

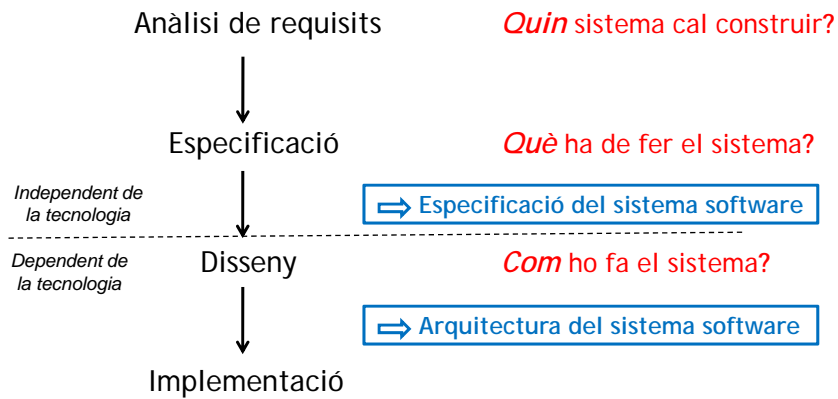
Introducció al disseny de software



Introducció al disseny de software

Etapes del desenvolupament de software
Disseny de software
Disseny amb patrons
Disseny orientat a objectes en UML
Arquitectura en capes d'un sistema d'informació
Patró *Domain Model*
Bibliografia

Etapes del desenvolupament de software



3

Disseny de software

- **Disseny de software** consisteix a definir un sistema software amb el detall suficient per permetre la seva construcció física (implementació)
- **Punt de partida:**
 - Resultat de l'especificació: (Què ha de fer el sistema?)
 - . Especificació de la interacció amb l'usuari (Casos d'ús)
 - . Especificació de dades (Esquema conceptual de les dades)
 - . Especificació de processos (Esquema del comportament)
 - . Requisits no funcionals del sistema
 - Tecnologia (Amb quins recursos?)
 - . Recursos hardware i software disponibles
- **Resultat del disseny:** (Com ho fa el sistema?)
 - **Arquitectura del sistema software:** descripció dels subsistemes i components d'un sistema software, i de les relacions entre aquests components.
- **Procés del disseny:**
 - Metodologies de disseny
 - Adaptació de solucions genèriques a problemes coneguts de disseny (patrons)

4

Determinació de l'arquitectura del software

Com es fa en general?

Dependència tecnològica:

- Propietats que es volen assolir amb l'arquitectura (requisits no funcionals)
- Recursos tecnològics disponibles
 - família de llenguatges de programació
 - família de sistema gestor de bases de dades
 - etc.



L'arquitectura del sistema software i els patrons (arquitectònics) que s'usaran per fer el disseny del sistema

