INGENIERÍA DE SOFTWARE I

UNIDAD I

PRODUCTOS DE SOFTWARE

- Productos GENÉRICOS – ENLATADOS - CERRADOS
  - Productos que son producidos por una organización para ser vendidos al mercado.
- Productos HECHOS A MEDIDA.
  - Sistemas que son desarrollados bajo pedido a un desarrollador específico.
- La mayor parte del gasto del software es en productos genéricos, pero hay más esfuerzo en el desarrollo de los sistemas hechos a medida.

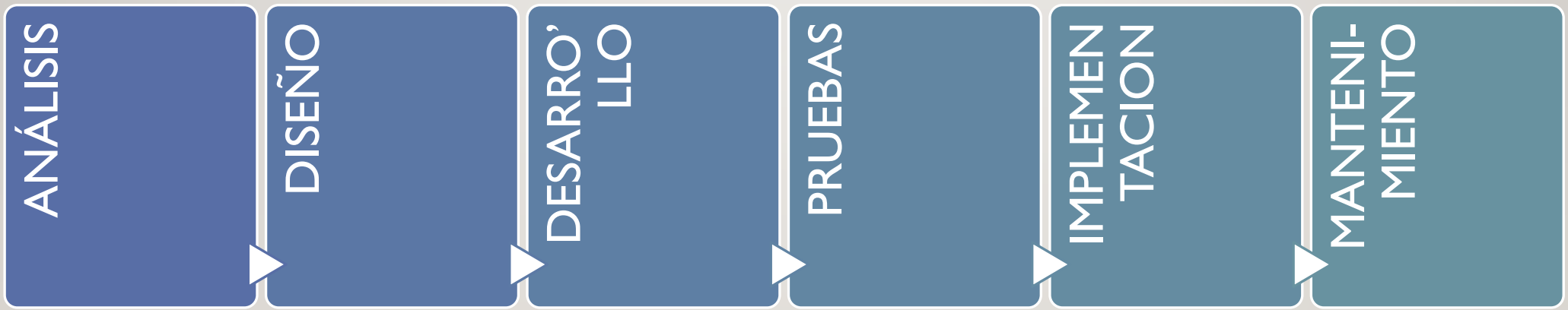
COSTOS DEL SOFTWARE

Los costos del software a menudo dominan al costo del sistema. El costo del software es a menudo más caro que el hardware.

Cuesta más mantener el software que desarrollarlo. Para sistemas con una larga vida, este costo se multiplica.

La Ingeniería de Software concierne a un desarrollo efectivo en cuanto a costos del software.

CICLO DE VIDA DEL SOFTWARE



ROLES


Administrador de proyectos


Analista

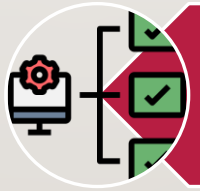

Diseñador


Programador


Téster


Documentador


Ingeniero de validación


Administrador de configuraciones


Cliente

CRM

ADMINISTRADOR DE PROYECTO

- El **Administrador de Proyectos** es el líder y estratega del equipo. Su principal objetivo es asegurar que el proyecto se complete a tiempo, dentro del presupuesto y cumpliendo con los estándares de calidad definidos.
- **Funciones Principales:**
  - **Planificación:** Define el alcance del proyecto, establece los objetivos, el cronograma y asigna los recursos necesarios.
  - **Gestión de Riesgos:** Identifica, evalúa y mitiga los posibles riesgos que puedan afectar el proyecto.
  - **Coordinación:** Facilita la comunicación entre los miembros del equipo, los clientes y otras partes interesadas.
  - **Supervisión:** Monitorea el progreso del proyecto y ajusta el plan según sea necesario.

ANALISTA

- El **Analista** actúa como el puente entre el cliente y el equipo de desarrollo. Su habilidad para entender las necesidades del negocio y traducirlas en requisitos técnicos es crucial para el éxito del proyecto.
- **Funciones Principales:**
  - **Levantamiento de Requisitos:** Se reúne con los clientes para comprender sus necesidades y expectativas.
  - **Análisis y Documentación:** Analiza los requisitos, los organiza y los documenta de manera clara y precisa (por ejemplo, a través de historias de usuario, casos de uso o especificaciones de requisitos de software).
  - **Validación:** Asegura que los requisitos sean claros, completos y consistentes antes de que el equipo de desarrollo comience a trabajar.

