

# Ingeniería del Software I

1 - Introducción (Capítulo 1)

# Dominio del problema

- Si tuvieran que desarrollar un programa de 10.000 líneas de código: ¿cuánto tardarían?
- Respuesta típica de un alumno: 2 a 4 meses  
Productividad: 2.5 a 5 KLOC por persona/mes
- ¿Cuál es la productividad típica en una organización de desarrollo de software?  
100 a 1000 LOC por PM
- ¿A qué se debe la diferencia?

# Dominio del problema

Software (IEEE 610.12-1990):

Colección de programas, procedimientos, y la documentación y datos asociados que determinan la operación de un sistema de computación.

# Dominio del problema

- Alumno: El desarrollador es el usuario.  
El error ("bug") es tolerable.  
La interfaz con el usuario no es importante.  
No existe documentación.
- Industria: Los usuarios son otros.  
El error no es tolerable.  
La interfaz con el usuario es muy importante.  
Se requiere documentación tanto para el usuario como para la organización y el proyecto.

# Dominio del problema

- Alumno: El software no tiene ningún uso crítico.  
La confiabilidad y la robustez no son importantes.  
No existe inversión (que perder).  
La portabilidad no es importante.
- Industria: Soporta funcionalidad/información importante.  
La confiabilidad y la robustez son fundamentales.  
Existe una fuerte inversión.  
La portabilidad es una característica clave.

# Dominio del problema

## Software de nivel industrial

- La diferencia clave con el software pensado por un alumno y el de nivel industrial radica en la calidad (incluyendo usabilidad, confiabilidad, portabilidad, etcétera).
  - Alta calidad requiere mucho testing, que consume entre 30 y 50% del esfuerzo de desarrollo total.
  - Requiere la descomposición en etapas del desarrollo de manera de poder encontrar "bugs" en cada una de ellas (cada uno de diferente índole).
  - Para la misma funcionalidad se incrementa el tamaño y los requerimientos: buena interface, backup, tolerancia a fallas, acatar estándares, etcétera.

# Dominio del problema

## Software de nivel industrial

- Contando  $1/5$  de productividad de un "estudiante" y la duplicación del tamaño del producto, el software de nivel industrial requiere un esfuerzo 10 veces mayor.
- El dominio de la Ingeniería de Software es precisamente el software de nivel industrial.
- En este curso llamaremos "software" sólo al software de nivel industrial.

# Dominio del problema

## El software es caro

- El principal costo en la producción de software es la mano de obra.
- Costos involucrados (estimados en Córdoba):
  - Productividad = 500 LOC / PM
  - Costo de la compañía  $\approx$  80K / PM (D\$3k)
  - $\Rightarrow$  Costo por línea  $\approx$  \$150
- Una simple aplicación administrativa puede tener entre 20 y 50 KLOC.
  - $\Rightarrow$  Costo  $\approx$  \$3000 K a \$7500 K
  - corriendo en hardware de entre \$80K y \$250K
  - $\Rightarrow$  En las soluciones de TI, los costos de hardware son pequeños comparados con los costos de software.

(Es el bruto, incluyendo miles de gastos además del salario del programador)

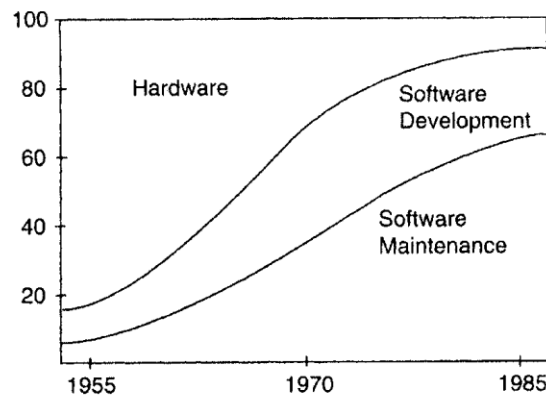
Notar la relación



# Dominio del problema

## El software es caro

- La relación hardware/software en el costo de un sistema de computación se incrementó significativamente desde sus comienzos:



- Conclusión: El software es **muy** caro, por lo que es importante optimizar el proceso de desarrollo con el fin de abaratar el costo del software.

