1.- Gestión de proyectos

1.1.- Introducción:

La gestión de proyectos implica planificación, supervisión, y control del personal, del proceso y de los eventos que ocurren mientras evoluciona el software desde la fase preliminar a la implementación operacional.

La gestión eficaz de un proyecto de software se centra en las cuatro P's: *personal, producto, proceso y proyecto.* El orden no es arbitrario.

Personal:

La necesidad de contar con personal para el desarrollo del software altamente preparado y motivado.

Los participantes: El proceso del software (y todos los proyectos de software) lo componen participantes que pueden clasificarse en una de estas cinco categorías:

1. *Gestores superiores,* que definen los aspectos de negocios que a menudo tienen una significativa influencia en el proyecto.
2. *Gestores (técnicos) del proyecto,* que deben planificar, motivar, organizar y controlar a los profesionales que realizan el trabajo de software.
3. *Profesionales,* que proporcionan las capacidades técnicas necesarias para la ingeniería de un producto o aplicación.
4. *Clientes,* que especifican los requisitos para la ingeniería del software y otros elementos que tienen menor influencia en el resultado.
5. *Usuarios finales,* que interaccionan con el software una vez que se ha entregado para la producción.

Para ser eficaz, el equipo del proyecto debe organizarse de manera que maximice las habilidades y capacidades de cada persona.

Producto:

Antes de poder planificar un proyecto, se deberían establecer los objetivos y el ámbito del producto, se deberían considerar soluciones alternativas e identificar las dificultades técnicas y de gestión. Sin esta información, es imposible definir unas estimaciones razonables (y exactas) del coste; una valoración efectiva del riesgo, una subdivisión realista de las tareas del proyecto o una planificación del proyecto asequible que proporcione una indicación fiable del progreso.

Un análisis detallado de los requisitos del software proporcionaría la información necesaria para las estimaciones, pero el análisis a menudo lleva semanas o meses. Aún peor, los requisitos pueden ser fluidos, cambiando regularmente a medida que progresa el proyecto. Y, aún así, se necesita un plan ¡ya!.

Por tanto, debemos examinar el producto y el problema a resolver justo al inicio del proyecto. Por lo menos se debe establecer el ámbito del producto y delimitarlo.

Proceso:
Un proceso de software (unidad anterior) proporciona la estructura desde la que se puede establecer un detallado plan para el desarrollo del software.
El gestor del proyecto debe decidir qué modelo de proceso es el más adecuado para los clientes que han solicitado el producto y la gente que realizará el trabajo; las características del producto en sí, y el entorno del proyecto en el que trabaja el equipo de software. Cuando se selecciona un modelo de proceso, el equipo define entonces un plan de proyecto preliminar basado en un conjunto de actividades estructurales. Una vez establecido el plan preliminar, empieza la descomposición del proceso. Es decir, se debe crear un plan completo reflejando las tareas requeridas a las personas para cubrir las actividades estructurales.

Proyecto:
Para evitar el fracaso del proyecto, un gestor de proyectos de software y los ingenieros de software que construyeron el producto deben eludir un conjunto de señales de peligro comunes; comprender los factores del éxito críticos que conducen a la gestión correcta del proyecto y desarrollar un enfoque de sentido común para planificar, supervisar y controlar el proyecto.

2.1.- Métricas:

El gestor recopila medidas y desarrolla métricas para obtener indicadores. Un *indicador* es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso del software, del proyecto de software o del producto en sí.
Una mala gestión de proyecto se manifiesta principalmente en: incumplimiento de plazos, incremento de los costos y entrega de productos de mala calidad lo cual conlleva a un perjuicio económico.

Métricas del proceso:

Los *indicadores de proceso* permiten a una organización de ingeniería del software tener una visión profunda de la eficacia de un proceso ya existente (por ejemplo: el paradigma, las tareas de ingeniería del software, productos de trabajo e hitos). También permiten que los gestores evalúen lo que funciona y lo que no. Las métricas del proceso se recopilan de todos los proyectos y durante un largo

período de tiempo. Su intento es proporcionar indicadores que lleven a mejoras de los procesos de software a largo plazo.

Métricas del proyecto:
Los *indicadores de proyecto* permiten al gestor de proyectos del software: evaluar el estado del proyecto en curso; seguir la pista de los riesgos potenciales; detectar las áreas de problemas antes de que se conviertan en críticas; ajustar el flujo y las tareas del trabajo, y evaluar la habilidad del equipo del proyecto en controlar la calidad de los productos de trabajo del software.

