Definicion de artefacto de SW

4 Preguntas principales, que quien cuando y como

conjunto de deficiones del proyecto en procesos definidos (p.18) cumplimiento de alcance de proyecto → plan de proyecto cumplimiento de alcance de producto → espeficicacion de requerimientos

Proceso: Que hacer y Ciclo de vida: En que momento y como

Estimación: Tamaño, esfuerzo, calendarización, costo, recursos críticos.

Riesgo: Problemas esperables o sucesos que comprometen el desarrollo del proyecto.

- Gestión:
    - Identifica, Analiza, Planifica, Seguridad y control y aprendizaje.

Métrica del SW:

- Metricas de proceso/proyecto/producto
- Básicas: Tamaño, esfuerzo, tiempo, defectos? Bugs/fixes e insights que podemos recibir de cada sprint (Ej, historias quemadas, tiempo, etc)
- Restricciones: Tecnología, personas, proceso

Manifiesto Agil: Forma de trabajo Sustentada en procesos empíricos

- Ciclos de vida iterativo e incremental, con retroalimentacion rápida
- Asume, construye, retroalimenta, revisa, adapta y arranca de nuevo
- Pilares: Transparencia, inspección, adaptación

Agile: Ideología con un conjunto definido de principios que guían el desarrollo del produto.

- El balance entre no tener proceso y tener demasiado (enfoque en el proceso por encima del resultado) → exige menos documentación
- Los métodos ágiles son:
    - Adaptables en lugar de predictivos
    - Orientados a la gente en lugar del proceso
    - Ej: Scrum, xp, Crystal, FDD y ATDD

Valores Ágiles:

1. Individuos e interacciones sobre procesos y herramientas
2. Software funcionando sobre documentación extensiva
    1. Documentación de lo más importante del producto (arquitectura, estándares, metodologías de testeo, por ejemplo)
3. Colaborar con el cliente sobre negociación contractual

    1. Problemas cuando el cliente solicita cambios en los requerimientos: ayudarle al cliente a obtener lo que necesita, no lo que cree que necesita (BDD, Buzzword-Driven-Design)
    2. Una mala relación el cliente conlleva problemas sobre la triple restricción del proyecto
4. Responder al cambio sobre seguir un plan
    1. Idea de construir juntos con el cliente, aceptando el cambio y buscando crecer.

12 principios, aprender. p.27

Triángulo ágil:

- “Modificación de la Triple Restricción”: Alcance - Calidad Intrínseca - Calidad Extrínseca

Requerimientos ágiles

Va de la mano con el manifiesto ágil, a cumplir con los valores y principios planteados por el manifiesto.

- Gestion de Req: Dedicada a mantener los req integros, identificados y trazados
- Desarrollos de Req: identificar, especificar y validar los requerimientos.

Esto no es igual en el enfoque ágil

Pilares de los Req Ágiles:

- Usar el valor para construir el producto correcto, dandole al cliente lo que necesita, le sirve y funciona.
- Determinar que es “Solo lo suficiente”: requerimientos encontrados de a poco, lo que invita al empirismo
- Usar historias y modelos para mostrar que construir: construir con el cliente y contar con alguien con un profundo conocimiento del negocio, del producto trabajando con el equipo técnico. La etapa de capturar requerimientos integra al cliente, con user stories se intenta crear una visión de qué producto se quiere construir.

Product Backlog

Evita funcionalidades sin uso, los requerimientos cambiantes son una ventaja competitiva si se puede actuar sobre ellos.

Prioridad alta, atendidos en cada iteracion, prioridad baja, se puede agregar y quitar del backlog cualquier requerimiento.

Just in time: Concepto - filosofía que apunta a evitar el desperdicio, no especificar requerimientos hasta que sea necesario.

Comunicacion Cara a Cara: Beneficios de retroalimentación instantanea, construcción y sinergia en la especificación de requerimientos.

Tradicional vs Agil

Tipos de requerimientos

En la gestion agil se trabaja con **req de negocio y de req de usuario.**

Las user stories son requerimientos de usuario alineados a un requerimiento de negocio.

