
TEMA 3 DISSENY DEL SOFTWARE

1. Introducció.

- Procés de disseny.
- Disseny de dades, disseny arquitectònic, disseny de la interfície, disseny procedimental.
- Principis (objectius) del disseny.

2. Conceptes del disseny.

- Abstracció.
- Modularitat.
- Refinament.

3. Disseny modular efectiu.

- Independència funcional.
 - Cohesió
 - Acoblament.
 - Heurístiques per a un disseny modular efectiu.
-

3.1

Introducció

- **Definició.** El procés d'aplicar diferents tècniques i principis amb el propòsit de definir un dispositiu, un procés o un sistema amb suficients nivells de detall com per permetre la seva realització física.

Està situat en el nucli tècnic del procés d'E.S. I s'aplica independentment del paradigma de desenvolupament utilitzat.

Domini del Problema => Domini de la Solució

- **Objectiu.** Produir un model o representació del sistema que pugui ser utilitzat, en una fase posterior, a fi d'implementar-lo.

- **Disseny preliminar:** centrat en la transformació dels requeriments de les dades i en l'arquitectura del software (estructura modular).
- **Disseny detallat:** conceptes més específics, refinament de l'estructura de dades i representació algorísmica dels mòduls.

	D. Preliminar	D. Detallat
D. Dades	X	X
D. Arquitectònic	X	
D. Procedimental		X
D. Interfície	X	

→ Moltes vegades es fa prèviament

3

D. Dades, D. Arquit., D. Procedural, D. Interf.

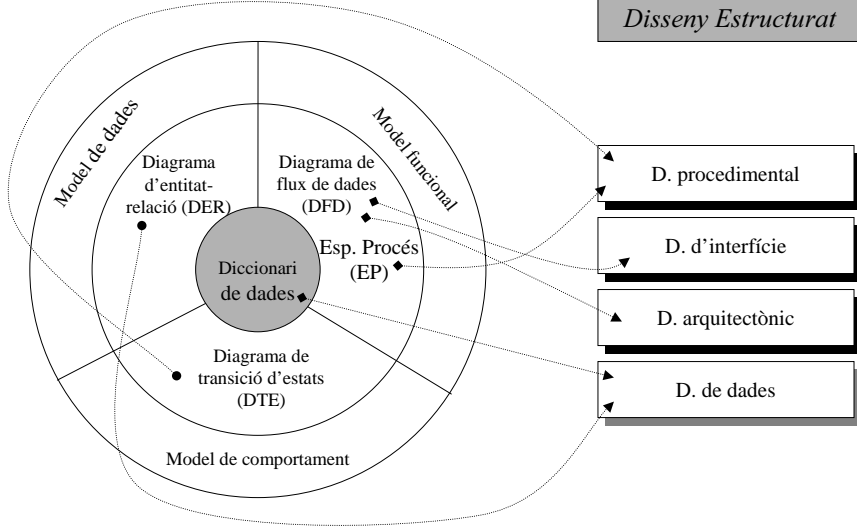
- **D. Dades.** Transforma el model de dades de l'anàlisi (diccionari, en l'estructurat o els atributs, en l'OO), en les estructures de dades que després s'implementaran.
- **D. Arquitectònic.** Defineix la relació entre els principals elements estructurals del programa. Es desenvolupa una estructura modular i jeràrquica del software.
- **D. Procedimental.** Transforma els elements estructurals de l'arquitectura del programa en una descripció procedimental dels elements del software. Descripció de les funcions a un nivell més proper al LP que el nivell de les EP. de l'anàlisi.
- **D. Interfície.** Descriu com es comunica el software amb ell mateix, amb els sistemes que operen amb ell i amb els operadors que l'utilitzen.

