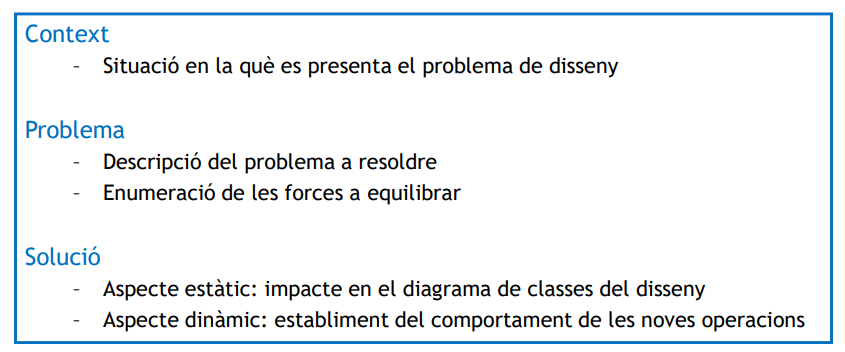
**Tema 10: Patrons de disseny**

**Concepte de Patró**

“Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”

**Patrons**

La Arquitectura en Capes forma part del que anomenarem un “Patró”. Un patró és una solució estandarditzada per a un problema comú que ens serveix de punt de partida per a atacar la situació actual.

****

**Catàlegs de patrons de disseny**

Patrons que determinen l’estructura general de les capes, proposats per Fowler (2003):

* Capa de domini:
  + Gran influència en la distribució de responsabilitats a capes.
  + Domain Model, Transaction Script.
* Capa de dades:
  + Determinen els serveis que ofereix la capa de dades.
  + Data Mapper, Pasarel·la Fila, Enregistrament Actiu.

**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.

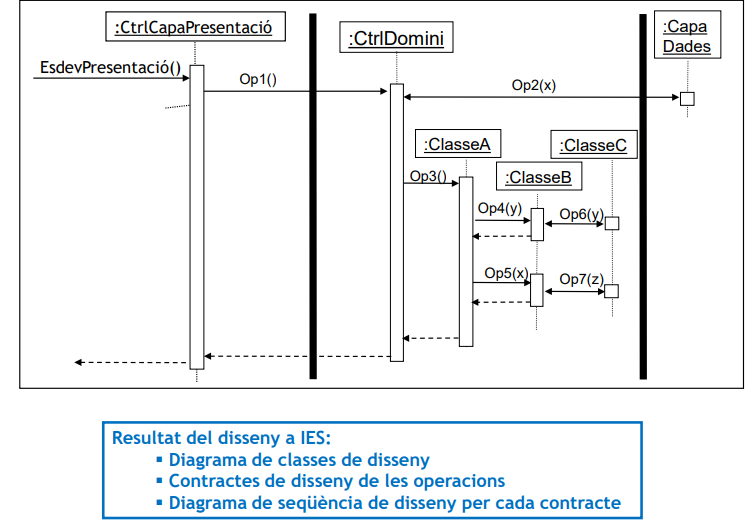
Requereix:

* Una transformació inicial de l’esquema conceptual d’especificació (dades i operacions) a un diagrama de classes i als contractes de les operacions de disseny.
* 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.

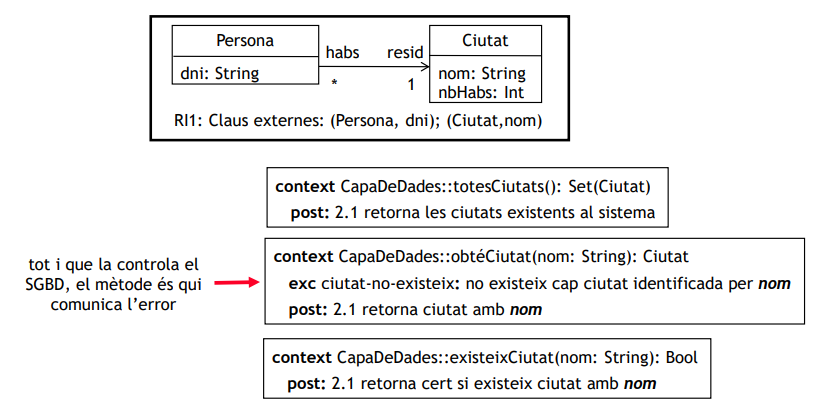
Visió general d’un diagrama de seqüència:



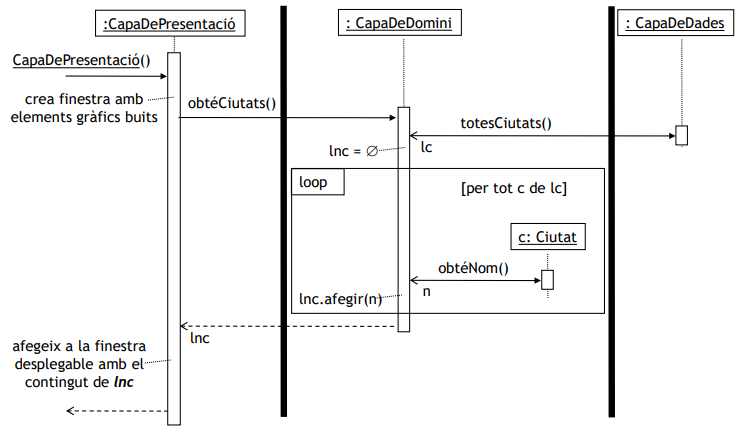
**Operacions de la capa de dades**

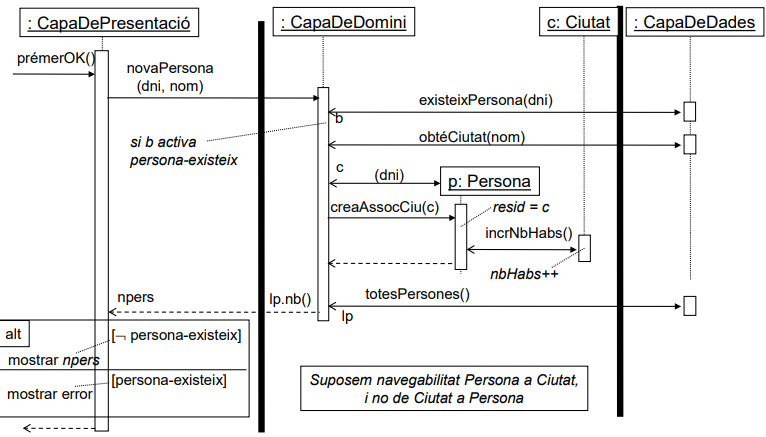
En Domain Model, les úniques operacions que contemplem són, per a cada classe, obtenir un objecte donada la seva clau, i obtenir totes les instàncies de la classe.

Les actualitzacions a la capa de dades són implícites.



Obtenció d’informació – exemple:





**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.

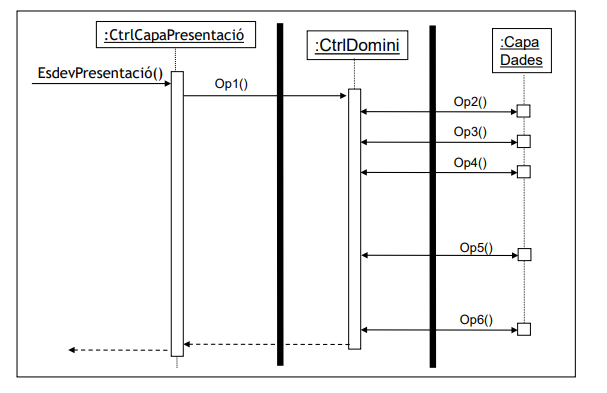
Bàsicament, doncs, tenim un procediment per cada transacció de negoci. La interacció amb la base de dades és totalment explícita:

* 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.

Visió general d’un diagrama de seqüència:



Descripció general:

Context:

* Els (sub)sistemes software reben esdeveniments. Ex: la capa de domini d’un SI rep esdeveniments externs.
* Un cop interceptats aquests esdeveniments, algun objecte del sistema ha de rebre’ls i executar les accions corresponents.

Problema:

* Quin objecte és el responsable de rebre un esdeveniment?

Solució:

* Assignar aquesta responsabilitat a un controlador.
  + Els clients del sistema desconeixen l’estructura interna del sistema.
* Un controlador és un objecte d’una certa classe.
  + El controlador delega sobre un o més objectes del sistema el tractament de l’esdeveniment.
* L’objecte que tracta l’esdeveniment no té coneixement sobre l’existència o el tipus de controlador.
* Variants analitzades:
  + Façana: Un objecte que representa tot el sistema .
  + Transacció (Command): Un objecte que representa una instància d’esdeveniment.

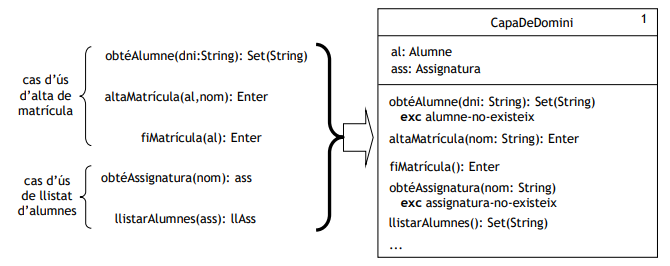
**Controlador façana**

Aspecte estàtic:

Classe singleton:

* tantes operacions com esdeveniments ha de capturar el sistema.
* eventualment, poden incloure’s atributs per compartir informació.

Controladors inflats si hi ha molts esdeveniments → poca cohesió.



**Controlador transacció**

Aspecte estàtic (1):

S’introdueix una classe concreta per cada operació del sistema (transacció).

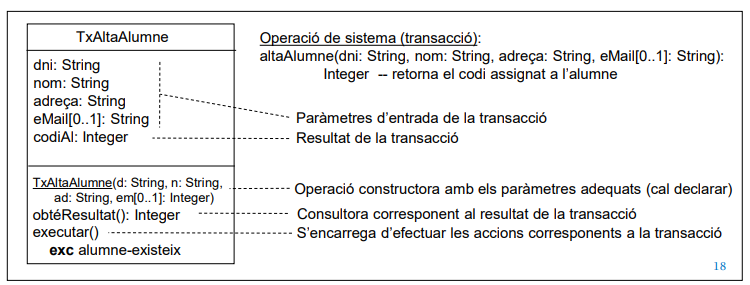
Cada paràmetre de l’operació dona lloc a un atribut de la classe:

* Si l’atribut és out o inout, s’afegeix una operació per consultar el seu valor.
* L’operació constructora de la classe té tants paràmetres com paràmetres in i inout té l’operació.

Si hi ha resultat, també es declara un atribut del tipus del resultat:

* S’afegeix una operació per consultar el seu valor.

S’afegeix una operació executar() que s’encarrega d’executar la transacció.



Aspecte estàtic (2):

S’introdueix una classe abstracta que actua de superclasse de tots els controladors transacció del sistema.

* Declara l’operació d’executar la transacció com a abstracta.
* Proporciona una vista unificada a les classes clients dels diferents tipus de transaccions.

