Diseño detallado

Especifica la lógica.

Process Design Language PLD

Típicamente se usan *lenguajes naturales* pero son imprecisos, ambiguos, y conducen a problemas de comprensión. En el otro extremo están los *lenguajes formales*, pero son muy complejos y tienen demasiado detalle.

Idealmente queremos:

- Sea tan preciso como sea posible
- No requiera demasiado detalle
- Sea independiente del lenguaje de implementación: Pero no tanto en la práctica porque nuestro cerebro siempre va a estar resolviendo los problemas con las herramientas (lenguajes) que más usamos. Además el PLD es bastante funcional.
- Pueda convertirse fácilmente en la implementación

PLD tiene la sintaxis externa de un lenguaje de programación estructurado, pero el vocabulario de un lenguaje natural.

Existen algunos automatizadores de PLD, para ciertos lenguajes. Son generadores de código.

PLD es un tipo de pseudocódigo; captura la lógica completa del procedimiento, aunque revela pocos detalles de implementación.

En PLD, el diseño puede expresarse en el nivel de detalle más adecuado para el problema y permite un enfoque de *refinamientos* sucesivos del lenguaje natural a la *formalización* de la descripción. Se deben refinar tanto las instrucciones como los datos.

Constructores básicos de PLD

- `IF-THEN-ELSE`
- `CASE`
- `DO (WHILE/UNTIL/FOR) criterio (EXCEPT criterio) sentencia ENDDO`

El criterio de iteración, condiciones y sentencias no necesitan establecerse formalmente.
Es importante que el lenguaje natural utilizado sea acotado, por ejemplo evitar tiempos verbales.

Verificación

El objetivo es mostrar que el diseño detallado cumple con las especificaciones dadas en el diseño del sistema.

Métodos de verificación

- *Recorrido del diseño*: Reunión informal entre diseñador y líder donde el autor *explica* el diseño paso a paso.
- *Revisión crítica del diseño*: Asegura que el diseño detallado satisface las especificaciones que se propusieron durante el diseño de alto nivel. Se trata de una reunión entre diseñador detallado, diseñador de alto nivel y programador. Se usan listas de control.
- *Verificadores de consistencia*: Si el diseño se realiza en PLD o en algún lenguaje formal, asegura consistencia automáticamente.

Métricas

- *Complejidad ciclomática*: Mide la complejidad de un módulo. Depende de las condiciones y sentencias de control.
- *Vínculos de datos*: Captura la interacción de datos entre las distintas porciones del software. Estas interacciones determinan el acoplamiento.
- *Métricas de cohesión*: Mide la dependencia de los distintos elementos dentro de un módulo. El valor aumenta si cada ejecución posible del módulo usa todos los recursos del módulo.