4

3.1

D. Dades, D. Arquit., D. Procedim., D. Interf.

Disseny Estructurat

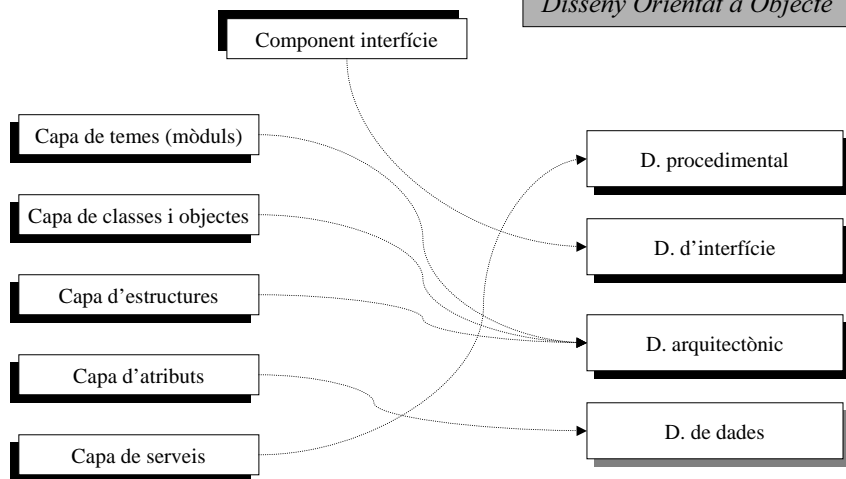


5

3.1

D. Dades, D. Arquit., D. Procedim., D. Interf.

Disseny Orientat a Objecte



6

Principis (objectius) del disseny

- **Verificabilitat.** Poder avaluar la correctesa del disseny.
- **Completesa.** Totes les components (estructures de dades, mòduls, interfícies externes, etc.) han de ser especificades.
- **Consistència.** Que no hi hagi inconsistències inherents.
- **Eficiència.** Utilització adequada dels recursos del sistema. Els recursos no són infinits.
- **Seguible.** Tots els elements del disseny han de poder-se “mapejar” cap a l’anàlisi de requeriments.
- **Simplicitat/Comprensible.** Cal tenir en compte que el document de disseny, igual com era el d’especificació de requeriments, serà utilitzat com a base per a les etapes posteriors.

Principis (objectius) del disseny

- El procés de disseny no ha de ser únic, s’han d’avaluar alternatives.
- Cada element del model de disseny s’ha de poder seguir enrere fins a l’anàlisi per saber els requeriments que satisfà.
- El disseny no ha d’inventar res de nou. Reutilitzar al màxim.
- Minimitzar la distància entre DP i DS.
- Ha de ser uniforme (un estil constant).
- Ha d’estructurar-se per admetre canvis.
- Ha de ser robust. Acceptar circumstàncies inusuals.
- El disseny no és escriure codi i escriure codi no és dissenyar.
- La qualitat del disseny s’avalua mentre es fa, no després.
- Cal revisar-lo per evitar errors conceptuals (semàntics): omissions, ambigüitats, inconsistències.

Conceptes del disseny

- **Abstracció.**
- **Modularitat.**
- **Refinament.**
- **Arquitectura del software.** Estructura global del software i la manera com aquesta estructura proporciona integritat conceptual a un sistema.
- **Jerarquia de control.** Estructura de programa.
- **Partició estructural.** L'estructura de programa es pot partir horitzontalment o verticalment.
- **Estructura de dades.** Relació lògica entre els elements individuals de dades.
- **Procediment del software.** Detalls de processament de cada mòdul individualment.
- **Ocultació de la informació.** Els mòduls s'han de dissenyar de manera que la informació (dades i processos) sigui inaccessible per altres mòduls que no la necessitin.

Abstracció

Permet definir components de manera abstracta a diferents nivells, sense preocupar-nos dels seus detalls d'implementació. Una abstracció descriu el *comportament extern* de la component, sense haver de parar atenció als detalls interns que produeixen el comportament.

L'abstracció és un ajut a la modularització ja que permet veure el comportament extern dels mòduls a fi de connectar-los.

