En la actualidad, debido al constante incremento de la potencia de los computadores, los proyectos que se realizan **van aumentando en complejidad y tamaño de los proyectos**.

Este hecho, unido a los requisitos temporales del proyecto, ha dado lugar a que el **éxito o el fracaso** del proyecto **dependan, en gran medida, de la capacidad de coordinación del grupo de trabajo**. Para ello se necesita un proceso que unifique todas las áreas del desarrollo.

Factores que influyen en el éxito

- **Comprensión del problema** y establecimiento de **objetivos realistas**.
- **Motivar** al personal mediante incentivos.
- **Seguir** el **progreso** del proyecto.
- Establecer un **margen de tiempo** para las tareas. A ser posible, Utilizar herramientas de software e interfaces existentes.
- Realizar un **análisis postproyecto** para extraer la información útil sobre el desarrollo de este.

Instituto de Ingeniería del Software crea "**Modelo de madurez de la capacidad de gestión del personal**": Ayudando a atraer, aumentar, motivar, desplegar y retener el talento. Define las siguientes **áreas clave**:

- Reclutamiento,
- Selección,
- Gestión de rendimiento,
- Entrenamiento,
- Retribución,
- Desarrollo de la carrera
- Diseño de la organización y del trabajo
- Desarrollo cultural y espíritu de equipo.

Características de un jefe de proyecto:

- Habilidad para **motivar**.
- Habilidad para moldear procesos (**resol. problemas**).
- Habilidad para **incentivar la creatividad**.
- **Dotes de gestión**.
- Saber **incentivar los logros** (incremento productividad).
- **Capacidad para crear** un **equipo** que presente cohesión.

Actividades de la Gestión

- Calidad del producto
- Evaluación del riesgo
- Métricas
- Estimación de costes
- Confección de agendas
- Comunicación cliente
- Personal
- Otros recursos
- Monitorización del proyecto

Factores y **Estructuras de equipo** de software:

	Descentralizado democrático (DD)	Descentralizado controlado (DC)	Centralizado controlado (CC)
Dificultad	Alta	Baja	Baja
Tamaño	Pequeño	Grande	Grande
Vida equipo	Largo	Corto	Corto
Modularidad	Baja	Alta	Alta
Fiabilidad	Alta	Alta	Baja
Rigidez entrega	Flexible	Flexible	Estricta
Comunicación	Alta	Baja	Baja

Las **actividades estructurales** (**comunes** a todos los proyectos) de la maduración de un proceso son:

- **Comunicación** con el cliente. Establecimiento de la comunicación **desarrollador cliente**.
- **Planificación**. **Definición** de **recursos y tiempo**.
- **Análisis de riesgos**. Evaluación de riesgos **técnicos y de gestión**.
- **Ingeniería**. Construcción de las **representaciones de la aplicación**.
- **Construcción y acción**. **Construcción, prueba, instalación**. Creación del **material de soporte al usuario**.
- **Evaluación del cliente**. Reunión final con el cliente.

Para planificar un proyecto se requieren **estimaciones cuantitativas.** Pasos:

- **Ámbito del software**
    - **Contexto**. Establece las **limitaciones** en función de este.
    - **Objetivos de información**. **Datos visibles al cliente** y **datos de entrada**.
    - **Función y rendimiento**. **Operaciones** para convertir los **datos de entrada en la salida deseada**. Características especiales de **rendimiento**.
- **Descomposición del problema**:
    - **Funcionalidad**
    - **Proceso a utilizar**. Un proceso son el conjunto de **actividades necesarias para convertir** los **requisitos** de un cliente **en** un producto **software**. Un proceso **debe**:
        - Proporcionar una **guía para ordenar las actividades**.
        - **Dirigir las tareas**, ya sea de un desarrollador o de todo el equipo.
        - **Especificar** los **modelos y diseños** que han de desarrollarse
        - **Establecer criterios sobre** los que apoyarse para llevar **el control** sobre el avance del proyecto
            - **Modelos de proceso secuenciales.** El **principal problema** de este modelo es que **cada fase depende de la anterior**, por lo que hasta que no se complete no se podrá avanzar.
                - **Modelo de Ciclo de vida en cascada**.
                    - Fases:
                        - **Análisis de requisitos** del software.
                        - **Diseño**. Se centra en:
                            - **Estructura de datos**
                            - **Arquitectura** del software
                            - Representación de la **interfaz**
                            - **Algoritmo**
                        - **Generación del código**.
                        - **Pruebas**.
                            - **Detección de errores**
                            - Comprobación de resultados.
                        - **Mantenimiento**.
                            - **Corrección de errores**.
                            - **Modificaciones**.
                            - **Mejoras de rendimiento**.
                    - Desventajas:
                        - **Acentúa el fracaso** de la industria software **frente al usuario final**.
                        - Se tarda **mucho tiempo en** pasar por todo el **ciclo**
                        - **No refleja el proceso real de desarrollo** software.
                        - Los proyectos reales **raramente siguen este flujo** secuencial.
                - **Construcción de prototipos**. Este modelo de proceso se utiliza, principalmente, en los proyectos en los que **el desarrollador no está seguro del correcto funcionamiento de un algoritmo** o en los que el **cliente no deja claros los requisitos**
                    - creación de un **prototipo a partir de un diseño rápido**. **realizado entre el cliente y el desarrollador**.

