GESTIÓN, IMPLANTACIÓN Y MANTENIMIENTO DE PROYECTOS SOFTWARE

10.1.- Introducción

Todo proyecto es un conjunto de etapas, actividades y tareas para alcanzar un objetivo que implica un trabajo no inmediato a un plazo relativamente largo. Precisamente es la división en trabajos mas sencillos lo que permite al personal del proyecto dominar la complejidad del software que se debe desarrollar.

Todo proyecto se plantea con un objetivo de obtención de beneficios que no necesariamente deben ser económicos, aunque todos ellos (servicio, imagen, etc.) tengan repercusiones en este aspecto. Por ello, la gestión de los proyectos software se aborda a través de distintos aspectos: **planificación, dirección, organización** y **control**. Estas áreas determinan la toma de decisiones sobre la información relativa a los proyectos software.

10.2.- Planificación

Dentro del área de planificación deben cubrirse dos aspectos importantes: la realización de un plan de proyecto por parte del director o jefe de proyecto y la gestión de compromisos.

El **Plan de proyecto** describe los trabajos que se van a realizar y la forma en la que el director del proyecto va a dirigir su desarrollo. Define un conjunto de tareas coordinadas en el tiempo, así como los recursos necesarios para cumplir los requisitos contractuales o los compromisos internos de la compañía.

El plan de proyecto debe:

- Proporcionar un resumen del proyecto a los altos directivos
- Permitir tanto al director del proyecto como a los clientes supervisar el progreso del proyecto desde el inicio hasta el final
- Ser un documento orientado al cliente
- Ser un documento base con la aprobación del cliente y actualizable a medida que avanza el proyecto

El contenido del plan de proyecto varía con cada proyecto, pero hay una serie de elementos esenciales que deberá incluir:

- Un resumen del proyecto que pueda ser comprendido por cualquier persona, indicando los productos **entregables** de forma que, cuando se produzcan, se pueda comprobar que se ajustan al plan.
- Una lista de **hitos** alcanzables
- Los procedimientos y estándares que se van a alcanzar
- Una especificación del proceso de revisión que determine quién, cómo y cuándo se va a revisar el proyecto y con qué objeto.
- Un plan que defina la comunicación entre la organización de desarrollo y el cliente.
- Un **Diagrama de descomposición del trabajo**
- Una lista del personal del proyecto y su asignación en el Diagrama de descomposición del trabajo

- Una red de actividades que muestre la secuencia de actividades en el tiempo y su relación con ellas.
- Presupuestos y calendarios para todas las actividades que tienen un responsable.
- Planes de subcontratación si algunas partes del proyecto van a ser desarrolladas por personal ajeno a la empresa
- Aseguramiento de la calidad
- Gestión de configuración
- Integración hardware/software.
- Planes de seguridad.

Página de título
Hoja de revisión
Prefacio
Tabla de Contenidos
Lista de figuras
Lista de Tablas

I. Introducción
A. Visión general del proyecto
B. Productos finales
C. Evolución del Plan de Proyecto
D. Documentos de referencia
E. Definiciones y acrónimos

II. Organización del Proyecto
A. Modelo de procesos
B. Estructura organizativa
C. Fronteras e interfaces organizativas
D. Responsabilidades

III. Procesos de Gestión
A. Objetivos y prioridades de gestión
B. Suposiciones, dependencias y restricciones
C. Gestión de riesgos
D. Mecanismos de supervisión y control
E. Plan de personal

IV. Proceso Técnico
A. Metodología, técnicas y herramientas
B. Documentación software
C. Funciones de apoyo al proyecto

V. Plan de Desarrollo
A. Paquetes de trabajo
B. Dependencias
C. Recursos
D. Presupuesto y distribución de recursos
E. Calendario

Componentes adicionales
Índice
Apéndices

Para confeccionar el plan de proyecto, su organización puede basarse en alguno de los estándares existentes, por ejemplo el estándar IEEE, 1988 que es el que se presenta en la figura adjunta, si bien cada organización puede ajustar esta estructura a sus proyectos particulares.

10.2.1.- Calendario del Proyecto

Uno de los apartados indicado en la tabla anterior es la determinación del Plan de Desarrollo. El principal objetivo de esta sección es establecer la configuración del **calendario del proyecto** o **programa de tiempos.**

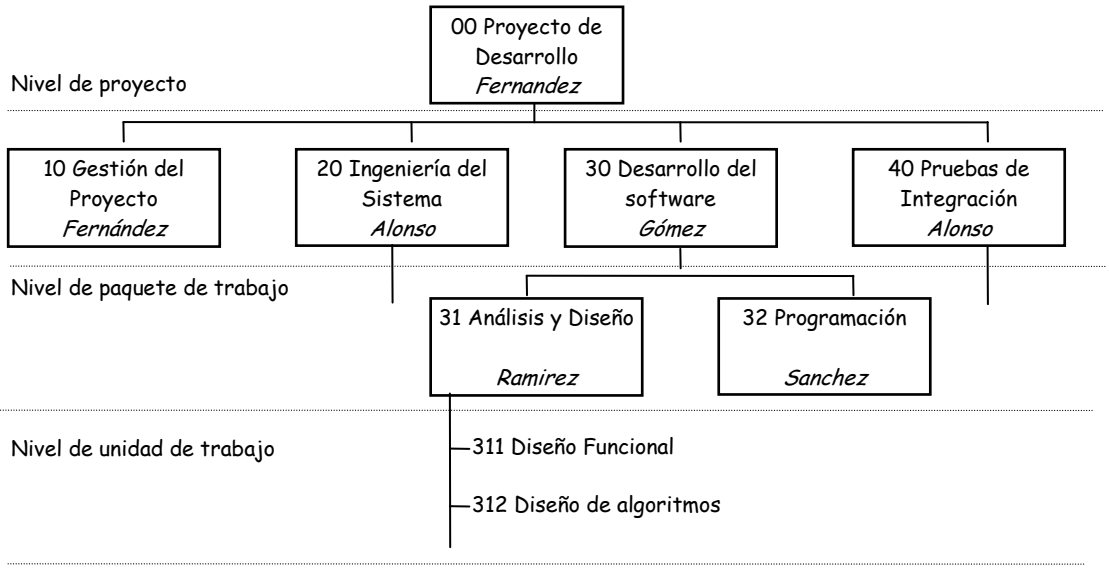
El calendario es *una representación gráfica de todas las actividades del proyecto necesarias para producir el resultado final* que permite al director del proyecto coordinar de una forma efectiva al equipo de desarrollo durante el transcurso del mismo. El calendario es dinámico, es decir, puede variar a medida que avanza el proyecto por cambios no previstos en su extensión, sus plazos, etc.

