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.

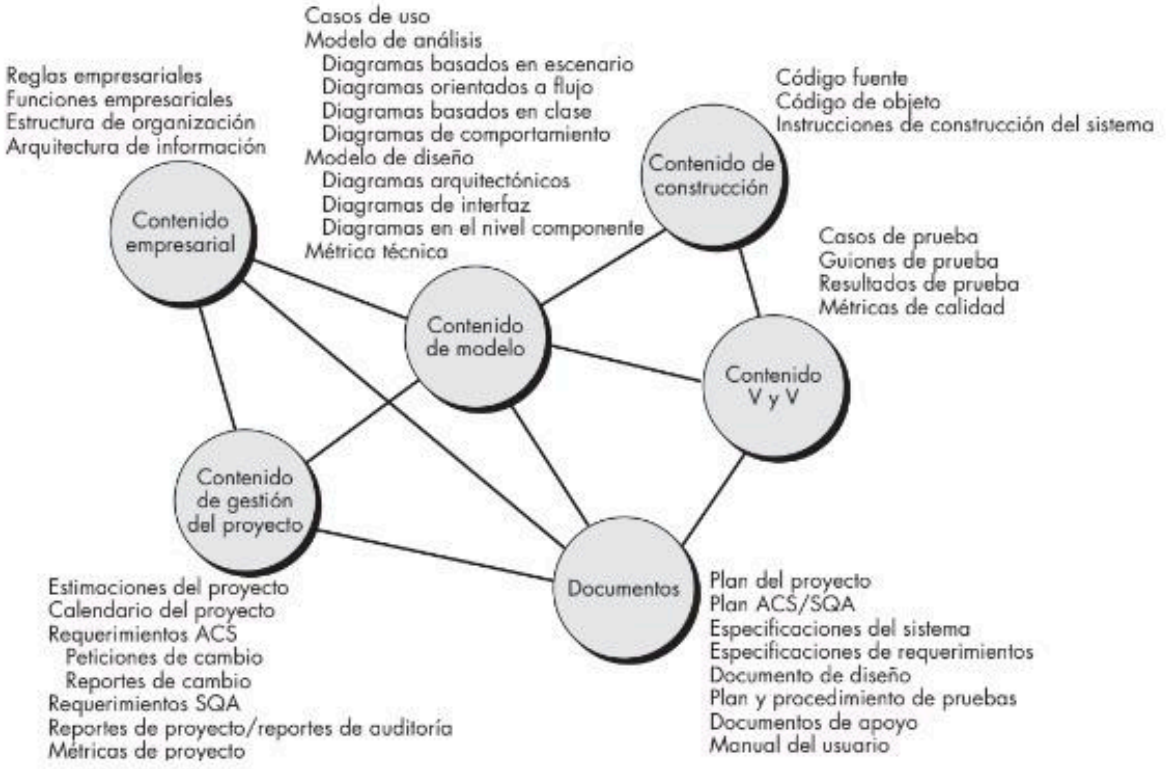
Elementos de la GCS (ECS)

