Análisis y Diseño de Sistemas

Tema: Ingeniería de Software

Urciuolo A.

UNPSJB – 2014

Temario

- Sistemas y Software.
- Necesidad de la Ingeniería de software
- Conceptos fundamentales de la Ingeniería de Software
- El Producto de software. Tipos
- Capas de la IS
- Fases de la IS
- El Proceso de desarrollo de software
- Concepto de Ciclos de vida
- Modelos de Proceso de Desarrollo de software

Sistemas automatizados

- Son sistemas hechos por el hombre y controlados por una o varias computadoras.
- Se componen de**:**

Hardware: CPU, discos, impresoras, etc.

Software: sistema operativos, bases de datos, programas de aplicación, etc.

Personas: proveen y/o consumen lo que produce el sistema.

Datos: información que se mantiene por período de tiempo.

Procedimientos: políticas e instrucciones para operar el sistema.

Documentación: manuales, formularios y otros modelos que describen en sistema.

El Software es parte de un sistema automatizado

Concepto de Software

- *Pressman:*

1. Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados
2. Estructuras de datos que permiten a los programas manipular adecuadamente la información, y
3. Documentos que describen la construcción y uso de programas

- *Sommerville:*

Programas de computadora y documentación asociada

Los productos de software pueden ser

- Productos genéricos.

  Desarrollados para clientes diversos.
- Productos hechos a medida.

  Desarrollados a pedido de un cliente particular según un requerimiento específico.

Importancia del SW

- Las economías de los países desarrollados y en vías de desarrollo dependen en gran parte del software.
- Cada vez más sistemas son controlados por software.
- El software se ha convertido en el "alma mater" de muchos sistemas productivos
- Está presente en sistemas de todo tipo: transportes, sanidad, telecomunicaciones, militares, procesos industriales, entretenimiento,etc.; cubriendo todas las actividades humanas
- El gasto en el desarrollo de Software, representa un alto porcentaje del PIB de todos los países

Problemas en el desarrollo del SW

- Particulares y compañías todavía desarrollan el software de forma muy arriesgada.
- Algunos profesionales siguen sin conocer o utilizar los métodos modernos lo cual afecta la calidad del software.
- Aplicaciones de software en situación crítica que presentan las siguientes características:

⇒ Aplicaciones escritas hace más de 20 años han sufrido varias generaciones de cambios.

⇒ Nadie tiene un conocimiento detallado sobre la estructura interna de las aplicaciones de ingeniería.

⇒ Sistemas empotrados a veces se comportan de forma inexplicable.

Problemas en el desarrollo del SW

- *La crisis del software se transformó en una "aflicción crónica" (Pressmann).*

- *Se deben afrontar los problemas, considerando los aspectos de fondo:*

- (1) La planificación y estimación de costos precisos.

- (2) La "productividad" de la comunidad de software en correspondencia con la demanda.

- (3) Calidad aceptable del software .

Algunas Dificultades en el SW

•Falta de metodología adecuada en el análisis de requisitos.

•Insatisfacción del cliente con el sistema terminado.

•Calidad del software cuestionable. Falta de adecuada importancia de la prueba sistemática y completa.

•La tarea de mantenimiento se lleva la mayor parte del dinero invertido en el soft.

"LA CLAVE PARA RESOLVER LOS PROBLEMAS ES DAR UN ENFOQUE DE INGENIERÍA AL DESARROLLO DEL SOFTWARE"

Porqué la Ingeniería de Software ?

- Necesidad de las empresas de producir productos de software más competitivos a mejores costos y con un fuerte componente de servicios post ventas.
- Se necesita cada vez más: Software *sin fallas* en tiempos reducidos y con *presupuestos preestablecidos.*

Porqué la Ingeniería de Software ?

- ***Consultores y académicos se deben esforzar por:***

➢ *Alinear objetivos del trabajo con tecnología.*
➢ *Poner énfasis en la formalidad del proceso de desarrollo.*
➢ *Trabajar con sistemas documentados.*
➢ *Realizar mediciones de cada actividad y análisis estadístico de los procesos.*
➢ *Madurez y mejora continua.*
➢ *Buscar soluciones integrales en paradigmas específicos.*

Qué es la Ingeniería de Software?

- La Ingeniería de Software (IS) es:
  * una disciplina de la Ingeniería
  * que comprende todos los aspectos de la producción de software:
    - Desde la especificación inicial al mantenimiento del sistema
    - Administración y gestión del proceso de producción
- Los Ingenieros de Software adoptan un enfoque sistemático para llevar a cabo su trabajo y utilizan los métodos, herramientas y técnicas necesarias para resolver el problema planteado.

Ingeniería de Software - Definiciones

"Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software", Glosario Standard de IEEE – 610-12 (1990).

"Disciplina tecnológica y de administración que se ocupa de la producción y evolución sistemática de productos de software que son desarrollados y modificados dentro de los tiempos y costos estimados", Richard Fairley

"Es la aplicación práctica de conocimiento científico al diseño y construcción de programas... y la documentación asociada requerida para su desarrollo, operación y mantenimiento", Barry Bohem.

IS y Ciencias de la computación

- La Ciencia de la Computación se refiere a las teorías y los fundamentos subyacentes en los sistemas de computación
- La Ingeniería del Software trata los problemas prácticos del desarrollo de software
- Con las teorías de la ciencia de la computación no es suficiente para desarrollar software (al menos cuando el sistema tiene suficiente envergadura)

IS e Ingeniería de Sistemas

- La Ingeniería de Sistemas concierne a todos los aspectos del desarrollo de sistemas basados en cómputo, que incluyen hardware, software y el proceso de Ingeniería. La Ingeniería de Software es solo parte de este proceso.
- IS define procesos para producir "sistemas de software".
- Al ser el software muchas veces la parte más importante del sistema, las técnicas de ingeniería del software se aplican en el proceso de ingeniería de sistemas.

