Bloque 1: Productos y Proyectos

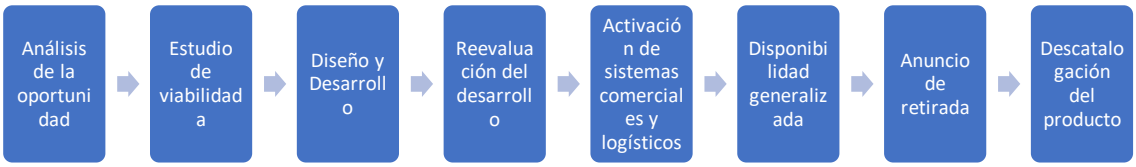
Ciclo de Vida

Progresión según una secuencia de etapas de desarrollo.

Consideramos:

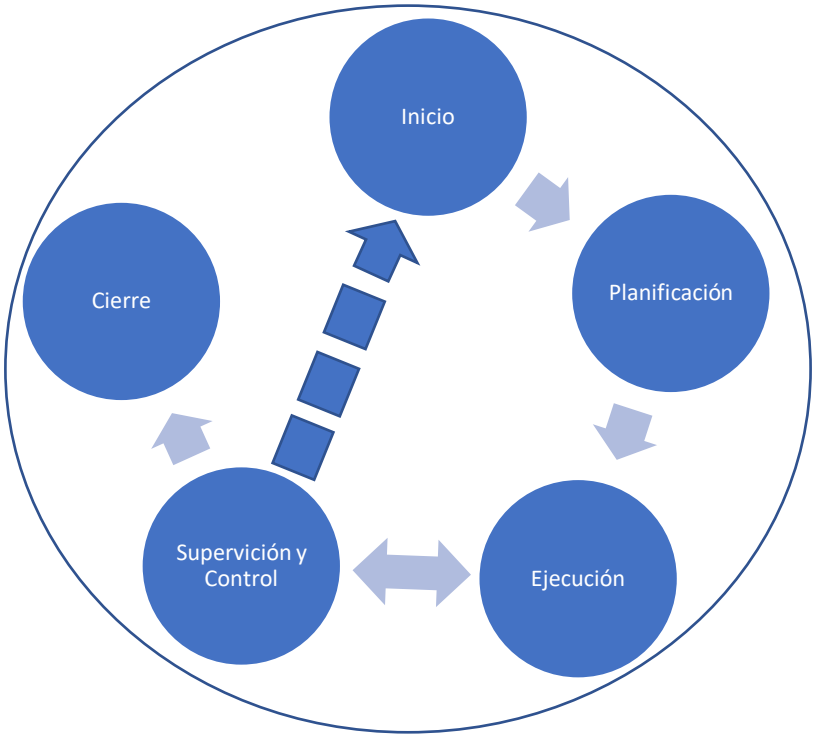
- Ciclo de vida del producto

Concepto – especificación – implementación – puesta en servicio – retirada



- Ciclo de vida del proyecto (metodología del proyecto
    - Dirigidos por planificación (predictivos, en cascada, tradicionales)
    - Dirigidos por cambios o ágiles (iterativos, incrementales
    - Adaptativos.

Proceso (Aquello que hay que hacer para gestionar el trabajo)



Metodología de proyectos VS proceso de gestión de proyectos

La metodología de proyectos varía con el segmento de negocio, la compañía, el tipo producto, etc.

El proceso de gestión de proyectos es válido con independencia de la metodología empleada.

¿Qué es un proyecto? (PMBOK®)

Es una empresa temporal acotada en el tiempo, que crea un producto, un servicio, un resultado concreto y diferenciado.

¿Qué es la gestión de proyectos?

(No es) Simplemente gestionar gente.

(No es) Saber usar un software.

Es una ciencia y un arte, sigue un proceso sistemático.

Requiere conocimientos técnicos, familiarizarse con procesos, conocer a los actores que intervienen y los roles que desempeñan.

El Project Management Institute (PMI) descompone la gestión de proyectos en 5 grupos de procesos y 10 áreas de conocimiento.

¿Qué es PMI?

Una organización internacional sin ánimo de lucro que asocia a profesionales relacionados con la gestión de proyectos. Que tiene los objetivos de:

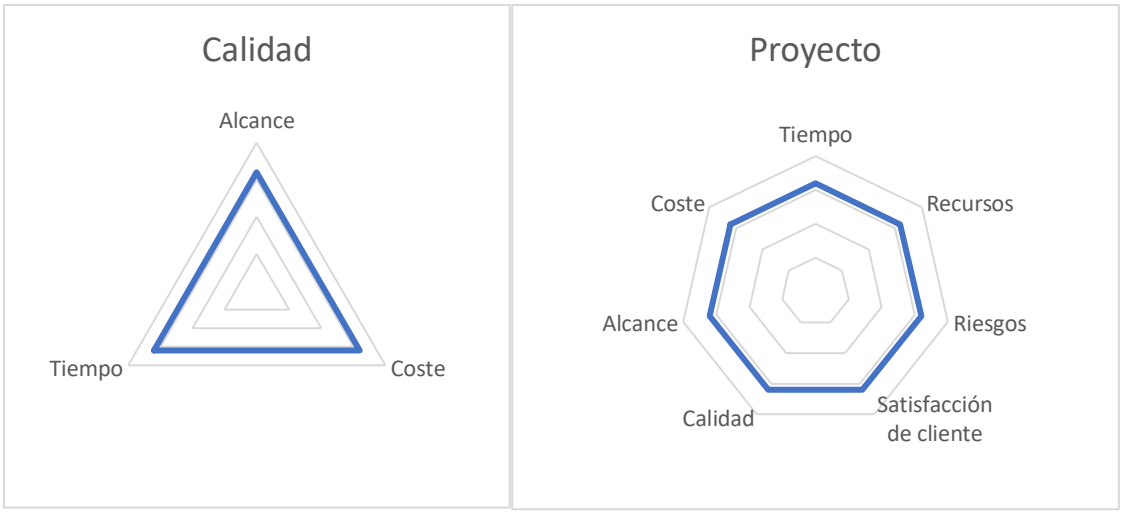
- Formular estándares profesionales en gestión de proyectos.
- Generar conocimiento a través de la investigación.
- Promover la gestión de proyectos como profesión a través de sus programas de certificación.

COLUMNAS (Grupo de procesos de _):

> Inicio, Planificación, Ejecución, Monitoreo y Control, y Cierre.

FILAS (Gestión del _ del proyecto):

> Integración, Alcance, Cronograma, Costes, Calidad, Recursos, Comunicaciones, Riesgos, Adquisiciones e Interesados.



¿Qué es la gestión de programas?

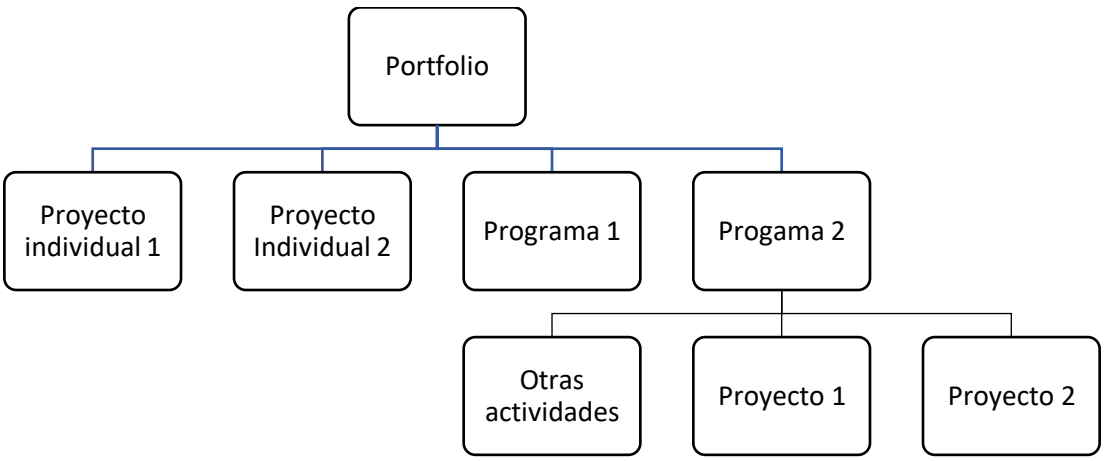
Un programa es un conjunto de proyectos relacionados.

Un programa se crea cuando las interdependencias existentes entre varios proyectos indican que pueden gestionarse con una supervisión conjunta que ayude a minimizar riesgos, explotar sinergias, etc.

Implica un *overhead,* es un coste añadido por seguridad, que al final, si se hace bien, los beneficios superan ese coste añadido.

¿En qué consiste la gestión de un Portfolio?

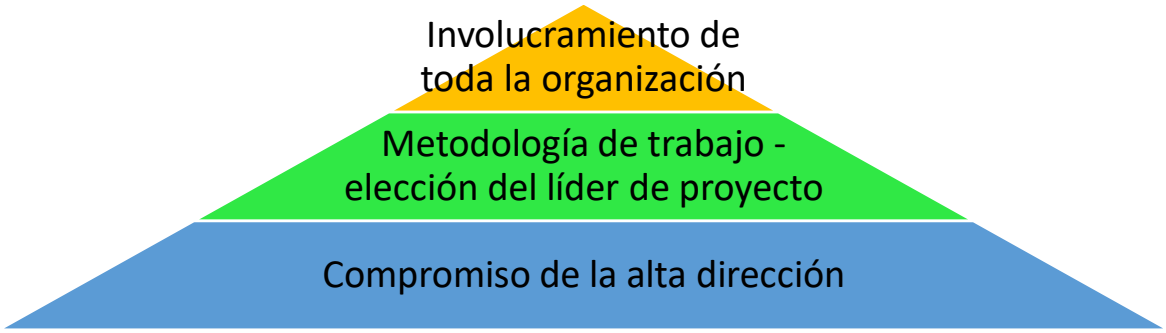
Un portfolio es un grupo de programas, proyectos individuales y otras actividades operativas que se gestionan de forma conjunta para conseguir un objetivo de negocio específico y estratégico.



Beneficios de la gestión de proyectos

- Define y controla el alcance
- Mejora la comunicación entre participantes
- Aumenta la comprensión de los objetivos
- Proyección de los recursos
- Mejor evaluación y mitigación de riesgos
- Prioriza actividades del proyecto vs funcionales
- Alinea la actividad de los grupos funcionales con el negocio de la compañía

Factores de éxito de un proyecto



Oficina de gestión de proyectos (PMO, Project Management Office)

Centraliza y normaliza la gestión de proyectos de una organización.

Tipos: Soporte, Control, Directiva.

Es una estructura departamental (no una persona), que:

- Gestiona las interdependencias.
- Ayuda a suministrar recursos.
- Recomienda la finalización del proyecto.
- Supervisa.
- Ayuda a difundir conocimiento.
- Proporciona plantillas.
- Centraliza la comunicación.
- Se implica más en el inicio del proyecto.
- Forma parte de los comités de cambios.
- Stakeholder.
- Prioriza proyectos.

Claves para la una buena implementación:

- Definición: Función definida claramente.
- Adaptabilidad: Permitir que evolucione.
- Compromiso: Alta dirección de la empresa.
- Rigor: La gestión de proyectos se lleve a cabo de forma profesional.