Para que un programa de tiempos sea efectivo debe tener las siguientes características:

- Comprensible para todas aquellas personas que van a utilizarlo
- Suficientemente detallado para servir de base para medir y controlar el progreso del proyecto
- Capaz de poder señalar las actividades críticas, es decir, aquellas actividades que influyen en la duración final del proyecto
- Flexible, fácilmente modificable
- Basado en estimaciones de tiempo fidedignas
- Ajustable a los recursos disponibles
- Compatible con los planes de otros proyectos que compartan los mismos recursos

Para desarrollar un calendario es necesario considerar los pasos siguientes en el orden en que se exponen:

1. Definición de los objetivos del proyecto especificándolos en términos cuantificables
2. Descomposición de actividades mediante la producción de un **diagrama de descomposición de trabajo** o técnica que permite representar las actividades que hay que realizar a distinto nivel de detalle por medio de un **diagrama de estructuras** indicando, para cada tarea, las personas responsables de su finalización:



3. Relación entre las actividades
4. Estimación de tiempos y costes de las actividades
5. Reajuste del programa de tiempos a las restricciones del proyecto
6. Asignación de los recursos/Definición de la organización del equipo

7. Revisión del calendario

En cuanto a las **técnicas para la realización del calendario** suelen utilizarse las siguientes:

10.2.1.1.- Diagrama de Hitos

Es el método mas simple para determinar el calendario. Consiste en un cuadro o tabla formado por dos columnas; en la primera se señalan las actividades y el la segunda se indican sus fechas de realización.

Su principal ventaja está en la facilidad de uso y el mínimo coste de preparación. Las desventajas se centran en la incertidumbre existente sobre las fechas de comienzo de las actividades y la imposibilidad de reflejar las interrelaciones entre ellas.

10.2.1.2.- Diagrama de Grant

Se utiliza en procesos pequeños (de menor de 25 actividades) y supera algunos de los inconvenientes de los diagramas de hitos. Es, seguramente el tipo de calendario mas utilizado porque es mas comprensible que las **redes de precedencia** y aunque no es posible representar las dependencias entre las actividades, es mas fácil representar sus posibles solapamientos que en una red.

Normalmente se utiliza para estimar los recursos y el presupuesto en función del tiempo, identificando el total de recursos (y presupuesto) necesarios por unidad de tiempo para cada actividad y calculando el total para todas las actividades que ocurran durante un periodo de tiempo específico.

El diagrama de Grant es un diagrama de barras en forma de tabla donde se hace una referencia cruzada entre las tareas (filas) y los tiempos de duración de las mismas (columnas): **Unidades de Tiempo**

Tarea	1	2	3	4	5	6	7	8	9	10
1.1										
1.2										
1.3										
2.1										
2.2										

10.2.1.3.- Programa de tiempos FULL WALL

Es una técnica que se ha utilizado con bastante éxito en los últimos años. El calendario se determina en una reunión en la que intervienen todas las personas que van a trabajar en el proyecto. En la sala de reunión se forma un cuadro en una pared en el que las líneas verticales representan semanas de trabajo y las horizontales, las actividades que debe hacer el equipo del proyecto.

Previamente, debe haberse desarrollado un diagrama de hitos y una lista de actividades donde se indica quienes son los responsables de cada una.

Cada actividad se escribe en dos tarjetas. La primera tarjeta se etiqueta como **inicio** y la segunda como **final**. A cada miembro del equipo se le dan las tarjetas apropiadas. Cada persona clava las tarjetas en la pared en la semana de su elección.

A medida que avanza el calendario, las actividades se pueden dividir en tareas.

La disposición de las tarjetas se modificará reiteradamente hasta que se hayan tratado todas las interrelaciones y restricciones de las actividades y el calendario sea asequible.

Posteriormente se obtiene una copia del calendario.

El método es apropiado cuando no son muy grandes ni el número de actividades (entre 50 y 100) ni el número de integrantes en el proyecto (entre 3 y 10)

La mayor ventaja del método es el alto grado de interacción y compromiso que se alcanza entre los participantes durante la reunión. En contrapartida, esta técnica requiere reunir a todas las parte, cosa muy difícil cuando el proyecto es grande y hay muchas personas involucradas. Así mismo, el método no representa con claridad las interrelaciones entre las tareas o el camino crítico.

10.2.1.4.- Redes de precedencia

Las redes de precedencia, (**PERT y CPM**) son técnicas basadas en grafos para la planificación de proyectos. Aunque son técnicas muy parecidas, proceden de estudios muy diferentes y se caracterizan por ampliar la relación existente entre el coste y la duración de las actividades. De esta forma surge la planificación de proyectos con un coste mínimo.

Es conveniente de utilizar estas técnicas cuando un proyecto:

- Tiene todas sus actividades bien definidas
- Las actividades se pueden comenzar, interrumpir y realizar de forma separada, dentro de una secuencia dada
- Las actividades se pueden relacionar cono otras
- Las actividades están ordenadas de forma que se pueda conseguir una secuencia
- Una vez comenzada una actividad, debe continuar sin interrupción hasta su finalización.