Las US no sirven para especificar requerimientos de SW, sino que sirven para identificar requerimientos de usuario.

El req de negocio y usuario forman parte del **dominio del problema**, y los requerimientos de SW (reqs funcionales/casos de uso) forman parte del domino de la solución.

4 y 6 son los principales

User Stories

- Sirven para identificar requerimientos
- Es una necesidad del usuario
- Descripción del producto
- Item para planificacion
- Token de conversación

Las US son verticales, lo que implica que la funcionalidad es completa: cada us define interfaz, logica de negocio y base de datos entre otros. El material dice que tienen 3 partes (Conversación, tarjeta y confirmación)

Criterio de aceptacion: Base sobre la cual se construye el caso de prueba, es info concreta para saber si lo que implementamos es correcto o no, el PO acepta la US porque acepta los criterios de aceptación de la misma.

- Definen limites para una US
- Ayudan a garantizar que la US provee valor
- Ayuda a crear pruebas para devs y testers
- Ayudan a limitar la funcionalidad agregada por los devs

Los criterios de aceptacion correctos:

- Definen una intencion no una solución.
- Son independientes de la implementacion
- Relativamente de alto nivel

Pruebas de aceptación

- Los detalles van en los casos de prueba(Formatos, saldos, datos a ingresar)
- Complementan la US
- Se contempla el exito y fracaso de la prueba (pasa y falla)

Definition of Ready

Medida de calidad que el equipo crea para definir si la US se puede ejecutar en esta iteración.

Seguir el modelo **INVEST** (p.41)

Explicacion sobre tipos y cualidades de un spike:

- Funcional: necesidades del usuario, para esclarecer como el usuario va a interactuar con el sistema, mejor evaluadas con prototipos para obtener retroalimentacion del usuario.
- Tecnicas: Formas de implementación, familiarización con código o arquitectura, cualquier situacion en la que el equipo necesite una comprensión más fiable antes de comprometerse a una nueva funcionalidad.

Algunas US requieren ambos tipos

A veces spike y ejecución pueden incluirse en la misma tarea.

Estimaciones

Planificacion y estimación no son lo mismo, uno planifica basado en la estimación. No es una instancia única. Hay tendencias a subestimar y sobreestimar.

Se estima porque el costo más grande es el esfuerzo, asumir que las estimaciones no van a ser precisas, estimar entre 2 y 4 veces más de lo estimado originalmente.

Métodos para estimar:

- Basados en la experiencia:
    - Datos historicos
    - Juicio Experto
        - Puro
        - Delphi
    - Analogía
- Basados en recursos
- Basados en el mercado
- Basados en componentes del producto o proceso de desarrollo
- Métodos Algorítmicos

Las estimaciones son un proceso y uno tiene que hacer foco en ese proceso.

El story point no es una medida basada en el tiempo sino que es el peso de la US y dicho peso se usa por comparación.

Se debe tener en cuenta, la complejidad, el esfuerzo y la incertidumbre.

Tamaño vs Esfuerzo

Esfuerzo es el factor más significativo del costo del proyecto, se suele confundir con tamaño. La complejidad es independiente de la persona que lo lleve a cabo mientras que el esfuerzo depende específicamente de la persona que lo ejecuta, no confundir esfuerzo con calendario, un esfuerzo lleva x cantidad de horas aproximadas, pero por eventualidades puede no llegarse a entregar en una fecha pactada.

El trabajo se estima en horas y el producto en SP

Scrum 2020

- Leer guía()
- herramientas
- capacidad del equipo, medida en SP
- Niveles de planificación: Daily → Sprint → Release → Producto

Que y cuando estimar, depende del nivel de granularidad que estemos buscando:

- Portfolio: Talles de remeras
- Backlog: Story points
- Sprint: Historias con DoReady, estimadas en story points

Planificacion del release

- Sprints necesarios
- duracion de sprints
- capacidad estimada del equipo
- objetivos y caracteristicas de cada sprint

*Las tareas van en el sprint backlog no en el product backlog

*Puede que no vaya

Unidad 3 SCM

Disciplina que surge con el proposito de mantener la integridad del producto de SW.