Estructuras y aspectos organizativos

Tipos:

- Funcionales: jefes de línea
- Orientadas a proyectos: jefes de proyecto
- Matriciales: jefes de proyecto + jefes de línea

Acta de constitución del proyecto (Project charter)

Documento generado por el patrocinador (sponsor) que recoge los requisitos de alto nivel, que autoriza formalmente el proyecto y designa al jefe de proyecto para usar los recursos de la organización.

Stakeholders

Parte interesada, parte afecta.

Gestión de Stakeholders – Herramientas y técnicas

Hay que definir la forma en la que nos vamos a comunicar

- Reuniones
  - Presenciales.
  - Virtuales.
- Habilidades interpersonales
  - Generación de confianza.
  - Resolución de conflictos.
  - Escucha activa.
  - Persuasión, superar la resistencia al cambio.

Gestión de proyectos software

Conseguir un objetivo de negocio.

Criterios de éxito de un proyecto

- Cumplimiento
- Plazo
- Presupuesto
- Equipo

ITIL – Utilidad y Garantía

- Utilidad (Utility)
  - Cumplir con los resultados esperados.
- Garantía (Warranty)
  - Satisfacer los requisitos acordados.

Particularidades de la gestión de proyectos de desarrollo software

- El producto es intangible
- Con frecuencia los proyectos software son únicos (“one-off”)
- Los procesos software suelen ser variables y específicos de cada organización

Factores que influyen en la gestión de proyectos

- Tamaño de la compañía
- Los clientes
- Tamaño del software
- Tipo de software
- Cultura corporativa
- Procesos de desarrollo de software

Actividades comunes a cualquier proyecto

- Planificación
- Gestión de riesgos
- Gestión de personal
- Informar (reporting)
- Redacción de propuestas

PMP: Actividades de planificación que habría que realizar antes de empezar a planificar riesgos

1. Determinar cómo vamos a planificar cada área de conocimiento.
2. Determinar los requisitos detallados del proyecto.
3. Crear el Project scope statement.
4. Valorar qué partes del proyecto se va a subcontratar y preparar los documentos asociados.
5. Determinar cuál va a ser el equipo de planificación.
6. Crear las WBSs y el diccionario WBS.
7. Crear la lista de actividades.
8. Crear un diagrama de actividades.
9. Estimar los recursos necesarios.
10. Estimar el tiempo y el coste.
11. Determinar el camino crítico.
12. Desarrollar el plan de tiempos.
13. Desarrollar el presupuesto.
14. Determinar los estándares de calidad que se van a aplicar, los procesos y las métricas.
15. Crear un plan de mejora de procesos.
16. Determinar todos los roles y responsabilidades.
17. Planificar la gestión de los Stakeholders.
18. Identificar riesgos.

Scope Baseline

Project Scope Statement

Una descripción del alcance del proyecto que incluye los entregables principales, las hipótesis de trabajo y las limitaciones identificadas del proyecto

Work Breakdown Structure (WBS)

Una descomposición jerárquica del alcance total del proyecto que debe ser ejecutado por el equipo para lograr los objetivos del proyecto y desarrollar los entregables requeridos.

Diccionario WBS

Es un documento en el que, para cada componente de la WBS, se proporciona información detallada de cada entregable, de las actividades asociadas y de su planificación temporal.

Definición de Workpackage

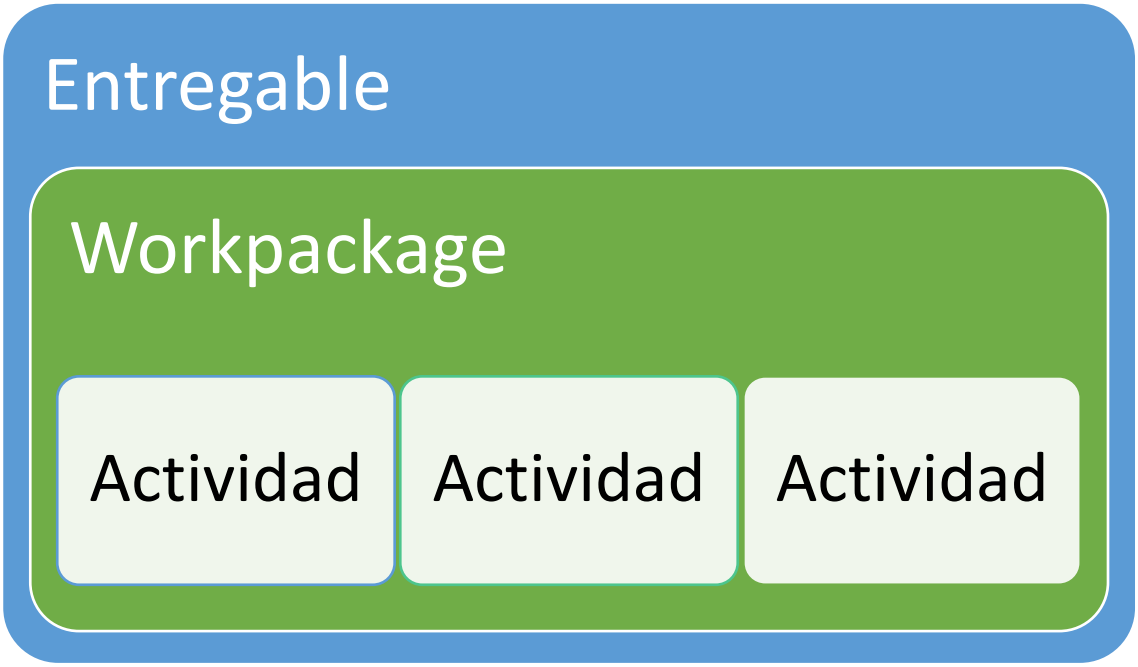
El nivel de granularidad más bajo de la WBS para el que se puede estimar y gestionar una duración y un coste.

Características:

- Es el nivel más bajo de entregable
- Proporciona un valor tangible a los Stakeholders del proyecto
    - Es requerido, reconocido y valorado

Definición de actividad

Es parte del trabajo necesario para conseguir un workpackage



Bloque 2: Riesgos

Definición de Riesgos

Sommerville: obstáculo

PMI: obstáculo u oportunidad

Razones por las que debemos gestionar los riesgos

Conseguir que el proyecto se complete

Impacto económico

Imagen profesional

Nuestra responsabilidad: identificar, estimar y comunicar al cliente

Definiciones: Riesgo, incertidumbre

Riesgo. Efecto de la incertidumbre sobre los objetivos.

Incertidumbre. Falta de conocimiento seguro sobre algún aspecto de la realidad.

Gestión de riesgos – Objetivos

Identificar – Clasificar – Planificar

Clasificación de riesgos

Dos dimensiones:

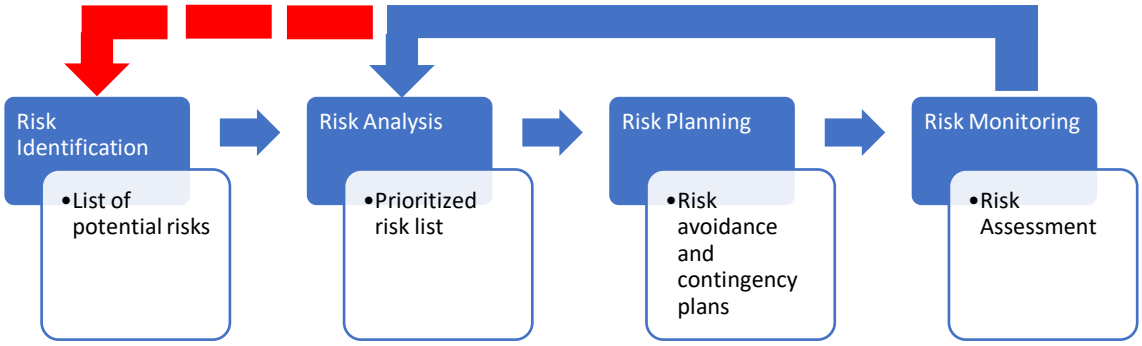
- Tipo de riesgo. El origen de la incertidumbre
- A qué afecta el riesgo. El objetivo comprometido

Riesgos de proyecto. Afectan al plan temporal del proyecto o a los recursos

Riesgos del producto. Afectan a la calidad o al rendimiento del software que se desarrolla

Riesgos de negocio. Afectan a la organización que desarrolla o proporciona el software

El proceso de gestión de riesgos (Sommerville)



Identificación de riesgos (Sommerville)

Puede ser una actividad del equipo o basarse en la experiencia del jefe de proyecto

Inputs a la gestión de riesgos

(Páginas 14 – 15 – 16, Gestión de proyectos – riesgos_2223.pptx)

Herramientas y técnicas para identificar riesgos

Revisión de documentación

Técnicas para recoger información

- Brainstorming
- SWOT/DAFO
- Root-cause Analysis
- Delphi

Brainstorming – Six Thinking Hats

Metodología desarrollada por Edward De Bono para conseguir reuniones o sesiones de Brainstorming más eficiente

SWOT/DAFO

Fortalezas – Oportunidades – Debilidades – Amenazas

Análisis de riesgos

PMP®: Junto con la probabilidad y el impacto es útil especificar en qué fase del proyecto se espera el riesgo.

Análisis de riesgos cualitativo

Se usa para revisar los riesgos identificados inicialmente asignándoles una estimación de impacto y probabilidad.

Nos permite clasificar los riesgos según el producto de ambas estimaciones.

Esta técnica nos permite identificar qué riesgos requieren una atención más inmediata y cuáles pueden posponerse.

Tiene un fuerte componente subjetivo.

Riesgos – Planificación de respuesta (Sommerville)

Considerar cada riesgo y desarrollar una estrategia para gestionarlo

- Evitándolo
- Minimizándolo
- Diseñando un plan de contingencia

Registro de riesgos

Es un documento en el que se recopilan cada uno de los riesgos identificados con claridad y detalle suficiente.

Análisis de riesgo cuantitativo

Objetivos:

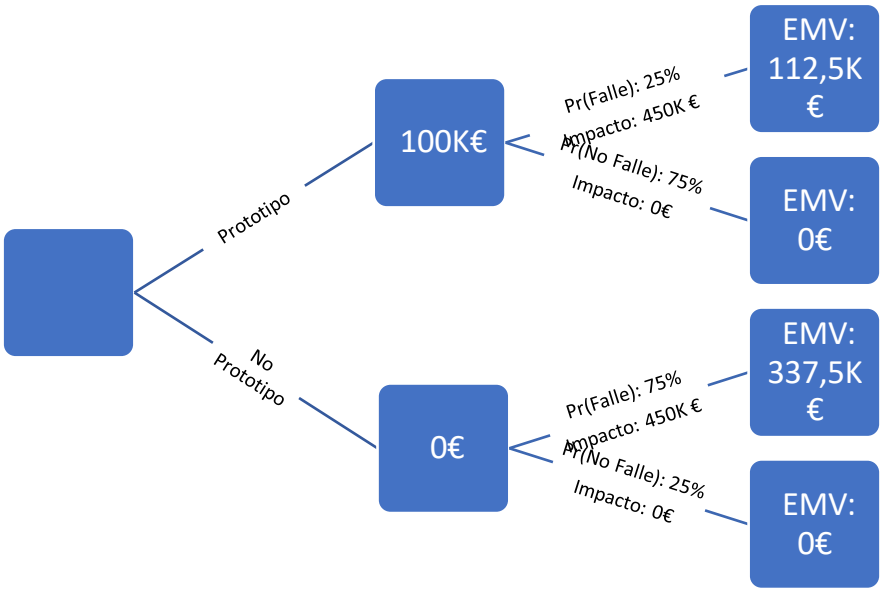
- Determinar con mejor criterio qué riesgos requieren una respuesta
- Determinar el riesgo global del proyecto
- Cuantificar la probabilidad de cumplir con los objetivos del proyecto
- Determinar las reservas de coste y de plazo
- Identificar los riesgos que requieren nuestra mayor atención
- Crear objetivos relistas y alcanzables (alcance, coste y tiempo)

Es bastante costoso y requiere mayor competencia y herramientas más complejas, por lo que su uso, salvo en proyectos de gran envergadura, es opcional.