DISEÑADOR

- El **Diseñador** se enfoca en la experiencia del usuario (UX) y la interfaz de usuario (UI). Su trabajo es asegurar que el software no solo sea funcional, sino también intuitivo, agradable de usar y visualmente atractivo.
- **Funciones Principales:**
  - **Diseño de Experiencia de Usuario (UX):** Investiga y analiza el comportamiento del usuario para crear un flujo de trabajo lógico y una experiencia fluida.
  - **Diseño de Interfaz de Usuario (UI):** Crea la maqueta visual y la disposición de la interfaz de la aplicación, incluyendo la selección de colores, tipografía e iconografía.
  - **Prototipado:** Desarrolla prototipos interactivos para probar la usabilidad del diseño antes de la implementación.

PROGRAMADOR

- El **Programador** es el motor del proyecto, encargado de convertir los diseños y requisitos en código funcional.
- **Funciones Principales:**
  - **Codificación:** Escribe el código del software según las especificaciones de diseño y los requisitos funcionales.
  - **Resolución de Problemas:** Identifica y corrige errores o "bugs" en el código.
  - **Desarrollo de Características:** Construye nuevas funcionalidades y módulos del sistema.
  - **Mantenimiento:** Actualiza y mejora el código existente para garantizar su correcto funcionamiento a largo plazo.

TÉSTER

- El **Tester** es el guardián de la calidad. Su misión es encontrar y reportar defectos para asegurar que el producto final cumpla con los estándares esperados y funcione como se espera.
- **Funciones Principales:**
  - **Planificación de Pruebas:** Diseña casos de prueba y estrategias para validar el software.
  - **Ejecución de Pruebas:** Realiza pruebas funcionales, de rendimiento, de seguridad, de usabilidad, entre otras.
  - **Reporte de Errores:** Documenta y comunica los errores encontrados al equipo de desarrollo para su corrección.
  - **Validación:** Verifica que los errores corregidos realmente hayan sido solucionados.

DOCUMENTADOR

- El **Documentador** se encarga de crear y mantener toda la documentación relevante del proyecto, tanto para el equipo de desarrollo como para los usuarios finales.
- **Funciones Principales:**
  - **Documentación Técnica:** Escribe manuales de usuario, guías de instalación y archivos de ayuda.
  - **Documentación de Código:** Comenta el código para que sea comprensible para otros programadores.
  - **Creación de Guías de Usuario:** Prepara tutoriales y manuales para que los usuarios puedan utilizar la aplicación de manera efectiva.

INGENIERO DE VALIDACIONES

- El ingeniero de validación (o "Tester") es una figura vital en la ingeniería de software. Su trabajo es asegurar la calidad del producto final. No se limita a encontrar errores, sino que se involucra en todo el ciclo de vida del software para prevenir fallos.
- Y en muchos casos, verifica valida el hardware para que el software, corre sin dificultades. Especialmente, cuando hay disparidad de modelos.

ADMINISTRADOR DE CONFIGURACIONES

- El **Administrador de Configuraciones** es responsable de gestionar y controlar las diferentes versiones del código y de los componentes del software.
- **Funciones Principales:**
  - **Control de Versiones:** Utiliza herramientas como Git para rastrear y gestionar los cambios en el código.
  - **Gestión de Ramas (Branches):** Crea y gestiona las ramas de código para permitir el trabajo paralelo.
  - **Integración Continua:** Configura sistemas que permiten la integración automática de los cambios de código, evitando conflictos.

CLIENTE

- El **Cliente** es la razón de ser del proyecto. Es la parte interesada que define las necesidades del negocio y evalúa el producto final.
- **Funciones Principales:**
  - **Definición de Requisitos:** Proporciona la visión y las necesidades de lo que el software debe hacer.
  - **Proporcionar Feedback:** Evalúa las versiones del software y ofrece retroalimentación para guiar el desarrollo.
  - **Aceptación Final:** Acepta o rechaza el producto final basándose en el cumplimiento de los requisitos.

CRM

- El **CRM** significa **C**ustomer **R**elationship **M**anagement, o en español, **Gestión de la Relación con el Cliente**.
- En el contexto de la Ingeniería de Software, un CRM no es un rol, sino un tipo de **producto de software** que una empresa desarrolla o utiliza para gestionar las interacciones con sus clientes actuales y potenciales. Su objetivo es centralizar la información, mejorar las relaciones con los clientes y optimizar procesos de negocio como ventas, marketing y servicio al cliente.