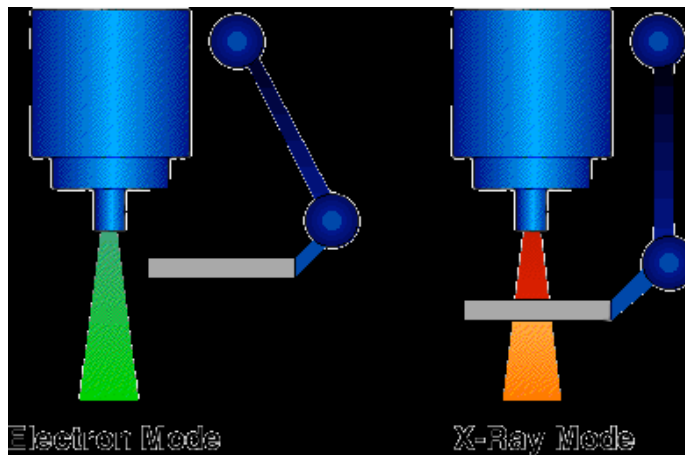
# Dominio del problema

Demorado y poco confiable

- La IS es aún un área débil, a pesar de su progreso.
- 35% de los proyectos de software categorizan como “desbocados” (en inglés: runaway):  
Tienen el presupuesto y el costo fuera de control.
- Además la producción de software poco confiables es muy alta:  
El 70% de todas las fallas de los equipamientos eran de software.

# Dominio del problema

Poco confiable



Therac-25:  
2 soft. errors  
(-3 personas)

# Dominio del problema

Demorado y poco confiable

- Muchos más bugs en:
  - The risk digest: <http://catless.ncl.ac.uk/Risks/>
- Otros:
  - [www.cs.tau.ac.il/~nachumd/verify/horror.html](http://www.cs.tau.ac.il/~nachumd/verify/horror.html)
  - [www5.in.tum.de/~huckle/bugse.html](http://www5.in.tum.de/~huckle/bugse.html)
  - [en.wikipedia.org/wiki/List\\_of\\_software\\_bugs](http://en.wikipedia.org/wiki/List_of_software_bugs)

# Dominio del problema

Demorado y poco confiable

- Las fallas de software son distintas de las fallas mecánicas o eléctricas.
- En software, en general, las fallas **no** son consecuencia del uso y el deterioro.
- Las fallas ocurren como consecuencia de errores (o "bugs") introducidos **durante** el desarrollo.
- Es decir: **la falla que causa el problema existe desde el comienzo**, sólo que se manifiesta tarde.

# Dominio del problema

## Mantenimiento

- Una vez entregado, el software requiere mantenimiento.
- ¿Por qué es necesario el mantenimiento si el software no se deteriora con el uso?

Para corregir errores residuales (updates)

=> mantenimiento correctivo.

Para mejorar funcionalmente el software (upgrades) y adaptarlo a los cambios de entorno

=> mantenimiento adaptativo.

- Durante la vida de un software, el mantenimiento puede costar más que el desarrollo.

# Dominio del problema

## Mantenimiento

- Una vez entregado, el software requiere mantenimiento.

- ¿Por qué es necesario el mantenimiento?  
El software se deteriora con el uso?

Para corregir errores residuales (bugs)  
=> mantenimiento correctivo.

Para mejorar funcionalmente el software  
cambios de entorno

=> mantenimiento adaptativo.

Incluye la comprensión del software existente (código y documentación), comprensión de los efectos del cambio, realización de los cambios (código y doc.), testear lo nuevo y re-testear lo viejo.

