Introducción

Ingeniería de Software

Hernán Astudillo & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María
<hernan@inf.utfsm.cl, g.marquez.o@gmail.com>

Contexto

El software y tú

- Contexto
  - Los submarinos hunden barcos
- Problema
  - ¿Cómo detectar submarinos?
- Propuesta
  - Hervir el mar para que submarinos deban salir a la superficie
- Implementación
  - Detalle técnico ☺ “How hard can it be?”

Demo a construir

- USB Wine (proyecto de curso previo)
- ¡Las demos son peligrosas!
  - Son promesas y alguien se las creyó…

¿Ingeniería de Software?

Problema

- 30-50% de funcionalidad es innecesaria (overhead)
  - [C. Ebert, R. Dumke - 2007]
- Estudio de 8000 proyectos en EEUU (2000)
  - En plazo y costo: 9%-16%
  - Retrasados/excedidos: 53%-60%
  - Cancelados (abortados): 31%
- ¿Porqué fueron cancelados?
  - Requisitos incompletos: 13%
  - Falta de participación del usuario: 12%
  - Falta de recursos: 11%
  - Expectativas descontroladas: 10%
  - Falta de apoyo gerencial: 9%
  - Reqisitos inestables: 9%
  - Falta de planificación: 8%
  - Obsolescencia pre-parto: 7%

Analogía

- Casi todos saben cocinar algo…
  - Un huevo frito
- Algunos saben preparar una cena…
  - Compras previas
  - Varios platos simultáneos, entrega sincronizada
- Pocos saben preparar un banquete…
  - Compras previas en gran escala
  - Equipo: mozos, maitre…
- Poquísimos saben hacer un banquete CADA NOCHE
  - Logística de entrega continua
  - Gestión de personal
  - Gestión de calidad noche tras noche

Analogía

- Uds saben *programar* …
  - Técnicas: programación (desarrollo)
  - Solos (mostly)
- Uds saben *hacer software* …
  - Técnicas: análisis (casos de uso), diseño (patrones, frameworks)
  - En *equipos* pequeños
- Pero Uds necesitan vivir de hacer software
  - Técnicas: estimación, gestión de calidad…
  - … sistemas complejos, compuestos de programas
  - Necesidad de mantener *organización* estable
  - … integrada con otras organizaciones
  - … en mejoramiento continuo

Ingeniería de Software

- Objeto de estudio
  - Construcción *sistemática, eficaz y eficiente* de software eficaz y eficiente
- Audiencia
  - *Grupos* (equipos y organizaciones) que deben vivir y prosperar haciendo software
  - *Profesionales* que estarán en estas organizaciones
  - *Gestores* de estas organizaciones y profesionales
  - No para *artistas* de software
- Una “ciencia de lo artificial” (Simon)
  - Énfasis en lo que *debe* ser más que en lo que es

Ingeniería de Software

- Propuesta de la disciplina
  - Enfoque ingenieril
  - Sistematizar y transmitir experiencia
- Origen del concepto
  - Conferencia de la OTAN en 1968
  - Motivación: resolver la mítica “crisis del software”
  - Idea: enfatizar el uso de filosofías y paradigmas de disciplinas ingenieriles ya establecidas

Vocabulario de la disciplina

- Procesos de desarrollo de software
  - *Proceso*: organización sistemática de personas, técnicas y herramientas
  - *Técnica*: método para una tarea específica
  - *Herramienta*: apoyo (automático o metodológico) a una técnica
- Áreas funcionales clásicas
  - *Requisitos* o requerimientos (incluye Análisis)
  - *Diseño* (incluye Arquitectura)
  - *Construcción* (incluye Reuso)
  - *Prueba* (verificación y validación)
  - *Gestión* del proceso de desarrollo
  - Evolución (“mantención”)
- Áreas transversales
  - Calidad (¿qué significa “eficaz y eficiente”?)