La red es un modelo gráfico que señala las relaciones secuenciales entre los sucesos claves de un proyecto. PERT y CPM pueden mostrar un camino crítico, que *es la secuencia mas larga de actividades conectadas a través de la red y que determina la duración total del proyecto*. Para disminuir el tiempo total se reducen los tiempos de las actividades que están dentro del camino crítico, teniendo en cuenta que esta disminución suele conllevar un aumento del coste de la actividad .

La técnica permite visualizar las tareas que no son críticas, permitiendo adaptarlas durante el proyecto.

Las principales diferencias entre PERT y CPM son:

- CPM se enfoca hacia las actividades, mientras que PERT se enfoca hacia los eventos o sucesos. Esto da ventaja al PERT ya que permite identificar los eventos como hitos del proyecto, facilitando el control de la gestión.
- PERT permite el tratamiento de la probabilidad para la estimación del tiempo mientras que CPM no. Normalmente PERT parte del estudio de un calendario para proyectos con un alto grado de incertidumbre, mientras que CPM se utiliza normalmente para

procesos industriales que incluyen actividades bien definidas con bajo grado de incertidumbre

Existen varias reglas que hay que considerar en el desarrollo de una red PERT o CPM:

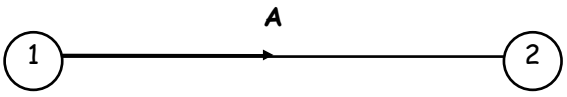
- *La red debería tener un mínimo de veinte eventos. Para proyectos pequeños es mas apropiado el diagrama de Grant*
- *Las redes que se realicen manualmente deben estar limitadas a un máximo de 300 sucesos. Si se estudian procesos de mas actividades es necesaria la utilización de una herramienta que gestione la técnica de forma automatizada.*
- *Los proyectos que justifican el uso de un gran número de actividades o eventos son:*

- Muy críticos
- De alto riesgo o incertidumbre
- Que involucran a muchas personas u organizaciones
- Técnicamente complejos
- Con actividad en diversas localidades geográficas.

10.2.1.4.1.- Técnica PERT

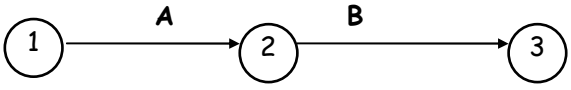
Esta técnica parte de la descomposición de un proyecto en actividades. Para su realización se consumen unos recursos determinados (máquina, mano de obra, etc.). Además, las actividades ocurren entre dos sucesos (*suceso inicial y suceso final*) entendiendo por **suceso** *un acontecimiento o punto temporal (una fecha) que no consume recursos*.

Se representa mediante un gráfico donde las actividades se reflejan en arcos y los sucesos mediante vértices:

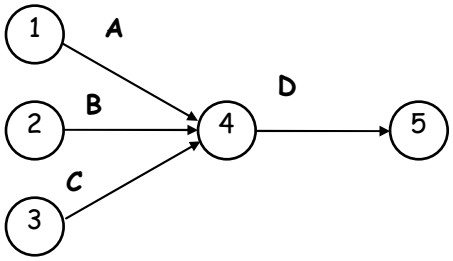


Las relaciones de precedencia pueden ser:

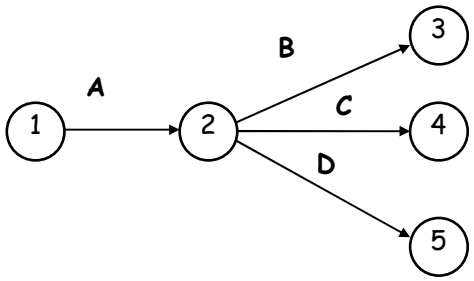
Lineales: Para iniciar la actividad B es preciso haber finalizado la actividad A. El suceso 2 es el suceso final de A y el suceso inicio de B:



Convergentes: Para iniciar la actividad D es necesario haber finalizado las actividades A, B y C:



Divergentes: Para poder iniciar cualquiera de las actividades B, C o D es necesario que haya finalizado la actividad A:

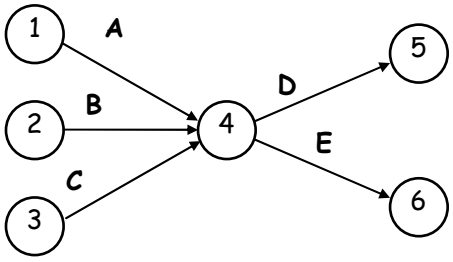


El siguiente paso es la determinación de las relaciones entre actividades, estudiando para cada actividad sus relaciones de precedencia, es decir, las actividades que han de estar finalizadas justamente antes del comienzo de la actividad dada

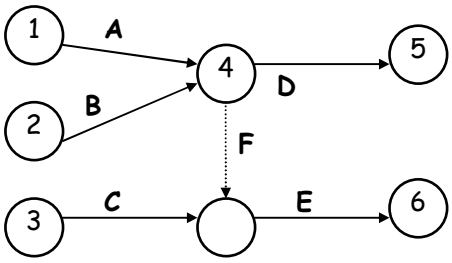
Pueden producirse conflictos entre actividades. Un ejemplo sería:

- Las actividades A y B preceden a la actividad D
- Las actividades A, B, C preceden a la actividad E

Sería el caso de la figura:



que se resuelve añadiendo una actividad ficticia F de duración 0:



Mediante un ejemplo se indica la representación de actividades del grafo PERT

Se supone que se tiene que realizar un proyecto que tiene las siguientes actividades: A, B, C, D, E, F, G y H. Las relaciones entre las actividades son:

- A precede a B, C y D
- B precede a E
- C precede a F
- D precede a G
- E, F preceden a H

Hay dos formas de recoger este conjunto de relaciones: la **matriz de encadenamientos** y el **cuadro de relaciones de precedencia.**

