Módulo 1: Introducción

Diseño y Desarrollo de Software (*1er. Cuat. 2019*)

Profesora titular de la cátedra:
Marcela Capobianco

Profesores interinos:
Sebastián Gottifredi
Gerardo I. Simari

Licenciatura en Ciencias de la Computación – UNS

Licencia

- Copyright ©2019 Marcela Capobianco.
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la GNU Free Documentation License, Version 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera.
- Una copia de esta licencia está siempre disponible en la página http://www.gnu.org/copyleft/fdl.html

¿Qué es el software?

- Producto que diseñan y desarrollan los ingenieros de software.
- Abarca los programas, los documentos y otros formatos en los que se almacene "*información*" (videos, bases de datos, etc.).
- **Objetivo:** desarrollar técnicas que hagan *más efectiva y menos costosa* la construcción de software *de calidad*.

Evolución del software

- Doble rol: producto final y vehículo de acceso.
- Mayor complejidad del hardware → Mayor complejidad de los sistemas → Mayor desafío para los desarrolladores.
- Internet produjo un cambio de filosofía.
- Computación omnipresente (ej., Y2K)
- Expansión constante: conexión entre casas, oficinas, autopistas, etc

Programador → Equipo

Algunas interrogantes principales:

- ¿Por qué lleva tanto tiempo desarrollar los programas?
- ¿Por qué son tan elevados los costos?
- ¿Por qué no podemos encontrar todos los errores?
- ¿Por qué nos resulta difícil constatar el progreso mientras se desarrolla el SW?

Características del SW

- El SW se *desarrolla*, no se fabrica en un sentido clásico.
- El SW no se *estropea* (pero se deteriora con los cambios).
- La mayoría del SW se construye a *medida* (pero se tiende al desarrollo con componentes).

Crisis del SW

- Existen numerosos ejemplos de estrepitosas fallas en desarrollos.
- Los sistemas de software son creaciones **complejas**.
- Es difícil que una sola persona pueda *comprenderlos completamente*.
- Están sujetos a constantes cambios.

Máquinas para votar en Florida

At Kings Point in suburban Delray Beach, voters encountered delays after a poll worker mistakenly shut down the voting machines:

"*They were on and then someone mistakenly, accidentally turned them off*," Anderson said while visiting H.L. Johnson Elementary School in Royal Palm Beach. "*Allegedly someone tripped over a line or something like that.*"

This will have resulted in lost votes and once again goes to show that the machines are unreliable and detrimental to the democratic voting process.

Toyota Prius recall

(8/2/2010) Toyota announced it will conduct a voluntary safety recall on approximately 133,000 2010 Model Year Prius vehicles and 14,500 Lexus Division 2010 HS 250h vehicles to update software in the vehicle's anti-lock brake system (ABS). The ABS, in normal operation, engages and disengages rapidly (many times per second) as the control system senses and reacts to tire slippage. Some 2010 model year Prius have reported experiencing inconsistent brake feel during slow and steady application of brakes on rough or slick road surfaces when the ABS is activated in an effort to maintain tire traction.

Aeropuerto de Denver

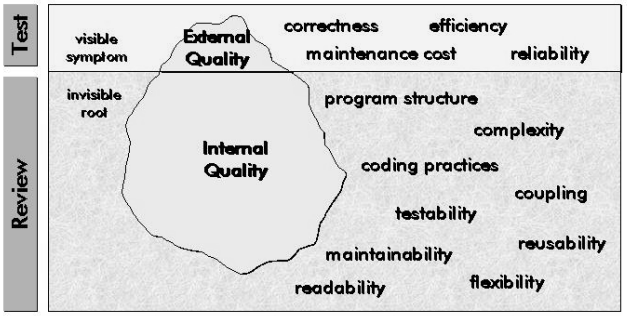
- En 1995, el nuevo aeropuerto de Denver pretendió desarrollar un sistema para el manejo automático del equipaje.
- Estaría compuesto por 4.000 "telecars" que toman equipaje de 20 aerolineas, 100 computadoras en red, 56 escáners y 5.000 "ojos eléctricos".
- El sistema tenía un presupuesto de 193 millones de dólares.

Aeropuerto de Denver

- El sistema nunca llegó a estar operativo.
- Retrasó la inauguración del aeropuerto por meses a un costo estimado de *1 millón de dólares por día*.
- Se intentó agregar más gente al proyecto y no hubo ninguna mejora (*ley de Brooks**).
- El aeropuerto tuvo que abrir sus puertas con un sistema manual.