        - construye un **prototipo** que **se muestra al cliente** y sobre el que se trabaja
        - Cuando se obtiene el producto deseado acaba la iteración.
        - Desventajas:
            - **No** se tiene en cuenta la **calidad del software** ni la **facilidad de mantenimiento**.
            - En caso de ser aceptado el prototipo habrá que **volver a construir el proyecto desde el principio** para mantener unos niveles de calidad.
            - La necesidad de que el programa funcione lo antes posible puede dar lugar a la **elección del lenguaje erróneo o de un algoritmo ineficiente**.
    - **Desarrollo rápido de aplicaciones (DRA)**
        - Fases:
            - **Modelado de gestión**
            - **Modelado de datos**. Definición de **relaciones y atributos de objetos**.
            - **Modelado del proceso**. **Operaciones de los objetos** creados en la fase anterior.
            - **Generación de aplicaciones**. Generación mediante componentes ya existentes y de componentes reutilizables.
            - **Pruebas y entrega**. Prueba de los nuevos componentes.
        - Desventajas:
            - Para **proyectos grandes** requiere una **elevada cantidad de recursos humanos**.
            - Requiere **clientes y desarrolladores comprometidos**.
            - Será **problemático** cuando el proyecto **no se pueda modularizar** de forma adecuada **o se conviertan las interfaces en componentes del sistema.**
            - Es **inadecuado** para proyectos en los que los **riesgos técnicos son elevados**; ya sea por el uso de nuevas tecnologías o porque los componentes deben interactuar con elementos ya instalados en el sistema.
- **Modelos de proceso evolutivos**
    - **Modelo de desarrollo incremental**.
        - Basado en **Incrementos**: **Entregas planificadas** que son para **cubrir una serie de funcionalidades**.
        - Se hacen pequeñas partes que se hacen prueban integran prueban...
        - Se van trabajando con parcelas mas pequeñas.
    - **Modelo en espiral**. Vamos haciendo **diferentes ciclos con diferentes actividades** y al final conseguimos que el software haga mas y mas cosas hasta al final entregar el software
        - Divide su tarea en **6 regiones**:
            - **Comunicación con el cliente**
            - **Planificación**
            - **Análisis de riesgos**
            - **Ingeniería**
            - **Construcción y acción**
            - **Evaluación del cliente**
    - **El Proceso Unificado(UP)**. El proceso unificado es un tipo de proceso que se construye **a partir de componentes interconectados mediante interfaces**. **Utiliza** el Lenguaje Unificado de Modelado (**UML**) para preparar los esquemas.
        - **3 características**:
            - **Dirigido por casos de uso**
                - **define el comportamiento del sistema en**

función del usuario que accede a este y a las acciones que realiza.
    - **representan los requisitos funcionales** del proyecto.
    - el modelo de casos de uso **define la funcionalidad total** del sistema.
    - **Es punto de partida del proyecto**. En revisiones posteriores será el patrón a seguir.
- **Centrado en la arquitectura**
    - **engloba los aspectos estáticos y dinámicos del sistema**. Estos se reflejan en los casos de uso
    - se ve **influida por** factores como **la plataforma** de lanzamiento **o** los **bloques de código reutilizables**.
    - Los **casos de uso y la arquitectura se desarrollan simultáneamente**. Esto se debe a que **una depende de la otra**.
    - La función corresponde a los casos de uso y el modo en que se conectan y la estructura se corresponde con la arquitectura.
    - Un proyecto que carezca de cualquiera que estas componentes no podrá llevarse a cabo.
    - El **arquitecto** se encarga de:
        - Crear un **esquema inicial independiente de los casos de uso**.
        - Especificar un **subconjunto de casos de uso en detalle** en lo referente a subsistemas, clases y componentes
        - **Actualizar la arquitectura en función de los casos de uso**. Este proceso durará mientras la arquitectura no se considere estable.
- **Iterativo e incremental**
    - **Adaptando los diseños** y casos de uso al estado del proyecto en cada momento.
    - **En cada iteración se desarrollaran un grupo de casos de uso**
    - **El proyecto se divide en fases y estas, a su vez en iteraciones**.
    - Las fases son:
        - **Fase de inicio**
            - **Descripción del producto** final **y análisis de negocio**.
            - Definición de principales funciones (**modelo casos de uso simplificado**).
            - **Arquitectura provisional**
            - Principales **riesgos**
            - **Planificación fase elaboración**
            - **Estimación de costes inicia**l.
        - **Fase de Elaboración**
            - **Especificación detallada de los casos de uso**.
            - **Diseño básico de la arquitectura** del sistema.
            - **Casos de uso críticos** de la fase anterior.
            - **Planificación de actividades**

