De l'especificació del sistema a l'especificació de les capes

- Problemàtica
- Transició de l'especificació al disseny. Etapes
- Construcció dels casos d'ús concrets
- Assignació de responsabilitats a capes
- Especificació de les capes
  - Capes de presentació i domini
  - Patrons predominants de la capa de domini
  - Capa de dades amb patró *Domain Model*
  - Capa de dades amb patró *Transaction Script*
- Què queda per fer?
- Bibliografia

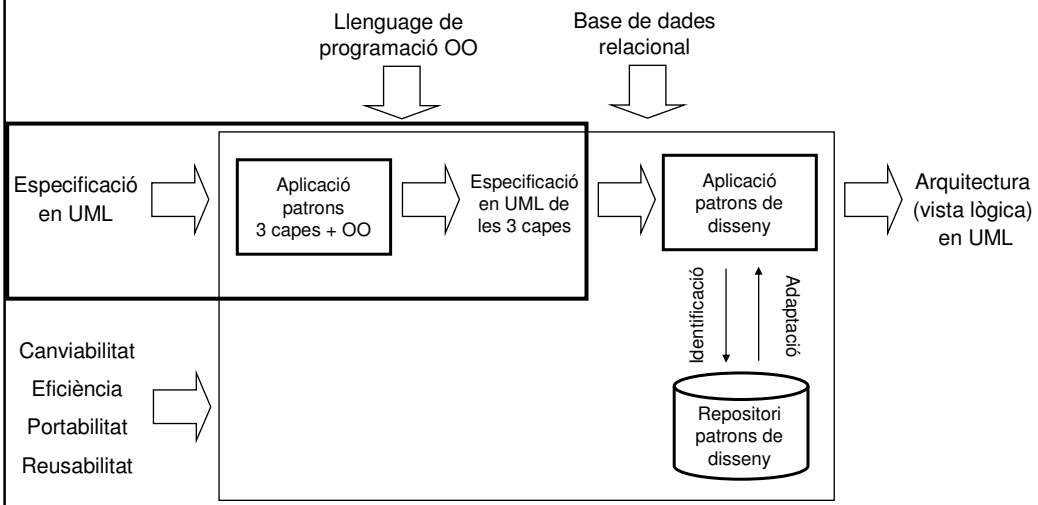
Problemàtica

- Els models de l'especificació:
  - defineixen els conceptes del món real (domini del problema)
  - veuen el sistema software com una caixa negra
    - ✓ existeix un únic objecte que modelitza el sistema software sencer
  - no consideren els requisits no funcionals
    - ✓ en particular, de comunicació amb l'usuari
  - són independents de la tecnologia
    - ✓ exploten al màxim la capacitat expressiva del llenguatge
- Els models del disseny:
  - defineixen els conceptes que es desenvoluparan per proporcionar una solució a les necessitats del món real (domini de la solució)
  - veuen el sistema software com una caixa blanca
    - ✓ distingeixen les tres capes pròpies del patrò arquitectònic elegit
    - ✓ identifiquen els objectes que formen part de cada capa
  - consideren els requisits no funcionals
    - ✓ en particular, de comunicació amb l'usuari
  - són depenents de la tecnologia
    - ✓ han d'adaptar-se a les restriccions pròpies de la tecnologia elegida

Transició de l'especificació al disseny

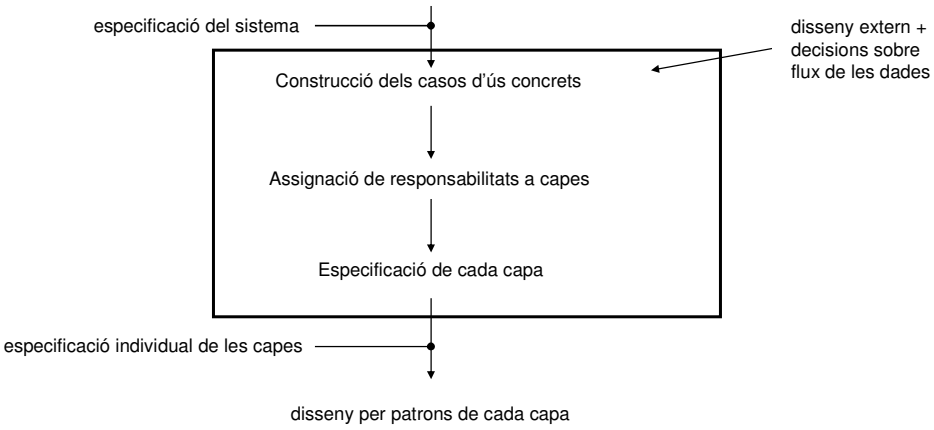
- En conseqüència, necessitem:
  - passar del món del problema al món de la solució
    - ✓ identifiquem i apliquem patrons de disseny
    - ✓ considerem necessitats de disseny (e.g., control dels errors)
  - assignar les responsabilitats dels contractes d'especificació a elements de disseny
    - ✓ assignació de responsabilitats a capes
    - ✓ els contractes de les operacions de les capes han de tractar les responsabilitats
  - produir arquitectures les propietats de les quals concordin amb els requisits no funcionals establerts
    - ✓ hem elegit canviabilitat, eficiència, portabilitat i reusabilitat
    - ✓ el disseny per patrons les soporta
    - ✓ es consideren els requisits de comunicació amb l'usuari
  - apropar els models d'especificació a la tecnologia
    - ✓ normalització en el marc de l'orientació a objectes
    - ✓ disseny d'esquemes de bases de dades relacionals