2.2 Mediciones del software:

Las métricas del software se pueden categorizar de maneras:
Medidas directas: Se obtienen del proceso o producto siendo observado y no dependen de otras medidas. Entre las medidas directas del proceso se encuentran el costo y el esfuerzo, del producto las líneas de código (LDC), el tiempo de ejecución, el tamaño de memoria requerido, los defectos observados en un período de tiempo.
Medidas indirectas: Dependen de medidas más elementales. Entre ellas se encuentran funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento y muchas otras capacidades.

Métricas Orientadas al Tamaño:
Las métricas del software orientadas al tamaño provienen de la normalización de las medidas de calidad y/o productividad considerando el «tamaño» del software que se haya producido. Si una organización de software mantiene registros sencillos, se puede crear una tabla de datos orientados al tamaño. La tabla lista cada proyecto de desarrollo de software de los últimos años y las medidas correspondientes de cada proyecto.

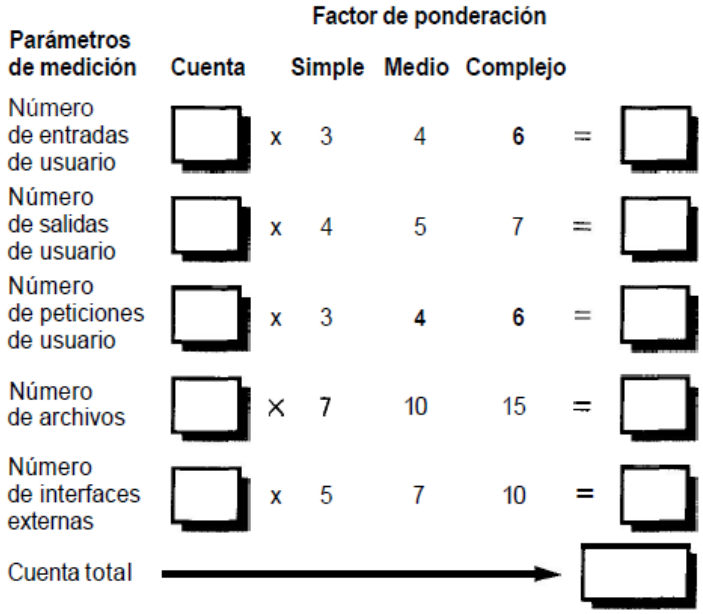
Proyecto	LDC	Esfuerzo	Coste £(000)	Pag. Doc.	Errores	Defectos	Personas
Alfa	12,100	24	168	365	134	29	3
Beta	27,200	62	440	1224	321	86	5
Gamma	20,200	43	314	1050	256	64	6
▪		▪	▪	▪			
			▪		▪		

El proyecto *alfa:* se desarrollaron 12.100 líneas de código (LDC) con 24 personas-mes y con un coste de E168.000. Debe tenerse en cuenta que el esfuerzo y el coste registrados en la tabla incluyen todas las actividades de ingeniería del software. Otra información sobre el proyecto alfa indica que se desarrollaron 365 páginas de documentación, se registraron 134 errores antes de que el software se entregara y se encontraron 29 errores después de entregárselo al cliente dentro del primer año de utilización. También sabemos que trabajaron tres personas en el desarrollo del proyecto alfa.

Métricas Orientadas a la Función

Las métricas del software orientadas a la función utilizan una medida de la funcionalidad entregada por la aplicación como un valor de normalización. Ya que la funcionalidad no se puede medir directamente, se debe derivar indirectamente mediante otras medidas directas. Los puntos de función se derivan con una relación empírica según las medidas contables (directas) del dominio de información del software y las evaluaciones de la complejidad del software

Los puntos de función se calculan completando una tabla. Se determinan cinco características de dominios de información y se proporcionan las cuentas en la posición apropiada de la tabla.



Número de entradas de usuario: Se cuenta cada entrada de usuario que proporciona al software diferentes datos orientados a la aplicación. Las entradas deben ser distinguidas de las peticiones, que se contabilizan por separado.
Número de salidas de usuario: Se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto la "salida" se refiere a informes, pantallas, manejos de error, etc.
Número de peticiones de usuario: Una petición está definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva. Se cuenta cada petición por separado.
Números de archivo: Se cuenta cada archivo maestro lógico (o sea, una agrupación lógica de datos que puede ser una gran base de datos o un archivo independiente).
Número de interfaces externas: Se cuentan todas las interfaces legibles por la máquina (por ejemplo archivos de datos en cinta o disco) que son utilizados para transferir información a otro sistema.