            - **Estimación de costes**.
        - **Fase de construcción**
            - **Desarrollo de la arquitectura**.
            - **Creación del producto** (con defectos).
            - Presentación al cliente (**Demo**).
        - **Fase de transición**
            - **Beta del producto**.
            - Línea de **asistencia**.
            - **Introducción de mejoras**.
            - **Corrección de errores**.
    - Las **ventajas** de utilizar el modelo iterativo controlado son:
        - Se **reduce el coste del riesgo** a los costes de una sola iteración.
        - Se **reduce el riesgo de no cumplir los plazos** previstos.
        - **Los objetivos a corto plazo mejoran la eficiencia** de los desarrolladores.
        - Permite **modificar los requisitos en función de las necesidades** del usuario.
    - El **producto final deberá contener**:
        - El **modelo de casos de uso**
        - Un **modelo de análisis** que contenga una **explicación detallada de los casos de uso**, así como la **inicialización de los objetos responsables del comportamiento del sistema**.
        - Un **modelo de implementación** que incluya los **componentes y la correspondencia de las clases con estos**.
        - Un **modelo de despliegue** que defina los **nodos físicos y la correspondencia de estos con los componentes**.
        - Un **modelo de prueba** para **verificar los casos de uso**.
        - Una **representación de la arquitectura**.

La **productividad** de un programador es una **medida de** la **"velocidad"** a la que los ingenieros implicados en el desarrollo del software **producen** dicho **software y su documentación asociada**. **Usos:**

- Realización de las **estimaciones.**
- **Evaluación** de efectividad de procesos o mejoras tecnológicas.

La **duración** del proyecto **depende sólo del esfuerzo** que se aplica al proyecto **independientemente del número de gente que participa** de este.

Una **estimación de costes** es una **predicción de los recursos necesarios** para la realización de un proyecto.

Factores de coste:

- **Oportunidad de negocio**. Margen de beneficio bajo a cambio de **asegurarse el contrato** (ganar reconocimiento o experiencia)
- **Estimación de costes**. Puede que el **coste del proyecto se vea modificado** tanto al alza como a la baja sobre el coste esperado.
- **Términos del contrato**. Puede que el cliente esté dispuesto a permitir que el **desarrollador** siga siendo el **propietario del código**
- **Volatilidad de los requerimientos**. Si los **requerimientos se pueden modificar** es probable que la empresa pueda reducir el precio a cambio de aceptar el proyecto. Una vez aceptado el contrato el precio se incrementará de forma notable.
- **Salud financiera**. En caso de tener problemas financieros la empresa deberá **reducir sus precios, aun a costa de reducir el margen de beneficios**.

Técnicas de estimación.

- Características generales:
  - **simples y fáciles** de calcular.
  - **empírica** e intuitivamente persuasivas.
  - **consistentes y objetivas**.
  - consistentes en el empleo de **unidades y tamaños**.
  - **independientes del lenguaje de programación**.
  - un eficaz mecanismo para la **realimentación de calidad**.
- **Puntos función**.
  - **Basados** en una combinación de características del programa:
    - **Entradas y salidas** externas.
    - **Interacciones** de usuario.
    - **Interfaces** externas.
    - **Ficheros** usados por el sistema.
  - Cada uno de estos factores se le asignara un peso.
  - Ventajas:
    - Pueden **calcularse a partir de la especificación (Fase Elaboración)**.
    - Usa información del dominio del problema.
    - Resulta **más fácil** a la hora de **reusar componentes**.
    - Se encamina a **aproximaciones orientadas a objetos**.
- **Puntos objeto**
  - Relacionada con la **funcionalidad cuando se utilizan lenguajes 4GL o similares**.
  - En **base a**:
    - Numero de **pantallas visualizadas** por separado
    - Numero de **informes**
    - Numero de **módulos 3GL**.
  - Ventajas:
    - Permite una **estimación de coste temprana**.
- **Juicio experto**
  - Un **conjunto de expertos**, basándose en su **experiencia**, estiman los costes.
  - Ventajas:
    - **Barato**
    - **Exacto con los expertos adecuados**.
  - Inconvenientes:
    - Muy inexacto si los expertos no son los adecuados.
- **Ley de Parkinson**
  - **En función de los recursos** disponibles.
  - Ventajas:
    - **No genera presupuestos abultados**.
  - Inconvenientes:
    - **No** suele **llegar a buen término**.
- **Estimación por analogía**
  - **Comparación con proyectos similares** en el mismo dominio de aplicación.
  - Ventajas:
    - **Preciso si se disponen de proyectos previos**
  - Inconvenientes:
    - Se debe haber abordado un problema de envergadura similar.
    - Requiere el **mantenimiento de una base de datos**.
- **Pricing to win**
  - Depende de **lo que el cliente esté dispuesto a pagar**.
  - Ventajas:
    - Obtención del **contrato segura**.
  - Inconvenientes:
    - El sistema **no se podrá construir por el coste** que nos ofrece el cliente.
- **COCOMO 2**
  - Características:
    - Método empírico **basado en proyectos anteriores**.