Etapa de disseny en el context d'ES2



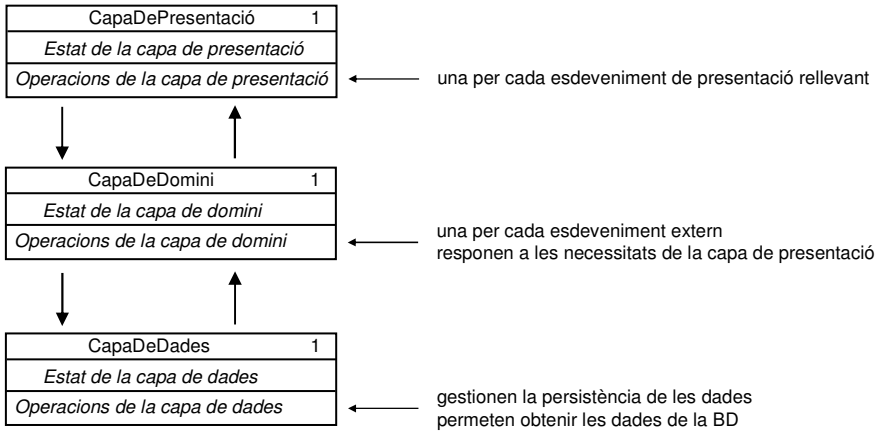
Etapa de disseny

Etapes inicials de la fase de disseny
Especificació en UML de les tres capes



Visió de les capes com a objectes

De moment, modelem cada capa amb una classe que té una única instància (controlador)



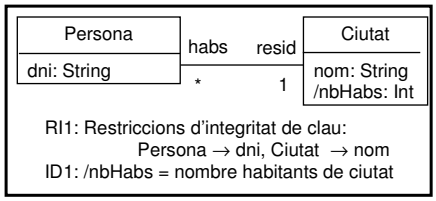
La comunicació entre la capa de domini i la capa de dades està fortament condicionada pel *patró predominant* de la capa de domini

Punt de partida: especificació

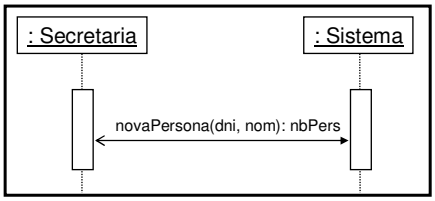
cas d'ús Nova Persona
actors Persona, Secretaria
curs principal
1. La *Persona* comunica el seu dni i ciutat de residència a la *Secretaria*
2. La *Secretaria* introdueix el dni i la ciutat en el *Sistema*
3. El *Sistema* enregistra les dades
4. El *Sistema* mostra el nombre de persones existents

extensions
dni-existeix: Ja existeix alguna persona amb el dni donat (2)
1. El *Sistema* avisa a la *Secretaria* que ja existeix la persona
2. Acaba el cas d'ús

ciutat-no-existeix: No existeix cap ciutat amb el nom donat (2)
1. El *Sistema* avisa a la *Secretaria* que no existeix la ciutat
2. Acaba el cas d'ús



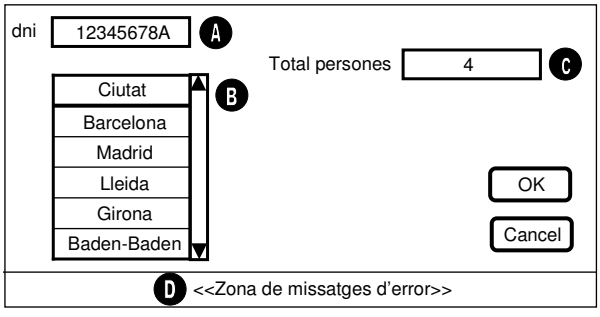
RI1: Restriccions d'integritat de clau:
Persona → dni, Ciutat → nom
ID1: /nbHabs = nombre habitants de ciutat



context novaPersona(dni: String, nom: String): Enter
-- dóna d'alta la persona amb el *dni* i la ciutat *nom*
pre: 1.1 existeix la ciutat *nom*
post: 2.1 dóna d'alta la instància de persona amb *dni*
post: 2.2 associa la persona amb la ciutat *nom*
post: 2.3 result = nombre de persones del sistema

Construcció dels casos d'ús concrets
Disseny extern de la interfície

- S'identifiquen els elements més importants de la comunicació actors – sistema
  - camps, desplegables, zones de missatge, etc.
  - els bategem per referir-nos-hi des dels casos d'ús concrets
- No ens preocupem de temes de baix nivell de detall
  - colors, formes, etc.
  - zooms, scrolling, minimitzar, etc.



Construcció dels casos d'ús concrets
Redacció dels casos d'ús

- Cal recollir tots els esdeveniments de presentació
- Cal analitzar l'impacte del disseny extern en els casos d'ús
  - explotar al màxim els avantatges dels elements de disseny
  - pot portar-nos a refer els diagrames de seqüència de les operacions dels sistema
- Cal determinar en quin moment les dades es fan persistents
  - implicacions en assignar responsabilitats a capes
- Cal establir clarament la relació entre el cas d'ús i els elements del disseny extern
  - en quin moment s'omple quin camp, etc.