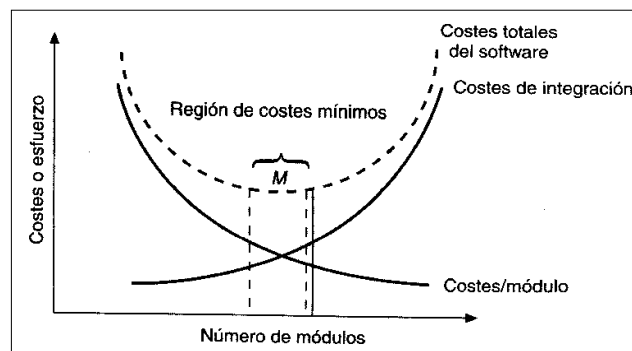
- estructurat
 - **Abstracció Funcional.** Abstracció d'una component des del punt de vista de la funció que realitza. Una seqüència d'instruccions que té una funció específica i limitada.
Exemple: porta => obrir, tancar.
- O.O.
 - **Abstracció de dades.** Col·lecció de dades que descriuen un objecte de dades.
Exemple: porta => tipus, pes, dimensions, direcció d'obertura, etc.
- temps real
 - **Abstracció de control.** Implica un mecanisme de control del programa sense especificar detalls interns.
Exemple: porta => si es prem el botó d'obrir llavors...

- Aplicar el principi de *divide&conquer* al disseny de software.
- **Mòdul.** És la unitat bàsica d'un programa, és finita i identificable respecte a la compilació i la lectura. És lògicament separable. Pot ser una macro, una funció, un conjunt de dades i funcions associades, un tipus abstracte de dades, etc.
- Mida recomanable d'un mòdul:
 - $C(p)$ grau de complexitat d'un problema p .
 - $E(p)$ esforç per programar una solució.
 - $C(p_1) > C(p_2) \Rightarrow E(p_1) > E(p_2)$
 - $C(p_1+p_2) > C(p_1) + C(p_2) \Rightarrow E(p_1+p_2) > E(p_1) + E(p_2)$

Però no es pot dividir infinitament.

Dividir implica augmentar la interdependència entre mòduls:

- Augmenta el cost associat a crear les interfícies.
- Costa més detectar l'origen dels errors i corregir-los.



Solució (criteri) per arribar a $M \Rightarrow \uparrow$ Cohesió, \downarrow Acoblament

- Dues estratègies per dissenyar una jerarquia de mòduls (arquitectura del software):
- **Top-down.** Refinaments successius. Es comença per un nivell més abstracte i a cada pas obtenim noves components que ofereixen una visió més concreta.
- **Bottom-up.** Es comença des de les components de més baix nivell de la jerarquia i de manera progressiva establim les components de nivell més alt.
- **Estratègia mixta.** Es comença des dels dos extrems de la jerarquia fins que es troben.

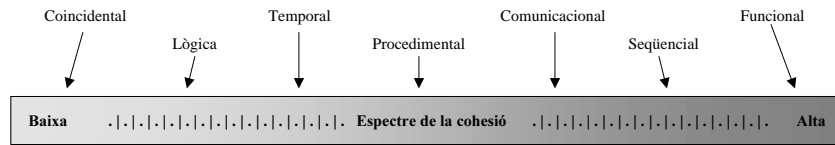
- **Independència funcional.** Un sistema es considera modular si està format per una sèrie de components de manera que cadascuna proporcioni una bona definició d'abstracció, i que un canvi introduït sobre una component tingui un impacte mínim sobre la resta.
La modularitat efectiva s'aconsegueix amb independència funcional, que vol dir mòduls amb una funció única i poca interacció entre ells.
- **Cohesió.** Cada mòdul té una funció única. És l'extensió del principi de la ocultació.
- **Acoblament.** És una mesura de la interdependència entre mòduls.

Objectiu: màxima cohesió i mínim acoblament.

