La Biblia de IS (I)

TEMA 0 -1 INTRODUCCIÓN Y EL PRODUCTO

Definiciones de IS:

- La ingeniería del software (IS) es una disciplina o área de la informática que ofrece métodos y técnicas para desarrollar software de calidad.
- IS es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y desarrollo del software; es decir, la aplicación de la ingeniería al software IEEE
- IS es la aplicación del sentido común al desarrollo de sistemas software. El sentido común es:
    1. Identificar los requisitos de la aplicación antes de planificarla
    2. Planificar antes de desarrollar
    3. Diseñar antes de programar
    4. Realizar diseños que funcionan y son fáciles de mantener.

        Antonio Navarro

El producto:

Hoy en día el software es el factor principal en el presupuesto de un proyecto industrial; por las siguientes razones:

- Si hacemos un software con un alto tiempo de desarrollo, incluso fuera de plazo; esto tiene un coste elevado
- El software puede estar entregado a clientes con errores (defectos), por lo que hay que tratar de evitarlo
- Falta de relación entre el trabajo realizado y la escritura total del software

El software no se estropea; se deteriora, y ocurren las siguientes desgracias:

- El software deteriorado no se puede reparar, pues es imposible revisar miles de líneas de código
- Muchas veces estas reparaciones dañan más al software

Por tanto, el software debe estar bien diseñado para facilitar su evolución, y debe ser:

- Mantenible
- Seguro
- Eficiente
- Amistoso

TEMA 2, EL PROCESO Y EL MODELO DE EQUIPO

La IS como tecnología estratificada

IS se trata de un establecimiento de principios y métodos de ingeniería con el fin de obtener software de manera rentable, fiable y que trabaje en máquinas rentables.

Está compuesta por una tecnología multicapa:

- Capa de calidad
- Capa de proceso
    - Desarrollo racional de la IS
    - Conjunto de actividades y resultados que sirven para construir un producto software
- Capa de métodos
    - Un método incluye:
        - Análisis de requisitos
        - Diseño
        - Construcción de programas
        - Prueba
        - Mantenimiento
- Capa de herramientas
    - Herramientas CASE (como IBM/RSAD)
    - Fundamental para la correcta aplicación de IS

Fases en IS

- Fase de definición:
    - Se centra en el *qué*
    - **Tareas principales**: realizar el Plan de Proyecto, análisis de requisitos y la ingeniería de sistemas o de información
    - Se identifican los requisitos del sistema y software
        - Función y rendimiento que se desean

        - Comportamiento del sistema
        - Interfaces
        - Restricciones de diseño

- Fase de desarrollo:
    - Se centra en el cómo
    - **Tareas principales**: diseño del software, generación del código y pruebas del software

- Fase de mantenimiento:
    - Se centra en el cambio asociado a corrección de errores, adaptaciones requeridas por la evolución del entorno software y cambios en los requisitos del cliente
    - Se vuelve a aplicar las fases de definición y desarrollo sobre software existente
    - Puede haber 4 tipos de cambio: corrección, adaptación, mejora y prevención

EL PROCESO

El proceso es el conjunto de actividades y resultados asociados que sirven para construir un producto software.
Hay que definir un marco de trabajo compuesto por un conjunto de **Actividades Estructurales (AE).**
Estas AE se pueden dividir en acciones y éstas en tareas. Por ejemplo, en un proyecto se puede dividir las AE Ingeniería en Reingeniería, Análisis y Diseño; el análisis a su vez se podría dividir en tareas como realizar diagramas de uso.
Por otro lado, tenemos las **actividades de protección**, que están presentes en las AE y por eso no hay que hacer una planificación temporal de las mismas; son:

- Seguimiento y control del proyecto
- Revisiones Técnicas Formales (RTF, tema de la calidad)
- Garantía de calidad del software (tema de calidad)
- Gestión de Configuración Software (GCS, tema de cambios)
- Preparación y producción de documentos
- Gestión de riesgos (tema 7)
- Mediciones

Modelos de Proceso

Un modelo de proceso o paradigma de IS es una plantilla, patrón o marco que define el proceso a través del que se crea software. Es una representación

abstracta de un proceso software que define las actividades estructurales y el flujo de realización.
Tenemos tres clases de modelos:

Modelos clásicos:

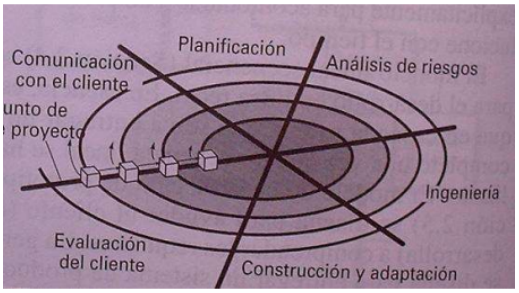
Surgieron en una primera etapa, y en ellos podemos encontrar el modelo en cascada, el modelo de construcción de prototipos y el modelo de desarrollo rápido de aplicaciones.
Modelo en cascada: muy probado, sencillo y fácil de entender. El proyecto pasa por una serie de **fases,** en las que no hay solapamiento y se puede llegar a bloqueos. Al final de cada una se revisan las tareas de trabajos y productos. Es poco realista y un error en las últimas fases puede ser fatal.
El equipo que le corresponde es el CC.

Ventajas	Desventajas
Sencillo y fácil de entender Muy probado	No hay solapamiento en fases y puede haber bloqueos. Poco realista Un error al final puede ser fatal Poca visibilidad

Modelos evolutivos:

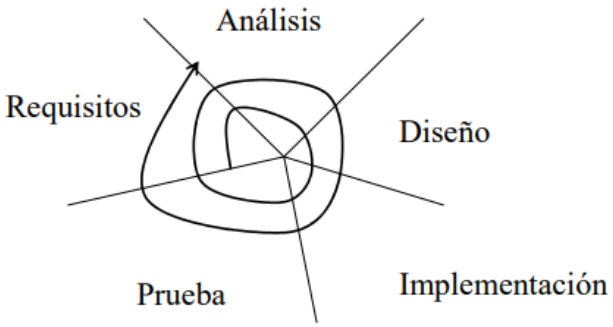
Modelos iterativos (construyen software siguiendo una serie de pasos) que gestionan bien la evolución del software y se basan en incrementos (productos operacionales definitivos de una parte del sistema).
Pueden producirse **prototipos;** forma o instancia preliminar de un sistema que sirve como modelo para fases posteriores o la versión completa. Se suele usar en la etapa de identificación de requisitos para minimizar el riesgo.