Construcció dels casos d'ús concrets
Redacció dels casos d'ús, exemple

cas d'ús Nova Persona
actors Persona, Secretaria
curs principal
1. La *Persona* comunica el seu dni a la *Secretaria*
2. La *Secretaria* introdueix a <<A>> el dni, i selecciona del desplegable <<B>> la ciutat
3. La Secretaria prem <<OK>>
4. El *Sistema* enregistra i fa persistents les dades
5. El Sistema mostra a <<C>> el nombre de persones existents

extensions
dni-existeix: Ja existeix alguna persona amb el dni donat (2)
1. El *Sistema* escriu a <<D>> que ja existeix la persona
2. Acaba el cas d'ús

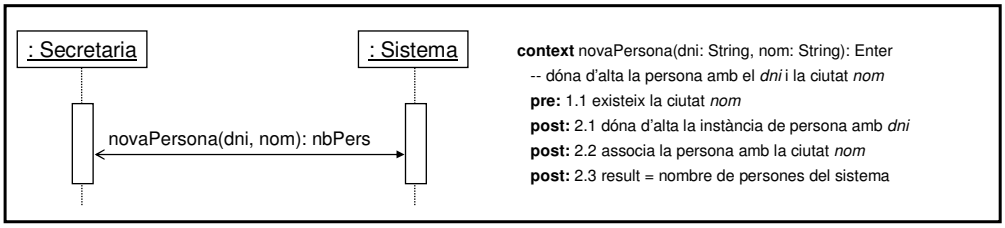
operació-cancel.lada: La Secretaria prem <<Cancel>> (2-3)
1. El *Sistema* escriu a <<D>> un missatge confirmant la cancel.lació
2. Acaba el cas d'ús

no-hi-ha-ciutats: No hi ha cap ciutat en el sistema (2)
1. El *Sistema* escriu a <<D>> un missatge informant d'aquesta situació
2. Acaba el cas d'ús

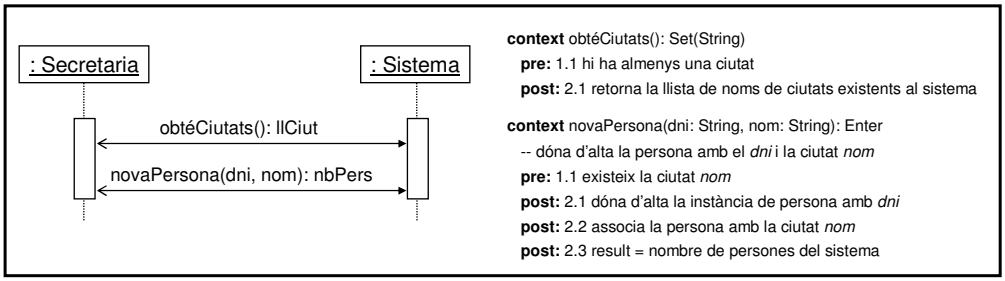
Construcció dels casos d'ús concrets
Canvis al model del comportament

- Cal traslladar els detalls generats durant l'escriptura del cas d'ús concret als models creats a l'especificació
  - operacions que apareixen, altres que són substituïdes, etc.
  - cada esdeveniment de presentació es correspon a una operació en el diagrama de seqüència
  - els paràmetres de les operacions indiquen el flux d'informació actors – sistema i, eventualment, l'estat del cas d'ús
- Com a resultat, en el disseny poden aparéixer noves versions de:
  - els diagrames de seqüència
  - els contractes
- No ens preocupem de les extensions que simplement acaben el cas d'ús:
  - el cas d'ús actua com a transacció

Construcció dels casos d'ús concrets
Canvis al model del comportament



context novaPersona(dni: String, nom: String): Enter
-- dóna d'alta la persona amb el *dni* i la ciutat *nom*
pre: 1.1 existeix la ciutat *nom*
post: 2.1 dóna d'alta la instància de persona amb *dni*
post: 2.2 associa la persona amb la ciutat *nom*
post: 2.3 result = nombre de persones del sistema



context obtéCiutats(): Set(String)
pre: 1.1 hi ha almenys una ciutat
post: 2.1 retorna la llista de noms de ciutats existents al sistema

context novaPersona(dni: String, nom: String): Enter
-- dóna d'alta la persona amb el *dni* i la ciutat *nom*
pre: 1.1 existeix la ciutat *nom*
post: 2.1 dóna d'alta la instància de persona amb *dni*
post: 2.2 associa la persona amb la ciutat *nom*
post: 2.3 result = nombre de persones del sistema

Assignació de responsabilitats a capes

- Responsabilitats: pre i postcondicions dels contractes + informació del model de dades
  - Cal assignar cada responsabilitat a una o més de les 3 capes de l'arquitectura
- Típicament:
  - Capa de presentació: gestió dels elements del disseny extern
  - Capa de domini: càlculs i consultes
  - Capa de gestió de dades: actualitzacions i obtenció d'elements
- La repartició de responsabilitats entre les capes de domini i de gestió de dades depèn del *patró predominant* a la capa de domini
  - *Domain Model* vs. *Transaction Script*
  - De moment, considerem les dues capes conjuntament
- Algunes responsabilitats no es tractaran en el mètode que implementa l'operació
  - Principalment, alguns invariants del sistema (restriccions d'integritat gràfiques i textuals)
  - Exemple típic: tractament de claus en el SGBD
  - Això sí, els mètodes han d'incloure les pre/post corresponents i, en concret, han de detectar els possibles errors
- Es pot decidir materialitzar informació derivada

