SOFTWARE

Software

- Programa
- Paradigmas de programación
- Cómo se produce software
- Modelos de procesos
- Atributos del buen software

Programa

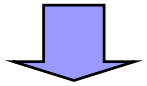
- Representación de un programa

- Cómo son los programas
  - Un programa
    - Modela un problema
      - En función del problema
        - Modelo de desarrollo
    - Cajero automático vs. Diagnóstico médico

Programa

Modelos de desarrollo

- Diferentes enfoques de implementación
- Cada uno
  - Modelo de construcción de programas (paradigma de programación)



  - Metodología de desarrollo de programas

Programa

Objetivos del paradigma y metodología

- Paradigma de programación: Forma arquitectónica para construir/desarrollar el programa
  - Asociado a lenguajes
- Metodología de programación
  - Pasos a seguir para construir/desarrollar el programa

Paradigmas de programación

- Funcional
- Lógico
- Imperativo
- Orientado a objetos
- ...

Paradigmas de programación

- Funcional
  - Manejo implícito de la memoria.
  - Esencial: concepto de función.
  - Programa: Conjunto de funciones + aplicación a datos
  - Ejemplos: LISP, Scheme, ML.

Paradigmas de programación

- Lógico
  - Basado en el calculo de predicados.
  - Mecanismo de demostración automática de teoremas.
  - Programa: Conjunto de axiomas y un objetivo.
  - Ejemplo: Prolog.

Paradigmas de programación

- Imperativo
  - Esencial: Asignación y secuenciación
  - Programa: Secuencia de instrucciones
  - Ejemplos: Fortran, Algol, Basic, C, Pascal

Paradigmas de programación

- Orientado a Objetos
  - Mundo real = objetos + interacción
  - Esencial: conceptos de objeto, herencia y mensaje
  - Programa: Conjunto de objetos + mensajes
  - Ejemplos: Smalltalk, Java, C++, Obliq, Dylan, CLOS, Squeak, etc.

Paradigmas de programación

- Independiente del paradigma
  - Ciclo de desarrollo/vida de un programa
  - Configuración de un programa

Ciclo de desarrollo/vida de un programa

- IS. Describir el problema
- P. Análisis
- P. Diseño
- P. Implementación
- P/IS. Prueba
- IS/P. Instalación
- U. Uso
- IS/P. Mantenimiento
- C. Obsolescencia

C.D.

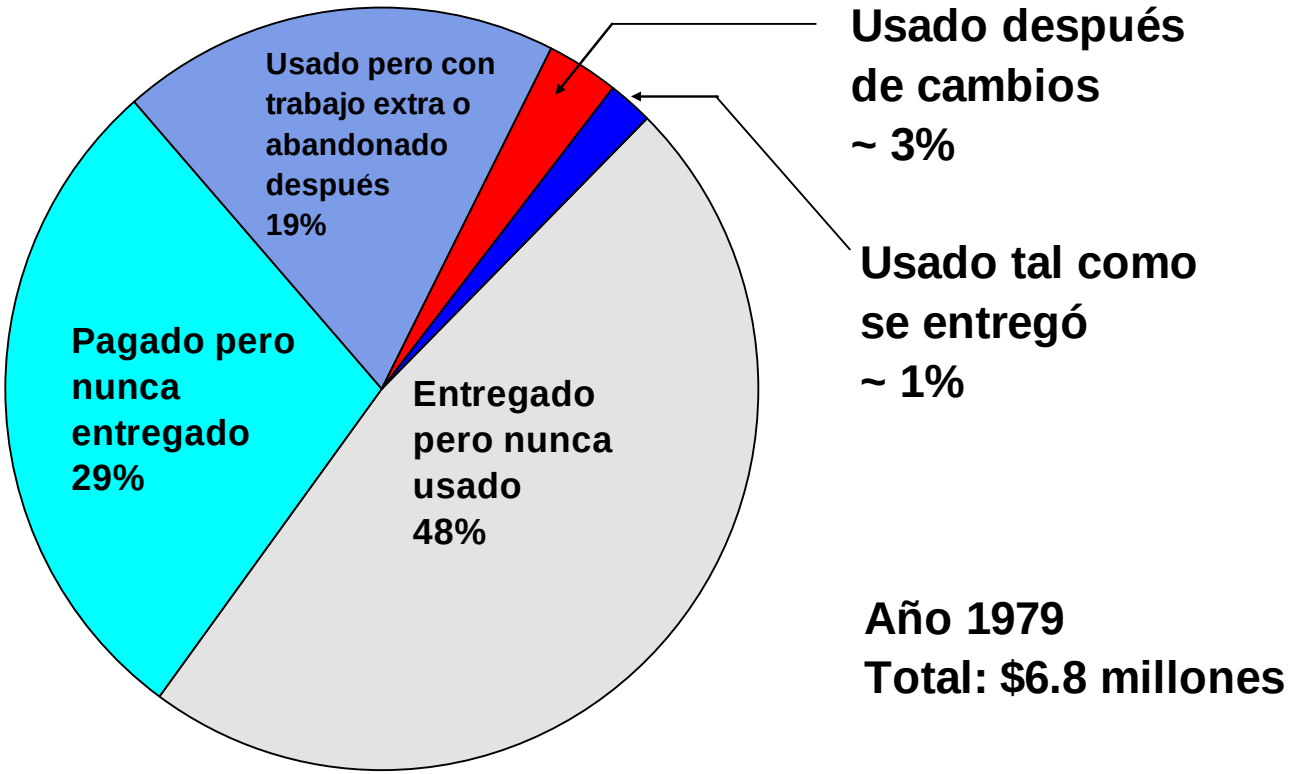
C.V.