(*) Postulada por Fred Brooks en 1975: "*Agregar recursos humanos a un proyecto que va atrasado hace que se atrase más.*"

The Software Quality Iceberg



Costo del Cambio

La crisis del software: Resumen

- El software se ha transformado en un elemento clave de la gran mayoría de los negocios (*businesses*).
- Existen problemas con el desarrollo del mismo que surgen del hecho de que la disciplina es aun relativamente jóven en comparación con otras.
- El SW es hoy en día un factor que limita la evolución de los sistemas.
- SW = programas + datos + documentos

El proceso de desarrollo: ¿*Qué es*?

- Serie de pasos *predecibles* que lleve a producir un SW de *calidad*.
- Es importante porque otorga estabilidad, control y organización a una actividad que de lo contrario puede volverse caótica.
- El proceso depende del SW que estamos construyendo, y en general es adaptado por el líder del proyecto.

Ingeniería de SW

Definición (IEEE): La ingeniería del SW es:

1) La aplicación de un enfoque *sistemático, disciplinado* y *cuantificable* hacia el desarrollo, operación y mantenimiento del SW; es decir, la aplicación de ingeniería al SW.
2) El estudio de enfoques como en (1).

Etapas de la ingeniería de SW

Existen tres fases genéricas, independientemente del proyecto particular

- Fase de definición: ¿*Qué*?
- Fase de desarrollo: ¿*Cómo*?
- Fase de mantenimiento: *Cambios*

Tipos de cambios:

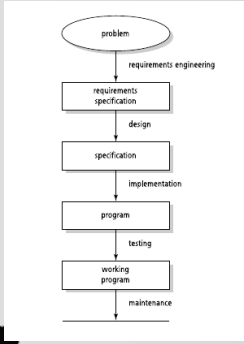
- Correctivos
- Adaptivos
- Mejoras
- Preventivos

Modelos del proceso (*repaso*)

- Estrategia de desarrollo (también llamado *paradigma* de Ingeniería de SW).
- Su elección depende del proyecto, los métodos y las herramientas.
- También depende de los controles y entregas.
- Son un intento de ordenar una actividad inherentemente caótica, y exhiben características del "*modelo del caos**".

(*) **Caos:** Propiedad de un sistema complejo cuyo comportamiento es tan impredecible que parece aleatorio. En realidad, éste es un reflejo de la gran sensibilidad a pequeños cambios en las condiciones.

Una visión simple



- No hay superposición de etapas
- No permite vuelta atrás
- No considera el mantenimiento
- Sin embargo, sirve para analizar los elementos básicos

Ingeniería de requerimientos

- *Descripción del problema a resolver*.
- Es literalmente un proceso fundamental.
- Requerimientos establecidos *por* el ambiente y *sobre* el ambiente en que funcionará el sistema.
- Incluye el estudio de factibilidad.
- Documento entregable: *especificación de requerimientos*.

Diseño

- Se desarrolla un *modelo* del sistema.
- El modelo consiste de *componentes* e *interfaces*.
- *Early decisions*: descripción arquitectónica.
- Documento entregable: *especificación técnica*.

Implementación

- Nos concentramos en los componentes individuales.
- Muchas veces es necesario introducir una fase de diseño adicional.
- Se busca un resultado confiable, flexible, correcto, legible, ...
- Resultados (entregables): ejecutable, código, documentación adicional.

Testing

- *No es necesariamente una fase que sigue a la implementación.*
- Se compone de dos actividades*:
  - Verificación: ¿El sistema hace lo que *debe*?
  - Validación: ¿El sistema hace lo que el *usuario pidió*?
- Consume más tiempo que toda las fases anteriormente nombradas.

(*) En inglés se plantea el juego de palabras:

- Verification: A*re we building the product right*?
- Validation: *Are we building the right product*?

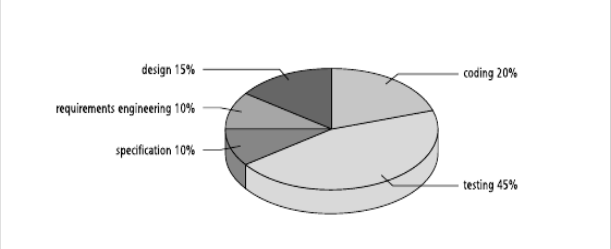
Mantenimiento

El mantenimiento puede tener por lo menos dos fines:

- Corrección de errores.
- Cambios y mejoras.

