Tema 1

La clave para que nuestro proyecto tenga éxito es mantener una **buena gestión** del mismo, por lo tanto, **ES RESPONSABILIDAD DE LOS GESTORES**:

a) Planificar el proceso de desarrollo
b) Hacer un seguimiento del trabajo de forma que:
   - Cumpla con los estándares establecidos
   - Sigua con la agenda planificada (QUE, COMO, CUANDO, QUIEN)
   - No se sobrepase el presupuesto

Esto no se consigue sin tener en cuenta las siguientes claves:

- **Personal (Esfuerzo humano)**: Consiste en mantener una buena comunicación y relación con el personal que trabaja en el proyecto, saber elegir a un líder (jefe de equipo) que sea capaz de guiar, motivar y establecer una estructura cohesionada entre el personal, ha de contar con la habilidad de ser capaz de gestionar un proyecto.

  Es de mera importancia que el equipo vea el proyecto como una misión a realizar en vez de un trabajo forzado para incrementar así la motivación de los miembros.

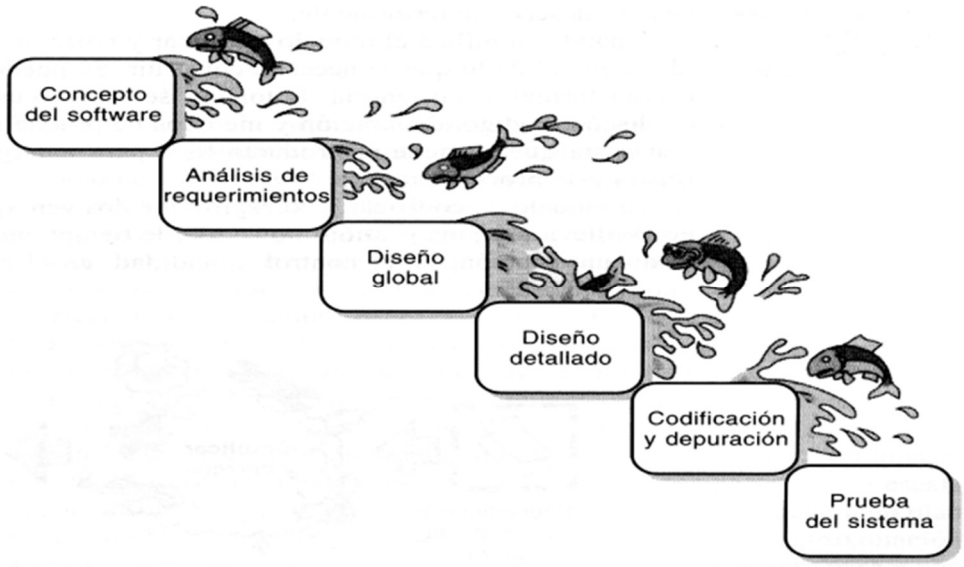
  Podemos decidir organizar nuestro personal en equipos o no, respecto a los equipos, hemos de destacar varios puntos:

  - El rendimiento de un equipo es INVERSAMENTE proporcional a la cantidad de comunicación que se deba entablar.
  - El tiempo en que los miembros del equipo convivan afectará a la moral de este.
  - Factores útiles para seleccionar al personal son: experiencia en el campo en el cualtrabaja el equipo, adaptabilidad y personalidad.

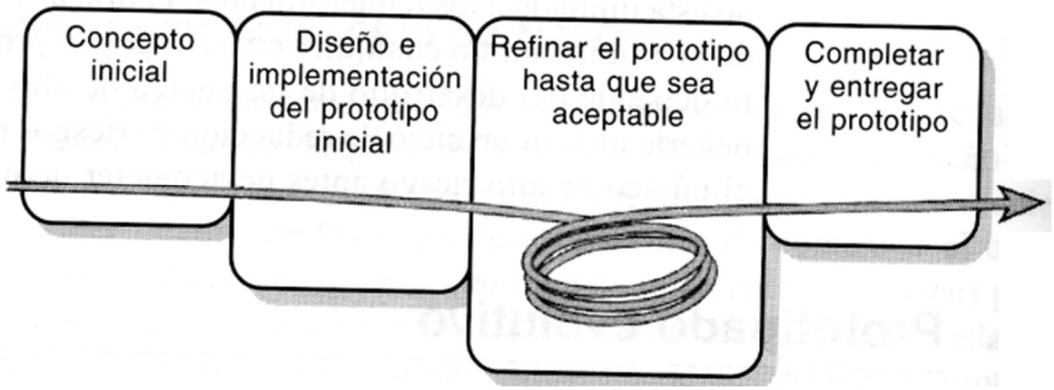
  A la hora de hablar de organización de equipo distinguimos tres tipos – **descentralizado democrático** (DD), **descentralizado controlado** (DC) y **centralizado controlado** (CC) – hay que tener en cuenta que no hay una estructura única de equipo mejor para todos los proyectos.

		DD	DC	CC
DIFICULTAD	ALTA	■		
	PEQUEÑA		■	■
TAMAÑO	GRANDE		■	■
	PEQUEÑO	■		
DURACION EQUIPO	CORTO		■	■
	LARGO	■		
MODULARIDAD	ALTA		■	■
	BAJA	■		
FIABILIDAD	ALTA	■	■	
	BAJA			■
FECHA ENTREGA	EXTRICTA			■
	FLEXIBLE	■	■	
COMUNICACIÓN	ALTA	■		
	PEQUEÑA		■	■

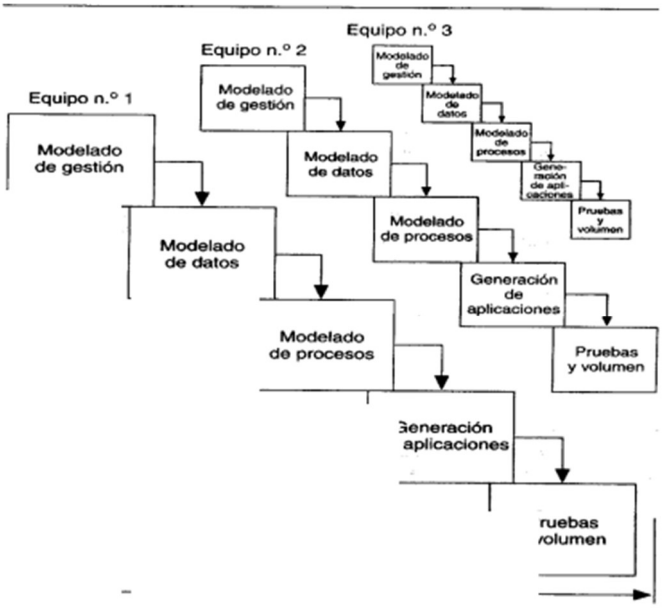
- **Problema (Minuciosa comunicación con cliente)**: Estimaciones cuantitativas. Pasos:
  - **Ámbito del software:** Contexto, Objetivos de información, Función y rendimiento.
  - **Descomposición del problema:** Funcionalidad y proceso
- **Proceso (Métodos técnicos y herramientas):** Se tiene que elegir el modelo de proceso adecuado para la ingeniería del software que debe aplicar el equipo del proyecto (Los proyectos pequeños necesitan menos tiempo para su desarrollo). Algunos modelos de proceso secuenciales:
  - **Modelo de ciclo de vida en Cascada:** Acentúa el fracaso frente al usuario final, se tarda mucho tiempo en pasar por todo el ciclo dado que hasta que no se termina una fase no se pasa a la siguiente, raramente es usado por algún equipo de desarrollo de software, por lo tanto no refleja el proceso real del mismo.