Espiral de Boston: se caracteriza por la vida de un sistema. Se ven resultados en cada iteración lo que implica mayor visibilidad. Proporciona un enfoque realista y garantiza actividades de protección continuas. Se usa en tipos de proyecto de desarrollo de nuevos productos, mejora y mantenimiento de los mismos.



Ventajas	Desventajas
Enfoque realista Actividades de protección continuas Mayor visibilidad Adecuado para programación OO	Convencer al cliente de que es un proceso con un enfoque controlable

Modelo unificado de desarrollo (RUP): modelo muy ligado al UML (el de nuestro proyecto) y basado en interfaces y componentes. Fue creado por Booch, Jacobson y Rumbaugh. Está dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental.
Flujos de trabajo:



Cada vuelta en la espiral se denomina iteración.
Un conjunto de iteraciones se llama fase.

- Fase de inicio: se describe el producto final
- Fase de elaboración: se especifican los casos de uso y se diseña la arquitectura del sistema
- Fase de construcción
- Fase de transición: periodo donde el producto se convierte en versión beta

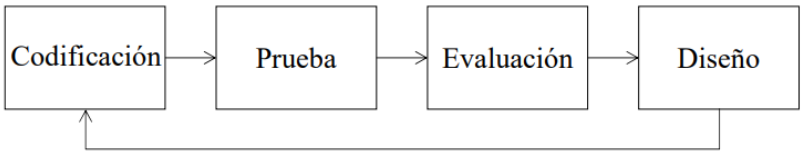
No todos los flujos de trabajo tienen el mismo peso dentro de cada fase.
Las agrupaciones de fases se denominan ciclo, y por cada ciclo se presenta una versión del producto.

Ventajas	Inconvenientes
Muy racional Tecnologías de componentes	Muy ligado al método

Modelos de desarrollo ágil:

Valoran al individuo sobre los procesos, que el software funcione sobre la implementación, y la respuesta al cambio sobre seguir un plan (entre otros).
El proceso ágil requiere una serie de características: Competencia técnica, Enfoque común: entregar al cliente un incremento dentro del plazo, Colaboración entre todos los participantes, Autonomía para la toma de decisiones, Capacidad de resolución de problemas confusos, Confianza y respeto mutuo en el equipo, Organización propia.

XP: es ligero, eficiente, bajo en riesgos, flexible, científico y predecible. Tiene una visibilidad altísima por sus ciclos cortos, se adapta bien a los cambios de negocio por su alta comunicación y busca un equilibrio entre las necesidades a corto plazo de los programadores y las de largo plazo del proyecto. Solo funciona para grupos de no más de 10 personas y no es compatible con especificaciones totales.



Actividades en XP

Ventajas	Desventajas
Muy bueno para especificaciones cambiantes y fundamentación práctica	Poco compatible con especificaciones y diseños totales. Grupos de menos de 10 personas.

Tipos de equipo:

Estas son las organizaciones de equipos Mantei:

Descentralizado Democrático:

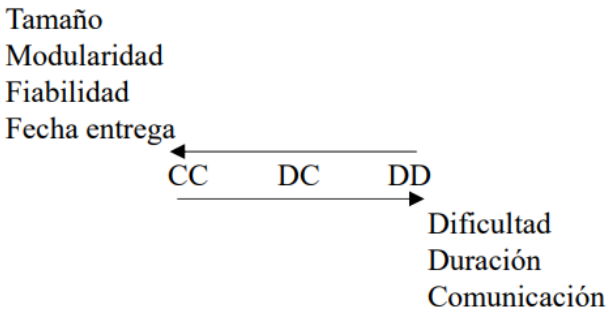
- No tiene jefe permanente
- Se nombra un jefe en función de cada tarea
- Decisiones, problemas y enfoques se llevan a consenso del equipo
- Comunicación horizontal entre los miembros
- Se usa principalmente con procesos ágiles

Descentralizado Controlado:

- Jefe encargado de distribuir las tareas entre los subgrupos, dirigidos por jefes secundarios
- Resolución de problemas se hace en grupo
- Comunicación horizontal entre los subgrupos y los individuos.
  Comunicación vertical entre el jefe y el subjefe
- Se usa principalmente con modelos evolutivos

Centralizado Controlado:

- Único jefe de equipo que resuelve los problemas a alto nivel y coordina al equipo
- Comunicación vertical entre el jefe y los miembros
- Se usa con los modelos clásicos, el de cascada



	Dificultad		Tamaño		Duración		Modularidad		Fiabilidad		Plazo		Sociabilidad	
	Alta	Baja	Grande	Pequeño	Alta	Baja	Alta	Baja	Alta	Baja	Estricto	Laxo	Alta	Baja
DD	X			X		X		X	X			X	X	
DC		X	X		X		X		X			X		X
CC		X	X		X		X			X	X			X

TEMA 3: Gestión de proyectos

La gestión eficaz de proyectos software se basa en personal, producto, proceso y proyecto.
Personal ya lo hemos definido antes; así que vamos con el resto.

