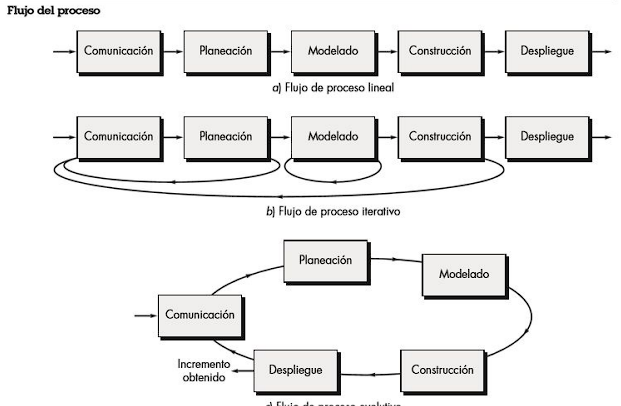
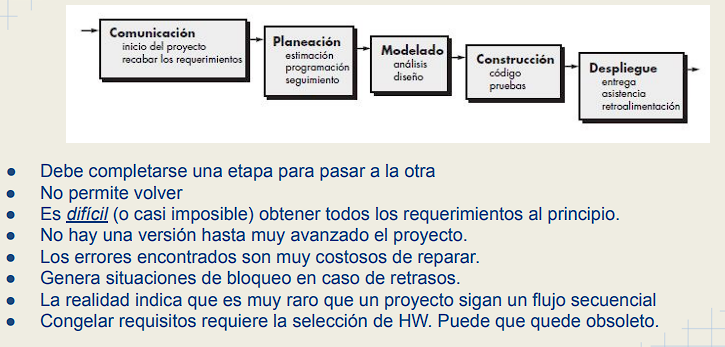
**RESUMEN DE LOS CONTENIDOS DE INTRODUCCION A LA ADMINISTRACION DE PROYECTO**

**MODELOS DE CICLO DE VIDA**

****

**MCV – CASCADA**

El modelo en cascada, propuesto por Royce en 1970, organiza el desarrollo de software en etapas secuenciales donde cada etapa debe completarse antes de comenzar la siguiente. Primero se identifican y documentan los requerimientos, luego se avanza al diseño y así sucesivamente. Este modelo proporciona una visión clara de la secuencia de actividades. Fue ampliamente utilizado, especialmente en proyectos del Departamento de Defensa de EE.UU., donde cada etapa incluía hitos y entregables que permitían medir el progreso del proyecto, como en la fase de pruebas unitarias e integración, donde el código probado servía como producto intermedio para la verificación del sistema completo.



**Etapas del desarrollo de software en el modelo en cascada:**

1. **Ingeniería y modelado de sistemas/información:** Se identifican los requisitos de todo el sistema (software, hardware, personas, datos) para definir qué parte le corresponde al software.
2. **Análisis de requisitos:** Se profundiza en los requisitos específicos del software, entendiendo su función, comportamiento y conexiones.
3. **Diseño:** Se traduce lo anterior en una representación del software, definiendo la estructura de datos, arquitectura, interfaces y algoritmos.
4. **Codificación:** Se genera el código basándose en el diseño detallado.
5. **Pruebas de integración:** Se verifica y valida que el software funcione correctamente y cumpla los requisitos.
6. **Mantenimiento:** Después de la entrega, se realizan cambios para corregir errores, adaptarse a nuevos entornos o mejorar el software.

Una variación del modelo en cascada permite retroceder entre etapas o incluso redefinir requisitos tras ver el producto final.

**Limitaciones del modelo en cascada:**

* Los proyectos reales rara vez siguen el modelo de forma estricta; no contempla bien alternativas ni decisiones cambiantes durante el desarrollo.
* Asume que los requisitos están completamente definidos al inicio y no cambiarán, lo cual es poco realista.
* Requiere definir hardware temprano, lo que puede llevar a usar tecnología obsoleta al finalizar el proyecto.
* No se obtiene una versión funcional hasta muy avanzado el desarrollo, dificultando la retroalimentación temprana del cliente.
* Produce "bloqueos" donde los equipos esperan a otros para seguir trabajando, generando ineficiencia.
* No muestra cómo transformar los artefactos de una fase a otra ni cómo manejar cambios en los productos.
* Trata el desarrollo de software como manufactura en serie, cuando en realidad es un proceso creativo y evolutivo basado en la comprensión progresiva del problema.