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

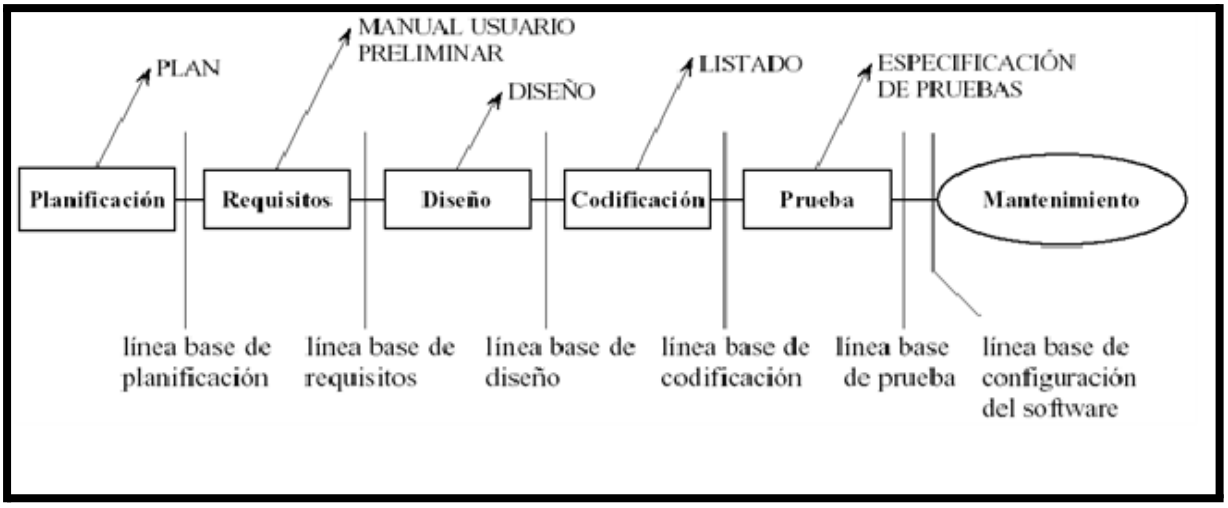
Punto de referencia en el desarrollo de software.

