UNIVERSITAT POLITÈCNICA DE CATALUNYA

DEPARTAMENT DE LLENGUATGES I SISTEMES INFORMÀTICS

CARME QUER i BOSOR

DETERMINACIÓ D'ESQUEMES D'INTERACCIÓ EN MODELS CONCEPTUALS ORIENTATS A OBJECTES AMB INTERACCIÓ IMPLÍCITA

TESI DOCTORAL

DIRIGIDA PEL DR. ANTONI OLIVÉ i RAMON

BARCELONA

1998

Memòria presentada per Carme Quer i Bosor
per tal d'aconseguir el grau de Doctor en Informàtica
per la Universitat Politècnica de Catalunya

Agraïments

Vull donar en primer lloc les gràcies al Dr. Antoni Olivé i Ramon per haver dirigit aquesta tesi. Els seus consells, dedicació i rigor científic han estat fonamentals en la realització d'aquest treball.

Així mateix, vull agrair als membres del tribunal per haver acceptat de ser-ho, la qual cosa ha suposat un gran honor per mi.

També vull donar les gràcies als meus companys de la secció de Sistemes d'Informació pels ànims que m'han donat en tot moment. En particular als membres del antic projecte Odissea, que és el marc en què vaig iniciar aquest treball: Dolors Costal, Carme Martin, Enric Mayol, Joan Antoni Pastor, Maria Ribera Sancho, Jaume Sistac, Ernest Teniente i Toni Urpí.

Finalment, vull agrair a la meva família, amics i molt especialment al meu marit Josep la seva estimació, paciència i comprensió durant la realització d'aquesta tesi.

Aquest treball ha rebut un ajut del programa CICYT, projecte TIC 95-0735.

Índex

1. Introducció

L'enginyeria de requeriments és una de les primeres etapes del procés de desenvolupament d'un sistema d'informació. En aquesta etapa, després de la determinació de requeriments mitjançant diferents tècniques, com ara l'anàlisi del domini i les entrevistes, es fa l'especificació d'aquests requeriments, que en el cas dels requeriments funcionals consisteix en la construcció d'un model conceptual del domini del sistema.

En tractar-se d'una de les primeres etapes del procés, els errors que es puguin introduir durant aquesta etapa són molt més cars de resoldre que els que es puguin introduir en etapes posteriors [Dav90]. Per aquest motiu és primordial que abans de passar a l'etapa següent en el procés de desenvolupament, que és la de disseny del sistema d'informació, es faci la validació i verificació del model conceptual.

Les primeres influències de l'orientació a objectes en el camp de la modelització conceptual es van produir cap al final dels anys vuitanta. Abans, aquestes influències ja havien tingut el seu efecte en altres camps relacionats amb el desenvolupament de sistemes d'informació, com ara els llenguatges de programació, el disseny de software o les bases de dades. La majoria dels llenguatges de modelització conceptual resultat d'aquestes influències tenen una gran similitud amb els llenguatges de disseny i de programació, cosa que es nota sobretot en la manera de definir les interaccions entre objectes.

En una gran part dels llenguatges de modelització conceptual i en tots els llenguatges de programació i disseny, les interaccions entre objectes es defineixen explícitament. Malgrat això, cada vegada més hi ha veus crítiques [CoD94, PaW97, QuO93], que diuen que en l'especificació dels requeriments d'un sistema no s'hauria d'haver de decidir com interaccionen els objectes, i que aquesta decisió s'hauria de deixar per a l'etapa de disseny del sistema, en la qual, tenint en compte altres factors, com ara la distribució dels objectes en una xarxa d'ordinadors, hi hagi més elements de judici per decidir quines interaccions interessa entre els diversos objectes. Com a resultat d'aquestes crítiques han aparegut alguns llenguatges de modelització conceptual amb interaccions implícites entre objectes [CoD94, CSO+97].

Els models conceptuals orientats a objectes amb interacció implícita, malgrat ser, potser, més adequats per a l'especificació dels requeriments funcionals d'un sistema que els d'interacció explícita, són més difícils de validar. El motiu principal és que resulta difícil veure quin pot ser l'efecte d'un o més esdeveniments que es comuniquen al sistema en un moment determinat, i, per tant, és difícil comprovar si el model correspon al que realment vol el dissenyador. En aquesta tesi donem un mètode per determinar Esquemes d'Interacció en models conceptuals orientats a objectes amb interacció implícita. La determinació d'Esquemes d'Interacció ens ajudarà a la validació d'aquest tipus de models, ja que ens permetrà conèixer l'efecte que canvis en l'estat d'un objecte poden causar en altres objectes.

Aquesta introducció està estructurada en cinc seccions. La primera caracteritza els models conceptuals orientats a objectes de sistemes d'informació. També classifica els llenguatges de modelització conceptual orientats a objectes, en llenguatges amb interacció explícita i llenguatges amb interacció implícita. A la segona secció es presenta la validació com un dels mitjans existents per aconseguir models conceptuals de qualitat, s'enumeren els diferents enfocaments d'ajuda a la validació d'un model conceptual, i, per a cadascun, les tècniques conegudes per fer-ho. A la secció tercera s'exposa quin és l'objectiu d'aquesta tesi, que s'emmarca en la validació de models conceptuals orientats a objectes amb interacció implícita. La secció quarta descriu els antecedents quant a treballs relacionats amb la validació d'aquests models i quant a tècniques proposades per fer-ho. Finalment, a la cinquena i darrera secció es presenta l'estructuració general de la tesi en capítols.

1.1 Els models conceptuals orientats a objectes de sistemes d'informació

Un model conceptual d'un sistema d'informació descriu el coneixement que el sistema ha de mantenir sobre l'estat d'un domini donat, i també sobre la possible

evolució d'aquest estat a causa de l'ocurrència d'esdeveniments [Gri82]. Aquests dos aspectes, estàtic i dinàmic, defineixen els dos submodels en que es pot considerar dividit un model conceptual, és a dir, el model estructural i el model del comportament.

Si el llenguatge de modelització usat és orientat a objectes, llavors el resultat de la modelització és un model orientat a objectes (MOO). Les primeres influències de l'orientació a objectes en el camp de la modelització conceptual es van produir cap al final dels anys vuitanta. L'orientació a objectes no ha provocat canvis importants en la modelització de l'aspecte estàtic dels sistemes. Ara bé, els canvis en la modelització de l'aspecte dinàmic han estat considerables i importants. La diferència principal que hi trobem és que els llenguatges de modelització orientats a objectes reparteixen el model del comportament entre les classes d'objectes en què s'estructura el sistema.

En qualsevol part de la realitat que sigui domini d'un sistema d'informació, en un instant determinat, hi trobem sempre uns mateixos elements. Trobem entitats amb unes certes propietats i trobem relacions entre aquestes entitats. En els MOO les entitats són els objectes, que en el model es troben agrupats en classes d'objectes, les propietats són els atributs que tenen els objectes d'una classe i les relacions són les associacions entre els objectes de les diferents classes. Així, en el model estructural es donen dues vistes complementaries d'un sistema, una primera vista que descriu l'aspecte estàtic dins cada objecte i una altra que mostra aquest mateix aspecte entre objectes.

D'altra banda, si en comptes de mirar aquesta part de la realitat en un instant determinat, l'observem en el temps, trobem que, entre altres canvis, apareixen noves entitats, d'altres deixen d'existir, les propietats de les entitats canvien de valor i apareixen i deixen d'existir relacions entre altres entitats. Aquests canvis en l'estat del sistema vénen provocats directament o indirecta per l'ocurrència d'esdeveniments que afecten aquestes entitats. A diferència del cas de l'aspecte estàtic, en què hem pogut fer una generalització de com el modelen els MOO, en l'aspecte dinàmic no podem fer el mateix, ja que hi ha gairebé tantes propostes com llenguatges. Aquestes propostes no difereixen només en una qüestió de notació, sinó també en termes dels elements amb els quals representen aquest aspecte, ja siguin regles, transaccions o altres, i en l'estructuració d'aquests elements en el model, és a dir, on apareixen aquests elements i com estan agrupats.

En el cas del model del comportament, també trobem les dues vistes d'un sistema de les quals hem parlat abans. La individual de cada objecte, que indica com canvia el seu estat en el temps, i la vista de l'aspecte dinàmic entre objectes, que indica com els canvis en l'estat d'un objecte poden afectar altres objectes, el que nosaltres anomenarem les

interaccions entre objectes. El terme *interacció* és l'adequat tenint en compte la seva definició en el camp de la física: "influència que les diferents parts constituents del món material exerceixen entre elles", entenent *parts* com a *objectes* i *món material* com a *sistema*.

Independentment dels elements que s'utilitzin per modelar l'aspecte dinàmic i independentment de l'estructuració d'aquests elements en el model, podem classificar els llenguatges de modelització orientats a objectes en dos grans grups: el grup dels que modelen les interaccions entre objectes de manera explícita i el grup dels que les modelen de manera implícita. A continuació trobem tres subseccions. En les dues primeres veurem què volem dir en referir-nos a interacció explícita i implícita i els llenguatges que es poden classificar en una categoria o l'altra. En la tercera inclourem algunes referències bibliogràfiques de crítiques dels llenguatges amb interacció explícita en favor dels altres.

1.1.1 Llenguatges de modelització orientats a objectes amb interacció explícita

La característica que diferencia els models en un d'aquests llenguatges és que en aquests cada esdeveniment és enviat a un objecte particular, que pot veure's afectat per aquest esdeveniment. Alhora, quan un objecte detecta una circumstància en el seu estat que pot afectar l'estat d'altres objectes, genera un o més esdeveniments i aquests esdeveniments s'envien també als objectes a què poden afectar.

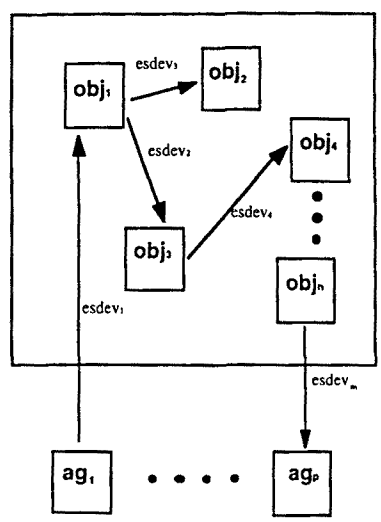


Figura 1.1.1 Model amb interacció directa entre objectes

Observant els llenguatges amb interacció explícita, trobem que es poden classificar segons el model d'interacció que utilitzen en els seus models. Concretament, es poden diferenciar dos models d'interacció: aquell en què són els objectes els que dirigeixen la interacció i aquell en què hi ha un component distribuïdor d'esdeveniments l'especialitzat a fer-ho.

En un model amb interacció directa entre objectes no hi ha cap component que dirigeixi les interaccions. Els esdeveniments en una transacció es dirigeixen cap als objectes que poden veure's afectats. En el cas que les operacions invocades per aquests esdeveniments tinguin esdeveniments efecte, i que aquests esdeveniments efecte puguin produir canvis en altres objectes, seran les mateixes operacions les encarregades de comunicar-los als objectes a què poden afectar. Un esquema d'aquest model d'interacció, el trobem a la figura 1.1.1.

A l'esquema, hi trobem tres elements: els objectes, els esdeveniments i els agents, $ag_1...ag_p$, que representen els usuaris que interaccionen amb el sistema. Els esdeveniments són comunicats al sistema pels agents que els dirigeixen directament als objectes a què poden afectar. Aleshores, si els objectes a què afecten generen algun altre esdeveniment, també l'envien directament als objectes per als quals sigui rellevant. Així, la responsabilitat de les interaccions queda repartida entre els diversos objectes.