**Aspectos positivos:**

* Las etapas están organizadas lógicamente: cada fase depende de decisiones tomadas en fases anteriores.
* Cada etapa incluye revisiones y aprobación antes de avanzar, ayudando a reducir errores.
* Permite cierta iteración: problemas detectados en fases posteriores pueden afectar fases anteriores.

**MODELO EN V**

El **modelo V** es una variante del ciclo de vida en cascada que enfatiza la validación de productos y muestra cómo las actividades de prueba se relacionan con el análisis y diseño. La codificación ocupa el centro de la "V", con análisis y diseño a la izquierda, y pruebas y mantenimiento a la derecha.

Propone que:

* Las **pruebas unitarias e integración** verifiquen tanto el código como el diseño del programa.
* La **prueba del sistema** asegure que todo el diseño del sistema esté bien implementado.
* La **prueba de aceptación**, dirigida por el cliente, valide que se cumplan todos los requisitos antes de aprobar el sistema.

Además, si durante las pruebas se encuentran problemas, se puede volver a las etapas de análisis, diseño o codificación para corregirlos. A diferencia del modelo en cascada, que se enfoca en los documentos, el modelo V se centra en las actividades y la exactitud del producto.

**PROTOTIPADO**

Cuando el cliente no tiene claro lo que necesita o el ingeniero duda de la solución, es útil crear un **prototipo**: una versión inicial y simplificada del software. El **modelo basado en prototipos** busca superar las limitaciones del modelo en cascada, ayudando a entender mejor los requisitos del usuario.

Los prototipos también sirven para:

* Verificar la viabilidad del diseño.
* Evolucionar de forma iterativa hacia el sistema final.

El desafío es seleccionar correctamente qué funciones incluir en el prototipo, evitando enfocarse en características secundarias. Tras usar el prototipo, el cliente da su opinión y luego se inicia el desarrollo completo.

**Etapas**.

1. Análisis preliminar y especificación de requisitos
2. Diseño, desarrollo e implementación del prototipo
3. Prueba del prototipo.
4. Refinamiento iterativo del prototipo.
5. Refinamiento de las especificaciones de requisitos.
6. Diseño e implementación del sistema final.

**PROTOTIPO DESECHABLE**

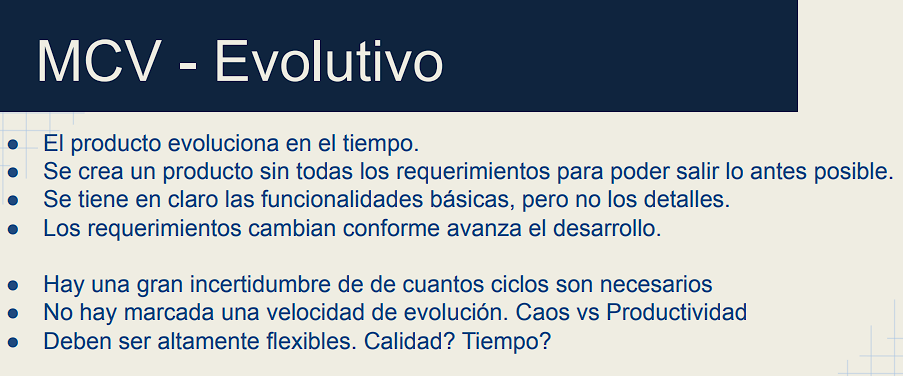
Ayuda a identificar requisitos poco claros; solo se implementan aspectos desconocidos y luego el prototipo se descarta.

**MAQUETA**

Muestra ejemplos visuales de entradas y salidas usando datos simples y estáticos, a diferencia del prototipo desechable que usa datos reales.

**EVOLUTIVO**

Es un modelo funcional y modificable del sistema que, tras definir todos los requisitos, se convierte en el producto final. Utiliza diseño detallado y datos reales desde el inicio.



