Tema 0

El proceso de desarrollo del software

- 0 El proceso de desarrollo del software
  - 0.1 El proceso: una visión general
  - 0.2 Modelos prescriptivos de proceso
  - 0.3 La práctica: una visión genérica

0.1 El proceso: una visión general

- Ingeniería del software (IS): una tecnología estratificada (gráfico de la pirámide)
  - Basado en el compromiso con la calidad
  - El proceso como contexto (donde se aplican los métodos…)
  - Los métodos proporcionan los “cómo” técnicos para construir software
  - Las herramientas proporcionan el soporte automatizado para el proceso y los métodos

- Marco de trabajo para el proceso (MTP)
  - Diferencia entre actividades genéricas y actividades sombrilla (discreto en el tiempo / continuado)
  - Marco de trabajo genérico del proceso. Actividades genéricas (diagrama actividades):
    - Comunicación con el cliente: captura requisitos…
    - Planificación: tareas a realizar, riesgos, recursos requeridos, productos trabajo a generarse, programa trabajo
    - Modelado: creación modelos para entender requisitos y para determinar un diseño para satisfacerlos
    - Construcción: generación código y pruebas
    - Implantación: entrega software al cliente (y evaluación)

- Actividades, acciones y tareas (esquema descomposición). Las tareas están acompañadas de productos, puntos aseguramiento calidad…
- Actividades sombrilla incluyen:
  - Aseguramiento de la calidad del software: actividades para asegurar que el software tenga calidad
  - Revisiones técnicas formales: evalúa los productos de trabajo para descubrir errores antes de que se propaguen
  - Medición: mediciones del proceso/proyecto/producto para entregar software que satisfaga al cliente
- Adaptación del modelo de proceso a: problema, proyecto, equipo, cultura organizativa
- Diferencia entre los modelos de proceso:
  - El grado y la forma en que se aplican las distintas actividades

0.2 Modelos prescriptivos de proceso

- Modelos prescriptivos
  - Prescriptivo: pretende dar unas normas claras
  - Se ajustan a las actividades del MTP, aunque cada uno prioriza unas actividades
- El modelo en cascada (ciclo vida clásico)
  - Secuencial, lineal: realizamos una actividad y pasamos a la siguiente (y así sucesivamente)
  - Actividades:
    - Comunicación:
      - Inicio del proyecto, recopilación de requisitos
    - Planificación:
      - Estimación, itinerario, seguimiento
    - Modelado:
      - Análisis, diseño
    - Construcción:
      - Código, prueba
    - Implantación:
      - Entrega, soporte, retroalimentación (si se incluye el modelo es iterativo)

- Problemas al aplicar el modelo:
  - Al ser secuencial hay que terminar una actividad para pasar a la siguiente
  - Adecuado cuando los requisitos están bien definidos, pero los requisitos no suelen estar bien definidos en casi ningún proyecto
  - Cuando se obtiene por fin un producto puede ser demasiado tarde

- El modelo incremental
  - El software se desarrolla de forma incremental: en cada incremento se incorporan nuevos requisitos.
  - Se aplica el modelo en cascada completo a cada incremento de software
  - Se pueden solapar en el tiempo actividades correspondientes a distintos incrementos de software (gráfico)
  - Se progresa desde un primer incremento o “producto esencial” hasta el último o producto completo. En cada paso intermedio se entrega un producto incompleto pero operacional.
  - Recomendado cuando no todo el personal está disponible o cuando hay factores que impiden, a priori, desarrollar el producto final

- Construcción de prototipos
  - Existe una indefinición en la especificación de requisitos
  - Se suele usar como técnica en los restantes modelos
  - Se omiten fases del modelo en cascada y las que se realizan se hacen de forma rápida
  - Al final de cada iteración se genera un prototipo que, ajustado por el cliente, sirve de base para la siguiente iteración, si es que ésta es necesaria.
  - Una vez construido el prototipo, y cumplido su propósito, se debe desechar y hay que rediseñar el sistema construido

- Problemática asociada
  - Tanto al desarrollador como al cliente les resulta atractivo puesto que se construye algo de inmediato
  - El cliente puede pedir “unos pequeños ajustes” para elaborar el producto final a partir del prototipo asumiendo la supuesta calidad y facilidad de mantenimiento de la aplicación
  - El desarrollador puede “olvidar” las soluciones inadecuadas que se adoptaron para que el prototipo funcionara con rapidez

0.3 La práctica: una visión genérica

- La práctica: conceptos, principios, métodos y herramientas a los que recurre el ingeniero de software

