Ingeniería del Software I

5 – Diseño detallado (Capítulo 8)

Diseño detallado

El diseño de alto nivel no especifica la lógica.

Esto es incumbencia del diseño detallado.

En este sentido, una notación textual provee mejor información:

Lenguaje natural:

Impreciso, ambiguo, conduce a problemas de comprensión.

Lenguajes formales (el otro extremo):

Usualmente tienen mucho detalle:

necesario para implementación, pero

no es importante para comunicar el diseño.

Estos detalles son usualmente un estorbo para la comprensión.

Diseño detallado y PDL

Process Design Language

Idealmente se desearía un lenguaje que:

- sea tan preciso como fuera posible,
- no requiera demasiado detalle,
- sea independiente del lenguaje de implementación,
- pueda convertirse fácilmente en la implementación.

=> PDL (Process Design Language).

PDL tiene la sintaxis externa de un lenguaje de programación estructurado (ej: Pascal, C, ...),

pero el vocabulario de un lenguaje natural,

(podría pensarse como un "inglés estructurado" o "castellano estructurado").

Diseño detallado y PDL

PDL

- PDL podría utilizarse para especificar el diseño completo (desde la arquitectura hasta la lógica).
- El grado de detalle deseado es decisión del diseñador.
- PDL es particularmente útil conjuntamente con la técnica de refinamiento top-down.
- Cierto procesamiento automatizado sobre PDL es posible.

Diseño detallado y PDL

PDL

Ej.: Determinar el mínimo y el máximo de un conjunto de números en un archivo:

```
minmax (in file) ARRAY a

DO UNTIL end of input
        READ an item into a
ENDDO
max, min := first item of a
DO FOR each item in a
        IF max < item THEN set max to item
        IF min > item THEN set min to item
ENDDO END
```

Diseño detallado y PDL

PDL

- PDL captura la lógica completa del procedimiento,
  pero revela pocos detalles de implementación.
- Para llevarlo a una implementación cada una de las pseudo-sentencias deberán convertirse en sentencias del lenguaje de programación.
- En PDL, el diseño puede expresarse en el nivel de detalle más adecuado para el problema.
- PDL permite un enfoque de refinamientos sucesivos.

Diseño detallado y PDL

PDL

Constructores básicos de PDL:

- Constructores IF-THEN-ELSE como en Pascal.
- Las condiciones y las sentencias no necesitan escribirse formalmente.
- Una sentencia general CASE. Ej.:

```
CASE OF transaction type
CASE OF operator type
```

- Un constructor DO utilizado para indicar repetición:

```
DO iteration criteria
   one or more sentences
ENDDO
```

Diseño detallado y PDL

PDL

Constructores básicos de PDL (continuación):

- El criterio de iteración no necesita establecerse formalmente.
  Ej.:

```
DO WHILE there are chars in the input ...
DO UNTIL the end of file is reached ...
DO FOR each item in the list EXCEPT when item is zero ...
```

- Se puede definir y utilizar una gran variedad de estructuras de datos: listas, tablas, escalares, registros, arreglos, etc. Todos los constructores deben ser programables.

Diseño detallado y PDL

Diseño de la lógica/algoritmo

El objetivo básico del diseño detallado es especificar la lógica de los distintos módulos especificados en el diseño del sistema.

i.e. diseñar los algoritmos.

- No existen procedimientos claros para ello.

  => Heurísticas y métodos.

- El método más común es el refinamiento paso a paso.
- El desarrollo se realiza gradualmente:

  Comenzar por convertir la especificación del módulo en una descripción abstracta del algoritmo.

  En cada paso, descomponer una o más sentencias del algoritmo actual en instrucciones más detalladas.

  Terminar cuando todas las instrucciones son lo suficientemente precisas como para llevarlas fácilmente al lenguaje de programación.

Diseño detallado y PDL

Diseño de la lógica/algoritmo

El objetivo básico del diseño detallado es especificar la lógica de los distintos módulos especificados en el diseño del sistema.

i.e. diseñar los algoritmos.

- No existen procedimientos claros para ello.

  => Heurísticas y métodos.

- El método más común es el refinamiento paso a paso.
- El desarrollo se realiza gradualmente:

  Comenzar por convertir la especificación del módulo en una descripción abstracta del algoritmo.

  En cada paso, descomponer una o más sentencias del algoritmo actual en instrucciones más detalladas.

  Terminar cuando todas las instrucciones son lo suficientemente precisas como para llevarlas fácilmente al lenguaje de programación.

Durante el refinamiento, se debe refinar tanto las instrucciones como los datos.

Diseño detallado y PDL

Diseño de la lógica/algoritmo

El objetivo básico del diseño detallado es especificar la lógica de los distintos módulos especificados en el diseño del sistema.

i.e. diseñar los algoritmos.

- No existen procedimientos claros para ello.

  => Heurísticas y métodos.

- El método más común es el refinamiento paso a paso.
- El desarrollo se realiza gradualmente:

  Comenzar por convertir la especificación del módulo en una descripción abstracta del algoritmo.

  En cada paso, descomponer una o más sentencias del algoritmo actual en instrucciones más detalladas.

