GRAU D'ENGINYERIA INFORMÀTICA

PROGRAMACIÓ II
CURS 12-13

Bloc 1:

Mòdul i abstracció de dades (1)

Laura Igual
Departament de Matemàtica Aplicada i Anàlisi
Facultat de Matemàtiques
Universitat de Barcelona

UNIVERSITAT DE BARCELONA

Temari Teoria

Bloc 1: Concepte de mòdul i abstracció de dades

Bloc 2: Programació orientada a objectes

Bloc 3: Programació orientada a esdeveniments

Índex Bloc 1:
Mòdul i abstracció de dades

1. Introducció
2. Complexitat intrínseca de les aplicacions
3. Descomposició de problemes complexos
4. Factors de qualitat
5. Modularitat
    1. Descomposició funcional
    2. Descomposició basada en objectes
6. TADs
7. Exemples de metodologies del software.

INTRODUCCIÓ

Introducció

- Anys 50,
  - Apareixen els primers llenguatges de programació.
  - Programes petits utilitzant assaig i error
- Finals dels anys 60:
  - Hardware més potent ➡ augmenta la dificultat dels problemes abordats
  - Sistemes grans sense estructuració ➡
    - Impossibles de mantenir
    - Molt costosos i escassament fiables
  - Necessitat d'ordenar i detallar el camí de resolució del problema abans de començar a programar.
  - Apareix l'*Enginyeria del Software*