Disciplina de soporte ya que se da de forma transversal al proyecto y ayuda al soporte durante el ciclo de vida del producto. En nuestro trabajo generamos artefactos conocidos como items de configuración.

La integridad del producto, un producto es íntegro cuando:

- Satisface la necesidad del usuario
- Ser fácil de rastrear en su ciclo de vida
- Satisfacer criterios de performance
- Cumple con la expectativa del costo

Conjunto de: Programas, procedimientos, reglas, documentación y datos (entre otras, todas estas son ítems de configuración)

La administración de configuracion de software engloba:

- Control de calidad de producto
- Prueba de software
- Control de calidad de proceso

Todo esto resulta en el aseguramiento de la calidad del software.

De este modo mantenemos la integridad durante el ciclo de vida del producto.

IC: Item de configuración:

Todos los artefactos de proyecto o producto, los cuales pueden sufrir cambios o necesitan ser de publico conocimiento para los miembros del equipo, y de los que se necesita saber su estado y evolución

Repositorio, contiene los items de configuración. Tiene estructura para mantener orden e integridad.

Linea base:

- Configuración revisada formalmente y sobre la que se llega a un acuerdo.
- Sirve como base para desarrollos posteriores y solo se cambia mediante un procedimiento formal de control de cambios.
- Permiten ir atrás en el tiempo y reproducir el entorno de desarrollo en determinado momento del proyecto.
- Pueden ser de: **especificación** u **operacionales**
    - De especificación: No poseen código(de requerimientos, o diseño, info de ingeniería del producto)
    - Operacionales: Lineas base que ya tienen código.

Concepto de branches asociado a linea base. Descartar o hacer merge, master es la rama principal.

Análisis de impacto: Conocer en que gasto incurrir dado determinado cambio, en tiempo y costo monetario.

Versión: Punto particular en el tiempo de un item de configuración, en un contexto dado. Se le aplica control de versiones, se representa gráficamente en forma de grafo (ejemplos de branches de Git)

Variante: una versión del item de configuración que evoluciona por separado.

Actividades principales de SCM

Se debe planificar la gestión de configuración

Identificacion de items

- Identificacion univoca de items
- Convenciones y reglas de nombrado
- Definicion de estructura del repositorio
- Ubicacion del item dentro de la estructura del repositorio

Tipos de items: de Producto, de proyecto y de Iteracion o Sprint

El producto siempre trasciende el proyecto en el que se crea.

Control de Cambios

Asociado a la linea base, busca mantener la integridad de las mismas. Cada vez que se quiera modificar la linea base, debe atravesarse un proceso formal de control de cambios.

De este modo se definen y mantienen las lineas base a lo largo del tiempo.

El CCC, comite de control de cambios, está formado por los involucrados en el desarrollo, referentes de: Análisis, implementacion, testing, etc.

Auditorias de Configuración

Estas necesitan un plan, pueden ser:

- Físicas: Controla que el repositorio sea íntegro, esté donde se determinó, que los items de configuración respeten los esquemas de nombrado definidos. Este tipo de auditoría, hace verificación.
- Funcionales: Se controla la funcionalidad y performance de los productos de SW, se valida que las mismas sean consistentes con la especificación de requerimientos, los items de configuracion deben corresponder a los requerimientos de los que se originan. Este tipo de auditoría hace validación.

Si la auditoría física falla, no se hace la funcional. Estas se hacen por parte de alguien externo al equipo.

Sirve los procesos de:

- Validación: Los problemas son resueltos de manera apropiada, y se le entrega el producto correcto al usuario.
- Verificación: Asegura que el producto cumple con los objetivos preestablecidos.

Informes de Estado

Tienen el objetivo principal de dar un registro de la evolución, para dar visibilidad a la toma de decisiones. Sirve para que los involucrados se enteren del estado de la situación de la gestión de configuración. Se hacen reportes, de inventario, todos los items de config. presentes en el repo en un momento dado, las lineas base de un proyecto. Estos reportes se obtienen de forma automática.