  Terminar cuando todas las instrucciones son lo suficientemente precisas como para llevarlas fácilmente al lenguaje de programación.

Pauta a seguir: en cada paso, la cantidad de descomposición debe ser tal que pueda manipularse fácilmente y represente una o dos decisiones de diseño.

Diseño detallado y PDL

Diseño de la lógica/algoritmo

Ej.: selection sort

```
1 Sea n la longitud del arreglo a ordenar a;
2 i := 1 ;
3 DO WHILE i < n
4         encontrar el menor de ai...an,
                  e intercambiarlo con el
                  elemento de la posición i;
5         i := i + 1;
6 ENDDO;
```

Diseño detallado y PDL

Diseño de la lógica/algoritmo

Ej.: selection sort

```
1   Sea n la longitud del arreglo a ordenar a;
2   i := 1 ;
3   DO WHILE i < n
4.1         j := n;
4.2         DO WHILE j > i
4.3                 IF a(i) > a(j) THEN
4.4                         intercambiar los elementos
                                    en las posiciones j e i;
4.5                 ENDIF;
4.6                 j := j - 1;
4.7         ENDDO;
5           i := i + 1;
6   ENDDO;
```


Diseño detallado y PDL

Diseño de la lógica/algoritmo

Ej.: selection sort


```
1   Sea n la longitud del arreglo a ordenar a;
2   i := 1 ;
3   DO WHILE i < n
4.1         j := n;
4.2         DO WHILE j > i
4.3                 IF a(i) > a(j) THEN
4.4.1                       x := a(i);
4.4.2                       a(i) := a(j);
4.4.3                       a(j) := x;
4.5                 ENDIF;
4.6                 j := j - 1;
4.7         ENDDO;
5           i := i + 1;
6   ENDDO;
```

Verificación

Objetivo: Mostrar que el diseño detallado cumple con las especificaciones dadas en el diseño del sistema.

Tres métodos de verificación:

- Recorrido del diseño.
- Revisión crítica del diseño.
- Verificadores de consistencia.

Verificación

Objetivo: Mostrar que el diseño d
dadas en el diseño del sistema.

Tres métodos de verificación:

- Recorrido del diseño.
- Revisión crítica del diseño.
- Verificadores de consistencia.

Básicamente es una reunión informal entre el diseñador y el líder (u otro diseñador) donde el autor explica el diseño paso a paso a la otra persona.

Verificación

Objetivo: Mostrar que el diseño de
dadas en el diseño del sistema.

Básicamente es una r
infor

Ver el libro

Sigue un proceso de revisión estándar.
El uso de listas de control es importante.

Tres métodos de verificación:

- Recorrido del diseño.
- Revisión crítica del diseño.
- Verificadores de consistencia.

Verificación

Objetivo: Mostrar que el diseño de
dadas en el diseño del sistema.

P

Solo si el diseño se realiza en PDL o algún lenguaje formal.
Asegura consistencia automáticamente (ej.: ¿existen los módulos invocados?, ¿respetan las interfaces?, etc.).
Mayor detalle es posible dependiendo (de la formalidad) del lenguaje.

Tres métodos de verificación:

- Recorrido del diseño.
- Revisión crítica del diseño.
- Verificadores de consistencia.

Métricas

- El diseño detallado provee muchos detalles de lógica de control y estructura de datos.
- Solo omite detalles de implementación específico del lenguaje de programación utilizado.
- Muchas métricas tradicionalmente destinadas al código son también útiles en el diseño detallado.

- Algunas métricas:
  - Complejidad ciclomática.
  - Vínculos de datos.
  - Métrica de cohesión.

Métricas

- El diseño detallado provee muchos detalles de lógica de control y estructura de datos.
- Solo omite detalles de implementación
  programación utilizado.
- Muchas métricas tradicionalme
  útiles en el diseño detallado.

Mide la complejidad de un módulo.
Depende de las condiciones y sentencias de control.
A medida que estas aumentan, la complejidad aumenta.

- Algunas métricas:
  - Complejidad ciclomática.
  - Vínculos de datos.
  - Métrica de cohesión.

Métricas

- El diseño detallado provee muchos detalles de lógica de control y estructura de datos.
- Solo omite detalles de implement programación utilizado.
- Muchas métricas tradicion útiles en el diseño detalla
- Algunas métricas:
  - Complejidad ciclomática
  - Vínculos de datos.
  - Métrica de cohesión.

Los distintos módulos están vinculados por los datos que se pasan en las invocaciones. Estos vínculos determinan el acoplamiento. Estas métrica capturan la interacción de datos entre las distintas porciones del sw.

Métricas

- El diseño detallado provee muchos detalles de lógica de control y estructura de datos.
- Solo omite detalles de implement programación utilizado.
- Muchas métricas tradicion útiles en el diseño detalla
- Algunas métricas:
  - Complejidad ciclomática
  - Vínculos de datos.
  - Métrica de cohesión.

Los distintos módulos están vincula

Mide la dependencia de los distintos elementos del módulo. El valor será más alto si cada ejecución posible del módulo usa todos los recursos (variables) del módulo.