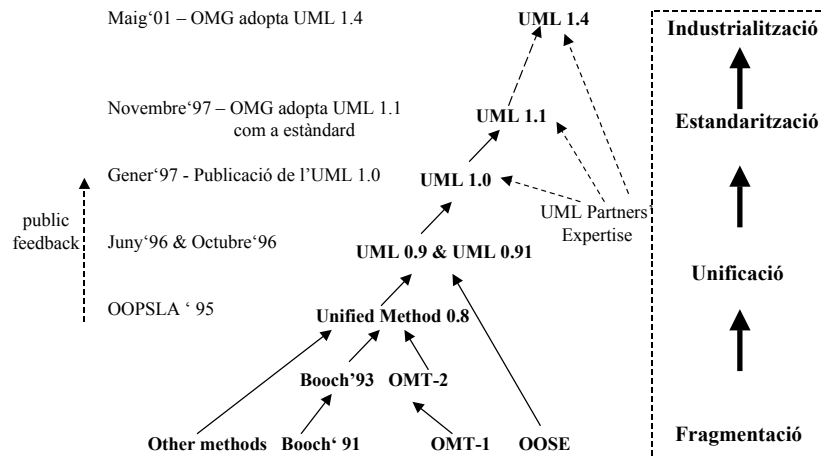
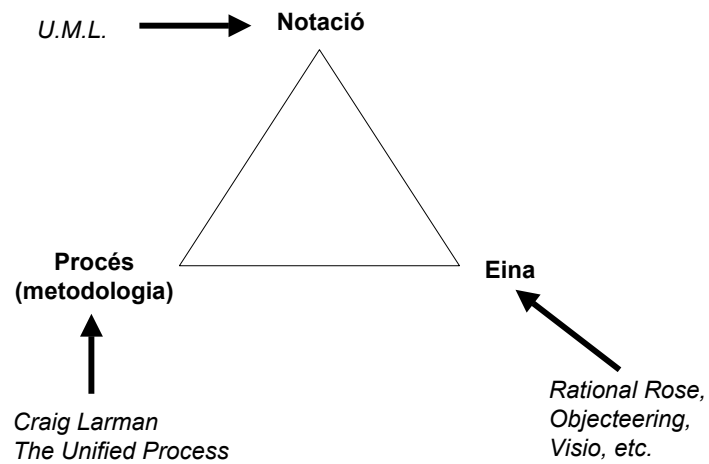


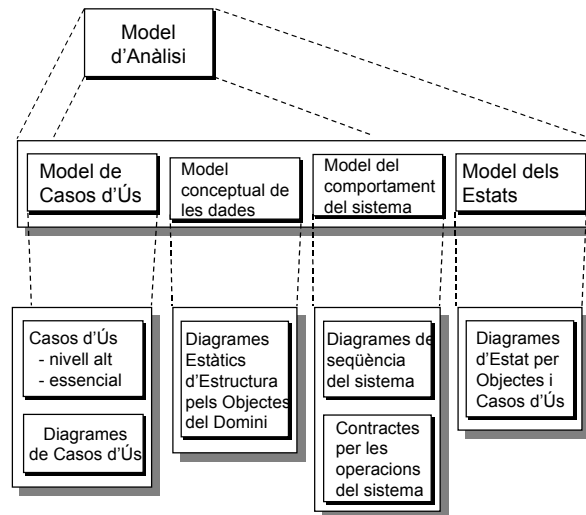
El llenguatge UML (Unified Modeling Language)



El triangle de l'èxit



Model d'Anàlisi (especificació)



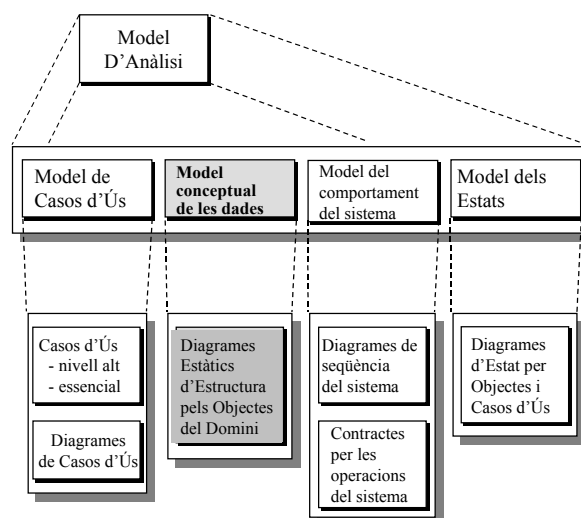
Bibliografia

- *Software Engineering: A Practitioner's Approach (5th edition)*
R. S. Pressman
Mc-Graw-Hill, 2001 (Cap. 20 i 21)
- *Object-Oriented Analysis and Design*
G.Booch
Benjamin/Cummings, 1994
- *Object Oriented Software Engineering: A Use Case Driven Approach*
I. Jacobson et al.
Addison-Wesley, 1992
- *Object-Oriented Modelling and Design*
J. Rumbaugh et al.
Prentice-Hall, 1991
- *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*
C. Larman
Prentice-Hall 2002
- *The Unified Software Development Process*
I.Jacobson, G.Booch, J.Rumbaugh
Addison-Wesley, 1999.

Model Conceptual de les Dades en UML

- Introducció
- Objectes i classes d'objectes
- Atributs
- Associacions
- Classe associativa
- Generalització/Especialització
- Agregació i composició
- Ampliacions
- Exemples

Model d'Anàlisi (especificació)



Model Conceptual de les Dades

Es la representació dels conceptes (objectes) significatius en el domini del problema.

Mostra, principalment:

- Classes d'objectes.
- Associacions entre classes d'objectes.
- Atributs de les classes d'objectes.
- Restriccions d'integritat

Objectes

Objecte:


Entitat que existeix al món real

Tenen identitat i són distingibles entre ells


l'avió amb
matrícula 327


una poma


un semàfor

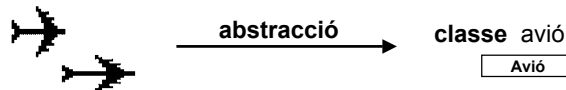

la factura 3443


l'avió amb
matrícula 999

Classe d'objectes

Classe d'objectes: descriu un conjunt d'objectes amb:

- les mateixes propietats
- comportament comú
- idèntica relació amb altres objectes
- semàntica comuna

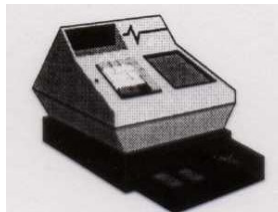


Abstracció: eliminar distincions entre objectes per a poder observar aspectes comuns

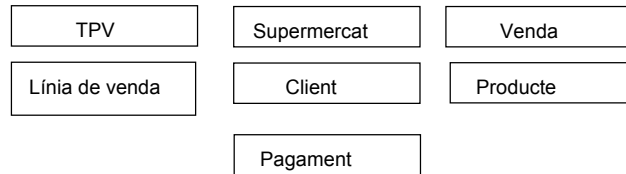
Els objectes d'una classe tenen les mateixes propietats i els mateixos patrons de comportament

Exemple: Terminal Punt de Venda

Un terminal de punt de venda (TPV) és un sistema computeritzat usat per enregistrar les vendes i gestionar pagaments. S'usa, principalment en supermercats i grans magatzems. Inclou components hardware (com l'ordinador i l'scanner del codi de barres) i software per executar el sistema. Se'ns demana que especifiquem el software d'aquest sistema.

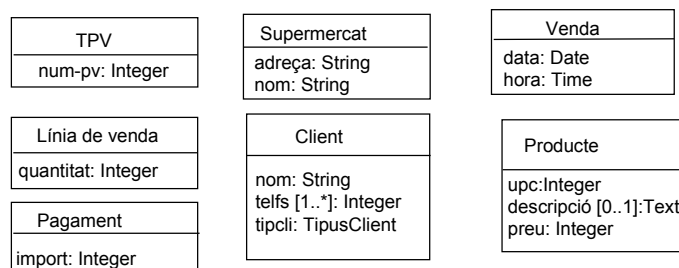


Exemple TPV: classes d'objectes



Exemple TPV: atributs

Un atribut és una propietat compartida pels objectes d'una classe

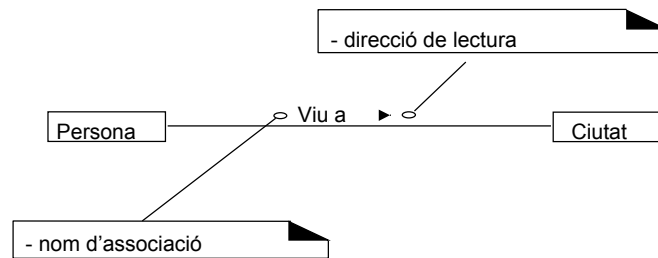


Els atributs:

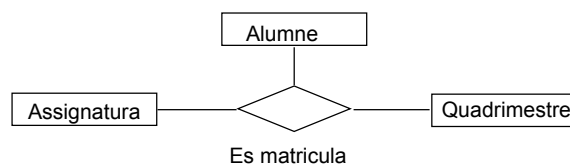
- Poden prendre valors **nuls** (per exemple: descripció)
- Poden ser **multivaluats** (per exemple: telfs)
- Poden ser **definits per l'usuari** mitjançant **enumeracions**
 - Per exemple, **TipusClient** es defineix a part com una **enumeració** amb els valors que pot tenir i que són: **Normal** i **Preferent**

Associacions

Es la representació de relacions entre dos o més objectes

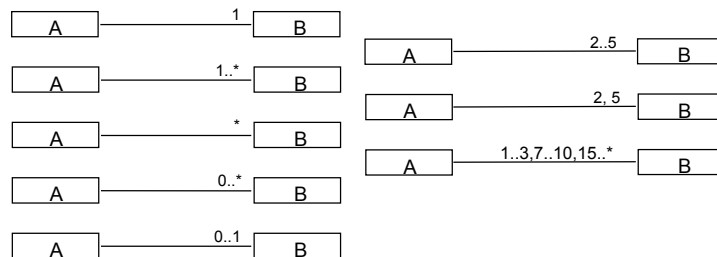


Associacions d'ordre superior a dos



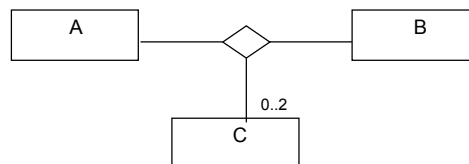
Multiplicitat de les associacions binàries

Donada una instància *a* de la classe A qualsevol, la multiplicitat del costat B defineix quantes instàncies de B es poden associar amb *a* en un moment de temps determinat

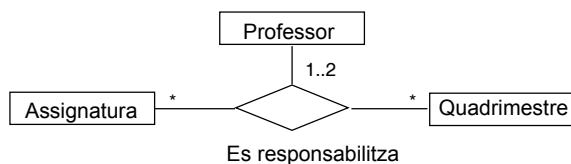


Multiplicitat a les associacions ternàries

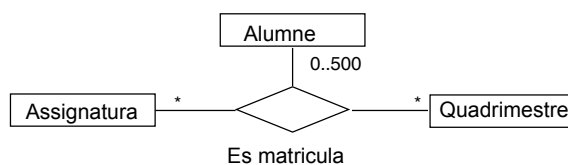
Donades una instància *a* d'A i una instància *b* de B qualssevol, la multiplicitat al costat C ens diu quantes instàncies de C es poden associar amb la parella (*a*,*b*).



Multiplicitat a les associacions ternàries – Exemples



Segons aquest esquema, per tota parella d'assignatura i quadrimestre, hi ha d'haver com a mínim un professor reponsable.

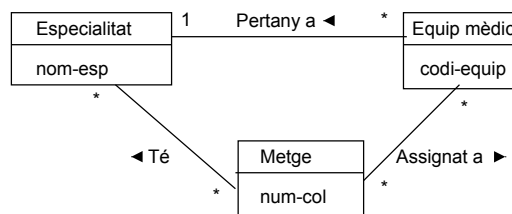


Aquest esquema permet que hi hagi alguna parella d'assignatura i quadrimestre, per la qual no hi ha cap alumne que s'hagi matriculat de l'assignatura en el quadrimestre.

Restriccions textuais

Les restriccions que no es poden especificar gràficament amb la notació UML s'especifiquen de forma textual

L'especificació textual es pot fer amb llenguatge natural, amb OCL, etc.



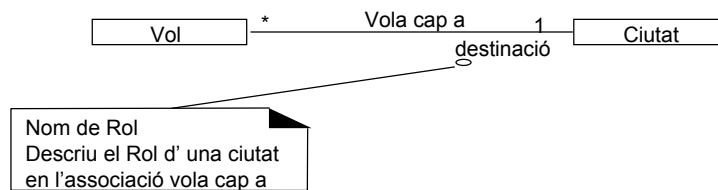
- 1- Dues especialitats diferents no poden tenir el mateix nom-esp
 - 2- Dos equips mèdics diferents no poden tenir el mateix codi-equip
 - 3- Dos metges diferents no poden tenir el mateix num-col
 - 4- Un metge no pot estar assignat a un equip mèdic que pertany a una especialitat que el metge no té
- Restriccions de clau externa*

Nom de rol a les associacions

Cada extrem d'una associació es un rol, que té diverses propietats com:

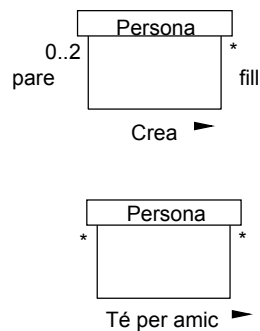
- nom
- multiplicitat

El nom de rol identifica un cap de l'associació i descriu el paper jugat pels objectes en l'associació.