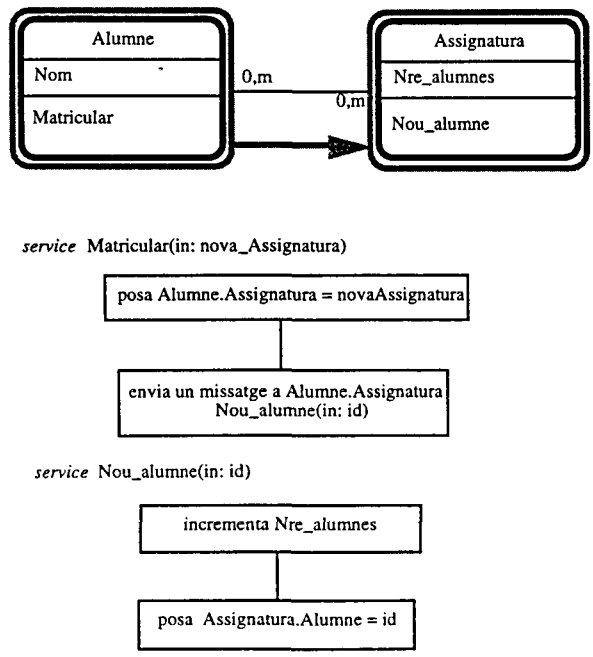


Figura 1.1.2 Model conceptual en OOA

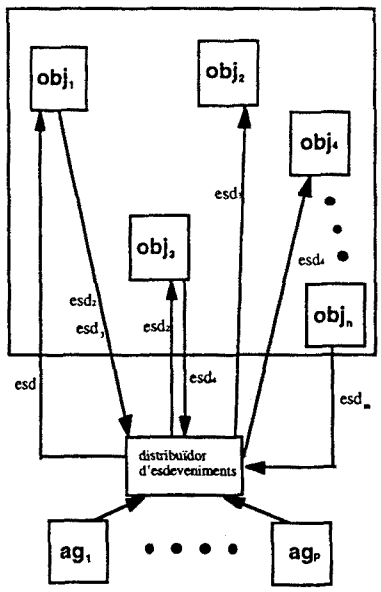


Figura 1.1.3 Model amb interacció a través d'un distribuïdor d'esdeveniments

Mètodes d'especificació de requeriments que proposen l'utilització d'un llenguatge amb interacció explícita i model d'interacció directa entre objectes, ho són: l'OOA de Coad i Yourdon [CoY91], l'OOA&D de Martin i Odell [MaO92], l'OMT [RBP+91], l'OOA de Shlaer i Mellor [ShM92].

Exemple 1.1.1

Tot seguit veurem un exemple senzill de model conceptual fet en el llenguatge proposat en el mètode OOA de Coad i Yourdon, figura 1.1.2. Es tracta de modelar un domini on trobem alumnes i assignatures en les quals els alumnes es poden matricular. Interessa conèixer quants alumnes hi ha matriculats en cada assignatura.

Com podem observar, tant en l'especificació dels serveis, *service*, com en el model d'objectes es veu explícitament quines seran les interaccions entre els objectes. En els serveis, d'una manera més concreta, mitjançant la invocació de l'enviament d'un missatge i en el model d'objectes, d'una manera més general, amb la fletxa o connexió d'invocació que assenyala que hi haurà enviament de missatges entre els objectes de les dues classes. ♦

En un model amb interacció a través d'un distribuïdor d'esdeveniments, aquest distribuïdor és el que dirigeix les interaccions entre objectes. En cap cas hi ha interaccions directes entre dos objectes del model. Quan en un moment determinat

ocorren un o més esdeveniments, aquest grup d'esdeveniments es comunica al distribuïdor, i és a través d'ell per on passa qualsevol esdeveniment generat per un objecte abans que sigui enviat als objectes que pot afectar. Un esquema d'aquest model d'interacció és el de la figura 1.1.3.

En aquest cas l'esquema té un element més, que és el distribuïdor d'esdeveniments o controlador d'interaccions. Totes les interaccions entre objectes passen a través seu. Aquí els esdeveniments que es comuniquen al sistema es dirigeixen al distribuïdor d'esdeveniments, que és qui sap quins són els objectes a què poden afectar. Aleshores, el distribuïdor els envia a aquests objectes, i en el cas que aquests generin algun esdeveniment, l'envien al distribuïdor, que tornarà a començar el procés. Així, la responsabilitat de l'enviament de les interaccions és tota del distribuïdor.

Aquest tipus de llenguatges amb interacció explícita i model d'interacció a través d'un distribuïdor d'esdeveniments són més habituals en l'àmbit del disseny de software que en la modelització conceptual, en què un dels pocs llenguatges que es poden considerar d'aquest tipus és el TROLL [JSH+96].

```
template Alumne
data types string, |Assignatural;
attributes Nom: string;
           Assignatures: set(|Assignatural);
events
        birth alta_alumne(in Nom: string);
        matricular(in Curs: |Assignatural);
...
endtemplate Alumne;

template Assignatura
data types nat, |Alumnel;
attributes Nre_alumnes: nat;
           matriculats: set(|Alumnel);
events
        birth alta_assignatura;
        nou_alumne;
...
endtemplate Assignatura;

relationship Matricula between Alumne, Assignatura;
data types |Alumnel, |Assignatural;
interaction
variables a: |Alumnel; c: |Assignatural;
        Alumne(a).matricular(c) >> Assignatura(c).nou_alumne;
endrelationship Matricula;
```

Figura 1.1.4 Model conceptual en TROLL

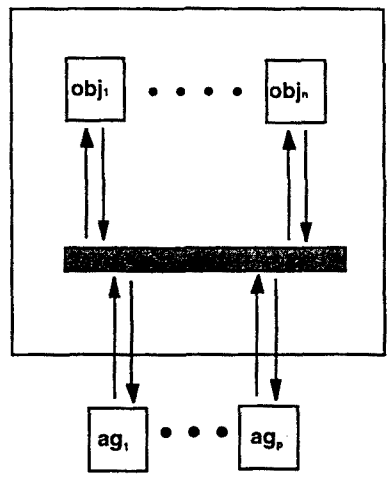


Figura 1.1.5 Model amb interacció implícita

Exemple 1.1.2

A la figura 1.1.4 trobem un exemple de model conceptual fet en TROLL. Es tracta del mateix cas de l'exemple 1.1.1. Com podem observar, en TROLL les interaccions entre objectes no es reparteixen en les especificacions de cada objecte sinó que s'aïllen en el que s'anomenen *relationship*(s). D'altra banda, encara que a la figura no es veu, també encapsulen les classes d'objectes respecte dels seus agents o usuaris mitjançant d'interfícies d'objectes, *interface*(s). Aquests elements, *relationship*(s) i *interface*(s), especificats d'una manera independent a les classes d'objectes, constitueixen el distribuïdor d'esdeveniments. ♦

1.1.2 Llenguatges de modelització orientats a objectes amb interacció implícita

En els models en un d'aquests llenguatges, tots els esdeveniments que es comuniquen al sistema o que són generats pel mateix sistema estan disponibles per a tots els objectes, i són els mateixos objectes els que han de determinar si hi estan o no interessats. En cas d'estar-hi interessats, aquests objectes han de modificar el seu estat o actuar de la manera especificada, ja sigui anunciant la violació d'una restricció d'integritat o bé generant nous esdeveniments que, alhora, podran afectar altres objectes. Un esquema d'aquests models és el de la figura 1.1.5.

A l'esquema, els agents comuniquen els esdeveniments al sistema. Ara bé, no els comuniquen a objectes concrets, sinó que els fan públics en una mena de espai de "publicació" d'esdeveniments ocorreguts. D'altra banda, els objectes estan pendents dels esdeveniments que es publiquen en aquest espai, i si n'hi ha algun que pot ser

rellevant per al seu estat, el tenen en compte. És dins dels diferents objectes que es troba especificat quins esdeveniments poden afectar el seu estat i com poden fer-ho. Generalment, aquesta especificació es fa amb algun tipus de regles.

Aquests models, a diferència dels que presenten interacció explícita, no inclouen el concepte d'operació, i consideren que la definició de les operacions i l'explicitació de les interaccions és una tasca de l'etapa de disseny. Els defensors dels llenguatges amb interacció implícita opinen que definir les operacions i explicitar les interaccions en l'etapa d'anàlisi és avançar decisions de disseny a aquesta etapa.

Els efectes d'un esdeveniment sobre un objecte poden ser de més d'un tipus. En el cas que un dels efectes sigui la generació de nous esdeveniments, l'objecte comunica aquests nous esdeveniments al sistema posant-los a l'espai de publicació d'esdeveniments. En el cas que aquests esdeveniments puguin ser rellevants per als agents o usuaris del sistema, aquests usuaris els obtenen d'aquest espai.

Llenguatges amb interacció implícita, ho són: el Syntropy [CoD94], els llenguatges per a MCDO [QuO93,QuO94] i el ROSES [BCC+96,CBC+96].

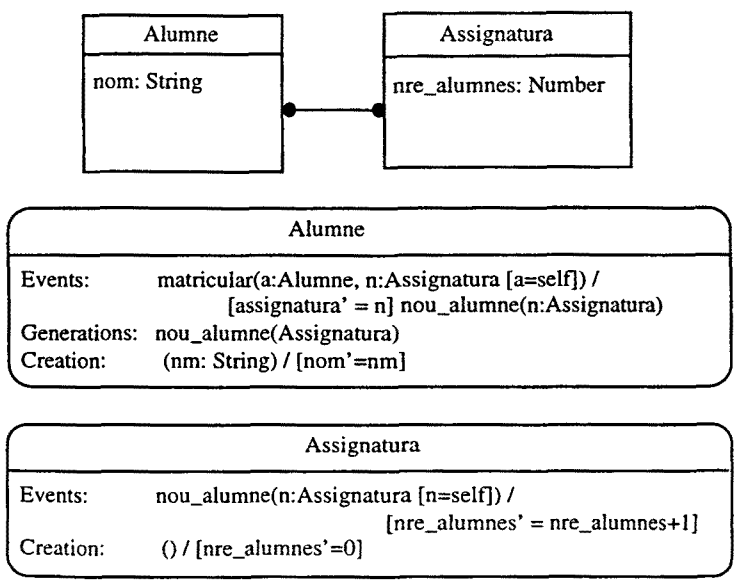


Figura 1.1.6 Model conceptual en Syntropy

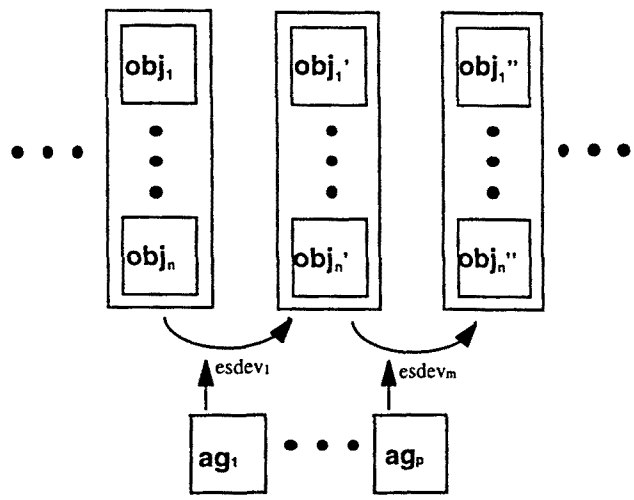


Figura 1.1.7 Model amb interacció implícita en els esdeveniments

Exemple 1.1.3

A la figura 1.1.6 trobem un exemple de model conceptual fet en Syntropy. Es tracta del mateix cas dels exemples anteriors. Com podem observar, als diagrames d'estats de cada classe d'objectes, hi ha una llista dels esdeveniments rellevants per als objectes de la classe. Quan es publiqui un d'aquests esdeveniments, els objectes de la classe comprovaran si pot tenir algun efecte sobre el seu estat o no. En el cas de la classe *Alumne*, hi ha un únic esdeveniment rellevant *matricular*, concretament és rellevant per l'alumne *a*, clàusula *[a=self]*, i s'indica quin serà l'efecte d'aquest esdeveniment sobre aquest alumne. Un dels efectes serà la generació d'un esdeveniment de la classe *nou_alumne*. Aquest esdeveniment no va dirigit a cap objecte de cap classe i podria afectar-ne qualsevol que així ho tingués especificat a la seva llista d'esdeveniments. Per tant, la generació farà que aquest esdeveniment es publiqui. Un cop publicat, els objectes que així ho tinguin especificat el detectaran i actuaran com calgui segons l'especificació. ♦

Llenguatges amb interacció implícita en els esdeveniments

Observant els mètodes d'especificació de requeriments que proposen l'utilització d'un llenguatge orientat a objectes, ens adonem que hi ha llenguatges que, fins ara, no hem classificat en cap grup.