Gestión del producto

El jefe de proyecto tiene un dilema al principio del mismo; pues necesita planificarlo y no tiene información suficiente. Podría pensar que la especificación de requisitos es la solución, pero aún no está disponible y el plan de proyecto hace falta ya. Así que la única solución que le queda es determinar el **ámbito del software.**

1. Ámbito del software es lo primero que hay que hacer en un proyecto software; y es:
   - Contexto:
     - Forma de integrarse el software en el sistema, producto o contexto de negocios mayor
     - Limitaciones resultantes de eso
   - Objetivos de información:
     - Datos visibles al cliente
     - Datos de entrada requeridos
   - Funciones
     - Funciones realizadas por el software para transformar la información de la entrada en la salida
     - Características especiales de rendimiento

En esta fase se lleva a cabo la partición horizontal; y se descompone en sistema en módulos, que son agrupaciones de funciones que llevan a cabo tareas de naturaleza similar.
El cliente es el único que puede ayudarnos a determinar el ámbito. La comunicación con él puede empezarse con las llamadas preguntas de contexto libre (para romper el hielo); y hay 3 tipos de preguntas.
El primer tipo se centra en el cliente y los objetivos (¿quién va a usarlo?), el segundo grupo permite comprender mejor el problema y que el cliente trate de resolverlo (¿con qué problemas se encontrará esta solución?). El último tipo se centra en la efectividad de la reunión (¿Es usted el adecuado para responder esto?).
Después de estas preguntas para romper el hielo deberían utilizarse técnicas de especificación más concretas, como las TUE, que son reuniones entre clientes y desarrolladores con el objetivo de identificar el problema, proponer elementos de la solución y especificar requisitos. Hacen falta varias TUE para conseguir la especificación concreta de los mismos.

2. Descomposición del problema : está muy ligada al análisis de requisitos y se lleva la partición vertical; llevando la descomposición al nivel de funciones. En esta etapa se pueden definir también nuevos submódulos.

Gestión del proceso

El problema que tiene aquí el jefe es qué modelo de proceso de elegir, para lo que lo escogerá en función al cliente, las características del producto y el equipo de desarrollo, para posteriormente comenzar con la descomposición del proceso.

1. Descomposición del proceso: las AEs deben descomponerse en acciones, y éstas engloban una serie de tareas de IS.
Por ejemplo, la AE ingeniería puede descomponerse en acciones Análisis y Diseño, y el análisis puede englobar tareas como hacer diagramas de casos de uso, por ejemplo.
Estas tareas las aplicaremos a nuestro grupo mediante la Estructura de Descomposición del Trabajo o EDT.
La EDT determina las tareas de trabajo concretas que se asignan a individuos concretos con plazos concretos con el fin de desarrollar un producto software.
Dicha asignación de individuos y plazos es la planificación temporal
Para hacer la EDT hay 2 opciones: o seguir el modelo de estructura de árbol, donde las hojas son actividades y tareas concretas o seguir la aproximación de Pressman, es decir; una tabla en la que las columnas son las AEs, en las filas los módulos y en las casillas de intersección las tareas concretas.

AE	Comunicación Cliente		Planificación		Análisis de riesgos		Ingeniería		Construcción y adaptación			Evaluación Cliente	
Acción	TUE	SRS	Estim.	Planif.	Valor	RSGR	Anal.	Diseño	Codif.	Prueba	Ensam.	Instal.	Eval.
Módulo dibujo													
Módulo transformaciones									Ana, Luis, Paco 16.11.20 19.11.20 Código m. transf.				
Módulo archivo													
Módulo impresión													

Gestión del proyecto

Para gestionar un proyecto debemos comprender qué puede ir mal.
Un tal Reel define 10 señales que son indicios de que el proyecto está en peligro:

- Desarrolladores no entienden las necesidades del cliente
- El ámbito del producto está definido terriblemente
- Cambios mal realizados
- Tecnología elegida cambia
- Necesidades del negocio cambian
- Fechas de entrega no realista
- Usuarios se quejan
- Se pierden recursos económicos
- Jefe y equipo no aplican las prácticas de IS

Estos proyectos mal gestionados cumplen la regla del 90-90:

El primer 90% de un sistema absorbe el 90% del tiempo y esfuerzo asignado. El último 10% se lleva el otro 90% del esfuerzo y tiempo asignado.

Para solucionar esto se aplica el sentido común: empezar con el pie derecho, saber mantenerse, hacer un seguimiento del proceso, tomar decisiones inteligentes y realizar un análisis al finalizar el proyecto.

TEMA 4: Métricas

Definiciones:

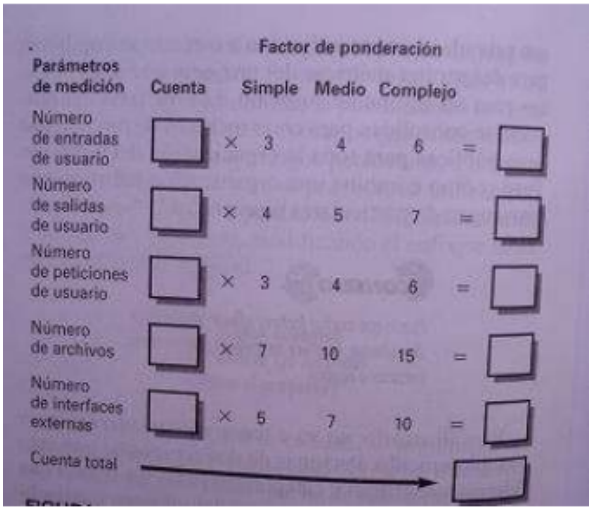
Una **medida** proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o proyecto.
Una **métrica** es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.
Es decir,
– La medida captura una característica individual
– La medición permite capturar dicha característica
– La métrica permite relacionar y comparar mediciones

Fórmulas:

MÉTRICAS DEL SOFTWARE	Productividad	Calidad
Tamaño	euros / LDC ; pgDoc / KLDC	errores / KLDC ; defectos / KLDC
PF (PF, PC, PF3D)	euros / PF ; pgDoc / PF	errores / PF ; defectos / PF
Otras	LDC / per-mes ; PF / per-mes ; euros / pgDoc	errores / per-mes ; e / e+d

Punto de Función (PF):

El PF se deriva de las mediciones del software.
Su cálculo es PF = cuenta_total * (0,65 + 0,01 * total_Fi*)
Cuenta total : (ver foto)



Parámetros de medición	Cuenta		Factor de ponderación Simple	Medio	Complejo	
Número de entradas de usuario		×	3	4	6	=
Número de salidas de usuario		×	4	5	7	=
Número de peticiones de usuario		×	3	4	6	=
Número de archivos		×	7	10	15	=
Número de interfaces externas		×	5	7	10	=
Cuenta total						

* Los valores de ajuste complejidad (Fi) se calculan respondiendo a las siguientes preguntas en una escala de 0 (no importante o aplicable) a 5 (absolutamente esencial)

1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?
2. ¿Se requiere comunicación de datos?
3. ¿Existen funciones de procesamiento distribuido?
4. ¿Es crítico el rendimiento?
5. ¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?
6. ¿Requiere el sistema entrada de datos interactiva?
7. ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?
8. ¿Se actualizan los archivos maestros de forma interactiva?
9. ¿Son complejas las entradas, las salidas, los archivos o las peticiones?
10. ¿Es complejo el procesamiento interno?
11. ¿Se ha diseñado el código para ser reutilizable?
12. ¿Están incluidas en el diseño la conversión e instalación?
13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14. ¿Se ha diseñado la aplicación para facilitar los cambios y ser fácilmente utilizada por el usuario?