Expected Monetary Value

Es el producto de la estimación cuantitativa de la probabilidad por la est. Cuantitativa del impacto.

Árboles de decisión



Con prototipo: 100K € + 112,5k € = 212,5K €

Sin prototipo: 0€ + 337,5K € = 337,5K €

Decisión: Incluir prototipo

Planificar respuestas a los riesgos

Objetivo: eliminar o reducir el impacto de los riesgos y promover o aumentar el impacto de las oportunidades

Si no se puede eliminar un riesgo, aplicamos un plan de contingencia y si este no funciona, aplicamos planes de respaldo

Estrategia de respuesta a los riesgos

Al realizar una valoración de riesgos hay que hacer un análisis cuidadoso de las posibles respuestas que se puede dar a cada riesgo.

- En ocasiones las respuestas pueden implementarse como cambios (alcance, coste, calidad, etc.)
- Otra estrategia es diseñar un plan de contingencia.

Respuestas posibles a las amenazas:

- Evitar
- Mitigar (la probabilidad y el impacto se analizan de forma separada)
- Transferir

Estrategia de respuesta a las oportunidades

- Explotar. Realizar modificaciones para que se materialice.
- Promover. Aumenten la probabilidad y/o el impacto positivo de las oportunidades
- Compartir.

Aceptar el riesgo (esto se debe comunicar a los stakeholders):

- Activamente. Plan de contingencia + asignar costes y plazos de reserva
- Pasivamente. Se postergan las decisiones y las acciones.

Comentarios finales a los planos de respuesta

Estrategias pertinentes y oportunas

Esfuerzo proporcional (que no nos cueste más evitar el riesgo que padecerlo)

Una respuesta puede dar solución a más de un riesgo

Un riesgo puede requerir de más de una respuesta

Se debe involucrar a los stakeholders

$$\frac{(EMV_{antes} - EMV_{después})}{Coste\ de\ la\ acción\ reductora}$$

Resumen esquemático del análisis de riesgos

(Página 49 – 50. Gestión de proyectos – riesgos_2223.pptx)

Supervisión de riesgos (Sommerville)

Durante la vida del proyecto hay que revisar y valorar todos los riesgos de forma periódica para ver si ha variado su probabilidad o su impacto.

Los riesgos clave deben discutirse en las reuniones de seguimiento de proyecto.

Control de riesgos

Una vez en marcha el proyecto, debemos seguir supervisando y gestionando los riesgos. Entre otras debemos considerar estas actividades:

- Identificar los triggers de riesgos.
- Supervisar los riesgos residuales.
- Identificar nuevos riesgos.
- Evaluar el plan actual.
- Recopilar y comunicar información a los stakeholders.
- Asegurarse que los procesos se están aplicando.
- Revisar si se ha modificado la probabilidad/impacto.

Bloque 3: Scrum

¿Cómo definen Scrum sus creadores?

Un marco de trabajo con el que las personas pueden acometer problemas complejos adaptativos, entregando productiva y creativamente productos del máximo valor posible.

Fundamentos

Enfoque iterativo e incremental.

- Transparencia
- Inspección
- Adaptación

Transparencia

Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado.

Inspección

Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el proceso hacia un objetivo, para detectar variaciones. (no deben interferir en el trabajo)

Adaptación

Si un inspector determina que uno o más aspectos de un proceso se desvían de los limite aceptables, y que el producto resultante no será aceptable, el proceso o el material que está siendo procesado deben ser ajustados.

Scrum Team

Compuesto por:

- Product Owner
- Development Team
- Scrum Master

Son polivalentes y eligen la mejor forma de llevar a cabo su trabajo.

Product Owner

Responsable de maximizar el valor del producto y del trabajo del equipo, y de gestionar el producto backlog.

Es una única persona. Toda la organización debe respetar sus decisiones.

No está permitido que nadie pida al equipo que trabaje en base a un conjunto diferente de requisitos.

Development Team

El equipo está formado por los profesionales encargados de desarrollar incrementos del producto “Terminados”, que potencialmente se puedan poner en producción, al final de cada Sprint.

Características

- Auto organizados
- Polivalentes (cross-functional)

- Scrum no asigna títulos
- Scrum no reconoce sub-equipos
- La responsabilidad es compartida por todo el equipo

Tamaño

Entre 3 y 9 miembros

El Product Owner y el Scrum Master no cuentan en el cálculo excepto que realicen trabajo del sprint (Sprint Backlog)

Scrum Master

Responsable de asegurar la eficacia de Scrum, asegurándose de que el equipo trabaja ajustándose a la teoría, prácticas y reglas de Scrum.

Servant leader

Ayuda a las personas externas al equipo a entender qué interacciones con el equipo pueden ser de ayuda y cuáles no.

Maximizar el valor creado por el equipo.

Cómo ayuda el Scrum Master al Product Owner

- Encontrando técnicas y métodos para gestionar el product backlog de manera efectiva
- Asegurando que el Product Owner sabe cómo ordenar el producto backlog para maximizar el valor
- Ayudando al equipo a entender la necesidad de contar con elementos del product backlog claros y concisos
- Comprendiendo la planificación del producto
- Comprendiendo y practicando la agilidad
- Facilitando los eventos Scrum

Cómo ayuda el Scrum Master al Development Team

- Ayudándole a organizarse
- Ayudándole a crear productos de alto valor
- Eliminando barreras
- Facilitando los eventos Scrum
- Guiándolo en el entorno de organizaciones en las que Scrum aún no ha sido adoptado y entendido por completo.

Cómo ayuda el Scrum Master a la organización

- Liderando y guiando a la organización en la adopción de Scrum
- Planificando las implementaciones de Scrum en la organización
- Ayudando a los empleados e interesados a entender y llevar a cabo Scrum y el desarrollo empírico de producto
- Promoviendo cambios que incrementen la productividad del equipo
- Trabajando con otros Scrum Master para incrementar la efectividad de Scrum en la organización

Eventos de Scrum

Scrum prescribe una serie de eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones ajenas al marco de trabajo que propone

Todos los eventos están acotados en el tiempo y tienen una duración máxima

Un Sprint tiene duración es fija, por lo que no puede acotarse o alargarse

Los demás eventos pueden terminar siempre que se alcance el objetivo del evento sin permitir desperdicio en el proceso

Cada uno de los eventos de Scrum proporciona una oportunidad formal para la inspección y adaptación, y posibilitar la transparencia y la inspección

La falta de alguno de estos eventos conduce a una reducción de la transparencia y constituye una oportunidad perdida para inspeccionar y adaptarse

Sprint

Núcleo de Scrum

Bloque de tiempo de un mes como máximo

La duración de los Sprints sea consistente

Incluyen:

- Sprint Planning Meeting
- Daily Scrums
- Trabajo de desarrollo
- Sprint Review
- Sprint Retrospective

Durante el Sprint:

- No se realizan cambios que puedan afectar al Sprint Goal
- No se reducen los objetivos de calidad
- El alcance puede ser clarificado y renegociado

Horizonte no mayor a un mes

Definición – Diseño y Flexibilidad – Trabajo y Producto resultante

Puede ser cancelado antes de que concluya el periodo asignado, solo el Product Owner tiene la autoridad para cancelar el Sprint, se cancela si el objetivo queda obsoleto.

Sprint Planning Meeting

Se planifica lo que se va a hacer en el Sprint

Colabora todo el equipo Scrum

Máximo 8 horas

¿Qué podemos entregar en el incremento resultante de este Sprint?

¿Cómo conseguiremos hacer el trabajo necesario para entregar el incremento?

Al finalizar la reunión de planificación del Sprint, el Development Team debe ser capaz de explicar al Product Owner y al Scrum Master como pretende trabajar para lograr el objetivo del Sprint y crear el incremento esperado.

Sprint Goal

Es una meta establecida durante la reunión de planificación e identifica qué puede alcanzarse mediante la implementación del producto backlog

Daily Scrum

Development Team -> 15 minutos

¿Qué hice ayer para lograr el objetivo del Sprint?

¿Qué haré hoy para lograr el objetivo del Sprint?

¿Observo algún impedimento que obstaculice que el Development Team o yo logremos el objetivo del Sprint?

Beneficios:

- Mejora la comunicación
- Reduce la necesidad de mantener otras reuniones
- En él se identifican y eliminan impedimentos relativos al desarrollo
- Favorece la toma rápida de decisiones
- Mejora el nivel de conocimiento del equipo

Evaluar el progreso y qué tendencia sigue este progreso en relación con el Sprint backlog

Sprint Review

Revisión del incremento resultante

Adaptar el product backlog si fuese necesario

Máximo 4 horas

Scrum Team + Stakeholders

Sprint Retrospective

Una oportunidad para que el equipo inspeccione a sí mismo y cree un plan de mejoras para el siguiente sprint

Tiene lugar después de la revisión del Sprint y antes del siguiente Sprint Planning Meeting

3 horas para Sprint de un Sprint mes o 45 – 50 minutos por semana para Sprint más cortos

Artefactos Scrum

Representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Y están diseñados específicamente para maximizar la transparencia de la información clave

Product Backlog

Lista ordenada de todo lo que el producto puede contener

Única fuente de requisitos

Product Owner es el responsable

Es un artefacto dinámico

A menudo, varios equipos Scrum trabajan sobre el mismo producto, pero se utiliza un solo product backlog

El refinamiento (refinement o grooming) del product backlog hace referencia a la adición de detalles, estimaciones y orden a los elementos del product backlog

El equipo Scrum decide cómo y cuándo se hace el refinamiento. No debe consumir más del 10% de la capacidad del Development Team

Los elementos prioritarios deben describirse de forma más clara y detallada

El Development Team es el responsable de proporcionar todas las estimaciones

Sprint Backlog

El conjunto de elementos del product backlog seleccionados para el Sprint, junto con un plan para entregar el incremento del producto y conseguir el objetivo del Sprint

Hace visible todo el trabajo que el Development Team identifica como necesario para alcanzar el objetivo del Sprint

Plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Daily Scrum

Proporciona una imagen visible del trabajo que el Development Team planea llevar a cabo durante el Sprint

Incremento

La suma de todos los elementos del producto backlog completados durante el Sprint

Definición de terminado (Definition of Done)

(Página 62 – 63 – 64. ISII – Scrum v5.pptx)

Bloque 4: Gestión de Personas

Una mala gestión del personal suele ser uno de los factores principales en el fracaso de los proyectos