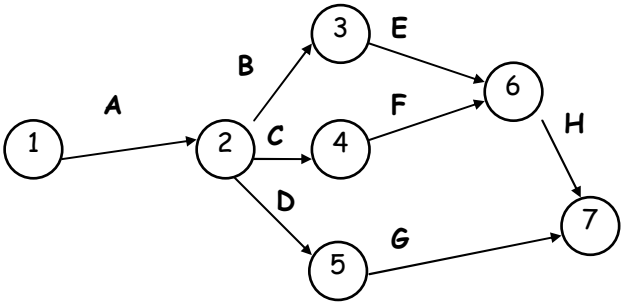
La **Matriz de encadenamientos** es una matriz cuadrada cuya dimensión coincide con el número de actividades en las que se descompone el proyecto. Sea M_{ij} un elemento de la matriz, si M_{ij} = X, entonces, para poder iniciar la actividad y es necesario que haya finalizado la actividad j .En el ejemplo anterior, la matriz quedará:

	A	B	C	D	E	F	G	H
A								
B	X							
C	X							
D	X							
E		X						
F			X					
G				X				
H					X	X		

El **cuadro relaciones de precedencia** es una tabla de dos columnas tal que en la primera se representan todas las actividades y en la segunda las actividades precedentes que deben estar completadas:

ACTIVIDADES	ACTIVIDADES PRECEDENTES
A	-
B	A
C	A
D	A
E	B
F	C
G	D
H	E, F

El grafo es, en este caso:



En este caso la representación resulta sencilla, pero si existe un número grande de actividades se hace preciso ordenar el grafo por niveles. En este caso se utiliza el **algoritmo de Demoucron** que parte de una matriz asociada M a un grafo G de n vértices. Un elemento a_{ij} de la matriz adoptará el valor 1 si existe un arco (actividad) del suceso y al suceso j , y toma el valor 0 si no existe.

Para ello se construye una matriz, que, en su parte izquierda, representa el grafo:

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	0	0	1	1	1	0	0
3	0	0	0	0	0	1	0
4	0	0	0	0	0	1	0
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0

Posteriormente se amplía la matriz con una columna V1 con un número de elementos igual al número de vértices del grafo. El contenido de la columna V1 es V1(i) = $\sum_{j=1}^{j=n} a_{ij}$ es decir, la suma de las filas de la matriz asociada. En la parte inferior se incluyen aquellos sucesos en los que V1(i) = 0, que, en este caso es el vértice 7 (V1(7) = 0). Este vértice formará parte del último nivel:

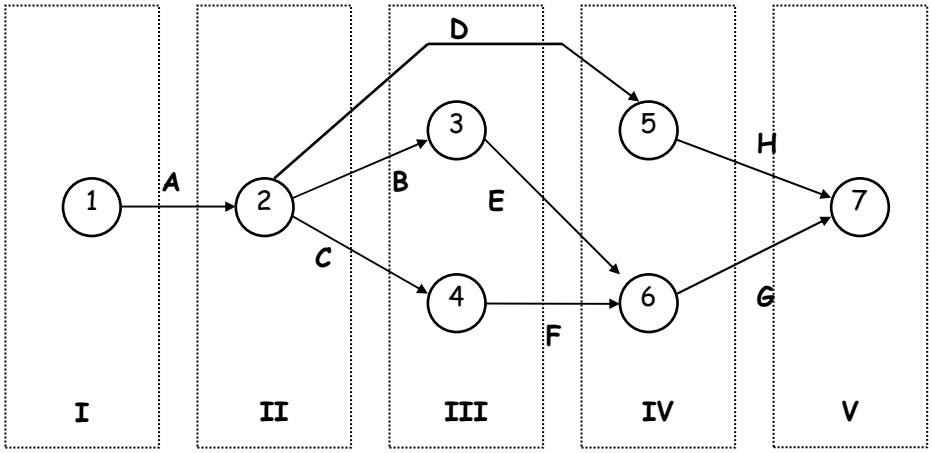
V1
1
3
1
1
1
0
7

A continuación se añade otra columna V2. Para calcular los valores V2(i) de esta columna se buscan primero los valores de la columna anterior donde V1(i) = 0. Los valores de V2(i) se hallan restando a los valores de V1(i) los valores de las columnas en los que los vértices y tienen un valor V1(i) = 0. En el ejemplo considerado, se coge la columna a_{i7} y en la fila y (en este caso las filas 5 y 6) en donde a_{i7} = 1, se resta del valor V1(i).

Los valores que en la columna V1(i) = 0, en la siguiente columna se reemplazan por V1(i) = X. Se realiza el mismo procedimiento hasta llegar al suceso inicial, que estará en el nivel 1

	1	2	3	4	5	6	7	V1	V1	V3	V4	V5
1	0	1	0	0	0	0	0	*1*	*1*	*1*	*1*	*0*
2	0	0	1	1	1	0	0	*3*	*3*	*2*	*0*	*X*
3	0	0	0	0	0	1	0	*1*	*1*	*0*	*X*	*X*
4	0	0	0	0	0	1	0	*1*	*1*	*0*	*X*	*X*
5	0	0	0	0	0	0	1	*1*	*0*	*X*	*X*	*X*
6	0	0	0	0	0	0	1	*1*	*0*	*X*	*X*	*X*
7	0	0	0	0	0	0	0	*0*	*X*	*X*	*X*	*X*
								7	***5***	***3***	***2***	***1***
									6	***4***		
								V	***IV***	***III***	***II***	***I***

El grafo resultante es:



A continuación se estudian las asignaciones de los tiempos de cada actividad. PERT considera que la duración de las actividades es una variable aleatoria de la que se conoce su ley de distribución. Para ello se consideran tres tiempos:

- **Estimación de tiempo pesimista (t_p)** representa el tiempo máximo en el que podría finalizarse la actividad si apareciesen todas las circunstancias negativas que pueden darse durante la ejecución
- **Estimación de tiempo mas probable (t_n)** representa el tiempo normal de duración de la actividad considerando que hay problemas durante las actividades, pero que no aparecen en su totalidad
- **Estimación de tiempo optimista (t_o)** representa el tiempo mínimo si no aparece ningún problema durante la ejecución de la actividad.