En els models en un d'aquests llenguatges es defineix per a cada classe d'esdeveniments externs, un grup de precondicions i un grup de postcondicions. Les precondicions indiquen quin se suposa que és l'estat del sistema abans de l'ocurrència d'un esdeveniment de la classe i les postcondicions indiquen quin és l'estat del sistema

després de l'ocurrència d'un esdeveniment de la classe. Un esquema d'un model amb interacció implícita en els esdeveniments és el de la figura 1.1.7.

A l'esquema, els agents comuniquen els esdeveniments al sistema, i això fa que el sistema passi a un nou estat on es compleixen les postcondicions especificades per l'esdeveniment.

En aquests models no s'explicita quines interaccions hi ha d'haver entre els objectes del sistema per aplicar uns certs canvis en el seu estat, ni l'ordre en què aplicar aquests canvis d'estat. Aquestes decisions es deixen per l'etapa de disseny del sistema.

Mètodes d'especificació de requeriments que utilitzen aquest tipus de llenguatges, ho són: el FUSION [CAB+94], el OOSE [Jac92] i el proposat per Larman [Lar98].

En aquesta tesi, quan parlem de llenguatges amb interacció implícita només farem referència al primer grup de llenguatges, és a dir, no inclourem els llenguatges amb interacció implícita en els esdeveniments. La raó és que, normalment, aquests llenguatges no són prou formals per poder deduir a partir d'un model les interaccions implícites.

1.1.3 Crítiques de la interacció explícita en favor de la implícita

No és objectiu d'aquesta tesi comparar aquests dos enfocaments, i tampoc coneixem cap treball que n'hagi fet una anàlisi detallada. El que farem en aquesta subsecció és donar un parell de referències bibliogràfiques en què es mostra una actitud crítica respecte de la interacció explícita. També farem una observació sobre la similitud que es pot observar entre els llenguatges operacionals i els llenguatges amb interacció explícita i entre els llenguatges deductius i els llenguatges amb interacció implícita. Tenint en compte aquesta similitud, les comparacions existents entre els llenguatges operacionals i deductius poden donar idees per comparar els altres dos tipus de llenguatges i per saber quin dels dos és més adequat per a la modelització conceptual d'un sistema d'informació.

Comencem amb les referències. És important dir que no n'hem trobat cap que defensi la interacció explícita contra la implícita. A parer nostre, la raó és que els mètodes amb un llenguatge amb interacció explícita no consideren la interacció implícita com una alternativa.

El primer text és del llibre de Cook i Daniels en què es proposa la modelització en Syntropy [CoD94]: "*Sigui quina sigui la naturalesa de la realitat, no trobem massa útil descriure-la en termes de pas de missatges o d'operacions en objectes individuals. Els esdeveniments reals poden ser detectats per molts observadors alhora; no són enviats punt a punt. El pas de missatges i la invocació d'operacions són constructors adequats per descriure l'execució de software, però normalment no per descriure què passa al món real, perquè tendeixen a sobreespecificar l'ordre de les conseqüències dels esdeveniments... Trobem molt útil poder descriure el software en termes abstractes, com en el model d'especificació, sense necessitat de considerar qüestions d'implementació com la concurrència i l'ordre entre missatges. La major part de la història de la informàtica ha estat la recerca de l'abstracció, és a dir, maneres de dir el que es necessita, mentre que s'omet el que és superflu. L'ordre entre missatges és superflu per especificar el comportament observable del software, i només esdevé rellevant quan necessitem descriure com la implementació del software es mapeja en processadors.*"

El segon text és de la revista *Communications of the ACM*. En són autors Parsons i Wand [PaW97]: "*Els models de representació no especifiquen el pas de missatges com a mecanisme de comunicació. Conseqüentment, suggerim anar amb compte en modelar un domini en termes de pas de missatges entre objectes. Per exemple, considerarem un sistema de processament de comandes. Seria un error descriure el fer una comanda com un client envia un missatge a l'ítem inventari. En realitat, la comanda es fa a una persona de vendes. Això és el que el model ha de mostrar.*"

Vegem ara les similituds dels llenguatges operacionals i deductius amb els que aquí ens interessen. Simplificant-ho, podem dir que en els llenguatges de modelització conceptual operacionals s'associa una operació a cada tipus d'esdeveniment extern que és rellevant per al domini del sistema. En una operació s'especifica com canvia l'estat del sistema quan ocorre un dels esdeveniments del tipus. En els llenguatges de modelització conceptual deductius, en canvi, el que es fa és definir quin és l'estat del sistema en un cert instant en base als esdeveniments ocorreguts en el sistema en el mateix instant i/o en instants anteriors.

Tenint en compte aquesta visió dels llenguatges deductius i operacionals, hi ha similitud entre els llenguatges operacionals i els llenguatges orientats a objectes amb interacció explícita i entre els llenguatges deductius i els llenguatges orientats a objectes amb interacció implícita:

- Els llenguatges orientats a objectes amb interacció explícita igual com els operacionals utilitzen el concepte d'operació. La diferència és que l'operació associada a un tipus d'esdeveniment extern en els llenguatges operacionals, en els orientats a objectes està dividida en diverses operacions en diferents objectes, que s'envien missatges entre elles quan ocorre un esdeveniment extern del tipus.

- Els llenguatges orientats a objectes amb interacció implícita igual com els deductius defineixen l'estat del sistema en un cert instant a partir dels esdeveniments ocorreguts en el mateix instant i/o en anteriors. La diferència és que en els primers les definicions dels diferents elements s'agrupen per classes d'objectes.

L'única diferència important entre ells està en com modelen l'aspecte estàtic dels sistemes, és a dir, en la utilització, en el cas dels llenguatges amb interacció implícita i explícita, de l'orientació a objectes.

Així, podem dir que els MOO amb interacció explícita són models operacionals en el món de l'orientació a objectes, i que els MOO amb interacció implícita són models deductius en el món de l'orientació a objectes. Tenint en compte aquest fet, ens adonem que una comparació entre llenguatges operacionals i deductius seria interessant per ajudar a fer una comparació entre llenguatges amb interacció explícita i llenguatges amb interacció implícita.

A [Oli86] hi ha una comparació entre l'enfocament de modelització deductiu i l'operacional. D'aquesta comparació es pot deduir que els llenguatges de modelització conceptual deductius tenen clars avantatges sobre els operacionals. Aquest és un motiu més per pensar que la interacció implícita és una manera millor d'especificar els requeriments que no la interacció explícita.

1.2 Validació de models conceptuals

La validació, juntament amb la verificació, són dues activitats bàsiques en el procés de desenvolupament de sistemes d'informació. Ambdues ajuden a assegurar l'obtenció d'un software de qualitat, és a dir un software lliure d'errors que es correspon totalment amb l'establert en els requeriments funcionals i no funcionals del sistema d'informació.

La verificació d'un model conceptual comporta la comprovació de propietats formals del model, com són la correctesa sintàctica, la consistència i la completesa, i per aquesta raó es tracta d'una activitat que pot ser automatitzada. En canvi, la validació és una activitat difícil d'automatitzar, ja que consisteix a comprovar si el model representa de manera correcta una part de la realitat i els requeriments funcionals dels usuaris, i, a més, habitualment hi intervenen els dissenyadors i/o els futurs usuaris del sistema

[DaJ97]. De fet, l'única automatització que es pot fer d'aquesta activitat és la d'ajudar les persones encarregades de fer-la a entendre millor allò especificat en el model, de manera que tinguin més elements de judici per afirmar si és cert, o no, que el model conceptual expressa els requeriments rellevants per a l'usuari i les seves necessitats.

A continuació veurem tres enfocaments per ajudar en la validació d'un model conceptual. Es tracta de l'enfocament de reducció de la complexitat, el de fer canvis en la presentació i el de facilitació del raonament. Aquesta classificació es basa en [Lin93, Gul93, DaJ97, Bub88].

Reducció de la complexitat

En observar un model conceptual de mida mitjana o gran i d'una certa complexitat, pot passar que una gran quantitat de detalls inclosos en aquest model no ens deixin acabar d'entendre allò especificat. Reduint la complexitat es pot afavorir aquesta comprensió [Sel94].

Les tècniques de reducció de la complexitat són tècniques d'estructuració, d'abstracció i de composició.

- Les d'estructuració, coneixent la semàntica de cadascun dels documents d'un model conceptual, combinen els seus elements en elements de més alt nivell, i així els presenten a qui fa la validació.
- Les d'abstracció, l'únic que fan és permetre a qui fa la validació de fer abstracció de parts del model que no són rellevants en un cert moment de la validació.
- Finalment, les de composició intenten simplificar la complexitat introduïda pels dissenyadors, i donar alternatives de modelització de tot o una part del model conceptual.

Canvi en la presentació

La dificultat de comprensió dels models conceptuals pot venir donada pel llenguatge en què estan escrits. La causa d'això és que en molts casos és l'usuari qui en fa la validació, o ha de participar-hi. Un canvi en la presentació del model consisteix en una reescriptura d'una part o tot el model en un llenguatge diferent, conegut pels qui participen en la validació, o, simplement, més senzill d'entendre que l'original.

Les tècniques d'ajuda a la validació que segueixen aquest enfocament són bàsicament de dos tipus, de parafrasejat i de traducció. Les primeres [DFv96,RoP92,Dal92] expliquen en llenguatge natural aquelles parts que interessen del model. Les altres fan una traducció del model d'un llenguatge més complex a un de més senzill.

Facilitació del raonament

Aquest enfocament consisteix a donar la possibilitat a qui fa la validació de raonar sobre si el model conceptual s'adequa als requeriments funcionals. Normalment, això es fa per mitja del plantejament de preguntes i l'observació de les respostes que s'obtenen, que ajuden a entendre el comportament del sistema modelat, i així es pot saber si aquest comportament és el mateix que el que s'espera del sistema un cop desenvolupat.

Les tècniques d'ajuda a la validació que segueixen aquest enfocament són molt diverses, però es poden agrupar en tècniques d'inspecció de models [Gul93], de simulació infològica [Bub86] i de prototipatge semàntic [LTP91].

- En les primeres es dóna la possibilitat de fer preguntes sobre propietats del model conceptual, sobre l'estructura del model o sobre el seu comportament. Les respostes a aquestes preguntes es donen a partir de l'estudi del model conceptual. La classificació d'aquest tipus de tècniques és difícil, ja que segons com, es poden veure com a tècniques de reducció de la complexitat.
- Les de simulació infològica [Cos95, CTU+96] permeten raonar sobre les especificacions mitjançant un estudi de com pot evolucionar l'estat del domini mantingut pel sistema des d'un estat inicial cap a un estat final. Observant aquestes possibles evolucions de l'estat, qui fa la validació pot comprovar si s'adequa al sistema que es vol desenvolupar.
- Finalment, les de prototipatge semàntic [San93, San94, Sis87, Sis92, LiK93, GrK97] permeten la simulació del comportament del domini que es modela. Observant aquest comportament, qui fa la validació pot comprovar si s'adequa al sistema que es vol desenvolupar.