        - **Bien documentado**.
    - Modelo de **3 niveles**:
        - **Nivel de prototipado**:
            - **Estimación mediante puntos objeto**
            - Formula simple para el **cálculo del esfuerzo**.
        - **Nivel inicial de diseño**:
            - **Establecer los requerimientos**.
            - **Conversión de puntos objeto a líneas de código**.
            - Basado en las **fórmulas estándar para** métodos **algorítmicos**
        - **Nivel post-arquitectura**:
            - **Estimación basada en código fuente**.
            - **La estimación se modifica** para que **tenga en cuenta**
                - **Volatilidad de requerimientos**.
                - **Reutilización del código**.
- **Estimación ascendente**
    - **Comienza con** los **componentes** y **estima el esfuerzo** necesario de cada uno por separado.
    - Este **esfuerzo se sumara** a la estimación final.
    - Ventajas:
        - Se puede **usar cuando la arquitectura es conocida y los componentes han sido identificados**.
        - Proporciona **estimaciones** bastante **exactas**.
    - Inconvenientes:
        - **Infra-estima el coste de la integración y la documentación**.
- **Estimación descendente**
    - Comienza en el **nivel de sistema** y **evalúa sus funcionalidades** y la subdivisión de estas en **subsistemas**.
    - Ventajas:
        - Se puede **usar sin conocer la arquitectura ni los componentes** del sistema
        - Tiene en cuenta la **integración, la documentación y la gestión de configuraciones**.
    - Inconvenientes:
        - **Infra-estima el coste de los problemas técnicos de bajo nivel** difíciles de resolver.
- Métricas de la calidad de especificación
    - Para calcular la calidad de la especificación se basa en: especificidad, compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización.
- Métricas del diseño arquitectónico
    - Calculan el valor de la métrica basándose en las características de la arquitectura del programa.
- Métricas de diseño a nivel de componentes
    - Se basan en las características internas de los componentes.
    - Tipos:
        - Cohesión
        - Acoplamiento.
- Métricas de diseño de interfaz
- **Modelo algorítmico de costes**
    - Aproximación basada en fórmulas obtenidas a partir de información histórica sobre proyectos de tamaño similar.
    - Inconvenientes:
        - Requiere una estimación previa de los atributos del proyecto terminado.

La **planificación** es la **actividad que más tiempo consume en la gestión** y **se realiza de forma continuada hasta la entrega** del producto. Se trata de **decidir de antemano el QUÉ, CÓMO, CUÁNDO y QUIÉN. Los pasos son:**

- **Establecer restricciones**
- **Hacer evaluaciones iniciales**
- **Definir hitos y entregas**
- Durante el proyecto
    - **Organizar Agenda**
    - Esperar
    - **Revisar progreso**
    - **Revisar estimaciones**
    - **Actualizar Agenda**
    - **Re-negociar estimaciones**
    - **Solucionar problemas**

Tipos de planes de proyectos

- Plan de calidad
- Plan de validación
- Plan de gestión de configuraciones
- Plan de mantenimiento
- Plan de gestión temporal

Las actividades se **deben** organizar de forma que **produzcan salidas válidas para la gestión** del progreso.
Los **hitos (milestones) marcan el final de una actividad** del proceso de desarrollo
Las **entregas (derivables) son resultados** del proyecto **que se entregan a los clientes**

Scheduling Consiste en la **organización temporal y asignación de recursos** a las actividades de un proyecto.

- **Pasos a seguir:**
    - **Determinación** de las **actividades a realizar**. (**requerimientos** software, **modelo de proceso**)
    - **Asignación** de **tiempos estimados**. (**dependencias y recursos**)
    - **Asignación** de **recursos**.
    - **Organización temporal** de las actividades. (**Diagramas** PERT y de Gantt)
- Inconvenientes:
    - La tarea de **estimar la dificultad** del problema y por lo tanto el coste asociado **es complicada**.
    - **La productividad no es proporcional al número de gente trabajando** en una tarea.
    - **Añadir gente a un proyecto** que va **con retraso lo retrasará más** todavía debido a la sobrecarga en cuanto a las comunicaciones personales.
    - **Lo inesperado SIEMPRE ocurre**. Se debe tener esto en cuenta a la hora de planificar.