PC

Consiste en ampliar la cuenta total de PF con el parámetro de medición algoritmos:

- Un algoritmo es un problema de cálculo limitado que se incluye dentro de un programa

- El factor de ponderación depende de la importancia que se quiera dar a este parámetro (e.g. 10, 15, 20)

Medidas de calidad:

La Eficacia de la Eliminación de Defectos (EED) indica la capacidad de filtro de las actividades de garantía de calidad y de control, al aplicarse a todas las actividades del marco de trabajo del proceso.

EED = e/(e + d) e = errores d = defectos

TEMA 5: Estimaciones

Estimación del proyecto del software:

Tenemos varias técnicas de estimación: retrasar la estimación lo máximo posible, utilizar el coste de proyectos similares ya terminados, estimar el coste en todo lo que el cliente pueda gastar; pero las más importantes son las de descomposición.

Descomposición del problema: Se basa en la descomposición del producto en funciones.

Con LDC: El procedimiento es el siguiente:

1. Estimar por cada función sus líneas de código óptimas, realistas y pesimistas en base a experiencia, intuición o datos históricos
2. Calcular el valor esperado VE = [Vo + (4Vm +Vp)]/6
3. Aplicar los datos históricos ya conocidos de productividad.

Función	Voptimista	Vmás probable	Vpesimista	VEsperado
MD	8500	10500	13800	10717
MT	11000	15000	17200	14700
MA	6200	7700	8500	7583
MP	4000	5900	7400	5833
I	2700	3500	5000	3617
				total: 42450

Un estudio dice que la productividad media para este tipo de sistemas es de 650 LDC/pm y una tarifa laboral de 7800€/pm
Realizando cábalas sencillas:

esfuerzo = 42450 (LDC) / 650 (LDC/pm) = 65 pm
coste = 42450 LDC * 7800 €/pm = 507000€

Con PF: el procedimiento es el siguiente:

1. Estimar los parámetros de medición (valor optimista, medio y pesimista)
2. Sacar el Valor Esperado con la siguiente fórmula VE = [Vo + (4Vm +Vp)]/6
3. Multiplicar el VE x el factor de ponderación correspondiente (simple, medio complejo; ver en tema 4 PF)
4. Sumar a este total el valor total de ajuste de complejidad (Fi, ver tema 4 PF)
5. Aplicar los datos ya conocidos de productividad

parámetro medición	Voptimista	Vmás probable	Vpesimista	VEsperado	peso medio	
#entradas	22	27	30	27	4	108
#salidas	15	20	29	21	5	105
#peticiones	17	25	28	23	4	92
# archivos	4	4	5	4	10	40
#interfaces externas	2	2	3	2	7	14
						total:359

F1: 4; F2: 2; F3: 0; F4: 4; F5: 4; F6: 5; F7: 4;
F8: 3; F9: 5; F10: 5; F11: 4; F12:3; F13: 5; F14: 5

$PF = total * (0,65 + 0,01*\Sigma_{i=1..14}F_i)$
$PF = 359 * (0,65 + 0,01*53) = 424$ (PF)

Una revisión de datos históricos indica que la productividad media de la organización para este tipo de sistemas es de 6 (PF/pm), y una tarifa laboral de 7800 (€/pm)

esfuerzo = 424 (pf) / 6 (pf/pm) = 71 pm
coste = 424 (pf) * 7800 (€/pm) = 553800€

Descomposición del proceso Es la más común; y utiliza la EDT ya empleada anteriormente. El procedimiento es el siguiente:

1. Sacar el esfuerzo por cada tarea de la EDT; es decir, multiplicar el número de personas que van a trabajar por el tiempo que lo van a hacer
2. Sumar todos los esfuerzos para sacar el esfuerzo total
3. Emplear los datos históricos de coste

A.E. →	Com. cli.	Plan.	A. riesgo	Ingeniería		Const. y adapta.		Ev. cli.	esf. total
acc. →				Análisis	Diseño	Codific.	Prueba		
función									
MD				1	6,5	2	5	no esfuerzo	14,5
I				0,25	3	1	1,25		5,5
MT				2	9,5	3	6,5		21
MA				0,75	4,5	1,75	3,5		10,5
MP				0,25	4	0,75	3		8
esf. total	0,5	0,5	0,5	4,25	27,5	8,5	19,25		**61(pm)**

La tarifa laboral es de 7800 €/pm

coste = 61 pm * 7800 €/pm = 475800€

Nótese que aunque sea el mismo proyecto, con LDC hemos sacado que el esfuerzo es 65pm, con PF: 71pm, y ahora 61pm, siendo la media por tanto 65,7 (pm) y la variación máxima un 8%, lo que es RAZONABLE (franja del 20%)

TEMA 6: Requisitos

Hay dos tipos de requisitos:

- **Requisitos de usuario**: frases en lenguajes natural junto a diagramas de los servicios que el sistema debe proporcionar, así como las restricciones bajo las que debe operar
- **Requisitos de sistema**: determinan los servicios del sistema y las restricciones en detalle; y sirven como detalle. Éstos a su vez se dividen en otros dos (full importante:
    - Requisitos funcionales: describen las funciones del sistema, la respuesta ante determinadas entradas, el comportamiento, y en algunos casos, lo que no debería hacer
    - Requisitos no funcionales: restricciones de los servicios o funciones que ofrece el sistema