Configuración de un programa

- C.P. = Código + Documentación
  - Descripción del problema
  - E/S/I
  - Algoritmo (Análisis + Diseño)
  - Pruebas
    - Resultado esperado
    - Casos de prueba
    - Resultados obtenidos
  - Código

Cómo se produce software

- A finales de los 70 (Crisis del Software):
  - Proyectos software contratados por el DoD Americano:



Cómo se produce software

- A finales de los 80. Capers Jones estudia el software adquirido por la Administración Pública Americana:
  - Sólo entre el 5% y el 10% era directamente usable.
  - Entre el 30% y el 40% nunca se había usado o nunca se podría usar.

Cómo se produce software

- En la década de los 90. Standish Group en su Chaos 2001 Report:
  - Proyectos de desarrollo “exitosos”:
    - 16% en 1994.
    - 27% en 1996.
    - 26% en 1998.
    - 28% en 2000.

Cómo se produce software

- Actualmente. Standish Group en su Chaos 2006 Report:
  - Proyectos de desarrollo “exitosos”:
    - 35% en 2006 (vs. 16% en 1994).
  - Proyectos “challenged”:
    - 46% en 2006 (vs. 53% en 1994).
  - Proyectos de desarrollo “completamente fallidos”:
    - 19% en 2006 (vs. 31% en 1994).
  - Conclusiones:
    - Se va mejorando progresivamente el desarrollo de software desde el primer Chaos Report en 1994

Cómo se produce software

Consecuencias (no satisfactorios)

- Desviaciones en costes y tiempos
  - Muchas veces inaceptables
    - No uso
- Calidad pobre en sistemas, incluso vitales
  - Funcionalidad recortada
  - Rendimiento mejorable
  - Etc.
- Documentación escasa o nula
- Mantenibilidad: difícil y costosa