MODELOS DE PROCESOS

EL PROCESO DE SOFTWARE

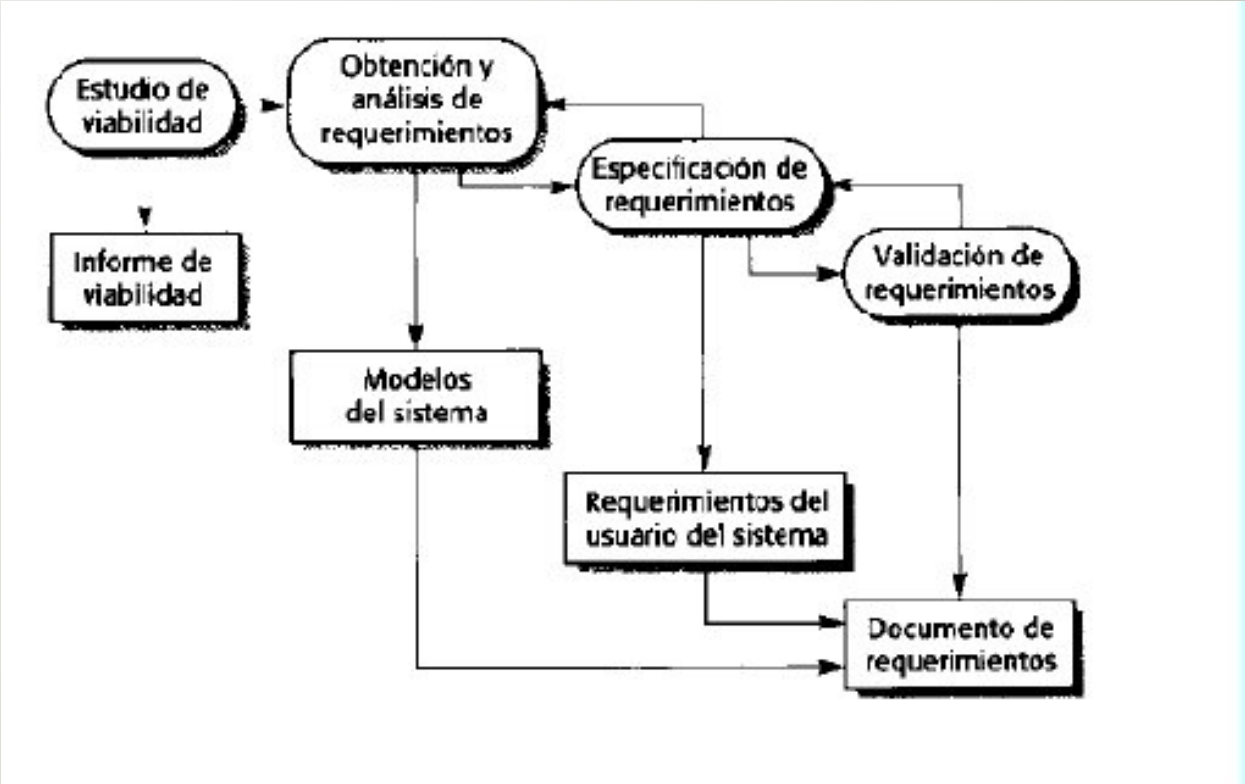
- El PROCESO es un método particular de hacer algo; generalmente involucra numerosos pasos y operaciones, incluyendo procedimientos, instrucciones de trabajo y estándares [Jenner-1995].
- Se puede definir: "*Una serie de actividades relacionadas entre sí, que convierten insumos en resultados (cambiando el estado de las entidades)*".
- También para describir un proceso, puede ser útil dibujar un diagrama de flujo, que capture los pasos o tareas del proceso y la relación interna y secuencias de estos pasos [Yeh-1993].
- Las actividades varían dependiendo de la organización y del tipo de sistema a desarrollar.

EL PROCESO DE SOFTWARE (2)

- Conjunto estructurado de actividades requeridas para desarrollar un sistema de software.
  1. Especificación del software.
  2. Diseño e implementación del software
  3. Validación del software
  4. Evolución del software.

1. ESPECIFICACIÓN DEL SOFTWARE

- Estudio de viabilidad.
- Obtención y análisis de requerimientos.
- Especificación de requerimientos.
- Validación de requerimientos.



2. DISEÑO E IMPLEMENTACIÓN DE SW

Diseño arquitectónico

Especificación abstracta

Diseño de interfaz

Diseño de componentes

Diseño de la estructura de datos

Diseño de algoritmos

3. VALIDACIÓN DE SOFTWARE

- Pruebas de componentes
- Pruebas de sistemas
- Prueba de aceptación

4. EVOLUCIÓN O MANTENIMIENTO DEL SW

- Mantenimiento correctivo
- Mantenimiento Adaptativo
- Mantenimiento Evolutivo

MODELOS GENÉRICOS DE DESARROLLO DE SOFTWARE

MODELOS GENÉRICOS DE DESARROLLO DE SOFTWARE

- Modelo de Cascada
  - Separa en distintas fases la especificación y el desarrollo.
- Desarrollo Evolutivo
  - La especificación y el desarrollo están intercalados.
- Prototipado
  - Un modelo sirve de prototipo para la construcción del sistema final.

