**Resumen épico de ingeniería II**

## Proceso de software

Un proceso de software es un conjunto de actividades y resultados asociados que producen un producto de software. El modelo es una representación abstracta

La clase uno no dice nada más importante, habla sobre un repaso , es útil para lo que es la parte práctica que esta mos haciendo ahora

## Gestión de riesgos

Los gerentes deben determinar si pueden presentarse eventos no deseados durante el desarrollo o el mantenimiento y hacer planes para evitar estos eventos o minimizar sus consecuencias negativas si son inevitables.

Los riesgos pueden concernir a lo que ocurrirá en el futuro ( pueden hacer que fracase el proyecto ), a como afectaran en cuanto al desarrollo ( cambios de requisitos, en las tecnologías, el éxito global y los plazos ) o a las elecciones( métodos que debemos usar, gente involucrada, importancia de la calidad)

Se pueden tener estrategias **reactivas**, donde se reacciona ante el problema y gestionar la crisis, o **proactivas** donde tenemos estrategias de tratamiento

El riesgo esta definido por la pérdida(consecuencias) + la incertidumbre( probabilidad de que un riesgo suceda )

Los riesgos pueden ser del proyecto, del producto o del negocio, estos tres a su vez se dividen en 2 tipos, que pueden ser genéricos, es decir que aparecen en cualquier producto, o específicos, es decir que pertenecen a un dominio particular.  
  
Ahora estos dos tienen una subdivisión en conocidos ( por experiencia previa ) , predecibles, o impredecibles.

Proceso de gestión de riesgos:

* Identificación de riesgos: Cuales son “verdaderos riegos” o hacer una lista de comprobación de elementos de riesgo
* Análisis de riesgos: Cada riesgo identificado con su probabilidad e impacto, se construye una tabla de riesgos
* Planeación de riesgos: Tener estrategias, como evitar el riesgo, minimizarlo o tener un plan de contingencia. Hay que tener en cuenta el costo de cada estrategia
* Supervisión de riesgos: Monitorear los riesgos en todas las etapas. En cada revisión hay que reflexionar y estudiar cada riesgo por separado. También hay que decidir si es menos probable que surja el riesgo , o si cambiaron la gravedad y las consecuencias del riesgo.

## GCS Y planificación

La GCS se encarga de responder como identifica y gestiona una organización las diferentes versiones existentes de un programa de forma que se puedan introducir cambios eficientemente, como controla la organización los cambios antes y después de que el software sea distribuido y quien tiene la responsabilidad de aprobar y asignar prioridades a los cambios.

El proceso de la GCS tiene 5 partes

* Identificación: Se identifica cada elemento con un nombre sin ambigüedades junto con una descripción
* Control de versiones: Gestiona las versiones de los ECS que se crean a lo largo del proceso para recuperarlos.
* Control de cambios: Combina los procedimientos humanos y herramientas adecuadas para proporcionar un mecanismo para el control del cambio

La autoridad de control de cambios ( ACC ) evalúa:

¿cómo impactara el cambio en el hardware?

¿cómo impactara el cambio en el rendimiento?

¿cómo afectara el cambio a la calidad y fiabilidad?

* Auditorias de la configuración: Se fija:

¿se ha hecho el cambio especificado en la orden de cambio?

¿se han incorporado modificaciones adicionales?

¿se ha llevado a cabo una RTF para evaluar la corrección técnica?

¿se han seguido adecuadamente los estándares de IS?

¿se han reflejado los cambios en el ECS {fecha, autor, atributos}?

etc...

* Generación de informes: Responde que pasó , quien lo hizo, cuando y quien más se vio afectado. Tiene un papel vital en el éxito del proyecto

Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único también se podría considerar como la división de un trabajo en tareas mas sencillas para manejar la complejidad del software. Tiene como características que es temporal, tiene un resultado y su elaboración es gradual

La gestión de proyecto de software se basa en 4 aspectos: Planificación , dirección , organización y control.