Un ejemplo de esto podría ser lo siguiente:

- El sistema debe poder hacer préstamos (requisito de usuario)

- Función préstamo: entrada código_ejemplar, salida fecha_devolución… (requisito del sistema)
- La biblioteca debe ser capaz de atender a 100 usuarios simultáneamente (requisito no funcional)

Proceso de ingeniería de requisitos

Tiene distintas fases, y cada una te saca un documento distinto (más papeleo que para sacarse el bono cultural)

1. Estudio de viabilidad —--> Informe de viabilidad
2. Obtención y análisis de requisitos —--> Requisitos
3. Especificación de requisitos —--> Requisitos especificados
4. Validación —--> Requisitos validados

Estudio de viabilidad:

- Se origina tras una especificación preliminar de requisitos
- Es una estimación sobre si se puede resolver el problema del usuario con las tecnologías software y hardware disponibles
- El estudio decide si el sistema es rentable y satisface las restricciones económicas

Obtención y análisis de requisitos:

- Interacción con los *stakeholders* (clientes, usuarios…) para determinar el dominio de la aplicación y los requisitos funcionales y no funcionales.
- El análisis de requisitos tiene los pasos siguientes:
    1. Descubrimiento de requerimientos
    2. Clasificación y organización de requerimientos
    3. Ordenación por prioridades y negociación de requerimientos
    4. Documentación de requerimientos

Especificación de requisitos:

- Descripción detallada y precisa de los requisitos, de tal forma que sirva como contrato entre desarrollador y cliente
- El objetivo es realizar la Especificación de Requisitos Software (SRS)
- Pueden usarse distintos tipos de lenguajes para especificar los requisitos del sistema: natural, de descripción de diseño, notaciones gráficas… Pero el más importante y el que se recomienda usar es el **natural estructurado**

Validación de requisitos:

- Se encarga de comprobar si los requisitos se ajustan a los deseos de los stakeholders
- Si la validación es inadecuada saldrán errores en diseño e implementación
- Fuerte repercusión sobre el coste

- Hay distintas formas o técnicas de validar; como generar prototipos, revisión sistemática de los requisitos o generación de casos de prueba
- Hay distintos tipos de validez: validez, consistencia, realismo, completitud y verificabilidad

Gestión de requisitos

Como los requisitos pueden evolucionar y cambiar, es necesario tener una gestión de éstos.
La gestión de requisitos es el proceso de entender y controlar los cambios en los requisitos del sistema. Tiene en cuenta aspectos técnicos no contemplados en la Gestión de la Configuración Software (GCS).
Desde un punto de vista evolutivo, los requisitos pueden ser:

- **Estables**: salen de la actividad principal del sistema y están directamente relacionados con el dominio
- **Volátiles**: probablemente cambiarán durante el desarrollo o la entrega del proyecto, y se dividen en:
    - Mutables: cambian debido a cambios en el entorno en el que la organización opera
    - Emergentes: aparecen durante el desarrollo del sistema por el mayor entendimiento por parte del cliente
    - De consecuencia: aparecen por la introducción del sistema
    - De compatibilidad: dependen de los procesos de negocio de la organización

En la gestión de requisitos debemos ser capaces de:

- Identificar los requisitos
- Tener un proceso de gestión del cambio
- Disponer de políticas de traza
- Disponer de soporte CASE

Identificar requisitos: se supone al tener una SRS

Proceso de gestión del cambio: es común al de cualquier Elemento de Configuración Software (ECS)

Políticas de traza deben llevar cuenta de diversa información de traza:

- De origen: liga los requisitos al stakeholder
- De requisitos: señalas las dependencias entre requisitos
- De diseño: liga los requisitos a los módulos de diseño

La información de traza de requisitos puede mostrarse en una matriz de traza

Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			R		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

D: depende de
R: relacionado con

Herramientas CASE: facilitan el almacenamiento de requisitos, la gestión de cambios y la gestión de la traza. Pueden ser herramientas específicas o gestionar el cambio de la SRS como cualquier otro ECS

El resto del tema es el Std. 830-1998 y el UML que le pueden dar bien por culo

TEMA 7: Análisis y gestión de riesgos

Riesgo es todo aquello que pueda afectar negativamente al proyecto software.

Tipos de estrategias de gestión:

Estrategia reactiva:

- se supervisa el proyecto en función de los riesgos
- se asignan recursos por si surgen problemas
- no se prioriza el riesgo hasta que algo vaya mal
- el equipo intenta solucionar el problema pero en el caso de que no se pueda se llevará a cabo la gestión de crisis

Estrategia proactiva: es el que hemos usado en el proyecto

- Comienza antes que los trabajos técnicos
- Se identifican riesgos potenciales
- Se evalúa probabilidad y consecuencia de los riesgos
- Se priorizan los riesgos
- Se produce un plan de gestión de riesgo y planes de contingencia

Tipos de riesgos

Del proyecto: amenazan al plan de proyecto

- Si se hacen realidad aumentan el esfuerzo y el coste
- Identifican problemas potenciales en presupuesto, planificación, personal, recursos, participantes y requisitos

Técnicos: amenazan la calidad del software

- Aparecen porque el sistema pueda ser más difícil de construir de lo esperado
- Identifican riesgos potenciales en requisitos, diseño, implementación, interfaz, verificación, mantenimiento…

Del negocio: amenazan la viabilidad del proyecto

- Identifican problemas en: construir un producto que no encaja con la compañía, construir un sistema no necesitado, construir un producto difícil de vender, perder la confianza de los jefes…

Otra posible división de los riesgos sería conocidos (el personal es consciente), desconocidos (el personal sería consciente si se siguiera un proceso de identificación) e impredecibles (en principio no eran esperables)

Gestión de Riesgo Boehm

1. Valoración del riesgo
   • Identificación del riesgo
   • Análisis del riesgo
   • Priorización del riesgo

2. Control del riesgo
   • Planificación de la gestión del riesgo
   • Resolución del riesgo
   • Monitorización del riesgo

Valoración del riesgo

Identificación del riesgo:

- La identificación del riesgo produce listas de elementos del riesgo específicos para el proyecto que comprometan seriamente el éxito
- Una técnica para la identificación es el uso de listas de comprobación

- Otra técnica es el análisis de supuestos (comparación)
- La tabla de los 10 riesgos más comunes de Boehm puede ser una forma de comprobarlo

Análisis del riesgo

- Determina la probabilidad y consecuencias asignados a cada riesgo
- La probabilidad indica la la probabilidad de que el riesgo se haga real
- Las consecuencias indican la gravedad de las consecuencias si el riesgo se hace real
- Para analizar esto vamos a usar la tabla SQAS-SEI

Probability	Description	Severity	Consequence
Frequent	Not surprised, will occur several times (Frequency per year > 1)	Catastrophic	Greater than 6 month slip in schedule; greater than 10% cost overrun; greater than 10% reduction in product functionality
Probable	Occurs repeatedly/ an event to be expected (Frequency per year 1-10^{-1})	Critical	Less than 6 month slip in schedule; less than 10% cost overrun; less than 10% reduction in product functionality
Occasional	Could occur some time (Frequency per year 10^{-1}-10^{-2})	Serious	Less than 3 month slip in schedule; less than 5% cost overrun; less than 5% reduction in product functionality
Remote	Unlikely though conceivable (Frequence per year 10^{-2}-10^{-4})	Minor	Less than 1 month slip in schedule; less than 2% cost overrun; less than 2% reduction in product functionality
Improbable	So unlikely that probability is close to zero (Frequency per year 10^{-4}-10^{-6})	Negligible	Negligible impact on program

Priorización del riesgo

- La priorización de riesgos genera una lista ordenada de elementos de riesgo identificados y analizados
- La primera opción de priorización es crear una tabla de riesgo que los ordena en función de 1º su probabilidad y 2º su consecuencia
- La segunda opción es calcular la exposición al riesgo, multiplicando probabilidad por consecuencias
- La tercera opción es usando la matriz SQAS-SEI

Probability / Severity	Frequent	Probable	Occasional	Remote	Improbable
Catastrophic	IN	IN	IN	H	M
Critical	IN	IN	H	M	L
Serious	H	H	M	L	T
Minor	M	M	L	T	T
Negligible	M	L	T	T	T
LEGEND	T=Tolerable	L= Low	M= Medium	H=High	IN=Intolerable

Control del riesgo:

Planificación:

- Convierte la información sobre riesgos en decisiones y acciones de presente y futuro
- La planificación incluye:
    - Desarrollar acciones para controlar riesgos individuales
    - Priorizar las acciones contra los riesgos
    - Crear un Plan de Gestión de Riesgos
- Se analiza la lista ordenada de riesgos y se decide como responder ante ellos:
    - Evitar el riesgo: elegir una alternativa de menor riesgo
- Controlar el riesgo
    - Asumir el riesgo
    - Transferir el riesgo: reducir el riesgo compartiéndolo
- Se deben evaluar todas las posibilidades teniendo en cuenta costes y planificación
- Para aquellos que se controlen, se debe especificar los mecanismos de reducción
- También deben proponerse planes de contingencia

Resolución y monitorización:

- Durante la resolución se lleva a cabo los pasos identificados para reducir y controlar los riesgos
- Durante la monitorización se:
  - Comprueba si están llevando a cabo los pasos de reducción del riesgo
  - Comprueba si los riesgos se están haciendo reales
  - Llevan a cabo las actividades correctivas necesarias

Plan RSGR

- La información esencial sobre el proceso de gestión del riesgo puede incluirse en un Plan de Reducción, Supervisión y Gestión del Riesgo (RSGR)
- Éste puede ser un documento independiente, o ser parte del plan del proyecto del software

Reducción: se proponen pasos de reducción para

- Evitar que el riesgo se convierta en realidad
- Tener soluciones (back-up) en caso de que el riesgo se convierta en realidad

Supervisión; se:

- Controla si el riesgo se ha hecho real
- Supervisa la efectividad/implementación de los pasos de reducción

Gestión: aquí el riesgo se ha hecho real y se aplican las soluciones back-up (planes de contingencia) que hemos preparado en la reducción de riesgos.

Principio de Pareto

"El 80% del riesgo real se debe al 20% de los riesgos identificados"

Tema 8

Es literalmente Gantt y el PdP, paso.

Tema 9: Garantía de calidad

Definición y tipos

Definimos calidad como concordancia con los requisitos funcionales y de rendimiento explícitamente definidos, los estándares de desarrollo explícitamente documentados y las características implícitas de todo software desarrollado profesionalmente.
Hay dos tipos de calidad:

- De diseño: características especificadas para un elemento software (SRS y Diseño)
- De concordancia: grado de cumplimiento de las especificaciones de diseño durante su realización (software: implementación)

En IS hay dos formas de conseguir calidad:

- Haciendo SRS, diseños e implementaciones correctos desde un punto de vista técnico
- Introduciendo en el modelo de proceso una serie de actividades que garanticen que todas las entregas resultantes de una actividad de trabajo sean correctas.

Las técnicas de IS para conseguir calidad en el software se denominan Garantía de Calidad del Software (SQA)
La SQA engloba

- Enfoque de gestión de calidad
- Revisiones Técnicas Formales (RTFs)
- Estrategia de pruebas

- Control de la documentación y de cambios
- Procedimientos que aseguren ajustes a los estándares de IS
- Mecanismos de medición y generación de informes

Control/Garantía/Coste

Control de calidad: es una serie de inspecciones, revisiones y pruebas utilizados a lo largo del proceso del software para asegurar que cada producto cumple con los requisitos que le han sido asignados.