Elementos que distinguen la IS

- ***producción y evolución.***
- ***métodos sistemáticos.***
- ***disciplina tecnológica y de administración.***
- ***cuantificación.***
- ***estimación de tiempos y costos.***
- ***desarrollo, operación y mantenimiento*.**

Ingeniería de Software: "Establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales" (Fritz Bauer).

Conocimientos del Ingeniero de SW

- Principios teóricos de representación y computación.
- Aplicación de métodos formales: lógica, matemática discreta, estadística, simulación.
- Uso de notaciones de modelización, especificación, diseño, programación.

- Los ingenieros de software deben adoptar un enfoque sistemático y organizado en su trabajo y utilizar las herramientas y técnicas más apropiadas dependiendo:
  - del problema a resolver,
  - las restricciones del desarrollo y
  - los recursos disponibles.

Conocimientos del Ingeniero de SW

La producción de software requiere de una combinación de recursos de metodología y tecnología.

Metodología: de documentación, análisis, especificación, diseño, implementación y prueba.
Tecnología: sistemas operativos, lenguajes, herramientas CASE, bases de datos, notaciones gráficas, sistemas generadores de interfaces, bibliotecas de código.

La producción de software requiere conocimiento de técnicas de administración de proyectos.

Administración de proyectos: planificación, análisis de riesgos, control de calidad, seguimiento de proyectos, control de subcontratistas, etc.

Disciplinas de la IS

- Requerimientos del software
- Diseño del software
- Construcción del software
- Prueba del Software
- Mantenimiento del software

- Gestión de la configuración del software
- Gestión de la Ingeniería del Software
- Proceso de Ingeniería del Software
- Herramientas y métodos de la Ingeniería del Software
- Calidad del software

Evolución de la Ingeniería de SW

La Ingeniería de Software es una familia de disciplinas técnicas y no técnicas que evoluciona continuamente.

Deriva sus principios, notaciones y técnicas de distintas ramas de la lógica, matemática, lingüística, administración y psicología.

Características del Proceso de Evolución

1) Expansión de su esfera de competencia.
2) Aumento en el nivel de formalización de procesos y productos.
3) Mecanización de operaciones.
4) Aumento en el nivel de conocimientos y profesionalismo que se exige de los ingenieros.

Evolución de la Ingeniería de SW

Década del 60

- Dificultades para desarrollar los primeros sistemas de software
- Los problemas desbordaban los recursos técnicos de una clase profesional formada por matemáticos, físicos, etc.
- Conferencias de la OTAN de carácter fundacional para la Ingeniería de Software (Garmish, 1968 y Roma, 1969) para buscar raíz de problemas.

Evolución de la Ingeniería de SW

Década del 70: Época de fundación y consolidación

- Se popularizaron los conceptos de: Diseño, modularidad, acoplamiento entre módulos, encapsulamiento de información.
- Se refinaron las técnicas de programación y diseño.
- Avanzaron los lenguajes de especificación y de programación.
- Se definieron "ciclos de vida" para los productos de SW
- Se reconoció la necesidad de definir técnicas de planificación y métodos de estimación cuantitativos.
- Se establecieron los métodos básicos de análisis de requerimientos y formalización de especificaciones de sistemas.
- Las técnicas de prueba de sistemas comenzaron a consolidarse sobre bases estadísticas.

Evolución de la Ingeniería de SW

Década de los 80:

- Se caracterizó por la expansión cultural y técnica: notaciones de especificación, lenguajes de programación disponibles, variedad de modelos, técnicas de análisis y diseño.
- Se comenzó a utilizar en forma corriente el término "Ingeniero de Software".
- Se multiplicaron los Congresos científicos.
- Se desarrolló una industria editorial específica y se incorporaron cursos en las carreras universitarias y postgrados.
- Surgieron los ambientes de programación y herramientas especializadas, los editores de notaciones gráficas, sistemas de bibliotecas, correo electrónico, etc.
- Se consolidó la industria CASE.
- Se popularizaron las métricas y técnicas de estimación y de

Evolución de la Ingeniería de SW

Década de los 90:

- ***Transformaciones:*** *de la programación artesanal a la producción masiva, del individuo (como constructor de sistemas) a la organización, de los métodos manuales a los métodos asistidos por computadora*.
- El software pasa a ser parte integral de toda clase de productos y servicios.
- La industria de software se ve sometida a presiones: reducción en el ciclo de vida de productos, personalización de productos, costos, calidad y certificación de procesos.
- Necesidad de garantizar seguridad, confiabilidad o la responsabilidad legal por daños y perjuicios ocasionados por un producto.
- Se agrega la preocupación por formalizar, medir y optimizar el proceso de desarrollo y los controles de calidad.

Evolución de la Ingeniería de SW

Década de los 00:

- Aplicación de buenas prácticas genéricas para su adaptación a problemas particulares (procesos de desarrollo y ciclos de vida genéricos y flexibles, lenguajes de modelado, técnicas de modelado basadas en el uso de patrones y frameworks,…)
- Se consolida la gestión del software a través de planificación de proyectos, estimaciones y gestión de recursos, aseguramiento de la calidad, gestión de la configuración y de documentación,..
- Extenso desarrollo de la ingeniería de requerimientos
- Desarrollo rápido de aplicaciones
- Reuso de software
- Desarrollo basado en componentes.
- Desarrollo de aplicaciones web, frameworks de aplicaciones.
- …

El Producto: Software

- La IS produce **productos de software**
- Un producto de software es un sistema de software distribuido a un cliente junto con su documentación.

