Guía Resumida de IS Basada en el SWEBOK - Primer parcial

Gestión de la Configuración del Software (SCM)

La Gestión de la Configuración del Software (SCM) es una práctica indispensable en la ingeniería de software que permite gestionar, rastrear y controlar los cambios en los elementos del software. Garantiza la calidad, la integridad y la trazabilidad del producto final durante todo su ciclo de vida.

SCM aplica dirección técnica y administrativa para identificar, documentar, controlar y verificar los elementos de configuración del software.

Ejemplo: En un sistema de comercio electrónico, la SCM asegura que los cambios en las funciones de búsqueda no afecten la estabilidad de otros módulos como el carrito de compras.

1. Gestión del Proceso SCM

La gestión del proceso SCM define cómo se implementarán las actividades de configuración dentro de un proyecto. Incluye planificación, ejecución y seguimiento para asegurar que los cambios sean controlados adecuadamente.

- Para qué sirve: Establece una estructura organizada para gestionar configuraciones y cambios de manera eficiente.

Plan SCM: Este documento central organiza y detalla el proceso SCM. Incluye seis categorías clave de información:

1. Introducción: Propósito, alcance y definiciones.
2. Gestión: Roles, políticas y procedimientos.
3. Actividades: Identificación, control y auditoría de configuración.
4. Cronogramas: Coordinación de actividades.
5. Recursos: Herramientas, infraestructura y personal.
6. Mantenimiento: Actualizaciones al plan según sea necesario.

Relación entre Calidad de Software y SCM

La calidad del software asegura que el producto cumpla con los estándares requeridos y satisface las expectativas de los usuarios. SCM respalda la calidad al proporcionar un marco para manejar cambios de manera controlada, minimizando errores y garantizando que todas las versiones sean consistentes y confiables.

Ejemplo: En un proyecto crítico como el software de gestión hospitalaria, el plan SCM ayuda a coordinar cambios en módulos como registro de pacientes y programación de citas.

2. Identificación de la Configuración del Software

Esta etapa identifica los elementos del software a controlar, como código fuente, documentación y librerías.

- Para qué sirve: Define qué componentes deben ser rastreados para garantizar consistencia.

Elementos clave:

- Elemento de Configuración del Software (SCI): Es cualquier entidad que se administra como una unidad, como el código fuente o un manual técnico.
- Versión de Software: Una instancia específica de un elemento de configuración en su estado actual. Puede evolucionar a través de variantes.
- Línea Base: Un punto de referencia aprobado formalmente para cambios.

Ejemplo: En un sistema bancario, los elementos incluyen módulos de autenticación y transferencia de dinero.

3. Control de Configuración del Software

Controla cómo se solicitan, evalúan y aprueban los cambios en el software.

- Para qué sirve: Permite tomar decisiones informadas y evitar cambios no autorizados.

Tablero de Control de Configuración: Es un grupo que evalúa y aprueba cambios. Incluye representantes de desarrollo, calidad y SCM.

- Decisiones: Aprobar, rechazar o diferir cambios según su impacto.

Diferencia entre exención y desviación:

- Exención: Acepta un elemento que no cumple requisitos, pero es funcional.
- Desviación: Autoriza cambios temporales en los requisitos.

Ejemplo: Si se encuentra un error crítico en una app móvil, el control de configuración asegura que se corrija sin afectar otros componentes.

4. Contabilidad del Estado de la Configuración del Software

Registra y reporta el estado de los elementos de configuración en todo momento.

- Para qué sirve: Proporciona visibilidad sobre el estado de los elementos controlados.

Ejemplo: En un proyecto con múltiples versiones activas, permite identificar cuál es la versión más reciente y aprobada.

5. Auditoría de Configuración del Software

Verifica que los elementos cumplen con los requisitos establecidos.

- Para qué sirve: Garantiza la calidad y consistencia de los elementos de configuración.

Tipos:

- Auditoría Funcional: Confirma que el software cumple sus especificaciones.
- Auditoría Física: Verifica que la documentación concuerde con el producto.