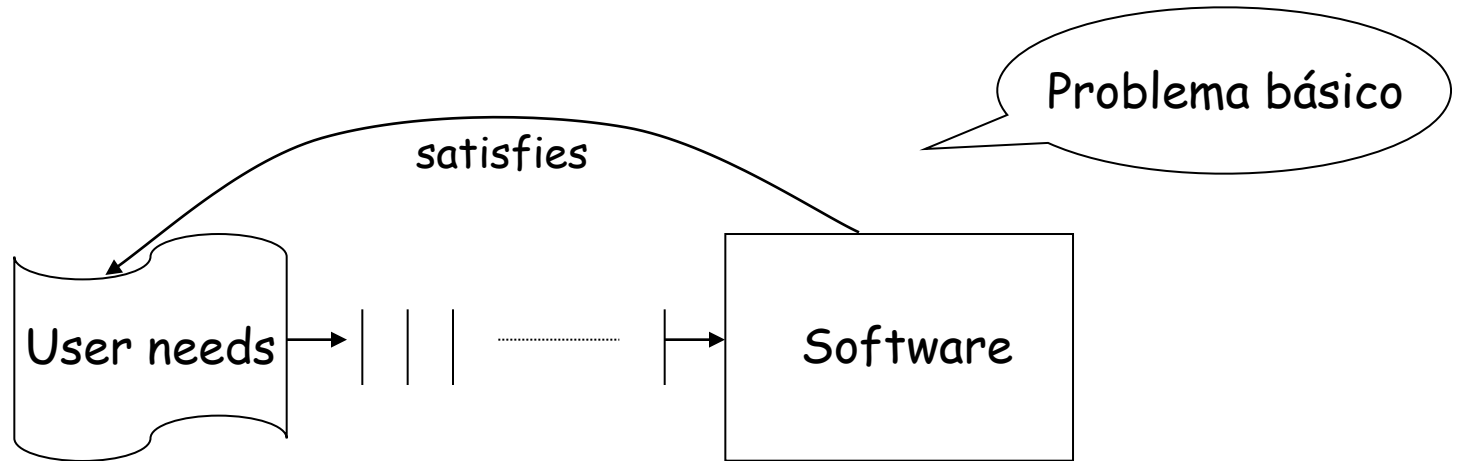
- Durante la vida de un software, el mantenimiento puede costar más que el desarrollo.

# Desafíos de la IS

- Ingeniería de Software: aplicación de un enfoque sistemático, disciplinado, y cuantificable al desarrollo, operación, y mantenimiento del software.
- Enfoque sistemático: metodología y prácticas existentes para solucionar un problema dentro de un dominio determinado.  
Esto permite repetir el proceso y da la posibilidad de predecirlo (independientemente del grupo de personas que lo lleva a cabo).



# Desafíos de la IS



- El problema de producir software para satisfacer las necesidades del cliente/usuario guía el enfoque usado en IS.
- Pero hay otros factores que tienen impacto en la elección del enfoque:  
Escala, Calidad, Productividad, Consistencia, Cambios,...

# Desafíos de la IS

## Escala

- IS debe considerar la escala del sistema a desarrollar  
Los métodos utilizados para desarrollar pequeños problemas no siempre escalan a grandes problemas.  
Ej.: contar alumnos en un aula vs. censo nacional.
- Los métodos de IS deben ser escalables.
- Dos claras dimensiones a considerar:  
Métodos de ingeniería.  
Administración del proyecto.

Pequeños sistemas: ambos pueden ser informales/ad-hoc.

Grandes sistemas: ambos deben ser formalizados.

# Desafíos de la IS

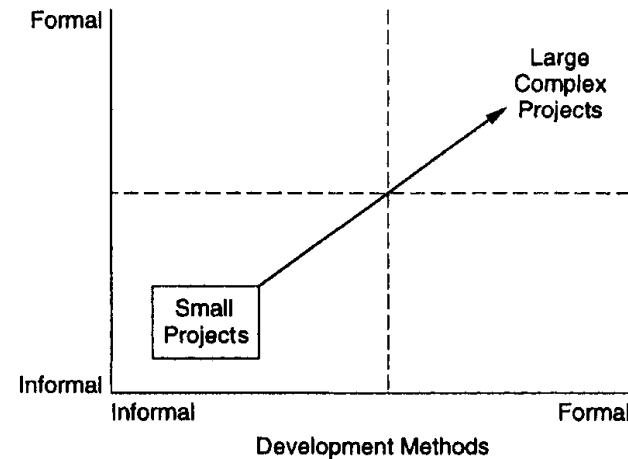
## Escala

- IS debe considerar la escala del sistema a desarrollar  
Los métodos utilizados para desarrollar no siempre escalan a grandes problemas.

Ej.: contar alumnos en un aula vs

- Los métodos de IS deben ser
- Dos claras dimensiones a considerar:  
Métodos de ingeniería.

Administración del proyecto.



Pequeños sistemas: ambos pueden ser

Grandes sistemas: ambos deben ser formalizados.