Métricas para la calidad del software

Las métricas que se usan tras la distribución se centran en el número de defectos no descubiertos en la prueba y en la facilidad de mantenimiento del sistema. Estas métricas incluyen:

Corrección*:* La corrección es el grado con que el software realiza la función requerida. La medida más común de corrección es *defectos por KLDC (miles de líneas de código)*.

Facilidad de mantenimiento: Es la facilidad con la cual se puede corregir un programa si se encuentra un error, adaptarlo si su entorno cambia, o mejorarlo si el cliente desea un cambio en los requisitos. Debemos utilizar medidas indirectas por ej. el tiempo medio entre cambios (TMEC).

Integridad*:* Este atributo mide la habilidad de un sistema para resistir ataques (tanto accidentales como intencionados) contra su seguridad. El ataque se puede realizar en cualquiera de los tres componentes del software: programas, datos y documentos. Para medir la integridad, se tiene que definir dos atributos adicionales: amenaza y seguridad. *Amenaza* es la probabilidad de que un ataque de un tipo determinado ocurra en un tiempo determinado. La *seguridad* es la probabilidad de que se pueda repeler el ataque de un determinado tipo.

Facilidad de uso: Es un intento de cuantificar la amistad con el usuario.

3.- Planificación del proyecto:

3.1.- Introducción:

La gestión de un proyecto de software comienza con un conjunto de actividades que globalmente se denominan *planificación del proyecto.* El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor del proyecto hacer estimaciones razonables de recursos, coste y planificación temporal. Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo de un proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto.

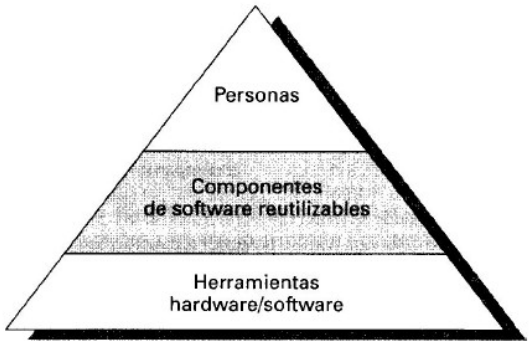
3.2.- Ámbito del software:

La primera actividad de la planificación del proyecto de software es determinar el ámbito del software el cual describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad.
Obtención del ámbito del software: La técnica utilizada con más frecuencia para acercar al cliente y al desarrollador, y para hacer que comience el proceso de comunicación es establecer una reunión o una entrevista preliminar.

Un equipo compuesto de clientes, usuarios, analistas y de desarrolladores, trabajan juntos para identificar el problema, proponer elementos de solución, negociar diferentes enfoques y especificar un conjunto preliminar de requisitos.

3.3.- Recursos:

La segunda tarea de la planificación del desarrollo de software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de software.
Los recursos se de desarrollo se ilustran en forma de pirámide como muestra la figura.



Recursos humanos:
Hay que especificar tanto la posición dentro de la organización (por ejemplo: gestor, ingeniero de software experimentado, etc.) como la especialidad (por ejemplo: telecomunicaciones, bases de datos, cliente/servidor).
El número de personas requerido para un proyecto de software sólo puede ser determinado después de hacer una estimación del esfuerzo de desarrollo (por ejemplo, personas-mes).

Recursos de software reutilizables:
La ingeniería del software basada en componentes destaca la reutilización, esto es, la creación y la reutilización de componentes estandarizadas para una fácil aplicación y validarse para una fácil integración.

Recursos de entorno:
El entorno es donde se apoya el proyecto de software, llamado a menudo *entorno de ingeniería del software (EIS),* incorpora hardware y software.
El ***hardware*** proporciona una plataforma con las herramientas (**software**) requeridas para producir los productos que son el

resultado de una buena práctica de la ingeniería del software. Un planificador de proyecto debe determinar la ventana temporal requerida para el hardware y el software, y verificar que estos recursos estarán disponibles.