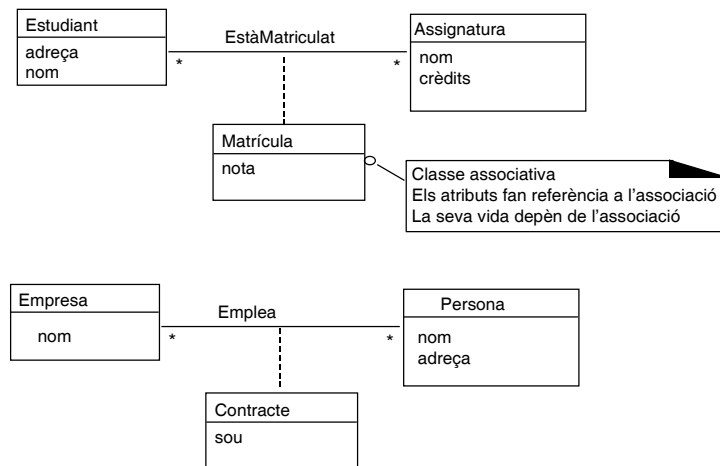
Associacions recursives

Associacions en les que una mateixa classe d'objectes hi participa més d'una vegada (amb papers diferents o no)

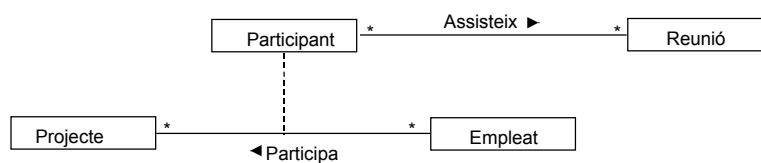


Classe associativa

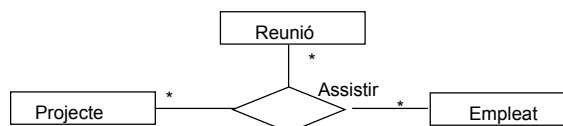
Representa una associació que es pot veure com una classe



Exemple de classe associativa

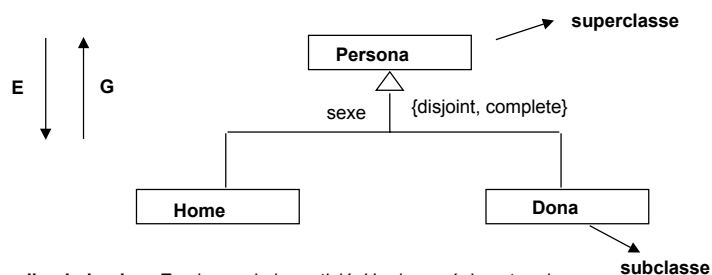


No és equivalent a:



Generalització / Especialització

Identificar elements comuns entre els objectes definint relacions de superclasse (objecte general) i subclasse (objecte especialitzat).



discriminador - És el nom de la partició. Ha de ser únic entre els atributs i rols de la superclasse

disjoint - Un descendent no ho pot ser en més d'una subclasse

overlapping - Un descendent pot ser de més d'una subclasse

complete - S'han especificat totes les subclasses

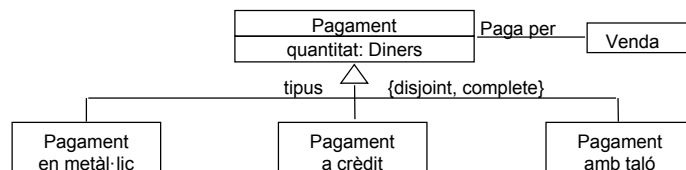
incomplete - La llista de subclasses és incompleta

La classificació pot ser **estàtica** o **dinàmica**

Generalització/Especialització

Permet entendre els conceptes en termes més generals, refinats i abstractes. Fa els diagrames més expressius.

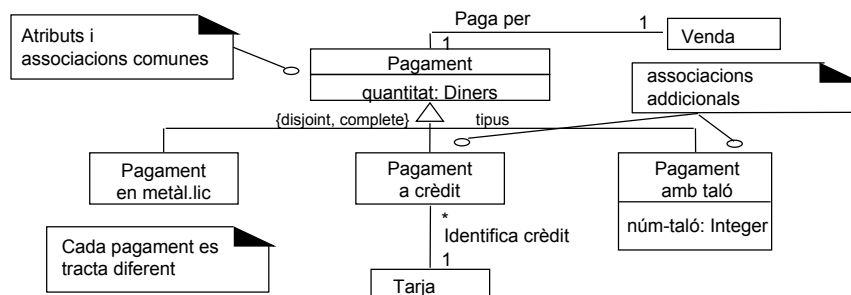
- Tots els objectes de la subclasse ho són també de la superclasse.
- La definició de la superclasse ha de ser aplicable a la subclasse.
 - atributs
 - associacions
 - restriccions
 - (- operacions)



Generalització / Especialització

Motivacions per particionar una classe en subclasses

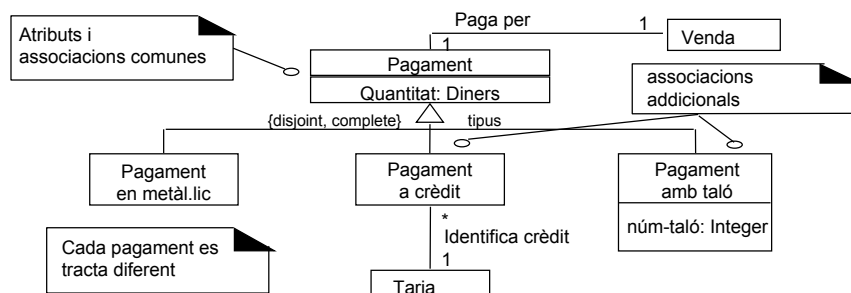
- La subclasse té atributs addicionals.
- La subclasse té associacions addicionals.
- La subclasse és tractada o manipulada de forma diferent a d'altres subclasses.
- La subclasse es comporta de manera diferent de la superclasse o d'altres subclasses.



Generalització / Especialització

Motivacions per definir una superclasse:

- Les subclasses potencials representen variacions d'un mateix concepte.
- Les subclasses tenen atributs que poden ser factoritzats i expressats a les superclasses.
- Les subclasses tenen associacions que poden ser factoritzades i relacionades amb la superclasse.



Agregació

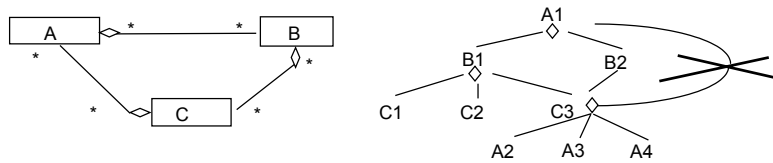
L'agregació és un tipus d'associació usada per modelar relacions "part-tot" entre objectes.

El "tot" s'anomena compostat i les "parts" components



La distinció entre associació i agregació és sovint subjectiva.

L'única restricció que afegeix l'agregació respecte l'associació és que les cadenes d'agregacions entre instàncies d'objectes no poden formar cicles.

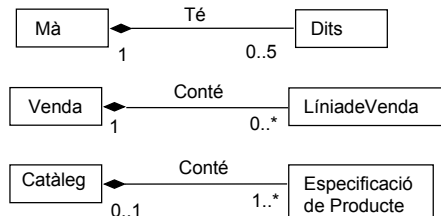


Composició

La composició és un tipus d'agregació per la qual:

- La multiplicitat del cap compost pot ser com a màxim 1 (com a màxim un compostat posseeix un component)

- Si un "component" està associat a un "composat" i el "composat" s'esborra aleshores el "component" també s'ha d'esborrar (no el pot sobreviure)



Agregació i composició - quan mostrar-la

Agregació/Composició

- Existeix un assemblatge tot-part físic o lògic.
- Algunes propietats del tot es propaguen a les parts (com destrucció, moviment, ...)

Composició

- La vida de la part està inclosa en la vida del compost.
- Existeix una dependència crear-esborrar de la part respecte del compost.

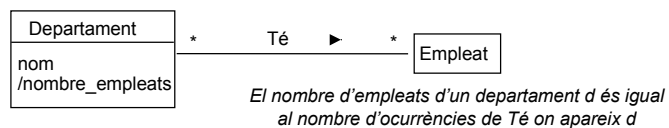
Informació derivada

Un element (atribut o associació) és **derivat** si es pot calcular a partir d'altres elements.

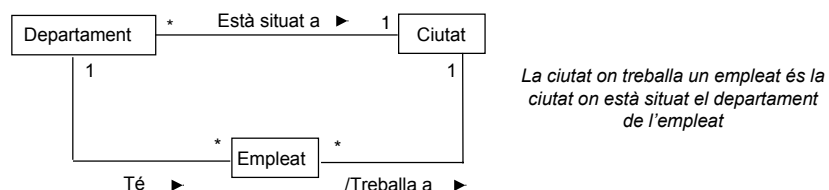
S'inclou quan millora la **claredat** del model conceptual

Una 'constraint' (regla de derivació) ha d'especificar com es deriva

Atribut derivat



Associació derivada

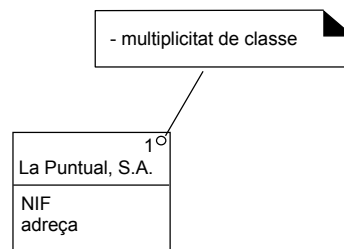


Multiplicitat de classe

La **multiplicitat de classe** estableix el rang de possibles cardinalitats per les instàncies d'una classe

Per defecte, és **indefinida**

En alguns casos, però, és útil establir una multiplicitat finita, **especialment** en casos de classes que poden tenir **una sola instància** (i que s'anomenen "singleton")



Altres restriccions sobre associacions

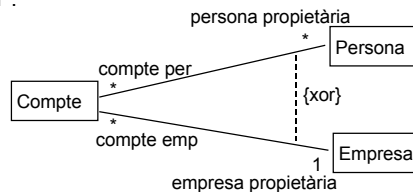
A part de la multiplicitat, és possible expressar altres restriccions sobre les associacions:

- Xor
- Subset

Xor

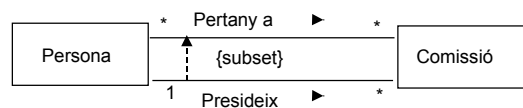
Uneix diverses associacions lligades a una mateixa classe bàsica

Una instància de la classe bàsica pot participar com a màxim en una de les associacions unides per "xor".



Subset

Indica que una associació és un subconjunt d'una altra associació

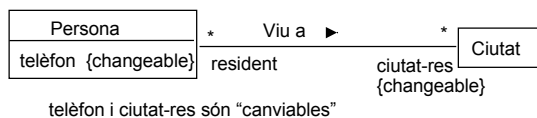


Canviabilitat

La canviabilitat indica si els valors d'un atribut o l'extrem d'una associació poden canviar o no

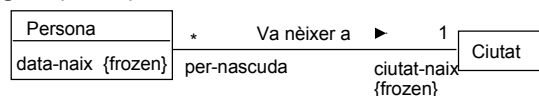
Canviable (changeable)

- opció per defecte -



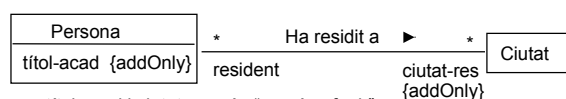
telèfon i ciutat-res són "canviables"

Congelat (frozen)



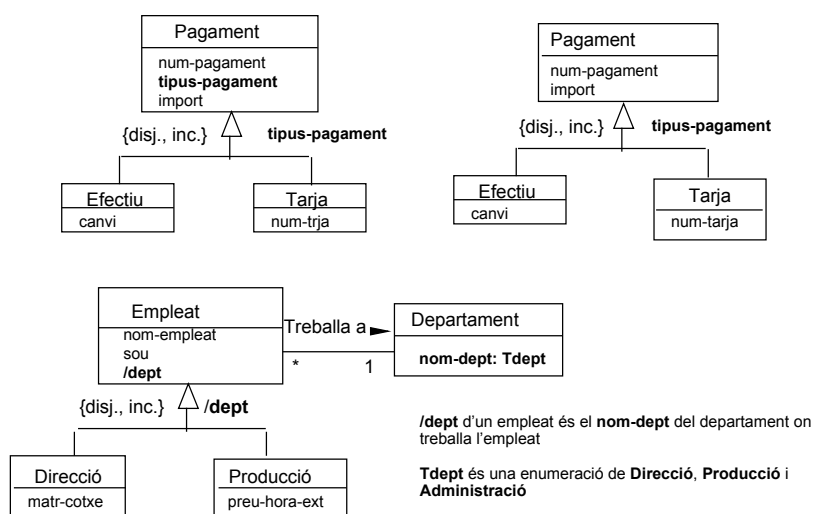
data-naix i ciutat-naix són "congelats"

Només-afegir (addOnly)