La generació d'explicacions a partir d'un model no pot ser considerada, pròpiament, una tècnica de facilitació del raonament, ja que l'únic que fa és ampliar qualsevol d'aquestes tècniques fent comprensible a l'usuari la informació que n'obté [Gul96,DaJ97]. És a dir, serveix de capa externa a aquestes tècniques. Les explicacions se solen donar amb llenguatge natural, encara que també hi ha propostes de completar aquest llenguatge amb representacions gràfiques.

Un cop enumerades les tècniques en els diferents enfocaments, es veu que totes són tècniques de validació "parcial". És a dir, no són excloents entre si, sinó més aviat complementàries, i segons el llenguatge, el model conceptual i/o el tipus de persona que fa la validació (dissenyador o futur usuari del sistema), pot ser més adequat utilitzar-ne una o una altra, i en pràcticament tots els casos més d'una. Així, per fer la validació

d'un model, el millor és utilitzar entorns de validació o eines CASE que donin la possibilitat d'utilitzar diverses tècniques.

Com veurem a la secció següent, el treball que és l'objecte d'aquesta tesi s'emmarca en el món de la validació de models conceptuals mitjançant la facilitació del raonament.

1.3 Objectius de la tesi

L'objectiu de la tesi és proposar un nou mètode de validació aplicable a MOO amb interacció implícita en un llenguatge suficientment formal.

Com hem vist a la secció 1.1, en els MOO amb interacció implícita s'especifica localment en cada objecte quins esdeveniments poden ser rellevants per a l'estat de l'objecte, i d'altra banda quins esdeveniments pot generar que poden ser rellevants per a altres objectes del sistema. Ara bé, encara que a l'hora d'especificar un sistema pot ser més adequat aquest tipus de models, a l'hora de validar-lo o a l'hora de fer-ne el disseny ens pot interessar conèixer explícitament les interaccions entre els seus objectes.

El nostre mètode permetrà conèixer, donat un MOO amb interacció implícita i donat un tipus de transacció (una o més classes d'esdeveniments externs), quines són les conseqüències que poden tenir un o més esdeveniments externs, de cada classe en el tipus de transacció, que ocorrin en un instant determinat, sobre l'estat dels objectes del sistema, figura 1.3.1. A més, la persona encarregada de la validació podrà saber com els canvis en l'estat dels objectes d'una classe a conseqüència d'aquests esdeveniments poden causar canvis en l'estat d'objectes d'altres classes i si cal mantenir un cert ordre entre aquests canvis. Aquesta informació es donarà en forma d'Esquemes d'Interacció en què apareixeran les interrelacions entre canvis en l'estat d'objectes de diferents classes com a tipus d'interaccions, i en què s'indicarà de manera explícita l'ordre de precedència que s'ha de complir entre les interaccions de diferents tipus.

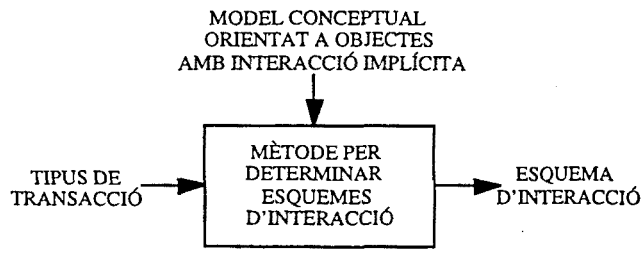


Figura 1.3.1

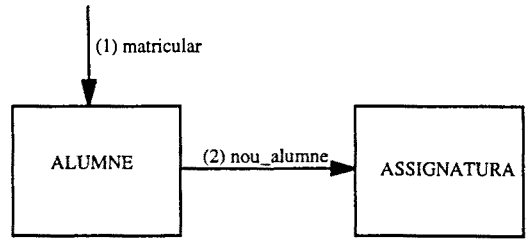


Figura 1.3.2 Resultat de la determinació d'interaccions

Per entendre millor el què fa el mètode, donem l'exemple següent:

Exemple 1.3.1

Suposem el MOO amb interacció implícita en llenguatge Syntropy de la figura 1.1.6. L'Esquema d'Interacció que s'obté mitjançant el nostre mètode per al cas de l'ocurrència d'un o més esdeveniments de la classe *matricula*, figura 1.3.2, indica que:

- El primer que cal fer (1) quan es produeix en un instant de temps un o més esdeveniments externs de la classe *matricular*, és enviar aquests esdeveniments a la classe d'objectes *Alumne*.

- El segon pas, que no es pot donar en paral.lel a l'anterior (2), és enviar tots els esdeveniments generats pels objectes de la classe *Alumne*, de la classe d'esdeveniments *nou_alumne,* a la classe d'objectes *Assignatura.*

Així, com a conseqüència d'un o més esdeveniments externs de la classe *matricular,* hi poden haver dos tipus d'interaccions. La primera entre l'exterior i objectes de la classe *Alumne* i la segona entre objectes de la classe *Alumne* i objectes de la classe *Assignatura*. Un o més esdeveniments de la classe *matricular* poden causar canvis d'estat en objectes de les classes *Alumne* i *Assignatura,* i aquests canvis s'han d'aplicar en un ordre. De la segona interacció deduim que quan a la classe *Alumne* s'afegeix un nou alumne, això afecta l'estat d'un o més objectes de la classe *Assignatura.*♦

Totes les ajudes del nostre mètode es mouran en el món del "potencial". En cap moment es podrà assegurar que es produirà una interacció, o que l'estat d'un objecte es veurà afectat, sinó que "podran" produir-se interaccions d'un cert tipus i l'estat d'un o més objectes d'una classe "podrà" canviar. El que realment passi o no dependrà dels esdeveniments concrets que es produeixin i de l'estat del domini en l'instant en què es produeixin.

En altres paraules, no es tracta, per exemple, de saber quines són les interaccions causades, en un estat del domini determinat, per un esdeveniment de baixa de matrícula

de l'estudiant de nom *Oriol* a l'assignatura *Expressió Oral*, sinó quines poden ser les interaccions que pot causar l'ocurrència d'un o més esdeveniments de la classe *baixa de matrícula*, independentment de l'estat del domini en l'instant que aquests esdeveniments ocorren. Encara que algú pot pensar que això és un defecte del mètode, també pot veure's com una virtut, ja que fent-ho d'aquesta manera es fa abstracció dels esdeveniments concrets que es produeixin i de l'estat del domini.

Des del punt de vista de la validació de models, el mètode formarà part del grup de tècniques d'inspecció de models que faciliten el raonament sobre un model conceptual. Algú podria pensar que aquesta classificació no és correcta i que s'hauria de classificar com a tècnica de prototipatge semàntic. Això no és així; la raó és que no es tracta de raonar sobre un estat concret del sistema i esdeveniments externs concrets, és a dir sobre objectes, esdeveniments i transaccions, sinó sobre classes d'objectes, classes d'esdeveniments i tipus de transaccions.

Igual que les tècniques enumerades a la secció anterior, el nostre mètode no oferirà una validació completa d'un model, sinó que oferirà una manera més de fer la validació d'un model que complementa les existents.

Finalment hem de dir que, a diferència de la majoria dels mètodes de validació existents, el nostre, serà un mètode *general* que podrà ser aplicat a qualsevol model escrit en un llenguatge orientat a objectes amb interacció implícita suficientment format. Això s'aconseguirà dividint l'aplicació del mètode en dos passos, figura 1.3.2:

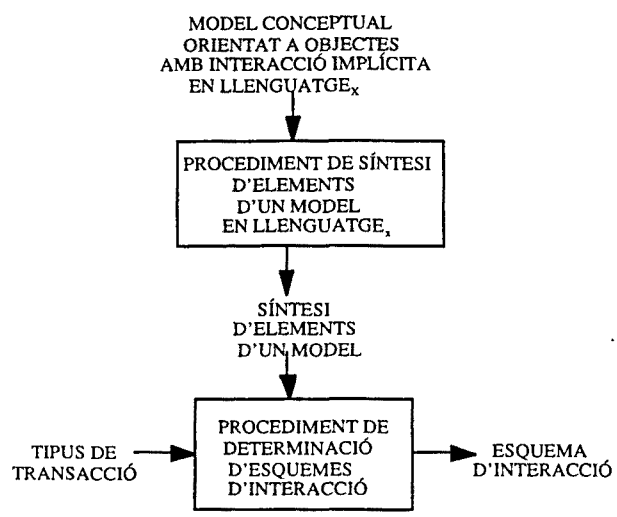


Figura 1.3.2 Mètode per determinar Esquemes d'Interacció

- Si observem els models escrits en els diferents llenguatges del tipus de què tractem, en tots trobem uns mateixos elements a partir dels quals es poden determinar els Esquemes d'Interacció. El primer pas serà aplicar un procediment de síntesi dels elements d'un model rellevants per a la determinació d'Esquemes d'Interacció. Aquest serà un procediment semiformal específic de cada llenguatge.

- Un cop es disposi de la síntesi d'elements d'un model, es podrà aplicar el procediment de determinació d'Esquemes d'Interacció, que per un tipus de transacció donat (una o més classes d'esdeveniments externs) ens donarà un Esquema d'Interacció vàlid. Aquest serà un procediment formal que no dependrà del llenguatge en que està el model.

1.4 Antecedents

Podem distingir diferents tipus d'antecedents: antecedents en validació de MOO, antecedents quant a eines d'inspecció de models conceptuals en general i antecedents en anàlisi estàtica de conjunts de regles.

Cap dels treballs que hem trobat de validació de MOO intenten validar un model determinant les possibles interaccions entre objectes, i tampoc ajuden a qui fa la validació donant informació sobre quines poden ser les conseqüències d'un o més esdeveniments externs d'un cert tipus. Els mètodes de validació que existeixen són bàsicament de parafrasejat [DFv96] i de prototipatge semàntic [LTP91, GrK97].

Observant els treballs fets en el camp de la inspecció de models conceptuals, no hem trobat cap documentació sobre mètodes especialitzats en aquesta classe d'ajuda a la validació. Els treballs existents no poden ser classificats com d'inspecció, sinó de canvi en la presentació, ja que habitualment es limiten a explicar una part del model en un llenguatge diferent d'aquell amb què està escrit, però no en dedueixen res.

Finalment, el nostre mètode es pot considerar relacionat amb els treballs d'anàlisi estàtica de conjunts de regles perquè els models als que s'aplica, tal com ja hem dit, generalment especifiquen el comportament d'un sistema mitjançant regles, i el nostre mètode el que fa és una anàlisi d'aquestes regles per arribar a determinar els Esquemes d'Interacció. Concretament, està relacionat amb els mètodes de càlcul de canvis en bases de dades deductives i els mètodes d'anàlisi estàtica de regles en bases de dades actives.

Un mecanisme de càlcul de canvis eficient és essencial en diferents parts del funcionament de les bases de dades deductives, com la comprovació de restriccions

d'integritat [BMM90], el manteniment de vistes materialitzades [CeW91] i la monitorització de condicions [RCB+89]. Alguns mètodes de càlcul de canvis dedueixen a partir de l'esquema d'una base de dades, un conjunt de regles que defineixen les condicions d'inducció dels diferents canvis possibles a la base de dades [UrO92, CeW91, Küc91]. L'eficiència d'aquests mètodes de càlcul de canvis dependrà de la forma de les regles i també de la tècnica d'avaluació utilitzada per calcular els canvis reals induïts per una certa modificació de l'estat de la base de dades. El nostre mètode i una d'aquestes tècniques d'avaluació de regles són similars, tenint en compte que ambdós actuen considerant els canvis potencials que poden ser induïts per l'ocurrència d'esdeveniments externs d'unes certes classes, en el cas del nostre procediment, i per una certa modificació de la base de dades, en el cas de les tècniques d'avaluació de regles. La diferència és que en el càlcul de canvis, l'èmfasi recau a avaluar el mínim nombre de regles i en el millor ordre per fer el mínim nombre d'accessos a l'estat de la base de dades per fer el càlcul de canvis [GrM94, DeG94, LeL96]. En [Que94] vaig presentar una versió prèvia del mètode proposat en aquesta tesi, adaptat com a tècnica d'avaluació de regles.