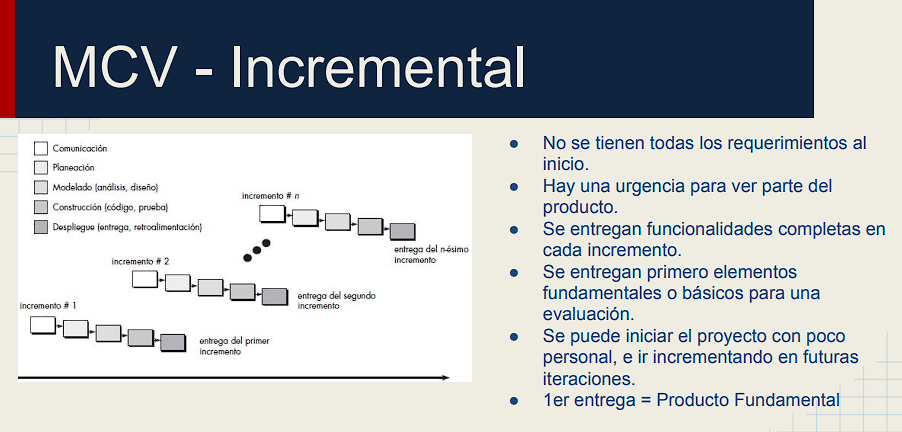
**MODELOS EVOLUTIVOS**

**MODELO INCREMENTAL**

También llamado **modelo de ciclo de vida con emisión gradual**, combina el enfoque secuencial del modelo en cascada con la filosofía iterativa de los prototipos. El sistema se divide en subsistemas según su funcionalidad y se desarrolla en versiones incrementales.

* Se empieza con un **producto núcleo** básico y se agregan nuevas funciones en cada incremento.
* Cada nueva versión entrega al cliente una **funcionalidad completa y operativa**.
* Permite **retroalimentación** después de cada entrega para ajustar y mejorar el producto.
* Es útil cuando el personal es limitado, ya que los primeros incrementos requieren menos recursos.
* Ayuda a **gestionar riesgos técnicos**, entregando primero funcionalidades que no dependan de elementos inciertos como hardware nuevo.

A diferencia de los prototipos, el enfoque incremental **entrega productos reales** y utilizables en cada etapa.



**MODELO ITERATIVO**

También llamado **modelo de refinamiento sucesivo**, utiliza las mismas etapas del modelo en cascada, pero aplicadas de forma cíclica y progresiva. En lugar de completar una etapa de forma lineal, cada producto se va **refinando y mejorando** en varias iteraciones.

* Se entrega un **sistema completo básico** desde el inicio, mejorando su funcionalidad y calidad en cada nueva versión.
* **Diferencia con el incremental:** en el incremental se agregan nuevas funciones en cada versión, mientras que en el iterativo todas las funciones básicas están presentes desde el principio y se perfeccionan luego.

**Ventajas de combinar desarrollo iterativo e incremental:**

1. Se puede empezar el **entrenamiento** de usuarios temprano.
2. Se pueden **crear mercados** rápidamente para nuevas funcionalidades.
3. Permite **detectar y corregir problemas** de forma temprana y global.
4. El equipo puede **especializarse** en diferentes áreas en cada versión (por ejemplo, interfaz en una versión y rendimiento en otra).

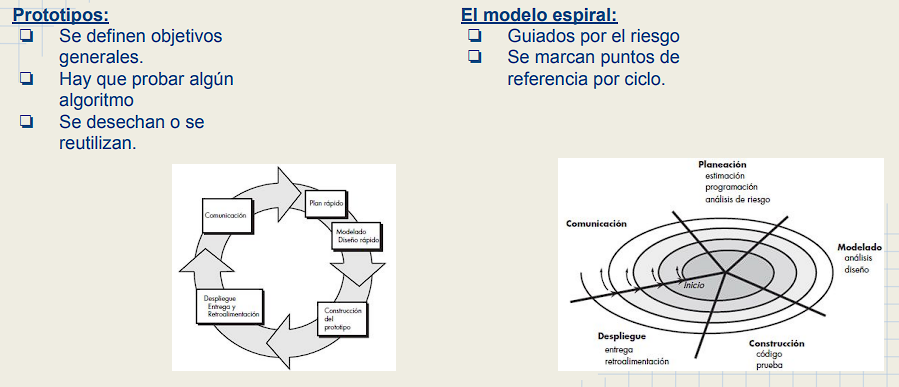
**MODELO ESPIRAL**

Presentado por Böehm, combina enfoques basados en especificaciones, prototipos y análisis de riesgos. Representa el proceso de desarrollo como una **espiral de ciclos iterativos**, donde:

* El eje radial muestra los **costos acumulados**.
* El eje angular indica el **progreso**.
* Cada ciclo incluye **análisis de riesgos** para identificar y reducir problemas potenciales.

**Ventajas:**

* Permite combinar modelos tradicionales y manejar mejor los riesgos.
* Fomenta la **reutilización de software existente**.
* Se enfoca en **eliminar errores** y **evaluar alternativas**.
* No distingue entre desarrollo y mantenimiento.
* Es ideal para **proyectos integrados de hardware y software**.



