Teoría de Ingenieria 2

Teoría 1: Repaso, comunicación y SRS..... 2
- Proceso de software..... 2
- Modelo..... 2
- Comunicación..... 3
- Elicitación de requisitos..... 3
  - Muestreo de la documentación, los formularios y los datos existentes..... 3
  - Observación del ambiente de trabajo..... 3
  - Visitas al sitio..... 4
  - Cuestionarios..... 4
  - Entrevista..... 4
  - JRP..... 5
  - Brainstorming..... 5
- Requerimientos SRS..... 5
  - Requerimientos..... 5
  - ConOps (IEEE Std. 1362-1998)..... 6
  - SRS (standard requirements specifications) IEEE Std. 830-1998..... 6
    - Sección 1 del SRS..... 7
    - Sección 2 del SRS..... 7
    - Sección 3 del SRS..... 7
    - Sección 4 del SRS..... 8
    - Sección 5 del SRS..... 8

Teoría 2: GCS, planificación organizativa..... 8
- Gestión de la configuración de software (GCS)..... 8
  - Cambios..... 8
  - GCS..... 8
  - Involucrados de la GCS..... 9
  - Elementos de la GCS (ECS)..... 9
  - Organización de un repositorio de ECS..... 10
  - Origen del cambio..... 10
  - Definición de Línea base..... 10
    - Definición de la IEEE..... 11
    - En el contexto de la Ingeniería de Software..... 11
  - Importancia del GCS..... 11
  - Proceso..... 12
    - Identificación..... 12
    - Control de versiones..... 13
    - Control de cambios..... 13
    - Auditoría de la configuración..... 14
    - Generación de informes de estado de la configuración (auditoría)..... 15

Gestión de Proyectos..... 15
Las 4 P de la Gestión de Proyectos de Software..... 15
Resultados de la mala gestión..... 15
Elementos clave de la gestión de proyectos..... 16
Planificación..... 16
Planificación organizativa..... 16
Participantes..... 16
Líderes de equipo..... 17
El equipo de software..... 17
Paradigmas Organizacionales del equipo..... 18
Paradigma cerrado..... 18
Paradigma abierto..... 18
Paradigma síncrono..... 19
Paradigma aleatorio..... 19
El equipo de software..... 19
Comunicación Grupal..... 20
Equipo ágil..... 20
Teoría 3: Riesgos..... 21
Riesgo..... 21
Deuda técnica..... 21
Estrategias de gestión de riesgos:..... 21
Categorización de riesgos..... 22
Proceso de gestión de riesgos..... 23
Identificación de riesgos..... 23
Análisis de riesgos..... 24
Planeación de riesgos..... 26
Ejemplo..... 27
Supervisión..... 28

Teoría 1: Repaso, comunicación y SRS.

Proceso de software

Un proceso de software es un conjunto de actividades y resultados asociados que producen un producto de software.
Estos fueron vistos el año pasado (la etapa elicitación y especificación de requerimientos). Específicamente nosotros usamos SCRUM.

Modelo

Representación abstracta de un proceso de software.

Los modelos de proceso de software cuentan con actividades fundamentales (se presentan en todos los modelos):

1. Especificación del software.
    a. Técnicas de elicitación.
    b. Especificación de requerimentos.
2. Desarrollo del software.
3. Validación del software.
4. Evolución del software.

Comunicación

La comunicación es el proceso más importante de la interacción humana, que se produce de forma verbal y no verbal.

Elicitación de requisitos

Proceso que trata de adquirir toda la información relevante para poder producir un modelo de los requerimientos de un dominio de problema.
Objetivos:

1. Conocer el dominio del problema para facilitar comunicación con clientes y usuarios y entender sus necesidades.
2. Conocer el sistema actual.
3. Identificar necesidades (explícitas e implícitas) de clientes, usuarios y las expectativas que tienen del sistema a desarrollar.

Muestreo de la documentación, los formularios y los datos existentes

Es una técnica de elicitación que recolecta hechos a partir de documentación preexistente. Por ejemplo de:

1. Organigrama (identificación de propietario, usuarios, claves).
2. Memos, notas internas, minutas, registros contables.
3. Solicitudes de proyectos de sistemas informáticos previos.

Permite conocer el historial que origina el proyecto.

Útil cuando el sistema está en funcionamiento y tenemos acceso al mismo. (Por ejemplo si necesitamos crear una transición entre el sistema nuevo y el actual mediante capacitación).

Observación del ambiente de trabajo

Técnica en la cual el analista se convierte en observador de cómo los trabajadores y sistemas interactúan de forma natural.
Lineamientos de la observación:

1. Determinar quién será observado y cuando.
2. Determinar si pido permiso al observado y explicó la necesidad de la observación, o no (avisar puede intimidar al observado, y conviene hacer una sin avisar y otra si).
3. Mantener bajo perfil (evitar intimidar al trabajador puesto que no actuara como suele trabajar en realidad).
4. Tomar nota.
5. Revisar notas con alguien apropiado.
6. No interrumpir.

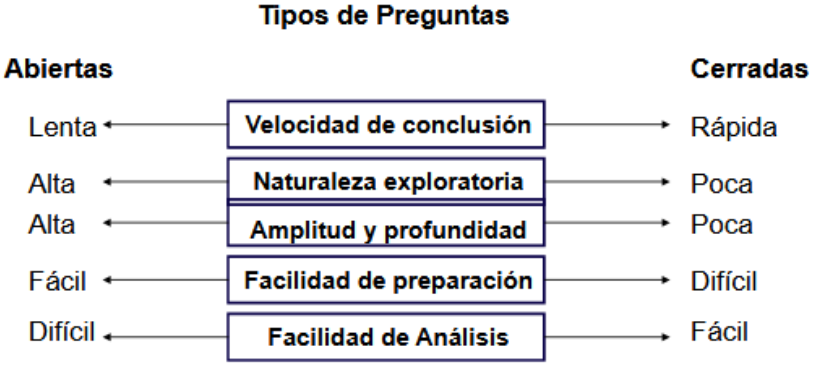
Visitas al sitio

1. Investigar el dominio del problema.
2. Patrones de soluciones (mismo problema en otra organización).
3. Revistas especializadas.
4. Buscar problemas similares en internet.
5. Consultar otras organizaciones.

Extracto del video que explica para qué sirve esta técnica:
"Supóngase que es el software para una fabrica, si, yo pienso y desarrollo el software pero después ¿las máquinas donde van a estar?, ¿donde va a estar el operador para cargar los datos? ... ¿puedo poner la computadora en cualquier lado?¿Voy a estar cerca del lugar que necesito?"

Cuestionarios

2 tipos de preguntas, abiertas y cerradas
Cerradas son las más fáciles de concluir y analizar pero no tienen mucha naturaleza exploratoria ni son amplias o profundas. Las abiertas son lo contrario.



Está bueno tener ambas, lo que está cerrado y categorizado lo hago cerrado y sin lo hago abierto para la próxima.
Sirven para cuando queremos preguntar a muchas personas y también cuando están separadas geográficamente, por lo que en ambos casos sería costoso encontrarse en persona con todos.