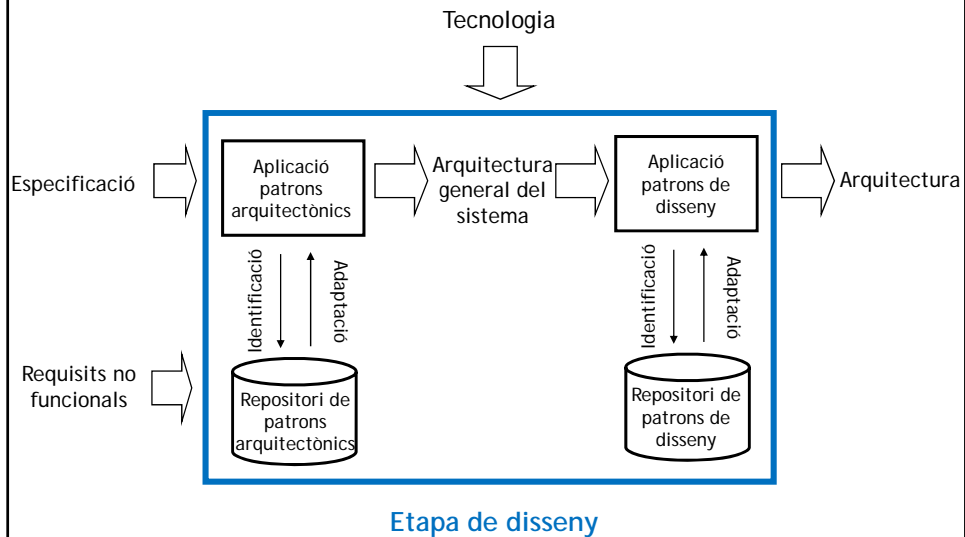
5

Propietats de l'arquitectura software

- **Canviable:**
 - **Extensible:** Noves funcionalitats o millores dels components
 - **Portable:** Canvis plataforma hardware, sistemes operatius, llenguatges
 - **Mantenible:** Detecció i reparació d'errors
 - **Reestructurable:** Reorganització de components
- **Interoperable**
 - Capacitat de dues o més entitats software d'intercanviar funcionalitat i dades
- **Eficient**
 - Temps de resposta, rendiment, ús de recursos
- **Fiable:**
 - **Tolerància a fallades:** Pèrdua de connexió i recuperació posterior
 - **Robustesa:** Protecció contra l'ús incorrecte i el tractament d'errors inesperats
- **Provable**
 - Facilitar les proves del sistema
- **Reusable:**
 - assolir el que es vol amb l'ajuda del que es té

6

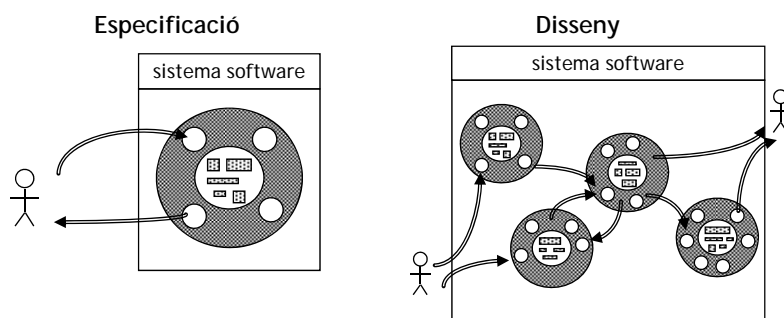
Disseny amb patrons



7

Disseny orientat a objectes

Visió d'un sistema software



Especificació: el sistema software es veu com una sola classe d'objectes que engloba tota la informació i totes les operacions.

Disseny: cada classe té les seves operacions de manipulació d'informació. Els objectes interactuen per satisfer les operacions del sistema.

8

Disseny orientat a objectes en UML

Punt de partida: especificació en UML

- **Especificació: QUÈ** fa el sistema software?
- **Resultat de l'especificació en UML:**
 - Casos d'ús:
Quina interacció hi ha entre els actors i el sistema software?
 - Esquema conceptual de les dades:
Quins són els conceptes rellevants del món real de referència ?
 - Esquema del comportament:
 - Diagrames de seqüència del sistema:* Quina resposta dona el sistema als esdeveniments externs? (quines operacions ha de tenir el sistema?)
 - Contractes de les operacions:* què fan les operacions del sistema?
 - Diagrama d'estats:* per quins estats evolucionen els objectes del sistema?

9

Disseny orientat a objectes en UML

Resultat a assolir: disseny en UML

- **Disseny: COM** estructurarem el sistema perquè faci el que ha de fer?
⇒ *el disseny és una activitat iterativa i és difícil seqüencialitzar tot el que s'hi fa.*
- **Resultat del disseny en UML:**
 - Casos d'ús:
Defineix la interacció real, amb una interfície concreta.
 - Esquema de classes del software (model estàtic):
Describeix les classes del software i les seves interfícies (operacions).
 - Esquema del comportament del software (model dinàmic):
 - Diagrames de seqüència:* defineixen la interacció entre les classes d'objectes per respondre a un esdeveniment extern
 - Contractes de les operacions:* defineixen què fan les operacions de les classes d'objectes