  - **Modelo de construcción de prototipos**

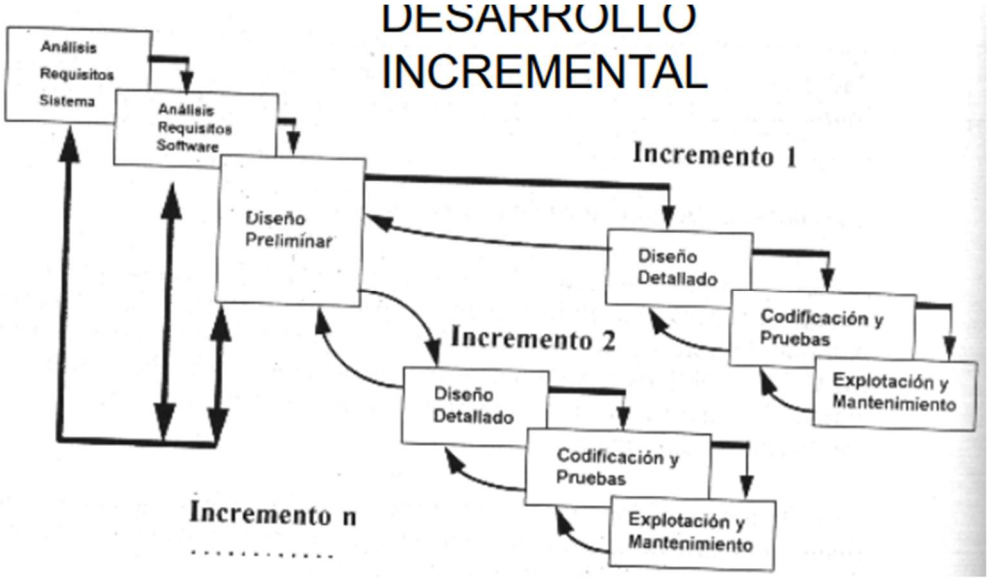


- **Modelo DRA (Desarrollo Rápido de Aplicaciones):**

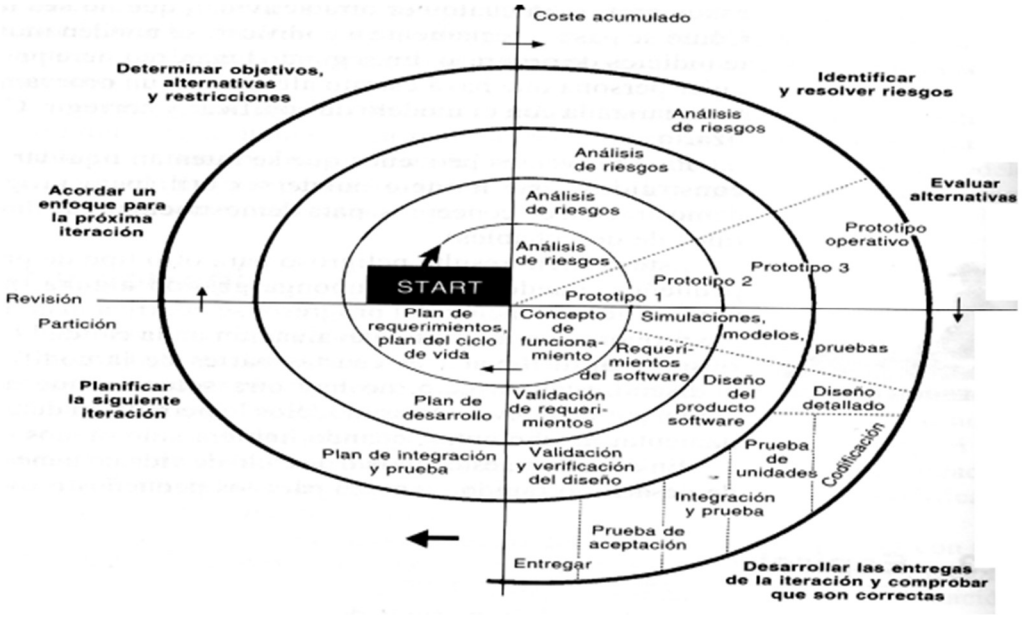


Modelos evolutivos:

- **Modelo de desarrollo incremental**



- **Modelo en espiral**



- **Desarrollo unificado:**
  - **Inicio:** Se define el ámbito del proyecto y se desarrollan los casos de uso
  - **Elaboración:** Plan de proyecto, estimaciones y diseño básico.
  - **Construcción:** Se implementa en base a Iteraciones: Secuencias de actividades con un plan establecido y un criterio de evaluación que resulta en una versión interna nueva. Al final de un ciclo obtenemos una versión para el cliente.
  - **Transición:** Fase de transición para entregar el producto a los usuarios (p.ej. pruebas beta)

Tema 2

Estimación: Consiste en predecir los recursos (monetarios, temporales, humanos, materiales, ...) necesarios para llevar a cabo el proceso de desarrollo del software.

Componentes principales del coste:

- Costes hardware y software
- Costes de viajes y aprendizaje
- Costes de esfuerzo, factor dominante (sueldo ingenieros, gastos seguros y seguridad social)
- Otros costes (alquiler, luz, redes, recursos compartidos...)

Factores del coste

- **Oportunidad de mercado:** Calidad/precio, diferenciar mi producto de la competencia. Una organización de desarrollo puede cotizar un precio bajo porque desea entrar en un nuevo segmento del mercado de software. Aceptar una baja ganancia en uno proyecto puede dar la oportunidad de obtener más ganancias más adelante. La experiencia adquirida puede permitir la creación de nuevos productos.

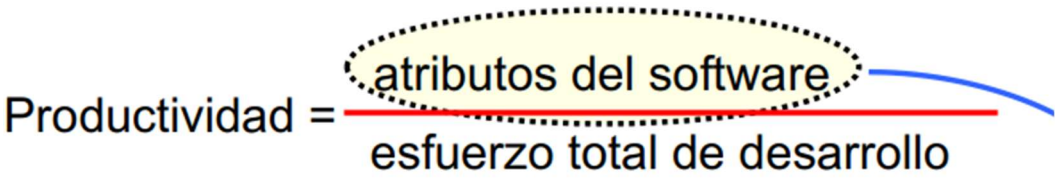
- **Incertidumbre en la estimación de costes:** Guardar datos de anteriores proyectos para poder estudiar errores cometidos para así corregirlos.
  Si una organización no está segura de su estimación de costos, puede aumentar su precio por alguna contingencia sobre y por encima de su beneficio normal.

- **Términos contractuales:** Venta de licencias = minimización de costes, manteniendo nosotros la propiedad del software y sus derechos comerciales.
  Un cliente puede estar dispuesto a permitir que el desarrollador retener la propiedad del código fuente y reutilizarlo en otros proyectos. El precio cobrado d puede entonces ser menor que si el código fuente del software se entrega al cliente.

- **Volatibilidad de los requerimientos:** Mantener mejor comunicación con el cliente a lo largo del proyecto.
  Si es probable que los requisitos cambien, una organización puede bajar su precio para ganar un contrato. Después de la adjudicación del contrato, los precios altos pueden ser cobrado por cambios en los requisitos

- **Salud financiera:** Cobrar un anticipo en la firma del contrato.
  Los desarrolladores con dificultades financieras pueden reducir sus precios para ganar un contrato. Es mejor hacer un pequeño beneficio o punto de equilibrio que salir del negocio

Productividad: La productividad de un programador es la velocidad a la que los ingenieros implicados en el desarrollo del software producen dicho software y su documentación.



Podemos medir la productividad con relación al tamaño **(líneas de código)** o en relación a la funcionalidad (puntos de función, puntos objeto).

Medir con relación al tamaño:

- Cuanto mayor sea la expresividad del lenguaje, más baja será su productividad aparente. (asm vs C++)
- Cuantas más líneas de código emplee el programador, mayor será su productividad.

Por lo tanto, podremos decir que comparar la productividad utilizando lenguajes diferentes puede llevar a conclusiones erróneas respecto a la productividad de los programadores.

Medir con relación a la funcionalidad:

- **Puntos de función:** La técnica de estimación de puntos de función se basa en la contabilización de unos contadores (características del programa):
  - Entradas y salidas externas
  - Interacciones de usuario
  - Interfaces externas
  - Ficheros usados por el sistema

  Se asocia un peso con cada uno de estos contadores y los puntos de función se calculan multiplicando cada factor por su peso y sumando todos ellos.

  Ventajas frente a líneas de código:
  - Independientes del lenguaje de programación
  - Se pueden calcular a partir de la especificación
  - Usa información del dominio del problema
  - Resulta más fácil a la hora de reusar componentes
  - Se encamina a aproximaciones orientadas a objetos

  Uso: Los Puntos de función pueden usarse para estimar el número de líneas de código* para un lenguaje dado(**LOC** = AVC * número de puntos de función). AVC es un factor dependiente del lenguaje (asm vs 4GL [Lenguajes de 4ª generación (alto nivel)]).
  Problema: Un problema que nos encontramos es que los puntos de función son muy subjetivos y son totalmente dependientes del estimador.

- **Puntos objeto:** Medida alternativa relacionada con la funcionalidad cuando se utilizan leguajes 4GL o similares para el desarrollo. El número de puntos de objeto en un programa es una estimación ponderada de:
    - Número de pantallas visualizadas por separado
    - Número de informes que el sistema produce
    - Módulos 3GL que deben desarrollarse para completar el código 4GL

  Ventajas frente a puntos de función
    - Son más fáciles de estimar a partir de una especificación que los puntos defunción, ya que solo se consideran pantallas, informes y módulos 3GL.
    - Pueden estimarse en fases tempranas del desarrollo (en estas etapas resulta muy difícil estimar el LOC de un sistema)

Algunos factores que afectan a la productividad son: Experiencia en el dominio de la app, calidad del proceso, tamaño del proyecto, entorno de trabajo...

Todas las métricas basadas en volumen/unidad de tiempo son engañosas debido a que no tienen en cuenta la calidad. Las métricas de productividad deberían usarse únicamente como guía

Técnicas de estimación (No existe una forma simple de obtener estimaciones exactas del esfuerzo requerido para desarrollar un sistema software)**:**

- **Modelado algorítmico de costes:** Aproximación que emplea fórmulas obtenidas a partir de información histórica. Suele basarse en el tamaño del software (componente exponencial). El tamaño de un sistema software puede conocerse con exactitud solo cuando está terminado, además, el mismo está condicionado por factores como el lenguaje de programación utilizado. Esta estimación de tamaño se va realizando de forma más precisa conforme el desarrollo del sistema avanza.
- **Juicio experto:** Un grupo de expertos utilizan su experiencia para predecir los costes de software. Se realizan iteraciones hasta que se alcanza un consenso.
  Ventajas: Método barato, además la estimación puede ser bastante precisa si se dispone de los expertos adecuados.
  Desventajas: En el caso de no disponer de los expertos adecuados puede llegar a ser un método muy impreciso.
- **Estimación por analogía:** El coste del proyecto se calcula por comparación con proyectos similares.
  Ventajas: Preciso si se disponen de los datos adecuados.
  Desventajas: Imposible de realizar si no hay proyectos comparables. Necesita un mantenimiento sistemático de una base de datos.
- **Ley de Parkinson:** Los costes del proyecto están en función de los recursos disponibles, utilizando todo el tiempo permitido.
  Ventajas: No realiza presupuestos “abultados”.
  Desventajas: El sistema normalmente no termina. (Normalmente se necesitan más recursos y tiempo del estimado)

- **Pricing to win:** El coste del proyecto está en función de lo que el cliente está dispuesto a pagar.
  Ventajas: La empresa desarrolladora consigue el contrato.
  Desventajas: La probabilidad de que el cliente obtenga el sistema deseado es pequeña, además, los costes no reflejan el trabajo requerido.