  - programas de computadora,

  - procedimientos,

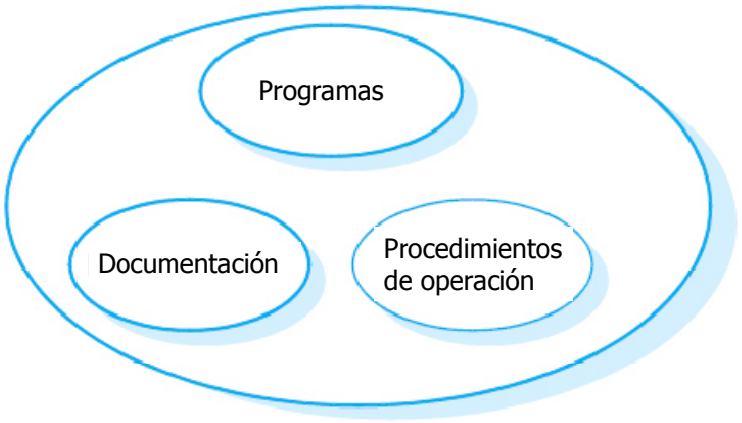
  - documentación asociada,

  - datos relativos a la operación de un sistema de cómputos..
- Los productos de software se clasifican:

  - ***Software a medida***: software desarrollado para un cliente particular bajo un contrato.

  - ***Software genéricos***: desarrollados para ser vendidos a un mercado abierto.

El Producto: Software



Componentes del Software

El Producto: Software (s/Pressman)

- Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados
- Estructuras de datos que permiten a los programas manipular adecuadamente la información
- Documentos que describen la construcción y uso de programas

Características del Producto SW

- A diferencia de otros productos de ingeniería el software es un producto particular.

  - El software *es lógico* y no físico.

  - El software *es maleable*.

  - El software *se desarrolla* no se fabrica. Su construcción requiere principalmente de *creatividad humana*, más que de manufactura

  - El software *no se estropea* (aunque se vuelve obsoleto).

  - Aunque la industria tiende a ensamblar componentes, aún la mayor parte del software que se construye es a medida

Software bien diseñado - Atributos

- Mantenible

  Capaz de evolucionar según las necesidades de cambio de los clientes
- Seguro

  No produce daños incluso bajo un fallo del sistema
- Eficiente

  No desperdicia los recursos del sistema (memoria, procesador, disco)
- Amistoso

  Buena interfaz
- Bien documentado

Aplicaciones de SW

- Software de sistemas: Colección de Programas de servicio a otros programas

  - Compiladores, Sistemas Operativos (SOs), etc.
- Software de tiempo real: Monitorea, analiza y controla eventos del mundo real a medida que ocurren

  - Control de aviones, pronóstico del clima, etc.
- Software empotrado (*embedded systems*): Reside en la ROM de un producto y controla sus funciones. Maneja componentes de hardware.

  - El producto puede ser un sistema de seguridad, señalización, etc.

Aplicaciones de SW

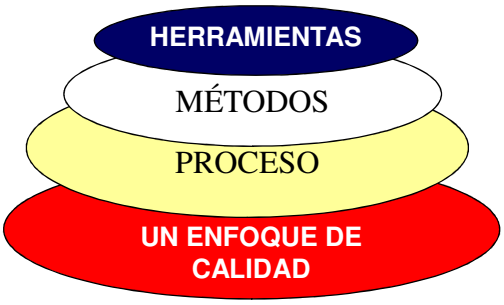
- Software de negocios: Diseñado para procesar aplicaciones de negocios. Herramientas de manejo de bases de datos.
  - Gestión de nóminas, control de almacén, etc.
- Software de ingeniería y científico: Basado en Algoritmos numéricos con altas necesidades de cómputos.
  - Programas CAD, MATLAB, predicción meteorológica, etc.
- Software de PC: Se venden en la gran distribución para computadoras personales.
  - Procesadores de texto, hojas de cálculo, etc.

Aplicaciones de SW

- Software de inteligencia artificial: Algoritmos no numéricos para resolver problemas complejos en forma declarativa, para los que no son adecuados el cálculo o análisis directo
  - Sistemas expertos
  - Reconocimiento de patrones (voz, imágenes, etc.)
  - Agentes software
- Software WEB: Desarrollado para aplicaciones WEB.
  - e-commerce, correo electrónico, etc..

Capas de la IS

- Capa de enfoque de calidad
- Capa de proceso
- Capa de métodos
- Capa de herramientas



Ingeniería de SW como tecnología multicapa.

Capas de la IS

- Capa de enfoque de calidad: Base de procesos de ingeniería.
  - La IS se basa en calidad >> mejores técnicas para construcción de SW
- Capa de proceso: Conjunto de actividades y resultados asociados que sirven para construir un producto SW.
- Capa de métodos: Forma en que se construye el SW
  - Los métodos corresponden a: análisis de requisitos, diseño, construcción de programas, prueba,….
- Capa de herramientas: Soporte automático o semiautomático para el proceso y los métodos. Herramientas CASE

Proceso

***Proceso de desarrollo de software**:*
Marco de trabajo de las tareas que se requieren para construir software de alta calidad.

La Ingeniería de software abarca un conjunto de tres elementos clave que se utilizan en todo el proceso: *métodos*, *herramientas* y *procedimientos*

Elementos clave del SW - Métodos

Los métodos indican cómo construir técnicamente el software

Incluyen métodos para:
- planificación y estimación del proyecto,
- análisis de los requerimientos del sistema y del software,
- diseño de estructuras de datos,
- arquitectura de programas y procedimientos algorítmicos,
- codificación, prueba y mantenimiento.