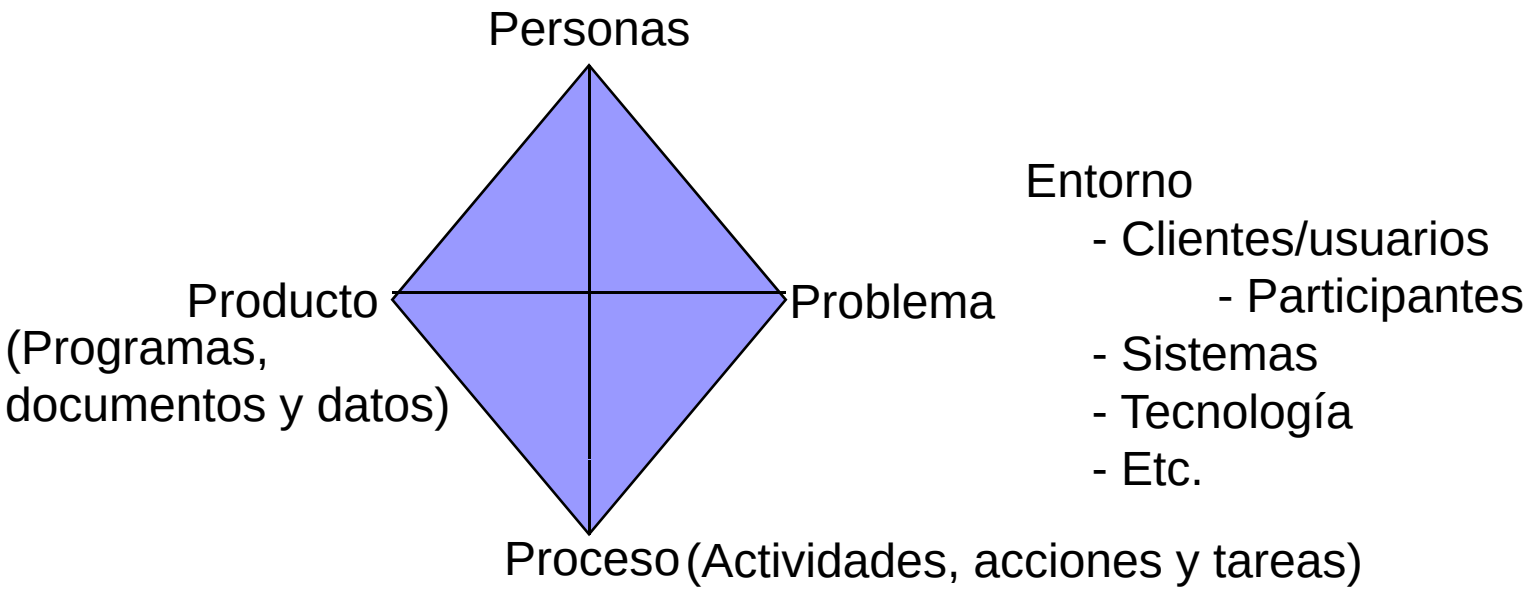
Cómo se produce software

Para seguir mejorando

- Seguir trabajando en los diferentes aspectos del desarrollo de software: las 4 “P”
  - Problema: es una realidad, que se plasma en un
  - Producto: que se obtiene a través de un
  - Proceso: que realizan las
  - Persona: que una vez conocido el problema obtienen el producto a través de un proceso

Cómo se produce software

Las 4 “P”



Ej. Relaciones complejas

- Un producto “satisfactorio” puede que no sea bueno
  - Sumar los 2 primeros números (1, 2, 3)

Cómo se produce software

Solución básica

- Modelos de proceso + Gestión
- Gestión
  - Proyecto
    - Estimación
    - Equipo trabajo
    - Recursos
    - Tareas
    - Productos a entregar
  - Otras
    - Gestión de la configuración
    - Gestión de la calidad
    - Gestión de riesgos