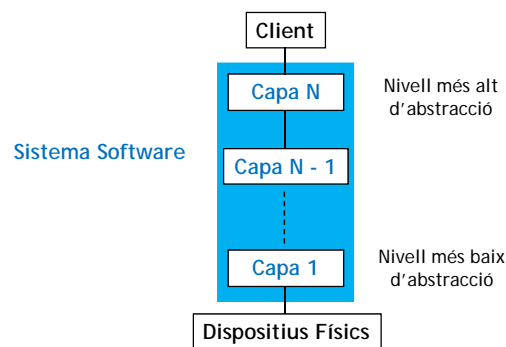
10

Patró arquitectònic: arquitectura en capes

Context:

Un sistema gran que requereix ser descomposat en grups de tasques (components), tals que cada grup de tasques està a un nivell determinat d'abstracció.

És el que farem servir a l'assignatura IES



11

Arquitectura en capes: beneficis i inconvenients

Beneficis:

Canviable, Reusable, Portable, Provable

Inconvenients:

Eficiència

Feina innecessària o redundant

Dificultat en establir la granularitat i el nombre de capes



Arquitectura en capes relaxat

Una capa pot usar els serveis de qualsevol capa inferior

Tots o només part dels serveis de la capa (opacitat parcial)

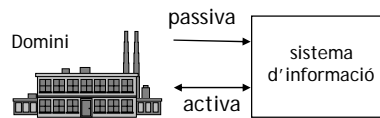
Conseqüències:

Possible guany en flexibilitat i eficiència

Possible pèrdua en la canvialilitat, reusabilitat

12

Arquitectura en capes: aplicació als sistemes d'informació



Funcionalitat passiva del sistema d'informació:

Mantenir una representació consistent de l'estat del domini:

- Capturar els esdeveniments que ocorren al domini
- Actualitzar l'estat del sistema d'informació com a conseqüència d'aquests esdeveniments
- Assegurar la consistència de la representació

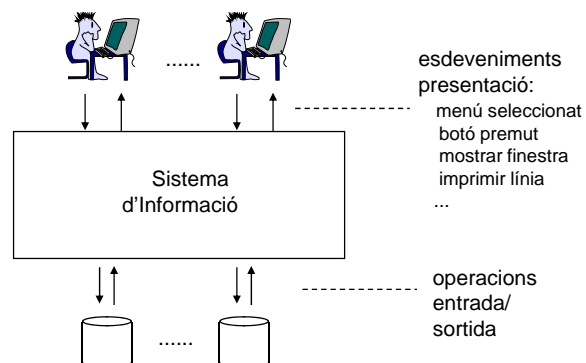
Funcionalitat activa del sistema d'informació:

Respondre a consultes sobre l'estat del domini.

Produir reaccions quan es donen certes condicions predefinides.

13

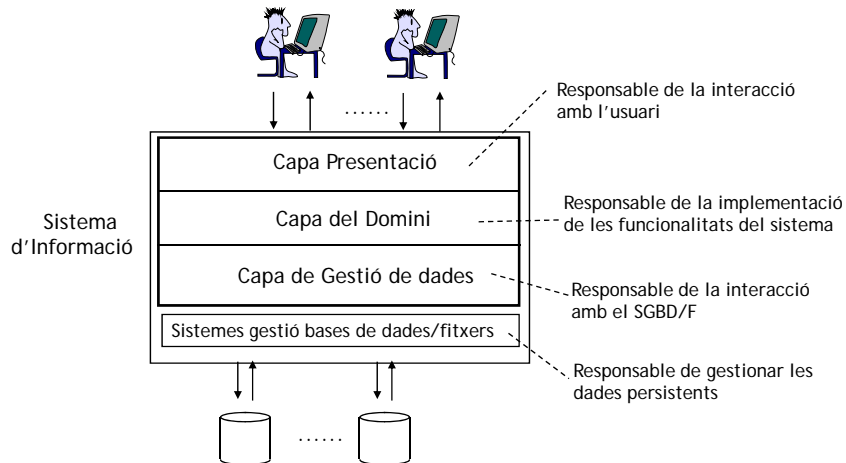
Arquitectura en capes d'un SI: context



14

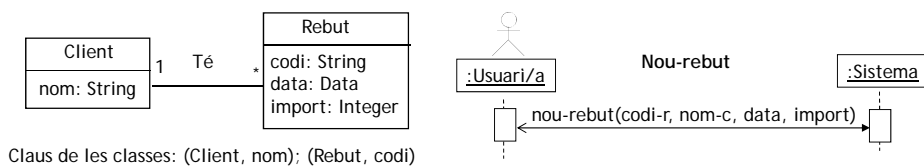
Arquitectura en capes d'un SI: visió general