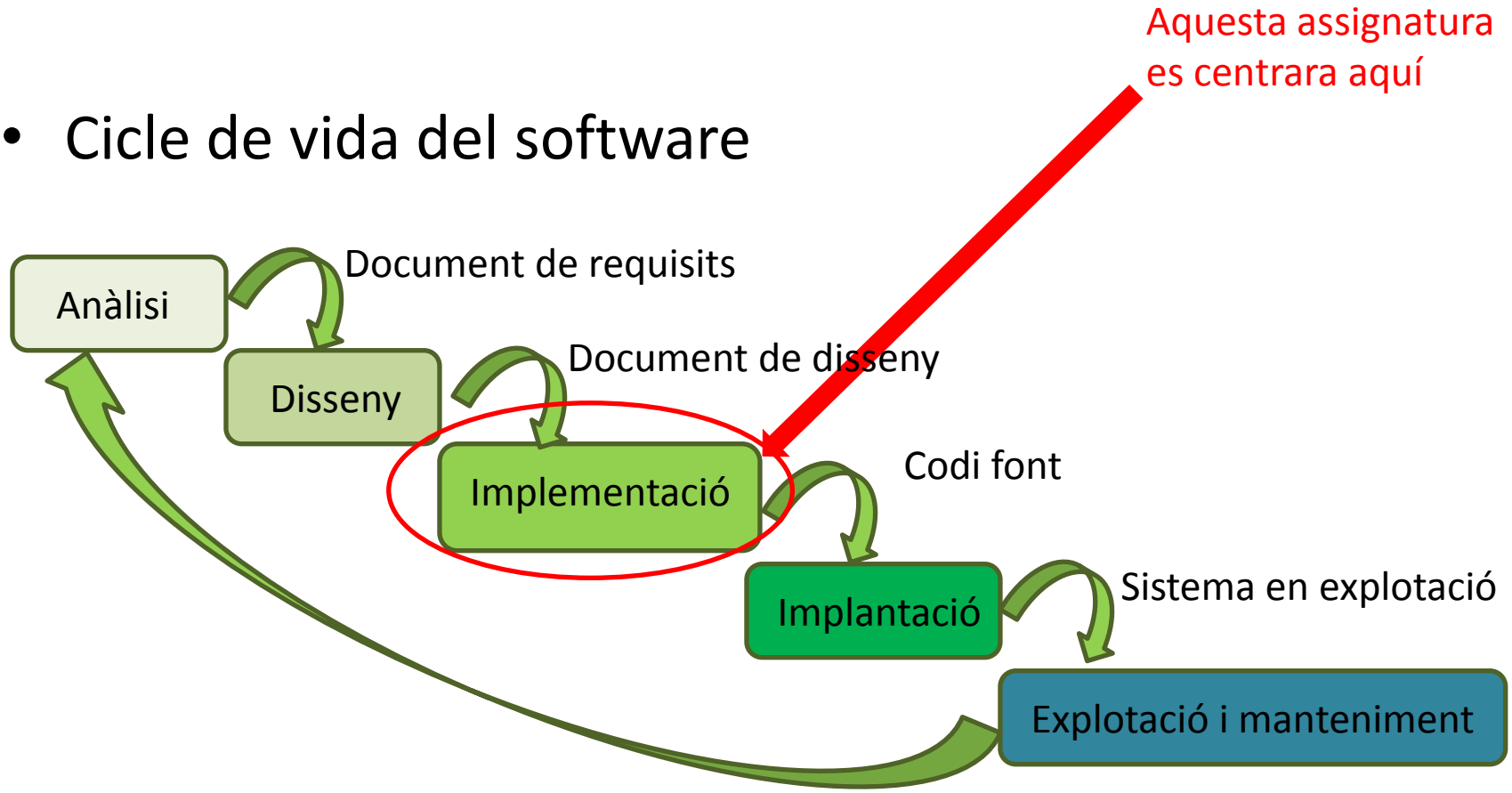
    ... evoluciona fins avui en dia

Introducció

- L'objectiu general de la **Enginyeria del Software** és produir software de qualitat
- Per qualitat s'entén l'adequació del software als requisits exigits
- El camí per a obtenir software de qualitat és mitjançant un plantejament rigorós del problema

Introducció

- Cicle de vida del software



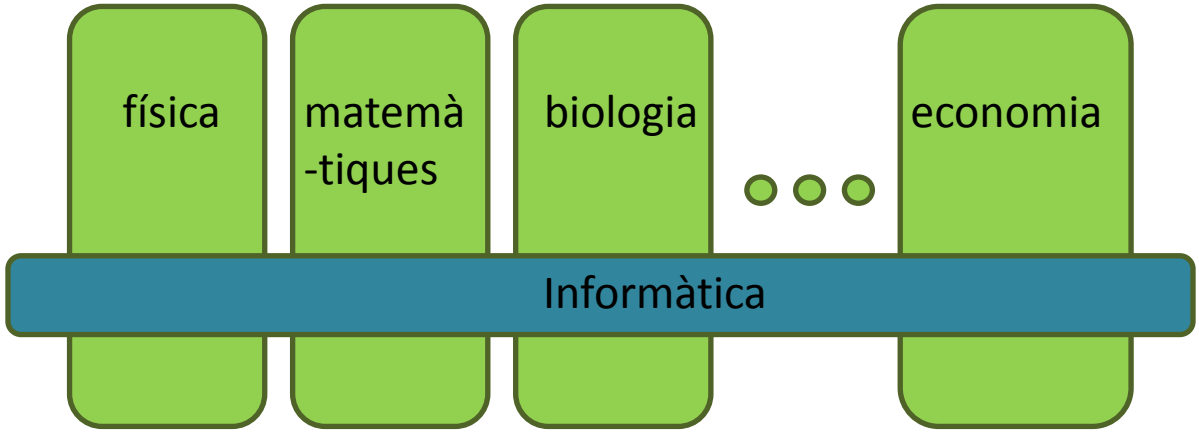
El **Cicle de vida del software o procés de desenvolupament de software** és aquell en el que les necessitats de l'usuari són traduïdes en requisits de software, aquestos transformats en disseny i el disseny implementat en codi. Després ve la implantació i explotació que porta de nou a l'anàlisi.

