidencias que causan bloqueos causan enormes copias de seguridad en algunas partes del proceso, y muy escasas en otros
 - Crecimiento incontrolado del backlog a lo largo del tiempo

115) Describa las tres magnitudes que miden la gestión de proyectos ágiles

- Tiempo: se utiliza “timeboxing” (un sprint en Scrum)
- Velocidad: cantidad de trabajo realizada en un periodo de tiempo
- Trabajo: mediante unidades relacionadas directa o indirectamente con el producto, se registra el trabajo ya hecho y se estima el trabajo a realizar.

116) Explique en qué consiste y qué mide: velocidad, velocidad absoluta y relativa, trabajo, tiempo, tiempo real e ideal, tiempo teórico o de tarea, trabajo ya realizado, trabajo pendiente de realizar, unidades de trabajo y puntos de función.

- Velocidad: magnitud determinada por el trabajo realizado en un periodo de tiempo ideal
- Velocidad absoluta: cantidad de producto construido en un sprint
- Velocidad relativa: cantidad de producto construido en una unidad de tiempo de trabajo
- Trabajo: medida del trabajo realizado o a realizar
- Tiempo: mediante timeboxing
- Tiempo real: tiempo efectivo para realizar un trabajo
- Tiempo ideal, teórico o de tarea: tiempo que sería necesario para realizar un trabajo en condiciones ideales
- Trabajo ya realizado: completo
- Trabajo pendiente de realizar: para estimar
- Unidades de trabajo: relacionada directa o indirectamente al producto, y debe ser consistente
- Puntos de función: unidad de medida relativa para determinar la cantidad de trabajo necesaria para construir una funcionalidad o historia de usuario del Product Backlog

117) ¿En qué consisten las métricas Lead Time y Cycle time?

- Lead time: tiempo total desde que se inicia una tarea hasta que se completa, incluyendo tiempos de espera
- Cycle time: tiempo total que lleva completar una tarea o trabajo desde que se empieza a trabajar activamente en ella hasta que se completa