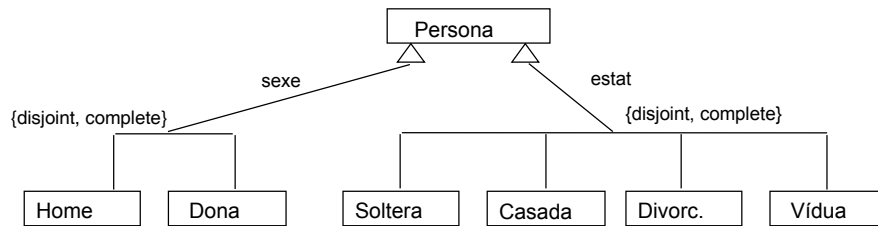
títol-acad i ciutat-res són "només-afegir"

Generalització/Especialització - Exemples discriminador



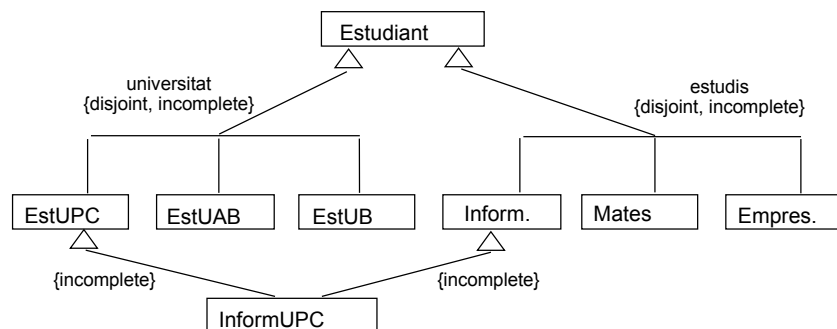
Classificació múltiple

Variant de generalització/especialització en la qual una superclasse pot tenir diverses jerarquies d'especialització en funció de diferents discriminadors



Herència múltiple

Variant de generalització/especialització en la qual una subclasse té més d'una superclasse

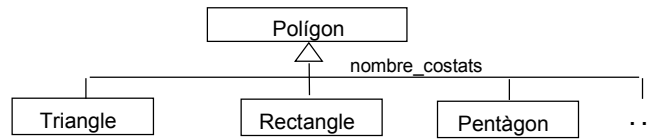


Només es pot utilitzar si no hi ha **conflictes d'herència**

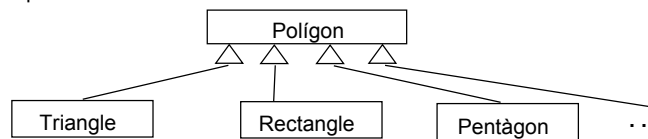
Conflicte d'herència: Una classe té més d'una superclasse amb un mateix atribut/associació/(operació) que no prové d'un únic antecessor

Equivalències notacionals (I)

En UML:



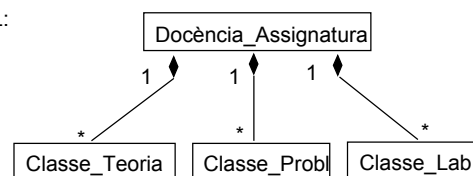
és equivalent a:



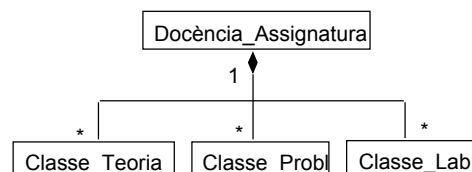
Problema?

Equivalències notacionals (II)

En UML:

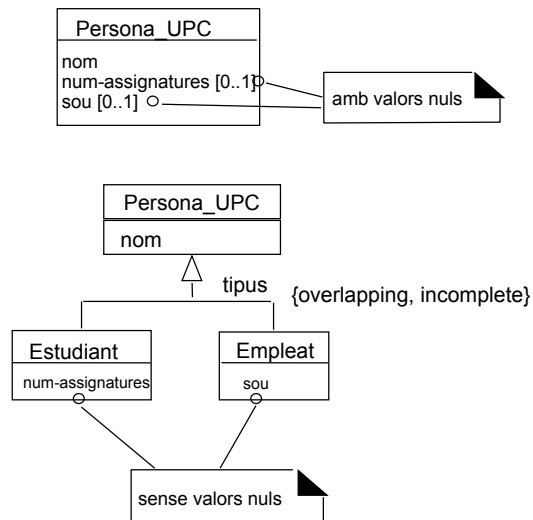


és equivalent a:



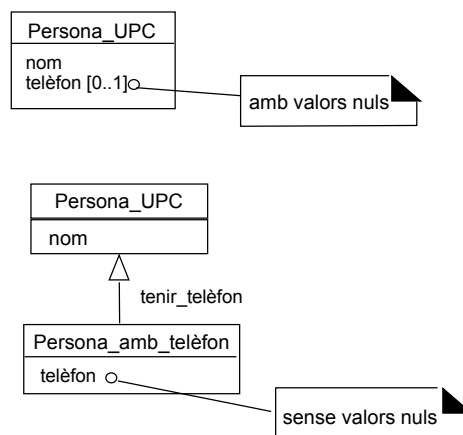
Exemples (I)

Quina solució és millor?

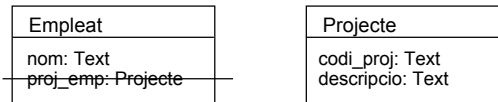


Exemples (II)

Quina solució és millor?



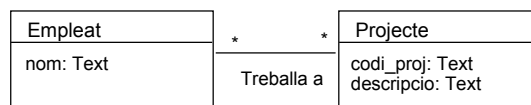
Exemples (III)



Un **atribut** no pot prendre **valors d'una de les classes** del model conceptual



Aquest cas és una **associació**



Bibliografia

- *The Unified Modeling Language User Guide*
G. Booch, J. Rumbaugh, I. Jacobson
Addison-Wesley, 1999
- *The Unified Modeling Language Reference Manual*
J. Rumbaugh, I. Jacobson, G. Booch
Addison-Wesley, 1999
- *Applying UML and Patterns An Introduction to Object-Oriented Analysis and Design and the Unified Process (second edition)*
C. Larman
Prentice-Hall 2002
Cap. 10, 11, 12, 26 i 27

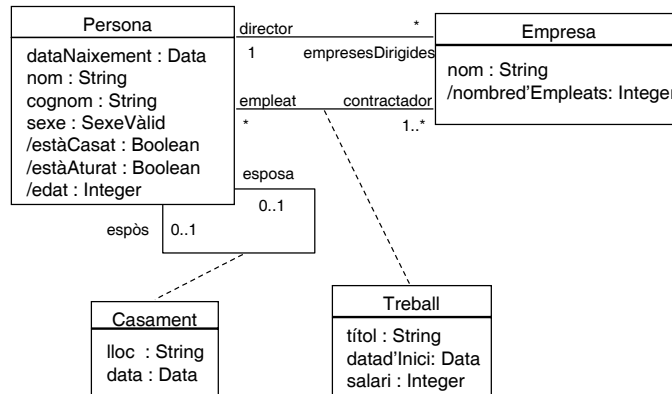
Object Constraint Language (OCL)

- Perquè serveix?
- Propietats del Model Conceptual
- Col·leccions: conjunts, bosses i seqüències
- Navegació per classes associatives
- Generalització / Especialització
- Com especificar en OCL

Perquè serveix OCL?

- Els models gràfics no són suficients per a una especificació precisa i no ambigua
- L'OCL:
 - és un llenguatge formal
 - és un llenguatge de navegació que permet definir expressions (o sigui, no té efectes laterals)
 - no és un llenguatge de programació, sinó d'especificació
 - és un llenguatge tipat
- S'usa per:
 - especificar invariants (restriccions i regles de derivació) del Model Conceptual
 - especificar precondicions, postcondicions i sortides de les operacions

Exemple



SexeVàlid es defineix com a enumeració d'Home i de Dona

Propietats dels objectes

- Cada expressió OCL:
 - s'escriu en el context d'una instància d'un tipus determinat
 - defineix una propietat d'aquesta instància
- Una propietat pot fer referència a:
 - atributs d'una classe d'objectes
 - navegació a través de les associacions

Propietats dels atributs d'una classe d'objectes:

- Propietat: edat d'una persona -- enter

context Persona inv

l'expressió ha de ser certa per a totes les instàncies de la classe

self.edat

instància contextual de l'expressió (punt de partida)
- Restricció de la propietat: "l'edat de les persones ha de ser superior o igual a zero"

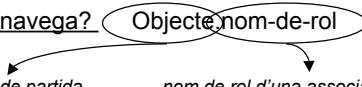
context Persona inv *o, alternativament:* **context** p:Persona inv

self.edat >= 0 p.edat >= 0

Propietats dels objectes (II)

Propietats d'una navegació a través d'associacions:

Partint d'un objecte concret, podem navegar a través de les associacions del Model Conceptual per referir-nos a d'altres objectes i a les seves propietats

- Com es navega? 

objecte de partida

nom de rol d'una associació de l'objecte

Resultat: conjunt d'objectes de l'altre extrem de nom-de-rol

- si no hi ha nom de rol especificat, es pot usar el nom de la classe d'objectes de l'altre extrem de l'associació (amb minúscules)

- Exemples de navegació:

```
context Empresa inv
  self.director -- director de l'empresa -- Persona
  self.director.nom -- nom del director -- String
  self.empleat -- empleats de l'empresa -- set(Persona)
  self.empleat.espos -- esposos dels empleats -- set(Persona)
```

Col·leccions: conjunts, bosses i seqüències

Una *col·lecció d'elements* pot ser del tipus:

- **Conjunt:** cada element ocorre una única vegada a la col·lecció
- **Bossa (multiconjunt):** la col·lecció pot contenir elements repetits
- **Seqüència:** bossa on els elements estan ordenats

Exemple:

- “nombre de treballadors diferents que treballen per a una persona”

```
context Persona inv
  num-treb = self.empresesDirigides.empleat -> size()
  Incorrecte: el resultat és una bossa i pot contenir repetits.
```

```
context Persona inv
  num-treb = self.empresesDirigides.empleat -> asSet() -> size()
```

Regles de navegació:

- si la multiplicitat de l'associació és 1 el resultat és un objecte (o un conjunt d'un únic objecte).
- si la multiplicitat de l'associació és >1 el resultat és un conjunt.
- si es navega per més d'una associació i la multiplicitat d'alguna d'elles és >1 el resultat és una bossa, encara que de vegades pot ser un conjunt.

Operacions bàsiques sobre col·leccions (I)

Select: especifica un subconjunt de la col·lecció

- “persones majors de 50 anys que treballen a una empresa”

context Empresa inv
self.empleat -> select(edat>50)

context Empresa inv
self.empleat -> select(p | p.edat>50)

context Empresa inv
self.empleat -> select(p:Persona | p.edat>50)

Collect: especifica una col·lecció que es deriva d'una altra, però que conté objectes diferents

- “edats (amb repetits) dels empleats d'una empresa”

context Empresa inv
self.empleat -> collect(dataNaixement)

versió simplificada: self.empleat.dataNaixement

Operacions bàsiques sobre col·leccions (II)

forAll: expressió que han de satisfer tots els elements

- “tots els empleats de l'empresa s'anomenen Jack”

context Empresa inv
self.empleat -> forAll(nom='Jack')

- “la clau externa d'Empresa és el seu nom”

context Empresa inv
Empresa.allInstances -> forAll(e1,e2 | e1<> e2 implies e1.nom<>e2.nom)

Exists: condició que satisfà almenys un element

- “com a mínim un empleat de l'empresa s'ha de dir Jack”

context Empresa inv
self.empleat -> exists(nom='Jack')

isUnique: retorna cert si l'expressió s'avalua a un valor diferent per cada element de la col·lecció

- “la clau externa d'Empresa és el seu nom”

context Empresa inv
Empresa.allInstances -> isUnique(nom)

Combinació de propietats

- Les propietats es poden combinar per formar expressions més complexes

- “Les persones casades han de ser majors d’edat”

context Persona inv

self.esposa -> notEmpty() implies self.esposa.edat >= 18
and self.espòs -> notEmpty() implies self.espòs.edat >= 18

- “Una empresa té com a màxim 50 empleats”

context Empresa inv

self.empleat -> size() <= 50

- “Una persona no pot tenir alhora un espòs i una esposa”

context Persona inv

not ((self.esposa -> size())=1) and (self.espòs -> size())=1

- “definició de l’atribut derivat nombred’Empleats”

context Empresa inv

nombred’Empleats = (self.empleat -> size())

- “definició de l’atribut derivat estàAturat”

context Persona inv

estàAturat = if self.contractador-> isEmpty() then true else false

Navegació per classes associatives

Navegació a una classe associativa:

Es fa servir el nom de la classe associativa (amb minúscula)

- “els sous de les persones que treballen a la UPC han de ser més alts que 1000”

context Persona inv

(self.contractador -> select(nom='UPC')).treball
-> forAll (t | t.salari > 1000)

Navegació des d’una classe associativa:

S’usa el nom de rol de l’extrem cap on es vol navegar

Si no s’ha especificat nom de rol, s’usa el nom de la classe.