# Desafíos de la IS

## Escala

Software	Tamaño (KLOC)	Lenguaje/ Herramienta
openssh	38	C, C++
apache	100	C, sh
perl	320	C, C++, yacc
gcc	980	C, C++, yacc
linux	30.000	C, C++
Windows XP	40.000	C, C++

Pequeño:  
< 10 KLOC

Mediano:  
10 a 100 KLOC

Grande:  
100 a 1000 KLOC

Muy grande:  
> 1000 KLOC

# Desafíos de la IS

## Productividad

- IS está motivada por el costo y el cronograma (schedule)  
Tanto una solución que demora mucho tiempo como una que entrega un software barato y de baja calidad son inaceptables.
- El costo del software es principalmente el costo de la mano de obra, por lo que se mide en Persona/Mes.
- El cronograma es muy importante en el contexto de negocios.  
Reducir "time to market".
- La productividad (en términos de KLOC / PM) captura ambos conceptos  
Si es más alta => menor costo y/o menor tiempo.
- Los enfoques de IS deben generar alta productividad.

También depende  
de la cantidad de  
personas

# Desafíos de la IS

## Calidad

- La otra motivación detrás de IS es la calidad.
- Desarrollar software de alta calidad es un objetivo fundamental.
- La calidad del software es difícil de definir (contrariamente al costo y al tiempo).
- El enfoque utilizado en la IS debe producir software de alta calidad.

# Desafíos de la IS

Calidad - Estándar ISO

Calidad de  
Software

```
graph TD; A[Calidad de Software] --> B[Funcionalidad]; A --> C[Confiabilidad]; A --> D[Usabilidad]; A --> E[Eficiencia]; A --> F[Mantenibilidad]; A --> G[Portabilidad];
```

Funcionalidad

Confiabilidad

Usabilidad

Eficiencia

Mantenibilidad

Portabilidad

# Desafíos de la IS

## Calidad - Estándar ISO

- **Funcionalidad:** Capacidad de proveer funciones que cumplen las necesidades establecidas o implicadas.
- **Confiabilidad:** Capacidad de realizar las funciones requeridas bajo las condiciones establecidas durante un tiempo específico.
- **Usabilidad:** Capacidad de ser comprendido, aprendido y usado.
- **Eficiencia:** Capacidad de proveer desempeño apropiado relativo a la cantidad de recursos usados.
- **Mantenibilidad:** Capacidad de ser modificado con el propósito de corregir, mejorar, o adaptar.
- **Portabilidad:** Capacidad de ser adaptado a distintos entornos sin aplicar otras acciones que las provistas a este propósito en el producto.



# Desafíos de la IS

## Calidad (Amy J. Ko)

Boehm '76 (propiedades del código detrás del sistema)

- **Correctitud:** Cuando un programa se comporta según las especificaciones.
- **Confiabilidad:** Cuando un programa se comporta de la misma manera a lo largo del tiempo en un mismo entorno operativo.
- **Robustez:** La medida en que un programa puede recuperarse de errores o entradas inesperadas.
- **Rendimiento:** La medida en que un programa utiliza los recursos informáticos de forma económica. El rendimiento está directamente determinado por la cantidad de instrucciones que debe ejecutar un programa para realizar sus operaciones.
- **Portabilidad:** El grado en que una implementación puede ejecutarse en diferentes plataformas sin ser modificada.
- **Interoperabilidad:** La medida en que un sistema puede interactuar sin problemas con otros sistemas, generalmente mediante el uso de estándares.
- **Seguridad:** La medida en que solo las personas autorizadas pueden acceder a la información en un sistema de software.

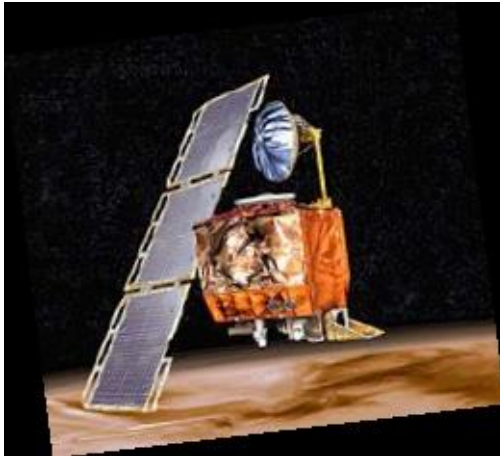
# Desafíos de la IS

## Calidad (Amy J. Ko)

Propiedades del código y de los desarrolladores interactuando con el sistema

- **Verificabilidad:** El esfuerzo requerido para verificar que el software hace lo que está destinado a hacer.
- **Mantenibilidad:** La medida en que el software se puede corregir, adaptar o perfeccionar. Esto depende principalmente de cuán comprensible sea la implementación de un programa.
- **Reutilización:** La medida en que los componentes de un programa se pueden utilizar para fines no deseados.