MODELO DE CASCADA

- Es un modelo clásico que exige un enfoque sistemático y secuencial del desarrollo de software.
- Versiones diferentes del modelo pueden subdividir las etapas / fases en algo diferente; pero todas ellas tienen tareas similares.

La Metodología Cascada:

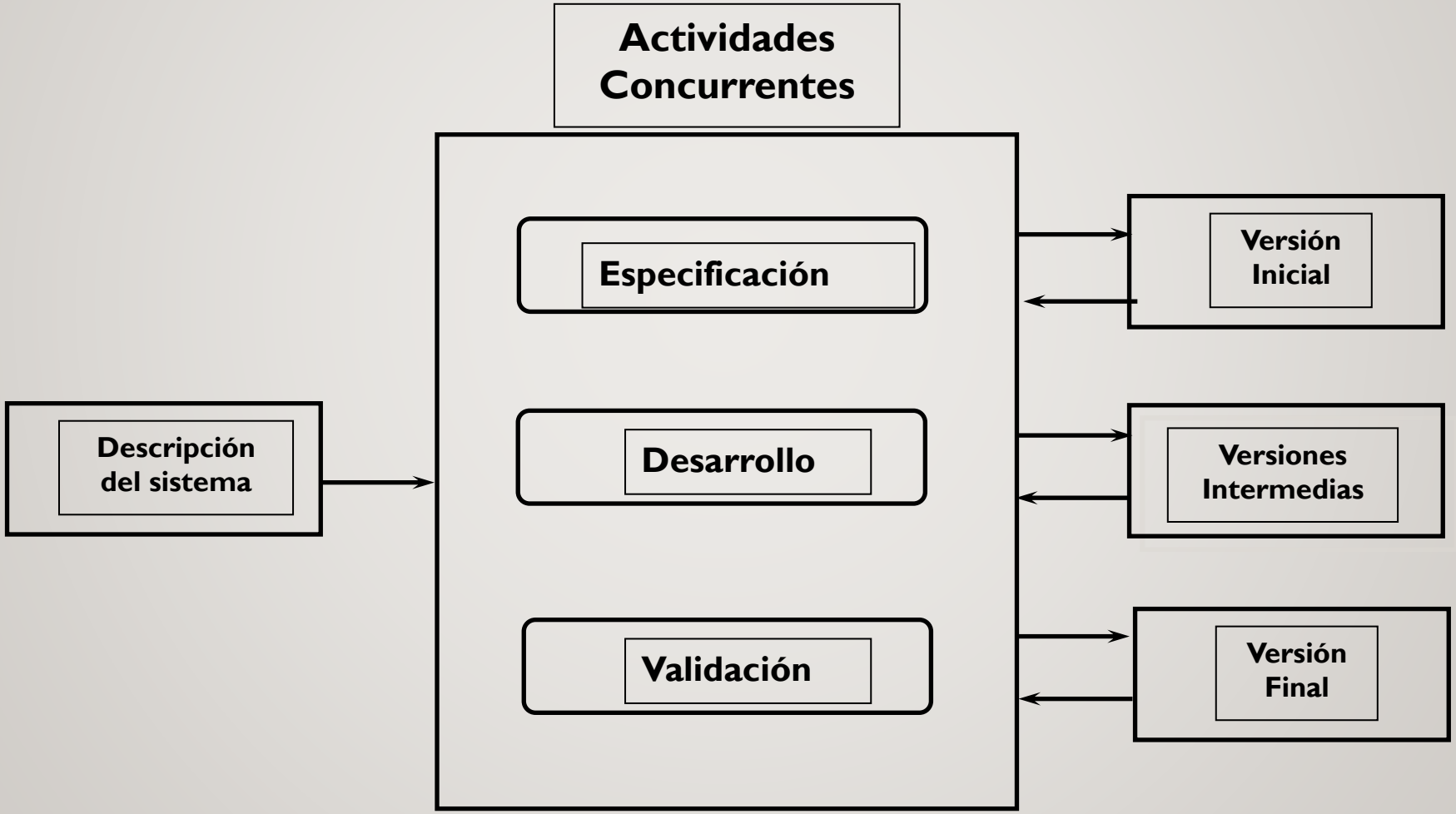
- Define actividades en **forma secuencial (mayor rigidez)**
- Se debe conocer con exactitud la información que maneja cada etapa.
- En la etapa de construcción e implementación se obtiene un producto visible.
- Se utiliza con muy baja frecuencia ya que en la actualidad los requerimientos pueden cambiar en cualquier etapa del desarrollo.