Representaciones graficas (**Diagramas**)

- Se utilizan para **ilustrar la agenda** del proyecto.
- Permiten **mostrar una vista de la división** en tareas del proyecto.
- Los **diagramas de actividades muestran** las **dependencias** de las tareas **y el camino crítico**.
    - **PERT**
        - Datos de **entrada**:
            - **Lista de precedencias** del proyecto
            - Asignación de **tiempos y recursos** a actividades
        - **Proceso** a realizar:
            - Representar grafo.
                - **Representación de las Precedencias**
                - **4 tipos**:
                - Convergente. Una tarea depende de muchas
                - Divergente. varias tareas dependen de una
                - Convergente-divergente. De una tarea que depende de varias a su vez dependen varias.
                - Lineal. Una tarea depende de otra.
                - Las **actividades ficticias** se representan con líneas discontinuas
            - Cálculo de **tiempos “early” y “last”**.
                - ESj. Tiempo más temprano de inicio (Earliest Start) de una tarea
                - **ESj = max(ESi + Wij)**

        - LSi. Tiempo más lejano de inicio (Latest Start) de una tarea
            - Se mira de derecha a izquierda.
            - Se calcula después del Early.
            - **LSi = min(ESj - Wji)**
    - Cálculo de **holguras**
        - Holgura total. Tiempo que se puede **retrasar una tarea sin que afecte a la duración** del resto **del proyecto**
            - **HTij = LSj - ESi - Wij**
        - Holgura libre. Tiempo que se puede **retrasar una tarea sin que afecte a las fecha de comienzo de las de las tareas sucesoras**.
            - **HLij = ESj - ESi - Wij**
    - Cálculo del **camino crítico**.
        - Camino **más largo** del grafo
        - **Depende de** la fecha más temprana de la finalización del proyecto (**EFDt**)
        - **Todas las actividades del camino crítico tienen holgura 0**.
        - El camino crítico **no es único**.
    - Confección de **agenda**.
        - Fecha de comienzo (Start Date) más temprana (Earliest)
            - **ESDij = ESi**
        - Fecha de comienzo (Start Date) más lejana (Latest)
            - **LSDij = LSj - Wij**
        - Fecha de fin (Finish Date) más cercana (Earliest)
            - **EFDij = ESi + Wij**
        - Fecha de fin (Finish Date) más lejana (Latest)
            - **LFDij = LSj**
- Los **diagramas de barras muestran la agenda** del proyecto.
    - **Gantt**
        - Eje de ordenadas (Vertical) → Representa Actividades o Recursos
        - Eje de abscisas (Horizontal) → Tiempo
        - Permite **observar con detalle la evolución** del proyecto.

Los diagramas se pueden construir de forma automática gracias a la herramienta **Microsoft Project**

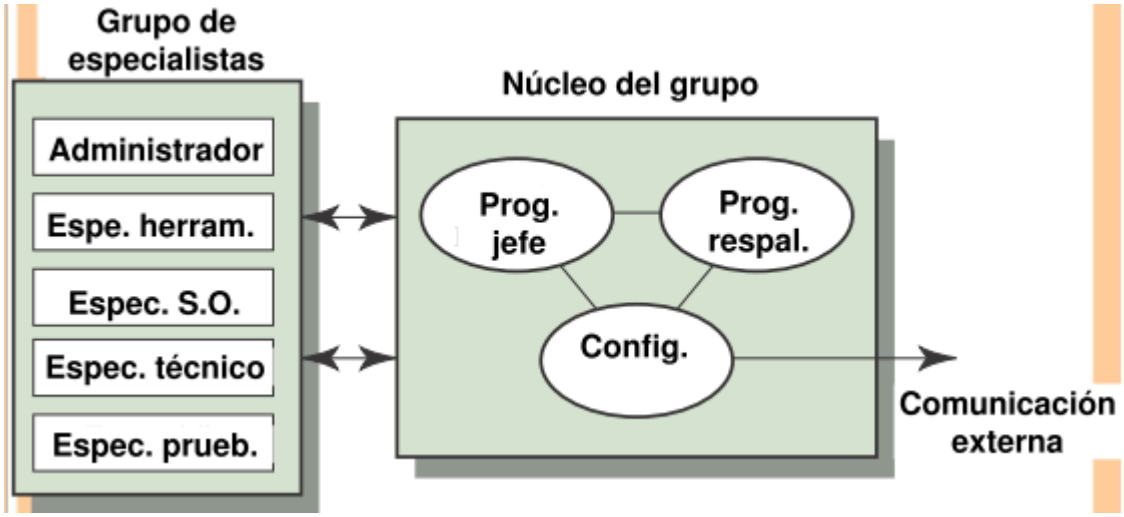
La **Gestión de riesgos** trata sobre la identificación de riesgos Y el desarrollo de planes de contingencia para minimizar sus efectos en el desarrollo.

- Se define como **riesgo** a la probabilidad de que pueda ocurrir alguna circunstancia adversa.
    - **Riesgos del proyecto (Afectan a la agenda o los recursos)**
    - **Riesgos del producto (Afectan a la calidad o realización del trabajo)**
    - **Riesgos del negocio (Afectan a la organización y gestión del proyecto)**
- El **proceso** es:
    - **Identificación** (**Lista de riesgos** potenciales)
    - **Análisis** (**Priorización de los riesgos**)
    - **Planificación** (**Planes de previsión y contingencia**)
    - **Monitorización** (Evaluación)
    - Vuelta al Análisis, **ciclo**

La **ingeniería del Software es principalmente** una actividad **colectiva**

Los **grupos de programación externa** son una **variante de la organización democrática**. Se pueden tomar **algunas decisiones de gestión por los miembros** del grupo y los **programadores** trabajan **en parejas**, compartiendo una **responsabilidad colectiva por el código**