Estimación ascendente y descendente: Cualquiera de las aproximaciones vistas anteriormente pueden utilizarse de forma ascendente o descendente:

- **Estimación descendente:** Evalúa la totalidad de funcionalidades del sistema y cómo éstas se subdividen en subsistemas. **Uso:** Se puede usar sin conocer la arquitectura del sistema, tiene en cuenta costes como la integración o la documentación. **Problema:** Puede infraestimar costes relacionados con la resolución de problemas de bajo nivel.
- **Estimación ascendente:** Estima el esfuerzo requerido para cada componente del sistema. Dichos esfuerzos se añaden a la estimación final. **Uso:** Se puede usar cuando la arquitectura del sistema es conocida y los componentes identificados, proporciona estimaciones bastante exactas si el sistema se ha diseñado con detalle. **Problema:** Puede infraestimar costes a nivel de sistema (integración, documentación...)

En conclusión: La estimación debería basarse en varios métodos, si el resultado de aplicar varios de ellos difiere mucho, es que no disponemos de suficiente información. Muchas veces el método Pricing to Win es el único aplicable.

Modelo COCOMO (Constructive Cost Model): Modelo empírico basado en la experiencia con proyectos grandes. La última versión COCOMO 2 es la que veremos en más profundidad. COCOMO 2 se trata de un modelo de 3 niveles que permite estimaciones cada vez más detalladas y que pueden realizarse a la vez que progresa el proyecto:

- **Nivel inicial de prototipado:** Estimaciones realizadas con puntos objeto y una fórmula simple para el cálculo del esfuerzo (PM = (NOP x (1 - %reuse/100)) / PROD). Donde **PM es el esfuerzo en personas-mes**, NOP es el número de puntos de objeto, y PROD es la productividad.
- **Nivel inicial de diseño:** Estimaciones realizadas con puntos de función convertidas en líneas de código. Formula: [PM = A x Tamaño_B x M + PM_m]
  M = PERS x RCPX x RUSE x PDIF x PREX x FCIL x SCED
  PM_m = (ASLOC x (AT/100)) / ATPROD
  A = 2’5 según la calibración inicial, Tamaño se da en KLOC y **B** va desde 1’1 a 1’24.
  Los multiplicadores reflejan la capacidad de los desarrolladores, la familiaridad con la plataforma de desarrollo, etc.
- **Nivel post-arquitectura:** Estimaciones basadas en líneas de código fuente y emplea la misma fórmula que la estimación anterior (estimación inicial de diseño). Se ajusta la estimación de tamaño aún más.

ESLOC = ASLOC × (AA + SU +0.4DM + 0.3CM +0.3IM)/100

- **ESLOC** es el número de líneas de código nuevo. **ASLOC** es el número de líneas de código reusable que debe modificarse, **DM** es el porcentaje de diseño modificado, **CM** es el porcentaje de código que se modifica, **IM** es el porcentaje del esfuerzo original de integración del software reusado.
- **SU** es un factor basado en la interpretación del coste del software, **AA** es un factor que refleja los costes de evaluación iniciales para decidir si el software puede reutilizarse.

Sobre el término B exponente antes comentado, es necesario decir que depende de 5 factores de escala, la suma de éstos se divide por 100 y se añade a 1’01. Factores (+ ejemplo):

- **Antecedentes:** Experiencia previa - Proyecto nuevo – 4
- **Flexibilidad desarrollo:** Nivel de flexibilidad. No implicación del cliente – Muy alto – 1
- **Arquitectura/resolución riesgos:** Riesgos safety. No análisis de riesgos – Muy bajo – 5
- **Cohesión del grupo:** Cómo de bien trabaja el equipo. Algún control – nominal – 3
- **Madurez proceso:** Madurez de la organización. Algún control – nominal – 3

El factor de escala en el ejemplo dado es 1’17.

Puntos clave:

- Es necesario estimar costes: (esfuerzo, tiempo de desarrollo y número de recursos)
- La productividad es un factor a tener en cuenta a la hora de realizar estimaciones
- Existen varias técnicas de estimación de costes. La estimación algorítmica de costes es difícil al necesitar una estimación previa de atributos del producto terminado
- Los modelos de estimación algorítmicos suponen una opción de análisis cuantitativo
- El tiempo necesario para completar un proyecto no es proporcional al número de personas que trabajan en el mismo

Resumen Estimación

```
Tecnicas de estimacion de costes
    -Modelado algorítmico de costes
        Aproximacion basada en tamaño software
        Los costes crecen de forma exponencial con tamaño
        Como el tamaño solo se conoce al final del desarollo, a medida que avanza
        el proyecto la estimacion se hace mas precisa
    -Juicio experto
        Expertos en el campo predicen el coste. Iteracion hasta consenso
        Muy barato y exacto. Cuesta encontrar expertos
    -Estimacion por analogia
        Comparar con proyectos similares
        Preciso si hay info. Imposible si no hay proyectos parecidos
    -Ley de Parkinson
        Coste en funcion de recursos disponibles, usando todo el tiempo
        Presupuesto NO abultados. Suele faltar tiempo
    -Pricing to win
        Coste cuanto quiera pagar cliente
        Costes no reflejan trabajo requerido. Raro obtener programa que cumpla lo que pides

        ·Estimacion Descendente
            Se puede usar sin conocer arquitectura ni componentes
            Tiene encuenta integracion, gestion de config y documentacion
            Infra-estima costes de resolucion problemas tecnicos dificiles

        ·Estimacion Ascendente
            Se puede usar con arquitectura y componentes conocidos
            Estimaciones bastante exactas si se diseña con detalle
            Infra-estima costes de sistema, integracion y documentacion

    Estimacion deberia basarse en varios metodos
    Si resultados de varios metodos difiere mucho, no hay suficiente info.
    Muchas veces Pricing to Win una opcion
```

Tema 3

Proceso de planificación:

- **QUÉ** hay que hacer
- **CÓMO** hay que hacerlo
- **CUÁNDO** se va a hacer
- **QUIÉN** lo va a hacer

Organización de las actividades: Las actividades se deben organizar de forma que el desarrollo pueda progresar favorablemente. **HITOS:** O milestone en ingles, marcan el final de una actividad del proceso de desarrollo. **ENTREGAS:** Resultados del proyecto que se entregan a los clientes. El proceso en cascada permitirá identificar de forma sencilla los hitos que marcan la continuidad del proyecto. **SCHEDULING**: Asignación de recursos a las actividades de un proyecto (Pasos por orden: Determinar actividades a realizar, asignar tiempos estimados, asignar recursos, organización temporal de las actividades)

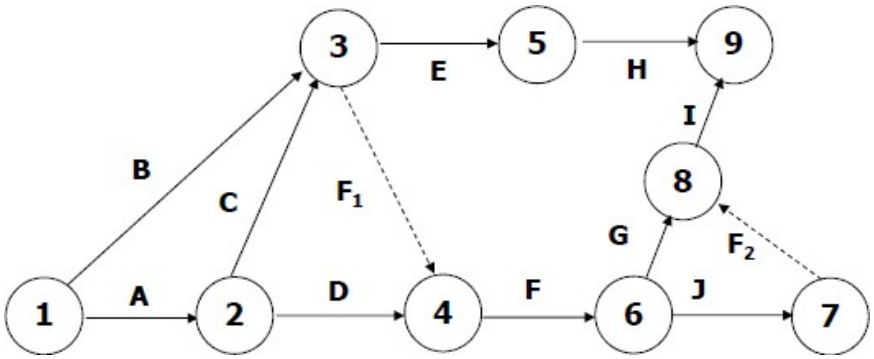
Representaciones gráficas: Permiten mostrar una vista de la división en tareas del proyecto, podremos ilustrar la agenda del proyecto. Hay de dos tipos:

- Diagramas de actividades: Muestran las dependencias entre tareas y el camino crítico
- Diagrama de barras: Muestran la agenda del proyecto

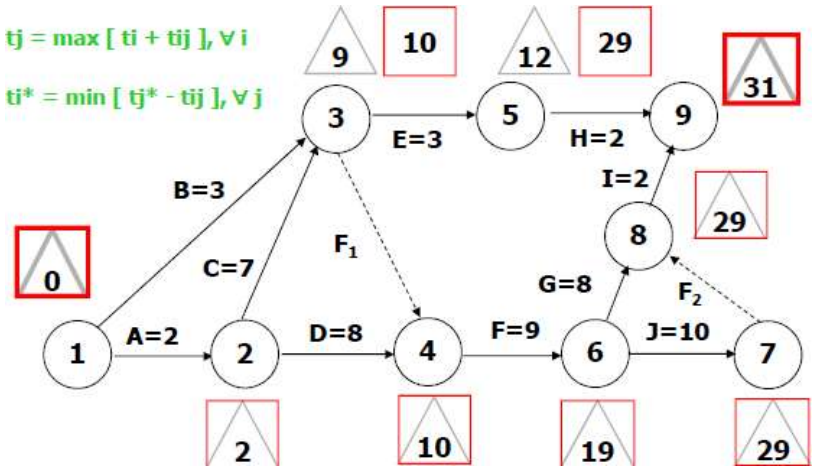
GRAFO PERT: Para entenderlo lo mejor es un ejercicio:

Actividades	Precedentes	Duraciones
A	--	2
B	--	3
C	A	7
D	A	8
E	B, C	3
F	B, C, D	9
G	F	8
H	E	2
I	G, J	2
J	F	10