MODELO DE CASCADA

MODELO DE CASCADA

- Es criticado por tomar demasiado tiempo (entre la definición de requerimientos y la entrega del producto)
- Tiende a enfocar la atención internamente en vez de hacerlo con los clientes (usuarios finales) y guiarse en una excesiva documentación.
- No refleja el proceso "real" de desarrollo de software, por ejemplo cuando hay redefinición de requisitos en la fase de codificación.
- Acentúa los problemas con el usuario final

DESARROLLO EVOLUTIVO



DESARROLLO EVOLUTIVO

- Problemas
  - Poca visibilidad en el proceso
  - Los sistemas están pobremente especificados
  - Se requieren habilidades especiales.
- Aplicabilidad
  - Para sistemas interactivos pequeños o medianos.
  - Para partes de sistemas grandes (p.ej. la interfaz de usuario).
  - Para sistemas de corta vida.

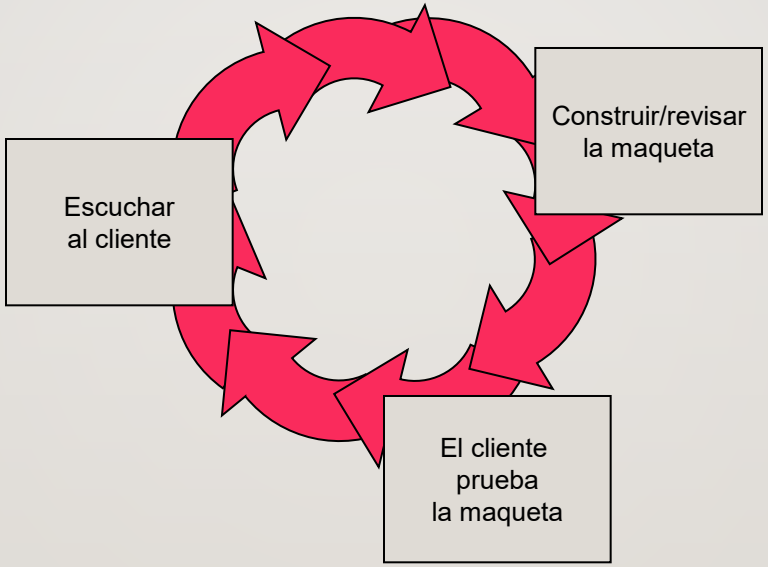
PROTOTIPADO

- **Modelo de construcción de prototipos**
  - Paradigma:
    - Inicia con la recolección de requisitos
    - Se hace un diseño rápido de aspectos visibles para el cliente/usuario para generar un prototipo.
    - El prototipo lo evalúa el cliente/usuario para refinar los requisitos.
  - Pudiera presentar problemas debido a:
    - El cliente solo ve el sistema por "fuera" y no la calidad por "dentro"; el mantenimiento no es prioridad cuando se desarrolla rápido.
    - El desarrollador, con frecuencia, hace uso de las herramientas más a la mano no de las más apropiadas.

PROTOTIPADO

- Prototipado exploratorio
  - El objetivo es trabajar con clientes hasta evolucionar a un sistema final, a partir de una especificación inicial. Se debe comenzar con unas especificaciones bien entendidas.
- Prototipado.
  - El objetivo es entender los requerimientos del sistema. Se puede comenzar con especificaciones poco entendidas.