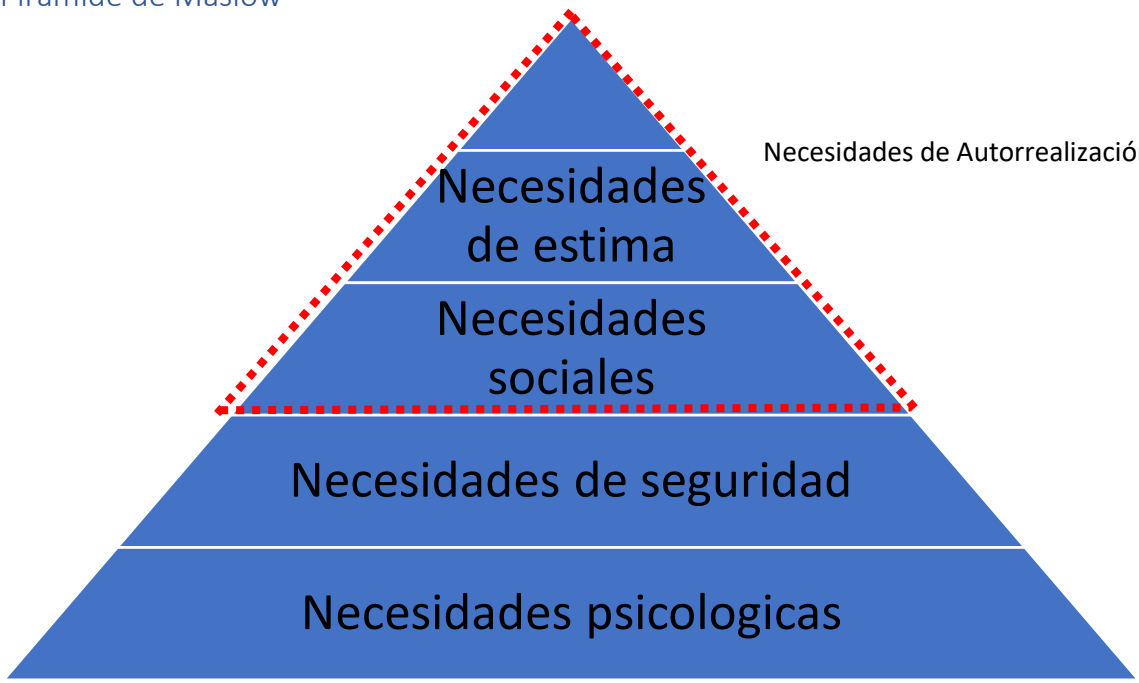
Factores

- Consistencia
  - Tratar por igual a todos miembros del equipo
- Respeto
  - Las habilidades diferentes deben ser respetadas y valoradas
- Inclusión
  - Todas las opiniones se consideran
- Honestidad
  - Señalando lo que va bien y lo que va mal

Motivación

Es tarea del jefe de proyecto el organizar el trabajo y el entorno de modo que anime a la gente a trabajar de forma efectiva. Es una tarea compleja en la que tenemos que considerar distintos niveles.

Pirámide de Maslow



Satisfacción de necesidades

En el ámbito del desarrollo de software, las necesidades fisiológicas y de seguridad básicas no suelen ser un problema

- Social
  - Áreas donde se facilite la interacción
  - Canales de comunicación informal
- Estima
  - Reconocimiento de logros
  - Recompensas

- Autorrealización
    - Promover la responsabilidad
    - Promover y facilitar la formación continua

Tipos de personalidad

- Gente orientada a la tarea
- Gente orientada a la interacción
- Gente orientada a sí misma

Trabajo en equipo

La ingeniería de software exige trabajo en equipo

- Cohesión – sentimiento de pertenencia
- Interacción
- Disponibilidad

Un buen equipo está cohesionado y participa del sentimiento de pertenencia

Asegurar una buena interacción es clave para su rendimiento

La realidad se impone. Conseguir los mejores resultados posibles con el equipo que se dispone

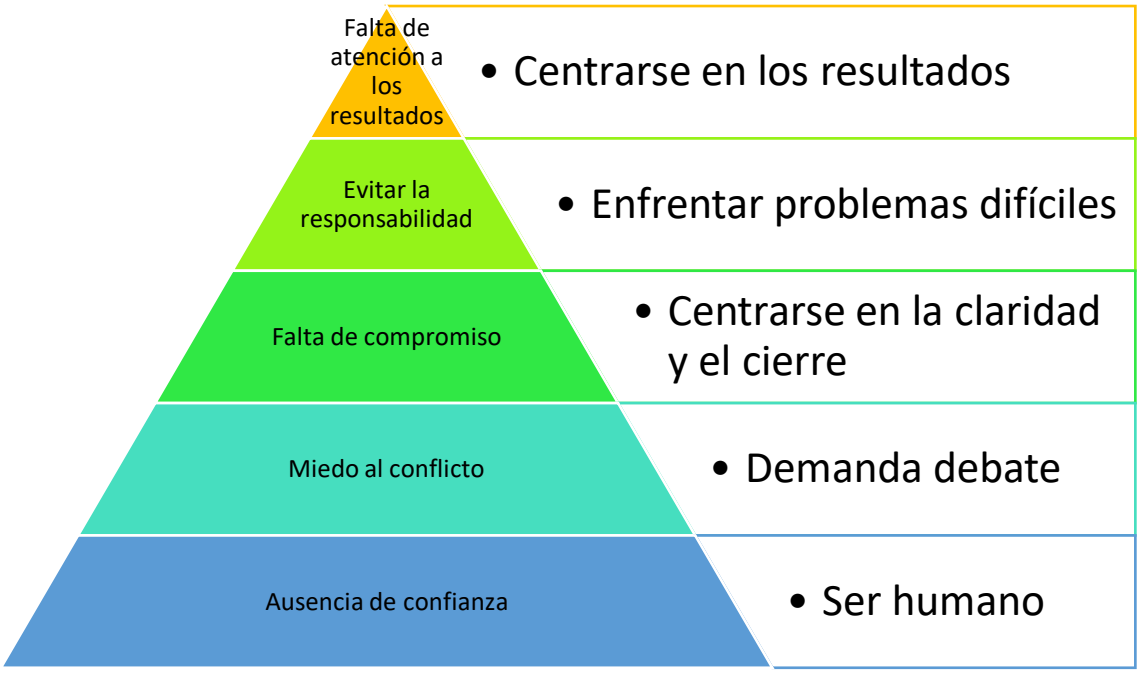
Patrick Lencioni – Rasgos para ser un buen colaborador

Ambición + Humildad = El que “la monta” accidentalmente

Ambición + Inteligencia Emocional = El “político” hábil

Humildad + Inteligencia Emocional = El holgazán adorable

Patrick Lencioni – Las 5 “inadecuaciones” de un equipo



Cohesión

Más importancia al bien del grupo que al bien individual

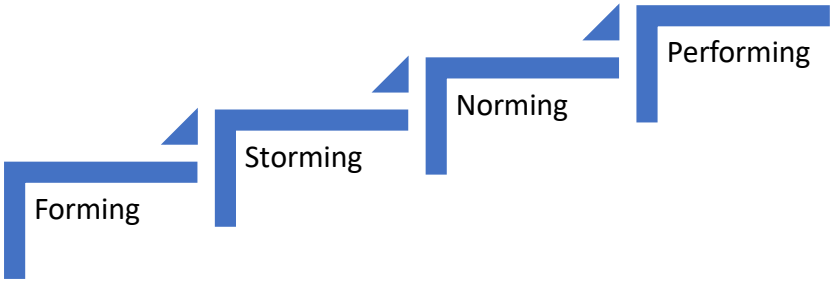
Ventajas

- Aprendizaje mutuo
- Desarrollo de estándares de calidad del grupo
- Continuidad del equipo
- Facilita la mejora continua y el refactoring

La efectividad en el equipo

- La gente. Encontrar la mezcla adecuada de perfiles
- La organización del grupo
- La gestión de las comunicaciones

Modelo de Tuckman



Organización del equipo

Aspectos claves que hay que considerar

- ¿Debe ser el jefe de proyecto quien lidere técnicamente el equipo?
- ¿Quiénes estarán involucrados en las decisiones críticas y de qué forma se tomarán éstas?
- ¿Cómo se van a gestionar las comunicaciones con la dirección y los stakeholders?
- ¿Cómo integrar a la gente que no está coubicada?
- ¿Cómo se puede compartir el conocimiento?

El desarrollo Agile considera que la estructura formal impide el intercambio de información.

Comunicación

Una buena comunicación es esencial

Establecer las reglas del juego

- Honestidad
- Señalar cómo deben resolverse las situaciones de conflicto
- Señalar cuando hay que alertar al jefe de proyecto
- ¿Se puede interrumpir durante una reunión?
- Las formas aceptables de interrupción
- Cómo gestionar las reuniones
- ¿Permitido llegar tarde a las reuniones?
- ¿Se permite que la gente siga atenta a sus móviles o su correo?
- ¿Quién puede comunicarse con dirección o los proveedores?
- Periodicidad del feedback al jefe de proyecto
- Coordinación y cambios en el calendario

Gestión de conflictos – Perspectiva

(Página 24. Gestión de proyectos – Personal_2022.pptx)

Gestión de conflictos

Los conflictos son inevitables

El jefe de proyecto es responsable de evitar y minimizar los conflictos

Fuentes más habituales de conflicto

1. Planificaciones temporales
2. Las prioridades del proyecto
3. Diferencias relacionadas con la asignación de recursos
4. Opiniones técnicas
5. Procedimientos administrativos
6. Costes
7. Personalidad

Técnicas/modos de resolución de conflictos

- Colaboración/intentar resolver el problema
- Acuerdo mutuo
- Abstención/evitar el problema
- Acomodación

- Obligación

Factores que influyen en la comunicación

- Tamaño del equipo
- Estructura
- Composición
- El lugar de trabajo

Matriz RACI (Responsible, Accountable, Consult, and Inform)

- Responsible. Responsable de ejecutar toda o parte de la actividad
- Accountable. Responsable, pero de forma aumentada. Es a la persona que se le pedirá explicaciones en caso de problemas
- Consult. Identifica a miembros del equipo o stakeholders a los que podemos acudir para aclaraciones
- Inform. Identifica a miembros del equipo a los que debemos informar del progreso/incidencias de la actividad

Organizational Breakdown Structure (OBS) Diagrama de descomposición de la organización

(Página 32. Gestión de proyectos – Personal_2022.pptx)

Bloque 5: Planificación de proyectos

Las organizaciones necesitan estimar lo que les costará tanto económicamente como en tiempo desarrollar o poner en marcha un sistema software

Técnicas basadas en la experiencia

Se basan en la posibilidad de reutilizar la experiencia adquirida en proyectos previos

Es útil contar con otros miembros que nos ayuden a comprender el detalle y las razones

Problemas

Los proyectos nuevos no tienen por qué tener mucho en común con los previos

La velocidad con que cambian las tecnologías en el mundo del software

Modelado algorítmico de costes

El coste se calcula como una función matemática que incluye atributos de producto, proyecto y procesos, cuyos valores son estimados por jefes de proyecto de la organización

$$Esfuerzo = A * Tamaño^{B} * M$$

A: constante dependiente de la organización

B: indicativo de que mayor envergadura tienen afectan exponencialmente al coste

M: factor compuesto de atributos de producto, proyecto y equipo

Precisión de las estimaciones algorítmicas

Las estimaciones de los factores que contribuyen a la definición de B y M son en gran medida subjetivas y varían de acuerdo con el criterio de quien realiza las estimaciones

Estimación PMP®

Técnicas de estimación

- Estimación sencilla (un punto)
- Estimación por analogía
- Estimación paramétrica
- Estimación de tres puntos
- Estimación por grupos (Delphi)
- Análisis de reserva

Estimación sencilla

En este caso quien estima da una única estimación por actividad

Puede basarse en la experiencia o ser simplemente una suposición

Puede ser perfectamente válida para tareas pequeñas y bien conocidas

Problemas: Proclive al padding y oculta riesgos e incertidumbres

Estimación por analogía

Se puede aplicar tanto a la estimación de tiempos como de costes

Basada en el conocimiento

Puede aplicarse a nivel global (útil en la fase de inicio)

A nivel de actividades si se dispone de información histórica que respalde la solidez de la estimación.