D'altra banda, les tècniques d'anàlisi estàtica de regles en bases de dades actives s'utilitzen per establir propietats formals de conjunts de regles que es compleixen independentment de les transaccions i l'estat en què pugui estar la base de dades [WiC96, AHW95, BCP95, LeL98], com ara l'acabament o la confluència. La similitud d'aquestes tècniques amb el nostre mètode és que en els dos casos es tracta d'una anàlisi de conjunts de regles feta en temps de compilació, que en els dos casos es fa abstracció de la transacció concreta o els esdeveniments externs concrets i de l'estat de la base de dades o del domini del sistema d'informació, i que en ambdós casos l'anàlisi de regles es fa estudiant les relacions entre els elements de les diferents regles. Ara bé, els objectius són diferents, i per tant els mètodes per a la comprovació d'una de les propietats i el nostre mètode per determinar Esquemes d'Interacció no tenen res en comú.

1.5 Estructura de la tesi

Aquesta tesi s'estructura en set capítols.

El primer és aquesta introducció en què hem vist què són els MOO amb interacció implícita de sistemes d'informació, els diferents enfocaments existents per a la validació de models conceptuals, l'objectiu de la tesi i els seus antecedents.

El segon i tercer capítols descriuen el mètode. En el segon definim una notació que ens permetrà caracteritzar els elements d'un model del tipus de què tractarem. A partir

de qualsevol d'aquests models, en qualsevol llenguatge, es podran sintetitzar, en aquesta notació, els elements que després necessitarem per a la determinació d'Esquemes d'Interacció.

Al tercer capítol, on hi ha l'aportació principal de la tesi, veurem què és un Esquema d'Interacció vàlid i com determinar un Esquema d'Interacció vàlid per al processament d'un tipus de transacció a partir de la síntesi dels elements d'un model. Com veurem, això es farà a partir de la síntesi dels elements del model conceptual.

El quart i cinquè capítols presenten l'aplicació del mètode a models conceptuals escrits en el llenguatge Syntropy [CoD94] i en qualsevol llenguatge d'especificació de MCDO [QuO93,QuO94]. En cadascun d'aquests capítols veurem com fer la síntesi dels elements necessaris per a la determinació d'Esquemes d'Interacció a partir de models escrits en el llenguatge corresponent.

El sisè capítol mostra un ús addicional de la síntesi d'elements d'un model. Es tracta d'utilitzar aquesta síntesi per respondre preguntes plantejades per un dissenyador, i que així, aquest dissenyador pugui obtenir una documentació complementària d'un model.

Finalment, al setè capítol trobarem les conclusions i el treball futur que pensem realitzar.

Parts d'aquest treball s'han publicat a [QuO93, QuO94, Que91, Que94].

2. Síntesi d'elements d'un model conceptual necessaris per a la determinació d'Esquemes d'Interacció

Fins ara s'han proposat diversos llenguatges per definir MOO amb interacció implícita, i res no impedeix que se'n puguin proposar d'altres en el futur. Aquesta diversitat ens obligaria, en principi, a trobar un mètode de determinació d'Esquemes d'Interacció específic per a cada llenguatge. Ara bé, el mètode que presentem en aquesta tesi és un mètode *general*, que és aplicable amb les adaptacions pertinents a cadascun.

Si observem els models en els diversos llenguatges orientats a objectes amb interacció implícita, ens adonem que en tots hi ha uns mateixos elements a partir dels quals es poden determinar els Esquemes d'Interacció implícits. Per aconseguir que el nostre mètode sigui general, el que farem serà definir un llenguatge abstracte en el qual sigui possible fer una síntesi de tots els elements rellevants en un d'aquests models, i després donarem un procediment de determinació d'Esquemes d'Interacció que prendrà com a punt de partida la síntesi d'elements d'un model feta en aquest llenguatge abstracte.

El propòsit d'aquest capítol és definir quins són els elements en un model conceptual rellevants per a la determinació d'Esquemes d'Interacció i donar el llenguatge abstracte en el qual s'escriurà la síntesi d'aquests elements. És important fer notar que no pretenem, de cap manera, proposar un nou llenguatge orientat a objectes per a la modelització conceptual, sinó un llenguatge intermedi a partir del qual determinar interaccions.

En capítols posteriors veurem com identificar els elements en models escrits en el llenguatge Syntropy [CoD94] i en qualsevol llenguatge per a MCDO [QuO93, QuO94], així sabrem com fer la síntesi dels elements en qualsevol model en un d'aquests dos llenguatges.

Tot seguit trobem quatre seccions. En cadascuna veurem la notació utilitzada per sintetitzar una de les parts de definició d'un MOO amb interacció implícita, independentment del llenguatge en què estigui escrit. A la secció 2.1 les classes d'esdeveniments externs. A la secció 2.2 tot el que fa referència a les classes d'objectes. A la secció 2.3 les classes d'esdeveniments estructurals i, per acabar, en l'última secció, i la més important, presentarem la notació utilitzada per sintetitzar les relacions entre classes d'esdeveniments.

2.1 Classes d'esdeveniments externs

Un esdeveniment extern és un missatge que rep el sistema des del seu entorn, i que comunica que s'ha produït un canvi en el domini del sistema.

Els esdeveniments externs són els causants indirectes de totes les interaccions que es produeixen en un sistema. Si en un instant determinat no es produeix cap esdeveniment extern, el sistema roman inactiu, i no hi ha cap interacció. Fem notar que, per tant, no considerarem models que incloguin alguna conducta "espontània" que s'origini pel simple pas del temps.

Exemple 2.1.1

Suposem un model en què trobem una classe d'objectes *préstec*, amb un atribut *caducat*, que indica si el préstec està o no caducat, i amb un altre atribut *venciment*, que indica quina és la data de venciment del préstec. En el cas que el llenguatge que s'ha utilitzat per modelar-lo sigui un llenguatge que utilitza regles per definir el valor dels atributs, la regla per definir l'atribut *caducat* podria ser la següent:

caducat(P,T) ← venciment(P,T1), temps(T), T>T1

Aquest és un exemple de model amb conducta espontània. Es pot observar que el simple pas del temps pot originar un fet del tipus *caducat*, i això podria causar diverses interaccions en el model. ♦

En determinar Esquemes d'Interacció mai no estem interessats en els esdeveniments concrets que es produeixen ni en els arguments que poden tenir. Només estem interessats en els noms de les classes d'esdeveniments externs en un model. Designarem *EsdExterns* el conjunt de noms de classes d'esdeveniments externs en un model. Usarem variables de nom CEx, subindexades quan calgui, per referir-nos a

classes d'esdeveniments externs, i noms que comencen en minúscula per referir-nos a classes concretes.

Exemple 2.1.2

Les classes d'esdeveniments externs d'un model d'un sistema de gestió d'una empresa podrien ser EsdExterns = {nouEmpleat, augmentSou, baixaEmpleat}. ♦

2.2 Classes d'objectes

La unitat bàsica d'estructuració d'un MOO és la classe d'objectes. Els objectes del sistema són instàncies d'almenys una classe. Cada objecte té un cert estat, caracteritzat per un conjunt d'atributs.

Objectes

Com en el cas dels esdeveniments externs, no estem interessats en els objectes concrets d'un sistema, sinó només en els noms de les classes d'objectes definides en un model. Designarem *Objectes* el conjunt de noms de classes d'objectes en un model. Usarem variables de nom CO, subindexades quan calgui, per referir-nos a classes d'objectes, i noms que comencen en minúscula per referir-nos a classes concretes.

Exemple 2.2.1

Les classes d'objectes d'un model d'un sistema de gestió d'una empresa podrien ser Objectes = {empleat, directiu, departament}. ♦

Jerarquies d'objectes

Un element essencial dels llenguatges orientats a objectes és la generalització (o especialització) de classes d'objectes o relació IS-A. Tots els llenguatges que volem abastar permeten definir aquesta relació.

Les superclasses d'una classe d'objectes CO són classes d'objectes que són generalitzacions directes de la classe CO. Les subclasses d'una classe d'objectes CO són classes d'objectes que són especialitzacions directes de la classe CO.

Atributs

Cada classe d'objectes en un model té definits zero o més atributs. A més d'aquests atributs, els objectes de la classe tenen també els atributs que la classe hereta de les seves superclasses.

En la síntesi d'un model, en tenim prou amb els noms dels atributs dels objectes de cada classe d'objectes. Designarem *Atributs(CO)* el conjunt dels atributs de la classe

d'objectes CO. Usarem variables de nom A, subindexades quan calgui, per referir-nos a atributs, i noms que comencen en minúscula per referir-nos a atributs concrets. Tenint en compte que, en els MOO, els atributs heretats d'una superclasse poden ser redefinits en les subclasses, diferenciarem un atribut en les classes d'objectes d'una jerarquia amb un prefix en el seu nom que referirà a la classe d'objectes. Considerarem que en un model no hi ha dos atributs definits amb el mateix nom.

Exemple 2.2.2

Els atributs de la classe d'objectes *empleat* de l'exemple anterior podrien ser Atributs(empleat) = {nom, adreça, fills, departament}. ♦

Esdeveniments generats

Hi ha llenguatges de modelització que permeten definir per a una classe d'objectes, una o més classes d'esdeveniments generats. Un esdeveniment generat és un esdeveniment que és detectat per un objecte quan es dóna una circumstància determinada que té interès per a altres objectes o per a l'entorn. A més de les classes d'esdeveniments generats definides en una classe d'objectes, els objectes de la classe hereten les classes d'esdeveniments generats de les seves superclasses.

Exemple 2.2.3

La classe d'objectes *departament* podria incloure la classe d'esdeveniments generats *pressupostExcedit*. Es podrien generar esdeveniments d'aquesta classe quan el volum de les despeses fetes per un departament excedissin el pressupost concedit. Una ocurrència concreta d'aquest esdeveniment podria interessar a l'objecte *empresa* (l'empresa a què pertany el departament podria estar interessada a conèixer si el pressupost d'un dels seus departaments s'ha excedit per considerar de donar al departament un increment d'aquest pressupost) o podria interessar a l'entorn (un dels usuaris del sistema de gestió de l'empresa podria estar interessat a saber quan un departament ha excedit el seu pressupost). ♦

Designarem *Generacions(CO)* el conjunt de les classes d'esdeveniments generats de la classe d'objectes CO. Usarem variables de nom G, subindexades quan calgui, per referir-nos a classes d'esdeveniments generats, i noms que comencen en minúscula per referir-nos a classes concretes. Tenint en compte que, en els MOO, les causes de generació d'esdeveniments d'una classe es poden redefinir en les subclasses que l'hereten, diferenciarem una classe d'esdeveniments generats en les classes d'objectes d'una jerarquia amb un prefix en el seu nom que referirà a la classe d'objectes. Considerarem que en un model no hi ha dues classes d'esdeveniments generats definides amb el mateix nom.

Exemple 2.2.4

Les classes d'esdeveniments generats de la classe d'objectes *departament* podrien ser únicament la de l'exemple 2.2.5. És a dir, Generacions(departament) = {pressupostExcedit}. ♦

Restriccions d'integritat

L'estat dels objectes ha de satisfer en tot moment un conjunt de restriccions d'integritat. La definició d'aquestes restriccions forma part de la definició de les classes d'objectes. La manera concreta com es fa depèn de cada llenguatge, però es pot generalitzar mitjançant els anomenats *predicats d'inconsistència* associats a les restriccions. Si, com a conseqüència d'un o més esdeveniments externs, es produeix algun fet d'un predicat d'inconsistència, llavors es diu que s'ha violat la restricció d'integritat que representa, i els esdeveniments externs es rebutgen. A partir d'un model concret es pot veure quins són els predicats d'inconsistència del sistema i els motius que fan que es puguin produir fets d'aquests predicats.