Plan de Gestión de Configuración

Responde a las preguntas respecto a la organización de las 4 actividades básicas de gestión de la configuración. Que voy a hacer y como, registros de informes y estructura del repositorio, entre otros.

Debería incluir:

- Reglas de nombrado
- Herramientas a utilizar para el SCM
- Roles de integrantes del CCC
- Procedimiento formar de cambios
- Procesos de Auditoría
- Plantillas de formularios

Existe para contrastar lo que se planifica contra lo que se ejecuta. Se compara la linea base con el Plan de gestión de configuración.

Se menciona **Integracion, entrega y despliegue continuo**

La diferencia entre las 3 practicas es el nivel de automatización de las pruebas.

Mientras que el manifiesto ágil se enfoca en el proyecto, la SCM se enfoca en el producto, por lo cual trasciende al proyecto. En ambientes ágiles, se puede prescindir de la auditoría de configuración.

1. **Complejidad**: La complejidad de una User Story se refiere a la dificultad inherente asociada con su implementación. Esta dificultad puede deberse a varios factores, como la cantidad de cambios necesarios en el código existente, la necesidad de integración con otros sistemas o componentes, la complejidad del flujo de trabajo

que la User Story describe, entre otros. Cuanto más complicada sea una User Story de entender o de implementar, mayor será su complejidad.
2. **Esfuerzo**: El esfuerzo de una User Story representa la cantidad de trabajo necesario para completarla. Esto incluye no solo el tiempo que se requiere para escribir y probar el código, sino también el tiempo dedicado a tareas relacionadas como la revisión del diseño, la resolución de problemas, las pruebas unitarias y la documentación. El esfuerzo puede ser influenciado por la experiencia y habilidades del equipo, así como por la disponibilidad de recursos y herramientas adecuadas.
3. **Incertidumbre**: La incertidumbre de una User Story se refiere al grado de incertidumbre o ambigüedad que rodea a la historia, especialmente en lo que respecta a los requisitos, las dependencias o los riesgos asociados. Una User Story con alta incertidumbre puede ser difícil de estimar con precisión y puede requerir más investigación o análisis antes de poder ser implementada. La incertidumbre puede disminuir a medida que se obtiene más información o se realizan pruebas adicionales, lo que permite una mejor comprensión y planificación de la implementación.

Dentro de la Guía Scrum 2020, hay varios conceptos fundamentales que son cruciales para comprender y aplicar eficazmente este marco de trabajo ágil. Aquí hay una explicación de algunos de los conceptos más importantes:

1. **Roles de Scrum**: Scrum define tres roles principales: el Product Owner, responsable de maximizar el valor del producto; el Scrum Master, encargado de facilitar el proceso Scrum y eliminar obstáculos; y el Equipo de Desarrollo, autoorganizado y multifuncional, responsable de entregar el Incremento del producto.
2. **Artefactos de Scrum**: Los artefactos de Scrum son elementos clave que proporcionan transparencia y oportunidades de inspección y adaptación. Estos incluyen el Product Backlog, una lista priorizada de todas las funcionalidades deseadas para el producto; el Sprint Backlog, las tareas seleccionadas del Product Backlog para el Sprint actual; y el Incremento, el conjunto de todas las funcionalidades completadas al final de un Sprint.
3. **Eventos de Scrum**: Los eventos de Scrum son oportunidades predefinidas para inspeccionar y adaptar el progreso del trabajo. Estos incluyen la Planificación del Sprint, donde se seleccionan las tareas para el Sprint; el Daily Scrum, una reunión diaria de 15 minutos para sincronizar al equipo; el Sprint Review, donde se inspecciona el Incremento y se recibe retroalimentación; y la Retrospectiva del Sprint, donde el equipo reflexiona sobre su desempeño y planifica mejoras.
4. **Transparencia**: La transparencia es uno de los pilares de Scrum y se refiere a la visibilidad de todos los aspectos relevantes del trabajo realizado por el equipo. Esto incluye la visibilidad del Product Backlog, el Sprint Backlog, el progreso del trabajo y cualquier impedimento que afecte al equipo.
5. **Compromiso con el Objetivo del Sprint**: Todos los miembros del equipo deben comprometerse con el objetivo del Sprint durante la Planificación del Sprint. Este objetivo proporciona una guía clara sobre qué se espera lograr al final del Sprint y ayuda a mantener el enfoque del equipo en la entrega de valor.