La administración de un proyecto debe guiarse por las 4p:

* Personal: Quien realiza el proyecto, el mas importante. Se debe identificar interesados, determinar sus expectativas y habilidades
* Producto: Es intangible , a veces es difícil ver el progreso del proyecto por

lo que es imprescindible lograr un buen producto. Hay que planear el proyecto, establecer los objetivos y el ámbito del producto, considerar soluciones alternativas, etc...

* Proceso: Proporciona el marco de trabajo desde donde se puede establecer un plan detallado para el desarrollo del software
* Proyecto los proyectos deben ser planeados y controlados para manejar su complejidad (es la única forma conocida)

Para evitar el fracaso del proyecto, se deben evitar un conjunto de señales de advertencia que son comunes y entender los factores de éxito cruciales que conducen a una buena gestión del proyecto. En caso de que la cagues hay incumplimiento de plazos, un incremento de costos y una entrega de productos de mala calidad.

La gestión de proyectos cubre todo el proceso de desarrollo, desde el inicio hasta el entregado, incluso su mantenimiento

Planificación especifica: indica que debe hacerse , con qué recursos y en qué orden

Al hablar de "planificación" se hablan de 2 tipos:

* La organizativa, se relaciona con cómo vamos a gestionar los RRHH con los que contamos
* La temporal

**Planificación organizativa**

El personal {RRHH} es el activo más grande, representa el capital intelectual. Una mala administración del personal es uno de los factores principales para el fracaso por lo que la gestión de nuestro personal es un factor crítico para el éxito o el fracaso del proyecto.

Los participantes son: gerentes ejecutivos {define los temas empresariales}, el gerente de proyecto {planifican, motivan, organizan y controlan a los profesionales} , profesionales {aporta habilidades técnicas}, clientes {especifican los requerimientos} y usuarios finales { ̶i̶d̶i̶o̶t̶a interactúan con el software}.

El equipo se debe organizar de manera que maximice las habilidades y capacidades de cada persona.

Factores para considerar cuando se planea una estructura de equipo: Dificultad del problema a resolver, el tamaño de programa resultante , el tiempo que el equipo permanecerá unido, el grado en que puede dividirse en módulos el problema a resolver, la calidad y confiabilidad requerida por el sistema a construir, la rigidez de la fecha de entrega y grado de sociabilidad requerido para el proyecto

Un punto crítico es la comunicación : la comunicación en el grupo es prioritaria y puede llegar a ser influenciada por el estatus del grupo. Los programadores pueden mejorar la productividad si cuentan con un entorno de trabajo provisto con recursos necesarios y áreas de comunicación adecuadas

Hay 3 organigramas de equipo genérico:

* Descentralizado democrático DD {socialista}: No tiene jefe permanente , se nombran coordinadores de tareas a corto plazo, las decisiones se toman por consenso y la comunicación entre los miembros del equipo es horizontal. Es útil para tareas rápidas y maneja problemas sencillos.
* Descentralizado controlado DC {jerárquica}: Tiene un jefe definido que coordina tareas específicas y jefes secundarios para subtareas, la resolución de problemas es una actividad del grupo, pero la implementación de soluciones se reparte entre subgrupos y la comunicación es horizontal y vertical. Tiene más probabilidades de éxito en la resolución de problemas complejos.
* Centralizado controlado CC {monarca}: El jefe del equipo se encarga de la resolución de problemas a alto nivel y la coordinación interna del equipo. La comunicación entre el jefe y los miembros es vertical. Es la mejor para problemas difíciles.

## Planificación temporal

La planificación temporal es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto.

Hay dos casos diferentes, el primero es si hay fecha establecida por el cliente, y el segundo es cuando la fecha final esta fijada por los desarrolladores, siendo la primera la más frecuente.

La calendarización esta compuesta por **tareas**, que son acciones por realizar en un plazo determinado, **tareas críticas,** cuyo retraso genera un retraso en todo el proyecto, e **hitos** que es algo que se espera que este hecho para alguna fecha.