Exemple 2.2.5

En la classe d'objectes *empleat*, amb un atribut *nom* de l'empleat, podríem trobar la restricció *tots els empleats tenen sempre un nom* definida amb una regla com la següent:

$$\forall E\ \forall T\ (\text{empleat}(E,T) \rightarrow \exists N\ \text{nom}(E,N,T))$$

on s'indica que si E és un empleat en un instant T, ha de tenir un nom en aquest mateix instant. En aquest cas, el predicat d'inconsistència no apareix en el model, però el podríem anomenar *noTéNom*. Es produirà un fet d'aquest predicat quan un o més esdeveniments externs provoquin que la restricció anterior sigui falsa. ♦

Designarem *Restriccions(CO)* el conjunt de les restriccions d'integritat definides en la classe d'objectes CO. Usarem variables de nom Ic, subindexades quan calgui, per referir-nos a restriccions, i noms que comencen en minúscula per referir-nos a predicats d'inconsistència corresponents a restriccions concretes. Considerarem que en un model no hi ha dues restriccions definides amb un mateix predicat d'inconsistència.

Exemple 2.2.6

Les restriccions d'integritat de la classe d'objectes *empleat* podrien ser únicament la de l'exemple 2.2.3. És a dir, Restriccions(empleat) = {noTéNom}. ♦

2.3 Classes d'esdeveniments estructurals

Tots els llenguatges de modelització permeten definir canvis en l'estat del sistema. La forma concreta de fer-ho depèn de cada llenguatge i és difícil de generalitzar. Una possible via de generalització consisteix a considerar que els canvis en l'estat del sistema són provocats per l'ocurrència d'*esdeveniments estructurals*. Es pot establir una analogia entre els esdeveniments estructurals i les operacions primitives d'actualització de l'estat del sistema. Cada operació primitiva (com ara inserir una persona, modificar el sou d'un empleat o esborrar un directiu) correspon a una classe d'esdeveniments estructurals. La invocació d'una d'aquestes operacions, en un instant determinat, correspon a un esdeveniment estructural concret.

Tots els llenguatges permeten definir d'una manera o altra les classes d'esdeveniments estructurals en un sistema, i les condicions i causes de l'ocurrència d'esdeveniments d'aquestes classes. Les classes d'esdeveniments estructurals que existeixen en un model depenen del mateix model i del llenguatge utilitzat. De tota manera, la llista següent és prou àmplia com per abastar tots els llenguatges de què volem tractar:

- Esdeveniments estructurals d'inserció d'objectes en una classe. Hi ha una classe d'esdeveniments d'aquest tipus per a cada classe d'objectes CO definida en un model, i la designem *Inserció(CO)*. L'ocurrència d'un esdeveniment concret de la classe *Inserció(CO)* provoca la inclusió d'un nou objecte en la classe CO. Usarem variables de nom I, subindexades quan calgui, per referir-nos a classes d'esdeveniments estructurals d'inserció, i noms que comencen en minúscula per referir-nos a classes concretes.

- Esdeveniments estructurals d'esborrat d'objectes d'una classe. Hi ha una classe d'esdeveniments d'aquest tipus per a cada classe d'objectes CO definida en un model, i la designem *Esborrat(CO)*. L'ocurrència d'un esdeveniment concret de la classe *Esborrat(CO)* provoca l'esborrat de l'objecte concret de la classe CO. Usarem variables de nom D, subindexades quan calgui, per referir-nos a classes d'esdeveniments estructurals d'esborrat, i noms que comencen en minúscula per referir-nos a classes concretes.

- Esdeveniments estructurals de modificació del valor dels atributs d'un objecte d'una classe. Hi ha una o més classes d'esdeveniments d'aquest tipus per a cada atribut A en un model, i les designem *Modificació(A)*. L'ocurrència d'un esdeveniment concret d'una d'aquestes classes provoca la modificació del valor de l'atribut corresponent. Usarem variables de nom M, subindexades quan calgui, per referir-nos a classes

d'esdeveniments estructurals de modificació, i noms que comencen en minúscula per referir-nos a classes concretes.

- Esdeveniments estructurals de generació d'esdeveniments. Hi ha una classe d'esdeveniments estructurals d'aquest tipus per a cada classe d'esdeveniments generats G en un model, i la designarem *Generació(G)*. L'ocurrència d'un esdeveniment estructural d'aquesta classe provoca l'ocurrència de l'esdeveniment generat corresponent. Usarem variables de nom GE, subindexades quan calgui, per referir-nos a classes d'esdeveniments estructurals de generació d'esdeveniments. Pel fet que hi ha una correspondència biunívoca entre les classes d'esdeveniments generats i les d'esdeveniments estructurals, si es vol es poden usar els mateixos noms per designar les unes i les altres.

- Esdeveniments estructurals de violació de restricció d'integritat. Hi ha una classe d'esdeveniments estructurals d'aquest tipus per a cada restricció Ic definida en un model, i la designarem *Restricció(Ic)*. L'ocurrència d'un esdeveniment d'aquesta classe provoca l'ocurrència de la violació corresponent (i, per tant, el rebuig de l'esdeveniment extern). Usarem variables de nom IcE, subindexades quan calgui, per referir-nos a classes d'esdeveniments estructurals de violació de restriccions d'integritat. Pel fet que hi ha una correspondència biunívoca entre les restriccions d'integritat i les classes d'esdeveniments estructurals de violació, si es vol es poden usar els mateixos noms per designar les unes i les altres.

Exemple 2.3.1

Sigui una classe d'objectes *empleat*, amb atributs *ésDirector* i *sou*, amb un únic esdeveniment generat *important* i una única restricció d'integritat *noTéNom*. Les classes d'esdeveniments estructurals d'aquesta classe són:

Inserció(empleat) = nouEmpleat
Esborrat(empleat) = baixaEmpleat
Modificació(ésDirector) = {altaÉsDirector, baixaÉsDirector}
Modificació(sou) = {souInicial, canviSou}
Generació(important) = important
Restricció(noTéNom) = noTéNom ♦

Totes les classes d'esdeveniments estructurals estan relacionades amb una classe d'objectes, ja que corresponen a elements definits en la classe d'objectes. Usarem l'expressió *EsdEstructurals(CO)* per referir-nos al conjunt de les classes d'esdeveniments estructurals de la classe d'objectes CO. Usarem variables de nom CEs, subindexades quan calgui, per referir-nos a classes d'esdeveniments estructurals,

i noms que comencen en minúscula per referir-nos a classes concretes. Quan vulguem referir-nos a esdeveniments que tant poden ésser externs com estructurals, usarem variables de nom CE.

Una classe d'esdeveniments CEs és classe d'esdeveniments estructurals de la classe d'objectes CO, si es compleix una de les condicions següents:

- CEs és la classe d'esdeveniments estructurals d'inserció d'objectes a la classe CO.
- CEs és la classe d'esdeveniments estructurals d'esborrat d'objectes de la classe CO.
- CEs és una de les classes d'esdeveniments estructurals de modificació d'un dels atributs de la classe CO.
- CEs és una de les classes d'esdeveniments estructurals de generació d'esdeveniments de la classe CO.
- CEs és una de les classes d'esdeveniments estructurals de violació de restriccions d'integritat de la classe CO.

Formalment:

$$\begin{aligned}&\forall\ CEs, CO, A, G, Ic\\ &\quad (CEs = Inserció(CO) \vee CEs = Esborrat(CO) \vee\\ &\quad (CEs \in Modificació(A) \wedge A \in Atributs(CO)) \vee\\ &\quad (CEs = Generació(G) \wedge G \in Generacions(CO)) \vee\\ &\quad (CEs = Restricció(Ic) \wedge Ic \in Restriccions(CO)) \rightarrow\\ &\qquad\qquad CEs \in EsdEstructurals(CO)).\end{aligned}$$

Exemple 2.3.2

Aplicant la definició anterior, veiem que el conjunt de classes d'esdeveniments estructurals de la classe d'objectes *empleat* de l'exemple 2.3.1 és EsdEstructurals(empleat) = {nouEmpleat, baixaEmpleat, altaÉsDirector, baixaÉsDirector, souInicial, canviSou, important, noTéNom}. ♦

2.4 Regles d'esdeveniments

En un sistema concret, els objectes interaccionen ja sigui:

- perquè des d'un objecte es vol consultar l'estat d'un altre objecte,
- perquè ha ocorregut un canvi en l'estat d'un objecte que s'ha de notificar a altres objectes
- o perquè un objecte ha detectat una certa circumstància que ha de ser notificada a altres objectes.

En aquesta tesi estem interessats en el segon i el tercer tipus d'interaccions, ja que són interaccions que poden tenir conseqüències en forma de canvis en l'estat del sistema.

La notificació d'un canvi o d'una circumstància sempre es farà mitjançant una interacció en què s'enviarà l'esdeveniment estructural que representa el canvi o la circumstància detectada a aquells objectes que pugui afectar. Els resultats poden ser nous canvis en l'estat dels objectes destinació de la interacció, la detecció de noves circumstàncies o la violació de restriccions d'integritat.

Per determinar aquestes interaccions, ens cal saber primer de tot les classes d'esdeveniments que poden ocórrer en un sistema, és a dir, les classes d'esdeveniments estructurals i les classes d'esdeveniments externs obtingudes a partir del seu model. Aquests esdeveniments són els missatges que s'envien els objectes entre si o que els objectes reben/envien a l'entorn. D'altra banda, també ens cal saber les relacions de dependència entre l'ocurrència d'esdeveniments de les diferents classes d'esdeveniments. És a dir, com afecta que es produeixin esdeveniments d'una certa classe a que es produeixin o no esdeveniments d'una altra classe. El resultat de la síntesi d'aquestes relacions de dependència entre l'ocurrència d'esdeveniments de diferents classes seran les regles d'esdeveniments.

A partir d'un model podem deduir per a cada classe d'esdeveniments estructurals una o més regles d'esdeveniments. Designarem *Regles(CEs)* el conjunt de les regles que es refereixen a la classe CEs. Usarem variables de nom R, subindexades quan calgui, per referir-nos a regles d'esdeveniments, i noms del tipus r_i per referir-nos a regles concretes.

Com es pot veure, cada regla es refereix a una única classe d'esdeveniments estructurals. Usarem l'expressió *Esdev(R)* per al•ludir a la classe d'esdeveniments estructurals a què es refereix la regla R. Una regla R es refereix a una classe d'esdeveniments estructurals CEs si R pertany al conjunt de regles d'esdeveniments que es refereixen a CEs. Formalment:

$$\forall R \, \forall CEs \, (R \in Regles(CEs) \rightarrow CEs = Esdev(R))$$

Cada regla indica una possible causa del fet que es produeixin esdeveniments estructurals de la classe a què es refereix. Així, cada regla tindrà un conjunt de classes d'esdeveniments causa tal que, si es produeixen un o més esdeveniments de cadascuna de les classes d'esdeveniments en el conjunt, això pot provocar que se'n produeixi un o més de la classe d'esdeveniments a què es refereix la regla. Si d'una o més d'aquestes classes d'esdeveniments causa no es produeixen esdeveniments mai es produiran esdeveniments de la classe d'esdeveniments a què es refereix la regla, per mitjà

d'aquesta regla. Designarem *Causa(R)* el conjunt de classes d'esdeveniments causa (estructurals i/o externs) de la regla R.

Exemple 2.4.1

En el cas d'un model en què trobem la classe derivada *mare*, amb un atribut multivaluat *fills* que pren per valor el nom dels diferents fills d'una mare, i subclasse de la classe bàsica *dona*, les classes d'esdeveniments estructurals són les següents:

- d'inserció de *mare*. Inserció(mare) = altaMare.
- d'esborrat de *mare* . Esborrat(mare) = baixaMare.
- d'inserció de *dona*. Inserció(dona) = altaDona.
- d'esborrat de *dona* . Esborrat(dona) = baixaDona.
- de modificació de *fills*. Modificació(fills) = nouFill.