Garantía de calidad: es el establecimiento de un marco de procedimientos organizativos que llevan a conseguir una alta calidad del software

Ejemplo:
- El control de calidad nos lleva a hacer una RTF del diseño de una parte del sistema
- Hacemos RTFs porque forman parte de la garantía de calidad de nuestra organización

Coste de calidad: incluye todos los costes que se derivan de la búsqueda de la calidad o en las actividades relacionadas en la obtención de la calidad. Hay 3 tipos:
De prevención: incluyen
- Planificación de calidad
- RTFs
- Equipo de pruebas
- Formación

De evaluación: incluyen:
- Inspección en el proceso y entre procesos
- Calibrado y mantenimiento del equipo
- Pruebas

De fallos: donde hay internos y externos
- Internos: se producen cuando se detecta un error antes de la entrega al cliente. Los costes asociados incluyen:
    - Revisión
    - Reparación
    - Análisis de fallos

- Externos: se producen cuando se detecta un error tras la entrega al cliente. Los gastos asociados incluyen:
    - Coste de los internos
    - Resolución de quejas
    - Devolución y sustitución de productos
    - Soporte de línea de ayuda
    - Trabajo de garantía

Evidentemente, cuanto más tardemos en solucionar un fallo más costoso será su resolución.

El equipo SQA

La SQA comprende tareas llevadas a cabo por dos tipos de personal:

- Desarrolladores: llevan a cabo el trabajo técnico
- Equipo SQA: ayudan al equipo de desarrolladores para que el software tenga una gran calidad

Actividades del equipo SQA:

- Establecer el plan SQA del proyecto
- Participar en el desarrollo de la descripción del proceso del software
- Revisión de las actividades de ingeniería del software para verificar su ajuste al proceso de software
- Auditoría de los productos de software designados para verificar el ajuste con los definidos como parte del proceso de software
- Asegurar que las desviaciones del trabajo y los productos del software se documentan y se manejan de acuerdo a un procedimiento establecido
- Registrar lo que no se ajuste a los requisitos y avisar a sus superiores
- Coordinar el control y gestión de cambios
- Analizar las métricas del software

Verificación y Validación

V&V son los procesos que determinan si los productos desarrollados de una actividad dada se ajustan a los requisitos de esa actividad y si el software satisface las necesidades de usuario y su uso deseado. En otras palabras, son los procesos que garantizan la calidad del software.
Según Boehm:

- Verificación se encarga de comprobar si estamos construyendo el producto correctamente
- Validación se encarga de comprobar si estamos construyendo el producto correcto

Dentro de V&V hay dos aproximaciones complementarias para el análisis y comprobación de los sistemas: las inspecciones/revisiones de software (estáticas) y pruebas del software (dinámicas)
Algunos resultados de V&V son:

- Facilitar la detección y corrección temprana de anomalías
- Mejorar la percepción de la gestión de riesgos del proceso y el producto
- Proporcionar una evaluación temprana del rendimiento
- Soportar las actividades de mejora del proceso

Revisiones del software

Son un filtro para el proceso de IS y purifican las actividades estructurales.
Nos vamos a centrar en las Revisiones Técnicas Formales (RTFs) o inspecciones formales.
El objetivo básico de las RTFs es detectar errores antes de que se conviertan en defectos.
Una RTF es una actividad de garantía de calidad llevada a cabo por los desarrolladores.
Los objetivos de las RTFs son:

- Descubrir errores
- Verificar que el software alcanza sus requisitos
- Garantizar que se desarrolla según los estándares
- Hacer que los proyectos sean más manejables
- Conseguir un software desarrollado de manera uniforme

Las RTFs revisan Elementos de Configuración Software (ECS), que son los resultados de cada tarea de trabajo.
Estas RTFs se instrumentan mediante una reunión, previamente planificada.
Normas de la reunión:

- Deben convocarse entre 3 y 5 personas
- No puede durar más de 2 horas
- Se debe preparar previamente, sin que suponga un esfuerzo para la persona de más de 2 horas

El procedimiento de la reunión es el siguiente:

1. El responsable del producto (productor) informa al jefe del proyecto del fin de un trabajo
2. El jefe de proyecto contacta con el jefe de revisión, que distribuye el producto a dos o tres revisores
3. Revisores y el jefe de revisión revisan el producto 1 o 2 horas
4. El jefe revisor planifica una reunión
5. A la reunión asisten productor, jefe de revisión y revisores
6. Uno de los revisores es el registrador
7. El productor expone su producto

8. Revisores exponen sus pegas
9. Cuando se descubre un problema o error, el registrador toma nota
10. Al final de la reunión se puede decidir
    - Aceptar el producto sin modificaciones
    - Rechazar el producto por los errores
    - Aceptar el producto supuesto que se llevan a cabo algunas modificaciones (sin nueva RTF)
11. Una vez llevada a cabo la reunión los asistentes firman el registro de revisión para indicar asistencia y conformidad
12. Se genera una lista de de problemas de revisión (sirve para identificar las áreas problemáticas dentro de un producto) y un informe sumario de revisión, que contiene qué se revisó, quién, qué se descubrió y consecuencias
13. Cuando el producto es aceptado con cambios, el revisor jefe o el equipo SQA deben encargarse del seguimiento de los cambios identificados en la lista de problemas de revisión.

TEMA 10: GCS

La Gestión de la Configuración Software (GCS) es una actividad de protección que gestiona el cambio a lo largo del ciclo de vida del software, puesto que el cambio se puede producir en cualquier momento.
GCS es responsable de:

- Identificar y controlar el cambio
- Garantizar la correcta implementación del cambio
- Informar del cambio a todos los que lo necesiten

Los elementos que componen toda la información generada como parte del proceso de IS se denominan colectivamente configuración del software.
A medida que el proceso de IS progresa, aumenta rápidamente el número de ECSs; y aquí es donde entra en juego el cambio.
Fuentes del cambio:

- Fallos
- Nuevas condiciones comerciales que dictan cambios en los requisitos

Línea base

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 cambios.

- Antes de que una ECS se convierta en línea base, el cambio puede llevarse a cabo de una forma rápida e informal
- Cuando forma parte ya de la línea base, si se quiere hacer un cambio, hay que seguir un procedimiento formal para evaluarlo y verificarlo

ECSs que forman un conjunto de líneas base: Plan de Proyecto, SRS, Diseño, Código, Casos de prueba, Manual preliminar de usuario, Documentos de mantenimiento, Manual de Usuario, Estándares y procedimientos de IS.