# Desafíos de la IS



Mars Climate Orbiter:  
Métrico vs Imperial  
(- \$327.6 million)

HMS Sheffield:  
Exocet amigo  
(-20 personas)



# Desafíos de la IS

## Calidad (Amy J. Ko)

Propiedades del diseño y experiencia del usuario con un sistema de software:

- **Learnability:** La facilidad con la que una persona puede aprender a operar un programa.
- **Eficiencia del usuario:** La velocidad con la que una persona puede realizar tareas con un programa.
- **Accesibilidad:** La diversidad de habilidades físicas o cognitivas que pueden operar con éxito el software.
- **Utilidad:** La medida en que el software resuelve un problema.
- **Privacidad:** La medida en que un sistema impide el acceso a información destinada a una audiencia o uso en particular.
- **Coherencia:** La medida en que la funcionalidad relacionada en un sistema aprovecha las mismas habilidades, en lugar de requerir nuevas habilidades para aprender a usarlas.
- **Usabilidad:** Esta cualidad engloba todas las cualidades anteriores. Lo usamos para representar cualquier cualidad que afecte la capacidad de alguien para usar un sistema.
- **Sesgo:** Las múltiples formas en que el software puede discriminar, excluir o ampliar o reforzar las estructuras discriminatorias o excluyentes en la sociedad.

# Desafíos de la IS

## Calidad

- Múltiples dimensiones en la calidad:
  - => No puede reducirse a un sólo número.
- El concepto de calidad es específico al proyecto:
  - => En algunos casos confiabilidad es más importante.
  - => En otros, usabilidad
  - => Para cada proyecto, el objetivo de calidad debe especificarse de antemano, y el objetivo del desarrollo será cumplir con el objetivo de calidad preestablecido.
- Confiabilidad es usualmente el principal criterio de calidad.

# Desafíos de la IS

## Calidad

- Confiabilidad inversamente relacionada a la probabilidad de falla. Es difícil medir la cantidad de defectos.

(+fallas => -confiable)

Aproximado por el número de defectos encontrados en el software.

- Para normalizar:

Calidad = densidad de defectos = Cantidad defectos en software entregado/tamaño

- En las prácticas habituales:

< 1 defecto/KLOC

- Pero: ¿qué es un defecto?

Depende del software.

# Desafíos de la IS

“A secure system is necessarily going to be less learnable, because there will be more to learn to operate it. A robust system will likely be less maintainable because it will likely have more code to account for its diverse operating environments. Because one cannot achieve all software qualities, and achieving each quality takes significant time, it is necessary to prioritize qualities for each project.”

— Amy J. Ko

# Desafíos de la IS

## Consistencia y repetitividad

- Algunas veces un grupo puede desarrollar un buen sistema de software.
- Desafío clave en IS: cómo asegurar que el éxito pueda repetirse (con el fin de mantener alguna consistencia en la calidad y la productividad).
- Un objetivo de la IS es la sucesiva producción de sistemas de alta calidad y con alta productividad.
- La consistencia permite predecir el resultado del proyecto con certeza razonable. Sin consistencia sería difícil estimar costos.
- Marcos como la ISO9001 o el CMM se enfocan mucho en este aspecto.