El propósito es mantener al sistema operacional después de que ha sido puesto en funcionamiento.

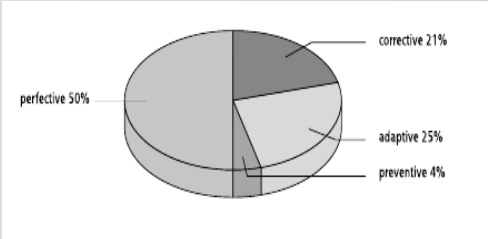
Porcentaje de esfuerzo por etapas



¡El mantenimiento consume entre un 50% y un 75% del esfuerzo total!

Mantenimiento

Tipos de mantenimiento (o *evolución*)



Algunos casos reales más

Ariane 5, Flight 501 (1996)

- 500 millones de dólares en pérdidas materiales.
- Error de *overflow* al convertir un número de 64 bits punto flotante a un entero signado de 16 bits.

Algunos casos reales más

- Sistema de ambulancias de Londres (1992): administra las ambulancias, decide cuál enviar, verifica las llamadas y minimiza los tiempos de respuesta a los incidentes.
- El sistema se atrasó más de 8 meses y finalmente, cuando se pudo poner en funcionamiento, falló desastrosamente.
- El *crash* se debió a un error en la administración de la memoria. Pero, ¿fue eso lo que hizo fallar el proyecto?

Responsabilidades

- La Ingeniería de Software comprende responsabilidades que van mas allá de la simple aplicación de habilidades técnicas.
- Para ser profesionales respetables, los ingenieros de software deben comportarse en forma *honesta y respetando principios éticos*.
- Un comportamiento ético es mucho más que simplemente “respetar la ley”.

Responsabilidad profesional

- *Confidencialidad*: los ingenieros deben respetar la confidencialidad de sus empleados o clientes, independientemente de si se firmó o no un contrato al respecto.
- *Competencia*: No se deben aceptar trabajos que estén fuera del área de competencia.
- *Derecho de propiedad intelectual*: se deben conocer los derechos de propiedad, patentes, etc. Se debe también proteger la propiedad intelectual de los empleados y clientes.

Responsabilidad profesional

- *Uso incorrecto de las computadoras*: los ingenieros de SW no deben usar sus habilidades técnicas para hacer un uso incorrecto de las computadoras de otras personas.
- ¿Qué ejemplos se le ocurre de uso incorrecto?

Responsabilidad profesional

- *Uso incorrecto de las computadoras*: los ingenieros de SW no deben usar sus habilidades técnicas para hacer un uso incorrecto de las computadoras de otras personas.
- Ejemplos de esto varían desde usar las computadoras para jugar hasta infectar las computadoras con *malware* o acceder a datos privados.

Código de ética del ACM

- Sociedades profesionales de los EE.UU. han cooperado para producir este código.
- Los miembros de estas organizaciones automáticamente se adhieren al código al ingresar.
- El código contiene 8 principios que involucran a distintos roles profesionales de la Ingeniería de SW (ingenieros, educadores, supervisores, estudiantes).

Preámbulo

- La *versión corta* del código resume las aspiraciones en un alto nivel de abstracción.
- Las cláusulas en la *versión completa* dan detalles y ejemplos.
- Un análisis en detalle de ambas partes permite comprender el código en forma completa.

Preámbulo

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

Principios

1) *Interés Público*: Los Ingenieros de SW actuarán en forma consistente con el interés público.
2) *Cliente y empleado*: Los Ingenieros de SW actuarán de manera acorde con los mejores intereses de su empleador y su clientes, consistentemente con el interés público.
3) *Producto*: los productos deben alcanzar los estándares más altos posibles.

Principios

4) *Juicio*: Los Ingenieros de SW mantendrán integridad e independencia en el juicio profesional.
5) *Gestión*: los directores de proyectos deben suscribirse a un manejo ético del desarrollo y mantenimiento de SW.
6) *Profesión*: Los Ingenieros de SW harán progresar la integridad y reputación de la profesión consistentemente con el interés público.