Actividades de GCS:

La GCS es una actividad de protección que puede considerarse dentro de la SQA.
Actividades de GCS:

- Identificación de ECSs
- Control de versiones
- Control de cambios
- Gestión del impacto
- Auditoría de configuración
- Informes de estado

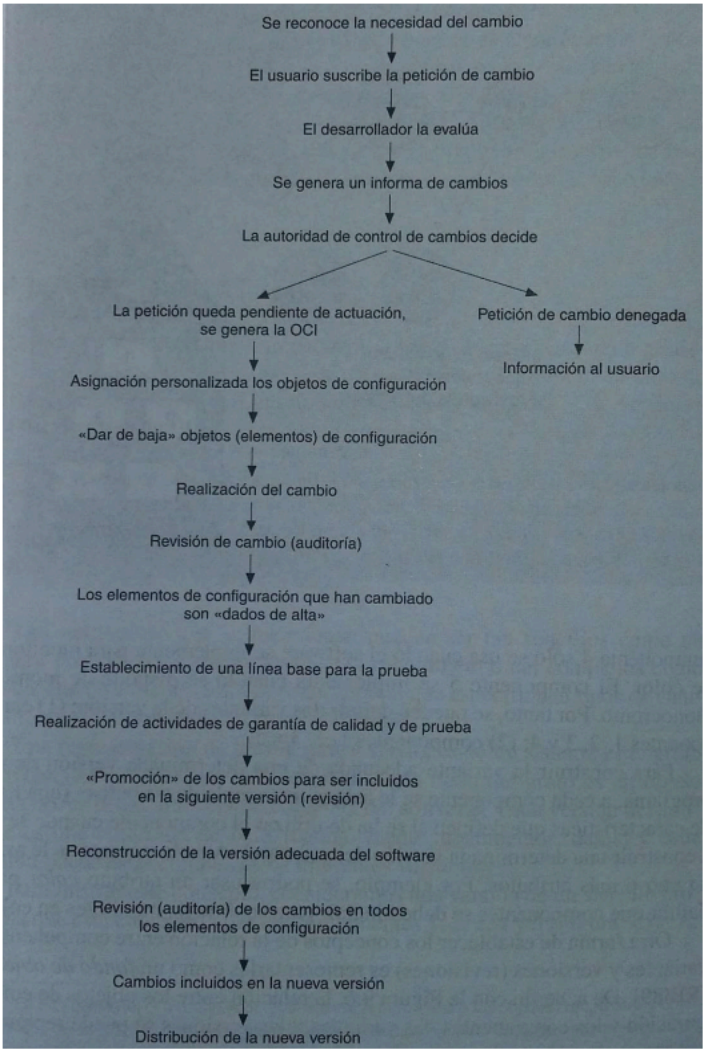
Identificación de ECSs: es fundamental para esto:

- Identificar qué ECSs van a formar parte de la línea base
- Distinguir entre un ECS compuesto (SRS) y básico (modelo del dominio dentro de la SRS)
- Tener una política de nombrado de los ECS
- Identificar las dependencias entre los ECS
- Identificar en los ECS los cambios llevados a cabo (historial de cambios)

Control de versiones: permite gestionar la versión del sistema.

- La versión del sistema viene dada por la versión de sus ECSs
- A su vez cada versión puede tener distintas variantes, que suelen darse cuando tenemos misma versión para diferentes plataformas

Control de cambios: Antes de que la ECS se convierta en línea base hay un control de cambios informal, pero cuando ya es línea base hay que emplear un mecanismo formal de control de cambios.



Gestión del impacto: tiene en cuenta el trabajo requerido para entender las interdependencias en una red de trabajo y controlar sus efectos sobre otros ECS y la gente responsable de ellos.
Incluye tres acciones:

- Una red de cambio indica las personas que deben efectuar o ser conscientes de los cambios llevados a cabo en los ECS
- Gestión del impacto del cambio, que evalúa el impacto del cambio en los miembros de la red de cambio e informa sobre el mismo
- Gestión del impacto hacia atrás. Examina los cambios hechos por otros miembros del equipo y su impacto en el trabajo propio

Auditorías software: la identificación, control de versiones y control de cambios promueven un seguimiento hasta la generación de la OCI. Podemos asegurar que el cambio se ha realizado correctamente gracias a las RTFs y las auditorías de configuración software.

- Las RTFs se preocupan de la corrección técnica del cambio

- La auditoría de configuración software tiene un carácter complementario y se encarga de si:
  - Se ha hecho el cambio especificado en la OCI
  - Se han incorporado modificaciones adicionales
  - Se ha llevado a cabo una RTF
  - Se han seguido adecuadamente los estándares de IS
  - Se han reflejado los cambios en el ECS, con fecha de cambio y autor
  - Se han seguido procedimientos de GCS para gestionar el cambio
  - Se han actualizado todos los ECS relacionados

Informes de estado: los Informes de Estado de Configuración Software (IECs) informan sobre qué pasó, quién lo hizo, cuándo y qué más se vio afectado. Se debe generar un IEC:

- Cada vez que se asigna una nueva identificación a un ECS
- Cada vez que la ACC (Autoridad Competente de Cambios) expide una OCI (Orden de Cambio de Ingeniería)
- Cada vez que se lleva a cabo una Auditoría de Configuración
- Regularmente, para mantener informados a los desarrolladores de los cambios importantes

Sistemas de Control de Versiones

Un Sistema de Control de Versiones (SCV) es un programa que ayuda a crear y mantener las diversas versiones de un conjunto de ficheros.
Hay 3 arquitecturas:

- Local: El contribuyente tiene un repositorio local responsable de almacenar las diferentes versiones del proyecto
- Centralizada (como SVN):
  - Servidor que administra el repositorio remoto responsable de almacenar las diferentes versiones del proyecto
  - Cliente instalado en los diferentes contribuyentes que acceden y modifican estas versiones del proyecto almacenadas en el repositorio
- Distribuida (como Git)
  - Servidor que administra el repositorio remoto responsable de almacenar las diferentes versiones del proyecto
  - Cliente instalado en los diferentes contribuyentes que replica el repositorio remoto local, permitiendo actualizaciones a/desde el repositorio local/remoto

El resto del tema es Github y tal.

David Castro García.