**MODELOS DE CICLO DE VIDA ALTERNATIVOS**

**MODELO DRA (Desarrollo Rápido de Aplicaciones)**

Es un modelo basado en el **cascada** adaptado para lograr ciclos de desarrollo **muy cortos** (60–90 días), usando **componentes reutilizables** y **herramientas automáticas**.

**Fases del DRA:**

* **Modelado de Gestión:** Se modela el flujo de información de la gestión.
* **Modelado de Datos:** Se definen objetos de datos, atributos y relaciones.
* **Modelado del Proceso:** Se transforman los datos para implementar funciones de gestión.
* **Generación de Aplicaciones:** Se reutilizan o crean componentes con técnicas de cuarta generación.
* **Pruebas y Entrega:** Se prueba principalmente lo nuevo, ya que los componentes reutilizados están previamente testeados.

**Ventajas:**

* Permite desarrollar sistemas funcionales rápidamente si los requisitos son claros y el proyecto tiene un alcance limitado.

**Inconvenientes:**

* Requiere **muchos recursos humanos** para proyectos grandes.
* Depende del **compromiso** activo de clientes y desarrolladores.
* No es adecuado para sistemas que no puedan **modularizarse bien**, donde haya **altos riesgos técnicos** o necesidad de **alto rendimiento**.

**Dato clave:**   
El DRA impulsa el desarrollo de **componentes reutilizables**, base de las tecnologías orientadas a objetos.

**ESTÁNDARES MILITARES Y PRÁCTICAS INDUSTRIALES**

Las empresas suelen adoptar variaciones del modelo clásico siguiendo **estándares militares** como la **MIL-STD-2167** (EE.UU.) o **ESA PSS-05-0** (Agencia Espacial Europea). Estos estándares regulan el ciclo de vida del software, definiendo **productos entregables, revisiones, hitos** y asegurando **calidad, gestión de configuración** y **verificación independiente**.

**Norma MIL-STD-2167 (EE.UU.):**

* Usada por las Fuerzas Armadas para proyectos de hardware y software.
* Actualmente aplicada solo a **software**.
* El sistema se divide en: **CSCI → CSC → CSU**.
* **Etapas principales:** análisis de requisitos, diseño (preliminar y detallado), codificación, integración, verificación y pruebas del sistema.
* **Crítica:** exige **mucha documentación**, con información redundante.

**Norma ESA PSS-05-0 (Europa):**

* Usada por la Agencia Espacial Europea para proyectos de software bajo contrato.
* Cubre además la **gestión del proyecto**, **configuración** y **calidad**.
* **Etapas principales:** definición de requisitos, diseño arquitectónico y detallado, desarrollo, transferencia al usuario, operación y mantenimiento.
* La **fase de transferencia** es nueva respecto al ciclo clásico, necesaria para instalar y formar usuarios en otro país.
* **Ventaja:** es **sencilla y clara**, útil como referencia para otros desarrollos.

**MODELO DE ENSAMBLAJE DE COMPONENTES**

Este modelo se basa en **reutilizar clases** (o componentes) previamente creadas y almacenadas en una **biblioteca de clases**.

* Primero se **identifican las clases candidatas**; si existen, se reutilizan; si no, se crean nuevas usando técnicas **orientadas a objetos**.
* La primera versión de la aplicación se construye combinando clases existentes y nuevas.
* El proceso sigue un enfoque **iterativo y evolutivo**, similar al **modelo en espiral**.

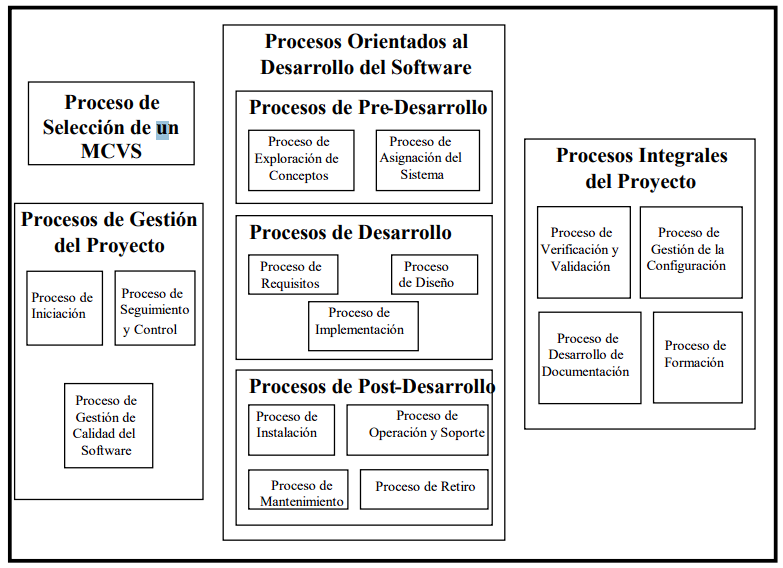
**Beneficios:**

* Favorece la **reutilización de software**, optimizando el uso de recursos.
* La **orientación a objetos** facilita la creación de clases reutilizables que encapsulan datos y algoritmos.