Aplicació del patró "Arquitectura en 3 Capes" a un Sistema d'Informació



15

Exemple: especificació de partida



Operació: nouRebut (codiReb, nomC, data, import)

Precondicions:

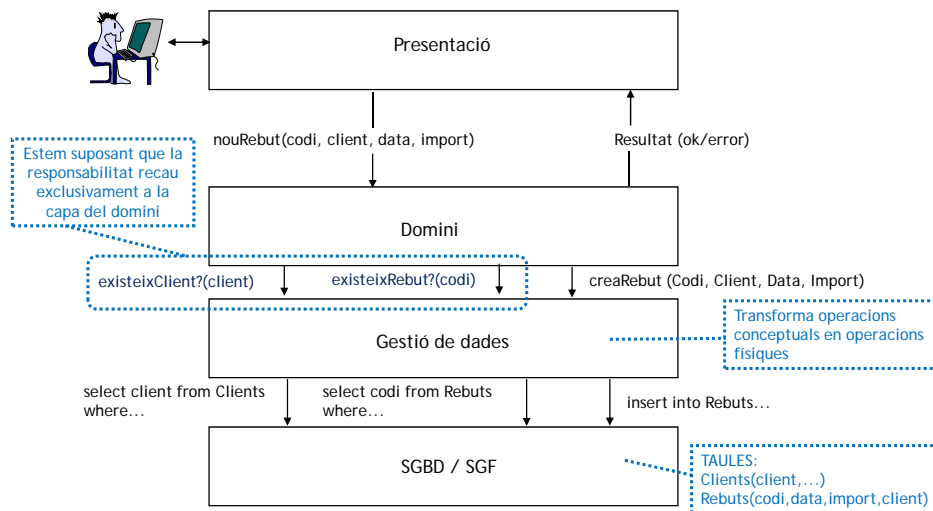
- Existeix el client identificat per nom

Postcondicions:

- És dona d'alta una instància de rebut R amb codi=codiReb, data=data i import=import
- És dona d'alta una instància de l'associació 'Té' que associa el rebut R i el client amb nom=nomC.

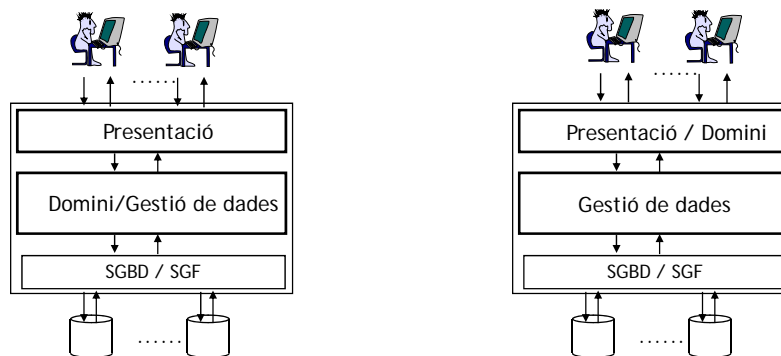
16

Exemple: comunicació entre capes



17

Arquitectura en capes d'un SI: dues capes



Propietats de l'arquitectura:

- Interfície:
 - portable, reusable i canviable (canvis a la interfície són locals)
- Implementació de funcionalitats:
 - poc portable, canviable i reusable (dependència del SGBD, localització i format de dades)

Propietats de l'arquitectura:

- Implementació de funcionalitats:
 - poc portable, canviable i reusable (dependència de interfície i del disseny de pantalles)
- Accés a les dades:
 - portable, reusable i canviable (canvis en format, localització i SGBD són locals)

18

Patrons predominants de la capa de domini

- Hi ha dues implementacions potencials de l'esquema conceptual de les dades:
 - A la capa de domini en forma de classes software
 - A la capa de gestió de dades en forma de taules (base de dades relacional)
- Dues opcions principals
 - *Patró Domain Model:*
L'esquema conceptual de les dades s'implementa en classes software
 - *Patró Transaction Script:*
Predomina la implementació relacional de l'esquema conceptual de les dades
- En ambdós casos cal determinar l'estratègia de transició de l'esquema d'especificació al patró escollit.