No Silver Bullet - Essence and Accident" es un ensayo incluido en el libro "The Mythical Man-Month" escrito por Frederick P. Brooks Jr. Este ensayo explora la idea de que no hay

una "bala de plata" universal que pueda resolver de manera instantánea los desafíos y complejidades del desarrollo de software. Aquí están los conceptos clave de este ensayo:

1. **Esencia y Accidente**: Brooks distingue entre los aspectos esenciales y accidentales del software. La esencia del software se refiere a los problemas inherentes a la naturaleza del desarrollo de software, como la complejidad inherente, la conformidad con los requisitos del cliente y la necesidad de una comprensión precisa de los problemas del dominio. Por otro lado, los aspectos accidentales son aquellos asociados con la implementación y la tecnología, como la elección de lenguajes de programación, herramientas y métodos de desarrollo.
2. **La Complejidad es la Esencia**: Brooks argumenta que la complejidad del software es esencial y no puede ser eliminada mediante soluciones tecnológicas. Esta complejidad proviene de la necesidad de representar conceptos del mundo real en software, adaptarse a los requisitos cambiantes del cliente y gestionar la incertidumbre inherente al proceso de desarrollo.
3. **Bala de Plata**: Brooks sugiere que la búsqueda de una "bala de plata" mágica que resuelva todos los problemas del desarrollo de software es una quimera. No hay un enfoque único o una tecnología revolucionaria que pueda eliminar la complejidad y los desafíos asociados con el desarrollo de software.
4. **Incrementalismo y Reuso**: A pesar de la falta de una solución universal, Brooks aboga por el enfoque incremental y el reuso de componentes probados como estrategias efectivas para abordar los desafíos del desarrollo de software. El enfoque incremental permite la entrega de valor de manera iterativa, mientras que el reuso de componentes reduce la duplicación de esfuerzos y acelera el desarrollo.

En resumen, "No Silver Bullet - Essence and Accident" destaca la complejidad inherente del desarrollo de software y argumenta que no hay una solución única para abordar todos sus desafíos. En lugar de buscar una solución mágica, Brooks aboga por un enfoque pragmático que reconozca la complejidad y adopte estrategias incrementales y de reuso para mejorar la efectividad del desarrollo de software.

Items de configuración

1. **Código Fuente**: Los archivos que contienen el código fuente del software, incluyendo archivos de código fuente, scripts de construcción, y archivos de configuración.
2. **Archivos de Configuración**: Archivos que especifican la configuración del entorno de desarrollo y de ejecución del software, como archivos de propiedades, archivos de configuración XML, archivos de script de configuración, etc.
3. **Documentación**: Documentación relacionada con el software, como especificaciones de requisitos, diseños técnicos, manuales de usuario, manuales de instalación, etc.
4. **Binarios y Ejecutables**: Los archivos binarios y ejecutables generados a partir del código fuente durante el proceso de construcción del software.
5. **Recursos de Datos**: Archivos de datos utilizados por el software, como archivos de base de datos, archivos de configuración de datos, archivos de recursos multimedia, etc.