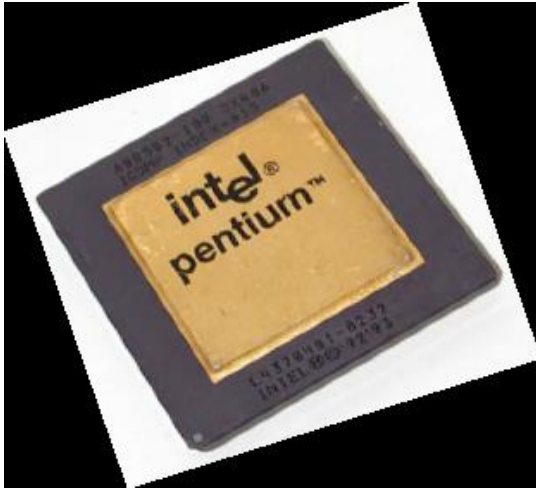
# Desafíos de la IS

## Cambio

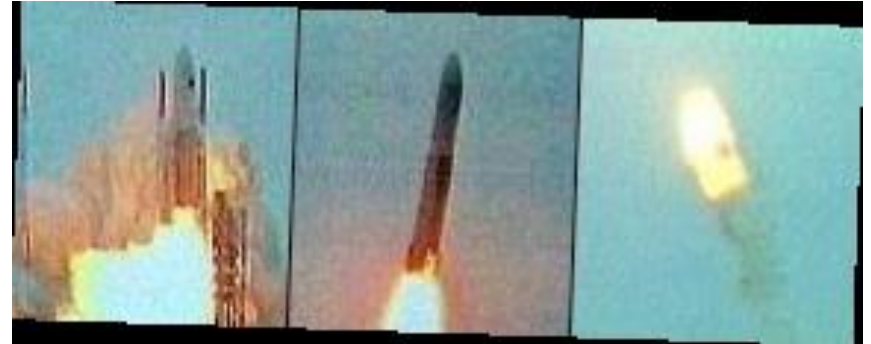
- Cambios en las empresas/instituciones es lo habitual.
- El software debe cambiar para adaptarse a los cambios de dicha institución.
- Las prácticas de IS deben preparar al software para que éste sea fácilmente modificable  
Los métodos que no permiten cambios (aún si producen alta calidad y productividad) son poco útiles.

# Desafíos de la IS

## Cambio



Pentium:  
FDIV  
(-\$730 million in 2018)

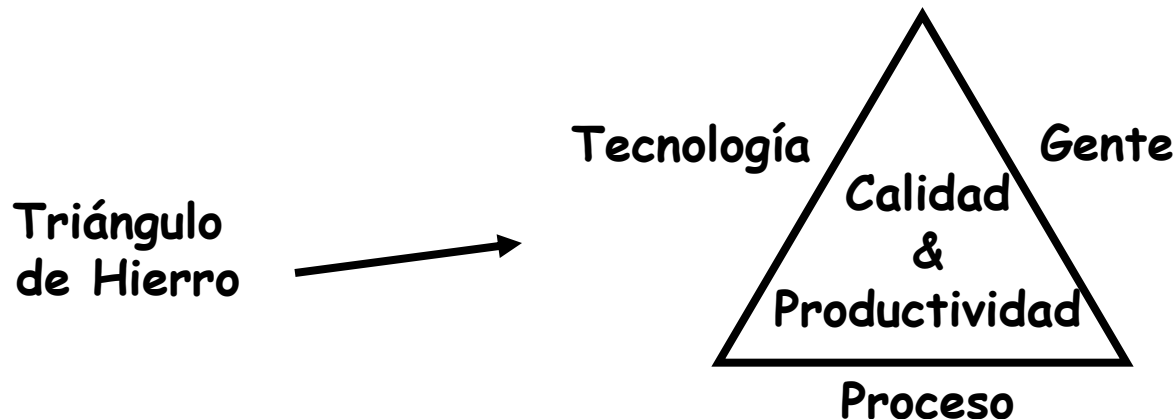


Ariane 5:  
32 vs 16 bits  
(-\$500 million)



# Enfoque de la IS

- Ya comprendemos el dominio del problema y los factores que motivan la IS:  
Consistentemente desarrollar software de alta calidad y con alta productividad (C&P) para problemas de gran escala que se adapten a los cambios.
- C&P son los objetivos básicos a perseguir bajo gran escala y tolerancia a cambios.
- C&P son consecuencia de la gente, los procesos y la tecnología.



# Enfoque de la IS

- La IS se enfoca mayormente en el proceso para conseguir los objetivos de calidad y productividad.
- El enfoque sistemático es realmente el proceso que se utiliza.
- La IS separa el proceso para desarrollar software del producto desarrollado (i.e. el software). Es aquí donde se distingue de las otras disciplinas informáticas.
- Premisa: El proceso es quien determina, en buena medida, la C&P  
=> un proceso adecuado permitirá obtener gran C&P.
- Diseñar el proceso apropiado y su control es el desafío clave de la IS.

# Enfoque de la IS

“What was amazing was that a large team of highly intelligent programmers could labor so hard and so long on such an unpromising project.....

Especially don't believe us when we promise to repeat an earlier success, only bigger and better next time.”