- La práctica de la ingeniería del software
  - La esencia de la práctica: la resolución de problemas
    1. Entender el problema (comunicación y análisis)
    2. Planificar una solución (diseño)
    3. Llevar a cabo el plan (generación de código)
    4. Examinar el resultado para probar la precisión (realización de pruebas y aseguramiento de la calidad)

- Preguntas que surgen en el ámbito de la IS
  1. Entender el problema:
     - ¿Quién es el cliente? ¿Qué funciones, características y comportamientos se requieren? ¿El problema se puede subdividir? ¿Se puede representar el problema (modelo análisis)?
  2. Planificar una solución:
     - ¿Hay un software existente que implemente lo que se requiere? ¿Se ha resuelto un problema similar cuya solución pueda reutilizarse? ¿Se puede representar la solución (modelo de diseño) de forma que nos conduzca a una implementación efectiva?
  3. Llevar a cabo el plan:
     - ¿El código generado "se puede seguir" conforme se estableció en el "modelo de diseño"?
  4. Examinar el resultado para probar la precisión:
     - ¿Es posible probar cada componente? ¿Se ha implementado una estrategia de prueba razonable? ¿La solución es acorde con lo requerido? ¿Se ha validado el software contra los requisitos del cliente?

- a) Prácticas de comunicación
  - Objetivo de la comunicación: la obtención de los requisitos del cliente

- Diferencia entre cliente y usuarios finales:
  - Cliente: solicita el software, define los objetivos de negocio para el software, proporciona los requisitos del producto, coordina los recursos económicos
  - Usuario: usará el software, definirá los detalles operativos del software para que el propósito del negocio pueda alcanzarse

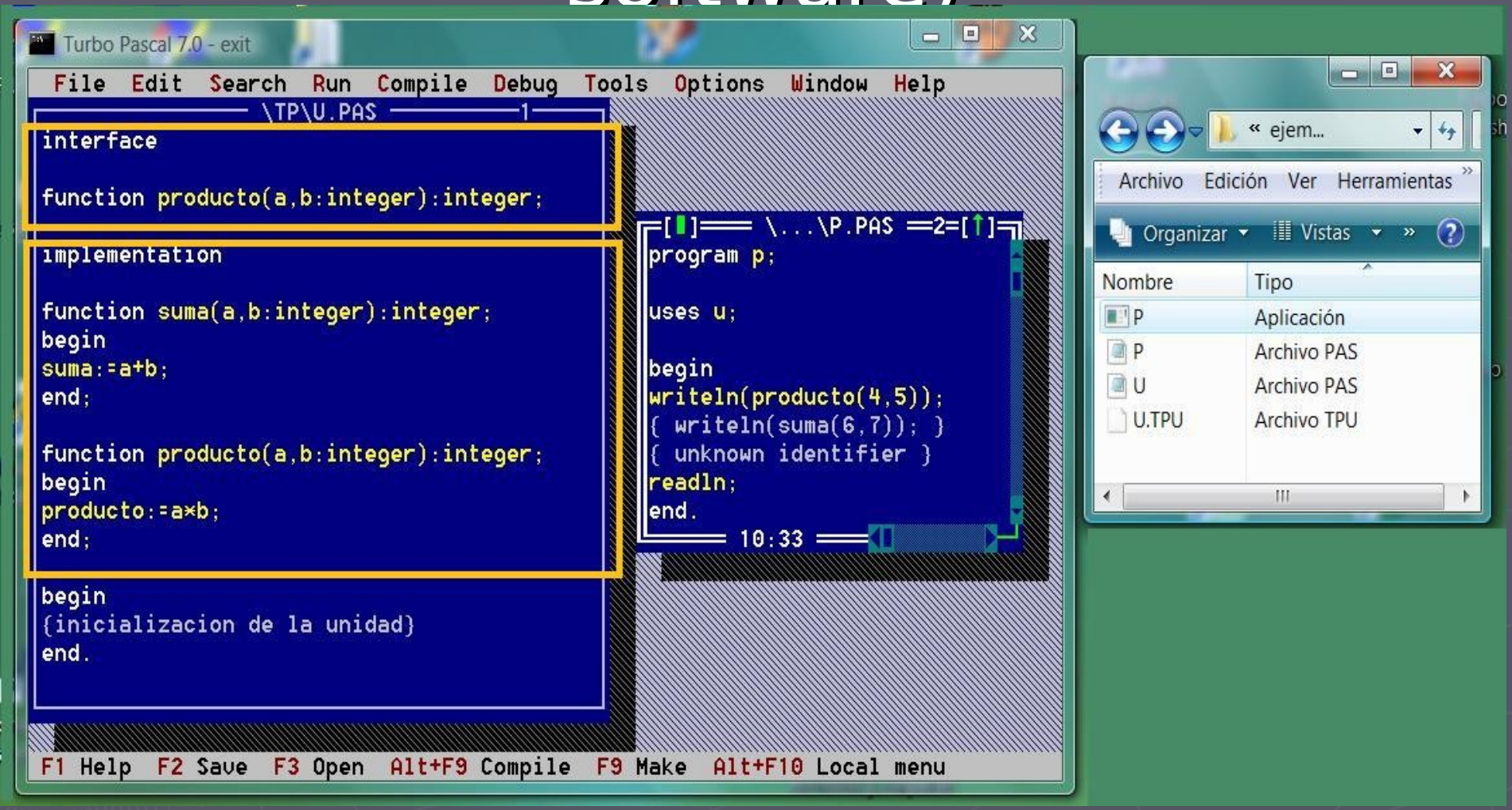
- b) Prácticas de la planificación
  - Objetivo: definir un plan para alcanzar las metas definidas con la actividad de comunicación.
  - La planificación debe proporcionar una guía útil y se debe producir con moderación