- “les persones que treballen no poden estar aturades”

context Treball inv

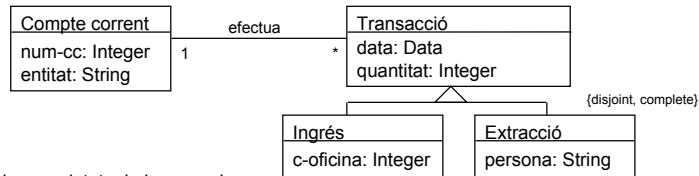
self.empleat.estàAturat = false

- “un casament ha de ser entre una dona i un home”

context Casament inv

self.esposa.sexe = #dona and self.espòs.sexe = #home

Generalització - Especialització (herència)



Navegació:

- *Accés directe a les propietats de la superclasse*
context Ingres inv
 self.compte-corrent.num-cc -- número de compte d'un ingrés
- *Accés a les propietats definides al nivell de subclasse*
context CompteCorrent inv
 self.transacció.oclAsType(Extracció).persona -> asSet() -- persones que han tret diners d'un compte corrent

Aspectes addicionals:

- *Selecció d'objectes que pertanyen a la subclasse*
context CompteCorrent inv
 self.transacció -> select(ocllsTypeOf=Ingres) -- ingressos d'un compte
- *Les propietats de les subclasses es poden ignorar*
context CompteCorrent inv
 self.transacció -> select(data.isafter(5-4-1998)) -- transaccions d'un compte posteriors al 5-4-1998

Com especificar en O.C.L.

- Una expressió O.C.L. s'especifica sempre començant en una classe d'objectes determinada: *instància contextual*
- Una *expressió* es pot especificar de diverses maneres, segons la instància contextual de partida.
 "els dos membres d'un matrimoni no poden treballar a la mateixa empresa"
context Empresa inv
 self.empleat.esposa -> intersection(self.empleat) -> isEmpty()
context Persona inv
 self.esposa.contractador -> intersection(self.contractador) -> isEmpty()
- Indicacions per escollir la instància contextual
 - si la restricció restringeix el valor de l'atribut d'una classe, aquesta és la classe candidata
 - si la restricció restringeix el valor dels atributs de més d'una classe, qualsevol d'aquestes n'és la candidata
 - normalment, qualsevol restricció hauria de navegar a través del menor nombre possible d'associacions

Com especificar en OCL: exemples

- “L’espòs i l’esposa d’un matrimoni han de ser majors d’edat”
 - context** Casament **inv**
self.esposa.edat >= 18 and self.espòs.edat >= 18 -- *és preferible a*
 - context** Persona **inv**
self.esposa -> notEmpty() implies self.esposa.edat >= 18 and
self.espòs -> notEmpty() implies self.espòs.edat >= 18
- “Totes les persones han de ser majors d’edat”
 - context** Persona **inv**
self.edat >= 18 -- *és preferible a*
 - context** Empresa **inv**
self.empleat -> forAll (edat >= 18)
- “Ningú no pot ser director i empleat d’una empresa”
 - context** Empresa **inv**
not(self.empleat -> includes(self.director))
 - context** Persona **inv**
not(self.empresesDirigides.empleat -> includes(self))
 - *quina és preferible en aquest cas?*

Operacions estàndard de tipus booleà

Operació	Notació	Resultat
or	a or b	booleà
and	a and b	booleà
or exclusiu	a xor b	booleà
negació	not a	booleà
igualtat	a = b	booleà
desigualtat	a <> b	booleà
implicació	a implies b	booleà
if-then-else	if a then b else b'	tipus de b o b'

Operacions estàndard de tipus string

Operació	Notació	Resultat
concatenació	string.concat(string)	string
tamany	string.size()	integer
substring	string.substring(int,int)	string
igualtat	string1 = string2	booleà
desigualtat	string1 <> string2	booleà

Operacions estàndard d'una classe d'objectes

Operació	Resultat
allInstances	retorna el conjunt de totes els elements de la classe d'objectes

Operacions estàndard de tipus col·lecció

Operació	Resultat
size()	nombre d'elements de la col·lecció
count(object)	nombre d'ocurrències de l'objecte
includes(object)	cert si l'objecte pertany a la col·lecció
includesAll(collection)	cert si els elements del paràmetre <i>collection</i> són a la col·lecció actual
excludes(object)	cert si l'objecte no pertany a la col·lecció
excludesAll(collection)	cert si els elements del paràmetre <i>collection</i> no són a la col·lecció actual
isEmpty()	cert si la col·lecció és buida
notEmpty()	cert si la col·lecció no és buida
sum()	suma de tots els elements (els elements s'han de poder sumar)
exists(expression)	<i>expression</i> és cert per algun element?
forAll(expression)	<i>expression</i> és cert per tots els elements?
isUnique(expression)	cert si <i>expression</i> avalua a un valor diferent per cada element de la col·lecció actual

Operacions estàndard (específiques) de tipus conjunt

Operació	Resultat
select(expression)	selecciona el subconjunt d'elements del conjunt actual per als quals <i>expression</i> és cert
reject(expression)	elimina el subconjunt d'elements del conjunt actual per als quals <i>expression</i> és cert
union(set)	resultat d'unir els dos conjunts
intersection(set)	resultat de la intersecció dels dos conjunts
union(bag)	resultat d'unir els conjunt actual amb el <i>bag</i>
intersection(bag)	resultat de la intersecció del conjunt actual amb el <i>bag</i>

Bibliografia

- OMG - Unified Modeling Language
Object Constraint Language Specification, v. 1.4.
Setembre 2001.
- J.Warmer; A.Kleppe
The Object Constraint Language: precise modeling with UML
Addison-Wesley, 1999.

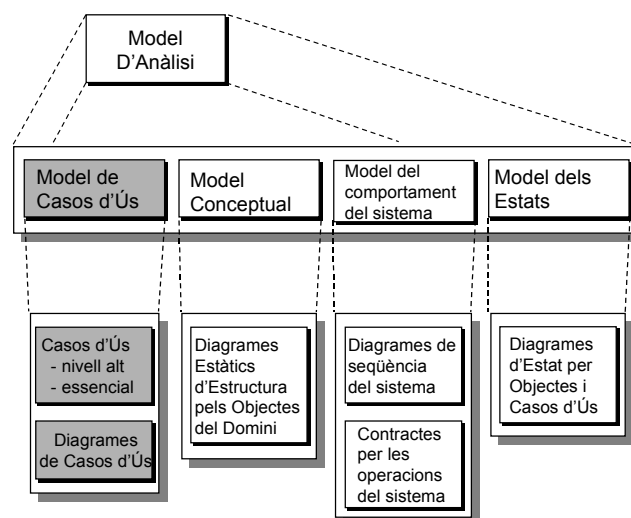
Pàgines web amb informació d'OCL:

- <http://www.omg.org>
- <http://www.software.ibm.com/ad/ocl>
- <http://www.rational.com>

Model de Casos d'Ús en UML

- Propòsit
- Casos d'ús
- Diagrama de Casos d'ús
- Especificació de Casos d'ús
- Estructuració de Casos d'ús
- Identificació de Casos d'ús

Model d'Anàlisi (especificació)



Determinació de requeriments d'un sistema software:

- Identificar i categoritzar les funcions del sistema (requeriments funcionals).
- Identificar i categoritzar els atributs del sistema (requeriments no funcionals).
- Relacionar els requeriments no funcionals amb els funcionals.

Especificació dels requeriments d'un sistema software:

- Comprensió dels requeriments.
- Organitzar els requeriments segons les funcions del sistema.
- Delimitar la frontera del sistema.



Model de Casos d'Ús: Quines son les funcions del sistema **PER CADA ACTOR**?
Èmfasi: USOS del sistema, valor afegit per cada actor

Casos d'Ús

Cas d'Ús: Document que descriu una seqüència d'esdeveniments que realitza un actor (agent extern) que usa el sistema per dur a terme un procés que té algun valor per a ell [Jacobson 92].

Extracció de
diners en
caixer

Actor: - Entitat externa al sistema que participa en la seqüència d'esdeveniments del cas d'ús.

- Pot ser una persona, un conjunt de persones, un sistema hardware, un sistema software o un rellotge.

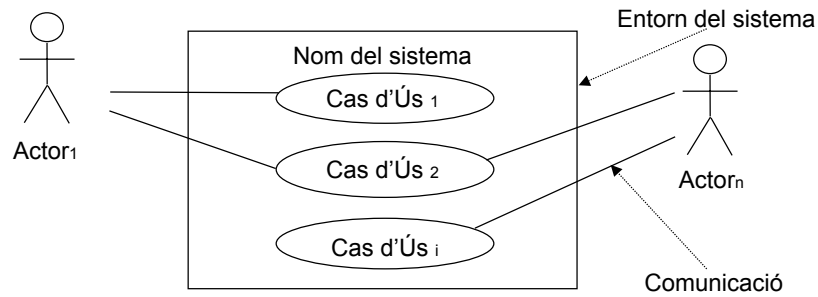
Iniciador: Genera l'estímul que provoca l'execució del procés (únic).

Participant: Intervé en el procés.



Diagrama de Casos d'Ús

Mostra conjuntament els diferents casos d'ús d'un sistema software, els actors i les relacions entre actors i casos d'ús.



Especificació de Casos d'Ús

D'alt nivell: Descripció breu de les accions del cas d'ús.

Cas d'ús: Nom del cas d'ús.

Actors: Llista d'actors, iniciador.

Propòsit: Objectiu del cas d'ús.

Resum: Descripció breu de les activitats que s'han de dur a terme.

Tipus: 1 - primari, secundari, opcional.

2 - real o essencial.

Expandida: Descripció detallada de les accions i els requeriments.

Afegeix a l'especificació d'alt nivell:

Referències creuades: Requeriments a què fa referència.

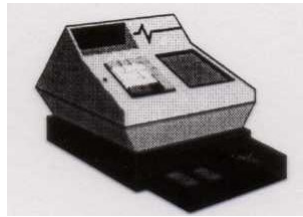
Curs típic d'esdeveniments: Descripció detallada de la interacció (conversa) entre els actors i el sistema.

Descripció dels esdeveniments pas a pas.

Cursos alternatius: Descriu excepcions al curs típic.

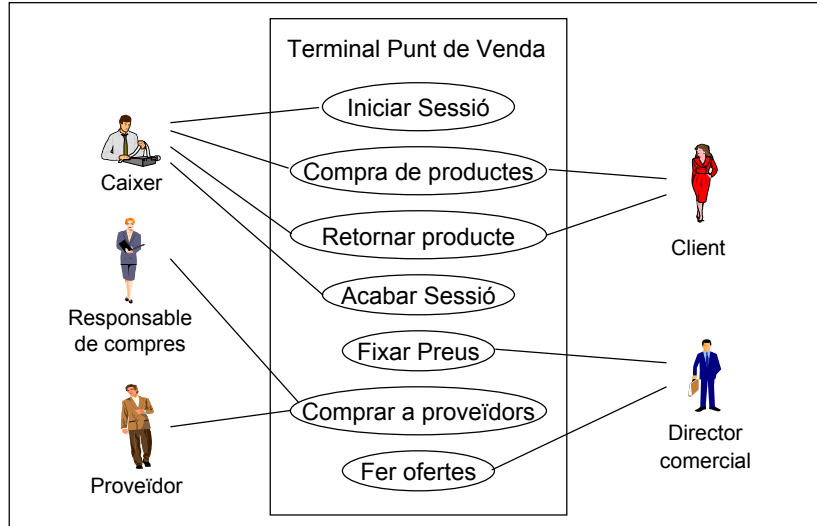
Exemple: Terminal de Punt de Venda

Un terminal de punt de venda (TPV) és un sistema computeritzat usat per enregistrar les vendes i gestionar pagaments. S'usa, principalment en supermercats i grans magatzems. Inclou components hardware (com l'ordinador i l'scanner del codi de barres) i software per executar el sistema. Se'ns demana que especifiquem el software d'aquest sistema.