Estimación paramétrica

Presta atención a la relación existente entre las variables asociadas a una actividad

El origen de los datos puede venir de proyectos anteriores, requisitos industriales, estándares o de otras fuentes

Basándonos en información histórica podemos crear estimaciones paramétricas

Estimación de tres puntos

La estadística muestra que es muy improbable que un proyecto pueda completarse en una fecha o con un coste exacto

- Optimista/mínimo (O)
- Probable (M, *Most Likely*)
- Pesimista/máximo (P)

Dos variantes:

- Triangular: $\frac{O+P+M}{3}$
- Distribución Beta:
  - $Duración\ esperada = \frac{O+4*M+P}{6}$
  - $Desviación\ estándar = \frac{P-O}{6}$

Estimación por grupos

(Esta no es muy importante, con mirarlo en la propia presentación)

(Página 18. Est-plan-calendar.pptx)

Cómo deben realizarse las estimaciones

Las estimaciones deben realizarlas las personas que van a ejecutar el trabajo o aquellos que estén más familiarizados con ese tipo de tareas

Evitar el padding

El papel del jefe de proyecto:

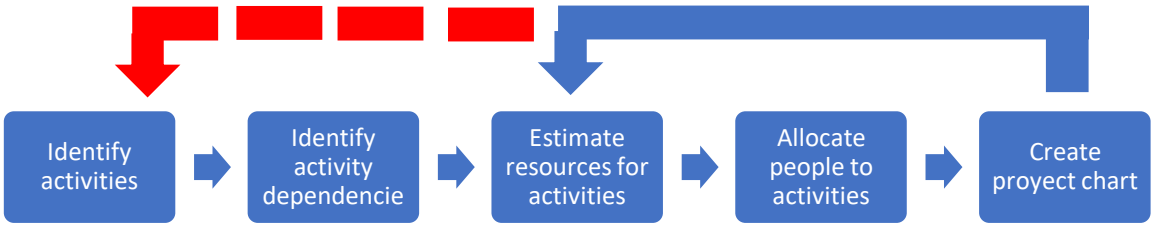
- Proporciona información suficiente
- Detalle que se espera
- Sanity-check
- Calcular las reservas
- Hay que asegurar que las hipótesis realizadas se documentan para su posterior consulta o revisión

Padding (acolchamiento)

Es un tiempo o coste adicional que se añade a la estimación porque el estimador no dispone de suficiente información

(Del resto del tema, leerlo y entenderlo directamente de la presentación)

El proceso de calendarización

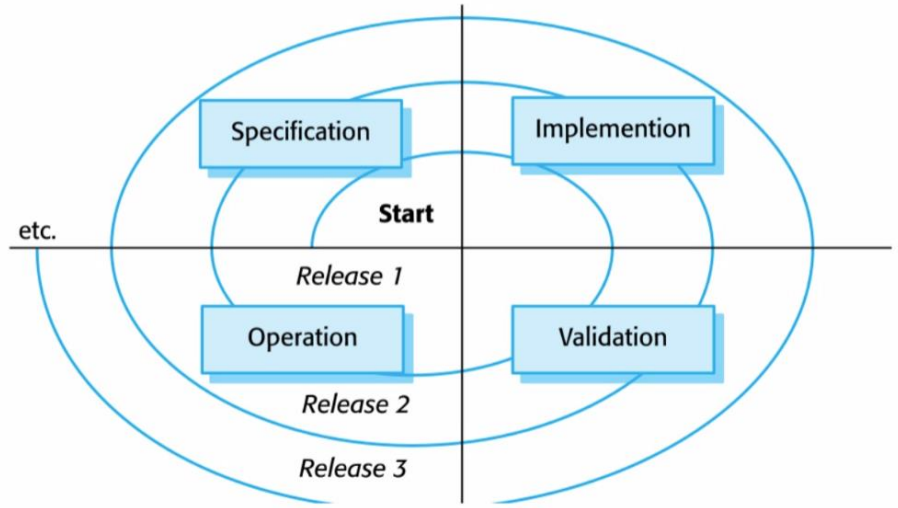


Bloque 6: Evolución del software

En los productos software el cambio es inevitable porque:

- Surgen nuevos requisitos
- El entorno de negocio cambia
- Aparecen fallos
- El sistema en el que se integra crece
- El rendimiento o la fiabilidad necesitan mejorar

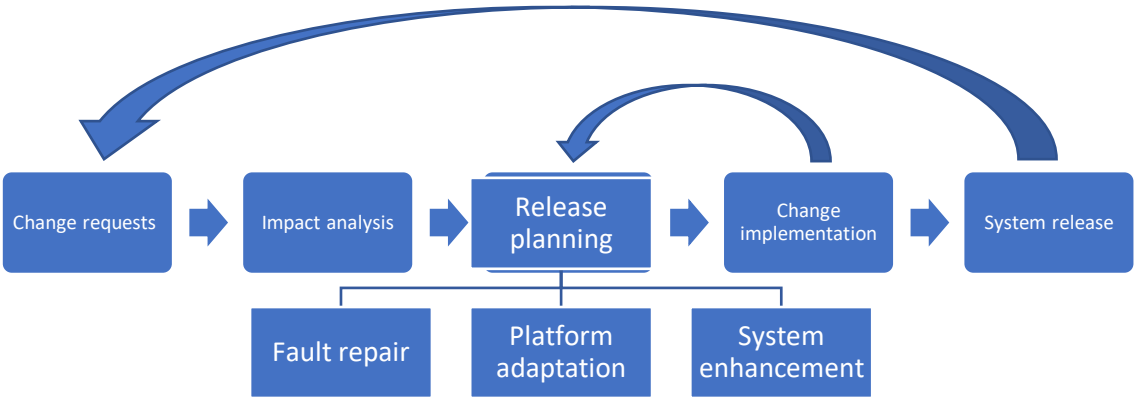
Modelo en espiral del desarrollo y evolución del SW



Visión alternativa: Evolución y mantenimiento

1. Evolución: Etapa durante la cual el software está disponible comercialmente, sigue evolucionando.
2. Mantenimiento: En esta etapa sólo se modifica el código para eliminar errores.
3. Phase-out: El cliente puede decidir seguir usando el software, pero no dispone de ningún tipo de soporte técnico para el mismo

El proceso de evolución del software



Solicitudes de cambio

Upgrade VS Update

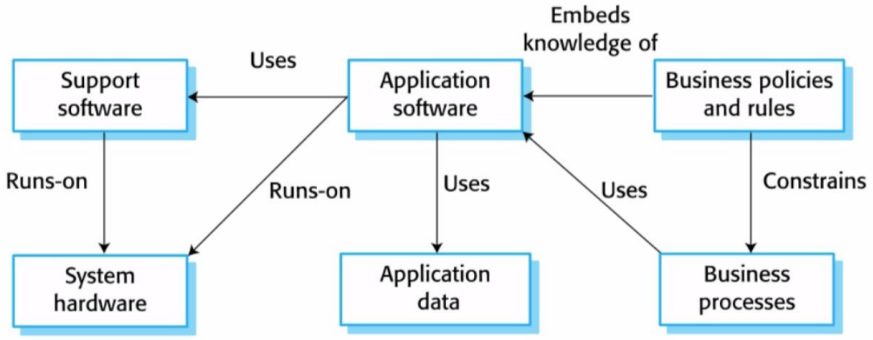
- **Upgrade:** Término que para muchos proveedores está asociado a un cambio de versión del producto
- **Update:** Una actualización del software "menor" que no implica cambio de reléase

El proceso de reparaciones de emergencia

¿Qué es un sistema legacy?

Sistemas antiguos, basados en lenguajes o tecnología que ha dejado de utilizarse en el desarrollo de nuevos sistemas

¿Qué aspectos son relevantes para comprenderlo?



(Página 24 – 25. Evolución de Software2022.pptx)

Gestión de sistemas legacy

- Descartar por completo el sistema existente
- Continuar manteniéndolo
- Transformarlo mediante una reingeniería que mejore sus posibilidades de mantenimiento
- Sustituir la totalidad o parte del sistema con uno nuevo

Aspectos que se han de considerar en el análisis

- Uso del sistema
    - En caso de que el sistema se use ocasionalmente o por un número reducido de usuarios puede estimarse que su valor de negocio es reducido
    - OJO, hay sistemas que se usan ocasionalmente, pero tienen un gran valor
- Los procesos de negocio a los que da soporte el sistema
    - Cuando se introduce un sistema, suele ir acompañado de nuevos procesos de negocio orientados a explotar sus capacidades
    - Un sistema puede tener un valor de negocio reducido si fuerza a que los procesos de negocio sean ineficientes
- La fiabilidad del sistema
    - Si el sistema no es fiable y los problemas afectan a los clientes de negocio, el sistema tiene un valor reducido de negocio
- La información generada por el sistema
    - Si el negocio depende de la información generada por el sistema, entonces el sistema tiene un elevado valor de negocio

Análisis del valor de negocio

Un análisis debe realizarse teniendo en cuenta varios puntos de vista:

- Los usuarios finales
- Los usuarios de negocio
- Los jefes de línea
- Los responsables de IT
- Directores
- Siempre que sea posible hay que entrevistar a todos los actores

Mantenimiento del software (Sommerville)

Modificación del software una vez que se ha puesto a disposición de sus usuarios

Las modificaciones se implementan cambios uno o más componentes existentes

Leyes de Lehman y Belady

(Página 50 – 51. Evolución de Software2022.pptx)

Tipos de mantenimiento (Sommerville)

- Reparación de fallos (eliminar errores)
- Adaptación al entorno (adaptar el software a un entorno operativo)
- Adición y modificación de funcionalidad (satisfacer nuevos requisitos)

Reingeniería de Software

Para mantener el software hay que conoces el programa que se ha de modificar antes de realizar los cambios

Muchos sistemas son difíciles de comprender y por tanto de modificar

Una solución a este problema es realizar una reingeniería del sistema con el fin de mejorar su estructura y su comprensibilidad

Consiste en reestructurar o reescribir una parte o la totalidad del sistema sin modificar su funcionalidad

Aplicable cuando algunos de los subsistemas requieren un mantenimiento frecuente

Esfuerzo añadido para que el software sea más fácil de mantener

Ventajas

- Reducción del riesgo
- Reducción del coste

Refactoring

En este caso la modificación del código está orientada a frenar la degradación del código a consecuencia de los cambios que se van introduciendo

Se puede considerar una actividad de mantenimiento preventivo dirigida a reducir los problemas de futuros cambios