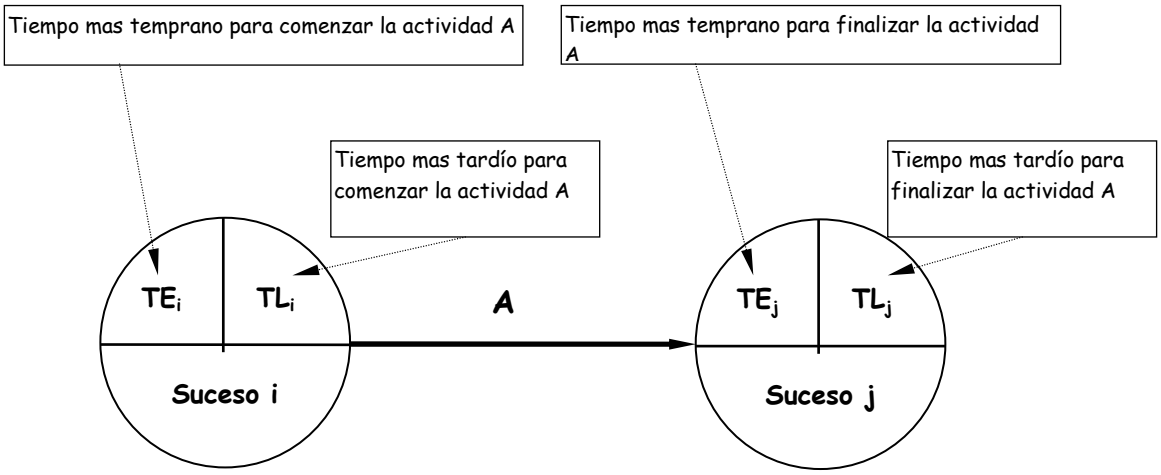
Con estos tres tiempos se determina el **tiempo PERT (T_μ)** dado por:

$$T_\mu = \frac{t_p + 4t_n + t_o}{6}$$

Siendo la varianza:

$$\sigma = \sqrt{\left(\frac{t_p - t_o}{6}\right)^2}$$

Posteriormente se calcula el tiempo **early** (tiempo mas temprano posible) y tiempo **last**(tiempo mas tardío posible) de cada suceso descrito en el grafo (TE_i y TL_i del suceso y respectivamente).



El cálculo del *tiempo más temprano posible (**early**) del suceso j* se calcula:

$$TE_j = \text{máx}\ [TE_i + T_{ij}]\ , \qquad \forall\ j$$

Siendo: TE_i el tiempo early del suceso i y T_{ij} la duración de la actividad que comienza en el suceso y y finaliza en el suceso j.

Análogamente, El cálculo del *tiempo más tardío posible (**late**) del suceso j* se calcula:

$$TL_i = \text{mín}\ [TL_j - T_{ij}]\ , \qquad \forall\ j$$

donde Tl_j es el tiempo last del suceso j y T_{ij} la duración de la actividad que comienza en el suceso y y finaliza en el suceso j.

El tiempo last del último suceso coincide con su tiempo early.

Se define como **Holgura de un suceso** como el número total de unidades de tiempo en las que se puede retrasar su realización de forma que no se aumente la duración total del proyecto. Se define por: $H_i = TL_i - TE_i$

Se dice que un suceso es **crítico** si $H_i = 0$

Se define como **Holgura total de una actividad** que une el suceso y con el suceso j como *el número de unidades de tiempo que puede retrasarse la realización de la actividad con respecto al tiempo PERT previsto sin que aumente la duración del proyecto*

$$H^T_{ij} = TL_j - TE_i - T_{ij}$$

Un aspecto a tener en consideración es que si una actividad consume parte o la totalidad de su holgura total, puede producirse una disminución de la holgura total de la siguiente actividad

Se dice que una actividad es **crítica** si su holgura total es igual a 0.

Uniendo todas las actividades críticas desde el suceso inicial al suceso final se obtiene un camino, **camino crítico**, definido como aquel en el que *cualquier retraso que sufra alguna de las actividades que lo forman implicará un retraso en el proyecto.*

10.3.- Estimación de costes y plazos

A pesar que que se está hablando de "estimación de costes" para proyectos software, los valores obtenidos no suelen medirse directamente en unidades monetarias. Las estimaciones suelen ser valoraciones (con un cierto porcentaje de error) del esfuerzo

esperado para el desarrollo del proyecto y de los plazos de tiempo requeridos para completarlo.

Dado que el software es un producto sin existencia física propia y cuyo coste principal reside en su desarrollo y diseño, es lógico pensar que el principal coste de producción que se asuma esté dominado por los gastos de personal, por lo que, la principal unidad de medición del coste de un proyecto suele ser el *número de salarios mensuales o anuales que deben pagarse*, los salarios suelen especificarse en personas-mes o personas-año.

La estimación de costes pretende responder a dos preguntas: *¿Qué costará?* y *¿Qué plazo de tiempo requerirá su desarrollo?.* Todos los métodos actuales dependen de la cantidad de información disponible. A medida que se avanza en el proyecto, se obtiene una mayor cantidad de detalles y de información mas fiable, por lo que la precisión de la estimación mejora progresivamente. La estimación por tanto, debe ser un proceso continuo con constantes refinamientos y mejoras mas que una actividad puntual.