Entrevista

Para preparar una entrevista debemos:

1. Leer los **antecedentes**.
2. Establecer los **objetivos** de la misma.
3. **Seleccionar** los **entrevistados.**
4. **Planificar entrevista** (fecha, hora lugar, duración)
5. **Selección** de **tipo** de **preguntas** y su **estructura**. Definir guión, con lenguaje claro y sin opiniones en las preguntas, que sean concisas, y evitar que sean largas y complejas.
   Claramente: Si sabemos poco vamos a tener que explorar un poco más, sino alcanza una sola entrevista habrá que solicitar otra.

3 tipos de organización de entrevistas:

1. Piramidal (inductiva, cerradas a abiertas).
2. Embudo(deductiva, abiertas a cerradas).
3. Diamante(combinación, cerradas a abiertas a cerradas).

El profe recomienda diamante si falta calle.

JRP

Requiere de una instalación física, más compleja de llevar a cabo al tener que reunir mucha gente. Tiene de bueno que al tener todos los actores involucrados en un mismo lugar se genera un ambiente que permite resolver muchas dudas y discrepancias entre los requerimientos de una.

Brainstorming

Consiste en que todos los participantes digan todas las ideas que tengan sobre un cierto tema para luego revisarlas para quedarse con las mejores. Promueve el desarrollo de ideas creativas.

Requerimientos SRS

Requerimientos

Los requerimientos pasan por 4 etapas:

1. Solicitud
2. Definición

3. Análisis
4. Especificación (acá usamos HU u otro visto por ejemplo).

Tipos de requerimientos:

- Funcionales: Definen el comportamiento del sistema, describen las tareas que el sistema debe realizar, es importante que se defina manteniendo un equilibrio entre generalidad y exceso de detalle al pedo.
- No funcionales: Cosas fuera del sistema, definen aspectos, deseables desde el punto de vista del usuario. O también restricciones como tiempos de respuesta, usabilidad, mantenimiento, etc.

ConOps (IEEE Std. 1362-1998)

Estándar de documento para describir características del sistema escrito desde el punto de vista del usuario, dirigido para el usuario.
Esta descripción es la forma en que se puede comunicar la visión general, cualitativa y cuantitativa de las características del sistema entre cliente y desarrollador.
Ofrece formato específico para ser completado, no especifica técnica exacta sino que proporciona guías para poder hacerlo, identifica elementos que al menos deben estar.

SRS (standard requirements specifications) IEEE Std. 830-1998

Un SRS especifica funciones de un producto de software, programa o conjuntos de programa en un entorno. El documento de especificación de requisitos puede desarrollarlo personal desarrollador o de la parte cliente (conviene ambos).
Características de SRS:

1. Alcance: Brindar una colección de buenas prácticas para escribir especificaciones. Se describen los contenidos y las cualidades de una buena especificación de requerimientos.
2. Naturaleza del SRS: El SRS es una especificación de un producto de software en particular, escrito por uno o más personas (desarrolladores o clientes).
3. Ambiente del SRS: El software puede contener toda la funcionalidad del proyecto o ser parte de un sistema más grande. En el último caso habrá un SRS que declarará las interfaces entre el sistema y su software desarrollado y pondrá qué función externa y requerimientos de funcionalidad tiene con el software desarrollado.
4. Correcto: Un SRS es correcto si cada requisito declarado se encuentra en el software.

5. No ambiguo: Un SRS es inequívoco si y sólo si cada requisito declarado tiene solo una interpretación.
6. Completo: Reconoce todos los requisitos externos impuestos por especificaciones del sistema
7. Consistente: Tiene que estar de acuerdo con los documentos de nivel superior (como una especificación de requerimientos).
8. Priorizado: Es priorizado por la importancia de sus requerimientos particulares.
9. Comprobable: Es comprobable si sus requisitos declarados son comprobables (no ambiguos).
    a. Requisito comprobable: Es comprobable si hay algún proceso para verificar que el producto de software cumple el requisito.
10. Modificable: Su estructura y estilo permite cambios a los requerimientos fácilmente y consistentemente conservando la estructura y estilo.
11. Trazabilidad: Claridad del origen de cada requerimiento y su trazabilidad hacia los requerimientos futuros.
12. Preparación conjunta del SRS: Preparar en conjunto con las partes involucradas y así formar un buen acuerdo.
13. Evolución de SRS: Debe evolucionar junto al software, registrar cambios y responsables y aceptar los mismos.
14. Prototipos: Usa prototipos para la definición de requerimientos.
15. Diseño incorporado en el SRS: Puede incorporar atributos o funciones externas como diseños para interactuar entre subsistemas.
16. Requerimientos incorporados en el SRS: Los detalles particulares son anexados como documentos externos.

Sección 1 del SRS

- Propósito: Define el propósito del documento y se especifica a quien va dirigido.
- Alcance o ámbito del sistema: Nombra al sistema, explica que hace y no hace el sistema, describe los beneficios, objetivos y metas futuras.
- Referencias: Lista completa de referencias de los documentos usados para escribir el SRS. (título, número de reporte, fecha y publicación de cada documento y las fuentes de donde se obtuvieron).

Sección 2 del SRS

Factores generales que afectan al producto y sus requerimientos

- Perspectiva del producto: Se declara si el producto es independiente y totalmente autónomo. Si es parte de un sistema más grande se

relacionan los requerimientos de este a la funcionalidad del software e identificar las interfaces entre ese sistema y el software.
- Funciones del sistema: Resumen de las funciones del sistema organizadas y puede usar gráficos para las relaciones entre las mismas.
- Características del usuario: Características generales de los usuarios del producto (nivel educativo, experiencia y especialización técnica).
- Evoluciones previsibles del sistema: Requerimientos a implementar en el futuro.

Sección 3 del SRS

Requerimientos no funcionales del software a detalle para el diseño del sistema y que se pueda probar que satisface estos requerimientos

- Requerimientos de rendimiento: Relacionados a la carga esperada para el sistema. Debe ser mesurable. Por ejemplo, tiempo esperado por transacción (por ejemplo, que todas tarden un 1 segundo)
- Seguridad: Relacionados a la protección de accesos, usos, sabotajes, modificaciones y destrucciones maliciosas o accidentales. Por ejemplo, logs de actividad
- Portabilidad: Relacionados a la facilitación de traslado a otras plataformas o entornos. Por ejemplo, el uso de un determinado lenguaje por su portabilidad.

Sección 4 del SRS

Identifica el tipo de mantenimiento necesario para el sistema y quien realiza estas tareas junto a cada cuanto se deben de hacer. Por ejemplo, generación de estadísticas de acceso semanal por el desarrollador.

Sección 5 del SRS

Información relevante que no forma parte del SRS. Por ejemplo, casos de uso.

Teoría 2: GCS, planificación organizativa.

Gestión de la configuración de software (GCS)

Cambios

Durante el proceso de software pueden ocurrir cambios grandes o chicos de ciertas partes del sistema en construcción que pueden ser un perjuicio para el desarrollo de no ser controlados correctamente. Estos cambios deben ser reportados a las personas correctas o llevará a que el proceso se nos vaya de las manos.

GCS

Es un conjunto de actividades orientadas a gestionar el cambio.
Gestión de la configuración es el proceso de:

1. Identificar y definir elementos del sistema.
2. Controlar los cambios de estos elementos a lo largo de su ciclo de vida.
3. Registrar y reportar el estado de los elementos y las solicitudes de cambio.
4. Verificar que los elementos estén completos y correctos.