En **grupos con jefe de trabajo**:



Problemas:

- Es **difícil encontrar buenos programadores y diseñadores**
- El **jefe de programadores puede asumir el éxito del trabajo en grupo**
- Puede **no ser posible formar el grupo**

Cada empleado varía su forma de pensar por unos factores:

- **Organización de la memoria** (Sentidos > Mem. corto plazo > Mem. de trabajo <> Mem. largo plazo)
  - En la transmisión de conocimientos entre la mem. corto plazo y la de trabajo está la resolución de problemas
  - Puede perderse información en dicha transmisión
  - En la transmisión entre mem. corto plazo y largo plazo se encuentra el procesamiento de la información
- **Representación del conocimiento**
  - **Conocimiento semántico:** Conocimiento de los **conceptos**. Se refiere a la **comprensión** de la realización de todas las funciones, significado de las entradas y las salida, y objetivos del sistema. Se adquiere con **experiencia y aprendizaje activo**.
  - **Conocimiento sintáctico:** Se refiere a los **detalles** de representación a la mecánica de interacción que se refiere para utilizar la interfaz de forma eficaz. Se adquiere mediante **memorización** y **conocimientos nuevos pueden chafar antiguos**
- **Motivación**

Actividades de un **Gestor de Proyectos**. Están **orientadas a la gente**:

- **Estimación** (Tiempo y esfuerzo **que costará**)
- **Planificación** (**Qué hay que hacer**)
- **Organización** (**Cómo se realizarán** las actividades)
  - **Selección del personal** (Basada en currículum, entrevista, recomendaciones y, en algunos casos, test de aptidud o psicológicos)
  - Los factores para seleccionar un individuo
    - Dominio de la aplicación
    - Dominio de la plataforma
    - Dominio del lenguaje de programación
    - Estudios previos
    - Capacidad de comunicación
    - Adaptabilidad
    - Actitud
    - Personalidad
  - **La composición de un grupo admite poca flexibilidad, hay que saber trabajar con la gente disponible**
  - La **interacción y comunicación** entre empleados **es crucial**

    - La **distribución del tiempo de trabajo** normalmente (**20% Actividades no productivas + 30% Trabajo + 50% Interacción**)
    - **No** es recomendable **trabajar con miembros con el mismo tipo de personalidad**, debe existir un **equilibrio** (Los **orientados a la tarea** suelen hacer las cosas como cada uno quiere, los **a sí mismos** tratarán de ser jefes todos, y los orientados **a la interacción** tendrán demasiadas charlas)
    - **Todos** los miembros del grupo **deben tener implicación**
    - Aunque una **organización democrática (descentralizada)** es **efectiva** (En **grupos pequeños** (< 8 componentes) y con **gente preparada**), debe **existir un jefe administrativo y técnico que mantenga una competencia** entre los miembros. En este tipo de organización:
        - **Decisiones por consenso**
        - **El líder es realmente un enlace, no realiza asignaciones**
        - **El trabajo se trata como un todo y se discute en grupo**
        - **Las tareas se reparten según la experiencia y habilidad** de los componentes
    - **Ventajas**
        - **Desarrollar estándares de calidad**
        - **Reducen las inhibiciones** causadas por la ignorancia e **incomunicación**
        - **Conocimientos y méritos compartidos** entre los miembros
        - **Programación sin ego (Concentrada en mejorar el trabajo de los demás)**
    - Esta cohesividad está **influenciada por cultura organizacional y las personalidades** de los empleados
    - Se puede **mejorar la cohesividad** mediante:
        - **Eventos sociales**
        - **Identidad y área propia del grupo**
        - **Actividades para el grupo**
        - **Sinceridad**
        - **Comunicación**. Esencial, debe informarse del estado, decisiones de diseño y cambios en decisiones previas respecto al proyecto. Está **influenciada por:**
            - **Status**
            - **Personalidades**
            - **Sexo**
            - **Canales de comunicación**
- **Solución a problemas** (**Utilizando los recursos existentes**).
    - Es **independiente del lenguaje de programación**
    - Integración de diferentes tipos de conocimiento
    - **Desarrollo** de un **modelo semántico de la solución**
    - **Prueba del modelo**
    - Debe **conducir a la representación de la solución en una notación adecuada**
- **Motivación** (De los integrantes)
    - **Es compleja**
    - Satisfacer la **pirámide de necesidades de un individuo** (**Fisiológicas > Seguridad > Sociales > Estima > Auto-realización**)
    - **Las necesidades pueden cambiar por circunstancias y eventos externos**
    - También **varían según el tipo de personalidad** del empleado (Orientados a la tarea (ingenieros), a sí mismos (políticos y abogados), a la interacción (médicos)
- **Control** (**Del progreso** del proyecto)

El **entorno de trabajo** (físico) juega un papel **importante en la productividad y satisfacción de los empleados**. Se deben tener en cuenta consideraciones respecto a su salud como una buena iluminación, una correcta climatización, un mobiliario cómodo...