Mejorar su estructura, reducir su complejidad o facilitar su comprensión

No debe añadirse nueva funcionalidad

Refactoring vs reingeniería

La reingeniería hace uso de herramientas que automatizan el proceso para generar un nuevo sistema

El refactoring se trata de un proceso de mejora continúo considerado durante los procesos de desarrollo y evolución del sistema

Code Smells “Tufillos”

- Aglomeración de datos (data clumping)
- Código duplicado
- Generalidad especulativa
- Métodos demasiado extensos
- Sentencias switch/case
    - Violaciones del principio open/closed
    - Código redundante

Bloque 7: Gestión de calidad

Definición de calidad

Sommerville

Los sistemas deben satisfacer las necesidades de sus usuarios, operando de forma eficiente y fiable y deben entregarse a tiempo y dentro del presupuesto

ITIL

La capacidad que tiene un producto, servicio o proceso de proporcionar el valor esperado

PMBOK®

La medida en que proyecto satisface sus requisitos

Insights

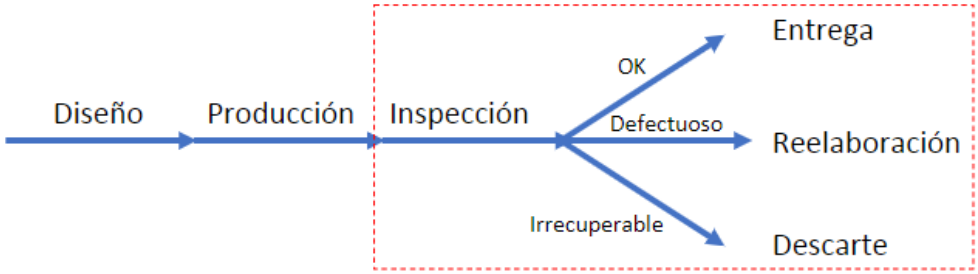
La calidad tiene múltiples dimensiones

Debe definirse para cada producto en base a lo que el cliente espera del mismo mediante características medibles dentro de sus límites de variabilidad (Critical-to-Quality Characteristics)

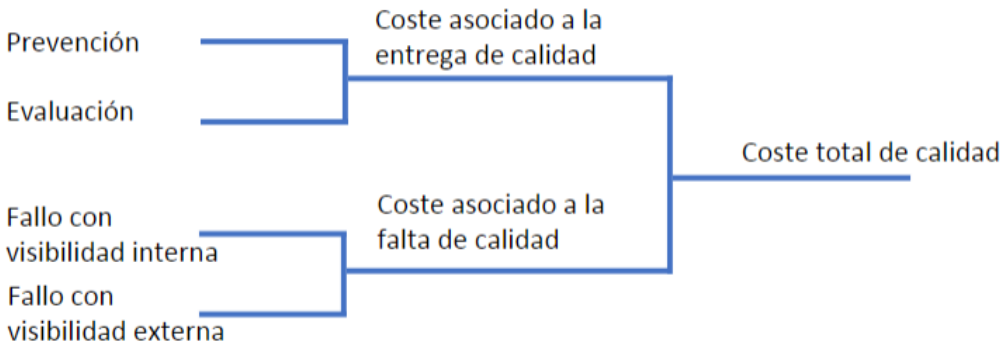
Debe satisfacer las expectativas del cliente

Es siempre relativa a un conjunto de requisitos

Visión "inicial" de la gestión de calidad



Costes de calidad



Gestión de la calidad del software

Aspectos que requieren especial atención

Nivel organizativo

Relacionada con el establecimiento de un marco de procesos

Nivel de proyecto

- Establecer un plan de calidad para el proyecto
- La aplicación y la verificación de los procesos que se han planificado se han seguido

Orígenes y disciplinas

Dos disciplinas

- Aseguramiento de Calidad/Quality Assurance (QA): definición de procesos y estándares, introducción de procesos de calidad
- Control de Calidad/Quality Control (QC): aplicación de los procesos definidos por QA

En la industria del software: conjunto de actividades dirigidos a definir estándares, procesos y procedimientos con el fin de asegurar la calidad del software

Otras van más allá incluyendo la gestión de la configuración actividades de verificación y validación aplicadas al proyecto

Estructura de un plan de calidad

- Introducción al producto: PM
- Planes de producto: PM
- Objetivos de calidad: PM
- Descripciones de procesos: QA
- Riesgos y gestión de riesgos: ED

Aplicabilidad de la gestión de calidad

Sistemas de gran envergadura y complejos

Calidad del software

Medida en que un producto cumple sus especificaciones

En la práctica se admiten ciertas tolerancias

A menudo es imposible saber de forma objetiva si un sistema software cumple con su especificación

- Algunos requisitos de calidad son muy difíciles de especificar de forma inequívoca
- Los requisitos “tensan” el producto en distintas direcciones
- Imposible medir con exactitud ciertas características

¿Cómo medimos la adecuación del software a su propósito?

La valoración de la calidad del software tiene una componente subjetiva reseñable

Adecuación al propósito del software

Se parte de la hipótesis de que el sistema se probará tomando como referencia sus requisitos

Características no funcionales

- La percepción de la calidad depende en gran medida de sus características no funcionales
- Si la funcionalidad no se ajusta a lo esperado, los usuarios buscaran la manera de que el software haga lo que necesitan

Conflictos de calidad

- Imposible implementar un sistema optimizado para todos los atributos
- Conjunto de atributos que son importantes
- Incluir una definición del proceso de valoración (Criterios de aceptación)

Relación entre la calidad de procesos y producto

En el sector industrial, la calidad del producto desarrollado está condicionada por la calidad del proceso de producción

Cultura orientada a la calidad

El establecimiento sensato de procesos de calidad tiene una influencia positiva en la calidad del software

Junto a la definición de procesos, los responsables de calidad deberían:

- Promover una cultura de calidad
- Los equipos se responsabilicen de la calidad de su trabajo
- Apoyar a los miembros del equipo que muestran interés

Estándares de software

Definen los atributos exigibles a un producto a un proceso

¿Por qué son importantes los estándares?

Compendio de buenas prácticas

Proporcionan un marco de referencia para la definición del significado de calidad en un contexto determinado

Facilitan la fluidez y continuidad

Estándares de producto y de procesos

Estándares de producto. Afectan al producto que se desarrolla

- Documentales
- Documentación
- Codificación

Estándares de proceso. Definen los procesos que deben seguirse durante el desarrollo y pueden incluir

Estándares de documentación

Los documentos proporcionan el único medio tangible de representar el software y sus procesos asociados

Tipos:

- Los que definen el proceso

- La estructura y presentación de los documentos
- Formatos electrónicos de publicación

Los problemas de los estándares

Demasiado pesados, que no son relevantes o que no están actualizados

Acciones que favorecen un desarrollo eficaz de estándares

- Involucrar a los desarrolladores
- Revisarlos regularmente
- Adopción de herramientas

Aplicación sensata de los estándares

Deben proporcionar valor en forma de calidad significativa añadida al producto

No tiene sentido aplicar estándares si sólo conducen a mejoras marginales de la calidad

Los responsables de calidad y de proyecto deben acordar al inicio que estándares deben usarse

Los estándares ISO 9001

Es aplicable a organizaciones que diseñan, desarrollan y mantienen productos software

La norma 9001 propone definir la calidad en torno a nueve procesos:

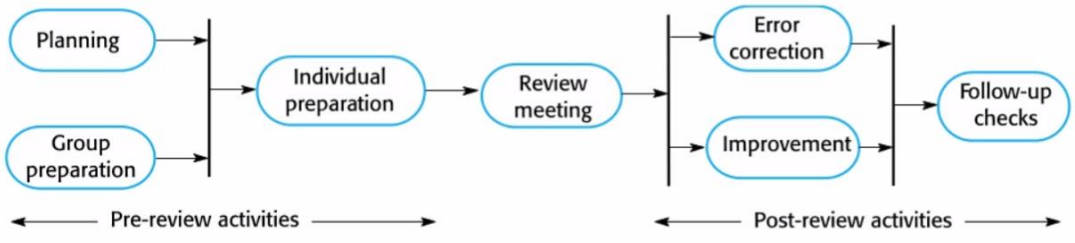
- Cinco, alrededor del producto
- Cuatro, asociados a procesos

La ISO 9001 no es adecuada, define la calidad como conformidad con los estándares que la propia organización se ha dado

Revisiones de calidad

Un grupo distinto al de desarrollo

Fases en el proceso de revisión



3 fases principales:

- Actividades previas
- Revisión
- Actividades posteriores

Roles de los participantes en una inspección

- Autor/Responsable
- Inspector
- Lector
- Moderador
- Secretario
- Moderador jefe

(solo en ambientes muy formales)

Inspecciones de programas

No requieren la ejecución del sistema

Son aplicables a cualquier representación del sistema

Ejemplos de buenas prácticas en Agile

- Antes de registrar el código – verificarlo
- No romper jamás el build
- Arregla los problemas en cuanto los detectes

Programación por parejas

Dos personas se responsabilizan del desarrollo de una parte del código y se va examinando y revisando constantemente por un compañero

Problemas

- Malinterpretaciones compartidas
- Mantener la reputación del binomio
- Las relaciones de trabajo

Medición y métricas del software

(Pág. 63 – 73. Gestión de calidad_2022.pptx)

(Básicamente, lo de SMART de calidad)

Métricas del producto

(Pág. 74 – 79. Gestión de calidad_2022.pptx)

Análisis de componentes software

(Leer)

Software Analytics – Definición Zhang

La utilización de técnicas basadas en datos que permiten la exploración y análisis con el fin de obtener información esclarecedora (que tenga sentido y sea útil para poder ejecutar la tarea objetivo) y accionable u operativa (que permita a los profesionales tomar alguna acción asociada)

Al final del tema Gestión de calidad_2022.pptx hay un resumen, leerlo

Bloque 8: Pruebas de software

¿Por qué son importantes las pruebas de software?

(Pág. 4 – 8. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

(Esto no importa una mierda, la verdad)

Objetivo de la prueba de programas

Mostrar que el programa hace lo que se pretende y descubrir defectos en el código

Prueba de programas (Program Testing)

La verificación nos ayuda a revelar la presencia de errores, no nos información concluyente

La prueba de programa es parte de un proceso de verificación más general que también incluye técnicas de validación estática

Objetivos de la validación de software

Demostrar que el software cumple con los requisitos acordados

Sirven para descubrir situaciones en las que el comportamiento del software es incorrecto, indeseable o no conforme a la especificación

Test de validación

Los test se definen para modelar el uso esperado del sistema y el sistema se comporta conforme a la especificación

Verificación de defectos

Los test se definen intentando generar situaciones que provoquen la aparición de fallos y, de estar bien definido, mostrará que el resultado es erróneo y lo gestionará mediante una excepción

Verificación vs Validación

Verificación:

- ¿Estamos construyendo correctamente el producto?

Validación:

- ¿Estamos construyendo el producto correcto?

Validación y Verificación -> Confianza en el producto

El objetivo es proporcionar confianza

Inspecciones y pruebas

Verificaciones estáticas – inspecciones o revisiones

Verificaciones dinámicas – pruebas de software

What Do Code Reviewers Look For?