Observación envidiosa

- La *productividad* de desarrollo de software crece más lento que la de hardware
  - Carácter lógico del software
    - Hacer hardware implica diseñar (difícil, ad-hoc) y replicar (automatizable)
    - Hacer software es diseñar una computación; la replicación es banal
  - Casi todo es negociable en el desarrollo
    - En principio sólo hay estimaciones, no predicciones exactas
    - Clientes y usuarios “ven” cada estatua posible en la roca, pero el escultor debe hacer una sola
  - Formación profesional
    - Labor tradicionalmente muy artesanal
    - Resistencia al rigor sistemático

Mitos No-informáticos del Software

- estándares y procedimientos bastan
- tecnología de punta basta
- más gente para ponerse al día
- programación inmediata
- fácil acomodo de los cambios
- programación: fin del trabajo
- calidad: sólo del ejecutable
- código es el único producto

Dificultades en Producción de Software

- Fred Brooks propuso (1986) distinguir:
  - 1. Dificultades accidentales de construir software
    - Deficiencias en técnicas, modelos, herramientas…
    - Solubles con avances de investigación
  - 2. Dificultades esenciales de construir software
    - Complejidad, Conformidad, Necesidad de cambios, Invisibilidad
    - Problemas esenciales: insolubles por definición

"No silver bullet" [F. Brooks, IEEE Computer, 1986]

Ingeniería de Software

- Objetivos
  - Maximizar calidad (de procesos y productos)
  - Maximizar productividad (de procesos y productos)
  - Minimizar riesgos (de procesos y productos)
- Posibles enfoques (diferentes comunidades)
  - Constructores básicos más poderosos
    - lenguajes, notaciones, abstracciones…
  - Mejores técnicas de control de calidad
  - Mejores herramientas y métodos
  - Filosofía global -- enfoque de procesos

Extra: Un poco de historia

Un poco de Historia [1]

- Inicialmente:
  - El desarrollo de software era tarea de una sola persona.
  - El problema (de tipo científico o ingenieril) estaba bien acotado y bien comprendido.
  - Generalmente, el usuario final (científico o ingeniero) era el mismo programador, quien desarrollaba software para apoyar su propio trabajo.
  - La aplicación era más bien simple y el desarrollo se reducía a la codificación en un lenguaje, típicamente de bajo nivel.
  - El modelo usado era de codificar-corregir:
    - Escribir el código.
    - Revisar y eliminar los errores o mejorar/aumentar la funcionalidad.

Un poco de Historia [2]

- El hardware aumentó sus capacidades y disminuyó sus costos
  - Se amplió el ámbito de aplicaciones y se masificó el uso de computadores.
  - Se incursionó en áreas donde los problemas no están bien acotados (p.ej. administrativos) y el desarrollo se tornó más complejo.

Un poco de Historia [3]

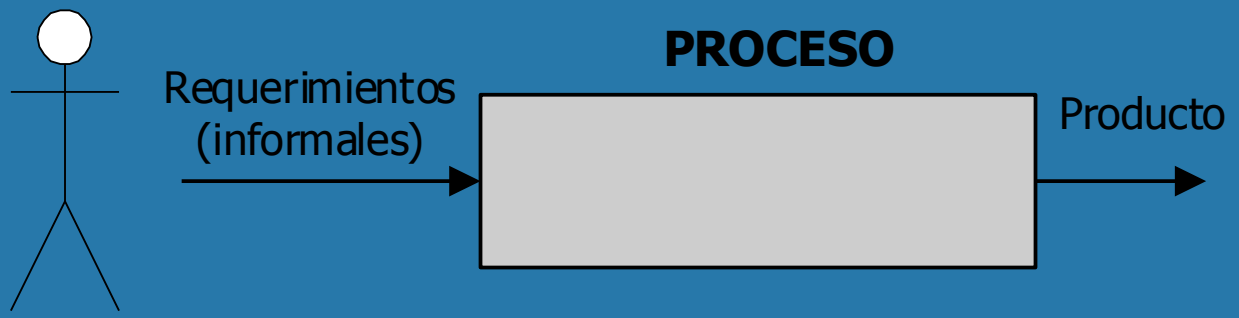
- Con el tiempo, pasó a haber una gran masa de software pre-existente
  - Aparece un problema aún mayor: mantener los sistemas.
  - Esto escapó de las manos de los usuarios-programadores, quienes ya no pudieron ejecutar ni controlar el proceso
    - Sólo dominaban su especialidad, no el desarrollo de software.
  - Se incorpora tópicos organizacionales y psicológicos
  - Surgen demandas por mayor calidad y confiabilidad de las aplicaciones.
  - Surgen demandas por mayor calidad y confiabilidad de las procesos de desarrollo.

Un poco de Historia [4]

- El desarrollo se convierte en actividad de grupo
  - Exige planificar, organizar y estructurar el trabajo en torno a “proyectos”.
  - La comunicación entre humanos (usuario-desarrollador y desarrollador-desarrollador) se convierte en un problema.
- Percepción en academia de una “crisis del software” a fines de los ’60.