Elementos clave del SW - Herramientas

Las *herramientas* suministran un soporte automático para los métodos.

Existen herramientas para soportar cada uno de los métodos.

Herramientas CASE: Cuando se integran las herramientas, de modo que la información creada por una herramienta, pueda ser utilizada por la otra, se establece un soporte para el desarrollo del software, llamado "ingeniería de software asistido por computadora" (CASE).

CASE combina software y bases de datos para crear un entorno de ingeniería de software**.**

Visión general de la IG - Fases

El trabajo que se asocia a la Ingeniería de software se puede dividir en tres fases genéricas, con independencia del modelo de proceso, área de aplicación, tamaño o complejidad del proyecto:

- definición
- desarrollo
- mantenimiento

Fase de definición: El qué

Se identifican los requrimientos del sistema y del software*:*

- *qué información ha de ser procesada,*
- *qué función y rendimiento se desea,*
- *qué comportamiento del sistema,*
- *qué interfaces se establecerán,*
- *qué restricciones de diseño y*
- *qué criterios de validación.*

Fase de definición: El qué

Tareas principales:

Ingeniería de sistemas: se define el papel de cada elemento de un sistema informático, asignando al software el papel que va a desempeñar.

Planificación del proyecto de software: Establecido el ámbito del software, se analizan los riesgos, se asignan los recursos, se estiman los costos, se definen las tareas y se planifica el trabajo.

Análisis de requerimientos: Se realiza una síntesis del ámbito de información y de las funciones del SW.

Fase de desarrollo: El cómo

Se define:

- Cómo diseñar las estructuras de datos
- Cómo organizar la arquitectura del software,
- Cómo implementar las funciones (detalles procedimentales),
- Como realizar la traducción del diseño a un lenguaje de programación,
- Cómo se realizarán las pruebas.

Fase de desarrollo: El cómo

Tareas principales:

Diseño del software: Traduce los requisitos del software a un conjunto de representaciones (gráficas, tabulares, incluso lenguajes) que describen la estructura de datos, la arquitectura, los procedimientos algorítmicos y las características de la interfaz.

Generación de código: Durante este paso, las representaciones de diseño son traducidas a un lenguaje, ya sea procedimental o no procedimental, dando como resultado instrucciones ejecutables para la computadora.

Prueba del software: Una vez que el software ha sido implementado, debe ser probado, para descubrir los defectos que puedan existir en la función, en la lógica y en la implementación.

Fase de mantenimiento: *El cambio*

Va asociado:

- a la corrección de errores,
- a las adaptaciones requeridas por la evolución del entorno del software
- a las modificaciones debidas a los cambios de requisitos del cliente dirigidos a readecuar el sistema.

Fase de mantenimiento: *El cambio*

Corrección: Es muy probable que el cliente descubra defectos en el soft; *el mantenimiento correctivo se refiere al "cambio" para corregir defectos.*

Adaptación: Con el paso del tiempo suele cambiar el entorno original para el que se desarrolló el soft; *el mantenimiento adaptativo consiste en modificar el soft para adecuarlo a los cambios de su entorno externo.*

Mejora: El cliente puede descubrir a través del tiempo, funciones adicionales que podría interesar incorporar al software; *el mantenimiento perfectivo amplía el software más allá de sus requisitos funcionales originales.*

Prevención: El mantenimiento preventivo, llamado reingeniería del soft hace cambios en los programas a fin de que se puedan mejorar más fácilmente.

Actividades de soporte

Estas fases se complementan con las *actividades de soporte*
-No crean software
- Mejoran su *calidad*
- Facilitan su desarrollo
Se aplican a lo largo de todo el proceso del software

Ejemplos de actividades *de soporte*
- Documentación
- Gestión de configuración
- Seguimiento y control del proyecto de software
- Revisiones técnicas formales
- Garantía de la calidad del software
- Gestión de reutilización
- Mediciones
- Gestión de riesgos.

Proceso de Desarrollo de SW

Proceso: Serie de operaciones usadas en la creación de un producto.

- Proceso de Software: Conjunto de estructurado de actividades que tienen que ser realizadas para producir un producto de software de alta calidad

- Proceso de Software (vinculado al concepto de Ciclo de vida): Proceso que se sigue para construir un producto de software desde la concepción de una idea, hasta la entrega y el retiro final del sistema.

Actividades del PDSW

En un Proceso genérico de SW (independientemente del modelo) se pueden distinguir las siguientes actividades:

Análisis de requerimientos.
Especificación
Diseño arquitectural.
Diseño de programas.
Implementación de programas.
Validación y verificación.
Entrega del sistema.
Mantenimiento.

Relaciones entre actividades >>> Modelos de Desarrollo

Actividades del PDSW

- Se describen en forma independiente, indicando rol y resultados
- Utilizan y producen documentos
- Se relacionan e interactúan de diferentes maneras conformando distintos modelo de procesos de desarrollo de software
- De acuerdo al modelo, una actividad puede jugar un rol preponderante o incluso pudiera no existir

Análisis de requerimientos

Identificar el problema
Recolectar los requerimientos
Involucrar usuarios

Sintetizar y Documentar requerimientos funcionales y no funcionales

Análisis de requerimientos

Documentos orientados al usuario y útiles para el analista:

DRS
ERS

Definición de Requerimientos SW
Especificación de Requerimientos SW

Especificación del sistema de SW

Requerimientos del software
Restricciones
Consideraciones técnicas

ERS

Modelado de Funcionalidad y Datos del SW

Especificación del sistema SW

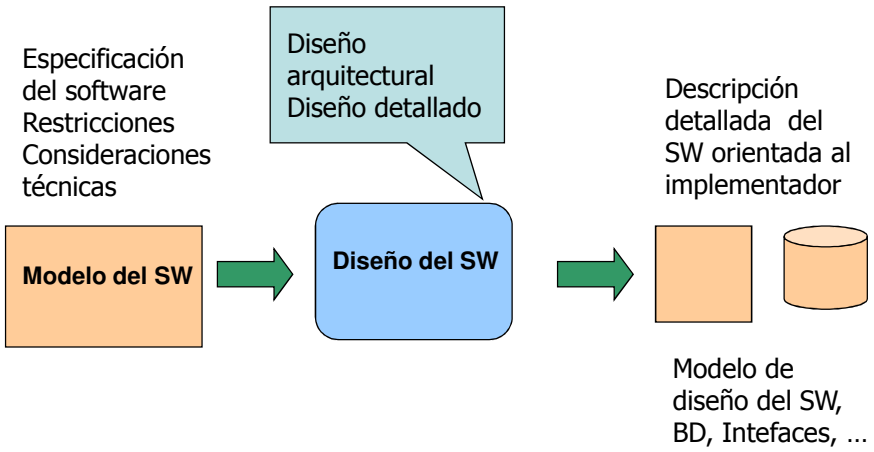
Descripción orientada al desarrollador

Modelo del SW

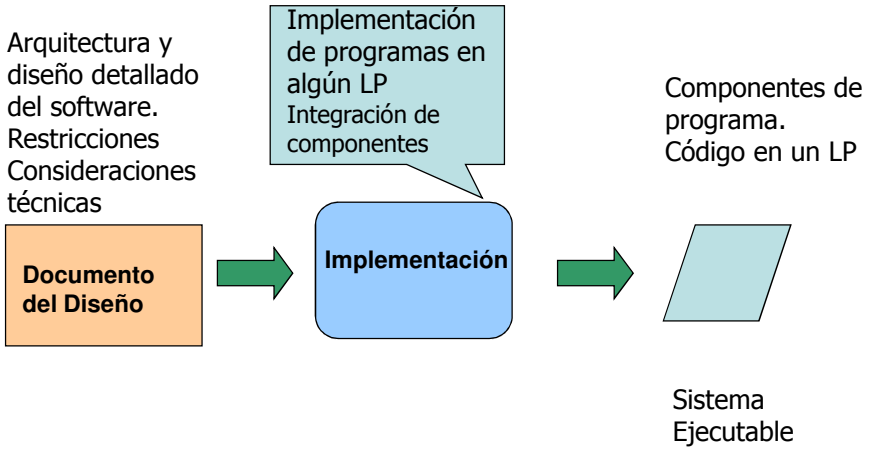
Describe el qué y no el cómo

Input al Diseño

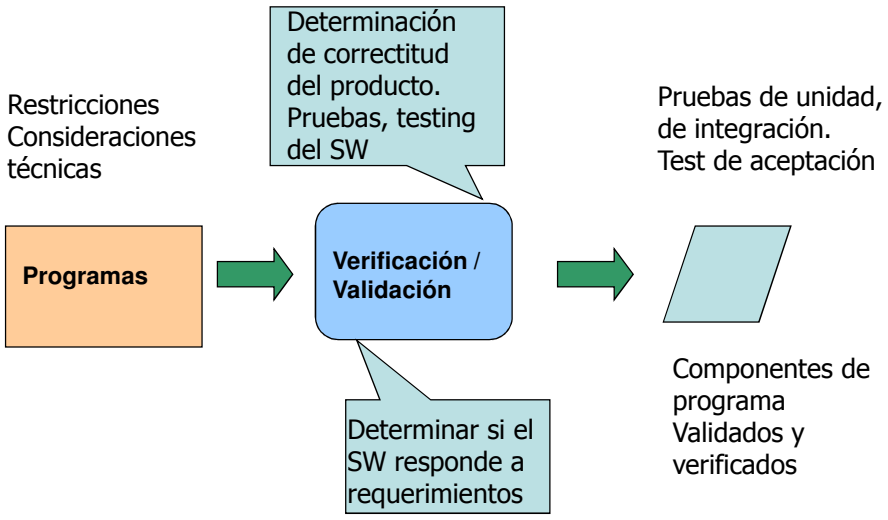
Diseño del SW



Implementación



Verificación y validación



Concepto de ciclo de vida del SW

- **Ciclo de vida:** Periodo de tiempo que comienza al concebir la idea de un nuevo sistema de software, y termina cuando este se retira y deja de funcionar.
- **Proceso** es un conjunto de actividades y tareas relacionadas, que al ejecutarse de forma conjunta transforman una entrada en una salida.
- Un modelo de proceso, o paradigma de IS, es una plantilla, patrón o marco que define el proceso a través del cual se crea software
- *Se utilizan ambos términos (proceso y ciclo de vida) para definir los modelos disponibles.*

Concepto de ciclo de vida del SW

Sirven para:

- Definir las actividades a llevarse a cabo en un proyecto de desarrollo de SW.
- Lograr congruencia entre los distintos proyectos de desarrollo de SW en una misma organización.
- Proporcionar puntos de control y revisión de las decisiones sobre continuar con el proyecto o no.