Principios

7) *Colegas*: Los Ingenieros de SW serán justos y proveerán soporte a sus colegas.
8) Los Ingenieros de SW participarán en un aprendizaje de por vida de su profesión y promoverán un acercamiento ético a la práctica de la misma.

A modo de ejercicio, veamos algunas situaciones ficticias en las que resulta evidente que no siempre es fácil "*actuar éticamente*"...

Situación 1

- Usted está en desacuerdo con las políticas de la empresa o dirección.

Situación 2

- Un ingeniero no realiza de forma adecuada el test de un sistema crítico.

Situación 3

- Le ofrecen participar en un proyecto del gobierno de un país para desarrollar armas de uso militar.

Situación 4

- Un profesional que trabaja en una institución determinada (*por ejemplo una universidad X*) ofrece sus servicios a una empresa Y para realizar trabajos dentro de la universidad X.

Situación 5

- Un ingeniero de SW desarrolla un programa para mensajería que al instalarse envía información de la computadora del usuario a un determinado servidor.

Situación 6

- Debe recomendar a un cliente que maneja una pequeña empresa qué software usar para su PC recién adquirida, sabiendo que el costo del software comercial que se usa actualmente es prohibitivo.

Tareas a realizar

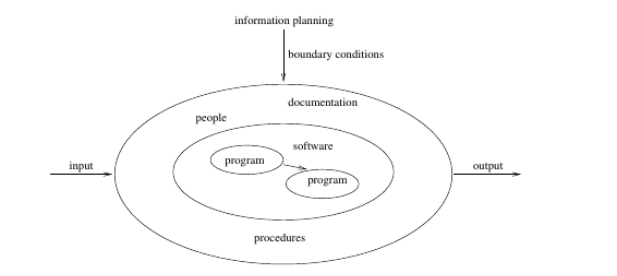
Leer el código de ética del consejo profesional de nuestra provincia (*Consejo Nacional de Ciencias Informáticas de la Provincia de Buenos Aires* – CPCIBA) y comparar con el código del ACM.

Proyectos

Continuando con el repaso sobre el proceso de desarrollo del software, recordemos que:

- Los aspectos *organizacionales y gerenciales* son tan importantes como los técnicos.
- Los productos se desarrollan en un *contexto*.
- No desarrollamos sólo software, desarrollamos un *sistema*.

Proyectos



Planificando un proyecto

Hay diferentes aspectos que entran en juego:

- Concepción
- Modelo de proceso
- Organización del proyecto
- Estándares, *guidelines*, procedimientos
- Actividades de administración
- Riesgos
- Equipo

Planificando un proyecto

Hay diferentes aspectos que entran en juego (*cont.*):

- Métodos y técnicas
- *Quality assurance* (QA)
- Recursos
- Work packages (WBS: "*Work Breakdown Structure*")
- Presupuesto y tiempos
- Cambios
- Entrega

Controlando el proceso

Varios factores tienen influencia sobre el proceso:

- Tiempo disponible
- Información
- Organización
- Calidad esperada
- Dinero disponible

El "*triángulo del alcance*" es una buena metáfora para la situación de compromiso que se da entre *alcance* y (1) recursos, (2), tiempo, (3) calidad

Madurez del proceso de SW

El SEI* ha desarrollado un modelo de los diferentes grados de madurez del proceso de una organización (cada nivel incluye al anterior):

1) *Inicial*: el proceso se caracteriza según el caso. Se definen pocos procesos. El éxito depende del esfuerzo individual.
2) *Repetible*: Se establecen los procesos de gestión para hacer un seguimiento del costo, planificación y funcionalidad. Sirve para repetir éxitos anteriores.

(*) SEI: *Software Engineering Institute* (https://www.sei.cmu.edu/)

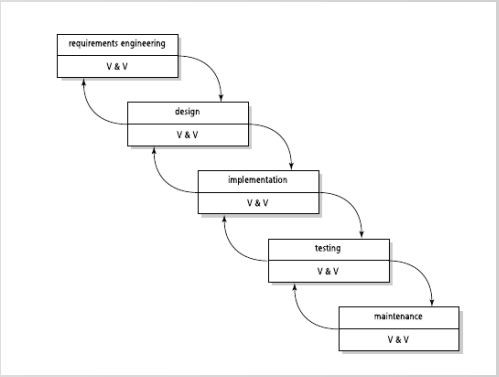
Madurez del proceso de SW

4) *Definido*: El proceso de SW (gestión e ingeniería) se *documenta* y *estandariza*. Todos los proyectos usan una versión aprobada y documentada del proceso de la organización.
5) *Gestionado*: Se recopilan *medidas* detalladas del proceso del SW y la calidad del producto.
6) *Optimizado*: Se *mejora* el proceso mediante la retroalimentación del proceso, ideas y tecnologías.