Dibujar el grafo



Tiempos last y early



Holguras y camino crítico

$$H^T_{ij} = t_j^* - t_i - t_{ij}$$

ACTIVIDAD (i-j)	A (1-2)	B (1-3)	C (2-3)	D (2-4)	E (3-5)	F (4-6)	G (6-8)	H (5-9)	I (8-9)	J (6-7)
HT_{ij}	0	7	1	0	17	0	2	17	0	0

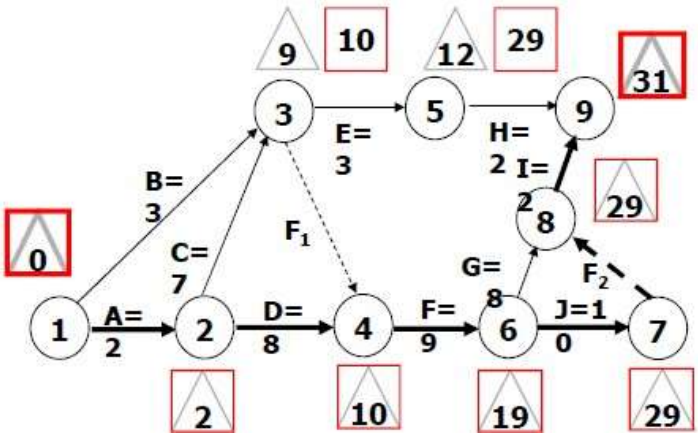
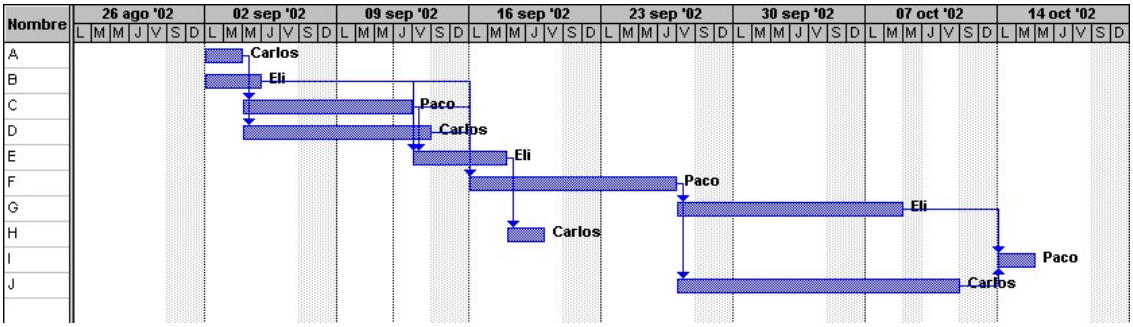


Diagrama Gantt: El eje de ordenadas representa actividades o recursos (vertical), mientras que el eje de abscisas representa el tiempo. Un diagrama de Gantt nos permitirá observar con detalle la evolución del proyecto. MS Project permite hacer Gantts y Perts.

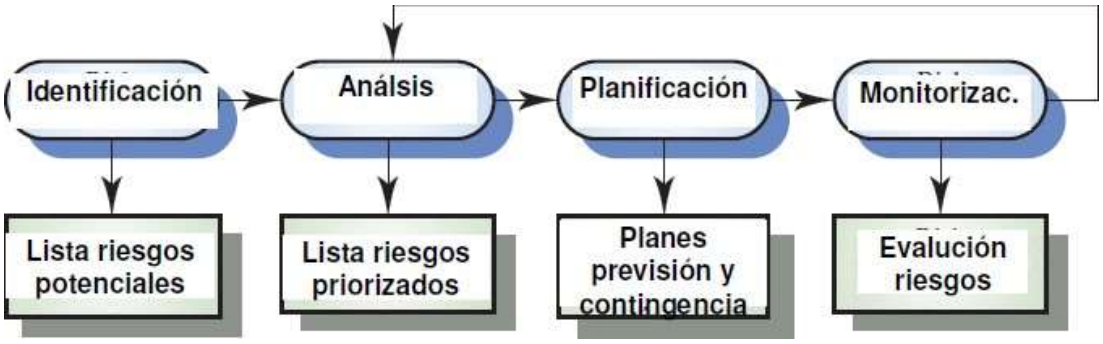


Gestión de riesgos: Un riesgo es una probabilidad de que pueda ocurrir alguna circunstancia adversa.

- Riesgos de **proyecto**: Afectan a la agenda o los recursos
- Riesgos del **producto**: Afectan a la calidad o realización del desarrollo
- Riesgos del **negocio**: Afectan a la organización que desarrolla o que gestiona el proyecto.

El proceso de gestión de riesgos sigue los siguientes pasos (**Cae mucho – veremos este año**):

- Identificación de riesgos
- Análisis de riesgos
- Planificación de los riesgos
- Monitorización de los riesgos



Ejemplo de plan de riesgos:

Identificar riesgos:

a. Es imposible seleccionar personal con las habilidades requeridas para el proyecto.

b. Baja en el personal.

c. Los problemas financieros en la organización causan reducciones en el presupuesto del proyecto.

Analizar riesgos:

a. Probabilidad alta. Efecto catastrófico.

b. Probabilidad media. Efecto serio.

c. Probabilidad baja. Efecto catastrófico.

Plan de prevención y minimización.

a. Hacer una campaña de selección donde se difundan muchos los puestos ofertados dentro de personal especializado. Prever cursos de formación.

b. Utilizar una estructura organizativa democrática descentralizada, donde todos hagamos todo tipo de tareas.

c. Preparar un informe para justificar la importancia de nuestro proyecto.

Seguimiento:

a. Revisar número de candidatos en otras convocatorias. Trabajar conjuntamente con las organizaciones educativas con el fin de conocer los perfiles.

b. Llevar un control de ausencias. Ver el ánimo del grupo. Hacer actos sociales con mi personal para conocer su situación.

c. Mantener contacto con directivos de la empresa, gestores de otros proyectos de nuestra empresa y mercado con el fin de ir conociendo el estado del sector.

Tema 4

La gente en el proceso:

- La gente es uno de los "bienes más preciados" de una organización.
- Las tareas de un gestor están esencialmente orientadas a la gente. A menos que haya algún entendimiento con la gente, la gestión será un fracaso.
- La ingeniería del software es fundamentalmente una actividad cognitiva. Las limitaciones del conocimiento limitan a su vez el proceso software.

Actividades de gestión:

- Solución de problemas
- Motivación
- Planificación
- Estimación
- Control
- Organización

Modelado del conocimiento:

- **Conocimiento semántico:** es el conocimiento de los conceptos tales como la operación de asignación, el paso de parámetros, etc. Se adquiere mediante experiencia y aprendizaje activo
- **Conocimiento sintáctico:** se refiere a los detalles de representación, por ejemplo un bucle en C. Se adquiere mediante memorización.
- El conocimiento semántico se almacena de forma estructurada, independientemente de la representación.

Motivación: Una tarea importante de un gestor es la de motivar a la gente que trabaja en un proyecto. La motivación es una tarea compleja. Se considera que hay diferentes tipos de motivación basadas en:

- Necesidades básicas (ej. comer, dormir, etc.)
- Necesidades personales (ej. respeto, autoestima)
- Necesidades sociales (ej. ser aceptado como parte de un grupo)

La motivación se consigue en la medida en que se puedan satisfacer las necesidades de un individuo. La motivación puede cambiar dependiendo de circunstancias externas al personal o eventos externos. La gente no solamente está motivada por factores personales, sino por los derivados de formar parte de un grupo y una cultura.

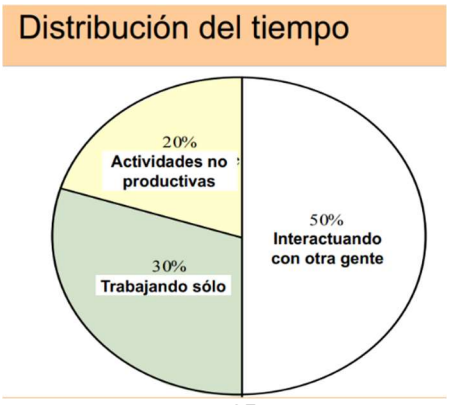
Pirámide o jerarquía de necesidades: Se trata de la simplificación de las necesidades que tenemos las personas. En la base de la pirámide están las necesidades fisiológicas básicas que debemos cubrir: comer, dormir, etc. Conforme vamos subiendo la pirámide podemos encontrar un nivel más alto en términos de necesidades:

Es importante como gestores de proyectos que sepamos motivar y retener al personal, por lo tanto, es de vital importancia tener en cuenta la pirámide. Un ejemplo de uso sería seleccionara personal cuyas necesidades podamos cubrir a corto plazo y formar equipos donde susnecesidades sean satisfechas (si todos quieren lo mismo podría llegar a ser imposible).

Tipos de personalidad / Composición del grupo: Tanto si trabajamos con grupos como si no, hay que distinguir tres tipos de personalidad. En este caso vamos a abordar los tipos de personalidades basándonos en composiciones de grupo:

- **Orientados a la tarea**: Cada uno quiere hacer las cosas según su propio criterio. La mayoría de los ingenieros se encontrarían en este grupo.
- **Orientados a sí mismos**: Cada uno quiere ser el jefe
- **Orientados a la interacción**: Demasiadas "charlas", no suficiente trabajo.

Un grupo efectivo tiene un equilibrio de todos los tipos de personalidad. Además es de mera importancia que todos los miembros del grupo se impliquen en las decisiones que afecten al grupo. Todo grupo ha de tener un **líder**, el cual se debe de basar en el **respeto**, no en un título que proporciona un "status". Debe haber un líder democrático y técnico.