El modelo **modelo CMM de personal** pretende ser un **marco para la gestión del trabajo del personal y las capacidades de los recursos humanos**. se divide en **5 etapa**s:

- **Inicial** (Gestión de recursos humanos ad-hoc)
- **Repetible** (Políticas de mejora de las aptitudes)
- **Definido** (Estándar de la gestión de recursos humanos)
- **Gestionado** (Metas cuantitativas en la gestión de recurso humanos)
- **Optimizado** (Esfuerzo para mejorar la competencia y motivación)

Y sus **objetivos**:

- **Mejorar la organización a través de las capacidades de trabajo**
- El **desarrollo** de software **debe implicar al mayor número de personal posible**
- **Igualar los niveles de organización y motivación** del grupo
- **Mantener a la gente preparada y con habilidades** críticas

La **Gestión de versiones NO** se refiere al **mantenimiento**. Es una **actividad de autoprotección** para

- **Identificar cambios** (Versiones y releases)
  - **Versión**. Difiere ligeramente de otras instancias previas. Es **interna**, para la empresa
    - Suele ser la mejor opción para corregir fallos (**modificaciones menores**)
  - **Release**. **Cambios mayores** o nueva plataforma. Se **distribuye a los clientes**. **Incluye:**
    - **Ficheros de configuración**
    - **Ficheros de datos**
    - **Instalación del programa**
    - **Documentación física y electrónica del sistema**
- **Controlar los cambios**
  - **Herramientas de Gestión de versiones**
    - **SCCS**
      - **Identificación de versiones**
      - **Cambios controlados**
      - **Gestión de almacenamiento**
      - **Registro de la historia de cambios**
    - **RCS**
      - Todo lo de SCCS +
      - **Graba los fuentes de la última versión**
      - **Soporta desarrollo en paralelo de releases**
      - **Capaz de mezclar versiones**
- **Garantizar la implementación de los cambios**
  - **Revisiones técnicas** formales
  - **Auditorías de configuración**. Responden
    - ¿Se ha cambiado?
    - ¿Se han seguido los estándares?
    - ¿Se ha señalado, registrado y divulgado apropiadamente?
    - ¿Se han actualizado los ECs?
- **Informar de cambios**
  - Informes de estado. Responden
    - ¿Que pasó?
    - ¿Quién lo hizo?
    - ¿Cuándo ha pasado?
    - ¿Qué más cambió?

La distribución de **nuevas releases se comentan con el cliente**

Para **planificar** la **Gestión de versiones**

- **Definir las entidades y su esquema**
- **Identificar los responsables** de los procedimientos
- Describir **cómo mantener elementos de configuración** (**EC**)
  - **Se requerirán para un futuro mantenimiento**
  - Nombrado jerárquico
    - **Asociado a proyectos particulares**
    - **No reusable**
  - **No** deberían **cambiar de forma aleatoria**
  - **Ejemplos típicos**
    - **Especificación del sistema**
    - **Plan del proyecto** software
    - Especificación de **requerimientos del software**
    - **Manual de usuario preliminar**
    - **Especificación de diseño**
    - **Diseño preliminar y detallado**
    - Listados de **código fuente**
    - **Planes y procedimientos de prueba**
    - **Manual de instalación, operación y usuario**
    - **Ejecutables software**
- **Describir las herramientas utilizadas**
- **Definir** la **Base de Datos de Configuración** (**BCD**)
  - **Registra cualquier información relacionada con las configuraciones**
  - **Apoyo para la evaluación del impacto del cambio**
  - Debe responder

        - **¿Qué clientes tienen una versión particular?**
        - **¿HW y SW necesarios?**
        - **¿Número de versiones y fechas de creación?**
        - **¿Qué versión cambiaría con el cambio de un componente?**
        - **¿Número de peticiones de una determinada versión?**
        - **¿Número de fallos en una versión?**
    - **Puede crearse por separado o integrarse con la gestión de versiones y el sistema de control de EC**

La **Construcción del sistema** implica:
- **Compilación**
- **Linkado**

Factores a considerar
- Hay que tener **cuidado con sistemas diferentes al del entorno de desarrollo**
- **¿Todos los componentes incluidos?**
- **¿Versión adecuada?**
- **¿Todos los ficheros disponibles?**
- **¿Los datos tienen el mismo nombre en el destino?**
- **¿La versión del compilador es correcta?**

Herramientas de **construcción de sistema**
- **MAKE**
    - **Mantiene correspondencia entre fuentes y versiones de código objeto**
    - Una vez especificadas las dependencias (Makefiles), **fuerza la compilación del código cambiado**
    - **Limitaciones**
        - Modelo físico de **dependencias (Makefiles)**
        - **Makefiles complejos, incomprensibles y difíceles de mantener**
        - **Cambios basados en fechas de actualización**
        - **No permite especificar la versión del compilador**
        - **No enlazado (finamente) con herramientas de GC** como RCS
- **Ventajas de GCS**
    - **Reduce esfuerzo para gestionar y realizar cambios**
    - **Mejora en la integridad y seguridad del software**
    - **Genera información sobre el proceso**
    - **Mantiene la BD de desarrollo de software**