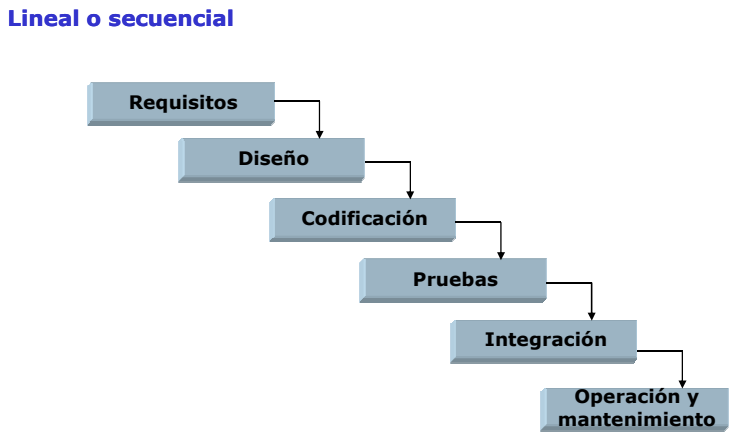
El ciclo de vida puede organizar las actividades de administración del proyecto, aumentando las probabilidades de que se aborden los temas en el momento adecuado.

Modelos de Desarrollo de SW

- Define la estructura de un proceso de desarrollo racional y controlable
- No existe un modelo universal
- Los modelos no son rígidos
- Son una guía respecto al orden en que deben adelantarse las actividades
- Se basa en el reconocimiento que el software tiene un ciclo de vida

- Modelos secuenciales: Lineal, Cascada, RAD
- Modelos evolutivos: Incremental, Espiral, basado en reuso, Prototipos
- Modelos formales: Transformacional

Modelos Lineal o Secuencial (1970)



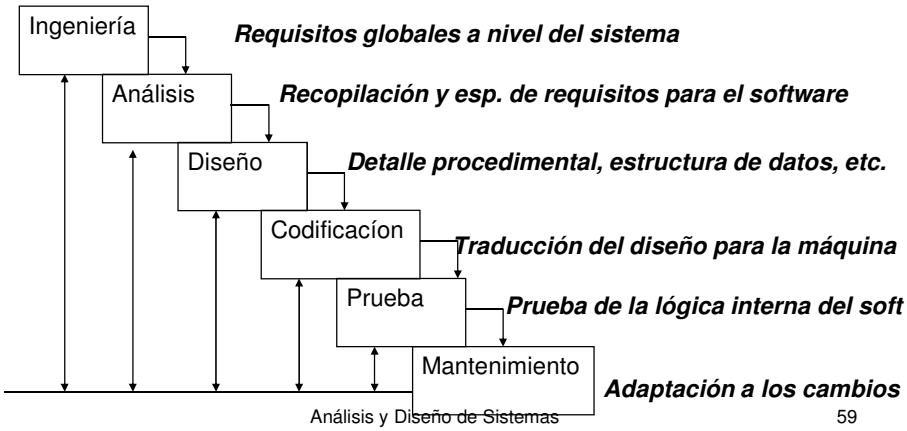
Modelos Lineal o Secuencial

- ✓ Propuesto por Royce en 1970.
- ✓ Este modelo refleja un desarrollo marcado por la sucesión escalonada de las etapas que lo componen.
- ✓ Es necesario terminar por completo cada etapa para pasar a la siguiente
- ✓ Resulta muy rígido porque cada fase requiere como elemento de entrada el resultado completo de la anterior.
- ✓ Resulta apropiado para:
  - ✓ Sistemas pequeños, sin previsión de evolución a corto plazo.

- ✓ El modelo prácticamente idéntico, que evita esta rigidez es el de cascada, que se expone a continuación.

Modelo en cascada (Boehm, 1981)

Sigue un enfoque ascendente y secuencial y una insistencia en la progresión lineal del desarrollo del software que progresa a través de las siguientes etapas:



Modelo en cascada

Dificultades del enfoque ascendente:

- ✓ No hay nada para mostrar al usuario hasta que todo esté terminado.
- ✓ Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final.
- ✓ La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema.
- ✓ La necesidad de prueba con la computadora aumenta exponencialmente durante las etapas finales de prueba.

Modelo en cascada

Dificultades de la progresión secuencial:

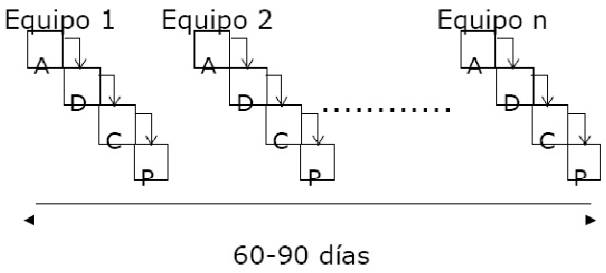
- ✓ El deseo de progreso ordenado de las etapas no es nada realista.
- ✓ Durante el tiempo que dura el proyecto, el usuario podría cambiar de parecer respecto a lo que debe hacer el sistema.
- ✓ Se apoya en técnicas anticuadas.
- ✓ Resulta apropiado para sistemas pequeños

Modelo RAD
Desarrollo rápido de aplicaciones

- Basado en el modelo lineal secuencial, con ciclos de desarrollo extremadamente cortos (90 días).
- Llevado a cabo por varios equipos de trabajo que siguen las etapas del proceso de manera simultánea. Cada equipo maneja una parte del sistema.
- Candidatos: sistemas que se pueden modularizar => equipos de desarrollo paralelos.
- Amplia participación del usuario en todas las etapas.
- Enfoque de construcción basado en el uso de componentes reutilizables y Herramientas de cuarta generación (generadores de código, herramientas de prueba automatizadas, etc.).

Modelo RAD

- Adaptación a "alta velocidad" del modelo lineal secuencial.
- Equipos trabajando en paralelo aplicando tecnología de componentes.