Ejemplo: Antes de lanzar un sistema de gestión de inventarios, una auditoría funcional asegura que los cálculos de stock sean precisos.

6. Gestión y Entrega de Lanzamientos de Software

Se encarga de empaquetar, documentar y distribuir versiones del software.
- Para qué sirve: Asegura que las versiones entregadas sean coherentes y funcionales.

Ejemplo: En un software educativo, una nueva versión incluye funcionalidades para evaluaciones en línea y debe estar correctamente documentada.

7. Herramientas de Gestión de Configuración de Software

Apoyan la implementación de SCM en áreas clave como control de versiones y auditorías.
- Para qué sirve: Automatiza tareas y facilita el seguimiento de configuraciones.

Categorías:
- Herramientas Individuales: Adecuadas para equipos pequeños.
- Herramientas para Proyectos: Soportan entornos distribuidos y complejos.
- Herramientas Empresariales: Automatizan procesos en gran escala.

Guia resumida de Scrum y Planificación - Primer parcial

Scrum es un marco de trabajo ágil que permite gestionar proyectos complejos a través de iteraciones llamadas sprints. Este enfoque promueve la adaptabilidad, la colaboración y la entrega de valor constante, asegurando que los equipos puedan responder rápidamente a los cambios en los requisitos.

El objetivo principal de Scrum es mejorar la planificación y ejecución de proyectos mediante roles definidos, prácticas claras y herramientas adaptables.

Roles en Scrum

1. Product Owner (PO):

- Es el responsable de la visión del producto y la representación de los stakeholders.
- Gestiona el Product Backlog y prioriza las tareas para maximizar el valor del producto.

Ejemplo: En un proyecto de e-commerce, el PO prioriza la función de pago en línea para aumentar las ventas.

2. Scrum Master:

- Facilita la implementación de Scrum y asegura que el equipo siga los principios ágiles.
- Elimina obstáculos y fomenta la comunicación y colaboración.

Ejemplo: Durante un sprint, un Scrum Master ayuda a resolver problemas técnicos que retrasan la entrega.

3. Equipo de Desarrollo:

- Es responsable de construir y entregar los incrementos del producto.
- Auto-gestionado, multidisciplinario y compuesto por un máximo de 9 miembros.

Ejemplo: En una aplicación de banca móvil, el equipo desarrolla funcionalidades como transferencias y pagos.

Sprints y Ciclo de Vida

- Un sprint es una iteración de tiempo fijo (1-4 semanas) en la que el equipo desarrolla uno o más incrementos funcionales.
- Durante el sprint, el alcance puede ajustarse, pero la duración permanece constante.

Ejemplo: Para un sistema de reservas, cada sprint entrega módulos como la búsqueda de vuelos o el pago.

Historias de Usuario y Criterios de Aceptación

- Las historias de usuario describen funcionalidades desde la perspectiva del usuario final.
- Redacción típica: “Como [rol], quiero [función] para [beneficio]”.
- Los criterios de aceptación definen cuándo una historia se considera terminada.

Ejemplo: “Como cliente, quiero buscar productos por categoría para encontrar lo que necesito rápidamente”.

Epics y Product Backlog

- Un epic es una funcionalidad grande que se divide en historias de usuario más pequeñas.
- El Product Backlog es la lista priorizada de todas las tareas del proyecto, gestionada por el PO.

Ejemplo: En un sistema de comercio electrónico:

Epic: Gestión de cuentas de usuario.

Historias de usuario derivadas:

- Como usuario, quiero crear una cuenta para acceder a mis compras.
- Como usuario, quiero recuperar mi contraseña para restaurar el acceso a mi cuenta.
- Como administrador, quiero bloquear cuentas inactivas para mantener la seguridad.

Sprint Backlog e Incrementos Funcionales

- El Sprint Backlog incluye las tareas necesarias para completar los objetivos del sprint.
- Cada sprint debe generar un incremento funcional, algo que idealmente puede pasar a producción.

Ejemplo: En un sprint, el incremento podría ser una funcionalidad de notificaciones en tiempo real.

Reuniones en Scrum

1. Sprint Planning:

- Define el alcance y los objetivos del sprint.
- Participan el PO, el Scrum Master y el equipo de desarrollo.