Grupos cohesivos / cohesión del grupo: En un grupo cohesivo, los miembros consideran que el grupo es más importante que un individuo de este.

Ventajas de un grupo cohesivo:

- Se pueden desarrollar estándares de calidad para el grupo
- Los miembros del grupo trabajan juntos, por lo que se reducen fallos producidos por la ignorancia.
- Los miembros del grupo aprenden y enseñan unos con otros.
- Se puede practicar la "programación sin ego" donde los miembros se esfuerzan por mejorar el trabajo de los demás

Cómo se puede desarrollar la cohesividad en un grupo: La cohesividad está influenciada por factores tales como la cultura organizacional y las personalidades del grupo. Propiciar mediante:

- Eventos sociales
- El desarrollo de una identidad de grupo y área propia
- Actividades explícitas de construcción de grupos

La sinceridad con la información es una forma sencilla de asegurar que todos los miembros se sientan parte del grupo.

Comunicaciones del grupo: Una buena comunicación es esencial para el trabajo efectivo del grupo. Debe intercambiarse información sobre el estado del trabajo, las decisiones de diseño y los cambios en las decisiones previas. Una buena comunicación fortalece la cohesividad del grupo y promueve un mayor entendimiento. Factores que influyen:

- Status de sus miembros
- Personalidades de sus miembros
- Composición sexual del grupo
- Canales de comunicación

Organización del grupo: Tamaño del grupo relativamente pequeño (menos de ocho personas). Dividir los proyectos grandes en múltiples proyectos pequeños. Los grupos pequeños pueden organizarse de forma informal y democrática. El jefe de programadores intentará hacer un uso efectivo de las habilidades y experiencia del grupo

Organización democrática: El grupo actúa como un todo y las decisiones se toman por consenso. El líder del grupo no realiza asignaciones específicas de trabajo. El trabajo se discute por el grupo como un todo y las tareas se reparten de acuerdo según la habilidad y experiencia de cada uno. Esta aproximación tiene éxito en grupos cuyos miembros son todos competentes y con experiencia.

Grupos de Programación extrema: Variante de la organización democrática, en dichos grupos se toman algunas decisiones de gestión por parte de los miembros del grupo. Además, los programadores trabajan por parejas y adquieren una responsabilidad colectiva del código que han desarrollado.

Grupos con jefe de trabajo (Problemas):

- Los buenos diseñadores y programadores no se encuentran fácilmente.
- Los restantes miembros del grupo pueden verse afectados por el hecho de que el jefe de programadores asuma el éxito de todo el grupo, dificultando su trabajo.
- Debido a la estructura impuesta por la organización, puede no ser posible formar este tipo de grupo

Selección y organización de personal + factores de selección: Responsabilidad del gestor de proyectos. A veces se usan test psicológicos o de aptitud.

Entorno de trabajo: El entorno físico de trabajo juega un papel importante en la productividad y satisfacción individual (confort, privacidad, climatización, mobiliario…)

Modelo P-CMM de personal: Pretende ser un marco para la gestión del trabajo realizado por la gente implicada en el desarrollo del software. Es un modelo de cinco etapas:

- **Inicial**: Gestión de recursos humanos
- **Repetible**: Se desarrollan políticas para mejora de las capacidades (aptitudes)
- **Definido**: Gestión de recursos humanos estandarizada para la organización
- **Gestionado**: Se establecen metas cuantitativas para la gestión de recursos humanos
- **Optimizado**: Se realiza un esfuerzo continuado para mejorar la competencia y motivación en el trabajo.

Objetivos:

- Mejorar las capacidades de la organización mejorando las capacidades de trabajo de la gente implicada
- Asegurar que las capacidades para el desarrollo del software no conciernen a un número pequeño de individuos
- Igualar la motivación de los individuos con la de la organización
- Ayudar a la "retención" de gente con conocimientos y habilidades críticas

Puntos clave:

- Los gestores de software deben comprender algunos de los factores humanos para poder llevar a cabo su trabajo con éxito.
- Los factores principales por considerar son:
  - La organización de la memoria
  - La representación del conocimiento
  - La influencia de la motivación
- La composición y comunicación de los grupos de trabajo resulta fundamental para el éxito del proyecto.
- El modelo P-CMM proporciona un marco para mejorar las capacidades de los recursos humanos de una organización

Tema 5

Gestión de la Configuración: Actividad de autoprotección para:

- Identificar el cambio
- Controlar el cambio
- Garantizar que el cambio se implementa adecuadamente
- Informar del cambio a todos aquellos que les interese
- **Gestión de configuraciones ≠ Mantenimiento**

Planificación de la gestión de configuración:

- Definición de entidades a gestionar y esquema formal de identificación
- Identificación de responsables de los procedimientos de la GC
- Descripción de cómo los registros de documentos del proceso de GC deberían ser mantenidas
- Descripción de las herramientas usadas
- Definición de la base de datos de configuración (BDC) usada para almacenar la información

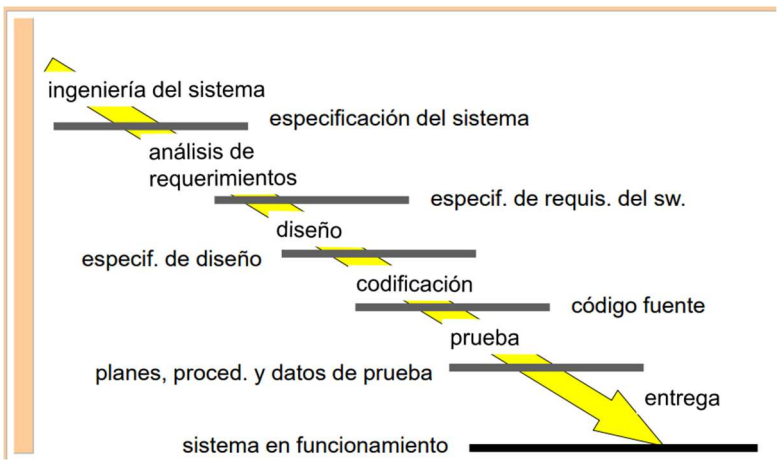
Identificación de elementos de configuración: Los elementos de configuración son aquellos documentos que se van a requerir para un futuro mantenimiento. No debería cambiar de forma arbitraria.

Base de datos de configuraciones: Usada para registrar cualquier información relacionada con las configuraciones, sirve de apoyo a la evaluación del impacto del cambio. Debe contestar a preguntas como:

- ¿A qué clientes se les ha entregado una versión particular del sistema?
- ¿Qué hardware y SO son necesarios para ejecutar una determinada versión?
- ¿Cuántas versiones del sistema se han creado y cuáles son sus fechas de creación?
- ¿Qué versiones de un sistema podrían verse afectadas si un componente particular es cambiado?
- ¿Cuántas peticiones de cambio se han hecho sobre una determinada versión?
- ¿Cuántos fallos se han registrado para una versión particular?

La BD de configuraciones puede ser un sistema separado o estar incluido en el sistema de control de versiones del proyecto que almacena documentos formales.

Líneas Base:



Control de versiones: Implica la identificación y seguimiento de diferentes versiones y "releases" del sistema. Diseño de procedimientos para asegurar que diferentes versiones del sistema puedan recuperarse cuando se requieran, y evitar que no sean cambiadas accidentalmente. Hay un contacto con el cliente para planificar cuándo deberían distribuirse nuevas releases.

- **Versión:** Instancia de un sistema que difiere de otras instancias.
- **Release:** Versión del sistema distribuida a los clientes. Debe de incluir la instalación del programa, documentación electrónica y en papel sobre el sistema y ficheros de configuración y de datos.

El grupo de GC debe decidir cuándo los componentes afectados por un cambio deben ser reconstruidos en una nueva versión o una nueva release. Decisión forzada a veces por fallos descubiertos por el cliente parcheo del código objeto (mejor solución: versión nueva sin documentación)

Herramientas de gestión de versiones:

- **Unix: SCCS, RCS (**Diseñados para trabajar con texto ASCII**)**
  - Identificación de versiones y releases
  - Cambios controlados
  - Gestión de almacenamiento
  - Registro de la historia de cambios
- **RCS:**
  - Graba el código fuente de la versión más reciente
  - Soporta el desarrollo paralelo de diferentes releases
  - Capacidad de "mezcla" de versiones

Auditoría de la Configuración: ¿Cómo podemos asegurar que el cambio se ha implementado correctamente?

- Revisiones técnicas formales «corrección técnica»
- Auditorías de configuración: ¿se ha hecho el cambio especificado? ¿se han seguido los estándares de IS? ¿se han seguido procedimientos para señalar el cambio, registrarlo y divulgarlo? ¿se han actualizado adecuadamente todos los ECS relacionados?