Code reviews should look at:

- **Design**: Is the code well-designed and appropriate for your system?
- **Functionality**: Does the code behave as the author likely intended? Is the way the code behaves good for its users?
- **Complexity**: Could the code be made simpler? Would another developer be able to easily understand and use this code when they come across it in the future?
- **Tests**: Does the code have correct and well-designed automated tests?
- **Naming**: Did the developer choose clear names for variables, classes, methods, etc.?
- **Comments**: Are the comments clear and useful?
- **Style**: Does the code follow our style guides?
- **Documentation**: Did the developer also update relevant documentation?

See **How To Do A Code Review** for more information.

Automatización de pruebas

Por desgracia no todos los test pueden automatizarse

Aplicabilidad de la automatización

Test manuales:

- Exploración de casos de test
- Verificar la usabilidad del sistema
- Test ad – oc (muy específicos y no repetitivos)

Test automáticos:

- Regresión
- Verificar rendimiento
- Verificar la carga que soporta el sistema
- Test funcionales muy repetitivos

¿Qué test o partes de un test puedo automatizar?

- Tareas repetitivas
- Captura de resultados
- Entradas de datos
- Test de responsividad
- Características no funcionales
- Configuración/arranque – cierre de entornos

Beneficios de la automatización de pruebas

- Rapidez en la ejecución
- Más fiable
- Requiere menos personal
- Facilita las pruebas de regresión
- Permite ejecuciones más frecuentes
- No requiere presencialidad

Riesgos asociados a la automatización

- Coste elevado
- Nunca automatización al 100%
- Necesidad de mantener e incorporar el control de versiones
- Dificultad de automatizar los casos asociados a la interfaz de usuario
- Establecer expectativas demasiado elevadas
- Gestionar la complejidad del entorno
- Dejarse cegar por el ahorro de tiempo

Etapas de la verificación

- Pruebas de desarrollo
- Pruebas de versión
- Pruebas de usuario

} Sommerville

- Pruebas de campo

Pruebas de desarrollo

Actividades que realiza el propio equipo que desarrolla el sistema

Analogía de la sopa (Srini Devadas)

- Testing
  - Comparar I/O
- Programación defensiva
  - Especificación
  - Modularidad
  - Verificar las condiciones I/O
- Debugging
  - Analizar porque pasa el error

Tres niveles

- Pruebas unitarias (funcionalidad)
- Pruebas de componente (interfaces)
- Pruebas de sistema (interacciones entre componentes)

Pruebas unitarias

Sentido

- Reducen el tiempo necesario para realizar test funcionales
    - Las pruebas funcionales:
        - Costosas
        - Requieren un conocimiento del sistema de pruebas
        - Depende del tester
- Protección contra la regresión
    - Las pruebas unitarias permiten volver a ejecutar fácil y rápidamente toda la funcionalidad
- "Documentación ejecutable"
    - Proporcionan una explicación del resultado esperado
- Reducir las dependencias

Pruebas de clases de objetos

(Pág. 43 – 44. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

Componentes de las pruebas automatizadas (SAC/CAL)

- Configuración (setup)
- Llamada (call)
- Declaración/aserto (assertion)

Criterios para escribir test unitarios

- Sencillos
- Independientes
- Fichero de pruebas por cada clase
- Si usamos TDD, escribimos antes las pruebas que la codificación de la clase

Recomendaciones

(Pág. 49 – 53. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

¿Cómo elegir casos de prueba?

Estrategias de pruebas

- Partición (Identificar grupos de entradas con características comunes y se procesan de la misma forma)
- Basada en guidelines (Aprovechar experiencia previa que ha quedado documentada en guías)
- Intuición
- Black – box Testing (Explorar los distintos caminos que ofrece la especificación)
- Glass – box Testing (Explorar los distintos caminos que ofrece el código)

Black – box Testing

Sin necesidad de conocer la implementación

Puede realizarse por otras personas distintas a los diseñadores

Glass – box Testing

Se parte de la observación del código implementado

Path – complete

También hay que incluir casos límites

Pruebas de componentes

Se centran en que la interfaz de cada componente se comporte de acuerdo con su especificación

Pruebas, Errores y Guidelines de interfaz

(Pág. 72 – 75. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

"Doubles" – Test doubles

Término genérico para referirse a cualquier objeto que definimos para que simule a otro real con la intención de realizar pruebas

Categorías

- Fake objetcs
- Stubs
- Mocks

Pruebas de sistema

(Pág. 80 – 85. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

Pruebas de versión

(Pág. 92 – 107. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

Pruebas de usuario

(Pág. 108 – 123. 3.1 Sesión - Cap. 8 2023- Pruebas de Software.pptx)

Desarrollo dirigido por pruebas (TDD)

Test – driven Development (TDD)

Las pruebas se escriben antes que el código

Beneficios

- Cobertura del código
- Verificación regresiva
- Eliminación de errores simplificada
- Documentación del sistema

Pruebas de regresión

Tienen el objetivo de comprobar que los cambios no han roto el código que previamente funcionaba.

Bloque 9: Gestión de configuración

Es necesaria porque es muy fácil perder la pista de los cambios y de las versiones de los componentes

Tiene que ver con la definición y aplicación de políticas, procesos y herramientas para la gestión de estos cambios

Actividades que intervienen en la gestión de la configuración

- Control de versiones
- Generación de sistemas
- Gestión de cambios
- Gestión de releases

Gestión de configuración y metodologías ágiles

Los desarrollos Agile serían imposibles de gestionar sin la utilización de herramientas de gestión de configuración

Fases de desarrollo de un producto software

1. Fase de desarrollo
2. Fase de pruebas del sistema
3. Fase de puesta en servicio (reléase)

Terminología gestión de configuración

(Pregunta al 99.9% segura, los puntos bajos, son la forma con la que ha aparecido en los exámenes de otros años)

- Configuration (version) control: el proceso de garantizar que las versiones de los sistemas y componentes se registren y mantengan para que los cambios se gestionen y todas las versiones de los componentes se identifiquen y almacenen durante la vida útil del sistema
- Configuration item: cualquier cosa asociada con un proyecto de software que se ha puesto bajo Configuration control
- Codeline: un conjunto de versiones de un componente de software y otros elementos de configuración de los que depende ese componente
    - Una secuencia de versiones de código (u otros elementos de la configuración) en la que cada versión deriva de una previa
- Baseline: una colección de versiones de componentes que forman un sistema
    - Especifica las versiones de los componentes que se incluyen en el sistema junto con una especificación de las librerías utilizadas, de los ficheros de configuración, etc.
- Branching: la creación de una codeline de código a partir de una versión en una codeline existente
- Mainline: una secuencia de baselines que representan diferentes versiones de un sistema.

(Pág. 11. Gestión de la configuración.pptx)

- Merging
- Release
- Repository
- System building
- Version
- Workspace

Gestión de versiones/Version Management

El proceso dirigido a identificar y controla las distintas versiones de los componentes de software o Configuration ítems

También se ocupa de asegurar que los cambios no interfieren entre sí

Puede considerarse como el proceso de gestionar los codelines y baselines del sistema

Codelines y baselines

(Pág. 14 – 16. Gestión de la configuración.pptx)

Gestión de almacenamiento con deltas

(Pág. 18 – 20. Gestión de la configuración.pptx)

Gestión de almacenamiento con snapshots

(Pág. 21. Gestión de la configuración.pptx)

Sistemas de control de versiones en la actualidad

Identifican, almacenan y controlan el acceso a las diferentes versiones de los componentes

- Centralizado (Ej.: SVN)
- Distribuido (Ej.: Git)

Características clave de los sistemas de control de versiones

- Identificación de versiones
- Registro histórico de cambios
- Soporte para el desarrollo independiente y coordinada
- Soporte para el proyecto
- Gestión del almacenamiento

Control de versiones centralizado

(Pág. 26. Gestión de la configuración.pptx)

¿Cómo trabajar sobre un fichero compartido?

Solución Lock – Modify – Unlock

El repositorio sólo permite que haya una persona modificando un fichero en un momento dado

Básicamente, bloque el fichero al resto

Problemas de la solución Lock – Modify – Unlock

- Puede dar lugar a problemas administrativos
- Puede provocar una secuencia innecesaria del trabajo
- El bloqueo puede generar una falsa sensación de seguridad

Solución Copy – Modify – Merge

El sistema de control de versiones facilita el proceso de combinación

¿Cuándo sigue siendo necesario el bloqueo?

(Pág. 37. Gestión de la configuración.pptx)

Ventajas del control de versiones distribuido

- Proporciona un sistema de back-up del repositorio
- Permite que los desarrolladores trabajen off-line
- La forma de trabajo por defecto es Project support

Desarrollos con código abierto

(Pág. 40 – 41. Gestión de la configuración.pptx)

Branching y Merging

Nos proporcionan mecanismos necesarios, pero no suficientes

- Establecer políticas de colaboración

Mantener siempre una rama con una versión estable y actualizada de nuestro sistema

TEMARIO DE DIS DE GIT ----------

Generación de sistemas software (System building)

Proceso de creación de un sistema ejecutable completo, compilando y enlazando los componentes del sistema, las librerías externas, los ficheros de configuración, etc.

Las herramientas de desarrollo y control de versiones deben estar integradas y comunicarse adecuadamente

Herramientas

(Pág. 58. Gestión de la configuración.pptx)

Funcionalidad requerida para generar sistemas

- Generación de scripts de configuración
- Control de versiones
- Recompilación mínima
- Generación de ejecutables
- Automatización de pruebas
- Informes
- Generación de documentación

Plataformas que intervienen en la generación de builds

- Sistema de desarrollo del programador
- Servidor de builds
- Entorno objetivo

Construcción de sistemas Agile

(Pág. 66. Gestión de la configuración.pptx)

Integración continua

(Pág. 67. Gestión de la configuración.pptx)

Ventajas y desventajas

- Pros
    - Permite que los problemas causados por las interacciones entre los distintos desarrolladores se descubran y se corrijan enseguida
    - El sistema más reciente en la rama principal (Mainline o master) es el sistema operativo definitivo
- Contras
    - Si el sistema es muy complejo, la generación y pruebas del código conlleva un tiempo considerable, particularmente si la aplicación requiere la integración con otros sistemas. En estas condiciones, este enfoque puede no ser práctico
    - Si la plataforma de desarrollo es distinta de la plataforma objetivo, puede no ser factible la ejecución de las pruebas de sistema en el espacio privado de trabajo del desarrollador

Generación diaria

(Pág. 69 – 70. Gestión de la configuración.pptx)

Minimizar la Recompilación

La computación puede suponer el uso de una cantidad importante de recursos de computación (tiempo)

Identificación de ficheros

- Marcas de tiempo
- Checksums del código fuente

Comparación de las marcas de tiempo y los checksums

(Pág. 73 – 74. Gestión de la configuración.pptx)

Gestión de cambios

Hay que asegurar que la evolución del sistema es un proceso gestionado

Actores del proceso de gestión de cambios

- Cliente
- Comercial/preventa local
- Equipo de soporte
- Strategic Product Manager
- Release responsable
- Organización de desarrollo
- Solution Architect
- Project Manager

Factores que debemos considerar en el análisis de cambios

- ¿Cuáles son las consecuencias de no realizar un cambio?
- ¿Cuáles son los beneficios de realizar un cambio?
- ¿Qué cantidad de usuarios/clientes se ven afectados por el cambio?
- ¿Qué coste tiene realizar el cambio?
- ¿Cómo afecta al ciclo de vida del producto?