Assignació de responsabilitats a capes
Exemple Alta Persona

Responsabilitat		**Assignació**	
Operació	**#**	**Capa**	**Justificació**
obtéCiutats	1.1	presentació	Comprovació en base a la informació obtinguda a 2.1
	2.1	domini+dades	Obtenció del noms de totes les ciutats del sistema
		presentació	Creació i presentació del desplegable de ciutats
novaPersona	1.1	presentació	Selecció té lloc en el desplegable de ciutats
	2.1	domini+dades	Alta d'una nova persona
	2.2	presentació	Selecció de ciutat del desplegable
	2.2	domini+dades	Associació de la persona amb la ciutat
	2.3	domini+dades	Comptatge de totes les persones del sistema
	2.3	presentació	Presentació del resultat

Responsabilitat		**Assignació**	
Element	**#**	**Capa**	**Justificació**
Multiplicitat	Persona -> Ciutat [1]	domini+dades	Assegurada per 2.2 de novaPersona
Inf. derivada	ID1	domini+dades	Atribut nbHabs materialitzat i actualitzat en novaPersona
Clau	RI1 (Persona)	dades (SGBD)	Primary Key al SGBD -> **exc** persona-existeix
Clau	RI1 (Ciutat)	N/A	No és afectada per AltaPersona

Especificació de cada capa

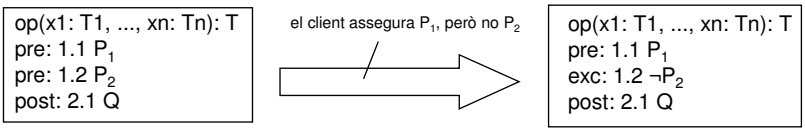
- Cal identificar les operacions de cada capa i escriure els seus contractes
  - Considerem cada capa com una caixa negra: classe (controlador) amb una única instància
  - Les operacions de la classe ho són de la capa
- Aprofitem els controladors per guardar l'estat del cas d'ús com a atributs
  - Ajuda a reduir l'acoblament entre capes
- L'assignació de responsabilitats entre les capas de domini i de gestió de dades depèn del patró arquitectònic predominant a la capa de domini
  - *Domain Model* vs. *Transaction Script*
  - Impacte en:
    - ✓ diagrama de classes de disseny
    - ✓ diagrames de seqüència de les operacions de la capa de domini
    - ✓ contractes de la capa de gestió de dades

Especificació de cada capa
Capes individuals

- Capa de presentació:
  - ✓ s'identifiquen els esdeveniments de presentació
  - ✓ les operacions venen determinades pel tipus d'element de disseny extern implicat
  - ✓ algunes postcondicions del contracte impliquen crides a la capa de domini
  - ✓ l'operació de creació del controlador pot exigir alguna operació addicional no prevista a la capa de domini
- Capa de domini:
  - ✓ es conserva tota operació que té responsabilitats a la capa de domini o dades
  - ✓ s'adapta la signatura a aquestes responsabilitats i es modifica el contracte convenientment
  - ✓ les precondicions (de les operacions del sistema) que queden sobre la responsabilitat de la capa de domini esdevenen excepcions (de les operacions de la capa de domini)
  - ✓ les precondicions que ja controla la capa de presentació poden quedar com a precondicions o esdevenir excepcions
- Capa de dades: segons el patró predominant a la capa de domini, oferirà operacions d'una o més de les categories següents
  - ✓ operacions de consulta:
    - ❖ obtenció d'elements donada la seva clau
    - ❖ obtenció de totes les instàncies d'una classe
    - ❖ altre tipus de consultes
  - ✓ operacions d'actualització

Especificació de cada capa
Tractament dels Errors

- Cal determinar com les operacions de les capes tracten els diferents errors:
  - se sap que un determinat error no es produirà mai quan s'invoca una operació
    - ✓ perque una capa superior el controla
    - ✓ l'operació pot establir l'absència d'error com a precondició → és l'opció per defecte
    - ✓ exemple típic: la capa de presentació detecta un error i no el propaga a la capa de domini → l'operació de la capa de domini no fa controls redundants
    - ✓ no obstant, també pot mantenir l'error com a excepció i comprovar-lo → reusabilitat, robustesa, etc.
  - altrament, l'operació ha de detectar l'error forçosament
    - ✓ es declara una excepció per a l'error
    - ✓ el diagrama de seqüència controla la condició (possiblement amb l'ajut d'altres mètodes)
    - ✓ en cas que es doni la condició d'error, s'activa l'excepció
    - ✓ el client de l'operació ha de comprovar i tractar, o propagar, l'excepció