Construcción del sistema: Proceso de combinar componentes en un programa que se ejecuta sobre una configuración destino particular. Dicha construcción implica una **compilación** y un **linkado**. Hay que llevar especial cuidado con sistemas desarrollados con un sistema distinto del de destino. El proceso se refiere normalmente a componentes físicos. Una herramienta ejemplo de construcción de sistemas UNIX es **MAKE.** Mantiene una correspondencia entre el código fuente y las versiones de código objeto de un sistema 4 El usuario especifica las dependencias de los componentes y MAKE fuerza automáticamente la recompilación de los ficheros cuyo código fuente haya cambiado después de que el código objeto fuese creado**. El cual tiene varias limitaciones:**

- Está basado en un modelo físico de dependencias
- Los makefiles crecen rápidamente, llegando a ser bastante difíciles de comprender
- Usa un modelo de cambio basado en timestamps, puede haber cambios que no requieran una recompilación.
- No permite fácilmente especificar la versión de las herramientas.
- No enlaza finamente con herramientas de gestión de configs. como RCS

Beneficios GCS (Gestión de Configuraciones del Software):

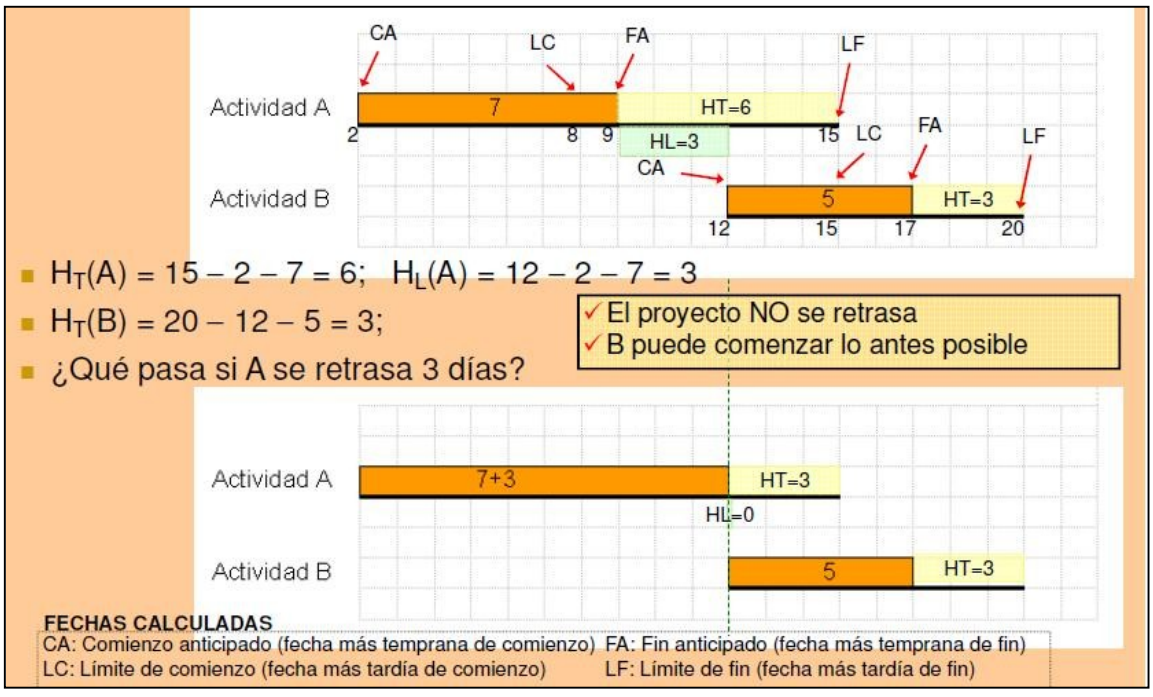
- Reduce el esfuerzo necesario para gestionar y realizar el cambio = mejora la productividad
- Conduce a una mejora de la integridad y seguridad del software = incremento de la calidad
- Genera información sobre el proceso = mejora de la gestión del control
- Mantiene una base de datos de desarrollo de software = mejor registro y seguimiento de informes

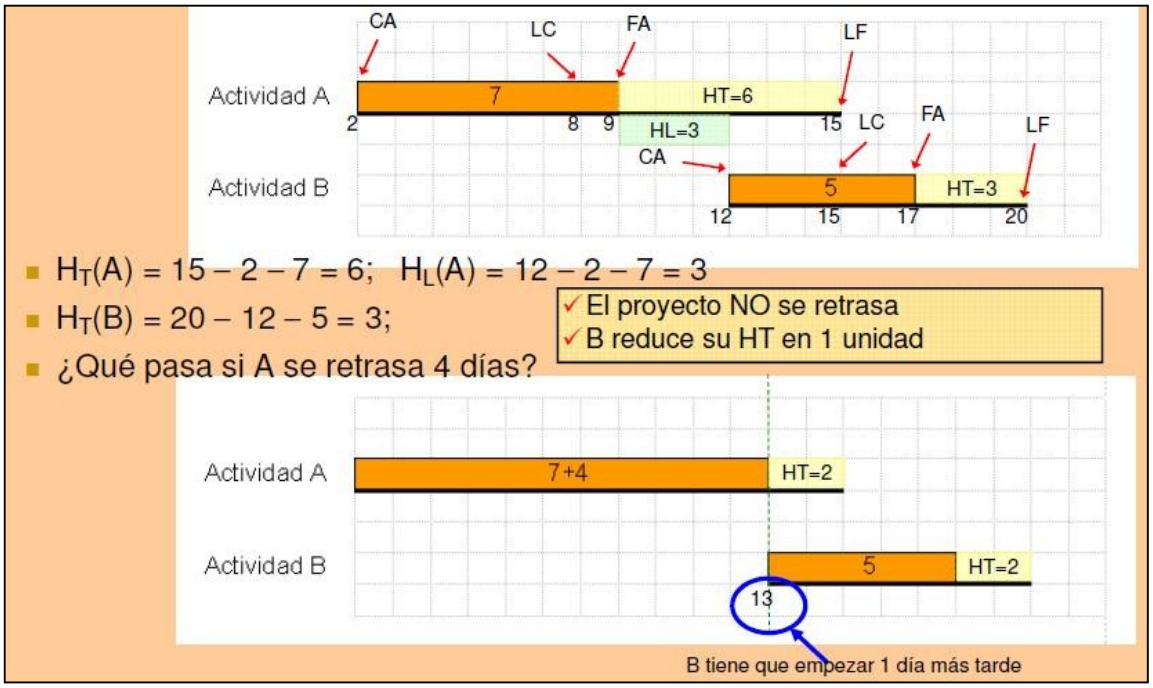
Tema 6

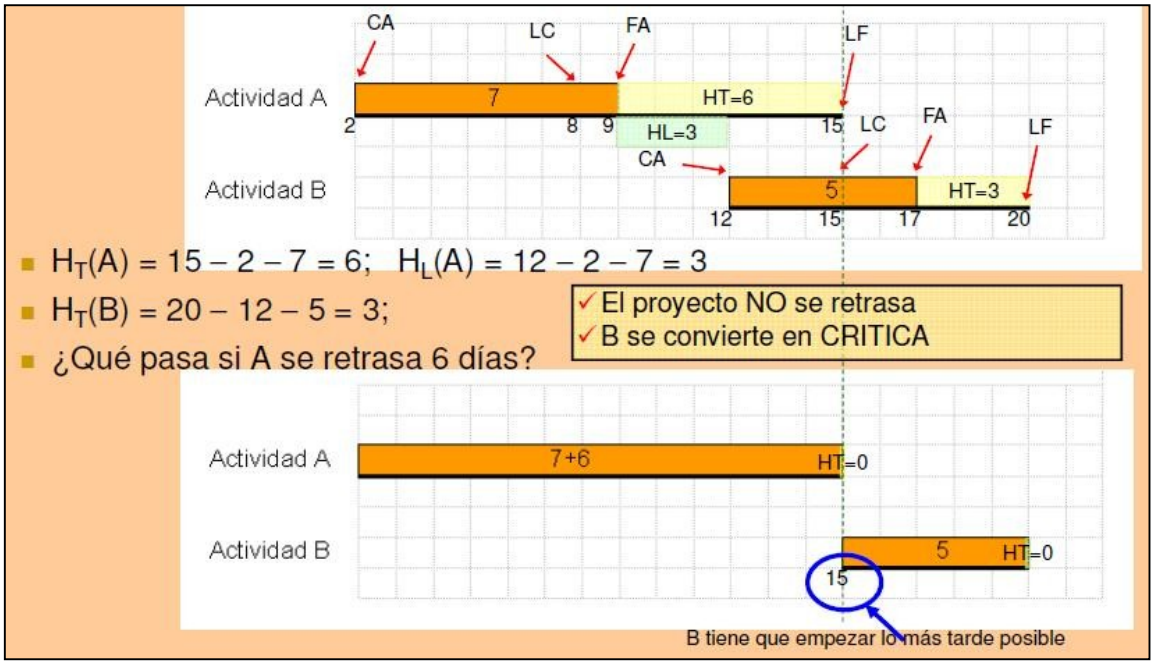
Monitorizar: Consiste en comprobar si la agenda real se ajusta a la planificada. A la agenda creada inicialmente la denominamos **agenda planificada** (se crea a partir de información planificada). A medida que el proyecto progresa se creará una **agenda real** (a partir de información real). Es imprescindible hacer uso de diversas métricas (Una métrica es cualquier tipo de medición que proporciona un valor cuantitativo para indicar el grado en el que un sistema, componente o proceso posee un determinado atributo) si queremos monitorizar la agenda del proyecto. A continuación, se muestras las métricas y cálculos que emplearemos para hacer el seguimiento del proyecto:

- **Fechas de inicio y fin:** Fechas más tempranas y tardías de las actividades
- **Holguras totales y libres:**
  - **La holgura total** es el tiempo que una actividad puede retrasarse sin retrasar el proyecto.
  - **La holgura libre** es el tiempo que una actividad puede retrasarse sin afectar a las siguientes ni a la finalización del proyecto.
- **Camino crítico:** Secuencia de actividades con holgura total a 0
- **Cálculos de EV**

Holguras:







La vista de Gantt detallado muestra gráficamente las holguras libres.