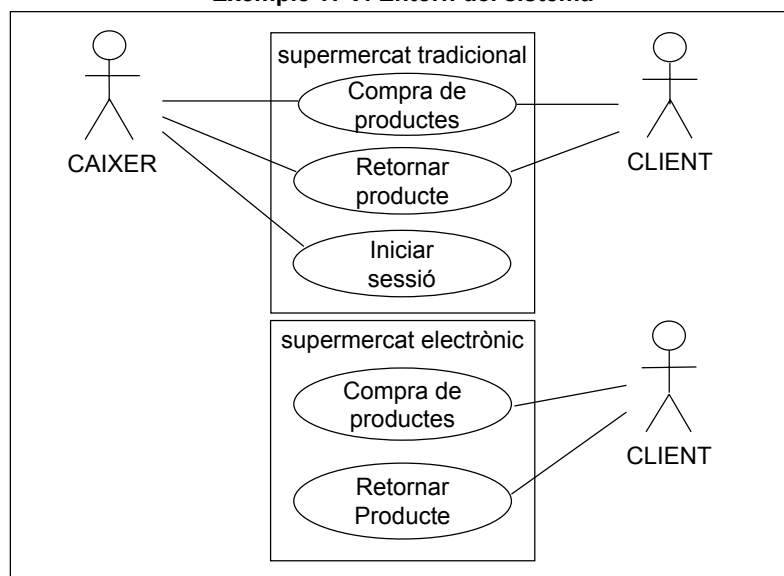


Ref #	Funció	Exemple TPV: Funcions bàsiques	Categoria
R1.1	Enregistrar la venda actual - els productes comprats.		evident
R1.2	Calcular el total de la venda actual, incloent impostos i càlcul de "punts de client".		evident
R1.3	Capturar la informació dels productes comprats d'un codi de barres, usant un scanner o bé a partir de l'entrada manual del codi de barres (Universal Product Code).		evident
R1.4	Descomptar les quantitats venudes de l'estoc, quan la venda es confirmi.		hidden
R1.5	Guardar informació sobre les vendes realitzades.		hidden
R1.6	El caixer ha d'identificar-se en iniciar una sessió amb un identificador i una clau d'accés.		evident
R1.7	Mostrar la descripció i el preu de cada producte comprat.		evident
R2.1	Tractar els pagaments en efectiu capturant la quantitat entregada pel client i calculant el canvi.		evident
R2.2	Tractar els pagaments amb tarja de crèdit capturant el número de la tarja des d'un lector de targes o manualment, demanar confirmació del pagament al servei d'autorització de crèdit (extern) amb una connexió via modem.		evident
R2.3	Enregistrar els pagaments amb tarja per tal que puguin ser facturats.		hidden

Exemple TPV: Diagrama de Casos d'Ús



Exemple TPV: Entorn del sistema



Exemple TPV: especificació del cas d'ús "compra de productes en efectiu"**Cas d'Ús:** Compra de productes en efectiu.**Actors:** Client (iniciador), Caixer.**Propòsit:** Capturar una venda i el seu pagament en efectiu.**Resum:** Un client arriba a la caixa amb productes per comprar.
El caixer enregistra els productes i gestiona el pagament en efectiu.
En acabar, el client se'n va amb els productes.**Tipus:** Primari i essencial.**Referències creuades:** R1.1, R1.2, R1.3, R1.7, R2.1**Curs típic d'esdeveniments****Accions dels Actors****Resposta del sistema**

- | | |
|---|---|
| <ol style="list-style-type: none"> 1. El cas d'ús comença quan un Client arriba a la caixa amb els productes per comprar. 2. El Caixer indica que comença una nova venda 4. El Caixer enregistra l'identificador de cada producte.
Si hi ha més d'una unitat del producte el Caixer pot entrar la quantitat. 6. En acabar l'entrada de productes el Caixer ho indica. | <ol style="list-style-type: none"> 3. Enregistra l'inici d'una nova venda del TPV 5. Determina el preu del producte i afegeix la seva informació al compte. 7. Calcula i mostra el total del compte. |
|---|---|

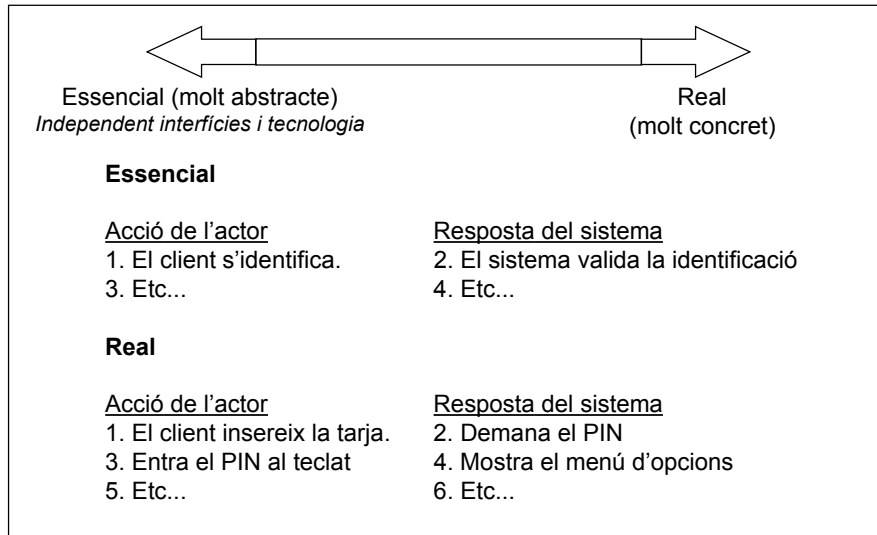
*(continua)***Accions dels Actors****Resposta del sistema**

- | | |
|--|---|
| <ol style="list-style-type: none"> 8. El Caixer li diu el total al client. 9. El Client entrega una quantitat de diners possiblement més gran que el total del compte. 10. El Caixer indica els diners que ha rebut. 13. El Caixer diposita els diners rebuts a la caixa i extreu el canvi.
El Caixer dóna el canvi i el rebut al Client. 14. El Client se'n va amb els productes comprats. | <ol style="list-style-type: none"> 11. Calcula i mostra el canvi al Client.
Imprimeix un rebut. 12. Enregistra la venda que s'acaba de fer. |
|--|---|

Cursos Alternatius

- Línia 4: S'entra un identificador de producte inexistent. Indicar error.
- Línia 9: El Client no té prou diners. Cancel·la la venda.

Casos d'ús essencials versus casos d'ús reals



Estructuració de Casos d'ús: seccions

Cas d'Ús: Compra de productes

Propòsit: Capturar una venda i el seu pagament en efectiu o amb tarja

Curs típic d'esdeveniments

Accions dels Actors

1. El cas d'ús comença quan un Client arriba a la caixa amb els productes per comprar.
2. (passos intermedis exclosos)...
9. El Client escull la forma de pagament:
 - a. **If** en efectiu, **see section** pagar en efectiu
 - b. **If** amb tarja **see section** pagar amb tarja
12. El Caixer dona el rebut al client.
13. El Client se'n va amb els productes comprats.

Resposta del sistema

10. Enregistra la venda que s'acaba de fer.
11. Imprimeix un rebut.

Estructuració de Casos d'ús: seccions

Section: Pagar en efectiu.

Curs típic d'esdeveniments

Accions dels actors

Resposta del sistema

- | | |
|--|---|
| <ol style="list-style-type: none"> 1. El Client entrega una quantitat de diners possiblement més gran que el total del compte. 2. El Caixer indica els diners que ha rebut. 4. El Caixer diposita els diners rebuts i extreu el canvi.
El Caixer dona el canvi al Client. | <ol style="list-style-type: none"> 3. Calcula i mostra el canvi al Client. |
|--|---|

Cursos Alternatius

- Línia 4: Efectiu insuficient per tornar el canvi. Demanar canvi a algú altre.

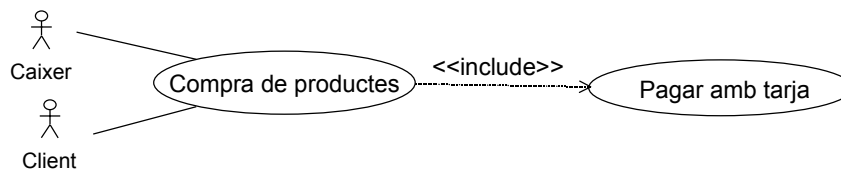
Section: Pagar amb tarja.

Cursos típics i alternatius per l'història del pagament amb tarja.

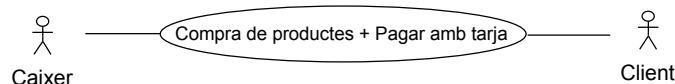
Estructuració de casos d'ús: relació <<include>>

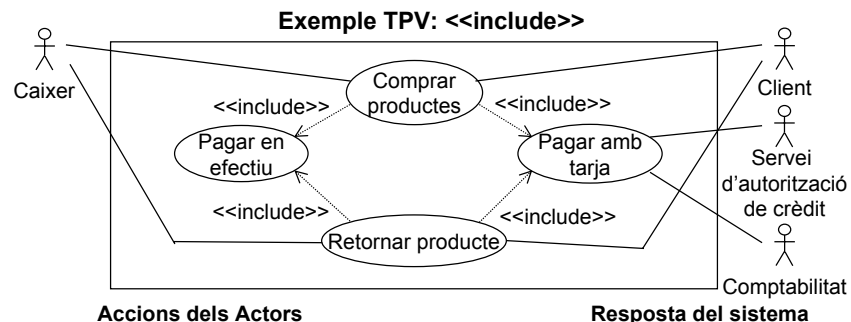
<<include>>: Relació d'un cas d'ús concret amb un d'abstracte, en la qual la conducta definida pel cas concret inclou (empra) la conducta definida a l'abstracte.

Permet reduir la redundància quan una seqüència d'accions és compartida per diversos casos d'ús



cas d'ús que s'efectua realment



**Accions dels Actors****Resposta del sistema**

1. El cas d'Ús comença quan un client arriba a la caixa amb els productes per comprar.
2. (Passos intermedis exclosos)...
9. El Client escull el tipus de pagament:
 - a. **If** és en efectiu **include** Pagar en efectiu
 - b. **If** és amb tarja **include** Pagar amb tarja
10. Enregistra la venda que s'acaba de fer.
11. Imprimeix un rebut.
12. El Caixer dóna el rebut al client.
13. El Client se'n va amb els productes comprats

Identificació de casos d'ús**Mètode basat en els actors**

1. Identificar els actors relatius al sistema.
2. Per cada actor, identificar els processos que inicia o en els quals participa.

Mètode basat en els esdeveniments

1. Identificar els esdeveniments externs als que el sistema ha de respondre.
2. Relacionar els esdeveniments amb els actors i casos d'ús.

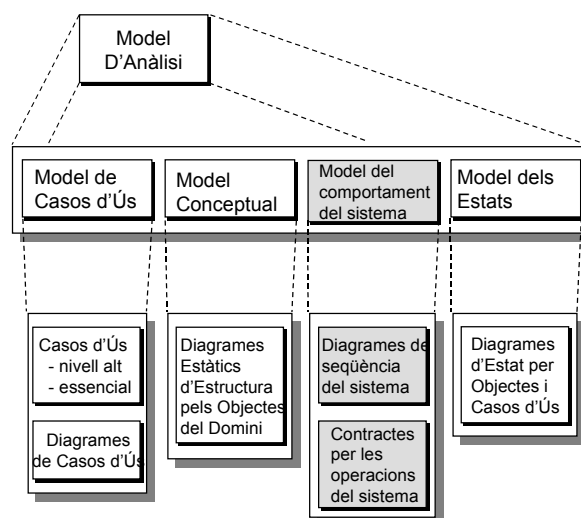
Bibliografia

- C. Larman
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (second edition)
Prentice-Hall, 2002. (Cap. 6, 9, 25)
- A. Cockburn
Writing Effective Use Cases
Addison-Wesley, 2001.
- I.Jacobson, M.Christerson, P.Johnson, G.Övergaard
Object-Oriented Software Engineering, a Use-Case Driven Approach
Addison-Wesley, 1992.
- I.Jacobson, G.Booch, J.Rumbaugh
The Unified Software Development Process
Addison-Wesley, 1999. (Cap. 6,7)

Model del comportament en UML

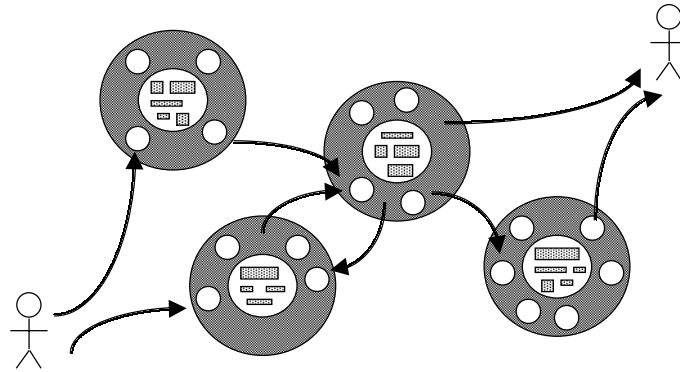
- Introducció
- Diagrames de seqüència del sistema
- Contractes de les operacions del sistema
- Altres consideracions
 - Compartició d'informació entre les operacions d'un diagrama
 - Informació elemental vs informació composta
 - Nombre d'esdeveniments del diagrama de seqüència
 - Redundància entre els models
- Bibliografia

Model d'Anàlisi (Especificació)

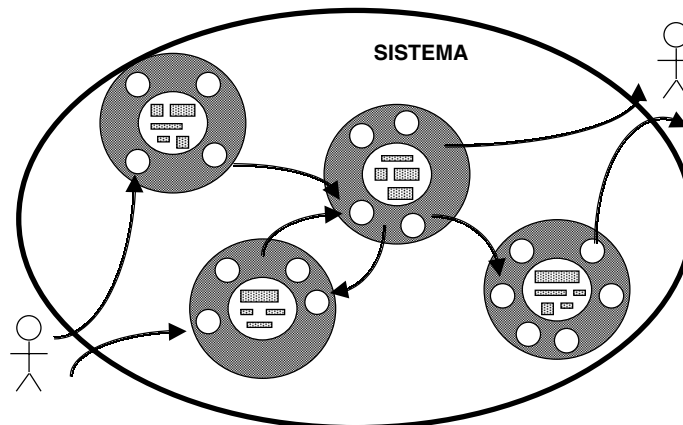


Descripció del comportament en OO

Els objectes es comuniquen mitjançant la invocació d'operacions d'altres objectes



“Especificació” del comportament en OO



- Considerem un tipus especial “**sistema**” que engloba tots els objectes
- L’**especificació del comportament** es fa amb el **model del comportament del “sistema”**

Model del comportament del sistema

- **Diagrames de seqüència del sistema:**
 - Mostren la seqüència d'esdeveniments entre els actors i el sistema.
 - Permeten identificar les operacions del sistema
- **Contractes per les operacions del sistema:**
 - Descriuen l'efecte de les operacions del sistema