Especificació de cada capa
Detall dels contractes, capa de presentació

context CapaDePresentació::CapaDePresentació()
post: 2.1 neteja el camp <<A>>
2.2 neteja el camp <<C>> buit
2.3 neteja el camp <<D>>
2.4 si hi ha ciutats al sistema, s'inicialitza el desplegable <<B>> amb els noms de ciutat del sistema
2.5 si no hi ha ciutats al sistema, mostra un missatge d'error en el camp <<D>>

comunicació amb la capa de domini

context CapaDePresentació::prémerOK()
pre: 1.1 hi ha algun valor seleccionat a <<B>>
pre: 1.2 camp <<A>> no buit
post: 2.1 si la persona corresponent al dni entrat al camp <<A>> no existeix:
2.1.1 alta de la persona amb aquest dni residint a la ciutat seleccionada en el desplegable <<B>>
2.1.2 mostra el nombre de persones del sistema en el camp <<C>>
2.2 altrament mostra un missatge d'error en el camp <<D>>

comunicació amb la capa de domini

context CapaDePresentació::prémerCancel()
post: 2.1 tanca la finestra

Especificació de cada capa
Detall dels contractes, capa de domini

context CapaDeDomini::obtéCiutats(): Set(String)
post: 2.1 retorna la llista de noms de ciutats existents al sistema

l'assegura la capa de presentació

tot i que la controla el SGBD, el mètode és qui comunica l'error

context CapaDeDomini::novaPersona(dni: String, nom: String): Enter
pre: 1.1 existeix la ciutat *nom*
exc persona-existeix**:** existeix la persona amb el *dni*
post: 2.1 dóna d'alta la instància de persona amb *dni*
2.2 associa la persona amb la ciutat *nom*
2.3 retorna el nombre de persones del sistema
2.4 incrementa el nombre d'habitants nbHabs de la ciutat *nom*

antiga informació derivada

en aquest punt, es pot propagar o capturar l'excepció (en el nostre exemple, la propaguem)

(dni)

p: Persona

context Persona::persona(dni: String)
exc persona-existeix**:** ja existeix una Persona amb dni
post: 2.1 crea una Persona amb dni

contracte de creadora amb excepció de control de clau

Especificació de cada capa
Detall dels contractes, capa de domini, variant

context CapaDeDomini::novaPersona(dni: String, nom: String): Enter
pre: 1.1 existeix la ciutat *nom*
exc persona-existeix**:** existeix la persona amb el *dni*
post: …

canviabilitat (p.e., la capa de presentació pot canviar)
reusabilitat (p.e., arquitectura orientada a serveis)

context CapaDeDomini::novaPersona(dni: String, nom: String): Enter
exc ciutat-no-existeix**:** no existeix la ciutat *nom*
persona-existeix**:** existeix la persona amb el *dni*
post: …

Patrons predominants de la capa de domini

- Distància conceptual entre el model del domini i la base de dades
  - Model del domini: orientat a objectes
  - Base de dades: tecnologia relacional
- Cal determinar l'estratègia de transició de l'un a l'altre
- Dues opcions principals
  - *Domain Model*: predomina el model del domini
  - *Transaction Script*: predomina la base de dades
- Combinen amb els patrons de la capa de dades (v. tema 7):
  - *Domain Model* + *Data Mapper*
  - *Transaction Script* + (*Pasarel.la Fila, Enregistrament Actiu,* …)

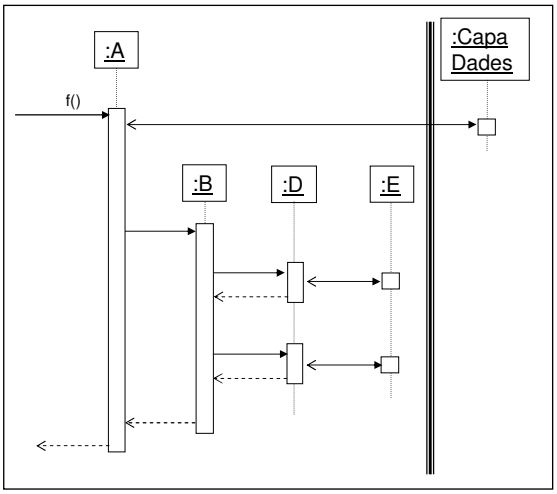
Patró *Domain Model*

- 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:
  - normalització del model conceptual de dades
    - sense impacte en els contractes si s'han fet a un alt nivell d'abstracció
  - conversió de classe Data a atributs
- Característiques:
  - (+) Explota la riquesa pròpia de l'orientació a objectes
  - (+) Té a l'abast una col.lecció rica de patrons de disseny
  - (+) La gestió de la persistència és transparent al dissenyador
  - (–) Pot no aprofitar-se completament de les funcionalitats ofertes pels SGBD

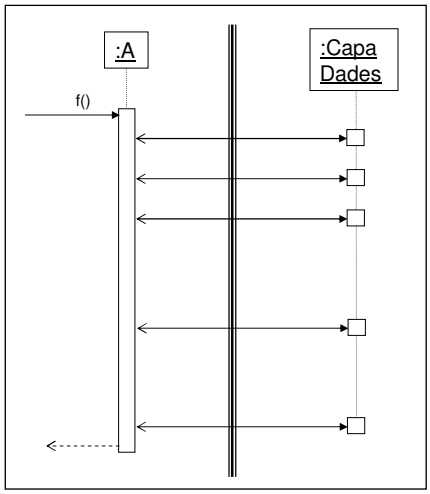
Patró *Transaction Script*

- Procediment que:
  - Reb 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.
    - ✓ en aquesta assignatura suposem SGBD relacional
    - ✓ cal fer el disseny de l'esquema de la base de dades segons aquest supòsit
- Característiques:
  (+) Paradigma fàcil d'entendre pels programadors
  (+) Capa de dades molt simple
  (+) Pot aprofitar-se complementament de les funcionalitats ofertes pel SGBD
  (–) Solució complexa quan la lògica del domini creix
  (–) La gestió de la persistència és explícita