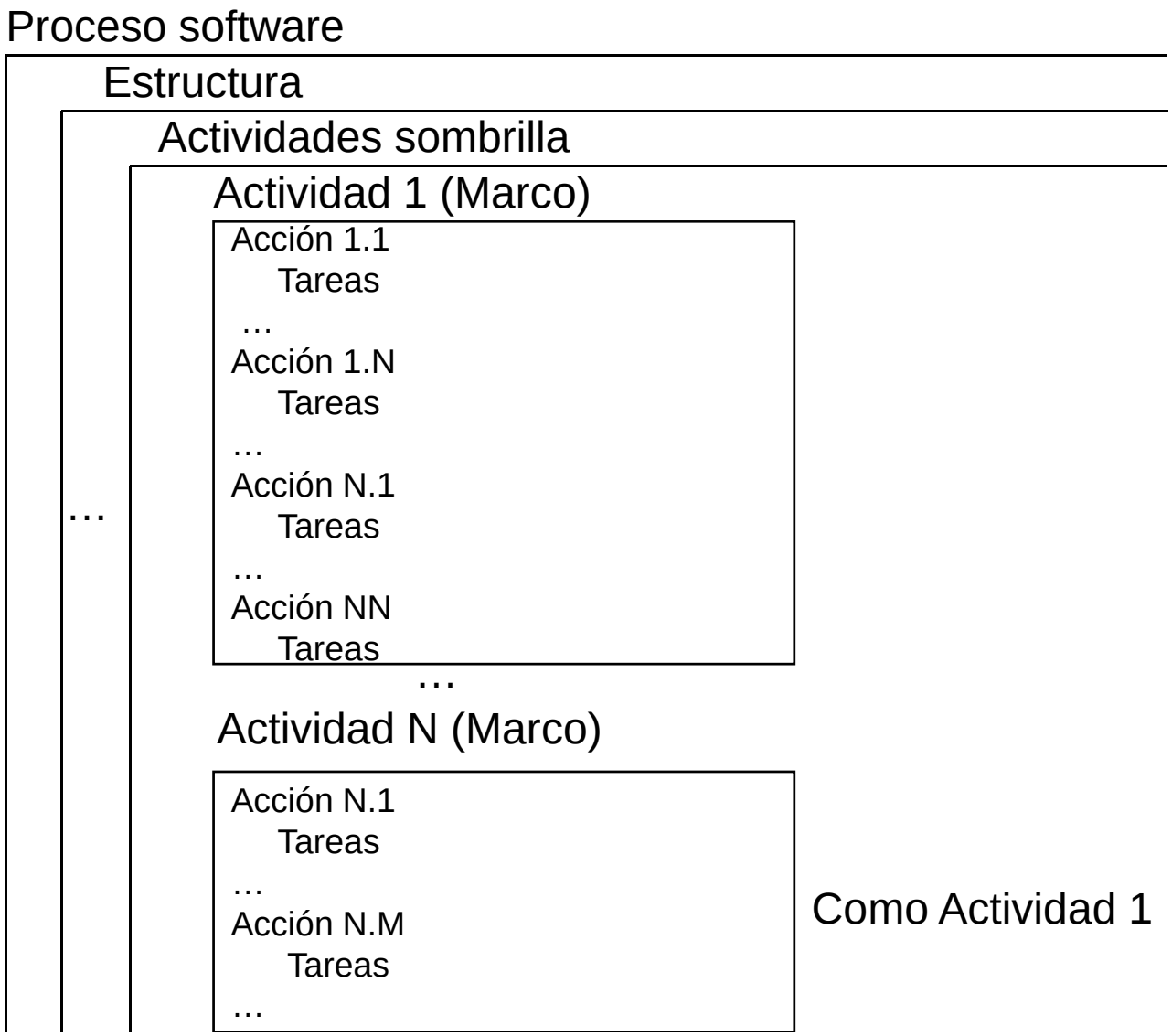
Modelos de proceso

- Definen un “Conjunto Distinto” de actividades, acciones, tareas y productos de trabajo que se requieren para producir software de alta calidad
- Son una guía, no perfectos (Flexibilidad vs. Dogmatismo)
- Los IS “escogen”, “adaptan” y “siguen” un modelo
  - Tipo de problema
  - Complejidad del problema
  - Recursos
  - Cultura organizacional
- Dan estabilidad, control y organización

Modelos de proceso

- Prescriptivos
  - Secuencial
    - Cascada
    - V
  - Incremental
    - Incremental
    - D.R.A
  - Evolutivo
    - Prototipado
    - Espiral
- Especializados
  - Desarrollo basado en componentes
  - Modelos de métodos formales
  - Desarrollo orientado a aspectos

Modelos de proceso



Modelos de proceso

- Actividades genéricas
  - Comunicación
  - Planificación (gestión)
  - Modelado
    - Análisis
    - Diseño
  - Construcción
    - Código
    - Prueba
  - Despliegue
    - Entrega
    - Retroalimentación

Modelos de proceso

- Actividades sombrilla
  - Seguimiento y control de proyectos
  - Administración del riesgo
  - Aseguramiento de la calidad
  - Revisiones técnicas
  - Administración de la configuración
  - Administración de la reutilización
  - Etc.

Atributos del buen software

- Mantenibilidad
  - Crítico
  - Cambio inevitable
  - Facilidad de evolución
- Confiabilidad. Ante fallos no se pueden causar daños físicos o económicos
  - Fiabilidad
  - Protección
  - Seguridad

Atributos del buen software

- Eficiencia
  - Optimización de los recursos del sistema
    - Tiempos de respuesta
    - Tiempos de procesamiento
    - Memoria
    - Etc.
- Usabilidad
  - Facilidad de uso por el usuario
    - Interface adecuada
    - Buena documentación

Retos

- Heterogeneidad
- Entrega
- Confianza

Preguntas antiguas y poliédricas. Realidades viejas

- Se tarda demasiado en construir el software
- Los costes de desarrollo son altos
- No se pueden encontrar todos los errores
- Mucho tiempo y esfuerzo en mantenimiento
- Dificultad de medir/cuantificar el avance
  - Desarrollo
  - Mantenimiento