Diagrames de seqüència del sistema

- **Objectius:**
 - identificar els esdeveniments i les operacions del sistema
- **Punt de partida:**
 - casos d'ús
 - la descripció dels diagrames de seqüència del sistema és posterior a la descripció dels casos d'ús
- **Casos d'ús:**
 - descriuen com els actors interaccionen amb el sistema software
 - l'actor genera esdeveniments cap al sistema que exigeixen l'execució d'alguna operació com a resposta (durant la interacció)
 - a partir dels casos d'ús podem identificar quins són els esdeveniments que van dels actors cap al sistema

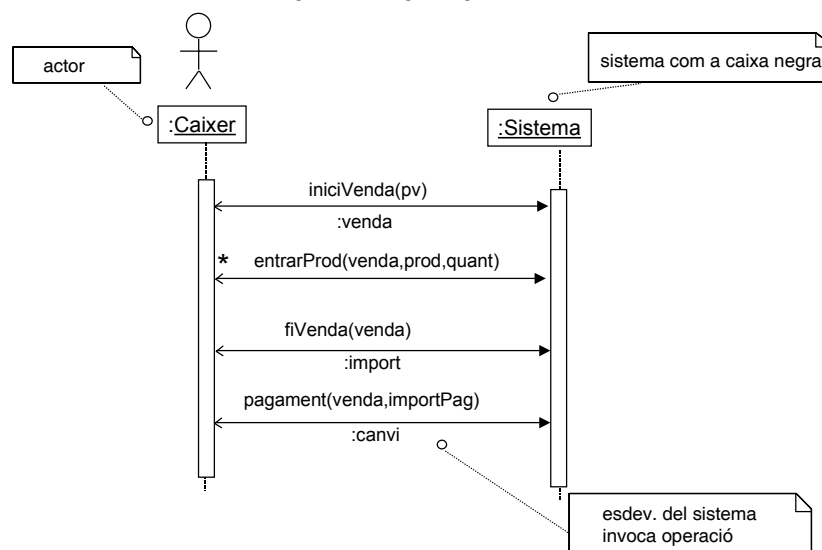


diagrames de seqüència del sistema

Diagrames de seqüència del sistema

- **Mostra**, per a un escenari particular d'un cas d'ús :
 - els esdeveniments generats pels actors externs
 - el seu ordre
 - els esdeveniments interns al sistema (operacions) que resulten de la invocació
- Definirem un diagrama de seqüència per cada **curs rellevant** d'esdeveniments d'un cas d'ús

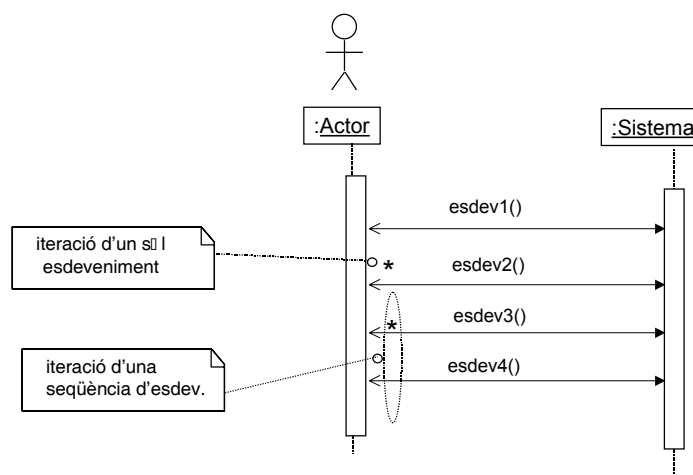
Exemple: comprar producte



Construcció d'un diagrama de seqüència

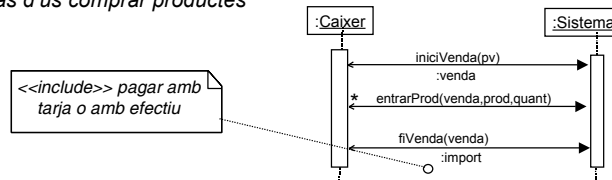
1. Dibuixar una línia vertical que representa el **sistema**
2. Dibuixar una línia per cada **actor** que interacciona **directament** amb el sistema
3. Del curs d'esdev. del cas d'ús, **identificar** els **esdeveniments** externs generats pels actors. Mostrar-los al diagrama

Representació d'iteracions d'esdeveniments

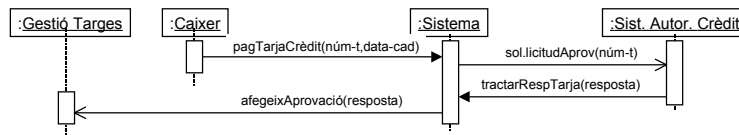


Diagrames de seqüència: <<include>>

- Els casos d'ús definits mitjançant <<include>> requereixen un diagrama de seqüència per la part comuna i un per cada cas d'ús que és inclòs.
- Exemple: cas d'ús comprar productes*

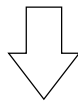


Part específica: pagar amb tarja



Esdeveniments i operacions

- Esdeveniment del sistema:** Esdeveniment extern generat per un actor
- Operació del sistema:** Operació interna que s'executa com a resposta a la comunicació de l'esdeveniment

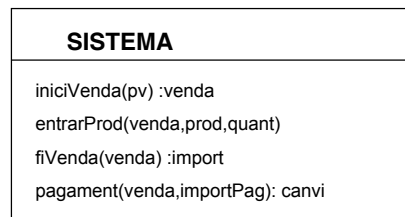


La comunicació d'un esdeveniment del sistema provoca l'execució d'una operació del sistema amb el **mateix nom** i els **mateixos paràmetres**

Operacions del sistema

- Les operacions del sistema **s'agrupen** com a operacions del tipus especial "sistema"
- En canvi, les operacions **no s'assignen a objectes** concrets durant l'etapa d'especificació

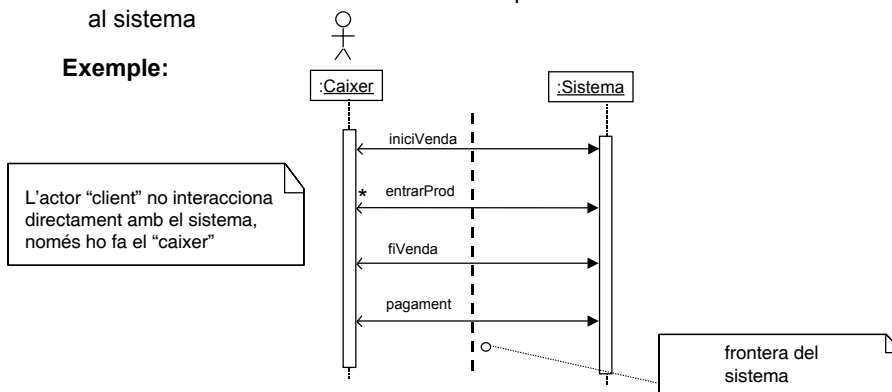
Exemple:



Esdeveniments i el límit del sistema

- Per identificar els esdeveniments del sistema és necessari haver delimitat clarament la **frontera** del sistema
- Els esdeveniments del sistema són els que estimulen **directament** al sistema

Exemple:



Contractes de les operacions

- **Contracte d'una operació**
 Descriu el comportament del sistema en termes de:
 - quins són els **canvis d'estat** de la base d'informació
 - quines són les **sortides** que el sistema proporciona quan s'invoca l'operació
- El tipus de descripció és **declaratiu**:
 - l'èmfasi es posa en el **què** farà l'operació més que en el **com** ho farà
- Els contractes de les operacions **inclouen** primordialment:
 - **precondicions** i **postcondicions** que descriuen els canvis d'estat
 - **sortides**

Contractes de les operacions: components

Name: nom i arguments de l'operació (*signatura de l'operació*)

Responsibilities:

Descripció informal del propòsit de l'operació

Exceptions:

Descripció de la reacció del sistema a situacions excepcionals

Preconditions:

Assumpcions sobre l'estat del sistema abans de la invocació de l'operació

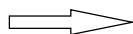
Postconditions:

Canvis d'estat que s'han produït:

- altes/baixes d'instàncies de classes d'objectes
- altes/baixes d'instàncies d'associacions
- modificació d'atributs
- generalització d'un objecte
- especialització d'un objecte
- canvi de subclasse d'un objecte

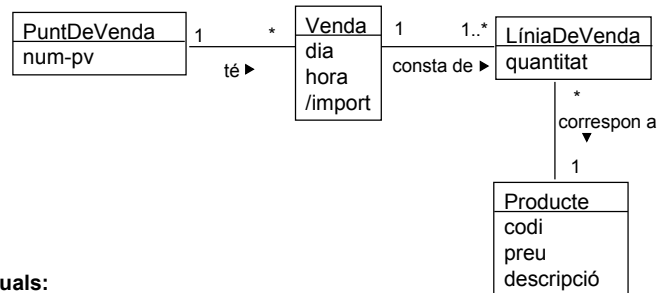
Sortida:

Descripció de la sortida que proporciona l'operació en pseudo-OCL



Observeu el lligam que existeix entre els **contractes** de les operacions i l'**esquema conceptual**

Exemple: esquema conceptual de partida



RI textuais:

- 1- La clau externa de PuntDeVenda és num-pv
- 2- La clau externa de Producte és codi
- 3- Un punt de venda no pot tenir més d'una venda amb el mateix dia i hora

Exemple: operació iniciVenda

Name: iniciVenda(pv) :venda

Responsibilities:

Iniciar l'enregistrament d'una venda

Exceptions:

Si no existeix cap PuntDeVenda amb num-pv=pv, indicar error

Preconditions:

Existeix un PuntDeVenda amb num-pv=pv

Postconditions:

- alta d'una instància V de Venda amb el dia i l'hora actuals
- alta d'una instància de l'associació 'té' que associa la venda V i la instància de PuntDeVenda amb num-pv=pv

Sortida: V

Exemple: operació entrarProd

Name: entrarProd(venda,prod,quant)

Responsibilities:

Enregistrar un línia d'una venda

Exceptions:

Si no existeix cap Producte amb codi=prod, indicar error

Preconditions:

Existeix un Producte amb codi=prod

Postconditions:

- alta d'una instància de LÍniaDeVenda L amb quantitat=quant
- alta d'una instància de l'associació 'consta de' que associa L i venda
- alta d'una instància de l'associació 'correspon a' que associa L i el producte amb codi=prod

Sortida:

Exemple: operació fiVenda

Name: fiVenda(venda) :import

Responsibilities:

Finalitzar l'enregistrament d'una venda i mostrar l'import a pagar

Exceptions:

Preconditions:

Postconditions:

Sortida: import = venda.import

Exemple: operació pagament

Name: pagament(venda,importPag): canvi

Responsibilities:

Mostrar el canvi a retornar

Exceptions:

Si importPag < venda.import indicar error

Preconditions:

importPag \geq venda.import

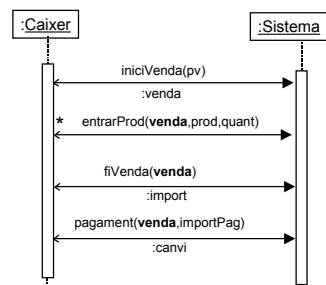
Postconditions:

Sortida: canvi = importPag - venda.import

Compartició d'informació entre les operacions d'un diagrama

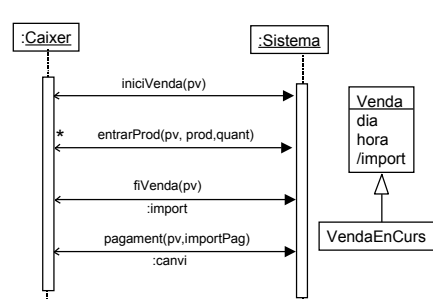
- UML no precisa de quina manera les operacions d'un diagrama de seqüència poden compartir informació
- Dues possibles solucions són:

**Mitjançant arguments
addicionals de les operacions**



Restricció implícita: El valor de l'argument 'venda' és el mateix a tots els esdeveniments del diagrama

**Mitjançant informació addicional a
l'esquema conceptual**



Informació Elemental vs Informació Composta

- La informació tractada per una operació s'expressa tant a nivell dels paràmetres com de la sortida que apareixen a la signatura de l'operació
- Hi ha dos tipus d'informació:
 - **Informació Elemental:** conté un únic element d'informació indivisible
 - Propietat
 - Classe d'objectes predefinida: Integer, Real, ...
 - **Informació Composta:** és una composició d'informacions elementals (i, per tant, cal especificar com es defineix la composició)

Exemple: ResumVendes(num-pv) que retorna un llistat de vendes amb la informació:

Per cada Producte p venut en aquell PuntDeVenda num-pv mostrar

- el codi del producte
- la quantitat total venuda de p a num-pv

ResumVendes (num-pv:Integer): ???

Informació Composta - Definició

Problema: com s'especifica el contingut d'una informació composta?