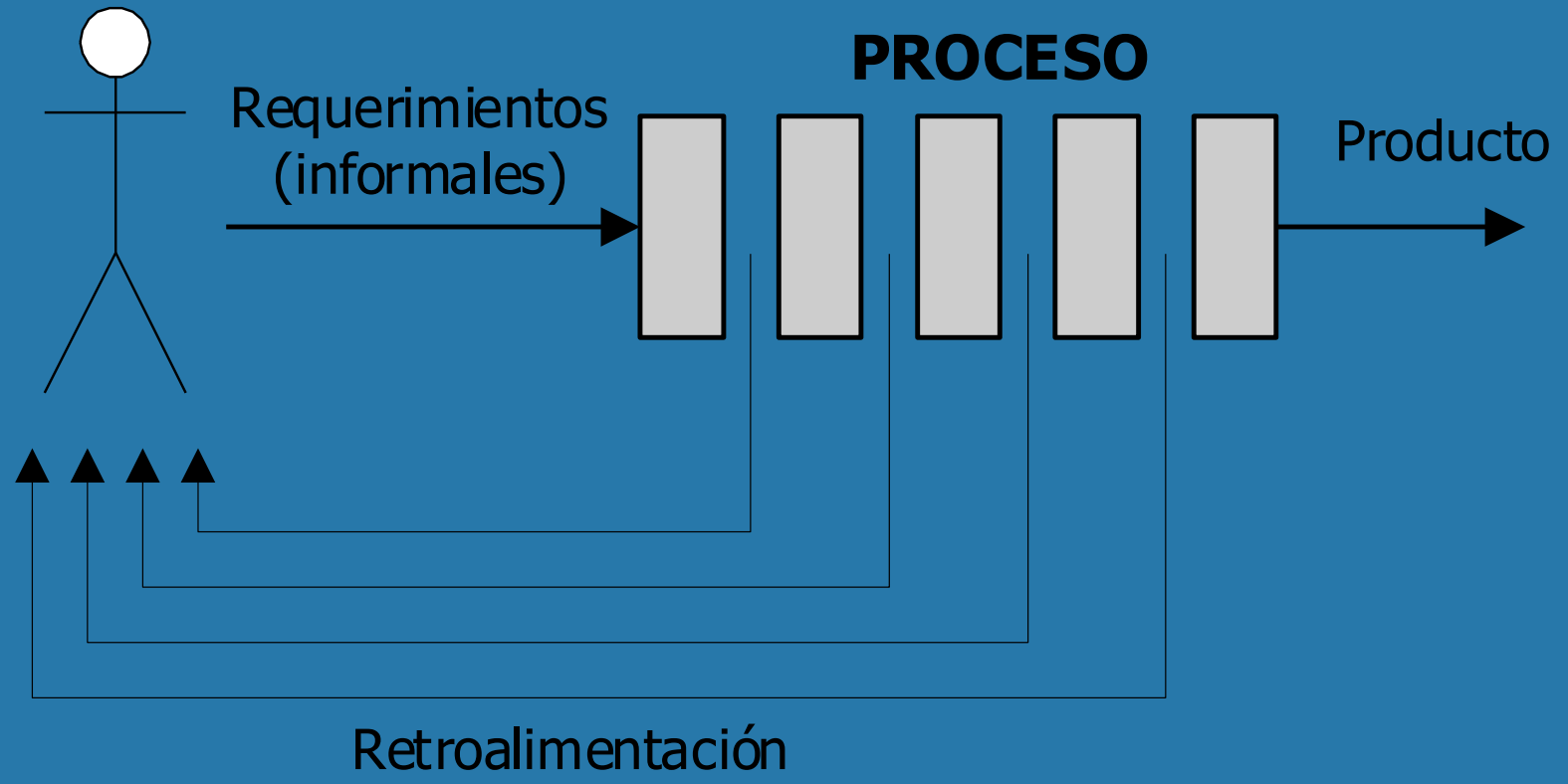
Un poco de Historia [5]

- El proceso de desarrollo a la antigua, como “caja negra”.



Un poco de Historia [6]

- El proceso de desarrollo como se concibe ahora.



Propósito de la disciplina

- Ideas-fuerza
  - “Entender problema, destino y camino antes de ponerse en marcha (si se puede)”
  - “Resolver el problema, no reinventar la rueda”
- Propósito: reducir los riesgos de construcción
  - Efectivos: no hacer algo inútil (buen producto)
  - Eficientes: no malgastar recursos (buen proceso)
- Amenazas
  - “Parálisis de análisis”: querer saber todo antes de empezar
  - “Preciosismo”: querer hacer joyita aunque no haga falta