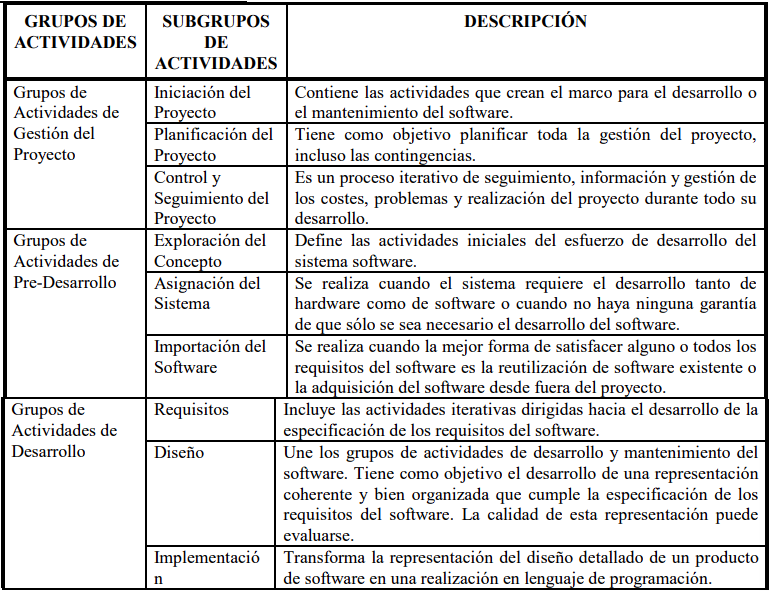
**Resumen de la actividad de ingeniería:**

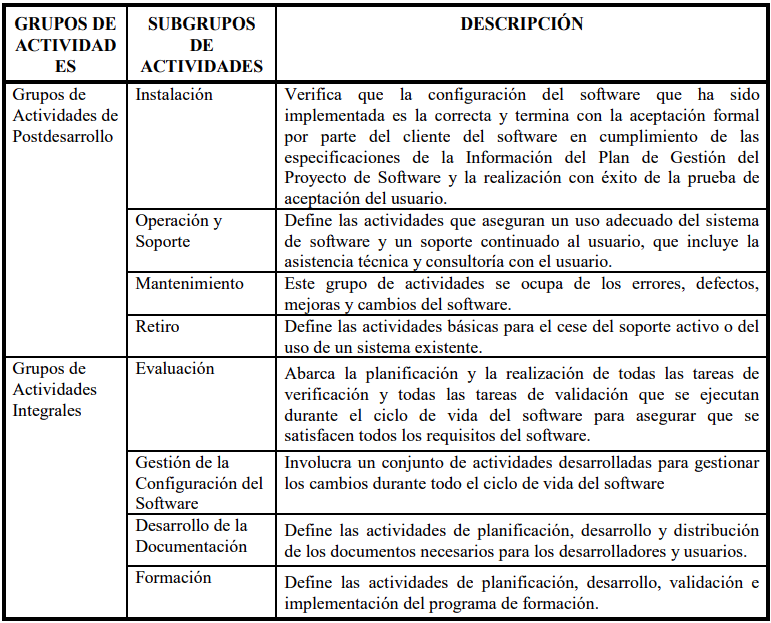
* Se identifican los datos y algoritmos necesarios.
* Se empaquetan juntos en nuevas clases si no existen en la biblioteca.