Actividad de **autoprotección** aplicada durante el proceso de software.

Involucrados de la GCS

Todos los involucrados en el proceso de software se relacionan en cierta medida con la gestión del cambio, aunque a veces se crean de apoyo especializadas para el proceso ACS.

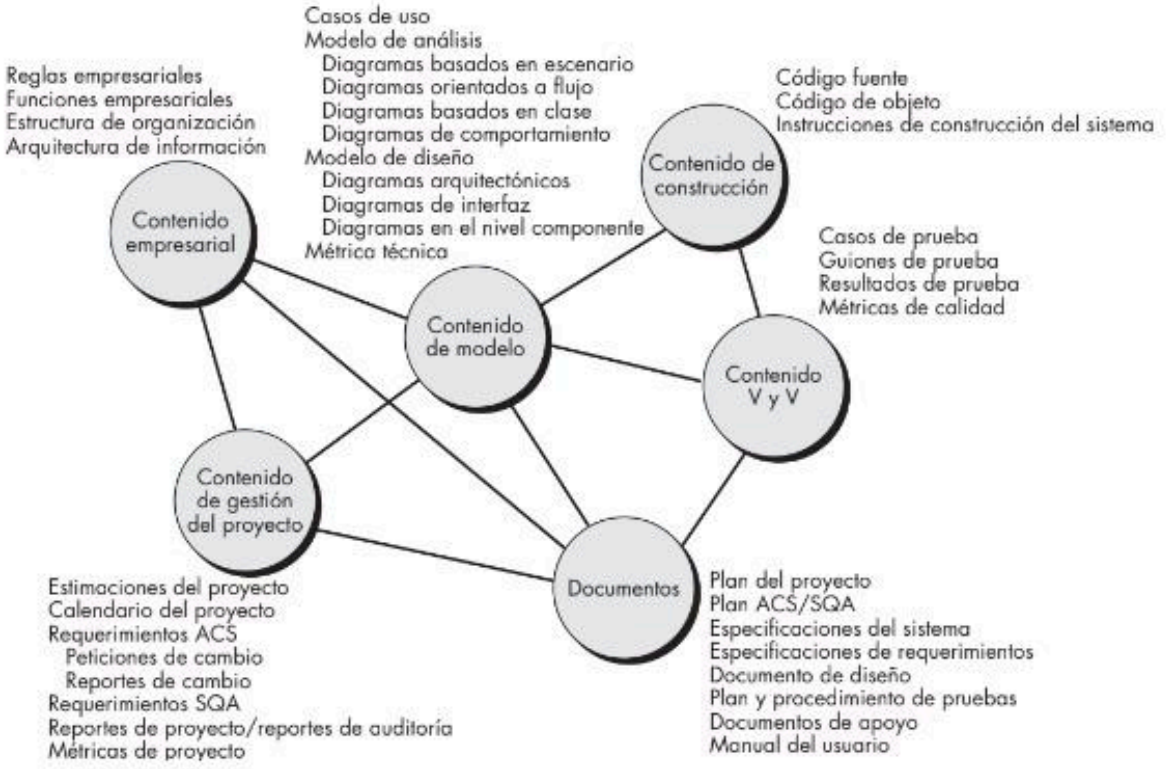
Elementos de la GCS (ECS)

1. Programas.
2. Productos de trabajo.
3. Datos.

¿Cuáles podrían ser elementos de la configuración?

✓Especificación del sistema ✓Plan del proyecto software ✓Especificación de diseño: ✓ a) Diseño preliminar ✓ b) Diseño detallado ✓Listados del código fuente ✓Planificación y procedimiento de prueba ✓ Casos de prueba y resultados registrados	✓Manuales de operación y de instalación ✓Programas ejecutables ✓Descripción de la base de datos a) Esquema, modelos b)Datos iniciales ✓Manual de usuario ✓Documentos de mantenimiento ✓Estándares y procedimientos de ingeniería del software

Organización de un repositorio de ECS



Origen del cambio

1. Condiciones de negocio o mercado cambiantes afectan los requerimientos.
2. Nuevas necesidades de las partes interesadas demandan modificaciones en funcionalidades, datos o servicios del sistema.
3. La reorganización, crecimiento o reducción de la empresa altera las prioridades del proyecto o la estructura del equipo.

4. Restricciones en el presupuesto o calendario llevan a redefiniciones del sistema.

Las actividades de la Gestión del Cambio en Software (GCS) se centran en:

1. Identificar el cambio.
2. Controlar el cambio.
3. Garantizar una implementación adecuada del cambio.
4. Informar a todas las partes afectadas por el cambio.

Definición de Línea base

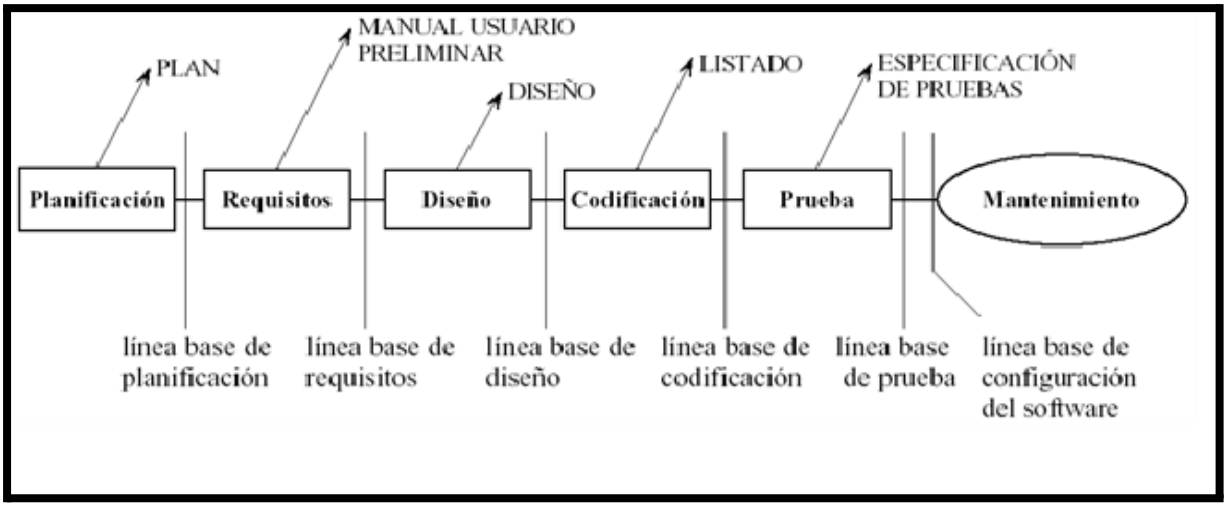
Punto de referencia en el desarrollo de software.