Patrons generals de la capa de domini
Vista general dels diagrames de seqüència



DOMAIN MODEL

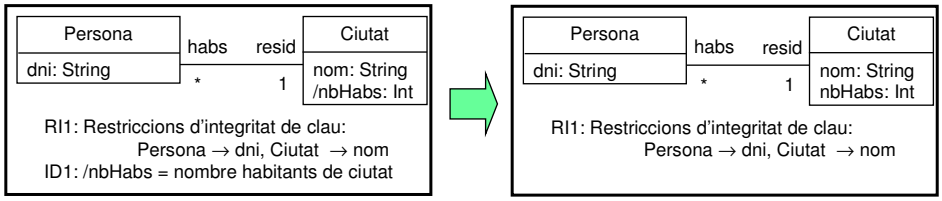


TRANSACTION SCRIPT

Especificació de cada capa
Diagrama de classes, capa de domini, normalització: cas *Domain Model*

En *Domain Model*, cal normalitzar el diagrama de classes usant les regles del procés de normalització (*cf.* ES1)

- Associacions *n*-àries ($n > 2$) a binàries
- Classes associatives a associacions
- Conversió d'informació derivada materialitzada (si així s'ha decidit en assignar responsabilitats)

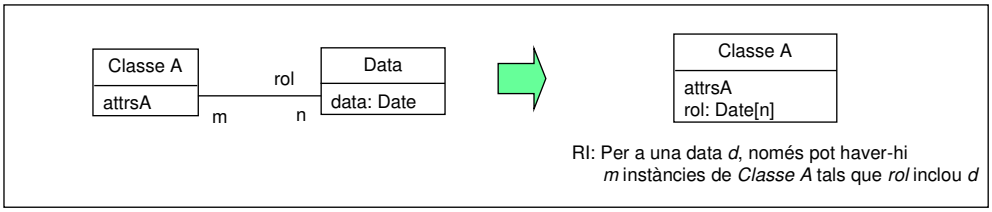


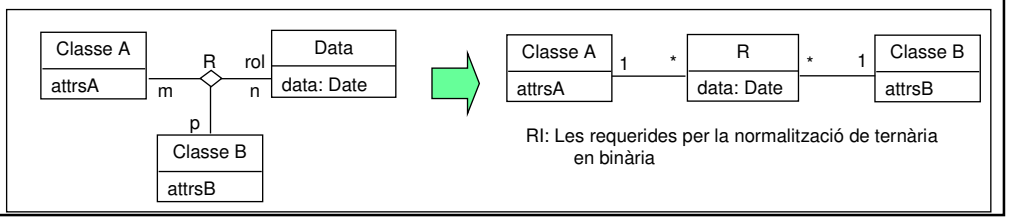
En el futur, aprofitarem aquest pas per introduir altres canvis provocats per l'aplicació de certs patrons (p.e., patró estat, patró controlador, etc.)

Especificació de cada capa
Diagrama de classes, capa de domini, dates: cas *Domain Model*

Tractament de les dates:

- Al model conceptual de l'especificació, sovint trobem una classe *Data*
- Per a la majoria de llenguatges, les dates són un tipus de dada més
- Eliminem la classe i convertim les associacions en atributs:





Especificació de cada capa
Detall dels contractes, capa de dades: cas *Domain Model*

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

context CapaDeDades::totesCiutats(): Set(Ciutat)
post: 2.1 retorna les ciutats existents al sistema

tot i que la controla el SGBD, el mètode és qui comunica l'error →

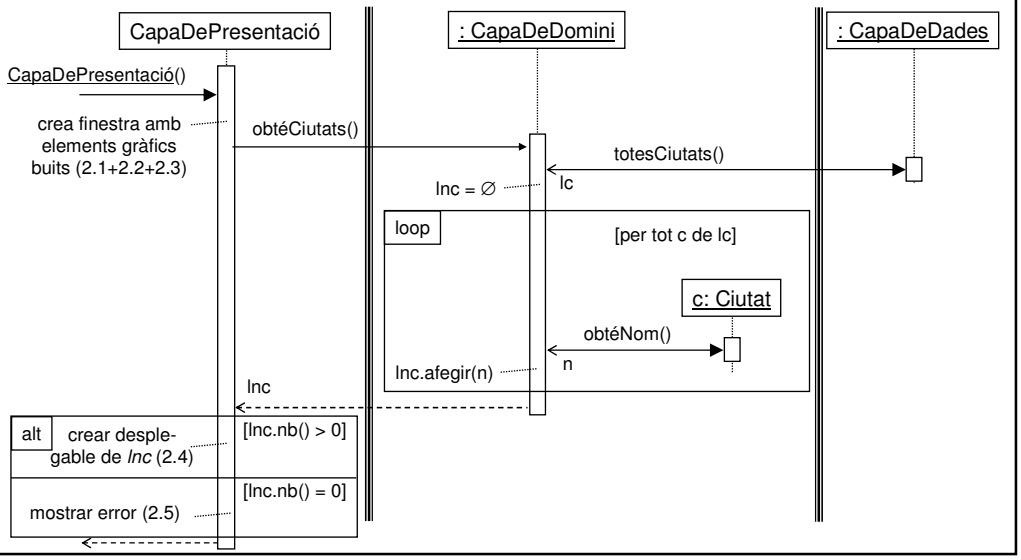
context CapaDeDades::obtéCiutat(nom: String): Ciutat
exc ciutat-no-existeix**:** no existeix cap ciutat identificada per *nom*
post: 2.1 retorna ciutat amb *nom*

context CapaDeDades::existeixCiutat(nom: String): Bool
post: 2.1 retorna cert si existeix ciutat amb *nom*