— C. A. R. Hoare ('80)

# Enfoque de la IS

## El proceso de desarrollo en fases

- El proceso de desarrollo consiste de varias fases.
- Cada fase termina con una salida definida.
- Las fases se realizan en el orden especificado por el modelo de proceso que se elija seguir.
- El motivo de separar en fases es la separación de incumbencias: cada fase manipula distintos aspectos del desarrollo de software.
- El proceso en fases permite verificar la calidad y el progreso en momentos definidos del desarrollo (al final de la fase).
- El proceso en fases es central en el enfoque de la IS para solucionar la crisis del software.

# Enfoque de la IS

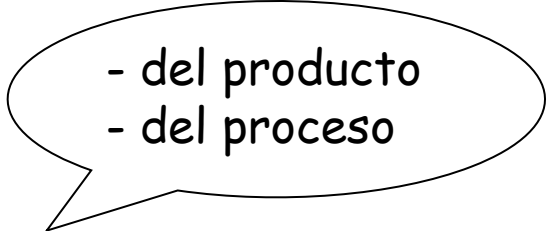
## El proceso de desarrollo en fases

- Se han propuesto varios modelos de procesos para el desarrollo de software (y cada organización usa su propia variante).
- En general consisten de:
  - Análisis de requisitos y especificación
  - Arquitectura y Diseño
  - Codificación
  - Testing
  - Entrega e instalación
- Los enfoques sistemáticos requieren que cada etapa se realice rigurosa y formalmente.

# Enfoque de la IS

## Administración del proceso

- El proceso de desarrollo no establece cómo asignar los recursos a las distintas tareas, ni cómo organizarlas temporalmente, ni cómo asegurar que cada fase se desarrolló apropiadamente, etc.
- Estas cuestiones se manejan a través de la administración del proceso.
- Sin la administración del proceso es virtualmente imposible cumplir con los objetivos de C&P.
- El planeamiento del proyecto es central a la administración del proceso para poder determinar cuestiones como:
  - ¿el proyecto se está desarrollando a término?
  - ¿el proyecto procede acorde con el presupuesto?
  - ¿se está cumpliendo con los objetivos de calidad?
- Son importantes para poder planear y administrar las métricas y medidas.



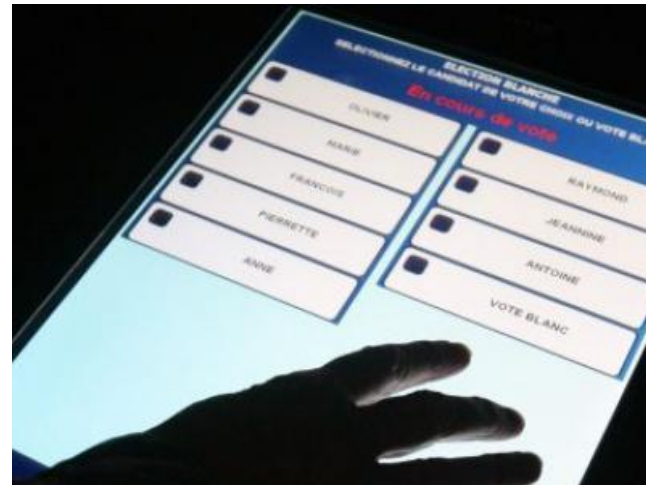
- del producto
- del proceso



# Dominio del problema

## Poco confiable

Voto electrónico:  
Integridad/Confidencialidad



# Lectura complementaria

- Capítulo 1 - Jalote
- The Emperor's Old Clothes by C.A.R. Hoare, Communications of the ACM, 1981  
<https://zoo.cs.yale.edu/classes/cs422/2010/bib/hoare81emperor.pdf>
- Crisis del software  
[https://en.wikipedia.org/wiki/Software\\_crisis](https://en.wikipedia.org/wiki/Software_crisis)
- Quality by Amy J. Ko  
<http://faculty.washington.edu/ajko/books/cooperative-software-development/quality.html>
- Pentium FDIV:  
[https://en.wikipedia.org/wiki/Pentium\\_FDIV\\_bug](https://en.wikipedia.org/wiki/Pentium_FDIV_bug)
- Ariane 5:  
<http://www-users.math.umn.edu/~arnold//disasters/ariane.html>
- Therac-25:  
<https://en.wikipedia.org/wiki/Therac-25>
- Mars Climate Orbiter:  
[https://en.wikipedia.org/wiki/Mars\\_Climate\\_Orbiter](https://en.wikipedia.org/wiki/Mars_Climate_Orbiter)
- Voto electrónico:  
<https://www.youtube.com/watch?v=8RtYqCPoSW8>