6. **Pruebas y Scripts de Pruebas**: Artefactos relacionados con las pruebas del software, como casos de prueba, scripts de pruebas automatizadas, registros de resultados de pruebas, etc.
7. **Herramientas y Utilidades**: Herramientas y utilidades utilizadas en el proceso de desarrollo y construcción del software, como compiladores, herramientas de construcción (build tools), herramientas de control de versiones, etc.
8. **Bibliotecas y Dependencias Externas**: Archivos de bibliotecas y dependencias externas utilizadas por el software, como archivos JAR en Java, paquetes de npm en Node.js, etc.
9. **Configuraciones del Entorno**: Configuraciones específicas del entorno de desarrollo y ejecución del software, como variables de entorno, configuraciones de servidor, configuraciones de red, etc.
10. **Entregables del Proyecto**: Los artefactos finales del proyecto, incluyendo los binarios de distribución, la documentación de usuario final, los paquetes de instalación, etc.

Miscelaneos

1. **Auditorías de Software**: Son revisiones sistemáticas de los productos de software o los procesos utilizados para desarrollarlos, con el objetivo de verificar la conformidad con estándares, regulaciones o mejores prácticas.Estas auditorías no solo verifican la conformidad con estándares y regulaciones, sino que también pueden identificar áreas de mejora en los procesos de desarrollo de software. Pueden abarcar aspectos como la seguridad, el rendimiento, la usabilidad y la mantenibilidad del software, así como la conformidad con estándares de calidad como ISO 9001 o normativas específicas de la industria.
2. **Gestión Ágil de Proyectos en Enfoque Lean Agile**: Este enfoque combina los principios ágiles con los principios Lean para maximizar el valor entregado al cliente mientras se minimizan el desperdicio y se optimiza el proceso de entrega. Este enfoque no solo busca maximizar el valor entregado al cliente, sino también optimizar todo el proceso de desarrollo eliminando actividades que no añaden valor (desperdicio) y fomentando la mejora continua. Combina los principios ágiles, como la entrega iterativa e incremental y la autoorganización de equipos, con los principios Lean, como la eliminación de desperdicios, la maximización del flujo de trabajo y el aprendizaje rápido.
3. **Frameworks para Escalar Scrum/Nexus**: Los frameworks como SAFe (Scaled Agile Framework) o LeSS (Large-Scale Scrum) proporcionan estructuras y prácticas para escalar Scrum a equipos más grandes y complejos. Estos frameworks proporcionan estructuras y prácticas para coordinar y sincronizar múltiples equipos que trabajan en un mismo producto o proyecto. Permiten escalar Scrum más allá de un equipo individual, facilitando la colaboración y la entrega integrada de software en organizaciones grandes y complejas.
4. **Gestión Ágil de Proyectos: Scrum**: Scrum es un marco de trabajo ágil que se centra en la entrega iterativa e incremental de productos. Se basa en roles definidos (Product Owner, Scrum Master, Equipo de Desarrollo), artefactos (Product Backlog, Sprint Backlog, Incremento) y eventos (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective).Scrum es un marco de trabajo ágil que se basa en la iteración y la colaboración. Además de los roles, artefactos y eventos mencionados, Scrum

fomenta la transparencia, la inspección y la adaptación continua. Los equipos Scrum se esfuerzan por entregar valor de forma rápida y regular, utilizando ciclos de trabajo cortos llamados sprints.

