UNIVERSIDAD DE MURCIA

Fundamentos de Informática
3. Construcción de Software

Fundamentos de Informática
Grado en Ingeniería Química

2

Contenidos

- *Introducción*
- *Ciclo de vida del software*
- *Análisis*
- *Diseño*
  - *Algoritmos*
  - *Diagramas de Flujo*
  - *Pseudocódigos*
- *Codificación*
- *Pruebas*
- *Mantenimiento*

3

Introducción

Software pequeño – software grande

Software 1 : Cálculo del área de triángulos
Software 2 : Gestión de una ITV
Software 3 : Creación de una red social
Software 4: Gestión de cuentas de un banco

Software pequeño: pocas personas, poco código, poco hardware
Software grande: muchas personas, mucho código, mucho hardware

Necesidad de metodología

4

Ciclo de vida del software

5

Ciclo de vida del software

La solicitud del usuario

6

Ciclo de vida del software

Lo que entendió el lider del proyecto

7

3. Construcción de software

Ciclo de vida del software

El diseño del analista de sistemas

8

3. Construcción de software

Ciclo de vida del software

El enfoque del programador

9

3. Construcción de software

Ciclo de vida del software

La recomendación del consultor externo

10

3. Construcción de software

Ciclo de vida del software

La documentación del proyecto

11

3. Construcción de software

Ciclo de vida del software

La implantación de los programadores

12

3. Construcción de software

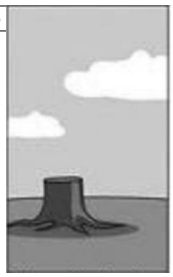
Ciclo de vida del software

El presupuesto del proyecto

13

Ciclo de vida del software

El soporte operativo



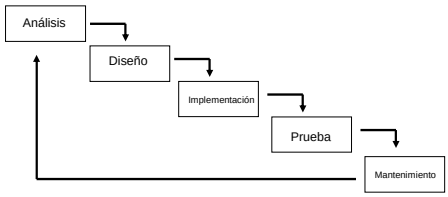
14

Ciclo de vida del software

Lo que el usuario realmente necesitaba

15

Ciclo de vida del software



16

Análisis

17

Análisis

¿Qué quiero hacer?

¿Qué entradas se necesitan?

¿Cuáles son las salidas deseadas?

¿Qué debe hacer el programa para obtener la salida a partir de la entrada?

18

Análisis

¿Qué entradas se necesitan?
- En detalle:
  - -Exactamente cuántos datos
  - -¿De qué tipo cada uno de ellos?
  - -¿Cuáles con los rangos permitidos?
  - -¿Hay excepciones en la entrada?

¿Usarán el sistema expertos o usuarios sin formación?
¿Qué interface será necesario?
¿Qué casos son posibles?
¿Es necesaria documentación? ¿Cuál?
¿Qué mejoras se introducirán probablemente en el futuro?
¿Cuánto de rápido debe ser el sistema?
¿Deberá modificarse mucho en el futuro?
¿Tengo toda la información para realizar el diseño exacto?
¿Necesito documentación técnica?

19

Análisis

Normalmente el usuario final no sabe exactamente qué quiere o no sabe expresarlo con precisión.

Es imprescindible tener mucha comunicación con el usuario, hasta que esté perfectamente claro qué se va a construir.

A veces se construyen prototipos (programas solo con el interfaz)

20

Análisis

Ejemplo : hacer un programa que diga si un año dado es bisiesto

Entrada: un año
un numero entero positivo

Salida: "SI" o "No"

Proceso: ¿Cómo se sabe si un año es bisiesto?

Un año es bisiesto si es divisible por 4, excepto el último de cada siglo (aquel divisible por 100), salvo que este último sea divisible por 400.

2011 2012 1900 1600

21

Análisis

Ejemplo 2: gestión de una ITV

Entradas: ¿?

Salidas: ¿?

Procesos: ¿?

22

Diseño

23

Diseño

El "Análisis" establece ¿qué? quiere hacerse

El "Diseño" establece ¿Cómo? se hace

Con exactitud, paso a paso, en detalle.

Se necesitan "Algoritmos"

Algoritmo

Método para resolver un problema mediante una serie de pasos precisos, definidos y finitos.