El 72% de los proyectos cuestan más de lo presupuestado o se entregan más tarde de lo acordado
El 52% de los proyectos se entregan con un 189% de coste de lo presupuestado

Se define como **seguimiento** o **monitorización** de la agenda de un proyecto a comprobar si se cumple los tiempos y costes planificados. Se pueden distinguir 2 agendas:

- **Agenda planificada**
  - **Creada al inicio**
  - Sólo necesita establecer una fecha de inicio
  - **Utiliza información planificada**
- **Agenda real**
  - Se crea **durante el progreso** del proyecto
  - **Utiliza información real**

Las **métricas de seguimiento** del proyecto **miden:**

- **Fechas de inicio y fin**
- **Duraciones**
- **Holguras** totales y libres
  - Holgura total. Tiempo que se puede **retrasar una tarea sin que afecte a la duración** del resto **del proyecto**
    - **HTij = LSj - ESi - Wij**
    - Traducción: **HTij = LCj - CAi - T**
  - Holgura libre. Tiempo que se puede **retrasar una tarea sin que afecte a las fecha de comienzo de las de las tareas sucesoras**.
    - **HLij = ESj - ESi - Wij**
    - Traducción: **HLij = CAj - CAi - T**
  - Leyenda:
    - CA. Comienzo anticipado (Fecha más temprana de comienzo)
    - LC. Límite de comienzo (Fecha más tardía de comienzo)
    - T. Tiempo de actividad (Tiempo en completar una actividad)
  - Una **tarea crítica** es aquella **con HT = 0**. No se puede retrasar sin afectar al progreso del progreso
- **Camino crítico**
  - Lo **componen actividades críticas** (con HT = 0)
- Análisis del valor acumulado (**EVA**)
  - A las tareas se le asignan porcentajes de completado
  - Permiten **predecir dificultades** en el cumplimiento de la agenda
  - 3 tipos de información básicos:
    - **Valor planificado. Trabajo que debería haberse completado** (BCWS)
      - **BCWS = coste recursos teórico** (salario teórico de empleado/dia(hora)) x duración de la tarea (día (hora))
      - BCWS = 0 significa que la tarea no ha comenzado
    - **Coste real**. Coste del proyecto en la actualidad (ACWP)
      - Sucede en caso de que un recurso cueste más de lo previsto
      - **ACWP = salario real de empleado/dia(hora)) x duración de la tarea (día (hora)**
    - **Valor acumulado**. Valor, en términos de coste de línea base, **del trabajo realizado en un instante concreto del desarrollo** (BCWP)
      - **BCWPi = BCWS - (% completado en el instante i * BCWS))**
  - Si BCWP > BCWS la tarea/ proyecto va adelantada (improbable)
  - Si BCWP > ACWP la tarea/ progreso cuesta menos de lo estimado (improbable)
  - De los que surgen **Indicadores de progreso**
    - Schedule Variance (SV)
      - **SV = BCWP - BCWS**
      - Si SV < 0 el proyecto lleva un retraso de trabajo
    - Schedule Performance Index (SPI)
      - **SPI = BCWP / BCWS**
      - Un SPI > 1 significa que va adelantado
  - E **Indicadores de productividad**
    - Cost Variance (CV)
      - **CV = BCWP - ACWP**
      - Un CV < 0 es que el proyecto está costando más de lo planeado
    - Cost Performance Index (CPI)

            - **CPI = BCWP / ACWP**
            - Un CPI > 1 significa que se está gastando menos de lo que estaba planificado
    - Si el progreso es lento, pero la productividad buena (SPI < 1 y CPI >1) significa que necesitamos más gente
    - Si se está gastando menos o tarda menos de lo esperado en ciertas tareas podemos asignar esos recursos a tareas que lo necesiten
    - Si nos encontramos con sobrecostes deberemos aumentar el presupuesto o reducir el margen de beneficio
    - El parámetro Budget at Completion (BAC)
        - **BAC = Cantidad de trabajo planificado al final del proyecto/ tarea**
        - Si BCWS = BAC la tarea debería haberse completado
        - Si BCWS < BAC Todavía no ha terminado

MS Project permite mostrar las holguras libres (Demora permisible) de forma gráfica, también permite determinar la holgura total (Margen de demora total).

Además también tiene campos para las actividades en los que especificar el Comienzo anticipado, Límite de comienzo, Fin anticipado y Límite de finalización

MS Project utiliza 5 tipos de **parámetros**
- duración
- trabajo
- f. comienzo
- f. fin
- coste

Que permiten **evaluar el progreso**:
- **Planificado**
- **Programado. (Información actualizada)**
- **Real**
- **Restante**

Si usas MS Project en español es posible que estos parámetros están traducidos
- BCWS = CPTP
- ACWP = CRTR
- BCWP = CPTR