10.3.1.- Métodos de estimación de costes

Existen cuatro métodos de estimación de costes de proyectos software:

- La **opinión de los expertos** esto es, la "adivinación basada en la experiencia personal"
- La **estimación por analogía**. Consistente en comparar el proyecto que se va a evaluar con uno o mas proyectos terminados de los que se disponen datos. En función de las similitudes y diferencias entre dichos proyectos se deduce el coste del nuevo desarrollo.
- La **descomposición**. Se descompone el producto que se va a construir en sus componentes (tareas sencillas), hasta que se pueda conseguir un nivel de detalle tal que se pueda estimar directamente el coste de cada una de las unidades elementales (por ejemplo usando una red PERT). El coste real del proyecto sera el resultado de sumar las estimaciones de todas las unidades.
- **Ecuaciones y modelos de estimación**. En general son fórmulas matemáticas que relacionan diversos parámetros del proyecto (tamaño del software, condiciones de entorno del proyecto, etc.) con el esfuerzo requerido. Estos modelos se clasifican en:
  - **Modelos empíricos**, basados en opinión de expertos y estimación (top down y botton up), apoyada en datos históricos.
  - **Modelos estadísticos**, obtenidos mediante el análisis de regresión estadística sobre los datos recogidos de una gran cantidad de proyectos.
  - **Modelos basados en una teoría**., que se apoyan en una teoría previa sobre el esfuerzo a desarrollar software
  - **Modelos compuestos**, que incorporan una combinación de distintas técnicas (modelo COCOMO) así como la calibración de datos históricos y el juicio de expertos.

10.4.- Seguimiento y supervisión de un proyecto software

El seguimiento y la supervisión del proyecto software implica seguir, revisar y comparar los logros y los resultados obtenidos frente a las estimaciones, los compromisos y los planes del proyecto, actualizándolos es función de estos resultados, además de detectar si se sigue o no se sigue el plan de proyecto.

Los objetivos que pretende conseguir el seguimiento y la supervisión del proyecto software son:

1° Comparar los resultados actuales con los planes previstos

2° Tomar acciones correctivas cuando existan desviaciones significativas de los planes previstos

3° Acordar compromisos con el personal afectado por las acciones correctivas

10.4.1.- Supervisión de resultados

Para conseguir el primer objetivo se han de desarrollar las actividades siguientes:

- ◊ *Desarrollar estándares que establezcan las condiciones o medidas que deben cumplirse cuando se realicen correctamente las diferentes tareas del proyecto.*
- ◊ *Establecer sistemas de supervisión y de informes, para lo cual hay que determinar: los datos necesarios, quien los recibe y cuando se reciben.*
- ◊ *Medir los resultados, lo que permite determinar la consecución o la desviación de los objetivos y estándares.*

Entre las actividades de supervisión se cuentan:

- * Seguimiento de costes y del calendario
- * Seguimiento de los recursos críticos del proyecto
- * Seguimiento del tamaño de los productos software
- * Seguimiento directo de los aspectos técnicos críticos del proyecto
- * Generación de datos históricos como soporte de estimación de futuros desarrollos
- * Seguimiento de los "hitos"

10.4.2.- Acciones correctivas

El progreso se determina, sobre todo, mediante la comparación con los planes previstos, en los hitos determinados, del tamaño del software, del esfuerzo, del coste y del tiempo empleado en finalizar los diferentes productos. Cuando no se cumplen los planes, se ejecutan acciones correctivas que pueden incluir desde la revisión del plan de desarrollo del software para reflejar los logros reales hasta la replanificación del trabajo.

En este último caso, suele declararse no válido el plan previo y volver a planificar para que la situación real se refleje en el plan del proyecto, si bien esta replanificación continua sólo consigue reducir el tiempo y el presupuesto disponibles para las fases posteriores del proyecto, lo que puede conducir a situaciones críticas

En general, las **acciones correctivas** suelen ser:

- * Añadir personal
- * Reducir el alcance o el contenido de una entrega
- * Alargar o retrasar el calendario

10.5.- Gestión de riesgos del software

Debido al gran porcentaje de proyectos cancelados, entregados fuera de plazo, presupuestos excedidos, problemas operativos, conflictos con los usuarios, etc. surge el tratamiento de riesgos software como un factor importante en la realización de un proyecto. Puede definirse el **riesgo** como cualquier elemento potencial que provoca

resultados insatisfactorios en un proyecto, es decir, *riesgo es la probabilidad de que en un punto del ciclo de vida no se alcancen los objetivos propuestos con los recursos disponibles*.

Las áreas o clases típicas de riesgo que se deben tratar son las siguientes:

- **Riesgos estratégicos.-** o riesgos relacionados con la estrategia de la organización, es decir, relacionados con las pérdidas y beneficios de la organización, su imagen , sus inversiones, etc.
- **Riesgos comerciales**.- relacionados con la venta del proyecto, el seguimiento del cliente, el precio, las posibles actualizaciones,...
- **Riesgos contractuales y financieros**.- relacionados con los términos contractuales negociados: penalizaciones, niveles de calidad, control de necesidades de evolución, calendarios de pagos, obligaciones, etc.
- **Riesgos de gestión**.- relacionados con la organización del proyecto: recursos y equipos, calendarios, estimaciones, subcontratas, etc.
- **Riesgos del proyecto**.- relacionados con la especificación, diseño, realización, integración y validación del proyecto.
- **Riesgos de explotación**.- referidos a fallos que puedan ocurrir durante la explotación.
- **Riesgos de mantenimiento**.- causados por sobrecostes en mantenimiento correctivo, mantenimiento preventivo y soporte.

10.6.- Implantación de proyectos software

En esta fase se debe tener en cuenta todas las tareas a realizar y los factores que rodean la puesta en marcha del sistema, contemplando:

- Formación y entrenamiento del usuario
- Planificación y modo de implantación
- Recursos necesarios

10.6.1.- Formación del usuario

En principio, la formación debe llegar a toda persona de las áreas afectadas por la mecanización que vayan a ser utilizadores del sistema. El **plan de formación** contendrá pues qué usuarios van a ser formados, quién les va a formar, sobre qué temas, cuando y donde se va a efectuar la formación.