COMPLEXITAT INTRÍNSECA DE LES APLICACIONS

Complexitat intrínseca de les aplicacions informàtiques

1. El domini del problema.

- La informàtica és una disciplina transversal
- Tracta problemes de caràcter interdisciplinar



A més,

- Dificultat en transmetre necessitats i expectatives dels usuaris
- Canvis en els requisits durant el desenvolupament
  - Per exemple: requisits d'un sistema de navegació aèria

Complexitat intrínseca de les aplicacions informàtiques

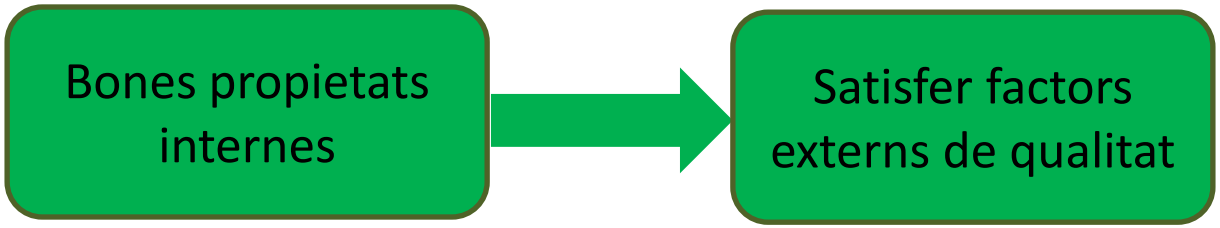
2. Sistemes grans.
   - Pot haver desenes de desenvolupadors implicats que, a més, poden estar separats geogràficament
   - Problemes de comunicació i coordinació
3. Parts comunes entre aplicacions són molt sovint dificilment reutilitzables.
4. Dificultat per caracteritzar el comportament de sistemes discrets.
   - Nombre molt gran d'estats, per això les proves no són completes.

Descomposició de problemes complexos

- No podem fer que la dificultat desaparegui, però, podem desenvolupar tècniques i eines que ens permetin **tractar-la i gestionar-la.**
  - Amb aquesta finalitat, definim **factors de qualitat del programari extens i interns**.

Descomposició de problemes complexos

- **Factors externs:**
  - Poden ser detectats per els usuaris
  - Qualitat externa es la que realment preocupa
- **Factors interns:**
  - Només es percebuts pels dissenyadors i implementadors
  - Mitjà per aconseguir la qualitat externa
- Objectiu:



FACTORS DE QUALITAT

Factors de qualitat externs

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- Compatibilitat
- Eficiència
- Portabilitat
- Facilitat d’us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Correcció:

  Capacitat dels productes de software per a realitzar amb exactitud les seves tasques tal i com es defineixen en les especificacions.

Factors de qualitat

- Correcció
- **Robustesa**
- Extensibilitat
- Reutilització
- Compatibilitat
- Eficiència
- Portabilitat
- Facilitat d'us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Robustesa:

  Capacitat dels sistemes de reaccionar apropiadament davant condicions excepcionals.

Factors de qualitat

- Correcció
- Robustesa
- **Extensibilitat**
- Reutilització
- Compatibilitat
- Eficiència
- Portabilitat
- Facilitat d’us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Extensibilitat:

  Facilitat d'adaptar els productes de software als canvis d'especificació.

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- **Reutilització**
- Compatibilitat
- Eficiència
- Portabilitat
- Facilitat d’us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Reutilització:

  Capacitat dels elements de software de servir per a la construcció de diferents aplicacions.

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- **Compatibilitat**
- Eficiència
- Portabilitat
- Facilitat d’us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Compatibilitat:

  Facilitat de combinar uns elements de software amb altres.

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- Compatibilitat
- **Eficiència**
- Portabilitat
- Facilitat d’us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Eficiència

  Capacitat d'un sistema software per exigir la menor quantitat possible de recursos hardware, tals com temps del processador, espai ocupat de memòria interna i externa o ample de banda utilitzat en els dispositius de comunicació

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- Compatibilitat
- Eficiència
- **Portabilitat**
- Facilitat d'us
- Funcionalitat
- Oportunitat

Factors de qualitat

- Portabilitat

  Facilitat de transferir els productes software a diferents entorns hardware i software

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- Compatibilitat
- Eficiència
- Portabilitat
- **Facilitat d'us**
- Funcionalitat
- Oportunitat

Factors de qualitat

- Facilitat d’us

  Facilitat amb la qual persones amb diferents formacions i aptituds poden aprendre a utilitzar els productes software i aplicar-los a la resolució de problemes.