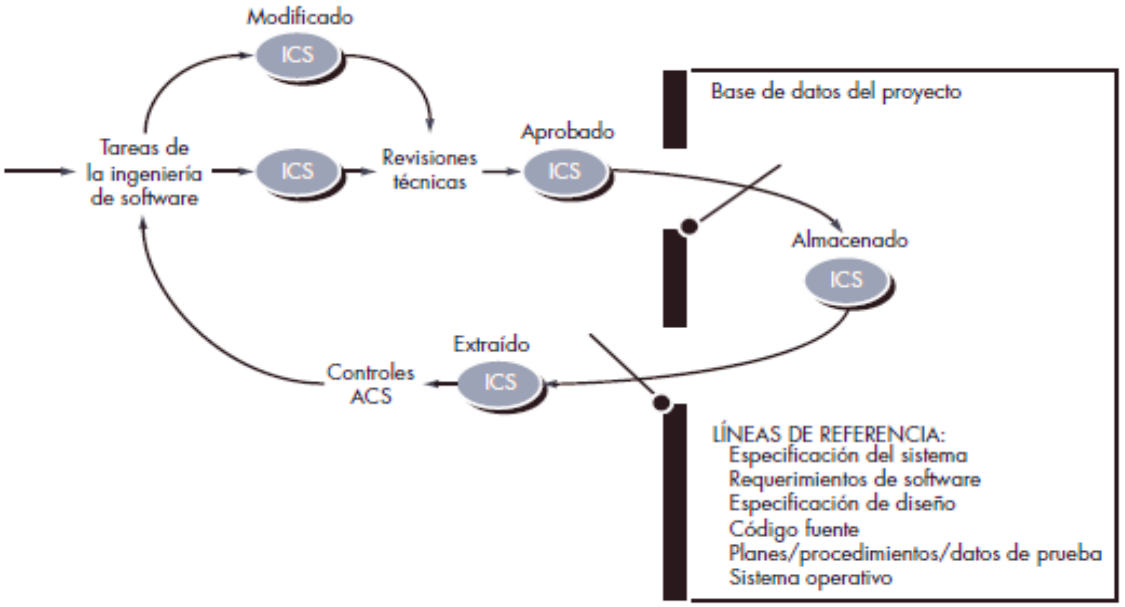
Definición de la IEEE

Una especificación o producto que se ha revisado formalmente y sobre el que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambio

En el contexto de la Ingeniería de Software

Una línea base es un punto de referencia en el desarrollo del software que queda marcado por el envío de uno o más ECS y su aprobación





Importancia del GCS

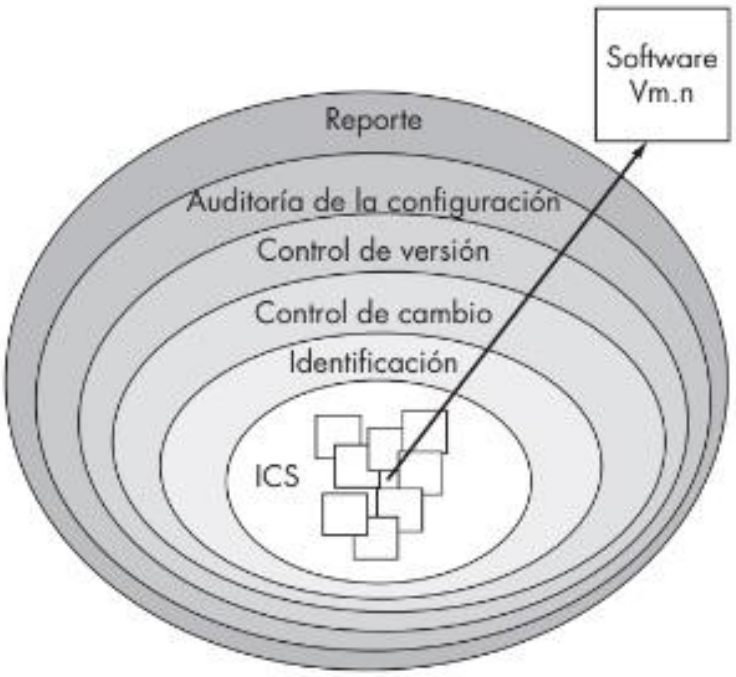
A qué preguntas el GCS le da respuesta:

1. ¿Cómo identifica y gestiona una organización las diferentes versiones existentes de un programa (y su documentación) de forma que se puedan introducir cambios eficientemente?
2. ¿Cómo controla la organización los cambios antes y después de que el software sea distribuido al cliente?
3. ¿Quién tiene la responsabilidad de aprobar y de asignar prioridades a los cambios?
4. ¿Cómo podemos garantizar que los cambios se han llevado a cabo adecuadamente?
5. ¿Qué mecanismo se usa para avisar a otros de los cambios realizados?

Proceso

Define tareas con objetivos precisos:

1. Identificación
2. Control de versiones
3. Control de cambios
4. Auditorías de la configuración
5. Generación de informes



Identificación

A cada elemento de la GCS se le da:

1. Nombre: cadena de caracteres sin ambigüedad
2. Descripción: lista de elementos de datos que identifican:
3. Tipo de ECS (documento, código fuente, datos)
4. Identificador del proyecto
5. Información de la versión y/o cambio

Control de versiones

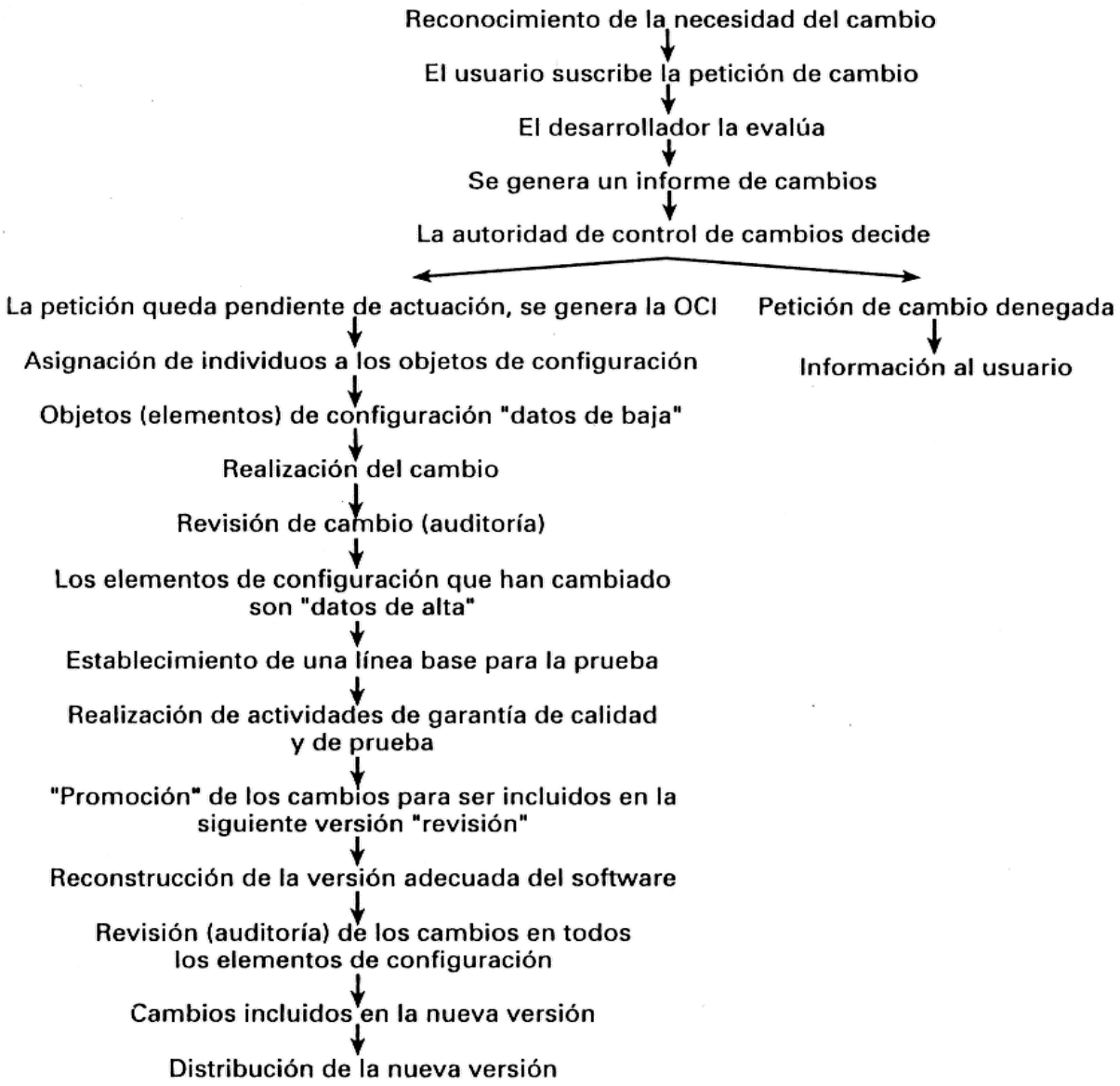
Combinación de procedimientos y herramientas para gestionar las versiones de los ECS que se crean a lo largo del proceso de software.
Ejemplo de versiones
Un programa puede contener los módulos 1-2-3-4-5
Una versión puede utilizar los módulos 1-2-3-5
Otra versión puede utilizar los módulos 1-2-4-5