Modelo RAD

- Para proyectos grandes, es necesario contar con recursos suficientes para formar los equipos necesarios.
- Compromiso de colaboración entre desarrolladores y clientes para un tiempo de espera corto.
- No todas las aplicaciones son susceptibles de aplicar este modelo (no modularizables, bajo reuso de componentes).
- No aplicable cuando:
  - Sistemas con gran mantenimiento.
  - El grado de interoperatividad con programas ya existentes es alto

Modelos Evolutivos

- Características:
  - Gestionan bien la naturaleza evolutiva del software
  - Se adaptan más fácilmente a los cambios introducidos a lo largo del desarrollo.
  - Son iterativos: construyen versiones de software cada vez más completas

- Se adaptan bien:
  - Los cambios de requisitos del producto
  - Fechas de entrega estrictas poco realistas
  - Especificaciones parciales del producto

- Modelos evolutivos:
  - Modelo en Espiral
  - Modelo Incremental
  - Modelo de Desarrollo Basado en reuso de Componentes

Modelo en espiral (Bohem, 1988)

Actividades principales

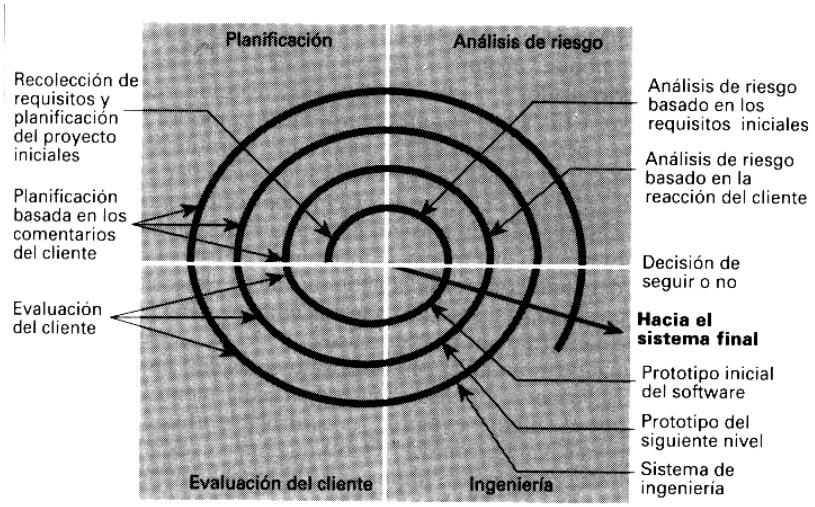
1) **Planificación:** determinación de objetivos, alternativas y restricciones. Incluye:
Recolección de requisitos y planificación del proyecto iniciales (1a vuelta).
Planificación basada en los comentarios del cliente (vueltas siguientes).

2) **Análisis de riesgos:** análisis de alternativas e identificación /resolución de riesgos. Incluye:
Análisis de riesgos basado en los requisitos iniciales (1a. vuelta).
Análisis de riesgos basados en la reacción del cliente (2a. vuelta).
Se toma la decisión de seguir adelante o no.

3) **Ingeniería:** Desarrollo del producto de "siguiente nivel".
Se construye un prototipo cero (inicial del software) y se va refinando en los niveles sucesivos.

4) **Evaluación del cliente:** valoración de los resultados de la ingeniería en cada etapa.

Esquema del Modelo en espiral



Con cada iteración alrededor de una espiral (comenzando en el centro) se construyen sucesivas versiones del SW

Esquema del Modelo en espiral

Durante la primera vuelta:

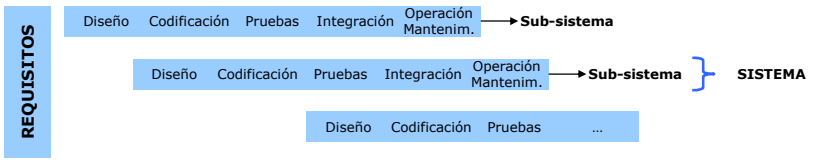
- Se definen los objetivos, las alternativas y las restricciones,
- Se analizan e identifican los riesgos,
- El cliente evalúa el trabajo de ingeniería y sugiere modificaciones.

En base a los comentarios del clientes se produce la siguiente fase

- Planificación y análisis de riesgos. (en cada bucle de la espiral este análisis se traduce en una decisión de "seguir adelante o no").
- En la mayoría de los casos, se sigue avanzando alrededor del espiral, con el prototipo del siguiente nivel (ingeniería) y la evaluación del cliente; ese camino lleva a un modelo completo del sistema.

Modelo incremental

Incremental



- El modelo incremental mitiga la rigidez del modelo en cascada, descomponiendo el desarrollo de un sistema en partes; para cada una de las cuales se aplica un ciclo de desarrollo.

Las ventajas que ofrece son:

- ✓ El usuario dispone de pequeños subsistemas operativos que ayudan a perfilar mejor las necesidades reales del conjunto.
- ✓ El modelo produce entregas parciales en periodos cortos de tiempo y permite la incorporación de nuevos requisitos que pueden no estar disponibles o no ser conocidos al iniciar el desarrollo.

Modelo incremental

Es un modelo cuyas etapas consisten de *incrementos expandidos de un producto de software operativo*.