Control de una agenda: holguras:

- Las actividades críticas no se pueden retrasar
- Las actividades críticas pueden retrasarse siempre y cuando el retraso no supere su holgura total
    - Si una actividad consume toda su holgura total (o parte de ella), puede afectar a la holgura total de las actividades siguientes
    - Las actividades siguientes pueden convertirse en críticas
    - Ejemplo: A precede a B
        - HT(A) = 3 y HL(A) = 2

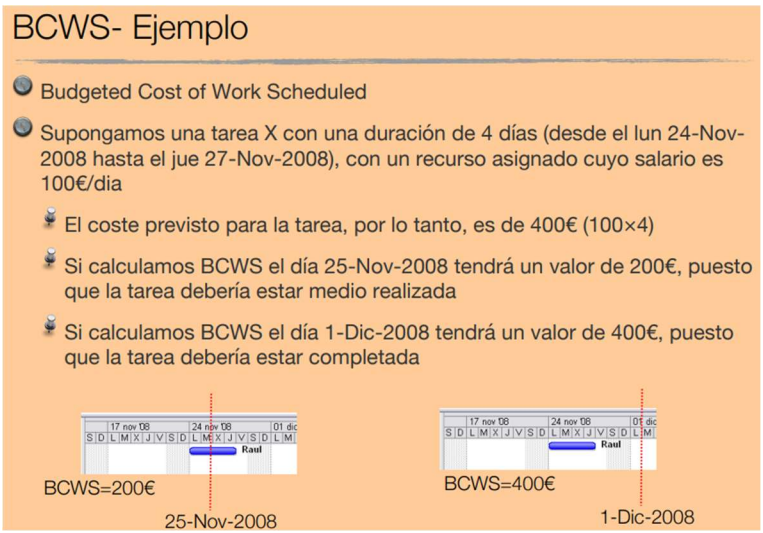
- HT(B) = 1
- Si A se retrasa 3 unidades, B se convierte en crítica

Earned Value Analysis (EVA)

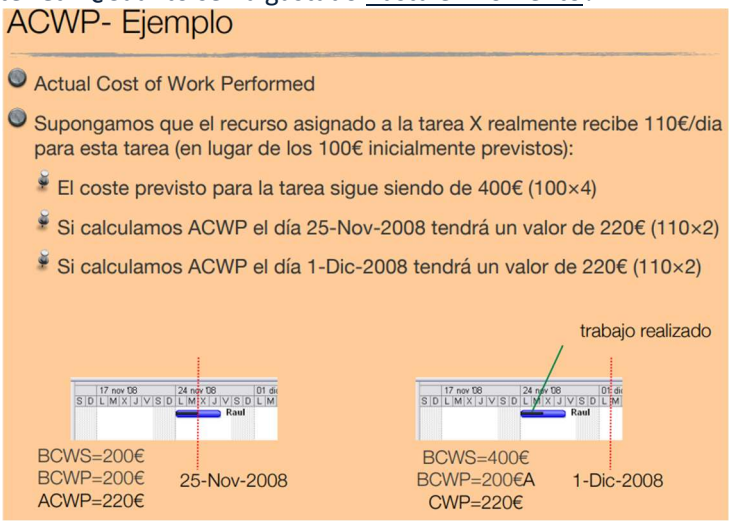
EV: El earned value o valor acumulado es una métrica que proporciona una información cuantitativa del progreso de un proyecto (A cada tarea se le asigna un valor devengado basado en su porcentaje estimado del valor total). Permite vislumbrar dificultades en la agenda antes de que éstas puedan ser aparentes (esto permite al gestor del proyecto tomar acciones correctivas antes de que el proyecto "entre en crisis").

EVA compara tres tipos de información:

- **BCWS** = Valor **planificado**: ¿Cuánto trabajo (del que se ha planificado) debería haberse completado hasta el momento?



- **ACWP** = Coste **real**: ¿Cuánto se ha gastado hasta el momento?



- **BCWP** = Valor **acumulado**: ¿Cuál es el valor, en términos del coste de línea base, del trabajo realizado hasta el momento?

BCWP- Ejemplo

- Budgeted Cost of Work Performed
- Supongamos que la tarea X anterior solamente se ha "completado" hasta la mitad (se ha hecho el 50% de lo que se había pedido):
  - El coste previsto para la tarea sigue siendo de 400€ (100×4)
  - Si calculamos BCWP el día 25-Nov-2008 tendrá un valor de 200€: el 100% del valor planificado
  - Si calculamos BCWP el día 1-Dic-2008 tendrá un valor de 200€: el 50% del valor planificado

- SI BCWP > BCWS: La tarea/proyecto va ADELANTADA según la agenda
- SI BCWP < BCWS: La tarea/proyecto va con RETRASO según la agenda

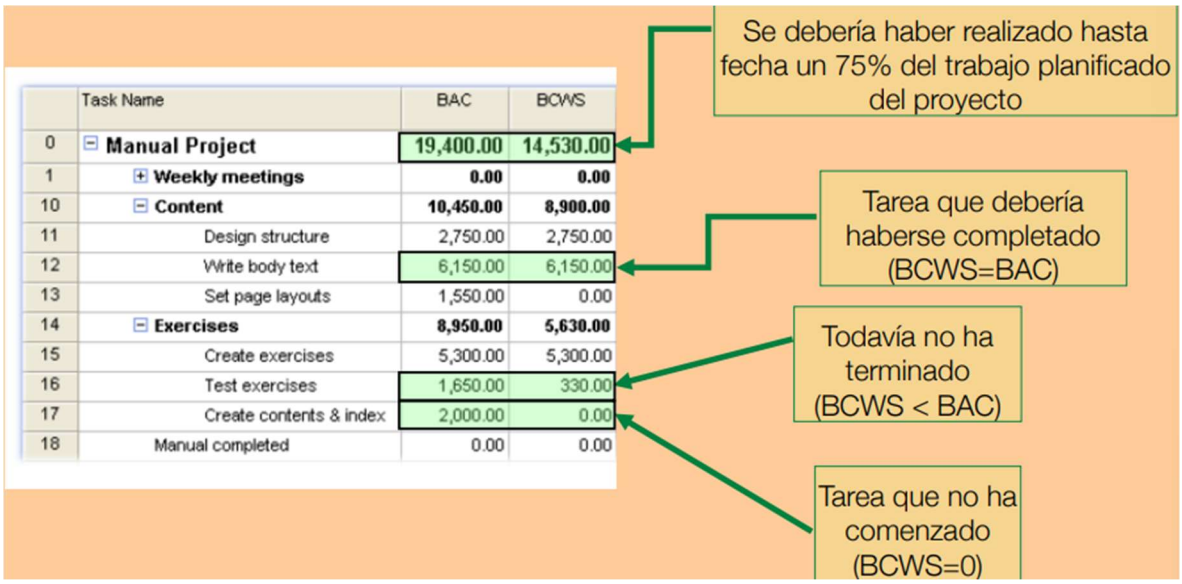
- SI BCWP > ACWS: La tarea/proyecto está por DEBAJO de lo presupuestado
- SI BCWP < ACWS: La tarea/proyecto está por ENCIMA de lo presupuestado

Un análisis EVA (análisis de valor acumulado) SIEMPRE se hace tomando como referencia un instante de tiempo concreto del desarrollo del proyecto.

¿Cómo funciona EVA?

- Indicadores de **PROGRESO**:
  - Schedule Variance (SV) = BCWP - BCWS
  - Schedule Performance Index (SPI) = BCWP / BCWS
  - Si BCWP > BCWS la tarea/proyecto va adelantada según la agenda planificada
  - P.ej. un SPI de 1,5 significa que sólo ha utilizado el 67% del tiempo planeado para completar una parte de la tarea en un determinado periodo de tiempo (BCWS = 0,67 BCWP)
- Indicadores de **PRODUCTIVIDAD**
  - Cost Variance (CV) = BCWP - ACWP
  - Cost Performance Index (CPI) = BCWP / ACWP
  - Si BCWP > ACWP la tarea/proyecto está gastando menos de lo planificado
  - P.ej. un CPI de 0,8 significa que se está gastando un 25% más de lo que estaba planificado (por cada euro presupuestado se está gastando 1,25€) (ACWP =1.25 BCWP)
- Si tenemos una buena productividad, y un progreso lento: NOS FALTA GENTE!!!

BAC: Cantidad de trabajo planificado al final del proyecto/tarea.