Dos variantes de un mismo programa

Repositorio	Se almacenan los archivos actualizados e históricos de cambio del proyecto.
Versión	Determina un conjunto de archivos
Master	Conjunto de archivos principales del proyecto
Abrir rama – branch	Bifurcación del máster para trabajar sobre dos ramas de forma independiente
Desplegar – check-out	Copia de trabajo local desde el repositorio.
Publicar - Commit	Una copia de los cambios hechos a una copia local es escrita o integrada sobre repositorio
Conflicto	Problema entre las versiones de un mismo documento
Cambio – diff	Representa una modificación específica
Integración – Merge	Fusión entre dos ramas del proyecto
Actualización – sync o update	Integra los cambios que han sido hechos en el repositorio y las copias locales

Control de cambios

En el desarrollo de proyectos, los cambios son inevitables y un control efectivo es crucial. Se emplea una combinación de procedimientos humanos y herramientas adecuadas para establecer un mecanismo eficiente de control de cambios.



Auditoría de la configuración

La identificación y control de versiones, junto con el control de cambio, son esenciales para mantener el orden en el equipo de desarrollo de software. Para garantizar la generación de la orden de cambio usamos:

- Revisiones técnicas formales
- Auditorías de configuración

Auditoría de la configuración responde:

1. ¿Se ha hecho el cambio especificado en la Orden de Cambio?¿Se han incorporado modificaciones adicionales?
2. ¿Se ha llevado a cabo una RTF para evaluar la corrección técnica?
3. ¿Se han seguido adecuadamente los estándares de IS?

4. ¿Se han reflejado los cambios en el ECS: fecha, autor, atributos?
5. ¿Se han seguido procedimientos de GCS para señalar el cambio, registrarlo y divulgarlo?
6. ¿Se han actualizado adecuadamente todos los ECS relacionados?

Generación de informes de estado de la configuración (auditoría)

Responde

1. ¿Qué pasó?
2. ¿Quién lo hizo?
3. ¿Cuándo pasó?
4. ¿Qué más se vio afectado?

La generación de informes de estado de la configuración desempeña un papel vital en el éxito del proyecto.

Gestión de Proyectos

Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único.
Características

1. Temporal: Tiene un comienzo y fin definido.
2. Resultado: Productos, servicios o resultados únicos
3. Elaboración gradual: Desarrollar en pasos e ir aumentando mediante incrementos

Las 4 P de la Gestión de Proyectos de Software

1. Personal (RRHH): Es el elemento más importante.
2. Producto: El producto de software es intangible. A veces es difícil ver el progreso del proyecto.
3. Proceso: Un proceso de software proporciona el marco de trabajo desde el cual se puede establecer un plan detallado para el desarrollo del software.
4. Proyecto: Los proyectos deben ser planeados y controlados para manejar su complejidad.

Resultados de la mala gestión

1. Incumplimiento de plazos,
2. Incremento de los costos,

3. Entrega de productos de mala calidad.

Elementos clave de la gestión de proyectos

1. Métricas
2. Planificación
    a. Estimaciones
    b. Calendario temporal
    c. Organización del personal
3. Análisis de riesgos
4. Seguimiento y control

Planificación

La planificación específica:

1. Que debe hacerse.
2. Qué recursos se van a usar.
3. Qué orden se va tomar.

Establece una secuencia operativa.

Planificación organizativa

El personal en una organización de software es su activo más valioso y capital intelectual. Una gestión deficiente del personal es un factor clave en el fracaso de proyectos. Por ello, se implementó el Modelo de Madurez de Capacidades de Personal (P-CMM), que destaca la necesidad de mejorar constantemente la capacidad de atraer, desarrollar, motivar, organizar y retener al personal.

Participantes

1. Gerentes ejecutivos (dueños del producto): Definen los temas empresariales
2. Gerentes de proyecto (líderes de equipo): Planifican, motivan, organizan y controlan a los profesionales
3. Profesionales especializados: Aportan habilidades técnicas
4. Clientes: Especifican los requerimientos
5. Usuarios finales: Interactúan con el software

Líderes de equipo

Los líderes de equipo en el desarrollo de software deben optimizar las habilidades individuales. Las prácticas de líderes destacados incluyen:

1. Modelar el camino: Practicar lo que predican y comprometerse con el equipo y el proyecto.
2. Inspirar y compartir visión: Motivar miembros del equipo y desde el inicio del establecimiento de metas involucrar a todos.
3. Desafiar el proceso: Buscar formas innovadoras de mejora, alentar la experimentación y asumir riesgos.
4. Permitir que otros actúen: Fomentar habilidades colaborativas, construir confianza y facilitar relaciones.
5. Alentar: Celebrar logros individuales y fomentar un sentido de comunidad en el equipo.

El equipo de software

La estructura para nuestro equipo de software depende del estilo gerencial de la organización, la cantidad de gente que conformará el equipo, el nivel de habilidad, y la dificultad del problema.
Los equipos no deberían tener más de 10 miembros, aunque depende del tamaño del proyecto.