- En l'actualitat, UML no proposa cap solució
- Utilitzarem una adaptació de la definició de fluxes de dades que es fa en l'anàlisi estructurada (Yourdon, 1993)
- Mecanismes bàsics de definició d'informació composta
 - Inclusió: $i1 = i2 + i3 + i4$
 - Selecció: $i1 = [i2 \mid i3 \mid i4]$
 - Repetició: $i1 = \{i2 + i3 + i4\}$
 - Opcionalitat: $i1 = i2 + (i3 + i4)$

Exemple:

LlistatVendes = num-pv + {codi-prod + quantitat}

num-pv = Integer; codi-prod = Integer; quantitat = Integer

ResumVendes(num-pv:Integer) : LlistatVendes

Informació composta - Contractes de les operacions

- Les operacions necessitaran mecanismes per manipular (accedir, construir, etc.) la informació composta que apareix a la seva signatura
- Es necessita estendre OCL per poder manipular informació composta

Exemple: contracte de l'operació ResumVendes (num-pv): LlistatVendes

Nom: ResumVendes (num-pv): LlistatVendes

Responsabilitats: emetre el resum de vendes demanat

Tipus: sistema

Excepcions: Si no existeix un punt de venda num-pv, aleshores error

Preconditions: Existeix un punt de venda num-pv

Postconditions: -

Sortida:

Mostrar (num-pv)

Per cada producte p resultant de

(Producte.allInstances ->

select (p | p.LiniaDeVenda.Venda.PuntDeVenda.num-pv->includes(num-pv))

Per

Qt = (p.LíniesdeVenda ->

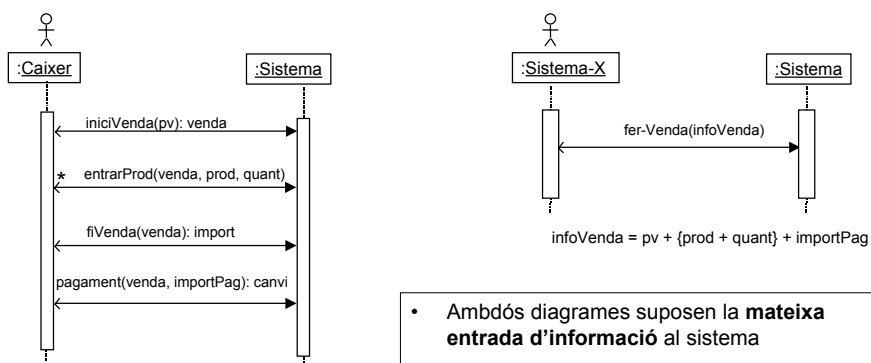
(select (lv | lv.Venda.PuntdeVenda.num-pv = num-pv).quantitat) -> Sum)

Mostrar (p.codi-prod)

Mostrar (Qt)

Diagrama de seqüència: quants esdeveniments?

- El nombre d'esdeveniments d'un diagrama de seqüència depèn de **com es produeix la interacció** entre els actors i el sistema software.



- Ambdós diagrames suposen la **mateixa entrada d'informació** al sistema
- Els dos diagrames de seqüència poden ser **correctes, segons les circumstàncies**

Redundància - Exemple

- L'esquema Conceptual conté **restriccions d'integritat** (gràfiques i textuais)
- Els contractes de les operacions tenen **precondicions**, que són requeriments del contingut de l'esquema conceptual per poder executar una transacció



Cal que les precondicions incloguin la comprovació de les restriccions del M. Conceptual?

Empleat
codi-emp
sou

R.I. textual: dos empleats
no poden tenir el mateix codi

Nom: AltaEmpleat (codi-emp, sou)
Responsabilitats: donar d'alta l'empleat
Excepcions: -
Preconditions: no existeix Empleat amb codi-emp
Postconditions:
 creació d'un nou objecte Empleat amb
 codi-emp i sou
Sortida: -



Cal posar aquesta precondició ???

Redundància - Definició

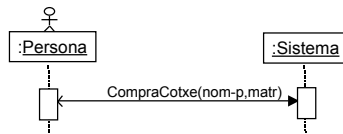
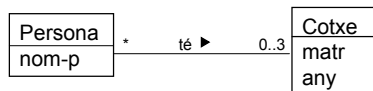
- Una especificació és redundant si un **mateix aspecte** del sistema software **està especificat diverses vegades**.
- La redundància **dificulta la modificabilitat** de l'especificació perquè si varia aquell aspecte cal modificar tots els models que hi fan referència.



L'especificació no hauria de ser redundant !!!

- Redundàncies possibles:
 - Entre l'Esquema Conceptual i els Contractes
 - Entre els Diagrames de Seqüència i els Contractes
 - ...

Redundància - Esq. Conceptual i Contractes (I)



• **Significat habitual:**

Nom: CompraCotxe (nom-p, matr)
Responsabilitats: compra d'un cotxe
Excepcions:
 ... (les que es dedueixen de les prec.)
Preconditions:
 - existeix Persona p amb nom-p
 - existeix Cotxe c amb matr
 - p no té c
 - ~~p no té ja 3 cotxes~~
Postconditions:
 - creació d'una associació té entre p i c
Sortida: -

• **Significat alternatiu:**

Nom: CompraCotxe (nom-p, matr)
Responsabilitats: compra d'un cotxe
Excepcions:
 ... (les que es dedueixen de les prec.)
Preconditions:
 - existeix Persona p amb nom-p
 - existeix Cotxe c amb matr
 - p no té c
Postconditions:
 - Si p té ja 3 cotxes llavors
 eliminar l'associació entre p i el
 cotxe c' de més antiguitat
 - creació d'una associació té entre p i c
Sortida: -

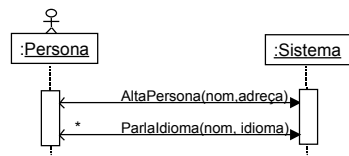
Redundància - Esq. Conceptual i Contractes (II)

- *Com s'han d'interpretar els contractes en relació al Model Conceptual?*
 - Precondicions:
 - **què** ha de contenir el Model Conceptual per **intentar executar una operació**
 - Restriccions del Model Conceptual:
 - Estan garantides després de l'execució de totes les operacions que participen en un cas d'ús
 - Es rebutgen totes les operacions d'un cas d'ús si la seva execució viola (globalment) alguna restricció d'integritat del Model Conceptual

Redundància - Esq. Conceptual i Contractes (III)



Cas d'ús: Afegir-Persona-1



Nom: AltaPersona(nom,adreça)

...

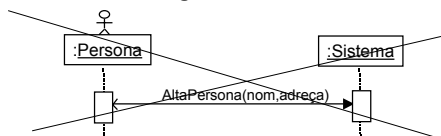
Preconditions:

Postconditions:

- creació d'un objecte Persona amb el nom i l'adreça especificats

Sortida:

Cas d'ús: Afegir-Persona-2



No permet afegir mai una persona !

Nom: Parlaldioma(nom,idioma)

...

Preconditions:

- existeix idioma amb nom=idioma
- no existeix l'associació Parla entre nom i idioma

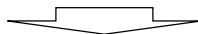
Postconditions:

- creació d'una associació parla entre la persona amb nom=nom i l'idioma

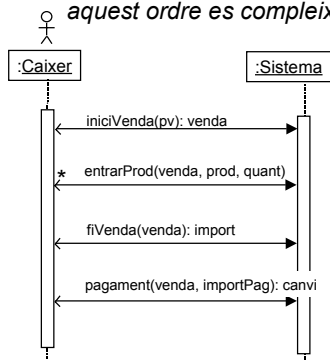
Sortida:

Redundància - Diagrames de Seqüència i Contractes

Els diagrames de seqüència defineixen un **ordre d'invocació** de les operacions



*Les operacions **no han d'incorporar informació** per garantir que aquest ordre es compleixi*



Nom: Pagament (venda, importPag): canvi

...

Preconditions:

- existeix venda "venda"
- la venda "venda" ha finalitzat

Postconditions:

...

Sortida:

són redundants

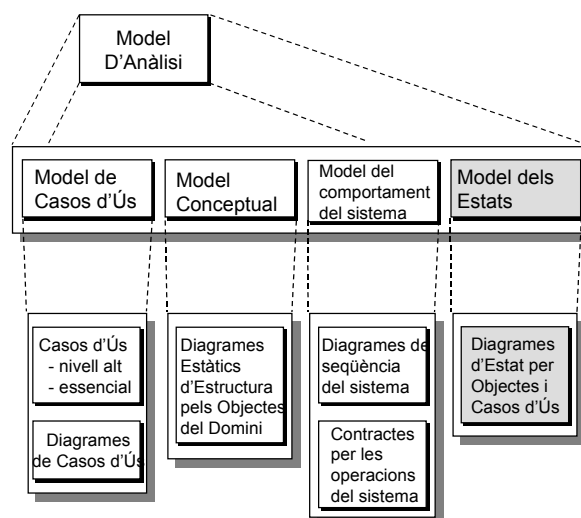
Bibliografia

- C. Larman
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (second edition)
Prentice-Hall, 2002. (Cap. 13, 15)
- J. Rumbaugh, I. Jacobson, G. Booch
The Unified Modeling Language Reference Manual
Addison-Wesley, 1999.
- G. Booch, J. Rumbaugh, I. Jacobson
The Unified Modeling Language User Guide
Addison-Wesley, 1999.
- E. Yourdon
Yourdon Systems Method
Yourdon Press, 1993.

Model dels Estats en UML

- Introducció
- Ús dels diagrames d'estat
- Exemples
- Accions i condicions d'una transició
- Estats imbricats
- Bibliografia

Model d'Anàlisi (especificació)

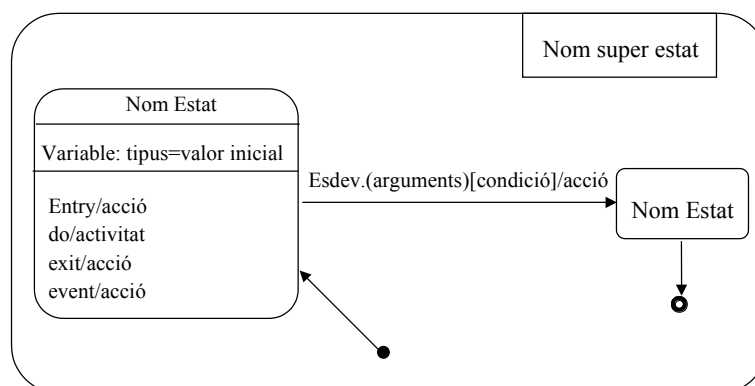


Model dels Estats

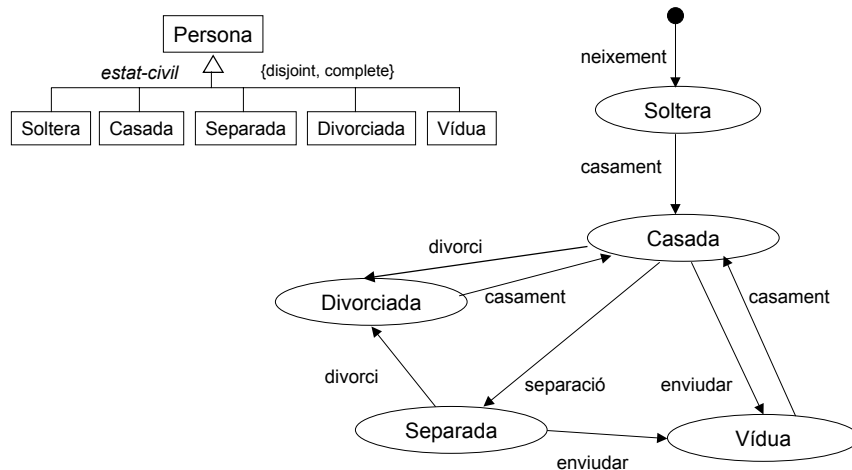
- **Objectius:**
 - crear diagrames d'estat per objectes i casos d'ús
- **Esdeveniments, estats i transicions:**
 - Esdeveniment:
 - tot allò que requereix una resposta del sistema software
 - Estat:
 - condició d'un objecte o d'un cas d'ús en un moment del temps
 - Transició:
 - canvi d'estat com a conseqüència d'un esdeveniment

Diagrama d'estats

Un diagrama d'estats mostra la seqüència d'estats que passa un objecte (o una interacció) durant la seva vida en resposta als estímuls rebuts, juntament amb les seves respostes.



Canvi d'estat civil d'una persona



Ús dels diagrames d'estat

- Un diagrama d'estats es pot especificar per a una:
 - **Classe d'objectes:**
 - per descriure perquè els objectes canvien de subclasse
 - les subclasses d'un diagrama d'estats no tenen perquè aparèixer explícitament a l'esquema conceptual
 - per descriure classes d'objectes amb important "comportament dinàmic"
 - **Cas d'ús:**
 - per descriure la seqüència legal en la que els esdeveniments es poden produir al món real

p.ex. en una compra de producte no es pot fer el pagament fins que no s'hagi tancat la venda.