Ejemplo: Planificar la entrega de una función de búsqueda avanzada.

2. Daily Scrum:

- Reunión diaria de 15 minutos para sincronizar al equipo.
- Preguntas clave: ¿Qué hice ayer? ¿Qué haré hoy? ¿Qué obstáculos tengo?

Ejemplo: Un desarrollador menciona un problema con la integración de API.

3. Sprint Review:

- Presentación del incremento desarrollado y recolección de feedback.

Ejemplo: Mostrar un nuevo módulo de análisis financiero a los stakeholders.

4. Retrospective:

- Analiza lo que funcionó y lo que se puede mejorar.

Ejemplo: Identificar que las historias de usuario eran demasiado grandes y necesitan dividirse.

Refinamiento del Product Backlog

- Proceso continuo para profundizar, desglosar y priorizar los elementos del backlog.

Ejemplo: Reorganizar las prioridades según el feedback del cliente.

Estimaciones Ágiles y Velocidad

Las estimaciones y la velocidad del equipo son fundamentales para planificar y gestionar el trabajo de manera efectiva. Las tareas o historias de usuario se estiman en puntos de historia, los cuales reflejan el esfuerzo relativo necesario para completarlas.

Criterios de Estimación

Se utiliza la serie Fibonacci (0, 1, 2, 3, 5, 8, 13, etc.) para asignar puntos de historia. Esta serie permite reflejar la complejidad y la incertidumbre de las tareas:

- Tareas simples, como "ajustar un botón en la interfaz", pueden asignarse 1 punto.

- Funcionalidades complejas, como "desarrollar un sistema de autenticación", se asignan 8 o más puntos debido al esfuerzo adicional.

La velocidad del equipo es el promedio de puntos completados por sprint y ayuda a medir cuánto trabajo puede manejar un equipo en cada iteración. Se calcula como:

Velocidad = Puntos entregados en los últimos sprints / Número de sprints.

Ejemplo: Un equipo asigna 90 puntos al backlog de un proyecto. Si la velocidad promedio del equipo es de 30 puntos por sprint, se necesitarán:

Número de sprints necesarios = Total de puntos / Velocidad = 90 / 30 = 3 sprints.

Herramientas para Scrum

- Software como Jira, Trello o ClickUp facilita la gestión de tareas y la colaboración.

Guía Resumida de IS Basada en el SWEBOK - Segundo parcial

Gestión de Ingeniería de Software

La gestión en ingeniería de software es esencial para el desarrollo exitoso de proyectos. Se centra en actividades como planificación, coordinación, medición, seguimiento, control y reporte.

El objetivo principal es asegurar que los productos y servicios de software cumplan con las expectativas de calidad y tiempos, beneficiando a todas las partes interesadas.

Las actividades de gestión se dividen en tres niveles clave:

1. Gestión Organizacional e Infraestructura: Abarca la estructura y recursos que soportan los proyectos.
2. Gestión de Proyectos: Incluye la organización y ejecución de tareas específicas dentro de un proyecto.
3. Gestión del Programa de Medición: Implica monitorear y mejorar continuamente mediante métricas y resultados.

Definición de Inicio y Alcance

En esta fase, el equipo se asegura de entender y definir correctamente los requisitos del proyecto y evalúa si es viable llevarlo a cabo.

La correcta definición del inicio y alcance ayuda a evitar confusiones y retrasos más adelante en el proyecto.

Ejemplo: Para un sistema de gestión de ventas, se podría definir el alcance inicial estableciendo que el sistema debe incluir funciones de registro de ventas, generación de

reportes y análisis de rendimiento.

1. Determinación y Negociación de Requisitos: Aquí se recopilan y negocian los requisitos del cliente, asegurando que todas las expectativas sean realistas y alcanzables.
2. Análisis de Viabilidad: Este análisis ayuda a determinar si los objetivos del proyecto son factibles, considerando tecnología, recursos y presupuesto.
3. Proceso de Revisión de Requisitos: A medida que el proyecto avanza, los requisitos se revisan para adaptarse a cualquier cambio o nueva necesidad.