Factores a considerar cuando se planea una estructura de equipo

1. Dificultad del problema a resolver
2. Tamaño del programa resultante
3. Tiempo que el equipo permanecerá unido
4. Grado en que puede dividirse en módulos el problema a resolver
5. Calidad y confiabilidad requerida por el sistema a construir
6. Rigidez de la fecha de entrega
7. Grado de sociabilidad requerido en para el proyecto

Problemas y cómo evitarlos :

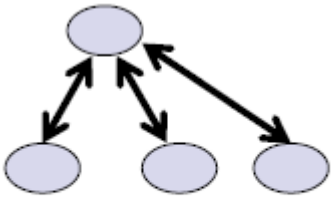
1. Atmósfera de trabajo frenética: El equipo tiene acceso a toda la información y las principales metas y objetivos no se modifican a menos que sea absolutamente necesario.
2. Fricción entre los miembros del equipo: se define responsabilidad a cada uno .
3. Proceso de software fragmentado o mal coordinado: entender lo que se va a crear y permitir que el equipo seleccione el modelo de proceso.
4. Definición imprecisa de roles: definir enfoques correctivos ante un miembro que no cumple.

5. Exposición continua y repetida al fracaso: utilización de técnicas basadas en la retroalimentación y solución de problemas.

Paradigmas Organizacionales del equipo

Paradigma cerrado

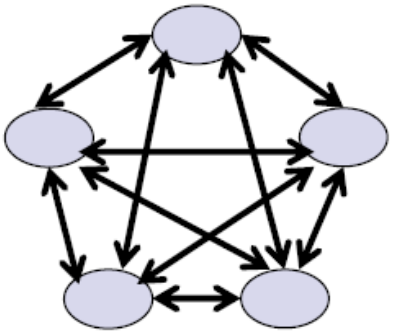
- Estructurado conforme a una jerarquía tradicional (comunicación vertical entre jefe y miembros del equipo).
- Trabajan bien produciendo software similar al de esfuerzos anteriores.
- Menos innovadores.



Trayectorias de Comunicación

Paradigma abierto

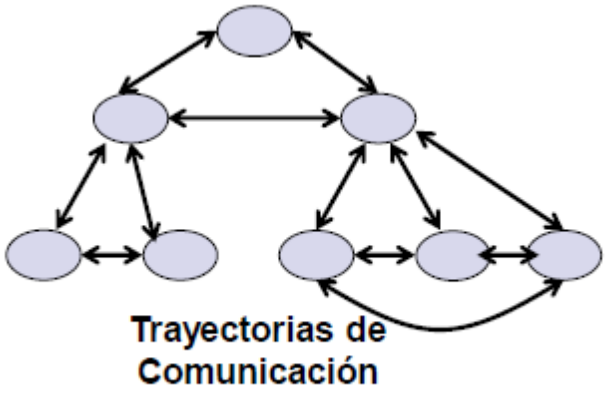
- Se intenta estructurar para lograr algunos controles asociados con el paradigma cerrado pero mucha innovación como el paradigma aleatorio.
- Trabajo hecho de forma colaborativa.
- Gran comunicación.
- Toma de decisiones consensuadas. (Equipo democratico).
- Estructura útil para resolución de problemas complejos. Pueden no desempeñarse tan eficazmente como otros equipos.



Trayectorias de Comunicación

Paradigma síncrono

- Apoyado en la compartimentalización natural de un problema.
- Organiza a los miembros del equipo para trabajar en trozos del problema con poca comunicación activa entre ellos.
- Equipo descentralizado y controlado a la vez (divide y vencerás).
- Bueno para problemas de gran tamaño y complejidad.
- Fraccionamiento puede llevar a que algunos subgrupos produzcan algo que no sirva debido a la comunicación.



Paradigma aleatorio

- Estructura el equipo de manera holgada y depende de la iniciativa individual de los miembros.
- Los miembros se organizan entre ellos y no requiere de un líder definido.
- Las trayectorias de comunicación pueden ser cualquiera de los tres paradigmas.
- Destaca cuando se requiere innovación. Pero es problemático cuando se requiere "desempeño ordenado".

El equipo de software

Sin importar la organización del equipo, el objetivo es un equipo con alta cohesión ya que estos suelen ser más productivos y motivados.
Beneficios de alta cohesión:

1. Pueden establecer propios estándares de calidad.
2. Los individuos aprenden de los demás y se apoyan mutuamente.
3. El conocimiento se comparte.
4. Se alienta la refactorización y el mejoramiento continuo.

Un equipo no consolidado suele sufrir toxicidad de equipo :

1. Atmósfera de trabajo frenética.

2. Fricción entre los miembros del equipo.
3. Proceso de software fragmentado o mal coordinado.
4. Definición imprecisa de roles.
5. Exposición continua y repetida al fracaso.
6.

Comunicación Grupal

- Las comunicaciones en un grupo se ven influenciadas por factores como: status de los miembros del grupo, tamaño del grupo, composición de hombres y mujeres, personalidades y canales de comunicación disponible.
- Se deben establecer mecanismos para la comunicación formal e informal entre los miembros del equipo.
- La comunicación formal: a través de reuniones, escritos y otros canales no interactivos.
- La comunicación informal: Los miembros comparten ideas sobre la marcha.

Equipo ágil

Muchas organizaciones defienden el desarrollo ágil de software, lo que conlleva a crear equipos pequeños y muy motivados, lo que se denomina equipo ágil.

- Se hace hincapié en la competencia individual y colaboración grupal.
- Son autoorganizados.

Se parece a un paradigma aleatorio con pocos miembros.

Teoría 3: Riesgos.