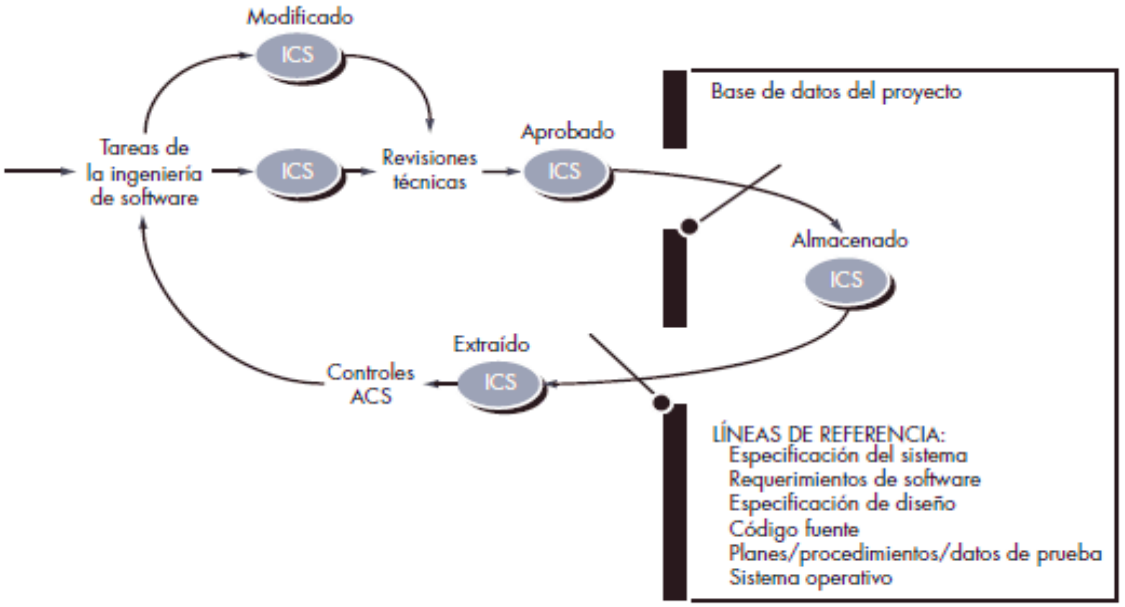
Definición de la IEEE

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

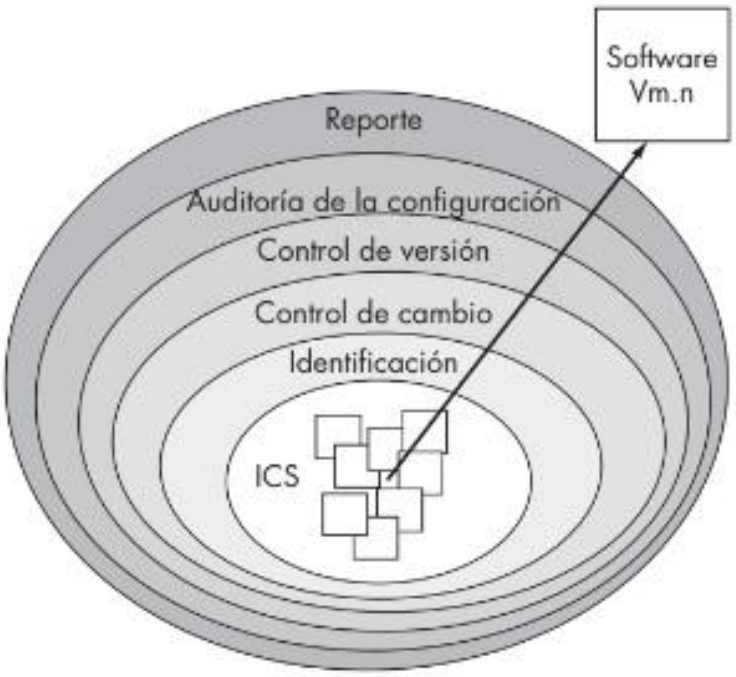
A qué preguntas el GCS le da respuesta:

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

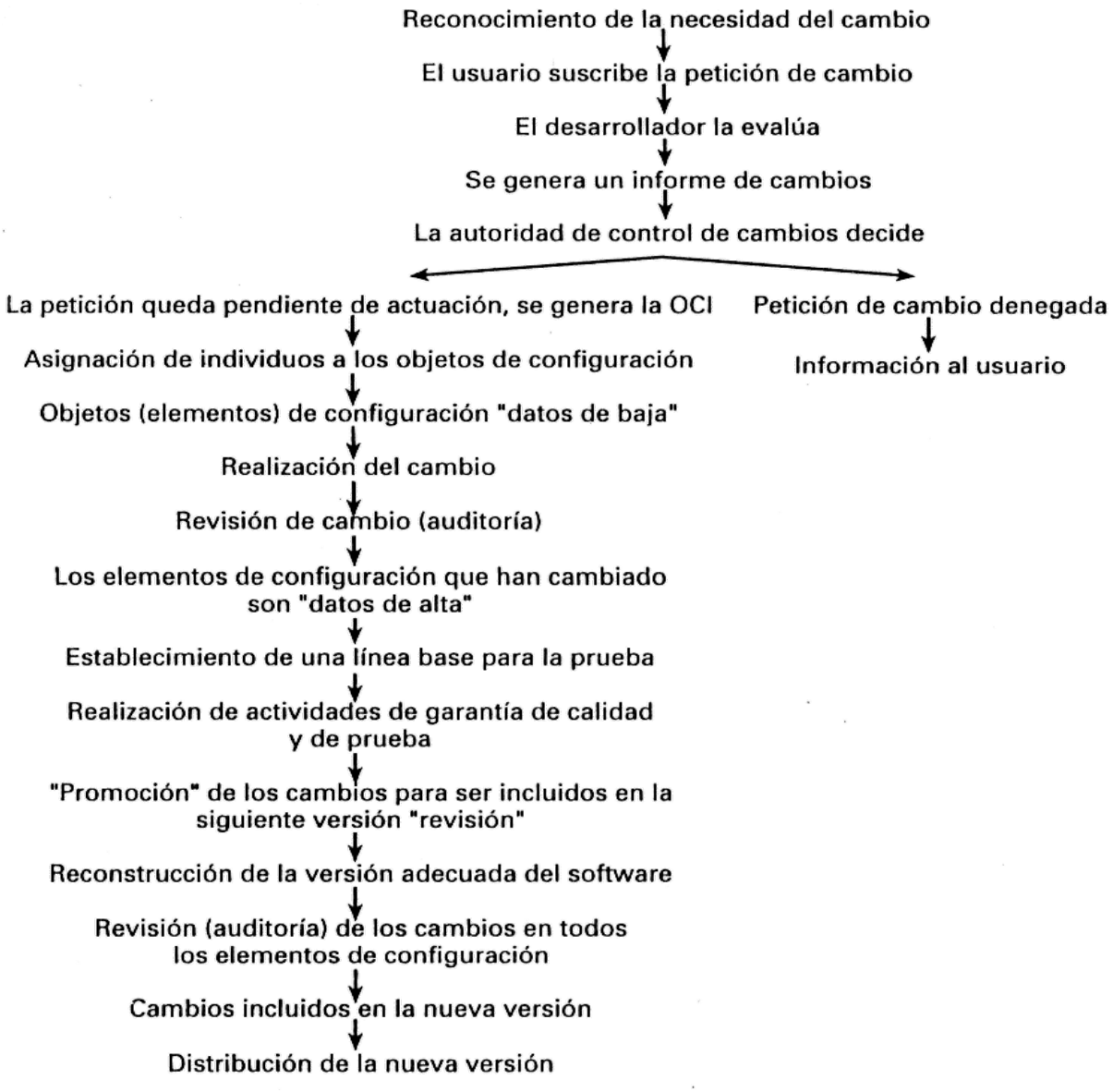
Combinación de procedimientos y herramientas para gestionar las versiones de los ECS que se crean a lo largo del proceso de software.
Ejemplo de versiones
Un programa puede contener los módulos 1-2-3-4-5
Una versión puede utilizar los módulos 1-2-3-5
Otra versión puede utilizar los módulos 1-2-4-5