Una técnica normalmente empleada es que la formación la impartan personas del grupo de desarrollo con aptitudes pedagógicas y organizativas y con un buen trato con los usuarios. En cuanto a los usuarios, dependiendo de la envergadura del sistema, suelen formarse todos los usuarios si el área afectada es pequeña.

Si el sistema es de tipo medio o grande, suelen formarse a un grupo reducido de usuarios para que, a su vez, sean monitores del resto de sus compañeros.

Otro factor importante de cara a los usuarios es la documentación y ayudas on-line de las que va a disponer en su entorno de trabajo. Sobre este aspecto, la tendencia general es hacer lo mas amigablemente posible la relación del usuario con la máquina, de forma que el usuario disponga de ayudas en todo momento y de información sobre la operación a realizar que pueden aparecer en pantalla por medio de ventanas a diferente nivel de detalle. Esta tendencia presenta dos ventajas claras:

1. Es mas rápida y fácil de utilizar por parte del usuario, que no tiene que acudir a un voluminoso manual
2. Es mas fácil, por parte del diseñador, actualizar las ayudas del sistema que un número grande de manuales distribuidos entre los usuarios. Esto último normalmente no se hace y, debido a las modificaciones que va teniendo el sistema, los manuales quedan obsoletos.

10.6.2.- Modos de implantación

Hay diferentes planteamientos en cuanto al modo de implantar el nuevo sistema dependiendo del tipo de organización, estado previo de informatización y nivel de los usuarios afectados.

Si el sistema estaba informatizado previamente, habrá que definir diferentes **programas de conversión** y planificar el calendario de su ejecución.

En general, habrá que distinguir tres enfoques de implantación:

IMPLANTACIÓN DIRECTA.- Cuando se decide pasar del sistema anterior al nuevo sistema en una fecha determinada y sin procesos paralelos de contraste de resultados.

IMPLANTACIÓN EN PARALELO.- Durante un cierto periodo de tiempo se tienen en funcionamiento los dos sistemas, el nuevo y el viejo, controlando los resultados obtenidos en ambos sistemas y ajustando el nuevo sistema.

IMPLANTACIÓN EN CENTRO PILOTO.- Se seleccionan uno o varios centros de experimentación del sistema de modo que el funcionamiento del sistema en estas unidades aporte información sobre problemas o defectos del sistema.
Tiene la ventaja de requerir menor recursos que en paralelo y, si los centros elegidos son suficientemente representativos, permite afinar el sistema en un porcentaje muy alto.

10.6.3.- Recursos Necesarios para la implantación

Los recursos necesarios para la implantación del sistema están destinados a hacer lo mas cómoda posible la transición del sistema viejo al nuevo.

Deberá pues planificarse con todo detalle el calendario de instalación de todos los medios necesarios para arrancar el sistema: hardware, instalaciones, materiales, adecuación de locales, etc. La implantación del sistema requiere pues una planificación y una organización de recursos en la que el propio software es un recurso mas. Es una etapa crítica de la que depende el trabajo de muchos meses y muchas personas.

10.7.- Mantenimiento del software

Los modelos de ciclo de vida tradicionales representan el mantenimiento como una fase que comienza una vez se han finalizado las pruebas. Distintos estudios indican que el coste de las actividades de mantenimiento representa entre un 80 % y un 95 % del presupuesto total de los distintos Centros de Procesos de Datos, habiéndose superado, en algunas empresas, este límite, hasta llegar al límite de recursos (*barrera de mantenimiento*) lo que les imposibilita a acometer nuevos desarrollos.

En resumen, se puede asegurar que el mantenimiento es la fase dominante y mas costosa del ciclo de vida

Los factores que afectan directamente a estos costes son:

- Inexistencia de métodos, técnicas y herramientas que puedan proporcionar una solución global al mantenimiento
- La complejidad de los sistemas se incrementa paulatinamente por la realización de continuas modificaciones
- La documentación del sistema es defectuosa o inexistente
- Se considera el mantenimiento como actividad poco creativa y, por tanto, mas sencilla y menos importante que el desarrollo.
- Las actividades de mantenimiento se suelen realizar bajo presión del tiempo
- Poca participación del usuario durante el desarrollo del sistema.

Muchos de estos problemas tienen su origen en el desconocimiento de las actividades realizadas durante el mantenimiento. Últimamente se identifican varias actuaciones comunes para mantener la operatividad del software:

- Corrección de defectos del software
- Creación de nuevas funcionalidades en el software por nuevos requisitos del usuario
- Mejora de la funcionalidad y del rendimiento.

Con todo lo expuesto, puede definirse el mantenimiento como "*el proceso de modificar un sistema o un componente software después de su entrega para corregir defectos, mejorar el rendimiento u otros atributos o adaptarlo a un entorno cambiante*".

10.7.1.- Tipos de mantenimiento del software

Según la definición anterior existen diferentes tipos de mantenimiento:

MANTENIMIENTO PERFECTIVO.- Conjunto de actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario.

MANTENIMIENTO ADAPTATIVO.- Conjunto de actividades que se realizan para adaptar el sistema a los cambios en su entorno tecnológico. Estos cambios pueden ser:

El entorno de datos: Cambio del soporte de los datos de una aplicación

El entorno de proceso: Cambio de plataforma de explotación o de sistema operativo

MANTENIMIENTO CORRECTIVO.- Conjunto de actividades dedicadas a corregir defectos de hardware o de software detectados por los usuarios durante la explotación del sistema.. Entre estos fallos se pueden considerar:

Procesamiento: Terminaciones anormales o salidas incorrectas del programa

Rendimiento: Tiempos de respuesta por encima de los necesarios

Implementación: Violaciones de estándares de programación o inconsistencias del diseño del programa.