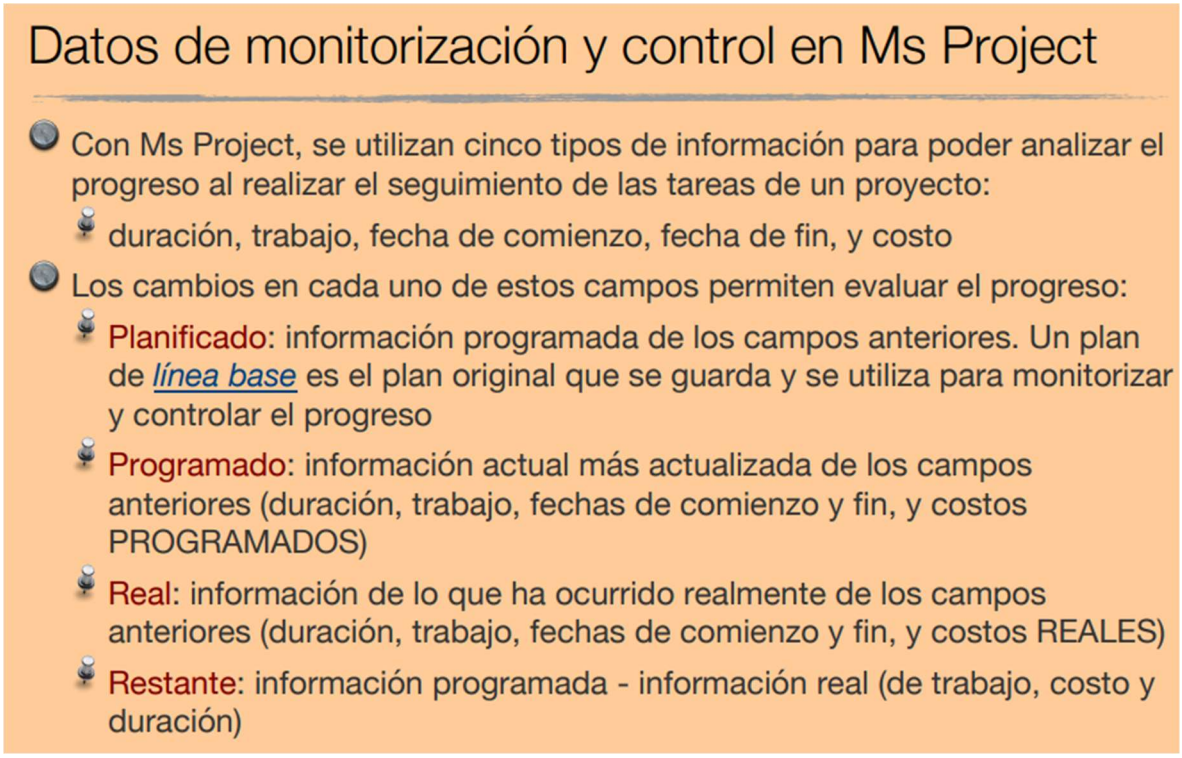
	Task Name	BAC	BCWS
0	**Manual Project**	**19,400.00**	**14,530.00**
1	**Weekly meetings**	**0.00**	**0.00**
10	**Content**	**10,450.00**	**8,900.00**
11	Design structure	2,750.00	2,750.00
12	Write body text	6,150.00	6,150.00
13	Set page layouts	1,550.00	0.00
14	**Exercises**	**8,950.00**	**5,630.00**
15	Create exercises	5,300.00	5,300.00
16	Test exercises	1,650.00	330.00
17	Create contents & index	2,000.00	0.00
18	Manual completed	0.00	0.00

Control de una agenda: EVA:

- Los ratios:
  - SPI es un indicador de progreso; CPI es un indicador de productividad
  - Si CPI >1 y SPI <1 necesitamos contratar a más gente
  - Si CPI <1 puede que estemos haciendo trabajo no planificado, o que hayamos estimado mal

Ms Project y seguimiento agenda

- Para hacer un seguimiento tenemos que:
  - Guardar una **LÍNEA BASE** del proyecto
    - Proyecto→Herramientas→Establecer línea de base
    - Los campos: duración, trabajo, comienzo, fin, costo se guardan como duración, trabajo,..., costo PREVISTOS
  - Establecer una **FECHA DE ESTADO**
    - Proyecto→Información del proyecto →Fecha de estado
  - Introducir la **INFORMACIÓN REAL** del proyecto
    - Herramientas→Seguimiento→Actualizar tareas
    - Campos: duración, trabajo, comienzo, fin, costo REALES
  - Comparar el **PROGRESO** con una vista "Gantt de Seguimiento"
    - Compara la programación de la línea base con la programación real



Conceptos que no salen en las diapositivas, pero entran en el examen

Fuente: Exámenes

Diferencias entre un presupuesto y un documento de estimación de costes: El documento de estimación de costes es un documento interno de la empresa que indica el esfuerzo necesario para realizar un proyecto (personas/mes), el coste temporal y económico del proyecto.

El presupuesto es un documento que se va a entregar al cliente para informarle sobre los precios de venta, tiempos de entrega y funcionalidades de la aplicación.

La **diferencia** fundamental es que la estimación habla de costes, mientras que el presupuesto habla en términos de precios de venta y fechas de entrega. El **documento de estimación** de costes debe tener estimaciones mediante distintas técnicas (Pricing to Win, Parkinson...)

El **presupuesto** debe tener los datos de proyecto, empresa y cliente, una descripción del producto a desarrollar, requisitos mínimos de SW y HW que necesita el programa, disposiciones legales, plazos de pago...

Visibilidad de un proyecto: Consiste en conocer exactamente qué es lo que se está haciendo en cada momento del desarrollo. La única forma de hacer que el proceso de desarrollo muestre el estado del producto es mediante la **documentación generada** en el proceso. La posibilidad de ver en qué estado se encuentra un producto software durante su desarrollo es de vital importancia ya que, gracias a esto, podremos hacer una planificación efectiva que nos permitirá **monitorizar y controlar** el desarrollo del mismo.

Ejemplos que dan visibilidad a un proyecto: Gantt, hacer selección de un plan adaptativo en vez de predictivo (hitos), informe de iteración (documentación), uso de diferentes niveles en el plan.

Cosas que hemos hecho en prácticas que sirven para estudiar prácticas

Fuente: Exámenes y prácticas

Qué hemos hecho éste año en prácticas (2017) (por orden):

- Establecer objetivos generales y funcionalidades iniciales
- Estimar costes del proyecto (diversas estrategias)
- Crear presupuesto del proyecto
- MS Project: Crear plantilla modelo UP
- MS Project: Crear Plan General del proyecto, crear plan detallado de las 2 primeras iteraciones
- MS Project: Definir recursos en el proyecto, definir estructura organizativa, asignar recursos a las tareas de las 2 primeras iteraciones.
- Definir riesgos y establecer plan de prevención de riesgos y minimización de impacto. Establecer mecanismos de seguimiento y control.
- MS Project: Ejercicios EVA (individual [BCWS, BCWP, ACWP...])
- ***~ PRESENTACIÓN DEL PLAN ~***
- Dar de alta datos de una empresa en la agencia de protección de datos y rellenar un formulario NOTA.
- Registrar un programa informático, definir competencias de un perfil profesional y crear el procedimiento para una tarea.

Documento NOTA: Formulario empleado para registrar ficheros en la AEPD. Obviamente dichos ficheros han de requerir mediante la LOPD ser registrados (datos sensibles). Podemos seleccionar la titularidad: pública o privada. También dispone de una opción exclusiva para empresas con certificado de veracidad. El tipo de operaciones que podremos realizar con estos documentos serán: dar de **alta**, dar de **baja** y **modificar**.

Códigos tipo (o formularios tipo): Trátense de documentos NOTA prerellenados y simplificados dependiendo el tipo de documento estándar que queramos dar de alta. Ejemplos: Nóminas, recursos humanos, gestión escolar, pacientes, comunidad de propietarios...

Registro de software: Hay dos alternativas principales, registrar la patente del software o registrar la propiedad intelectual. El más común es a través del registro de propiedad intelectual. Se hace a través del ministerio de Educación, cultura y Deporte. Hay que rellenar los impresos de autores o titulares, y el modelo de programas de ordenador.

En cuanto a los autores: Diferenciamos tres tipos principales, Autores 1 A-T, Autores 1 TIV y Autores 1 TMC:

- **Autores 1 A-T:** a cumplimentar por los titulares de derechos de propiedad intelectual que sean autores y titulares de una obra.
- **Autores 1 TIV:** a cumplimentar por los titulares que hubieran adquirido los derechos por transmisión inter vivos, ya sea por contrato de cesión o por relación laboral, y siempre que se trate de una primera inscripción de derechos.
- **Autores 1 TMC:** a cumplimentar por los herederos que hubieran adquirido los derechos por transmisión mortis causa del autor, siempre que se trate de una primera inscripción de derechos.

Ejemplos: Un profesional autónomo crea un programa software y para registrarlo utiliza el impreso A-T, mientras que una empresa de desarrollo con empleados utilizaría el formulario TIV.

Resumen EVA

```
Tema 6  EVA

    HL  Holgura Libre:  cuánto puedo retrasar una actividad sin afectar
                        a las siguientes ni al proyecto
    HT  Holgura Total   Cuanto se puede retrasar una actividad sin
                        afectar a la finalizacion del proyecto

    BCWS:   Cuanto trabajo deberia haberse competado hasta el momento
    ACWP:   Cuanto se ha gastado hasta el momento
    BCWP:   Cual es el valor del trabajo realizado hasta el momento
    BAC:    Cantidad de trabajo planificado al final del proyecto

    Progreso
        SV: Scheduled Variance                  BCWP - BCWS
        SPI: Scheduled Performance Index        BCWP / BCWS
        if BCWP > BCWS      proyecto adelantado segun ajenda

    Productividad:
        CV: Cost Variance                       BCWP - ACWP
        CPI: Cost Performance Index             BCWP / ACWP
        if BCWP > ACWP      proyecto gasta menos de lo planeado

    BCWS = BAC  Tarea deberia haber terminado
    BCWS < BAC  Tarea no Terminada
    BCWS = 0    Tarea no empezada
    BCWS = BAC != BCWP          Deberíamos haber terminado pero no lo hemos hecho
    BCWS = BCWP = BAC           Hemos terminado.
    BCWS < BAC && BAC = BCWP    Hemos terminado y antes de lo esperado.

    BCWP > BCWS Tarea Adelantada
    BCWP < BCWS Tarea atrasada
    BCWP > ACWP El proyecto esta por DEBAJO de lo presupuestado
                GASTA MENOS de lo planeado
    BCWP < ACWP El proyecto esta por ENCIMA de lo presupuestado
                GASTA MAS de lo planeado

    Si CPI > 1 && SPI < 1 necesitamos contratar a más gente
    Si CPI < 1 puede que estemos haciendo trabajo no planificado, o que hayamos estimado mal
```