Gestión de releases

Una release de sistema es una versión del software que se distribuye a los clientes

- Major releases, incluye cambios importantes
- Minor releases

Componentes de release

- Ficheros de configuración
- Ficheros de datos
- Programa de instalación
- Documentación
- Empaquetado o publicidad específica
- Pricing
- Product Packaging and Supply
- Learning and Education

Factores que influyen en la planificación de releases

(Pág. 90. Gestión de la configuración.pptx)

Creación de releases

Consiste en generar todo el conjunto de ficheros y documentación que incluya todos los componentes de la release

Bloque 10: Reutilización y evolución del software

Ingeniería del software basada en la reutilización

- Reutilización de sistemas
- Reutilización de aplicaciones
- Reutilización de componentes
- Reutilización de objetos y funciones

Beneficios de la reutilización del software

(Pág. 7 – 8. Reuttilización software_2022.pptx)

Problemas de la reutilización del software

(Pág. 9 – 10. Reuttilización software_2022.pptx)

Factores para tener en cuenta en la planificación cuando se reutiliza

- Plan de tiempos
- Ciclo de vida del software
- Formación, habilidades y experiencia del equipo
- Lo crítico del software y de sus requisitos no funcionales
- Dominio de aplicación
- Plataforma software en la que se va a ejecutar

Frameworks de aplicación

(Pág. 14 – 27. Reuttilización software_2022.pptx)

Funcionalidades WAF

- Seguridad
- Página web dinámicas
- Soporte de bases de datos
- Gestión de sesiones
- Interacción con el usuario

Extensión de Frameworks

Los frameworks son genéricos y se extienden para crear una aplicación más específica o un subsistema

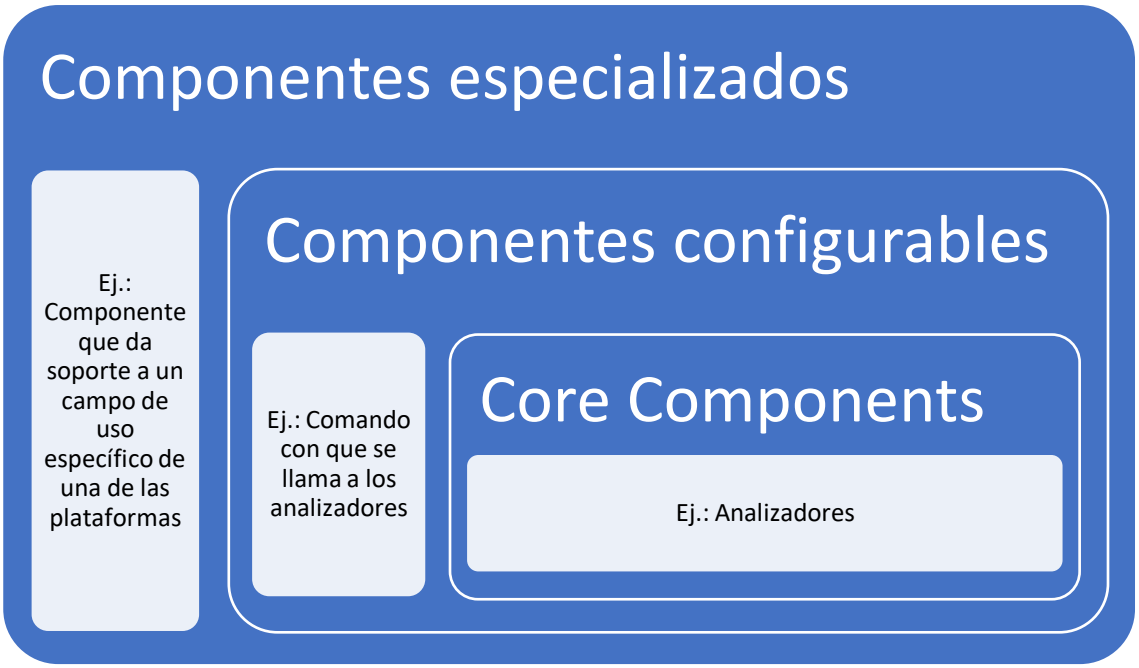
Inversión del control

Cuando usamos un framework, el control de flujo recae sobre el framework que ocasionalmente invoca a nuestro código para conseguir el comportamiento que deseamos

Líneas de producto software

Proporcionan una funcionalidad genérica que puede adaptarse y configurarse para su uso en un contexto específico

Esquema de referencia para la creación de líneas de producto



Comparativa de frameworks de aplicación y líneas de producto

(Pág. 38 – 39. Reuttilización software_2022.pptx)

Especialización de líneas de producto

- Plataforma (se modifica únicamente los componentes que interactúan con el hardware y con el sistema operativo)
- Entorno
- Especialización funcional (adaptarse a requisitos específicos de cada cliente)
- Especialización en base a procesos

Desarrollo de un nuevo miembro dentro de la línea de producto

1. Obtener los requisitos de los stakeholders
2. Selección del sistema existente que se aproxima más a los nuevos requisitos
3. Negociación de requisitos
4. Adaptación del sistema existente
5. Poner el nuevo miembro de la línea en servicio

El resto mirar directamente de las diapositivas

(Pág. 47 – 62. Reuttilización software_2022.pptx)

Licencias de software

¿Qué son?

Son documentos que proporcionan directrices legalmente vinculantes para el uso y la distribución de software

Sus términos y condiciones suelen incluir el uso razonable del software, las limitaciones de responsabilidad, las garantías y las exenciones de responsabilidad

¿Cómo funcionan?

Suelen expresarse como un acuerdo de usuario final (end-user license agreement – EULA) o de empresa (Enterprise license agreement – ELA)

¿Por qué son importantes?

- ¿Cómo protegen los acuerdos de licencia a los desarrolladores?
- ¿Cómo protegen los acuerdos de licencia a los usuarios?

Tipos

- Dominio público
- Permisiva
- Lesser General PL
- Copyleft
- Propietaria

Aspectos regulados

(Pág. 69. Reuttilización software_2022.pptx)

Autoevaluación I

¿Cuál de las siguientes opciones **es la menos adecuada** para valorar el éxito de un proyecto?

A. Conseguir mantener un equipo de desarrollo coherente y eficaz
B. Proporcionar un software que cumpla con las expectativas del cliente
C. Entregar el software al cliente en la fecha acordada
D. Concluir el proyecto utilizado sólo el 60% del presupuesto

Respuesta:

Correcta: D

A, C y D son criterios de éxito de un proyecto. B lo sería si el margen no fuese tan amplio. Habitualmente se considera que un proyecto bien gestionado tolera una variación de costes de +/- 10%.

¿Cuál de los siguientes conceptos asociados a la gestión de riesgos expresa el punto específico en el que un riesgo se vuelve problemático?

A. Tolerancia al riesgo
B. Apetito de riesgo
C. Umbral de riesgo
D. Margen de riesgo

Respuesta:

Correcta: C

El apetito de riesgo es una descripción genérica del nivel que se está dispuesto a aceptar: conservador, moderado, arriesgado.

Un umbral de riesgo expresa el punto específico en el que el riesgo se vuelve problemático. Por ejemplo, el cliente pide que el proyecto esté listo para el 15 de abril o nos indica que dispone de un presupuesto de 500.000€.

La tolerancia al riesgo está relacionada con el apetito y con el umbral. Nos da una medida de hasta qué punto está dispuesto a asumir una desviación respecto al umbral identificado. Usando el ejemplo anterior, una opción sería que el cliente asumiera dos semanas de retraso o una desviación presupuestaria respecto al umbral del 10%, debido a la introducción de cambios que afecten a la incertidumbre del proyecto.

Como comentario adicional, si un cliente no es capaz de especificar umbrales y tolerancias, el equipo de proyecto debe tener esto en cuenta como un riesgo adicional, pues sin referencias concretas aumenta la probabilidad de que la aceptación y cierre del proyecto sean problemáticos.

¿En base a qué dos atributos se suelen clasificar a los stakeholders?

A. Poder y función
B. Poder e interés
C. Seniority y formación técnica
D. Ubicación geográfica e idioma

Respuesta:

Correcta: B

La clasificación habitual se hace obviamente basándonos en el poder y el interés del Stakeholder, para así poder planificar la mejor manera de gestionar sus expectativas en beneficio del proyecto.

Un acta de constitución de proyecto o Project charter es un documento detallado y de un volumen considerable en el que se especifican los requisitos del proyecto y se nombra al jefe de proyecto que lo llevará a cabo

A. Verdadero
B. Falso

Respuesta:

Correcto: B

Es cierto que el acta de constitución es un documento y que en él se asigna al jefe de proyecto. Sin embargo, no se trata de un documento con un alto nivel de detalle o voluminoso. Suele describir los objetivos a alto nivel y puede aportar requisitos que se conozcan en el momento de su redacción, pero éstos no suelen ser muy detallados.

Dentro de los grupos de proceso que define la metodología PMBOK®, ¿cuál es el que consume la mayor parte de los recursos asignados?

A. Cierre
B. Inicio
C. Planificación
D. Monitorización y control
E. Ejecución

Respuesta:

Correcto: E

Aun cuando los porcentajes relativos de esfuerzo pueden variar según los proyectos, lo más normal es que sea el grupo de procesos de ejecución el que consume la mayor parte del esfuerzo y presupuesto del proyecto.

Un programa se crea cuando las interdependencias existentes entre varios proyectos indican que pueden gestionarse con una supervisión conjunta que ayude a minimizar riesgos, explotar sinergias, etc.

A. Falso
B. Verdadero

Respuesta:

Correcto: B

La razón para crear un programa es que haya beneficios económicos o de reducción de riesgos al gestionar los proyectos de forma conjunta y no aisladamente, por lo que la afirmación es correcta.

El ciclo de vida de un producto y de su proyecto asociado están perfectamente alineados en una relación 1:1

A. Verdadero
B. Falso

Respuesta:

Correcto: B

Aun cuando están relacionados, los ciclos de vida y de producto están claramente diferenciados. Lo normal es que haya varios proyectos asociados al producto durante su ciclo de vida, por ejemplo, las fases de desarrollo y mantenimiento se suelen gestionar como proyectos independientes con presupuestos e incluso equipos de desarrolladores independientes.

Según Sommerville, y en el contexto del proceso de gestión de riesgos ¿cuál es el resultado generado en la fase de análisis de riesgos?

A. Una valoración de los riesgos
B. Una estrategia documentada para evitar los riesgos y gestionar las contingencias
C. Una lista de riesgos potenciales
D. Una lista priorizada de riesgos potenciales

Respuesta:

Correcto: D

La respuesta "una lista de riesgos potenciales" no es válida porque ese es el resultado de la fase de identificación. En ella básicamente observamos-registramos riesgos potenciales, pero aún no los cualificamos.

La opción "una valoración de los riesgos" es insuficiente. El resultado de esta fase no es solo la valoración cualitativa de los riesgos, sino que también implica una "traducción numérica" de dicha valoración en términos de probabilidad e impacto de forma que con el producto de ambos atributos podamos ordenar los riesgos y facilitar su gestión.

La alternativa "Una estrategia documentada para evitar los riesgos y gestionar las contingencias" tampoco es válida porque es el resultado de una fase posterior, la de planificación, en la que ya consideramos las respuestas que pensamos que podemos dar a los riesgos.

Por tanto, la respuesta válida es "Una lista priorizada de riesgos potenciales"