PROTOTIPADOS



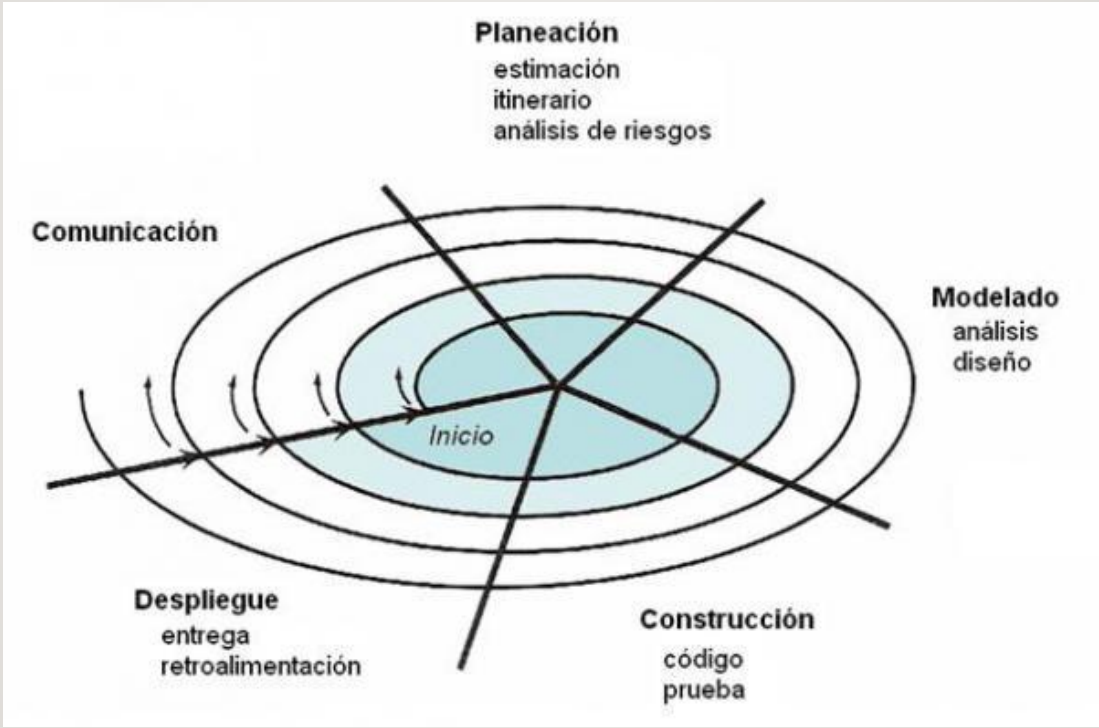
PROBLEMAS Y RIESGOS CON LOS MODELOS.

- Cascada.
  - Alto riesgo en sistemas nuevos debido a problemas en las especificaciones y en el diseño.
  - Bajo riesgo para desarrollos bien comprendidos utilizando tecnología conocida.
- Prototipado.
  - Bajo riesgo para nuevas aplicaciones debido a que las especificaciones y el diseño se llevan a cabo paso a paso.
  - Alto riesgo debido a falta de visibilidad
- Evolutivo.
  - Alto riesgo debido a la necesidad de tecnología avanzada y habilidades del grupo desarrollador.

MODELOS DE PROCESOS HÍBRIDOS

- Los sistemas grandes están hechos usualmente de varios subsistemas.
- No es necesario utilizar el mismo modelo de proceso para todos los subsistemas.
- El prototipado es recomendado cuando existen especificaciones de alto riesgo.
- El modelo de cascada es utilizado en desarrollos bien comprendidos.

MODELO DE PROCESO DE ESPIRAL BARRY W. BOEHM



MODELO DE PROCESO DE ESPIRAL

- Planteamiento de Objetivos
  - Se identifican los objetivos específicos para cada fase del proyecto.
- Identificación y reducción de riesgos.
  - Los riesgos clave se identifican y analizan, y la información sirve para minimizar los riesgos.
- Desarrollo y Validación.
  - Se elige un modelo apropiado para la siguiente fase del desarrollo.
- Planeación.
  - Se revisa el proyecto y se trazan planes para la siguiente ronda del espiral.

Metodologías de gestión de riesgos

La Metodología Espiral:

Considera aspectos **incrementales y evolutivos de desarrollo del producto, adicionando una variable de gestión de riesgos.**

El **enfoque en los riesgos** se debe a la ejecución de proyectos que en su esencia **son cambiantes, inestables o cambiantes los resultados esperados.**

PLANTILLA PARA UNA RONDA DEL ESPIRAL

- Objetivos.
- Restricciones.
- Alternativas.
- Riesgos.
- Resolución de riesgos.
- Resultados.
- Planes.
- Garantías (commitments).

VENTAJAS DEL MODELO DE ESPIRAL

- Centra su atención en la reutilización de componentes y eliminación de errores en información descubierta en fases iniciales.
- Los objetivos de calidad son primordiales.
- Integra desarrollo con mantenimiento.
- Provee un marco de desarrollo de hardware/software.

PROBLEMAS CON EL MODELO DE ESPIRAL

- El desarrollo contractual especifica el modelo del proceso y los resultados a entregar por adelantado.
- Requiere de experiencia en la identificación de riesgos.
- Requiere refinamiento para uso generalizado.

VISIBILIDAD DEL MODELO

Modelo de Proceso	Visibilidad del Proceso
Modelo de Cascada	Buena visibilidad, cada actividad produce un documento o resultado
Desarrollo Evolutivo	Visibilidad pobre, muy caro al producir docuementos en cada iteración.
Modelos Formales	Buena visibilidad, en cada fase deben producirse documentos.
Desarrollo orientado a la reutilización	Visibilidad moderada. Importante contar con documentación de componentes reutilizables.
Modelo de Espiral	Buena visibilidad, cada segmento y cada anillo del espiral debe producir un documento.