**ESTANDAR IEEE PARA EL DESARROLLO DE PROCESOS DE CICLO DE VIDA DEL SOFTWARE.**

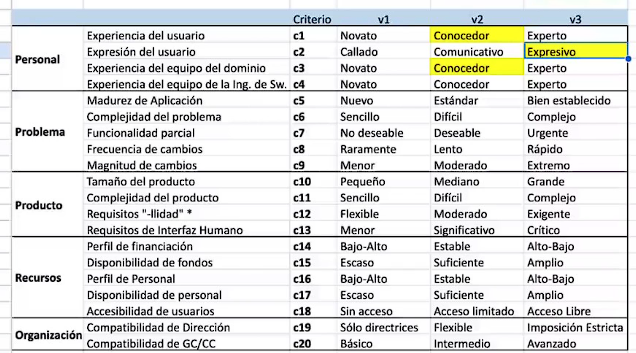
****

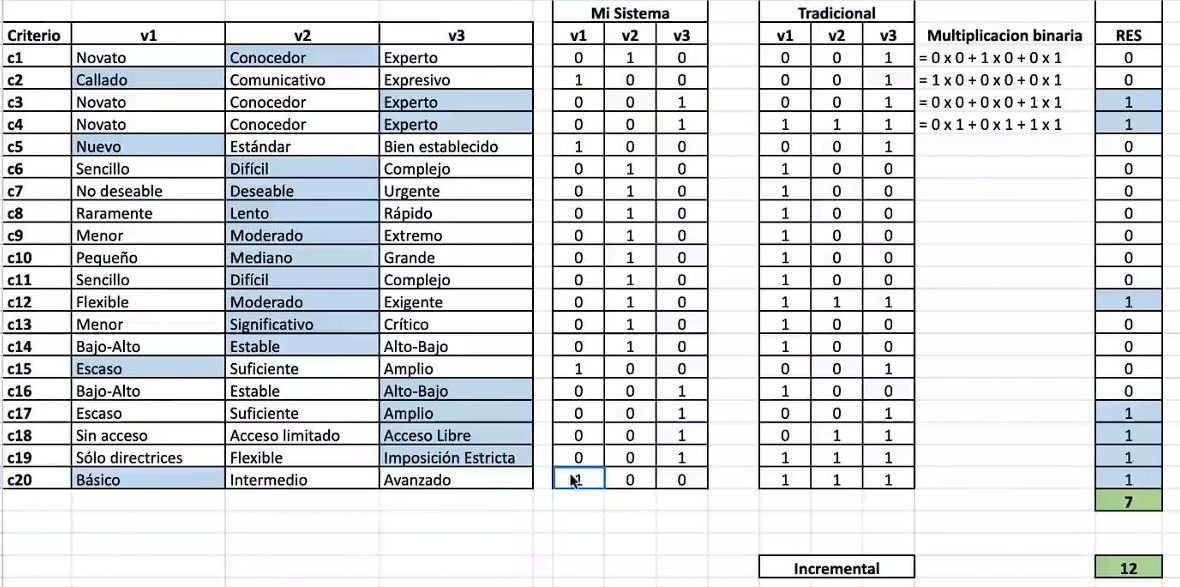
* Grupos de Actividades de Gestión del Proyecto: crean la estructura del proyecto y aseguran el nivel apropiado de gestión del mismo durante todo el ciclo de vida del software.
* Grupos de Actividades Orientadas al Desarrollo del Software: producen, instalan, operan y mantienen el software y lo retiran de su uso. Se clasifican en grupos de actividades de predesarrollo, desarrollo y post-desarrollo
  + Grupos de Actividades de Pre-Desarrollo: son las actividades que se deben realizar antes de que comience el desarrollo propiamente dicho del software.
  + Grupos de Actividades de Desarrollo: son las actividades que se deben realizar para la construcción del producto de software.
  + Grupos de Actividades de Post-Desarrollo: se realizan después de la construcción del software, es decir, se aplican a las últimas fases del ciclo de vida del software.
* Grupos de Actividades Integrales del Proyecto: son necesarios para completar con éxito las actividades del proyecto de software. Son simultáneos a las actividades orientadas al desarrollo del software e incluyen actividades no de desarrollo.

****

****

Los Grupos de Actividades de Gestión del Proyecto establecen las condiciones para el desarrollo del proyecto. Involucran las actividades de iniación del proyecto, asignación de recursos, la planificación, el seguimiento y el control durante todo el ciclo de vida del software. Los Grupos de Actividades Orientadas al Desarrollo inician el esfuerzo de desarrollo con la identificación de la necesidad de automatización. Una aplicación nueva o un cambio de todo o parte de una aplicación puede ser necesario para satisfacer la necesidad. En base del informe de la necesidad, los Grupos de Actividades de Desarrollo producen el software (código y documentación) con el apoyo de los Grupos de Actividades Integrales y según el Información del Plan de Gestión del Proyecto del Software. Por último, deben realizarse las actividades para la instalación, operación, soporte, mantenimiento y retiro del producto. Los Grupos de Actividades Integrales son simultáneos y complementarios a los procesos orientados al desarrollo. Incluye las actividades que son esenciales para asegurar que el sistema construido es fiable y se utiliza al máximo de sus posibilidades.