  També cobreix la facilitat d'instal·lació, d’operació i de supervisió.

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- Compatibilitat
- Eficiència
- Portabilitat
- Facilitat d’us
- **Funcionalitat**
- Oportunitat

Factors de qualitat

- Funcionalitat

  Conjunt de possibilitats que proporciona un sistema.

Factors de qualitat

- Correcció
- Robustesa
- Extensibilitat
- Reutilització
- Compatibilitat
- Eficiència
- Portabilitat
- Facilitat d’us
- Funcionalitat
- **Oportunitat**

Factors de qualitat

- Oportunitat

  Capacitat d'un sistema de software de ser llançat quan els usuaris ho desitgen o abans.

Factors de qualitat

- **Reutilització**
- Beneficis esperats:
  - Oportunitat
  - Fiabilitat
  - Eficiència
  - Consistència
  - Inversió

- Reutilitzar codi de qualitat
- Imitar a bons programadors/dissenyadors

"S'ha de ser consumidor de reutilització abans de ser productor."

Factors de qualitat

Qualitats claus:

- Correcció
- Robustesa

Fiabilitat

- Reutilització
- Extensibilitat

→ Modularitat

(Factor intern)

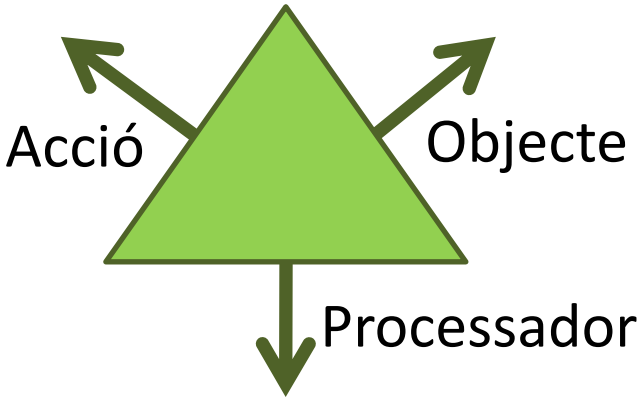
Introducció a la Modularitat

- Objectius principals a l'hora de fer disseny del software:
  1. Fiabilitat
  2. Modularitat

- Requereixen mètodes sistemàtics de descomposició dels sistemes en mòduls

Modularitat

- Quins criteris s'han d'utilitzar per trobar els mòduls del nostre software?
  - Les 3 forces de la computació:



En resum: "executar un sistema de software és utilitzar certs processadors per aplicar certes operacions a certs objectes."

Acció → què fa un sistema
Objecte → A qui li ho fa

- Mòduls com:
  - Unitats de descomposició funcional
  - Unitats basades en els principals tipus d'objectes

Modularitat

- Diferència entre
  - **Els enfocaments tradicionals** construeixen cada mòdul sobre alguna **unitat de descomposició funcional**- un cert aspecte de la acció
  - **L'enfocament orientat a objectes** construeix cada mòdul al voltant **d'algun tipus d'objecte.**

Modularitat

- Element clau:

  Continuïtat modular

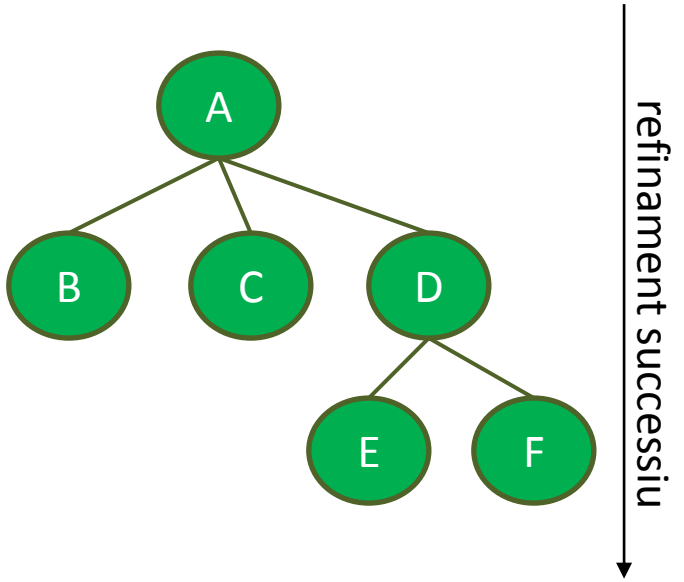
  – Un mètode de disseny satisfà aquest criteri si ens proporciona **arquitectures estables**.

    - Mantenen la quantitat de canvi en el disseny proporcional al tamany dels canvis en l'especificació (que s'ha definit prèviament).

  – Important, si es considera l'evolució del sistema a llarg termini.

Descomposició funcional

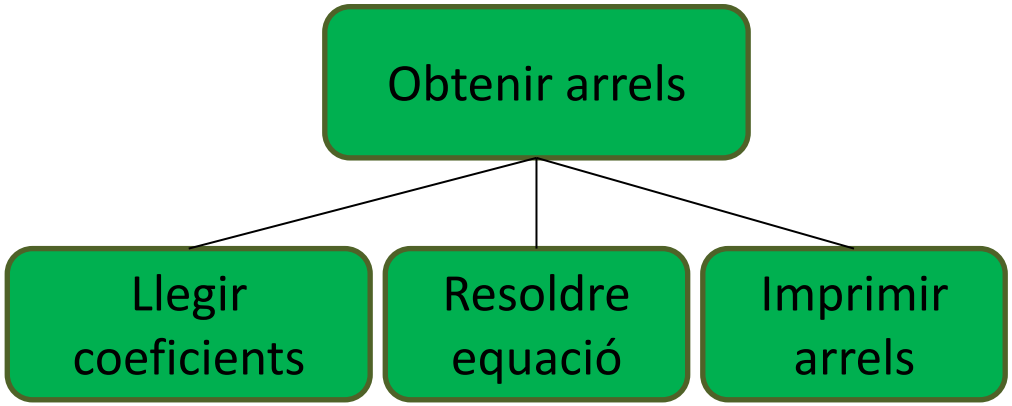
- Disseny descendent:
  - Construeix un sistema per **refinament successiu**.
  - Pot veure's com el desenvolupament en forma d'un arbre
- Procés:
  - Comença expressant un enunciat de la funció al nivell més alt d'**abstracció**.
  - Continua amb una seqüència de passos de refinament, reduint el nivell d'abstracció
  - Descompon cada operació en una combinació d'una o més operacions més senzilles



Exemple

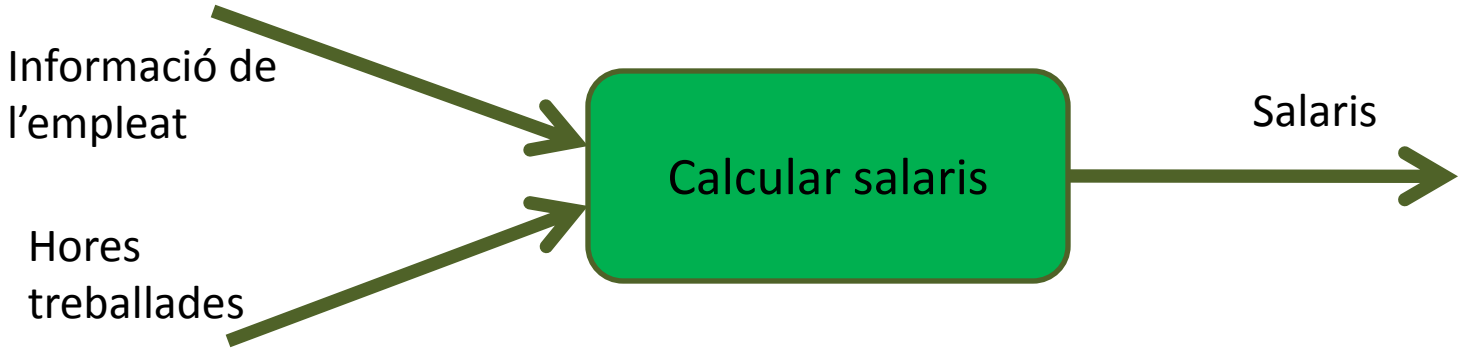
- Obtenir les arrels d'una equació de segon grau.