MODELOS RECIENTE: PROCESO UNIFICADO

La Metodología de Proceso Unificado o UP:

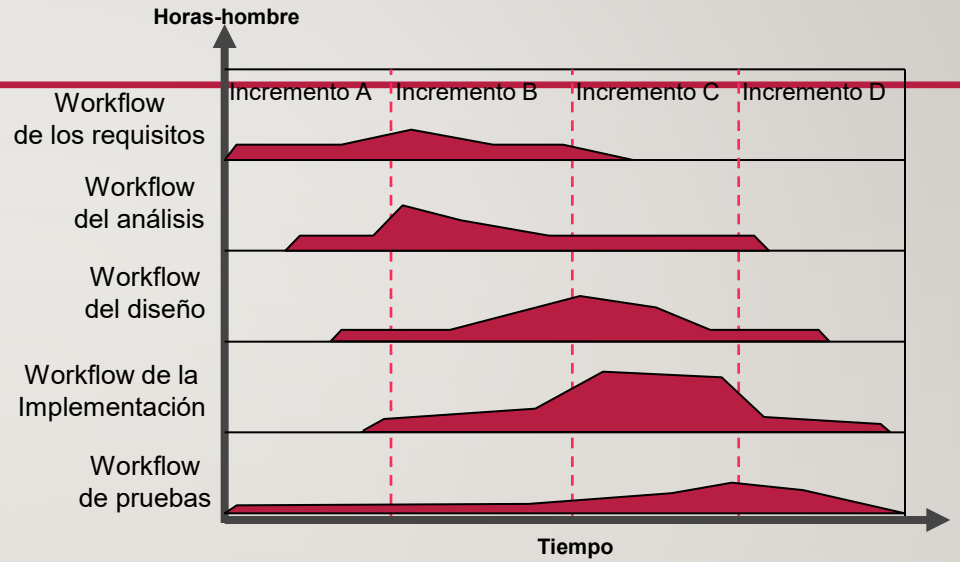
- Incorpora el concepto de **iteración** y el modo **evolutivo incremental** al cual adiciona el concepto de **ciclo de vida**.
- Centrada en los **casos de uso** y promueve el uso del modelamiento visual a través del **Lenguaje de Modelamiento Unificado UML**.
- Esta metodología se puede relacionar también con **RUP**.

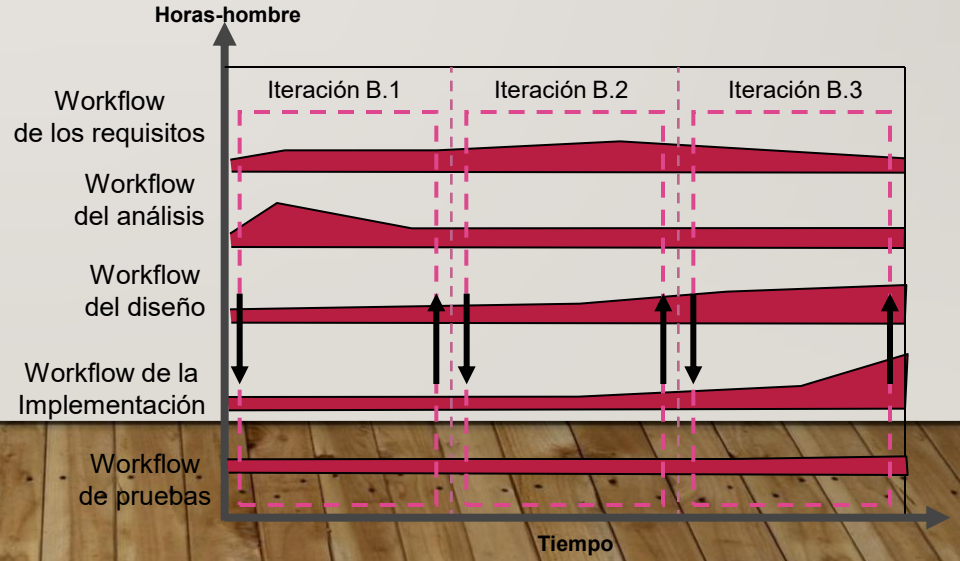
MODELOS RECIENTE: PROCESO UNIFICADO

- Características principales:
  - Dirigido por casos de uso
  - Centrado en la arquitectura
  - Iterativo e incremental
- Soporta el estándar UML
- Se deben repetir los cinco *workflow*.