\uparrow Cohesió \Rightarrow \downarrow Acoblament

3.3

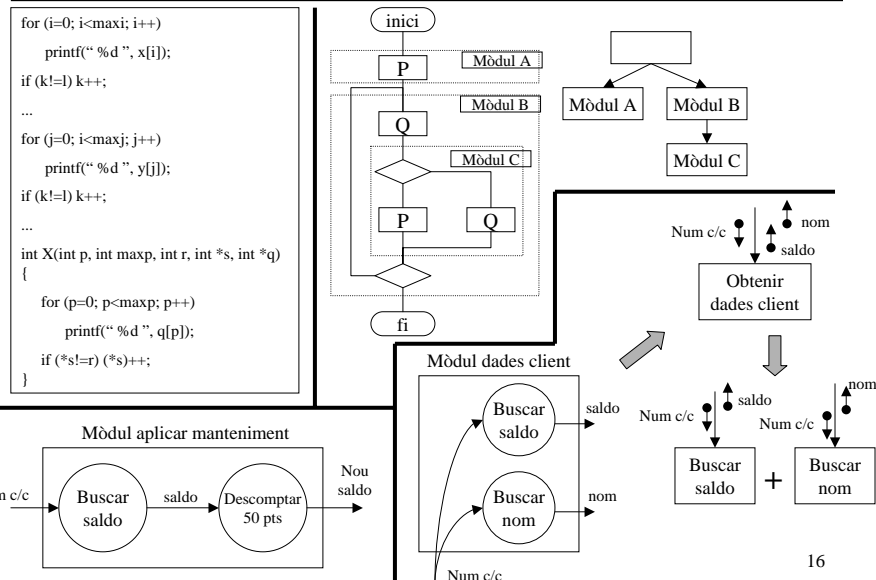
Cohesió



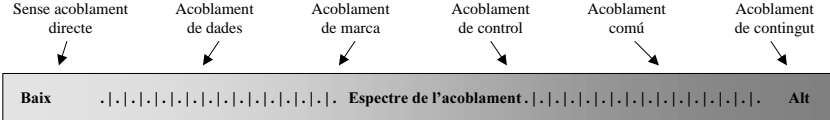
- **Coincidental.** Tasques poc relacionades. Per exemple, partir un programa arbitràriament en mòduls cada x línies.
- **Lògica.** Tasques relacionades lògicament. Per exemple, un mòdul amb totes les funcions d'entrada/sortida.
- **Temporal.** Tasques que s'han d'executar durant el mateix període de temps. Per exemple, alliberar memòria, tancar fitxers, etc.
- **Procedimental.** Partició a partir del diagrama de flux, és a dir, elements relacionats que s'executen en un ordre específic.
- **Comunicacional.** Tasques que operen sobre les mateixes dades.
- **Seqüencial.** Les dades resultat de cada element de processament del mòdul són l'entrada del següent element.
- **Funcional.** Tots els elements del mòdul es relacionen per realitzar una única funció.

3.3

Cohesió



Acoblament



- **Acoblament de dades.** Dos mòduls es comuniquen amb paràmetres on cadascun és una dada simple o bé un conjunt de dades homogènies.
- **Acoblament de marca (per registre).** Un mòdul passa a un altre un registre o estructura de dades amb no homogènia (amb diferents camps). Cal una comprovació de les estructures de dades. Per alta banda, cal evitar passar registres amb molts camps dels quals només se n'utilitza algun.
- **Acoblament de control.** Un mòdul passa un paràmetre a un altre amb intenció de controlar el seu comportament. És més acceptable que el mòdul inferior doni un control al superior (per exemple, notificació d'error).
- **Acoblament comú.** Diversos mòduls accedeixen a la mateixa àrea global de dades.
 - Qualsevol error en un mòdul pot afectar a un altre via dades globals.
 - Difícil seguir el programa pel que fa a qui modifica les dades globals.
 - Manteniment difícil si es canvia l'estructura de dades global (molts mòduls hi accedeixen).
 - Si hi ha moltes dades globals, és difícil establir quines dades utilitza cada mòdul.
- **Acoblament de contingut.** Un mòdul accedeix directament a una part de l'altre mòdul (sentències tipus *go to*).

Acoblament