19

Patró Domain Model

- La lògica de l'aplicació resideix bàsicament a la capa del domini
- La capa de domini implementa les seves operacions mitjançant la col·laboració d'instàncies de les seves classes:
 - Ús intensiu del concepte d'assignació de responsabilitats a nivell de classe
- Requereix:
 - Adaptació de l'esquema conceptual de les dades i dels contractes de les operacions
 - Conversió de la classe Data a atribut
- Característiques:
 - (+) Explota la riquesa pròpia de l'orientació a objectes
 - (+) Té a l'abast una col·lecció rica de patrons de disseny
 - (-) Pot no aprofitar-se completament de les funcionalitats ofertes pels SGBD

20

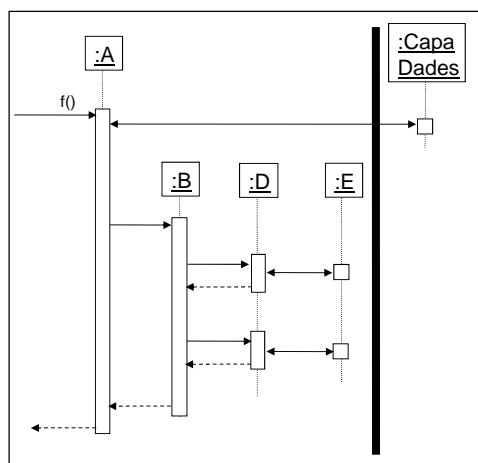
Patró *Transaction Script*

- Procediment que:
 - Rep les dades de la capa de presentació
 - Fa totes les validacions i càlculs necessaris
 - Es comunica amb la capa de dades per consultar i actualitzar la BD
 - Comunica els resultats a la capa de presentació
- Bàsicament, doncs, tenim un procediment per cada transacció de negoci
- La interacció amb la base de dades és totalment explícita
 - El disseny del software es fa considerant el SGBD que s'utilitzarà a la implementació
 - Serà diferent segons usem un SGBD orientat a objectes, relacional, etc.
- Característiques:
 - (+) Paradigma fàcil d'entendre pels programadors
 - (+) Capa de dades molt simple
 - (-) Solució complexa quan la lògica del domini creix
 - (-) La gestió de la persistència és explícita

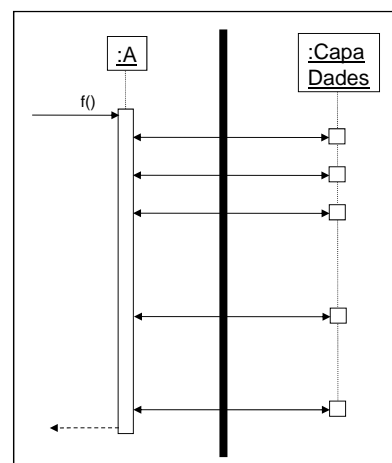
21

Patrons generals de la capa de domini

Vista general dels diagrames de seqüència



DOMAIN MODEL



TRANSACTION SCRIPT

22

Bibliografia

- Larman, C. *"Applying UML and Patterns. An Introduction to Object-oriented Analysis and Design"*, Prentice Hall, 2005, (3ª edició).
- Pressman, R.G. *"Software Engineering. A Practitioner's Approach"*, Mc Graw-Hill, 2005 (7a edició).
- Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. *"Pattern-oriented software architecture. A system of patterns"*, John Wiley & Sons, 1996.
- Fowler, M. *"Patterns of Enterprise Application Architecture"*, Addison-Wesley, 2002.