Diagrama d'estats del cas d'us "comprar productes"

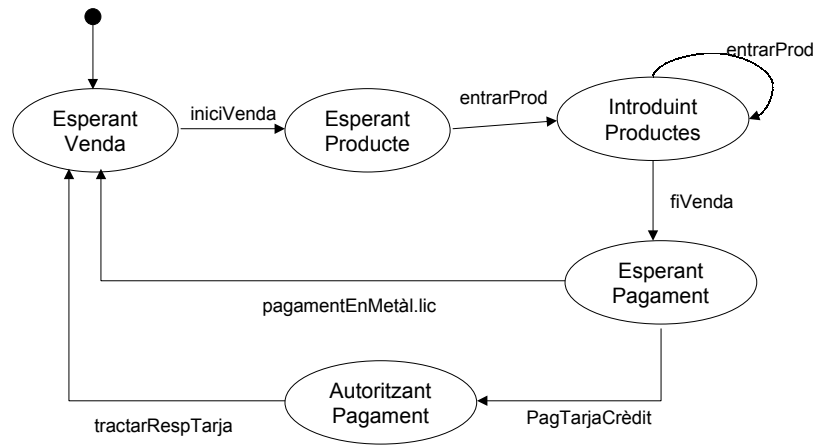
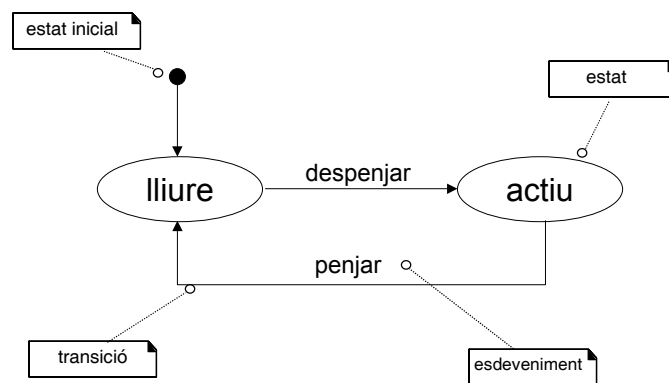
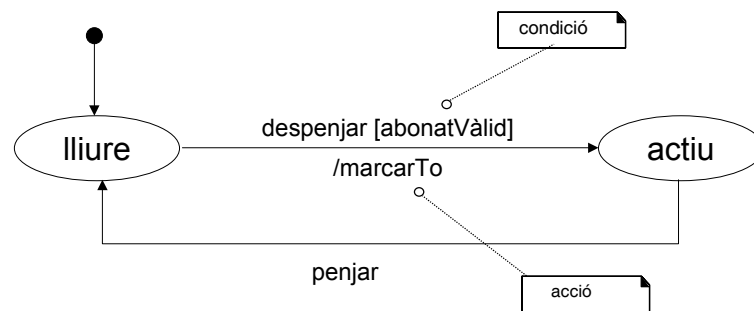


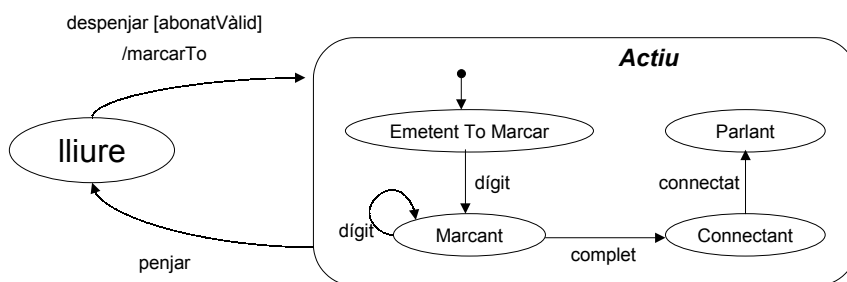
Diagrama d'estats d'un telèfon



Accions i condicions d'una transició



Estats imbricats



Bibliografia

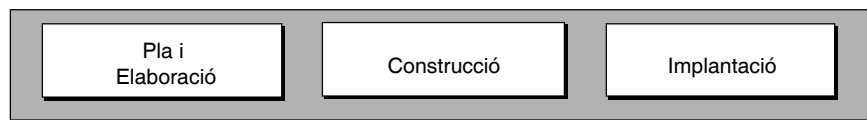
- C. Larman
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (second edition)
Prentice-Hall, 2002 (Cap. 29).
- G. Booch, J. Rumbaugh, I. Jacobson
The Unified Modeling Language User Guide
Addison-Wesley, 1999
- B. Powell Douglass
Real-Time UML.
Addison-Wesley, 1998.

El Procés Unificat de Desenvolupament de Software

- Etapes del procés iteratiu de desenvolupament del software
- Cicles de desenvolupament
- Exemple: compra de productes
- Bibliografia

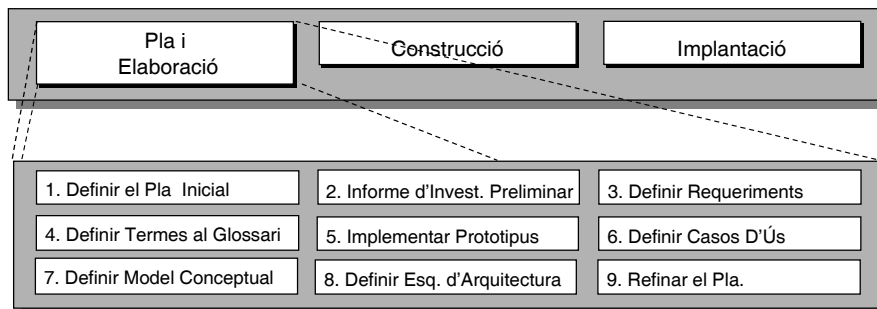
Procés de desenvolupament del software - Macro Nivell

1. **Planificar i Elaborar** - Planificar, definir requeriments, construir prototipus, ...
2. **Construir** - Desenvolupament del sistema (especificació, disseny, etc.)
3. **Implantar** - Implantar el sistema pel seu ús.



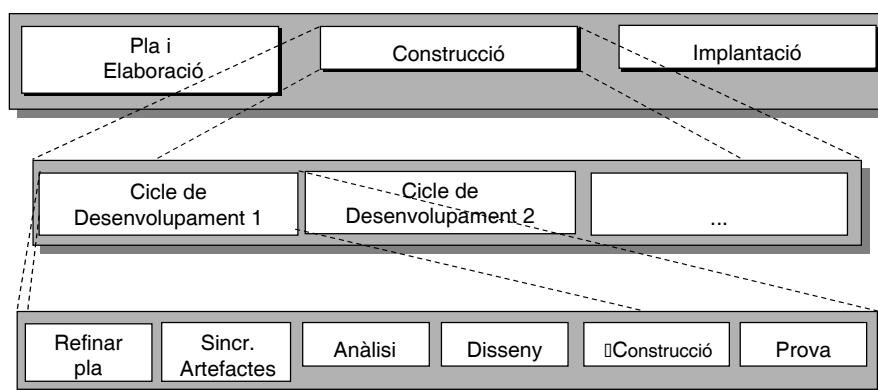
Etapa de planificació i elaboració

Inclou la concepció inicial del projecte, detecció de problemes, determinació d'objectius, investigació d'alternatives de canvi, planificació i especificació dels requeriments.



- **Pla:** calendari, pressupost, etc.
- **Informe d'investigació preliminar:** motivació, alternatives, necessitats de negoci.
- **Especificació de requeriments:** descripció declarativa dels requeriments
- **Glossari:** diccionari de termes (conceptes, noms) i qualsevol informació associada

Etapa de construcció

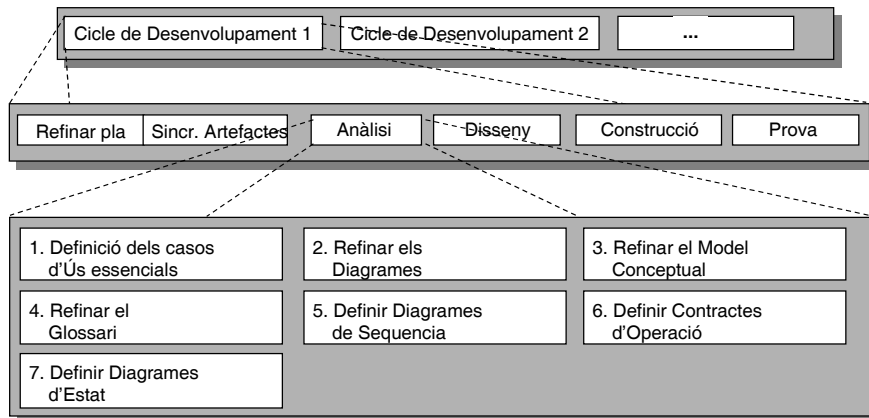


Avantatges:

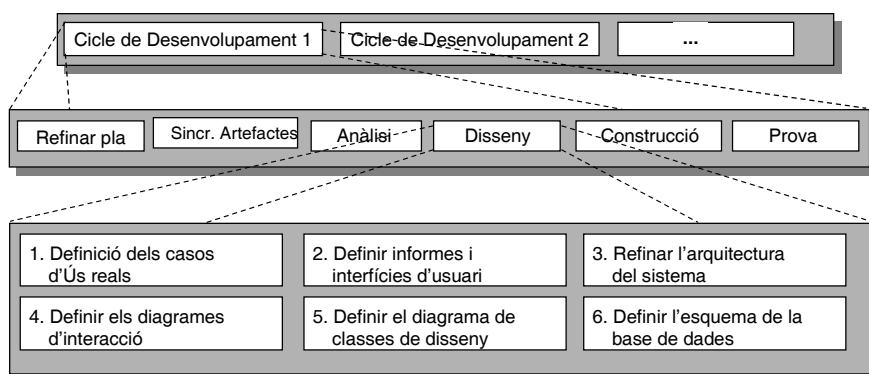
- La complexitat mai no sobrepassa al dissenyador
- Primeres impressions molt ràpidament, perquè la implementació d'una petita part del sistema es fa molt ràpidament.

Etapa de construcció: cicles de desenvolupament (I)

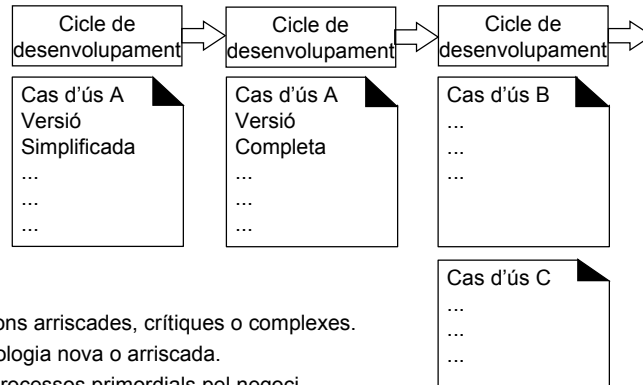
L'etapa de construcció inclou diversos cicles de desenvolupament en els quals el sistema es va estenent de forma progressiva.



Etapa de construcció: cicles de desenvolupament (II)



Ordenar i prioritzar casos d'ús

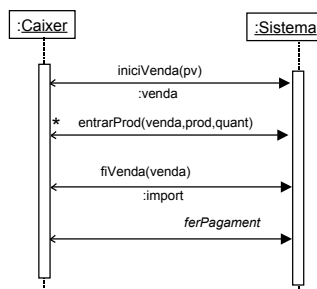


Com prioritzar?

- Inclouen funcions arriscades, crítiques o complexes.
- Involucra tecnologia nova o arriscada.
- Representen processos primordials pel negoci.
- Impacte significatiu en el disseny (afegeix moltes classes al domini o requereix molts serveis).
- Permet obtenir informació significativa respecte al disseny amb poc esforç.

Exemple: compra de productes

- En un primer cicle de desenvolupament pot no interessar-nos distingir entre les diverses formes possibles de pagament

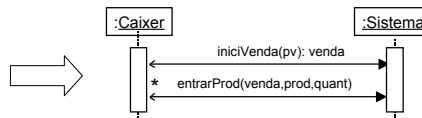


- Interessa desenvolupar el subsistema necessari per poder efectuar una compra
- El contracte de **pagament** hauria de ser prou genèric per no entrar en detalls de la seva forma de pagament

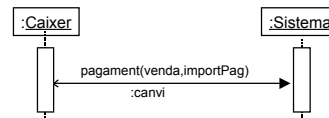
Compra de productes - 2on cicle

- Tenim tants diagrames de seqüència com formes de pagament possibles
- Tots aquests diagrames parteixen del diagrama que s'ha fet al primer cicle de desenvolupament.

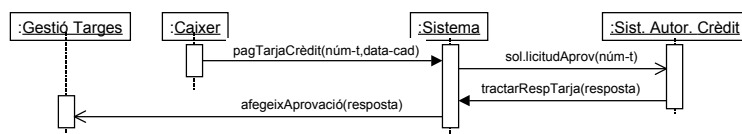
Interacció comuna a tot tipus de pagament



Interacció específica al pagament en metàl·lic



Interacció específica del pagament amb tarja



Nom: pagTarjaCrèdit (num-tarja, data-cad)

Responsabilitats: pagar amb la tarja de crèdit

Excepcions:

Preconditions:

Postconditions:

- creació d'un nou PagamentAmbTarja pmt
- nova associació entre pmt i la venda actual
- ...

Sortida:

- s'envia una sol.licitud d'aprovació al servei d'autorització de crèdit

Nom: tractarRespTarja (resposta)

Responsabilitats:

respondre a la resposta d'aprovació rebuda

Excepcions:

Preconditions:

Postconditions:

- ...

Sortida:

- s'envia una aprovació al sistema de Gestió de Targes perquè l'enregistri

Bibliografia

- C. Larman
Applying UML and Patterns.
An Introduction to Object-Oriented Analysis and Design.
Prentice Hall, 1998. (Cap. 13, 32)
- I.Jacobson, G.Booch, J.Rumbaugh
The Unified Software Development Process.
Addison-Wesley, 1999.