**SELECCIÓN MVC  
**



**PERSONAL**

* Desde 1960 Se estudia la formación de personal Motivado y Capacitado
* Factor humano es tan importante que se desarrolló el People-CMM .
* Modelo de madurez del personal
  + Mejorar habilidades
  + Motivar, desarrollar, mejorar y conservar la fuerza de trabajo
  + Define areas claves como:
    - Ambiente de trabajo / Clima
    - Capacitación
    - Competencias
    - Desarrollo profesional
    - Bonificación por desempeño
    - Beneficios
* Participantes:
  + Gerentes ejecutivos
    - Gran influencia sobre los proyectos
  + Gerentes de proyectos
    - Quienes administran (planifican, controlan, organizan)
  + Profesionales
    - Aportan habilidades y conocimientos
  + Clientes
    - Realizan las especificaciones del sw.
  + Usuarios finales
    - Quienes interactúan con el sw
* Lideres del equipo
  + Requiere mucho trato con la gente
  + Habilidades blandas
  + Modelo MOI para liderazgo técnico
    - Motivación
    - Organización
    - Ideas/innovación
  + Gerentes del proyecto
    - Resolución de problemas
    - Identidad
    - Incentivos por logro
    - Influencia y construcción de equipo
* El equipo SW
  + Cada equipo tiene una estructura y depende de:
    - Organización
    - Dificultad del problema
    - Tamaño del producto
    - Tiempo
    - Grado de modularización
    - Rigidez de la fecha
    - Grado de sociabilidad/comunicación

**PRODUCTO**

* Se debe pensar en el producto antes de pensar o planear un proyecto
* Objetivos y Ámbito
* Analizar alternativas
* Restricciones
* Sino es muy difícil hacer estimaciones precisas
  + Costos
  + Riesgos
  + Tiempos
* ¿Se puede descomponer?

**PROCESO**

* Proporciona el marco conceptual.
* Se debe elegir el marco “adecuado” que pueda adaptarse
  + Personal, producto y entorno
* Variar la cantidad de actividades
  + Se deben seleccionar los procesos/subprocesos correctos
* Actividades sombrilla o “cross”
  + Ocurren a lo largo del proceso
  + No importa el tamaño del SW ni el marco de proceso elegido

**PROYECTO**

* Se planean y se controlan:
  + Única forma de controlar su complejidad
* Se debe definir como planificar, monitorear y controlar
  + Gerente proyecto es responsable
* Se deben establecer indicadores que notifiquen ciertos parámetros
  + Alertas
  + Semáforos

**TIPOS DE PLANIFICACION**

*Fecha resultante*: Es la que sale suponiendo una fecha de comienzo y estimando hacia adelante.

*Fecha estipulada*: Es la fecha que debemos terminar, por lo tanto, de debe comenzar a estimar hacia atrás

La estimación de alimenta de Recursos, Actividades, Costos, Riesgos, Limitaciones, Esfuerzo. ¿Qué herramientas debo usar? ¿Cuál es el nivel de precisión? ¿Como se estimará la contingencia? ¿Como se va a controlar? ¿Cuándo y cómo se presentan los niveles de avance? 1.Tenemos que conocer previamente el ámbito del SW, documentos, casos de usos. ¡Validar! 2. Recursos a utilizar: Personal. Habilidades. SW reutilizable. Componentes existentes, comerciales, nuevos. Entorno de desarrollo. HW y SW para el desarrollo y producción. La precisión se basa en: El grado con que se estimó el tamaño del SW La habilidad para traducir la estimación en personas, tiempo y dinero.

Tareas:

* 1. Establecer el ámbito del proyecto
  2. Determinar la factibilidad
  3. Analizar los riesgos
  4. Definir recursos requeridos
     + A. humanos
     + B software reutilizable
     + Ambientales
  5. Estimar costo y esfuerzo
     + Descomponer
     + Dimensionar
  6. Desarrollar un calendario de proyecto
     + Tareas significativas
     + Red/secuencia de tareas
     + He4rramienta de calendarización
     + Mecanismos de seguimiento