MODELOS RECIENTE: PROCESO UNIFICADO

- Se plantea un desarrollo por incrementos aplicando *workflows* (flujo de trabajo) de requisitos, análisis, diseño, implementación y pruebas, presentes a lo largo del todo el ciclo de vida, sin embargo a veces un *workflow* predomina más sobre los otros cuatro.
- Las iteraciones se dan dentro de los incrementos, el número de estas varia dependiendo del incremento, y en cada iteración también se deben repetir los cinco *workflow*.



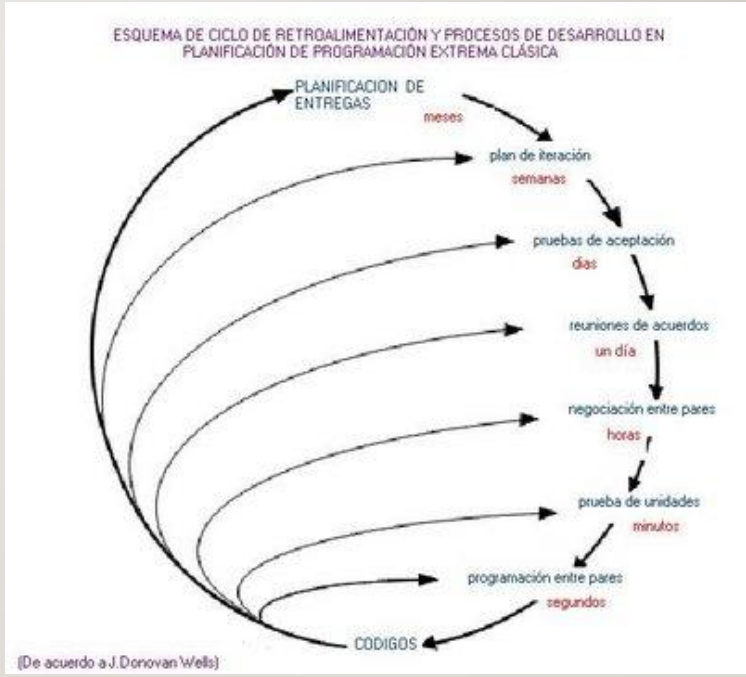


MODELOS RECIENTES: PROG. EXTREMA (UP)

- Disciplina de desarrollo de software creada por Kent Beck para proyectos cortos con requerimientos cambiantes o poco claros (respaldada por gran parte de la industria y rechazada por otro tanto).
- Se basa en la simplicidad, la comunicación y el reciclado continúo de código.
- Pretende:
  - La satisfacción del cliente
  - Potenciar al máximo el trabajo en grupo
  - Reducir el costo del cambio en las etapas de vida del sistema
  - Combinar las que han demostrado ser las mejores practicas de desarrollo de software y llevarlas al extremo.

MODELOS RECIENTES: PROG. EXTREMA (UP)

- Establece cuatro variables
  - Coste
  - Tiempo
  - Calidad
  - Ámbito
- Plantea cuatro valores
  - Comunicación
  - Sencillez
  - Retroalimentación
  - Valentía
- Define cuatro actividades básicas
  - Codificar
  - Hacer pruebas
  - Escuchar
  - Diseñar



La Metodología Extreme Programming XP:

- Define principios y prácticas para desarrollar software que promueve la **comunicación, simplicidad y la interacción entre el cliente y el desarrollador**.
- Las prácticas que define se orientan a una planificación simplificada, definición de historias de usuario y programación en pares.
- Genera versiones pequeñas y frecuentes de software a un ritmo sostenido.
- La aplicación de pruebas es constante y en conjunto con el cliente, se van logrando mejoras de diseño hasta lograr el producto que cliente necesita.

RESUMEN

- La Ingeniería de software concierne a las teorías, métodos y herramientas para el desarrollo, administración y evolución de productos de software.
- Los productos de software conformados por programas y documentación. Los atributos de los productos deben ser, mantenibilidad, confiabilidad, eficiencia y usabilidad.
- Un proceso de software consiste en aquellas actividades involucradas en su desarrollo.

RESUMEN

- El modelo de cascada considera cada actividad del proceso como una actividad discreta.
- El modelo de desarrollo evolutivo considera actividades del proceso en forma concurrente.
- El modelo de espiral se basa en análisis de riesgos.
- La visibilidad del proceso involucra la creación de documentos o resultados de las actividades.
- Los Ingenieros de software deben tener responsabilidades éticas, sociales y profesionales.

BUENA SEMANA…

MUCHAS GRACIAS