Planificación de Proyectos de Software

La planificación es el pilar de un proyecto de software, ya que establece los pasos necesarios para lograr los objetivos del proyecto. Comienza con la selección de un modelo de ciclo de vida adecuado.

1. Planificación de Procesos: Dependiendo de la naturaleza del proyecto, el equipo puede elegir un modelo SDLC (Software Development Life Cycle) predictivo (como cascada) o adaptativo (como ágil).
   - Ejemplo de SDLC: En un proyecto ágil, el equipo trabaja en iteraciones cortas, ajustando los requisitos y funcionalidades en función del feedback del cliente.
2. Determinación de Entregables: Se identifican los productos que el proyecto debe entregar, como documentación técnica, código fuente y pruebas.
3. Estimación de Esfuerzo, Cronograma y Costos: Se calculan estos elementos para cada tarea, utilizando datos históricos y la experiencia.
4. Asignación de Recursos: Aquí se distribuyen los recursos necesarios, incluyendo personal, herramientas y espacio.
5. Gestión de Riesgos: Se identifican posibles riesgos y se desarrollan estrategias para reducir su impacto.
6. Gestión de la Calidad: Se establecen métodos de control de calidad para asegurar un producto final satisfactorio.
7. Gestión de Planes: Como el cambio es inherente en los proyectos de software, es vital revisar y ajustar los planes a medida que se desarrollan los trabajos.

Ejecución del Proyecto de Software

La ejecución es donde el plan se pone en práctica. Durante esta fase, el equipo de desarrollo sigue el plan, ajustando los procesos según sea necesario para cumplir con los objetivos del proyecto.

1. Implementación de Planes: Todas las actividades se ejecutan siguiendo el plan y los recursos se utilizan conforme a lo planeado.
2. Adquisición de Software y Gestión de Contratos: Aquí se manejan los acuerdos con clientes y proveedores, determinando aspectos clave como los derechos de propiedad y las sanciones por incumplimiento.
3. Monitoreo del Proceso: Permite evaluar el cumplimiento de plazos, costos y calidad,

ajustando el plan si se detectan problemas.

4. Control de Procesos: A partir del monitoreo, se toman decisiones para mantener el proyecto en curso, como reasignar recursos o revisar plazos.
5. Informes: El equipo informa el progreso a las partes interesadas, enfocándose en el avance y posibles ajustes.

Revisión y Evaluación

La revisión es un proceso continuo para asegurar que el proyecto está en el camino correcto y que se cumplen los objetivos. Se revisan tanto los hitos como el rendimiento del equipo.

Ejemplo: Durante una fase de pruebas de un sistema de inventario, se revisa que todas las funcionalidades requeridas estén cubiertas y que cumplan con los estándares de calidad.

Cierre del Proyecto

Al finalizar una fase o el proyecto completo, se evalúa el cumplimiento de los objetivos y se realizan actividades de cierre. Esto incluye el archivado de documentación y una retrospectiva para identificar lecciones aprendidas.

1. Determinación del Cierre: Se cierra formalmente un proyecto o fase cuando se cumple con todos los requisitos y objetivos.
2. Actividades de Cierre: El material del proyecto se archiva y se eliminan datos confidenciales según sea necesario.

Medición en Ingeniería de Software

La medición en ingeniería de software permite mejorar la calidad del desarrollo y optimizar los recursos. Basándose en el estándar IEEE 15939:2008, se establecen procesos de medición y se utilizan métricas para evaluar el progreso y los resultados.

Ejemplo: Medir el tiempo de desarrollo por funcionalidad ayuda a entender si el equipo está cumpliendo con el cronograma y facilita ajustar recursos si es necesario.

Herramientas de Gestión de Ingeniería de Software

Las herramientas de gestión facilitan la planificación y seguimiento de proyectos. Estas pueden incluir herramientas de estimación de costos, diagramas de Gantt para cronogramas, y sistemas de gestión de riesgos.

Ejemplo: Una herramienta de gestión de riesgos permite al equipo monitorear los factores críticos y responder de manera proactiva si se detectan problemas.