Cada tarea tiene 4 parametros:

* Precursor: Evento o conjunto de eventos que deben ocurrir antes de que pueda comenzar la actividad
* Duración: Cantidad de tiempo necesaria para completar la actividad
* Fecha de entrega: Fecha para cuando la actividad debe estar completada
* Resultado: Hito o componente listo

Es importante la definición de una **red de tareas**, que es una representación grafica del flujo de las tareas desde el inicio hasta el fin del proyecto, en algunos casos los conjuntos de tareas permiten hacer actividades en paralelo. Muestran la secuencia de las tareas y su interdependencia.

Hay dos métodos de planificación temporal

**Método PERT:** Se utiliza para controlar la ejecución de proyectos con gran numero de actividades que implican investigación, desarrollo y pruebas. Tienen una red de tareas con tiempo mas probable, optimista y pesimista. Es un método probabilístico.

**Método CPM:** Se utiliza en proyectos en los que hay poca incertidumbre en las estimaciones. Usa un tiempo de inicio temprano y tardío además de una red de tareas. Es un método determinístico.

Lo que se hizo fue unir ambos métodos y hacer un método nuevo llamado **Método del camino critico (PERT CPM)**

Propone seguir 6 puntos: Establecer la lista de tareas, fijar dependencia y duración de las tareas, construir la red de tareas, numerar cada nodo de la red, calcular la fecha temprana y tardía de cada nodo, y luego se calcula el camino critico que establece la red.

Del PERT- CPM se obtiene el camino critico, la ventana temporal para cada actividad, la fecha temprana y tardía de una tarea y el margen total.

## Métricas y estimaciones

La métrica es la clave tecnológica para el desarrollo y mantenimiento del producto y tiene como objetivo entender que ocurre , controlar que ocurre, mejorar el proyecto y evaluar la calidad.

Las métricas pueden ser utilizadas para que los profesionales o investigadores tomen mejores decisiones.

Las **métricas del proyecto** se usan para realizar ajustes en el calendario y evitar demoras, valorar el estado de un proyecto, rastrear riesgos, descubrir problemas ajustar el flujo y evaluar la habilidad del equipo.

Las **métricas del proceso** se usan para obtener retroalimentación, para establecer metas y métricas claras y la idea es no excluir métricas.

Las **métricas de un producto**  se dividen en métricas dinámicas y estáticas, las dinámicas están hechas en un programa en ejecución, valorando la eficiencia y fiabilidad, mientras que las estáticas están hechas en base a representaciones de un sistema, valorando la complejidad, comprensibilidad y mantenibilidad.

### Estimaciones

Las diferencias entre las métricas y estimaciones es que las estimaciones son aproximadas. Se debe tener cuidado de cuando se utilizan.

Existen las **estimaciones de recursos** que miden los recursos de personal, los recursos de entorno y los recursos de software reutilizable. Después están las **estimaciones de costo** que usan 3 parametros que es el esfuerzo, el hardware y software y los viajes. Y por ultimo las **estimaciones de tiempo.**

Hay técnicas de estimación, una es el juicio experto, otra la técnica Delphi y la ultima la división de trabajo. También hay modelos empíricos de estimación de que utilizan formulas especificas que predicen costos o esfuerzos requeridos en el proyecto.

## Conceptos de diseño

Es una representación de algo que se va a construir, es donde se unen los requisitos, las necesidades y las consideraciones técnicas se van a unir, se dice que es el núcleo técnico de la IGS. Es independiente del modelo de proceso utilizado. Es importante porque un problema se transforma en una solución, además que se puede evaluar la calidad. Con un mal diseño pueden haber fallos.

El **diseño de datos** es donde se transforma el modelo de dominio a estructuras de datos, relaciones, etc. El **arquitectónico** define la relación entre los elementos estructurales del software. El **a nivel de componentes** es cuando se transforman los elementos estructurales de la arquitectura de software en una descripción de los componentes del software. Y el de **interfaces** comprende como se comunica el sistema con si mismo y con el exterior.

Es importante evaluarlo porque debe implementar todos los requisitos explícitos y ajustarse a los implícitos. Debe ser una guía legible para los que den mantenimiento y código. Tiene que presentar una imagen completa del software.

La **abstracción** nos permite concentrarnos en un problema sin tener en cuenta los detalles de bajo nivel, el mas alto son los requisitos funcionales, el mas bajo el código.