Riesgo

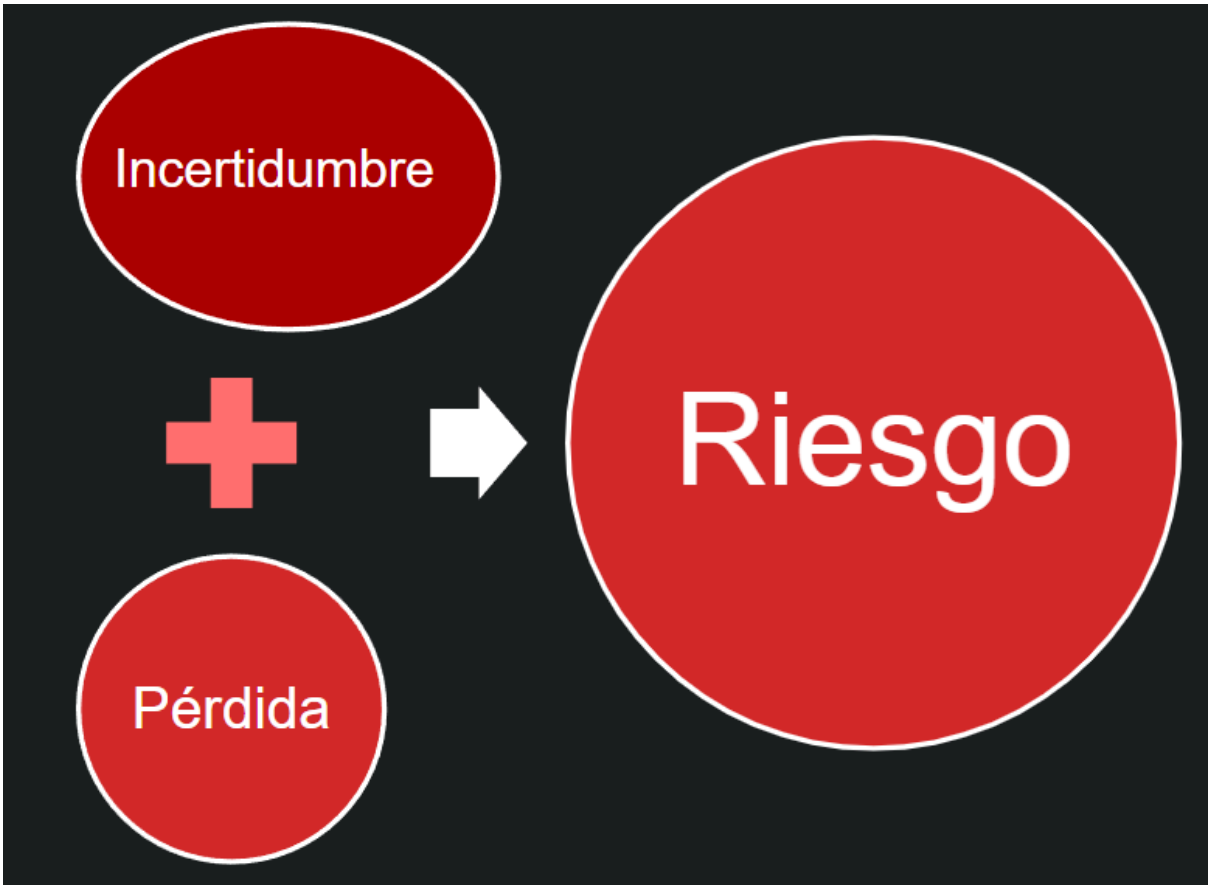
1. Evento no deseado que acarrea consecuencias negativas.
2. Los gerentes deben determinar la probabilidad de que ocurran durante el desarrollo y planear para evitarlos o mitigarlos.
3. El riesgo concierne a:
    a. Lo que ocurrirá en el futuro: ¿Cuáles son los riesgos que pueden hacer que fracase el proyecto?
    b. Cómo afectarán los cambios al desarrollo: ¿Como afectarán al éxito global y a los plazos los cambios en los requisitos del cliente, en las tecnologías de desarrollo, etc...?
    c. A las elecciones: ¿Qué métodos y herramientas debemos usar, cuánta gente debe estar involucrada, cuánta importancia hay que darle a la calidad?

Deuda técnica

- Término que describe el costo asociado con aplazar actividades como documentación y refactorización.
- No sé paga literalmente sino que hace que el producto sea de mala calidad.
- Implica que el uso de recursos, tiempo y esfuerzo para luchar contra los temas técnicos puede reducirse si se afrontan los problemas al inicio.
- En Agile se sigue gestionando los riesgos, porque si no generaría deuda técnica.

Estrategias de gestión de riesgos:

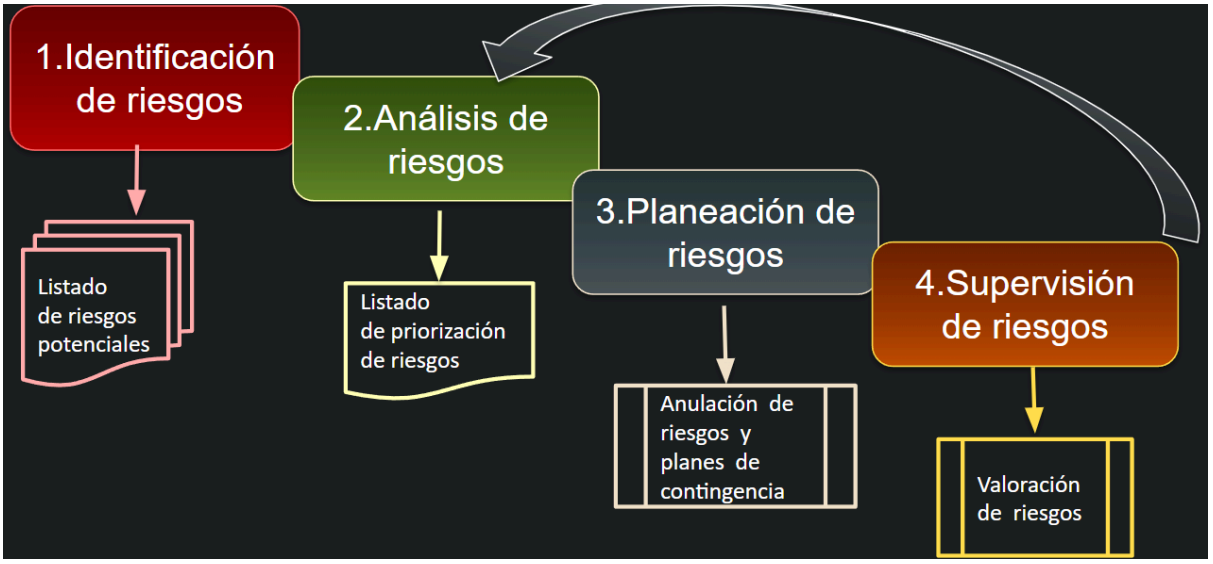
1. Reactiva: Reaccionar contra el problema y “gestionar la crisis” a lo Indiana Jones.
2. Proactivas: Plantear estrategias de tratamiento de riesgos antes de que ocurran.



Categorización de riesgos

1. Proyecto: Por ejemplo presupuesto, recursos y calendario.
2. Producto: Por ejemplo calidad o cambios en requerimientos.
3. Negocio: Por ejemplo sistema que nadie quiere, sistema que no se adapta a la empresa, o la gerencia deja de apoyar el producto.

Proceso de gestión de riesgos



El proceso de gestión de riesgos consiste en un esfuerzo grupal e iterativo, que sirve para gestionar los riesgos y la forma de lidiar con ellos a lo largo del proyecto.

Identificación de riesgos

Generalmente se usa un enfoque un "Brainstorming" con nuestro grupo de trabajo y generamos este listado, o en base a la experiencia.
Los riesgos pueden ser:

- Conocidos.
- Predecibles.
- Impredecibles.

Además debemos determinar si cada uno es un riesgos de proyecto son de:

- Proyecto.
- Predecibles.
- Impredecibles.

Hay ciertas preguntas que nos podemos hacer, si la respuesta es negativa, muy probablemente estemos ante un riesgo:

1. ¿Los gerentes de software y de cliente se reunieron formalmente para apoyar el proyecto?
2. ¿Los usuarios finales se comprometen con el proyecto y sistema/producto que se va a construir?
3. ¿El equipo y sus clientes entienden por completo los requisitos?
4. ¿Los clientes se involucraron plenamente en la definición de los requisitos?
5. ¿Los usuarios finales tienen expectativas realistas?
6. ¿El ámbito del proyecto es estable?
7. ¿El equipo tiene la mezcla correcta de habilidades?

8. ¿Los requisitos del proyecto son estables?
9. ¿El equipo tiene experiencia con la tecnología que se va a implementar?
10. ¿El número de personas que hay en el equipo es adecuado para hacer el trabajo?
11. ¿Todos los clientes/usuarios están de acuerdo en la importancia del proyecto y en los requisitos para el sistema/producto que se va a construir?