5. **Componentes de un Proyecto de Software**: Los componentes incluyen requisitos, diseño, implementación, pruebas, despliegue y mantenimiento. Cada fase del ciclo de vida del software tiene sus propios componentes y actividades específicas. Además de las fases mencionadas, cada componente implica actividades específicas. Por ejemplo, en la fase de diseño se pueden incluir la arquitectura del software y el diseño detallado de la interfaz de usuario. En la fase de implementación, se desarrolla y codifica el software según los requisitos y el diseño establecido.
6. **Gestión de la Configuración de Software**: Es el proceso de gestionar y controlar los cambios en el software, incluidos los cambios en el código fuente, la documentación y la configuración del entorno de desarrollo. Este proceso también implica la identificación y control de cambios en la configuración del software a lo largo del ciclo de vida del desarrollo. Se utiliza para garantizar la integridad y la trazabilidad de los elementos de configuración, así como para facilitar la reproducción de versiones específicas del software.
7. **Procesos Definidos y Procesos Empíricos**: Los procesos definidos se basan en planes y predicciones, mientras que los procesos empíricos se basan en la inspección y adaptación continua a medida que se desarrolla el trabajo. Los procesos definidos se basan en la planificación detallada y la ejecución según el plan, mientras que los procesos empíricos se basan en la experiencia y la adaptación continua. En entornos ágiles, los procesos empíricos son preferidos ya que permiten responder rápidamente a los cambios y a la retroalimentación del cliente.
8. **Planificación del Producto en Ambientes Ágiles-Lean**: La planificación del producto en entornos ágiles y Lean implica la colaboración cercana entre los interesados para priorizar el trabajo en función del valor y adaptarse rápidamente a los cambios. Esta planificación implica la participación de todos los interesados, incluidos los clientes y los miembros del equipo de desarrollo. Se enfoca en la entrega temprana de valor al cliente, priorizando las características y funcionalidades más importantes y realizables en cada iteración o entrega.
9. **Requerimientos en Proyectos Ágiles/User Story**: Los requisitos se expresan generalmente como historias de usuario, breves descripciones de funcionalidades desde la perspectiva del usuario final, que se priorizan y se implementan en iteraciones cortas. Las historias de usuario son escritas en lenguaje simple y orientado al usuario, lo que facilita la comprensión y la colaboración entre los desarrolladores y los interesados. Además, las historias de usuario suelen incluir criterios de aceptación que definen cuándo se considera completada una historia.
10. **Prácticas Continuas del Desarrollo de Software**: Incluyen la integración continua (CI), entrega continua (CD) y despliegue continuo (CD), junto con estrategias como despliegue de azul/verde y canario, que permiten la entrega rápida y segura de cambios al software. Estas prácticas no solo automatizan el proceso de construcción, pruebas y despliegue del software, sino que también fomentan una cultura de colaboración, transparencia y mejora continua en el equipo de desarrollo.
11. **Calidad de Producto**: Se refiere a la adecuación del producto a los requisitos y expectativas del cliente, así como a su fiabilidad, usabilidad, rendimiento y

seguridad. Además de las características mencionadas, la calidad del producto también puede incluir aspectos como la escalabilidad, la portabilidad y la compatibilidad con otros sistemas o dispositivos.

12. **Métricas de Software**: Son medidas cuantitativas que se utilizan para evaluar diversos aspectos del software, como la calidad del código, el rendimiento, la seguridad y la satisfacción del cliente. Estas métricas pueden ser utilizadas para identificar áreas de mejora en el proceso de desarrollo, así como para evaluar la calidad del producto final y la satisfacción del cliente.
13. **Aseguramiento de Calidad del Proyecto y el Producto**: Incluye actividades como pruebas de calidad, revisión de código, análisis estático, pruebas de rendimiento y seguridad, y garantiza que tanto el proceso de desarrollo como el producto final cumplan con los estándares de calidad. Este proceso no solo se centra en la detección y corrección de defectos en el software, sino también en la prevención de defectos mediante la aplicación de buenas prácticas de desarrollo y pruebas.
14. **Estimaciones del Software**: Implica estimar el esfuerzo, el tiempo y los recursos necesarios para completar un proyecto de software, utilizando técnicas como estimación por puntos de historia, estimación por expertos y simulaciones. Las estimaciones pueden ser utilizadas para planificar y asignar recursos, así como para establecer expectativas realistas sobre los plazos y costos del proyecto.
15. **Procesos Definidos vs Procesos Empíricos**: Los procesos definidos se basan en la planificación y la predicción, mientras que los procesos empíricos se basan en la inspección y la adaptación continua a medida que se desarrolla el trabajo. Los enfoques ágiles suelen utilizar procesos empíricos para adaptarse rápidamente a los cambios y la incertidumbre. Los procesos definidos suelen ser más adecuados para entornos predecibles y estables, mientras que los procesos empíricos son más adecuados para entornos complejos y cambiantes donde la incertidumbre es alta. Los procesos ágiles se basan en la inspección y la adaptación continua, lo que los hace más empíricos que los enfoques tradicionales de gestión de proyectos.