La **arquitectura** es la estructura organizacional de los componentes de un programa y las formas en como interactúan entre si.

Los **patrones** son una solución a un problema recurrente dentro de un contexto dado, proporciona una descripción que determina si es aplicable, reutilizable o guía para otro patron.

La **modularidad** se basa en dividir al software en módulos para un mejor entendimiento, puede ser un software monolitico o modularizacion excesiva, la cantidad de un programa depende de la complejidad del desarrollo.

El **ocultamiento de información** es que un modulo oculta info para que otros no la usen, para ser eficiente hay que ver que info necesita cada modulo

La **independencia funcional** esta dado por la unión entre la modularidad, la abstracción y el ocultamiento. Esto se mide con la cohesión y el aplicamiento, se busca una alta cohesión y bajo acoplamiento.

La **cohesión** es la relación existente entre las sentencias o grupos de sentencias de un mismo modulo. Es poco coherente cuando realiza tareas diferentes entre si. La mas baja es la cohesión coincidental que es cuando no tienen relación entre si, la lógica es que lógicamente son iguales, temporal que se usan al mismo tiempo, procedimental es que tienen que seguir un orden especifico, comunicacional cuando se relacionan con datos de entrada y de salida y la funcional es cuando se relacionan pero realizan una única función ( es la cohesión mas alta ).

El **acoplamiento** es la medida de interconexión entre los módulos, se busca que los módulos casi no se comuniquen entre si, es decir bajo acoplamiento. El acoplamiento de datos(bajo) es donde los módulos interactúan mediante datos, el de marca(bajo) donde interactúan por estructura, el de control(medio) donde interactúan por un indicador, el común(medio) que usan variables globales, el externo(malo) que es por protocolos de comunicación y el de contenido(malo) cuando uno usa un dato dentro de otro modulo.

La abstracción permite especificar procedimientos y datos sin considerar detalles de grado menor, mientras que el refinamiento ayuda a revelar los detalles de grado menor mientras se realiza el diseño. Son complementarios.

## Diseño arquitectónico (momento arquitectura)

Define la relación entre los elementos estructurales para lograr los requisitos del sistema, es el proceso de identificar subsistemas dentro del sistema y establecer el control y comunicación entre ellos. El modelo arquitectónico presenta los componentes y los vínculos entre ellos.

Es fundamental porque afecta a los requerimientos no funcionales, los mas críticos son los siguientes, hay que orientar el sistema a ellos ( algunos chocan entre si ):

* Rendimiento: Se deben agrupar las operaciones en un grupo reducido de subsistemas
* Seguridad: Se debe utilizar una arquitectura en capas protegiendo los recursos mas críticos en capas internas
* Protección: Se deben diseñas las operaciones relacionadas a la protección en un único subsistema, permite reducir costos y problemas de validación
* Disponibilidad: Se diseña con componentes redundantes de forma que sea posible reemplazar uno sin detener el sistema, es muy tolerante a fallos
* Mantenibilidad: Se diseña con componentes autocontenidos que puedan modificarse sin problema

### Organización del sistema

La organización representa la estrategia básica para estructurar el sistema. Los subsistemas deben intercambiar información de forma efectiva, una forma puede ser compartir los datos en una bd central o cada subsistema tiene su propia info y la intercambia con otros.

Se utilizan patrones arquitectónicos para las formas de organizar el sistema.

**Patron de repositorio:** Los sistemas grandes se organizan alrededor de una bd compartida ( repositorio) , los datos se generan por un subsistema y utilizados por otros.

Es una forma eficiente de compartir grandes cantidades de datos, los subsistemas que producen datos no deben saber como se utilizan, las actividades backup, protección, control de acceso están centralizadas y el modelo compartido es visible a través del esquema del repositorio, además las herramientas se integran de forma directa ya que son compatibles con el modelo de datos.

Como desventajas cabe resaltar que los subsistemas deben estar acordes a los modelos de datos del repositorio, la evolución puede ser difícil a medida que se genera un gran volumen de información de acuerdo con el modelo de datos, son difíciles de migrar, los diferentes subsistemas pueden tener distintos requerimientos de protección pero el repositorio le da los mismos a todos y es difícil distribuir el repositorio en varias maquinas.