Cada recurso requiere:
- Descripción
- Informe de disponibilidad
- Fecha en que se lo requiere
- Tiempo que se lo necesita

3.4.- Estimación del proyecto de software:

La estimación del coste y del esfuerzo del software nunca será una ciencia exacta. Son demasiadas las variables (humanas, técnicas, de entorno, políticas) que pueden afectar al coste final del software y al esfuerzo aplicado para desarrollarlo. Sin embargo, la estimación del proyecto de software puede dejar de ser un oscuro arte para convertirse en una serie de pasos sistemáticos que proporcionen estimaciones con un grado de riesgo aceptable.

Para realizar estimaciones seguras de costes y esfuerzos tenemos varias opciones posibles:
1. Dejar la estimación para más adelante.
2. Basar las estimaciones en proyectos similares ya terminados
3. Utilizar técnicas de descomposición relativamente sencillas para generar las estimaciones de coste y de esfuerzo del proyecto.
4. Utilizar uno o más modelos empíricos para la estimación del coste y esfuerzo del software.

Técnica de descomposición
Descomponemos el problema, volviéndolo a definir como un conjunto de pequeños problemas y calcula el costo de cada uno (estimándolo mediante cantidad de líneas de código LCD o punto función PF) y luego calcula el costo general.

Modelos empíricos de estimación:
Un modelo de estimación para el software de computaras *utiliza fórmulas derivadas empíricamente* para predecir los datos que se requieren en el paso de planificación del proyecto de software. Los datos empíricos que soporta la mayoría de los modelos se obtienen de una muestra de proyectos limitada. Por esta razón, un mismo modelo de estimación no es adecuado para todas las clase de software ni para todos los entornos de desarrollo.

Herramientas automáticas para la estimación:
Las técnicas de descomposición y los modelos empíricos de estimación descritos en las secciones anteriores se pueden implementar con software. Las herramientas automáticas de estimación permiten al planificador estimar costes y esfuerzos.

4.- Análisis y gestión de riesgos

4.1.- Introducción:

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

4.2.- Riesgo del software:

El riesgo siempre implica dos características:

- *Incertidumbre:* el acontecimiento que caracteriza al riesgo puede o no puede ocurrir; por ejemplo, no hay riesgos de un 100 por 100 de probabilidad'.
- *Pérdida:* si el riesgo se convierte en una realidad, ocurrirán consecuencias no deseadas o pérdidas.

Cuando se analizan los riesgos es importante cuantificar el nivel de incertidumbre y el grado de pérdidas asociado con cada riesgo. Para hacerlo, se consideran diferentes categorías de riesgos.

- Riesgos del proyecto: amenazan el plan del proyecto. Identifican los problemas potenciales de presupuesto, planificación temporal, personal, recursos, cliente y requisitos.
- Riesgos del producto: afectan la calidad o rendimiento del software.
- Riesgos del negocio: amenazan la viabilidad del software a construir. Los riesgos del negocio a menudo ponen en peligro el proyecto o el producto. Ej: construir un producto excelente que no quiere nadie en realidad.

4.3.- Identificación del riesgo

La *identificación del riesgo* es un intento sistemático para especificar las amenazas al plan del proyecto (estimaciones, planificación temporal, carga de recursos, etc.).
Existen dos tipos diferenciados de riesgos para cada categoría: riesgos genéricos y riesgos específicos del producto.
Riesgos genéricos: son una amenaza potencial para todos los proyectos de software.
Riesgos específicos: *Los riesgos específicos de producto* sólo los pueden identificar los que tienen una clara visión de la tecnología, el personal y el entorno específico del proyecto en cuestión. Para

identificar los riesgos específicos del producto, se examinan el plan del proyecto y la declaración del ámbito del software.

4.4.- Proyección del riesgo

También denominada *estimación del riesgo,* intenta medir cada riesgo de dos maneras la probabilidad de que el riesgo sea real y las consecuencias de los problemas asociados con el riesgo, si ocurriera.
Pasos a seguir:

- Se considera por separado cada riesgo identificado y se decide la probabilidad y el impacto
- Se construye la tabla de riesgos
- Establecer una escala que refleje la probabilidad observada de un riesgo
    - Bastante improbable < 10%
    - Improbable 10-25%
    - Moderado 25-50%
    - Probable 50-75%
    - Bastante probable > 75%
- Estimar el impacto en el proyecto:
    - 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)