- c) Práctica del modelado
  - El modelo debe ser capaz de representar la información que el software transforma, la arquitectura y las funciones que permiten que ocurra la transformación, las características que desean los usuarios, y el comportamiento del sistema conforme se realiza la transformación
  - Los modelos cubren estos objetivos en diferentes grados de abstracción: primero al presentar el software desde el punto de vista del cliente (modelos de análisis) y después en un nivel más técnico (modelos de diseño).
  - Los modelos de análisis representan los requisitos del cliente en tres dominios (o “perspectivas”):
    - La información
    - Las funciones (dominio funcional)
    - El comportamiento

- Principios del modelado del análisis (los distintos métodos técnicos comparten estos principios):
  1. El dominio de **información** de un problema debe representarse y entenderse. Este dominio lo forman los datos que fluyen hacia el sistema, los que fluyen desde el sistema y los almacenamientos de datos
  2. Se deben definir las **funciones** que ejecuta el software: las funciones, de transformación o de control, proporcionan un beneficio directo al usuario. Las funciones se pueden describir en muchos grados de abstracción
  3. Se debe representar el **comportamiento** del software, motivado por unos eventos externos (“interacción con el exterior”).

- Los modelos de diseño representan:
  - La arquitectura
  - La interfaz de usuario
  - El detalle a nivel de componentes

Componente (unidad de software)



Principios del modelado del diseño

1. El diseño debe ser rastreable hasta el modelo de análisis:
   - La información se traduce en una arquitectura
   - Las funciones más importantes, en subsistemas de diseño
   - Las clases de análisis, en diseños al nivel de componentes
2. El establecimiento de una arquitectura del software, o esqueleto del sistema que se va a construir, afecta a la manera en que pueden realizarse las pruebas. Es una etapa previa a la consideración de los aspectos al nivel de componente
3. Las interfaces (internas y externas) deben diseñarse con cuidado: la forma en que se realiza el flujo de datos entre componentes puede evitar la propagación de errores, facilitar la integración, la realización de pruebas y la validación (comprobación) de las funciones de componentes.

4. Los componentes deben estar “conectados” entre sí (a través de interfaz de componente, mensajes, variables globales), de forma mínima, para evitar la propagación del error y facilitar el mantenimiento.
5. Las representaciones del diseño (modelos de diseño) deben ser comprensibles, pues su propósito es comunicar información a los profesionales que generan código, a quienes prueben el software y a quienes lo mantendrán

- e) Práctica de la Implantación
  - Abarca tres acciones:
    - Entrega
    - Soporte (documentación y soporte humano)
    - Retroalimentación: el software entregado representa un beneficio para el usuario final y una guía para que el equipo de software realice las modificaciones adecuadas

  - Se debe ensamblar y probar un paquete de entrega completo: p.ej DVD incluyendo software ejecutable y documentación, para que lo prueben los usuarios reales

Conjunto de tareas genéricas para la implantación / p.128

1. Crear paquetes de entrega
   - Ensamblar y probar todos los archivos ejecutables
   - Ensamblar y probar todos los archivos de datos
   - Crear y probar toda la documentación del usuario
   - Probar los paquetes de entrega con un grupo pequeño de usuarios representativos
2. Establecer la persona o grupo encargado del soporte humano
   - Establecer mecanismos para la localización e informe de los problemas
   - Establecer una base de datos para el informe de problemas/errores
3. Establecer mecanismos de retroalimentación del usuario
4. Distribuir los paquetes a todos los usuarios
5. Realizar las funciones de soporte continuas
6. Recopilar la retroalimentación del usuario

Retroalimentación del usuario