MANTENIMIENTO PREVENTIVO.- Conjunto de actividades para facilitar el mantenimiento futuro del sistema

Por otra parte, si se analizan las actividades que deben realizar los programadores, pueden establecerse las siguientes:

- Estudiar las peticiones
- Estudiar la documentación
- Estudiar el código

- Implementar el cambio
- Realizar pruebas
- Actualizar la documentación del programa.

10.8.- La reingeniería del software

La reingeniería del software se concibe como una tarea nueva, dentro de la ingeniería del software, que engloba un gran conjunto de actividades y estrategias tanto para la reducción del esfuerzo de mantenimiento de los sistemas como para la reutilización de componentes de sistemas existentes.

Estas actividades pueden dividirse en tres grupos:

TECNOLOGÍA DE LA REINGENIERÍA

MEJORA DEL SOFTWARE	Reestructuración, Redocumentación, anotación, actualización de la documentación Ingeniería para reutilización Remodularización Reingeniería de procesos de negocio Reingeniería de datos Análisis de facilidad de mantenimiento, análisis económico
COMPRENSIÓN DEL SOFTWARE	Visualización Análisis, mediciones Ingeniería inversa, recuperación de diseño
CAPTURA, CONSERVACIÓN Y EXTENSIÓN DEL CONOCIMIENTO SOBRE EL SOFTWARE	Descomposición Ingeniería inversa y recuperación de diseño Recuperación de objetos Comprensión de programas Transformaciones y bases de conocimiento

La reingeniería del software:

* Puede reducir los riesgos evolutivos de una organización
* Puede ayudar a las organizaciones a recuperar sus inversiones en software
* Puede hacer el software mas facilmente modificable
* Amplía las capacidades de las herramientas CASE
* Es un catalizador para la automatización del mantenimiento del software
* Puede actuar como catalizador para la aplicación de técnicas de inteligencia artificial para resolver problemas de reingeniería.

10.8.1.- Conceptos en Reingeniería del software

Ingeniería directa.- Desarrollo del software tradicional. El adjetivo directa se utiliza exclusivamente para diferenciarlo del concepto de ingeniería inversa.

Reestructuración.- Modificación del software para hacerlo mas fácil de entender y cambiar.

Ingeniería inversa.- Proceso de análisis de un sistema para identificar sus componentes e interrelaciones y crear representaciones del sistema de otra forma o a un nivel mas alta de abstracción

Redocumentación.- Creación o revisión de una representación equivalente semánticamente dentro del mismo nivel de abstracción relativo.

Recuperación de diseño.- Subconjunto de la ingeniería inversa, en el cual, aparte de las observaciones del sistema, se añaden conocimientos sobre su dominio de aplicación, información externa y procesos deductivos con el objetivo de identificar abstracciones significativas a un mayor nivel.

Rediseño.- Consolidación y modificación de los modelos obtenidos, añadiendo nuevas funciones requeridas por los usuarios. Esta ampliación de funcionalidades se realiza a nivel de análisis o diseño (y no del código) y bajo soporte de herramientas CASE, lo que facilitará, posteriormente, la generación de código.

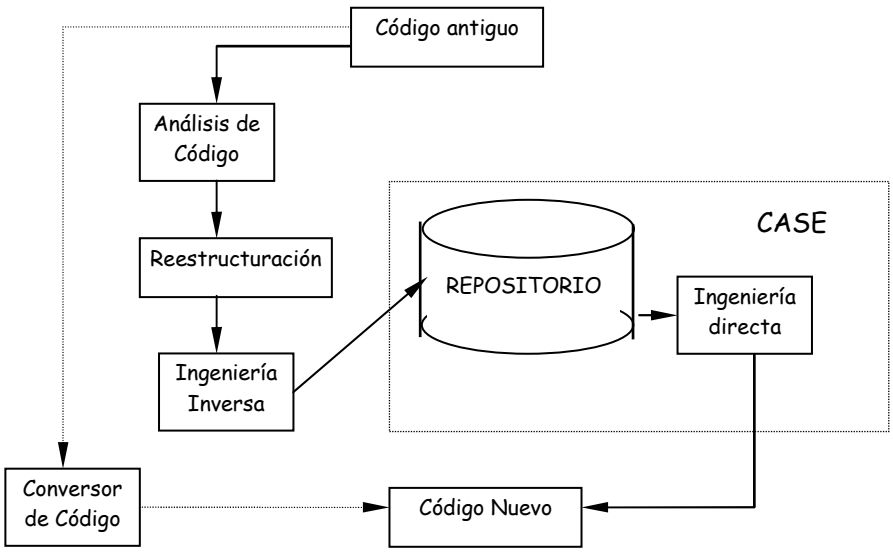
Reingeniería de software.- Examen y alteración de un sistema para reconstruirlo de una nueva forma y la subsiguiente implementación de esta nueva forma. Esto es lo mismo que decir que es cualquier actividad que:

- mejore el entendimiento acerca del software
- prepare o mejore el propio software normalmente para incrementar su facilidad de entendimiento, reutilización o evolución.

El proceso de reingeniería del software sigue los siguientes pasos:

1. Análisis del código
2. Reestructuración
3. Ingeniería inversa
4. Rediseño
5. Ingeniería directa

Los pasos de la reingeniería del software son:



El paso de un código de un lenguaje a otro se puede hacer mediante herramientas denominadas **conversores**. Con este tipo de herramientas se pueden cargar datos a un **repositorio**, lo que imposibilita la adición de nuevas funciones.

Como se puede observar en la figura, las herramientas CASE que soportan el proceso de ingeniería directa, deberían implementar en el futuro las funciones de análisis de código, reestructuración e ingeniería inversa. Este soporte se suele realizar con herramientas

aisladas que, en la mayoría de los casos, no se pueden integrar con herramientas CASE. En la actualidad se están ampliando las funcionalidades de las herramientas CASE para soportar, en algún grado, los aspectos descritos, aunque los resultados son aún limitados.