Suposem que la part del model on es defineix quan es produiran esdeveniments de les classes *altaMare* i *baixaMare* és la següent:

altaMare(D) ← altaDona(D), nouFill(D,N).
altaMare(D) ← dona(D), fills(D,F), F= ∅, not (baixaDona(D)),
nouFill(D,N).
baixaMare(D) ← baixaDona(D).

Es tracta d'un model escrit en un llenguatge de modelització conceptual de regles. La primera regla indica que hi ha una inserció de D com a mare, si D s'ha inserit com a dona i D ha estat mare d'un fill. La segona regla indica que hi ha una inserció de D com a mare, si D és una dona que no té fills, D no ha estat eliminada com a dona i D ha estat mare d'un fill. Per acabar, l'última regla indica que hi ha una eliminació de D com a mare, si D ha esta eliminada com a dona.

Exemple 2.4.2

A partir de les regles de l'exemple anterior, es poden deduir dues regles d'esdeveniments. Els detalls de com deduir les regles d'esdeveniments i les seves classes d'esdeveniments causa a partir de models escrits en determinats llenguatges, els trobarem en capítols posteriors.

Regles(altaMare) = $\{r_1\}$
Regles(baixaMare) = $\{r_2\}$

Les classes d'esdeveniments causa per a cadascuna, són:

Causa (r_1) = {nouFill}
Causa (r_2) = {baixaDona}

El significat d'aquestes regles d'esdeveniments és que si ocorren un o més esdeveniments de la classe d'esdeveniments *nouFill*, això pot causar que n'ocorrin un o més de la classe d'esdeveniments *altaMare*. D'altra banda, si ocorren un o més

esdeveniments de la classe d'esdeveniments *baixaDona,* això pot causar que n'ocorrin un o més de la classe d'esdeveniments *baixaMare.*

Algú es pot estranyar que les dues regles en el model definides pel predicat *altaMare* es passi a una única regla d'esdeveniments i no a dues amb conjunts de classes d'esdeveniments causa:

Causa (r_x) ={altaDona,nouFill}

Causa (r_y) = {nouFill}

El motiu és que un dels dos conjunts de classes d'esdeveniments causa subsumeix l'altre. ♦

Intuïtivament es pot veure que les regles d'esdeveniments ens ajudaran a la determinació d'Esquemes d'Interacció. Només cal pensar en el cas que es produeixin un o més esdeveniments de cadascuna de les classes d'esdeveniments causa d'una regla d'esdeveniments R, que es refereix a una classe d'esdeveniments estructurals d'una classe d'objectes CO. Si això passa, caldran interaccions per enviar els esdeveniments de cadascuna de les classes d'esdeveniments causa de la regla R als objectes de la classe d'objectes CO corresponents.

Exemple 2.4.3

Tenint en compte les regles d'esdeveniments de l'exemple 2.4.2, si ocorren un o més dels esdeveniments de la classe d'esdeveniments *nouFill,* aquests esdeveniments s'hauran d'enviar als objectes de la classe *Mare* corresponents, i podran provocar la inserció d'una o més dones com a *Mare.* ♦

El conjunt de classes d'esdeveniments causa d'una regla no pot ser buit, ja que aquesta regla deixaria de tenir sentit com a tal. D'altra banda, si per a una classe d'esdeveniments estructurals CEs, no es pot deduir d'un model cap regla que s'hi refereixi, és a dir, cap conjunt de classes d'esdeveniments causa, això vol dir que és impossible que ocorrin esdeveniments de la classe CEs al sistema tenint en compte aquest model.

En fer la síntesi de les regles d'esdeveniments a partir d'un model, trobem també altres classes d'esdeveniments rellevants per a les regles d'esdeveniments i la determinació d'Esquemes d'Interacció. Es tracta de les classes d'esdeveniments prerequisit. Cada regla tindrà un conjunt de classes d'esdeveniments prerequisit. El que es produeixin o no esdeveniments d'una d'aquestes classes pot afectar a l'ocurrència o no d'esdeveniments de la classe a què es refereix la regla a través de la regla.

Les classes d'esdeveniments prerequisit són aquelles classes de les quals s'han de conèixer quins esdeveniments s'han produït, abans d'avaluar si realment s'han produït esdeveniments de la classe d'esdeveniments estructurals a què es refereix la regla a través de la regla. Designarem *Prerequisit(R)* el conjunt de classes d'esdeveniments prerequisit (estructurals i/o externs) de la regla d'esdeveniments R.

El conjunt de classes d'esdeveniments prerequisit d'una regla pot ser buit. Això, el que vol dir és que, a part de les classes d'esdeveniments causa, no hi ha cap esdeveniment de cap altra classe d'esdeveniments que pugui afectar a que es produeixin o no esdeveniments de la classe d'esdeveniments estructurals a què es refereix la regla, a través de la regla.

Exemple 2.4.4

Els conjunts de classes d'esdeveniments prerequisit de les regles d'esdeveniments r_1 i r_2 de l'exemple 2.4.2 es troben a continuació. Els detalls de com deduir-les en cada llenguatge concret, els trobarem en capítols posteriors.

Prerequisit (r_1) = {altaDona, baixaDona}

Prerequisit (r_2) = { }

És a dir, per avaluar si ocorreran esdeveniments de la classe d'esdeveniments estructurals *altaMare* a causa de a l'ocurrència d'un o més esdeveniments de la classe *nouFill*, cal saber, a part dels esdeveniments que han ocorregut de la classe *nouFill*, els esdeveniments que han ocorregut de les classes *altaDona* i *baixaDona*. D'altra banda, en el cas de la classe *baixaMare*, per poder avaluar si ocorreran esdeveniments d'aquesta classe a causa de l'ocurrència d'un o més de la classe *baixaDona*, n'hi ha prou de conèixer quins esdeveniments de la classe *baixaDona* han ocorregut. ♦

Intuïtivament es pot veure que les classes d'esdeveniments prerequisit ens ajudaran a la determinació de l'ordre de les interaccions dins dels Esquemes. Només cal pensar en el cas que es produeixin un o més esdeveniments de cadascuna de les classes d'esdeveniments causa d'una regla d'esdeveniments R, que es refereix a una classe d'esdeveniments CEs d'una classe d'objectes CO. Si això passa, les interaccions per enviar els esdeveniments de cadascuna de les classes d'esdeveniments causa de la regla R, als objectes de la classe d'objectes CO, no es podran produir fins que es coneixin els esdeveniments ocorreguts de cadascuna de les classes d'esdeveniments prerequisit de la regla R. Si les interaccions es produeixen abans, no es disposarà de tota la informació necessària poder avaluar els esdeveniments que s'han pogut produir de la classe d'esdeveniments estructurals CEs.

Exemple 2.4.5

Tenint en compte les regles d'esdeveniments dels exemples 2.4.2 i 2.4.4, si ocorre un o més esdeveniments de la classe d'esdeveniments *nouFill*, aquests esdeveniments no s'han d'enviar a la classe *Mare* abans de conèixer els esdeveniments ocorreguts de les classes d'esdeveniments prerequisit *altaDona* i *baixaDona*. Si els esdeveniments de la classe *nouFill* s'envien abans, el resultat de l'avaluació de quins esdeveniments de la classe *altaDona* s'han produit, pot no donar el resultat correcte. ♦

3. Determinació d'Esquemes d'Interacció

Tot seguit presentem la part més important del mètode per determinar Esquemes d'Interacció. És a dir, el procediment que a partir de la síntesi d'elements d'un model en el llenguatge abstracte presentat al capítol 2 i una o més classes d'esdeveniments externs ens donarà un Esquema d'Interacció vàlid per a aquest model i per al cas de l'ocurrència d'esdeveniments de cadascuna de les classes.

El capítol s'estructura en tres seccions. A la secció 3.1 es donen una sèrie de definicions de conceptes bàsics per a la resta del capítol, entre els quals hi ha el concepte d'Esquema d'Interacció. A la secció 3.2 donem una definició declarativa del que és un Esquema d'Interacció vàlid. Per acabar, a la secció 3.3 oferim un procediment concret per determinar Esquemes d'Interacció vàlids.

3.1 Definicions preliminars

Les definicions que trobem a continuació són la base per a la resta del capítol.

Transacció i tipus de transacció

Una transacció TR és un conjunt d'esdeveniments externs que ocorren en un mateix instant de temps T.

Exemple 3.1.1

Un exemple de transacció és la formada pels esdeveniments externs següents: canviDepartament(josep,direcció),canviDepartament(oriol,manteniment),nouDirector (oriol,manteniment). ♦

Un tipus de transacció TTR és un conjunt d'una o més classes d'esdeveniments externs. Una transacció TR de tipus TTR està formada per un o més esdeveniments externs de cadascuna de les classes d'esdeveniments externs a TTR.

Exemple 3.1.2

Un exemple de tipus de transacció és el tipus de transacció al qual pertany la transacció de l'exemple 3.1.1, és a dir: TTR = {canviDepartament, nouDirector}. ♦

Interacció i tipus d'interacció

Una interacció consisteix en l'enviament d'un esdeveniment a un objecte. Una transacció pot provocar diverses interaccions entre els objectes d'un model.

Exemple 3.1.3

Un exemple d'interacció consisteix en l'enviament de l'esdeveniment extern *canviDepartament (josep,direcció)* a l'objecte *direcció* de la classe *departament*. Una altra interacció consisteix en l'enviament de l'esdeveniment extern *canviDepartament(oriol,manteniment)* a l'objecte *manteniment* de la classe *departament*. ♦

Les interaccions d'un mateix tipus són aquelles interaccions en què la classe dels objectes destinació és la mateixa i els esdeveniments que s'envien pertanyen a la mateixa classe d'esdeveniments. Ja hem dit al capítol 1 d'aquesta tesi que l'objectiu del nostre procediment no és determinar quines seran les interaccions concretes causades per l'ocurrència d'una transacció, sinó determinar quins són els tipus d'interacció que poden donar-se com a conseqüència d'un tipus de transacció i quina ha de ser la precedència entre els tipus d'interacció.

Un tipus d'interacció representa les interaccions en què esdeveniments de la classe d'esdeveniments CE s'envien a objectes de la classe CO. Direm que un tipus d'interacció consta de dos components que es modelen mitjançant dos predicats anomenats *què* i *on*[1]. Així, si TI és un tipus d'interacció, el predicat què(TI,CE) indica que CE és la classe dels esdeveniments que s'envien i el predicat on(TI,CO) indica que CO és la classe d'objectes destinació.

Es pot assegurar que, cada tipus d'interacció té un únic valor per a cadascun dels seus components. Formalment:

---

[1] Més endavant afegirem un component més als tipus d'interacció.

$$\forall \text{ TI, } CE_1, CE_2 \text{ (què(TI,}CE_1\text{)} \wedge \text{què(TI,}CE_2\text{)} \rightarrow CE_1 = CE_2\text{)}$$

$$\forall \text{ TI, } CO_1, CO_2 \text{ (on(TI,}CO_1\text{)} \wedge \text{on(TI,}CO_2\text{)} \rightarrow CO_1 = CO_2\text{)}$$

Per simplificar la notació, en algunes parts d'aquest capítol utilitzarem directament tipusInteracció(CE,CO) per referir-nos a un tipus d'interacció TI amb components què(TI,CE) i on(TI,CO).

Exemple 3.1.4

Un exemple de tipus d'interacció és el tipus d'interacció *ti* al qual pertanyen les interaccions de l'exemple 3.1.3. Els seus components són: què(ti,canviDepartament) i on(ti,departament). ♦

Inducció potencial

La inducció potencial és una relació entre una classe d'esdeveniments estructural CEs i un conjunt de classes d'esdeveniments ConjCE que indica que l'ocurrència d'un o més esdeveniments de cadascuna de les classes en el conjunt ConjCE pot induir l'ocurrència d'un o més esdeveniments de la classe CEs. Això no vol dir que hagin d'ocórrer esdeveniments de totes i cadascuna de les classes d'esdeveniments del conjunt ConjCE perquè se n'indueixi un o més de la classe CEs. Només vol dir que si això passa, és possible que se n'indueixi un o més.