Obtenir arrels -->
    Llegir coeficients
    Resoldre equació -->
        Calcular discriminant
        Calcular arrels -->
        SI *el discriminant és negatiu* LLAVORS
        Calcular arrels complexes
        SI-NO
        Calcular arrels reals
        FIN-SI
    Imprimir arrels



Exemple

- Sistema de nòmines



Si canviem l'objectiu i en lloc de només calcular el salari volem:
- Extreure algunes estadístiques addicionals.
- Alguns empleats es pagaran mensualment però altres trimestralment!
- El personal vol poder accedir a l'aplicació interactivament !

Pros i Contras de la descomposició funcional

- Avantatges:
  - Disciplina de pensament lògic i ben organitzada
  - Es pot ensenyar amb eficàcia
  - Promou el desenvolupament ordenat de sistemes
  - Ajuda al dissenyador a trobar un camí a través de la complexitat que sovint presenten els sistemes en les seves etapes inicials de disseny.
  - Pot ser útil per a petits programes i algorismes individuals
- Limitacions quan tractem un problema gran:
  - Assumir que el sistema té només una funció principal (el "cim")
  - Base de la descomposició modular: propietats subjectes a canvi.

Funcions i evolució

Perills del disseny descendent

- Suposa que tot el sistema es caracteritza per una función principal
- Es centra en la interfície externa per resoldre la següent pregunta:
  - Què fa el sistema per a l'usuari final?

➢ Exemple:

- Programa amb dos versions:
  - Per "lots"
  - Interactiva

- Èmfasis prematur en restriccions d'ordre.

Descomposició basada en objectes

- Podem trobar una caracterització més estable d'un sistema?
  - Exemple: sistema de nòmines

    → Tipus d'objectes manipulats pel sistema.

    L'esquema orientat a objectes definiria:
    - Tasques de l'empresa
    - Persona
    - Contracte
    - Aplicació

Construcció del software orientat a objectes

- La construcció del software orientat en objectes és el mètode de desenvolupament de software que basa l'arquitectura de qualsevol sistema software en mòduls deduïts dels tipus abstractes d'objectes que manipula.

No pregunti primer què fa el sistema:
pregunti a què li ho fa!

Procés ascendent

Qüestions

- Cerca dels tipus d'objectes
- Descripció de tipus d'objectes
- Descripció de les relacions i estructura del software

Unitat estables de reutilització

Rutina (unitat de descomposició funcional)

vs.

Tipus abstracte d'objectes

EXEMPLE:

Descomposició funcional descendent i Descomposició orientada a objectes

Exemple

- Sistema de reserves per a una companyia Aérea
  - Estats: passos de processament
    - Identificació de l'usuari,
    - Consulta sobre vols,
    - Consulta sobre places,
    - Reserva.