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 :

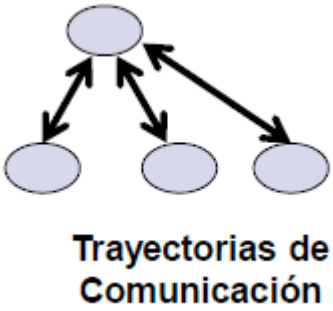
1. Atmósfera de trabajo frenética: El equipo tiene acceso a toda la información y las principales metas y objetivos no se modifican a menos que sea absolutamente necesario.
2. Fricción entre los miembros del equipo: se define responsabilidad a cada uno .
3. Proceso de software fragmentado o mal coordinado: 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

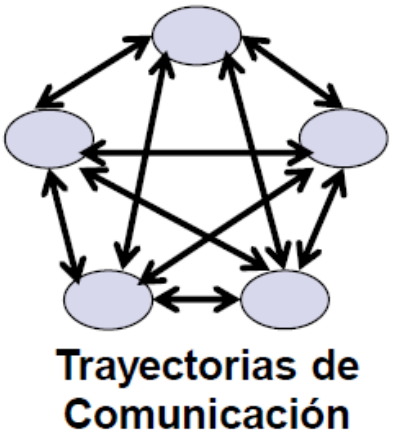
Paradigma cerrado

- 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.



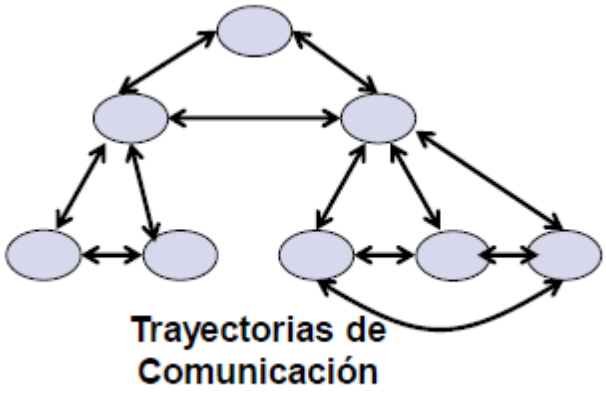
Paradigma abierto

- 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.



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.