← evita acoblaments quan l'únic que cal és comprovar l'existència d'un objecte

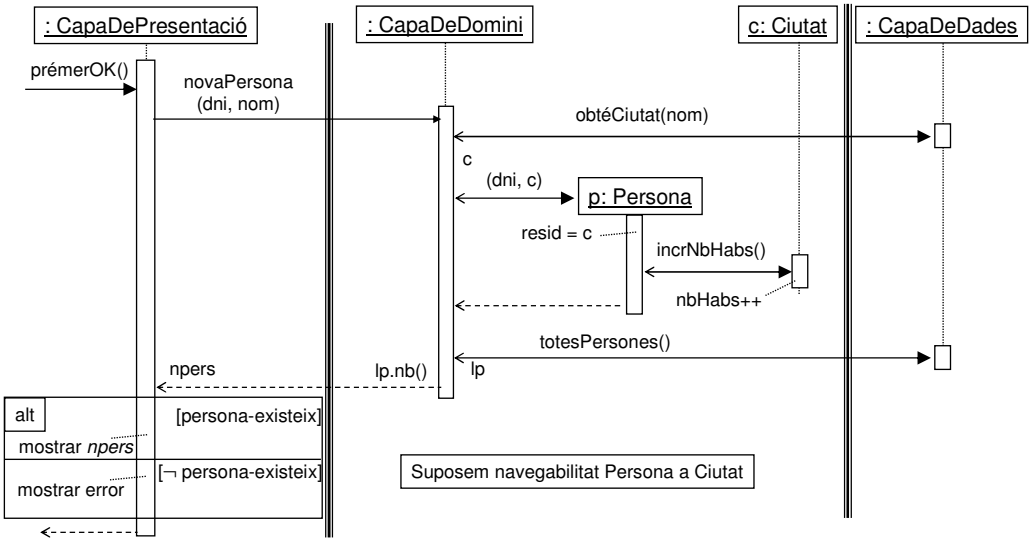
Detall dels diagrames de seqüència (1)

En *Domain Model*, el codi de la capa de domini és ric (explota al màxim la OO)



Detall dels diagrames de seqüència (2)

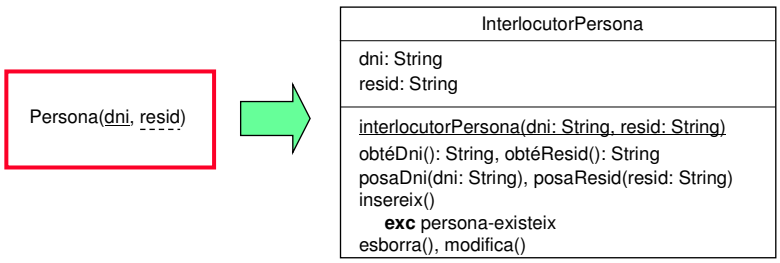
En *Domain Model*, el codi de la capa de domini és ric (explota al màxim la OO)



Especificació de cada capa
Detall dels contractes, capa de dades: cas *Transaction Script*

En *Transaction Script*, a banda de les operacions de *Domain Model*, la capa de dades ofereix una classe interlocutor per cada taula de la BD

- els interlocutors contenen atributs que representen camps de les taules corresponents (modificables mitjançant operacions més o menys senzilles)
- els interlocutors permeten actualitzar la BD (altes, baixes i modificacions)
- les actualitzacions a la capa de dades s'han d'invocar explícitament



Ho estudiarem en detall al tema 7!

Contractes de la capa de dades
Cas *Transaction Script*: operacions d'obtenció d'interlocutors

Hi ha diverses alternatives que estudiem al tema 7

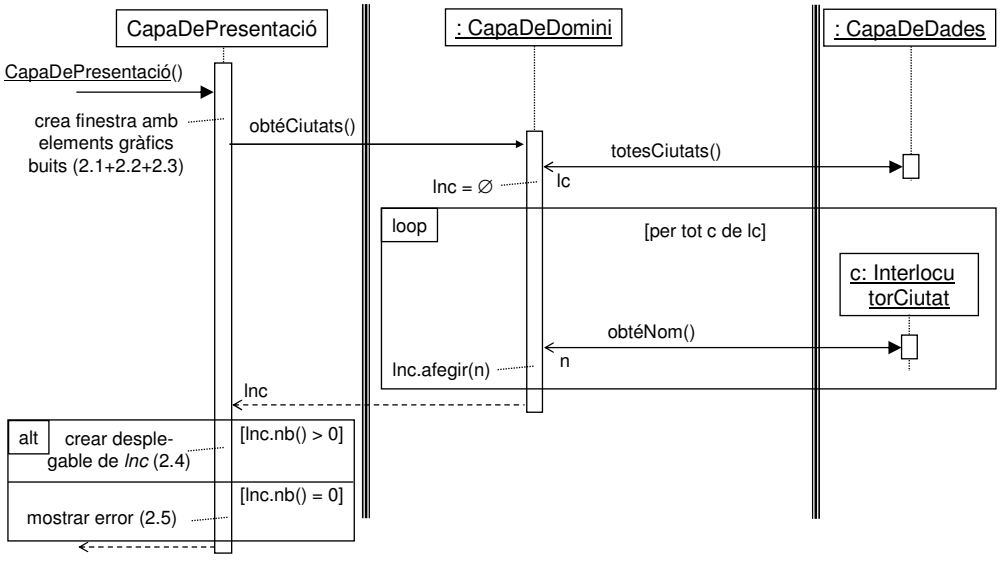
Oferim una versió simplificada (v. tema 7 per versió definitiva)