Exemple

- Sistema interactiu multi-panel
  - Patró general:
    - *Estat* (panel amb certes consultes)
    - *Transició* (selecció del següent pas a realitzar)

Exemple: Panel

– Consulta de vols –

Vol des de: Barcelona Destí: París

Sortida prevista: 22 Maig Arribada: 22 Maig

Companyia aérea:

Requisits especials:

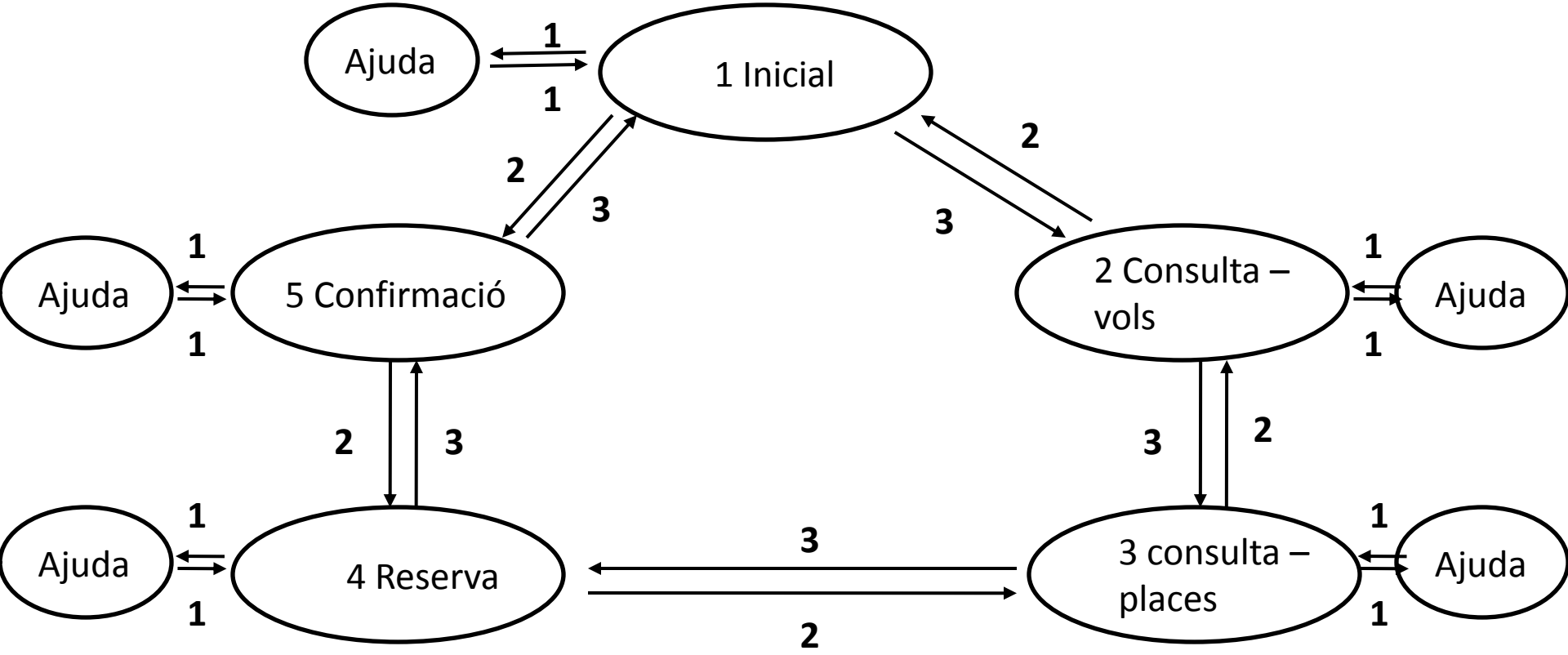
VOLS DISPONIBLES: 1

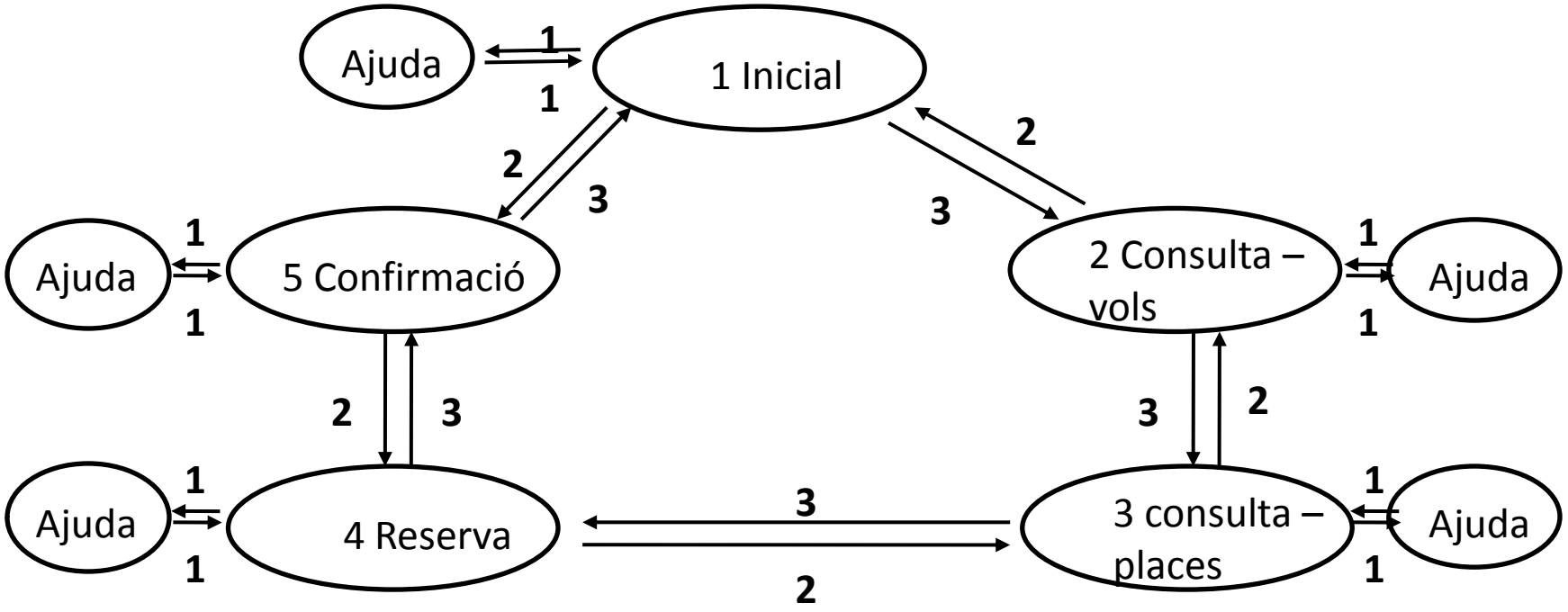
Vuelo: AA 42 Sortida 8:25 Arribada 10:05 Escala: -

Escollir una opció:

0 – **Sortida**
1 – **Ajuda**
2 – **Següent petició**
3 – **Reserva plaça**

Exemple: Un diagrama de transició





Taula de transició:

Estat / Opció	0	1	2	3
1 (Inicial)	-1	0	5	2
2 (Vols)		0	1	3
3 (Places)		0	2	4
4 (Reserves)		0	3	5
5 (Confirm.)		0	4	1
0 (Ajuda)		Tornar		
-1 (Final)				

Exemple: Esquemes del programa

- Primer intent simple
- Solució funcional descendent
- Solució orientada a objectes

→ Per pensar...

Referències

- Fonts utilitzades per a la presentació
  - Modularitat i TADs:

    Capítols 5 i 6 del llibre de Bertrand Meyer, "**Construcción de software orientado a objetos", Prentice Hall, 1998.**
  - Estructura de dades: http://sanchezcrespo.org/Docencia/ED/Estructura_de_Datos_01.ppt

Lectures recomanades

- Capítol 1 del llibre de Bertrand Meyer, "**Construcción de software orientado a objetos", Prentice Hall, 1998.**

- **"No Silver Bullet Essence and Accidents of Software Engineering"** Computer Magazine by Frederick P. Brooks. (April 1987)