Modelo Lineal Secuencial

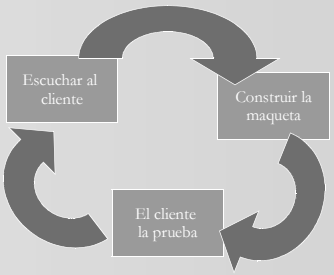
Enfoque sistemático y secuencial:

Modelo en cascada

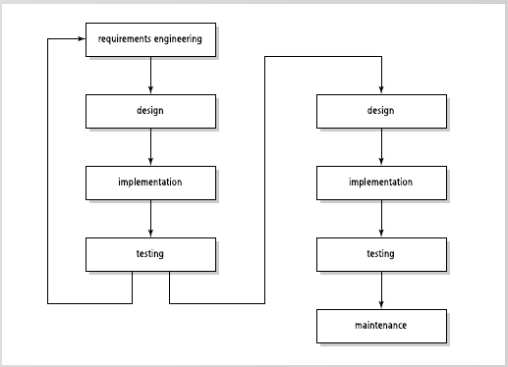


Modelo de Prototipos

Una alternativa para cuando el cliente no tiene claros los requisitos:



El prototipo como un modelo



Modelo DRA: Desarrollo Rápido de Aplicaciones

Modelo lineal secuencial que enfatiza un *ciclo de desarrollo corto*; comprende las siguientes fases:

- Modelado de Gestión
- Modelado de Datos
- Modelado del Proceso
- Generación de aplicaciones
- Pruebas y entrega

Por las limitaciones de tiempo, la aplicación debe cumplir el requisito del "*ámbito en escalas*": cada componente puede completarse en menos de tres meses y desarrollarse independientemente.

Modelos evolutivos

- Modelo para un producto que evoluciona con el tiempo.
- Permiten desarrollar versiones cada vez más completas del software.
- Existen distintas alternativas, entre ellas:
  - Modelo incremental: modelo secuencial + prototipo
  - Modelo espiral
  - Modelo espiral WinWin: Agrega una fase de *negociación* entre las partes antes de cada vuelta del espiral.
  - Modelo de desarrollo concurrente

Desarrollo basado en componentes

- El paradigma de objetos dio un gran impulso a este modelo hace alrededor de dos décadas.
- Evolutivo por naturaleza.
- Se basa en identificar las clases candidatas y luego elegir las que ya existen en la librería para reutilizarlas.
- Las componentes que no estén disponibles se desarrollan.

Modelo de métodos formales

- Especificación matemática del SW de computadoras.
- Se especifica, desarrolla y verifica mediante una notación formal.
- Una variante es la "*ingeniería de sala limpia*".
- *Problema*: aplicabilidad al entorno de gestión por cuestiones de tratabilidad computacional y experticia requerida.
- Uso en sistemas de extrema seguridad.

Desarrollo Ágil

- Necesitamos desarrollar reglas de comportamiento para mejorar la construcción del software y que el proceso puede adaptarse ágilmente ante los cambios.
- Las reglas deben poder adaptarse y evolucionar de acuerdo con la necesidad de cada proyecto.
- *Cada técnica tiene sus limitaciones*.
- Continuaremos con este tema en el Módulo 3 del curso.

Bibliografía

- *Software Engineering: A Practitioner's Approach.* R. Pressman, 8th ed., 2015. Capítulos 1 y 2.
- *Software Engineering: Principles and Practices*. Hans van Vliet, 3rd ed., 2008. Capítulo 1 (Suplementarios: 2 y 3).
- *Código de ética ACM/IEEE*. Disponible en: `https://ethics.acm.org/code-of-ethics/software-engineering-code/`
- *Código de Ética Profesional*. Consejo Profesional de Ciencias Informáticas de la Provincia de Buenos Aires. Disponible en: `http://www.cpciba.org.ar/archivos/documentos/codigo_etica_profesional.pdf`