**ESTIMAR COSTO Y ESFUERZO**

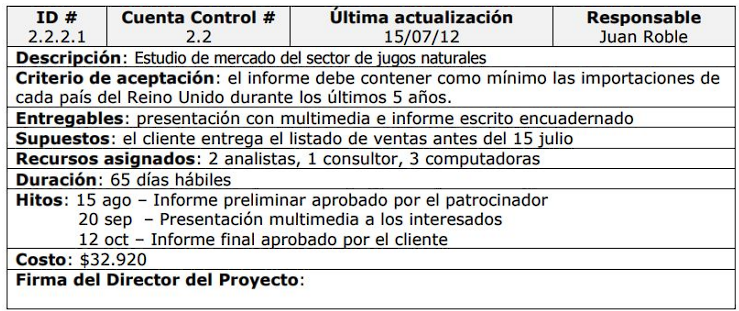
* Descomponer el problema
* Desarrollar dos o más estimaciones usando tamaño, puntos de función, tareas de proceso o casos de uso
  + Reconciliar estimaciones

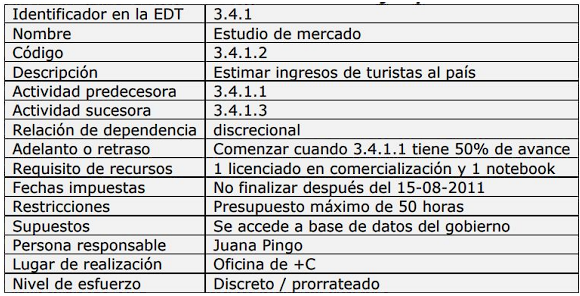
**DESCOMPOSICION**

ETD/WBS (Estructura de desglose de trabajo) : Organigrama jerárquico del proyecto donde se subdivide el mismo en menores componentes.

El de mas bajo nivel se llama “Paquete de trabajo”

* Consiste en dividir en componentes menores para facilitar la planificación y control.
* Es un organigrama jerárquico donde se va subdividiendo en menores componentes.
* El nivel más bajo se denomina paquete de trabajo, generalmente se agrupan bajo “Paquetes de control”.
* No define secuencia





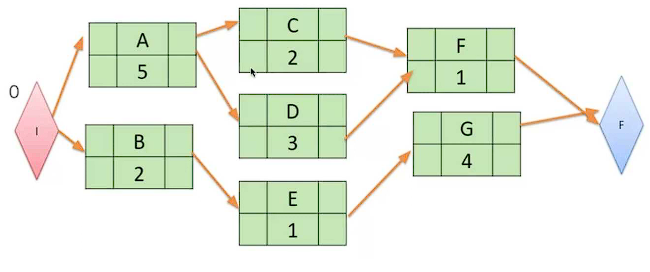
**SECUENCIAR ACTIVIDADES**

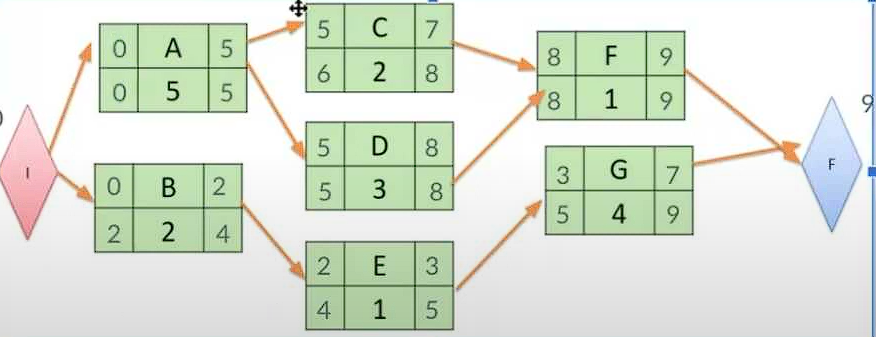
Tipos de dependencias:

* 1. **Dependencias obligatorias**: “lógica dura”. Una dependencia obligatoria es inherente a la naturaleza del trabajo (se debe diseñar antes de construirla) o es requerida por el contrato.
  2. **Dependencias discrecionales**: “preferida, preferencial o lógica blanda”. Se puede cambiar una dependencia discrecional si es necesario, mientras que no puedes cambiar fácilmente los otros tipos de dependencias.
  3. **Dependencia externa**: se basa en las necesidades o deseos de una parte externa del proyecto.
  4. **Dependencia interna**: se basa en las necesidades del proyecto puede controlar.

**CALCULO DEL CAMINO CRITICO**

Es la secuencia o camino de actividades que, si se atrasa, se atrasa el proyecto





Holgura: diferencia de la primera fila: A: 0-2 y 12-14. Inicio tardío – inicio temprano. Ej: E: 35-9 = holgura 26

Camino cítrico(holgura=0): BDHJK. Duración: 69hs.

Si la actividad c se retrasa dos días,