24

Diseño: algoritmos

Ej: cambiar una bombilla fundida

Comprobar si hay bombillas de repuesto
Si las hay
    Sustituir la fundida por la nueva
Si no las hay,
    Comprar una nueva en la tienda
    Sustituir la fundida por la nueva

25

Diseño: algoritmos

Mejora: cambiar una bombilla fundida

Comprobar si hay bombillas de repuesto

Si no hay
    Comprar una nueva en la tienda

Sustituir la fundida por la nueva

26

Diseño: algoritmos

Refinamiento: cambiar una bombilla fundida

Comprobar si hay bombillas de repuesto
    Ir al cajón de las bombillas
    Abrirlo
    Mirar si hay bombillas

Si no hay
    Abrir la puerta
    Bajar las escaleras
    Ir a la tienda
    Comprar una bombilla

Coger la bombilla
Coger una silla
Subirse a la silla
Colocar la bombilla en la lámpara

27

Diseño: algoritmos

Refinamiento: cambiar una bombilla fundida

¿Cómo se coloca una bombilla en la lámpara?

¿Cómo sé qué bombilla es la adecuada?

¿Y si la tienda está cerrada?

¿Hay que dejar después todo en su sitio?

28

Diseño: algoritmos

Los algoritmos están compuestos por:

Acciones, decisiones, repeticiones

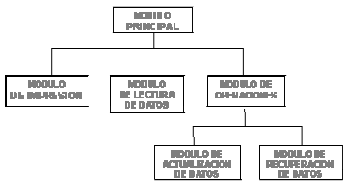
(Ej: fragmento de un algoritmo)
Ordenar la lista
Inicializar los contadores
Para todos los elementos de la lista
    Si el elemento es par
        Sumar uno

29

Diseño: algoritmos

Modularización:
Es recomendable descomponer un problema en partes más sencillas

Ej: Gestión de ITV



30

Diseño: pseudocódigo

- Lenguaje de especificación de algoritmos
- Lenguaje intermedio entre el lenguaje natural y el programa informático
- No hay un estándar en la notación
- No es tan rígido como un lenguaje de programación
- Nivel de abstracción a gusto del diseñador
- Permite fácilmente la modularidad
- Contiene palabras clave como “If” o “SI”, “Mientras” o “While”, etc.

- Muy útil para la fase de diseño, previo a implementar el programa.

- Nosotros lo usaremos para diseñar.

Diseño: pseudocódigo

También de un nivel de abstracción mayor

```
PROGRAMA Reordenación
VARIABLES
      A, B, C: vectores
INICIO
      Poner todos los valores de C a cero
      Ordenar las filas de A
      Ordenar las filas de B
      Mezclar A y B en C
      SI (suma(C)>200)
            Imprimir C
Fin Reordenación
```

```
PROGRAMA Mezclar
INICIO
      Para cada elemento de C
            . . .
Fin Mezclar
```


Codificación
(y compilación)

Codificación

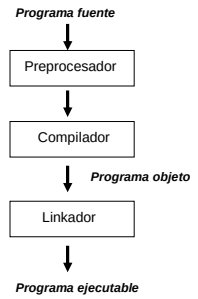
Paso del diseño teórico al ordenador

Escritura del "programa fuente" en el lenguaje de programación elegido, depuración inicial de errores y compilación.

Es importante documentar el programa, con documentación interna (comentarios) y externa (otros documentos).

Codificación

Se utiliza un "entorno de programación" : programa para editar, compilar, enlazar, etc. un código fuente.



Pruebas
(verificación y depuración)

Pruebas (verificación y depuración)

Fase de pruebas: verificación y depuración del programa

Se utilizan un conjunto amplio y variado de datos de entrada, para buscar posibles errores:
- Valores normales
- Valores extremos
- Valores especiales

Errores de compilación o interpretación: suelen ser errores de sintaxis
Errores de ejecución: ej. Divisiones por cero, desbordamiento de rangos, etc.
Errores lógicos: el programa no hace lo que debiera, está mal programado.

87

Mantenimiento
(y documentación)

88

Documentación y Mantenimiento

Documentación:
Escritura de las diferentes fases del ciclo de vida, esencialmente el análisis, diseño, y codificación, unidos a manuales de usuario y de referencia, así como normas para el mantenimiento.

Mantenimiento:
El programa se actualiza y modifica cada vez que sea necesario, de modo que se cumplan las necesidades cambiantes del usuario

A medida que se utiliza el programa, se anotan pequeños errores y posibles mejoras para introducirlos en la siguiente versión.

89

3. Construcción de software

- - - - -