Ejemplos de riegos:

Riesgo	Repercute en	Descripción
Rotación de personal	Proyecto	Personal experimentado abandonará el proyecto antes de que éste se termine.
Cambio administrativo	Proyecto	Habrá un cambio de gestión en la organización con diferentes prioridades.
Indisponibilidad de hardware	Proyecto	Hardware, que es esencial para el proyecto, no se entregará a tiempo.
Cambio de requerimientos	Proyecto y producto	Habrá mayor cantidad de cambios a los requerimientos que los anticipados.
Demoras en la especificación	Proyecto y producto	Especificaciones de interfaces esenciales no están disponibles a tiempo.
Subestimación del tamaño	Proyecto y producto	Se subestimó el tamaño del sistema.
Bajo rendimiento de las herramientas CASE	Producto	Las herramientas CASE, que apoyan el proyecto, no se desempeñan como se anticipaba.
Cambio tecnológico	Empresa	La tecnología subyacente sobre la cual se construye el sistema se sustituye con nueva tecnología.
Competencia de productos	Empresa	Un producto competitivo se comercializa antes de que el sistema esté completo.

Análisis de riesgos

Acá a cada riesgo del listado lo calificamos según su probabilidad de ocurrir e impacto. Luego elegimos aquellos a priorizar según las calificaciones que le dimos y hacemos una "línea de corte" (más tarde se explica), esos serán nuestros riesgos a tratar.

Calificaciones de probabilidad de riesgo:

- Bastante improbable : < 10%
- Improbable : 10-25%
- Moderado : 25-50%
- Probable :50-75%
- Bastante probable : >75%

Calificaciones de impacto de riesgo:

- Catastrófico: cancelación del proyecto
- Serio: reducción de rendimiento, retrasos en la entrega, excesos importante en costo

- Tolerable: reducciones mínimas de rendimiento, posibles retrasos, exceso en costo
- Insignificante: incidencia mínima en el desarrollo

Tabla de riesgos de ejemplo:

Riesgos	Categoría	Probabilidad	Impacto
El cliente cambiará los requisitos	Proy	80%	2
Falta de formación en las herramientas	Proy	80%	3

Boehm recomienda supervisar los 10 riesgos más altos (hacer la línea de corte a ese nivel), según el profe alrededor de la mitad si son poco, sino el 10%, siempre que sea un número manejable.
Aquellos elegidos para supervisar se le prestara atención, los otros serán reevaluados y tendrán un prioridad de segundo orden.

Los riesgos de gran impacto con poca probabilidad no toman un tiempo significativo

Los riesgos de gran impacto con media o mucha probabilidad y los de poco impacto con mucha probabilidad se toman en cuenta

Ejemplo:

Riesgos	*Categoría*	*Probabilidad*	*Impacto*
El cliente cambiara los requisitos	Proyecto	80%	2
Falta de formación en las herramientas	Producto	80%	3
Menos reutilización de la entrevista	Proyecto	70%	2
La estimación del tamaño puede ser muy baja	Proyecto	60%	2

Habrá muchos cambios de personal	Proyecto	60%	2
La fecha de entrega estará muy ajustada	Proyecto	50%	2
LINEA DE CORTE			
Se perderán los presupuestos	Negocio	40%	1
Los usuarios finales se resisten al sistema	Negocio	40%	3
La tecnología no alcanzará las expectativas	Producto	30%	1
Personal sin experiencia	Proyecto	30%	2
Mayor número de usuarios de los previstos	Negocio	30%	3

Planeación de riesgos

Aquellos riesgos que elegimos tratar, generamos una estrategia para tratarlo:

- Intentamos evitar el riesgo completamente.
- Si no podemos evitarlos, minimizamos la probabilidad de que ocurra.
- Si no podemos minimizar la probabilidad de que ocurran, minimizamos el impacto del mismo.

Supervisión de riesgos, A medida que va pasando el tiempo revisamos:

Aparecieron.

a. Cómo funcionó el plan de contingencia.
b. Si hay que sacar o agregar riesgos.
c. Además de cambiar las probabilidad de que ocurran.
d. Volvemos al paso 2, análisis de riesgos.

Considera los riesgos sobre la línea de corte y les asigna una estrategia para evitarlos.

- Evitar el riesgo: Diseña el sistema de forma que el evento no pueda ocurrir
- Minimizar el riesgo: Reduce la probabilidad de que el evento ocurra
- Plan de contingencia: Acepta que el evento ocurra y busca minimizar las consecuencias

Ejemplo

Riesgo	Estrategia
Problemas financieros de la organización	Prepare un documento informativo para altos ejecutivos en el que muestre cómo el proyecto realiza una aportación muy importante a las metas de la empresa y presente razones por las que los recortes al presupuesto del proyecto no serían efectivos en costo.
Problemas de reclutamiento	Alerte al cliente de dificultades potenciales y de la posibilidad de demoras; investigue la compra de componentes.
Enfermedad del personal	Reorganice los equipos de manera que haya más traslape de trabajo y, así, las personas comprendan las labores de los demás.
Componentes defectuosos	Sustituya los componentes potencialmente defectuosos con la compra de componentes de conocida fiabilidad.
Cambios de requerimientos	Obtenga información de seguimiento para valorar el efecto de cambiar los requerimientos; maximice la información que se oculta en el diseño.
Reestructuración de la organización	Prepare un documento informativo para altos ejecutivos en el que muestre cómo el proyecto realiza una aportación muy importante a las metas de la empresa.
Rendimiento de la base de datos	Investigue la posibilidad de comprar una base de datos de mayor rendimiento.
Subestimación del tiempo de desarrollo	Investigue los componentes comprados; indague el uso de un generador de programa.

Exposición de riesgo (E)

- Se determina con la ecuación E = P x C
- P es la probabilidad de ocurrencia para un riesgo
- C es el costo para el proyecto si ocurre el riesgo

Ejemplo:

Riesgo: sólo 70 por ciento de los componentes de software calendarizados para reutilización se integrarán en la aplicación. La funcionalidad restante tendrá que desarrollarse a la medida.

Probabilidad: 80%

Impacto: Se planificaron 60 componentes de software reutilizables. Si sólo puede usarse 70 por ciento (42 componentes), tendrán que desarrollarse 18 componentes desde cero (además de otro software a la medida que se calendarizó para desarrollo).

Dado que el componente promedio es de 100 LOC (lines of code) y que el costo de la ingeniería del software para cada LOC es US$14.00, el costo global (impacto) para desarrollar los componentes sería 18 x 100 x 14 = US$25.200.

Exposición al riesgo. E= 0.80 X 25.200 = US$20.200.

Supervisión

- Evaluar si la probabilidad de cada riesgo cambio
- Evaluar que tan efectivas son las estrategias
- Detectar la ocurrencia de un riesgo que fue previsto
- Asegurar que se están cumpliendo los pasos definidos para cada riesgo
- Recopilar información para el futuro
- Determinar si existen nuevos riesgos
- Reevaluar periódicamente los riesgos
- Monitorizar los riesgos en todas las etapas del proyecto. En cada revisión administrativa, es necesario reflexionar y estudiar cada uno de los riesgos clave por separado decidiendo si es más o menos probable que surja el riesgo, y si cambiaron la gravedad y las consecuencias del riesgo