**Patron cliente servidor:** Es un modelo donde el sistema organiza como un conjunto de servicios y servidores asociados, mas unos clientes que utilizan los servicios. Los clientes conocen el nombre del servidor y el servicio que brinda, pero el servidor no necesita conocer al cliente.

**Patron de arquitectura en capas:** El sistema se organiza en capas, donde cada una de ellas presenta un conjunto de servicios a sus capas adyacente.

Como ventaja esta que soporta el desarrollo incremental, es portable y resistente a cambios, una capa puede ser reemplazada siempre que se mantenga la interfaz y si varia genera una capa para adaptarlas, y permite generar sistemas multiplataforma, ya que solo las capas mas internas son dependientes de la plataforma.

Es difícil de estructurar, las capas internas proporcionan servicios que son requeridos por todos los niveles. Los servicios requeridos por el usuario pueden estar brindados por las capas internas teniendo que atravesar varias capas adyacentes. Si hay muchas capas, un servicio de la capa superior puede tener que ser interpretado varias veces en diferentes capas.

### Descomposición modular

Se pueden dividir los subsistemas en módulos , se puede aplicar los mimos criterios que vimos en la descomposición pero esta es mas pequeña y permite utilizar otros estilos alternativos.

Un subsistema es un sistema cuyo funcionamiento no depende de los servicios proporcionados por otros. Los subsistemas se componen de módulos que se utilizan para comunicarse con otros subsistemas

Un modulo es un componente de un subsistema que proporciona uno o mas servicios a otros módulos. Por lo general no se los considera sistemas independientes.

La descomposición orientada a un flujo de funciones los datos fluyen de una función a otra y se transforman a medida que pasan por una secuencia de funciones hasta llegar a los datos de salida, se pueden ejecutar en secuencial o en paralelo.

La descomposición orientada a objetos estructura el sistema en un conjunto de modelos débilmente acoplados y con interfaces bien definidas.

### Modelos de control

Puede ser centralizado o basado(llamaste?) en eventos.

Si es centralizado un subsistema se diseña como controlador y tiene la responsabilidad de gestionar la ejecución de otros subsistemas, la ejecución puede ser secuencial o en paralelo. Se puede usar el modelo de llamada y retorno ( estilo cuoorita ) o el modelo de gestor

Si es dirigido por eventos se rige por eventos generados externamentes al proceso, puede ser una señal binario, un valor, un comando, etc. Puede ser un modelo de transmisión donde un evento se transmite a todos los subsistemas y cualquier subsistema programado para manejar ese evento lo atenderá, o un modelo de interrupciones donde se usan sistema en tiempo real donde las interrupciones son detectadoas por un manejador y ese invoca a un componente para que lo atienda.

### Arquitectura de los sistemas distribuidos

Un sistema distribuido es un sistema en el que el procesamiento de información se distribuye sobre varias computadoras.

Sus ventajas son que pueden compartir recursos, tienen una mayor apertura, concurrencia y escalabilidad, aparte de contar con una gran tolerancia a fallos. Como desventaja tiene que son complejos, tienen problemas de seguridad, no son manejables o mantenibles de forma sencilla y son impredecibles en cuanto a tiempos de respuesta a las peticiones.

En la arquitectura multiprocesador el sistema está formado por varios procesos que pueden o no ejecutarse en procesadores diferentes, la asignación de los procesos a los procesadores puede ser predeterminada o mediante un dispacher. Es común en sistemas grandes de tiempo real que recolectan información, toman decisiones y envían señales para modificar el entorno.

En la arquitectura cliente-servidor la aplicación se modela como un conjunto de servicios y un conjunto de clientes que usa estos servicios. Los clientes y servidores son procesos diferentes, un servidor puede atender varios clientes y brindar varios servicios, los clientes no se conocen entre si. También pueden ser de dos niveles donde los clientes pueden ser clientes livianos o pesados dependiendo de que implementen la lógica de la aplicación o no.