context CapaDeDades::totesCiutats(): Set(InterlocutorCiutat)
post: 2.1 retorna les ciutats existents al sistema

context CapaDeDades::obtéCiutat(nom: String): InterlocutorCiutat
exc ciutat-no-existeix**:** no existeix cap ciutat identificada per *nom*
post: retorna ciutat amb *nom*

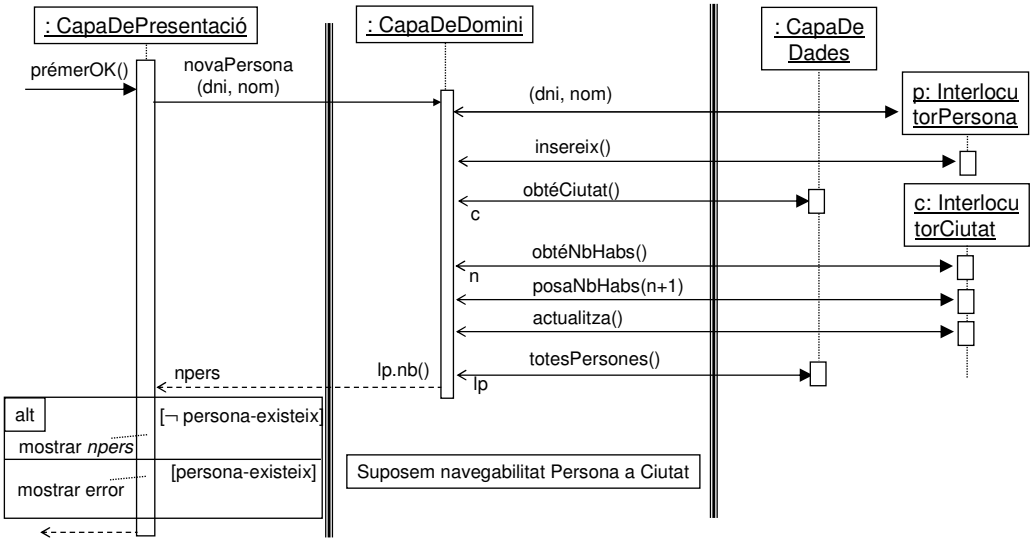
context CapaDeDades::existeixCiutat(nom: String): Bool
post: 2.1 retorna cert si existeix ciutat amb *nom*

Detall dels diagrames de seqüència (1)



Detall dels diagrames de seqüència (2)

En *Transaction Script*, el codi de la capa de domini és procedimental (crides a la BD)



Què queda per fer?

- Una vegada es disposa de l'especificació de cada capa, cal dissenyar cada capa independentment:
  - si podem, d'adalt a baix, per si detectem errors o mancances
  - seguim els principis del disseny per contracte
    - ✓ el disseny de cada operació ha de satisfer la seva especificació
  - apliquem patrons de disseny a cada capa:
    - ✓ a la capa de presentació i domini són similars
    - ✓ a la capa de dades hi ha patrons més lligats al tractament de la persistència
  - cada capa té les seves particularitats
    - ✓ presentació: elements del disseny extern → quins i on?
    - ✓ domini: més patrons a aplicar
    - ✓ dades: pas del diagrama de classes a un esquema de base de dades relacional
  - el patró predominant de la capa de domini condiciona la resta del procés

Resolució de les capes de domini i de gestió de dades

- Cas *Domain Model*:
  - aplicar patrons de disseny a la capa de domini (tema 5)
  - especificar la transformació del diagrama de classes en esquema relacional de BD
    - ✓ patró *Data Mapper* (tema 7)
- Cas *Transaction Script*:
  - fer el disseny lògic de la BD (regles similar a les que s'apliquen en *Data Mapper*)
    - ✓ en aquest moment podem completar l'especificació de la capa de gestió de dades com hem vist en aquest tema
  - decidir el tipus d'interlocutor (al tema 7 introduïm els més coneguts com a patrons)
  - dissenyar els diagrames de seqüència de la capa de domini (estil procedimental)
- En tots dos casos
  - el disseny lògic de la BD serà normalment independent del patró predominant
    - ✓ taules
    - ✓ responsabilitats assignades a la BD (ja sabem claus; a més, Delete on Cascade, etc.)
  - podem decidir de permetre consultes complexes directament sobre la BD
    - ✓ en *Domain Model*, permet explotar les facilitats del model relacional sense perdre el context orientat a objectes (però cal anar amb cura per no abusar-ne)
    - ✓ en *Transaction Script*, permet estalviar interlocutors quan siguin innecessaris

Bibliografia

- *Applying UML and Patterns*
  C. Larman
  Prentice Hall, 2005 (Tercera edició), caps. 13, 17 i 18
- http://www.uml.org/#UML2.0