Cada iteración devuelve un "**Incremento**" del SW

- Un incremento es un producto **operacional del SW**.
- El primer incremento suele ser un núcleo básico desarrollado en base a los requerimientos centrales.
- Muchas funciones suplementarias se dejan para después
- Se evalúa (p.ej., por el cliente) el producto entregado
- Como resultado se desarrolla un plan para el incremento siguiente

Modelo incremental

Ventajas

- Es interactivo
  Con cada incremento se entrega al cliente un producto
  operacional al cliente, que puede evaluarlo
- Personal
  Permite variar el personal asignado a cada iteración
- Gestión riesgos técnicos
  Por ejemplo, disponibilidad de hardware específico

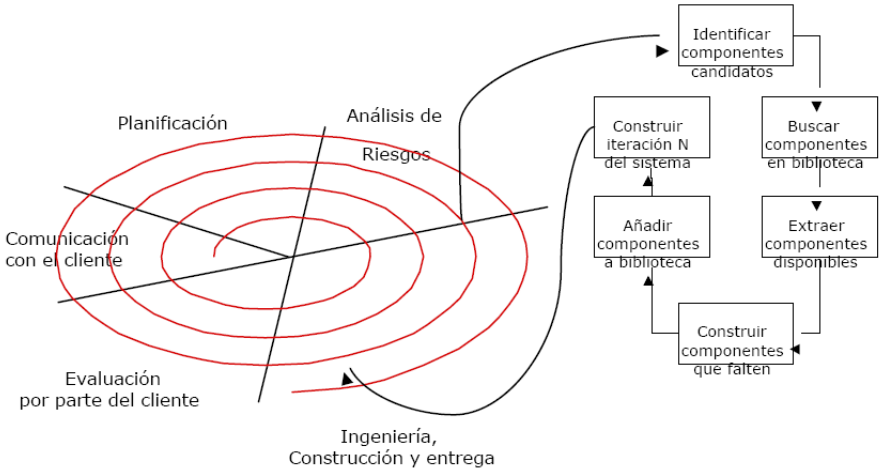
Inconvenientes

La primera iteración puede plantear los mismos problemas

Modelos Evolutivos: Ensamblaje de comp.

- El modelo de ensamblaje de componentes es un espiral de Boston adaptado al uso de componentes software reutilizables
- Ventajas: Reuso de componentes.
- Inconvenientes: Encontrar los componentes adecuados.

Modelos Evolutivos: Ensamblaje de comp.



Modelo de prototipos

Facilita al programador la creación de un modelo de SW.
Puede considerarse un modelo en sí mismo

El prototipado consiste en la construcción de modelos de prueba, que simulen el funcionamiento que se pretende conseguir en el sistema.

Esta forma de trabajo previo suele tener como principal objetivo la experimentación con un entorno similar al pretendido, para obtener retro-información del usuario o cliente que ayuda a los desarrolladores en la concreción de los requisitos.

Modelo de prototipos

✓Los prototipos pueden ser:

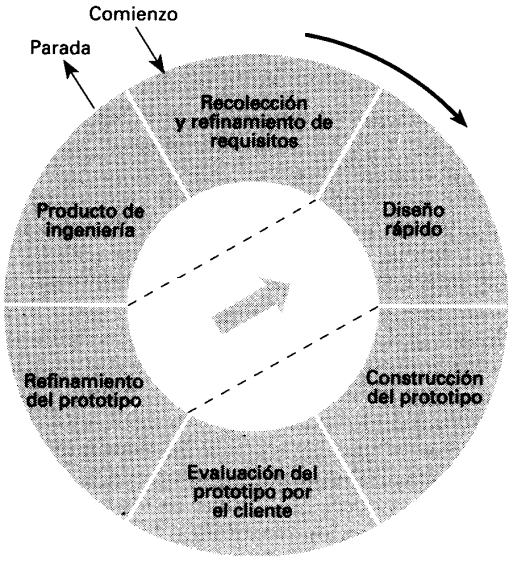
- **Ligeros:** dibujos de pantallas de interfaz con simulación de funcionamiento por enlaces a otros dibujos…

- **Operativos:** Módulos de software con funcionamiento propio que se desarrollan sin cubrir las funcionalidades completas del sistema, normalmente en entornos RAD (rapid application development").

Modelo de prototipos.

- ✓ La construcción de prototipos comienza con la recolección de requisitos.
  - El analista y el cliente se reúnen, definen los objetivos globales para el software, identifican los requisitos.
- ✓ Luego se produce un "diseño rápido" que conduce a un prototipo.
  - El prototipo es evaluado por el cliente / usuario y se utiliza para refinar los requisitos del software a desarrollar.
- ✓ El prototipo es refinado sucesivamente en un proceso interactivo para que satisfaga las necesidades del cliente.
- ✓ Generalmente este primer prototipo llamado "prototipo cero" se tira, porque es lento o grande o difícil de usar.

Modelo de prototipos



77

Modelo de prototipos

Desventajas

- ✓ El cliente ve *el prototipo y lo confunde con el sistema real*. No entiende la necesidad de volver a hacerlo.
- ✓ El equipo de desarrollo toma *decisiones rápidas* para poder construir el prototipo, que más tarde *son difíciles de revertir* (por ejemplo el lenguaje de programación).

Modelo transformacional

- ✓ Basado en *especificaciones formales*: el desarrollo de software como una secuencia de pasos que gradualmente *transforma* una especificación en implementación hasta un programa ejecutable
- ✓ Se deben transformar requerimientos informales en especificaciones formales*: métodos de especificación formal.*
- ✓ Pretende reducir errores automatizando pasos del desarrollo. Las transformaciones preservan la corrección.

Modelo transformacional

Modelo transformacional

- ✓ Problemas
  - Hace falta una formación especializada para aplicar la técnica
  - Muchos aspectos de los sistemas reales son difíciles de especificar formalmente
    - Interfaz de usuario
    - Requisitos no funcionales
- ✓ Aplicabilidad
  - Sistemas críticos en los que la seguridad y fiabilidad debe poder asegurarse antes de poner el sistema en operación

Bibliografía

- ✓ "Fundamentals of Software Engineering". Carlo Guezzi.
- ✓ "Ingeniería del Software" – Un enfoque Práctico" R.Pressman.
- ✓ "Software Engineering" . Ian Sommerville.