Riesgos	**Categoría**	**Probabilidad**	**Impacto**
cliente cambiará los requisitos	Proyecto	80 %	2
Falta de formación en las herramientas	Proyecto	80%	3
Menos reutilización de la prevista	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
Se perderán los presupuestos	Negocio	40%	1
Los usuarios finales se resisten al sistema	Negocio	40%	3
La tecnología no	Producto	30%	1

alcanzará las expectativas			
Personal sin experiencia	Proyecto	30%	2
Mayor número de usuarios de los previstos	Negocio	30%	3

5.- Planificación temporal del proyecto

5.1.- Introducción:

La realidad de un proyecto es que hay que realizar cientos de pequeñas tareas antes de poder alcanzar el objetivo final. Algunas de estas tareas quedan fuera del camino principal y pueden contemplarse sin preocuparse del impacto de la fecha de terminación del proyecto. Otras se encuentran en el camino crítico y si estas tareas se retrasan, la fecha de terminación del proyecto entero se pone en peligro.
El objetivo del gestor del proyecto es definir todas las tareas del proyecto, construir una red que describa su interdependencia, identificar las tareas que son críticas dentro de la red y después hacer un seguimiento para asegurarse de que el retraso se reconoce.
La *planificación temporal de un proyecto* es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto asignando el esfuerzo a las tareas específicas.

La planificación temporal del proyecto de software se guía por unos principios básicos:

- *Compartimentación*: el proyecto debe dividirse en un número de actividades y tareas manejables.
- *Interdependencia*: debe determinarse las interdependencias de cada actividad o tarea compartimentada. Algunas tareas pueden ocurrir en una secuencia determinadas y otras pueden darse en paralelo.
- *Asignación de tiempo*: asignar un numero de unidades de tiempo (ej: personas-día de esfuerzo) a cada tarea.
- *Validación de esfuerzo*: Valida que el numero de personas asignadas a las tareas no sea mayor al disponible en un momento dado.
- *Responsabilidades definidas*: cada tarea que se programe debe asignarse a un miembro del equipo específico.
- *Resultados definidos*: cada tarea programada debería tener un resultado definido (Ej.: el diseño de un módulo)

- *Hitos definidos*: Todas las tareas o grupos de tareas deberían asociarse con un hito del proyecto. Se considera un hito cuando se ha revisado la calidad de uno o más productos.

Cada uno de los principios anteriores se aplica a medida que evoluciona la planificación temporal del proyecto.

5.2.- Relación persona – esfuerzo:

Cuando el proyecto es pequeño una sola persona puede analizar los requisitos, realizar el diseño, generar el código y llevar a cabo las pruebas. A medida que el tamaño del proyecto aumenta debe involucrarse a más gente.

Si existe un retraso en el tiempo podemos suponer que se solucionará añadiendo programadores al proyecto, pero esto a menudo causa un efecto destructivo en el mismo, haciendo incluso que los tiempos se alarguen aún más. Esto se debe a que la gente que se añada al proyecto tendrá que comprender el sistema y la gente que les enseñe será la misma que esta realizando el trabajo.

5.3.- Definición de tareas y paralelismo:

Cuando en un proyecto están involucradas más de una persona, es posible que las actividades de desarrollo se puedan realizar en paralelo.

El planificador debe determinar las dependencias entre tareas, para asegurar el progreso continuo hasta la terminación. Además el director del proyecto debe identificar las tareas que están en el *camino crítico*, es decir, las tareas que tienen que terminarse a tiempo para que el proyecto se termine a tiempo.

5.4.- Herramientas para la planificación temporal:

Para la planificación temporal pueden aplicarse distintos métodos:

- Técnica de evaluación y revisión del programa PERT
- Método del camino critico CPM
- Diagrama de Gantt

5.5.- Seguimiento de la planificación temporal:

La planificación temporal del proyecto le proporciona al gestor un mapa de carretera.

El seguimiento se puede llevar a cabo de las siguientes formas:

- Realizando reuniones periódicas sobre el estado del proyecto.
- Evaluado los resultados de las revisiones realizadas.
- Determinando si los hitos se han alcanzado en la fecha programada.
- Comparando la fecha de comienzo real con la planeada para cada tarea.

El *control* se utiliza para hacer frente a los problemas y para dirigir al personal. Si las cosas van bien, no se requiere mucho control. Si aparecen problemas, se debe ejercer un control para reducirlos lo más rápido posible (recursos adicionales, reorganización de personal o redefinición de los tiempos).