Les relacions d'inducció potencial són derivades a partir de les regles d'esdeveniments. Recordem que una regla d'esdeveniments estableix una relació entre una classe d'esdeveniments estructurals i un conjunt de classes d'esdeveniments causa. Tal com hem vist al capítol 2, aquesta relació indica que cal que es produeixin un o més esdeveniments de cadascuna de les classes d'esdeveniments causa perquè sigui possible que es produeixi un o més esdeveniments de la classe d'esdeveniments estructurals a què es refereix la regla.

Diem que una classe d'esdeveniments estructurals CEs és induïda potencialment per un conjunt de classes d'esdeveniments ConjCE, si existeix una regla d'esdeveniments R que es refereix a la classe CEs en què cada classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de R és, o bé induïda potencialment pel conjunt ConjCE, o bé una de les classes d'esdeveniments en el conjunt ConjCE.

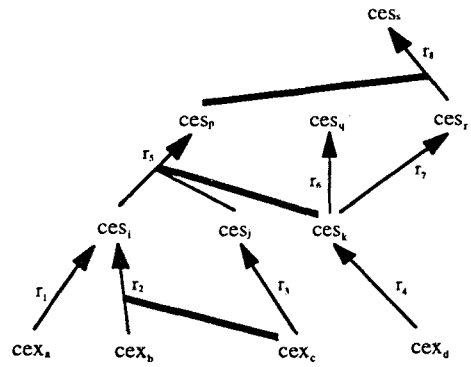


Figura 3.1.1

Exemple 3.1.5

La figura 3.1.1 mostra la representació gràfica de les regles d'esdeveniments sintetitzades a partir d'un model. Les línies fines marquen les classes d'esdeveniments causa de cada regla i les línies gruixudes marquen les seves classes d'esdeveniments prerequisit. Com es pot observar, hi ha vuit regles $r_1,...,r_8$, que relacionen quatre esdeveniments externs, EsdExterns = $\{cex_a,cex_b,cex_c,cex_d\}$, i set esdeveniments estructurals, EsdEstructurals = $\{ces_i,ces_j,ces_k,ces_p,ces_q,ces_r,ces_s\}$, entre si.

$Regles(ces_s) = \{r_8\}$, $Causa(r_8) = \{ces_r\}$, $Prerequisit(r_8) = \{ces_p\}$
$Regles(ces_r) = \{r_7\}$, $Causa(r_7) = \{ces_k\}$, $Prerequisit(r_7) = \{\}$
$Regles(ces_q) = \{r_6\}$, $Causa(r_6) = \{ces_k\}$, $Prerequisit(r_6) = \{\}$
$Regles(ces_p) = \{r_5\}$, $Causa(r_5) = \{ces_i,ces_j\}$, $Prerequisit(r_5) = \{ces_k\}$
$Regles(ces_k) = \{r_4\}$, $Causa(r_4) = \{cex_d\}$, $Prerequisit(r_4) = \{\}$
$Regles(ces_j) = \{r_3\}$, $Causa(r_3) = \{cex_c\}$, $Prerequisit(r_3) = \{\}$
$Regles(ces_i) = \{r_1,r_2\}$, $Causa(r_1) = \{cex_a\}$, $Prerequisit(r_1) = \{\}$,
$Causa(r_2) = \{cex_b\}$, $Prerequisit(r_2) = \{cex_c\}$

A partir d'aquestes regles podem deduir, entre d'altres, la relació d'inducció potencial següent: *inducció($\{ces_k\}$,ces_s)*. Això es pot confirmar observant la semàntica de les regles d'esdeveniments r_7 i r_8. La regla r_7 indica que l'ocurrència d'un o més esdeveniments de ces_k pot causar l'ocurrència d'un o més esdeveniments de la classe ces_q, i la regla r_8 indica que l'ocurrència d'un o més esdeveniments de ces_q pot causar l'ocurrència d'un o més esdeveniments de la classe ces_s. Així, l'ocurrència d'un o més esdeveniments de la classe ces_k pot causar, és a dir induir, l'ocurrència d'un o més esdeveniments de la classe ces_s. ♦

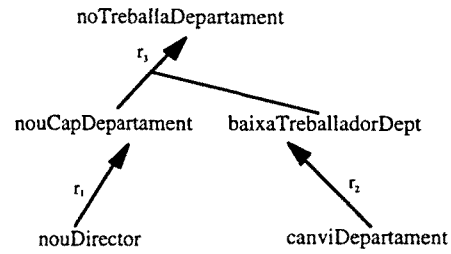


Figura 3.1.2

El predicat inducció(ConjCE,CEs) indica que la classe d'esdeveniments estructurals CEs és induïda potencialment pel conjunt de classes d'esdeveniments ConjCE.

En lògica de primer ordre, la definició del predicat *inducció* és la següent:

$$\forall \text{ ConjCE, CEs}$$
$$(\text{inducció(ConjCE,CEs)} \leftarrow \exists \text{ R } (\text{R} \in \text{Regles(CEs)} \wedge$$
$$\forall \text{ CE } (\text{CE} \in \text{Causa(R)} \rightarrow (\text{inducció(ConjCE,CE)} \vee \text{CE} \in \text{ConjCE})))).$$

Com podem observar, la definició anterior utilitza les funcions *Regles(CEs)* i *Causa(R)* definides al capítol 2.

Exemple 3.1.6

A partir de les regles d'esdeveniments representades graficàment a la figura 3.1.2 i de la definició anterior, deduïm que les relacions d'inducció potencial del conjunt de classes d'esdeveniments {nouDirector, canviDepartament} són:

inducció({nouDirector, canviDepartament},nouCapDepartament)
inducció({nouDirector, canviDepartament},baixaTreballadorDept)
inducció({nouDirector, canviDepartament},noTreballaDepartament) ♦

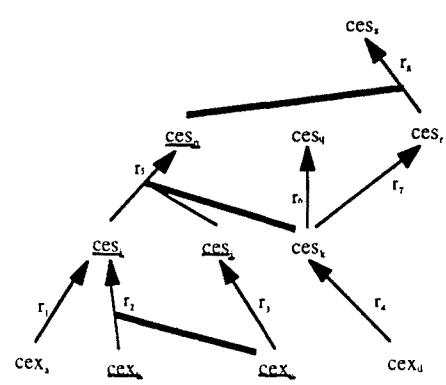


Figura 3.1.3

Una informació interessant, tenint en compte que es vol arribar a determinar quines classes d'objectes poden veure alterat el seu estat a conseqüència d'un tipus de transacció i quines parts d'aquest estat poden canviar, és el conjunt de classes d'esdeveniments estructurals induïdes potencialment per les classes d'esdeveniments externs d'un tipus de transacció TTR. D'aquesta manera es poden saber les classes dels esdeveniments estructurals que es poden produir com a conseqüència de l'ocurrència d'una transacció de tipus TTR i, per tant, quins canvis pot causar TTR en els objectes de les diferents classes en un model.

Exemple 3.1.7

Partint del conjunt de regles d'esdeveniments de la figura 3.1.2, i suposant un tipus de transacció TTR = {nouDirector}, podem arribar a saber quins són els canvis que es poden produir com a conseqüència d'una transacció d'aquest tipus. Només cal saber quines són les classes d'esdeveniments estructurals CEs en què *inducció({nouDirector},CEs)*. Aplicant la definició, veiem que hi ha una única classe induïda potencialment per aquest tipus de transacció:

inducció({nouDirector},nouCapDepartament) ♦

Exemple 3.1.8

A la figura 3.1.3 hi ha novament la representació gràfica de les regles d'esdeveniments utilitzades a l'exemple 3.1.5. En aquesta figura poden veure's subratllades les classes d'esdeveniments induïdes potencialment pel tipus de transacció TTR = {cex_b, cex_c}. ♦

Estat

Els llenguatges de modelització conceptual consideren els canvis d'estat causats per l'ocurrència d'un o més esdeveniments externs en un cert instant (és a dir, d'una transacció) com a instantanis, i només es pot observar l'estat de la base d'informació abans dels canvis i l'estat després dels canvis, figura 3.1.4.

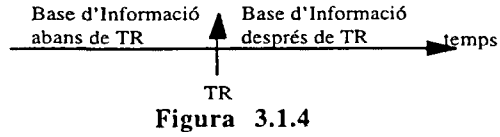


Figura 3.1.4

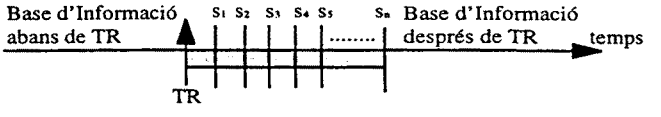


Figura 3.1.5

Tenint en compte que el nostre objectiu és aclarir al dissenyador l'efecte que pot tenir una transacció d'un cert tipus sobre l'estat d'un sistema, el que farem és suposar el processament de les transaccions, com dividit en subestats, figura 3.1.5. El nombre de estats (subestats) màxim en què té sentit aquesta divisió és el número de tipus d'interacció que es poden donar com a conseqüència d'un tipus de transacció. El nombre d'estats mínim és 1. Si es divideix en el nombre d'estats màxim, cada tipus d'interacció estarà assignat a un estat diferent. Si es divideix en el nombre d'estats mínim, tots els tipus d'interacció estaran assignats al mateix estat.

Per a qualsevol de dos estats S_1 i S_2 diferents del processament d'un tipus de transacció sempre es podrà establir una relació d'ordre total, és a dir, sempre serà cert que $(S_1 < S_2) \vee (S_1 > S_2)$. El que farem per facilitar-ho és identificar els estats amb una sèrie de números naturals correlatius que començaran sempre pel número 1.

Si un tipus d'interacció s'assigna a un estat S, això voldrà dir que les interaccions d'aquest tipus es donaran entre l'estat S-1 i l'estat S. Res no impedirà que dos tipus d'interacció que puguin donar-se simultàniament s'assignin a un mateix estat.

Esquema d'Interacció

Un Esquema d'Interacció per al processament d'un tipus de transacció és el conjunt de tipus d'interacció que es poden produir en el processament de les transaccions del tipus. Un Esquema d'Interacció defineix els tipus d'interacció potencials, és a dir, aquells que podrien donar-se durant el processament d'una transacció concreta, però que no s'han de produir necessàriament en totes les transaccions del tipus.

Atès que un tipus d'interacció en un Esquema ha d'estar assignat a un estat, la definició de tipus d'interacció s'ha de modificar afegint-hi un nou component que és l'estat en què les interaccions del tipus es produiran. El nou component es modelarà mitjançant el predicat *estat*. Si TI és un tipus d'interacció amb què(TI,CE) i on(TI,CO), el predicat estat(TI,S) indica que S és l'estat en què s'envien els esdeveniments de la classe CE a objectes de la classe CO.

Com en el cas dels altres dos components, es pot assegurar que, cada tipus d'interacció té un únic valor pel component estat. Formalment:

$$\forall \text{ TI, } S_1, S_2 \text{ (estat(TI,}S_1) \wedge \text{ estat(TI,}S_2) \rightarrow S_1 = S_2)$$

Com s'ha dit anteriorment, per simplificar la notació, en algunes parts d'aquest capítol utilitzarem tipusInteracció(CE,CO,S) per referir-nos a un tipus d'interacció TI amb components què(TI,CE), on(TI,CO) i estat(TI,S).

Exemple 3.1.9

Si tenim un model amb les regles d'esdeveniments de la figura 3.1.2, un Esquema d'Interacció per a un tipus de transacció format per les classes d'esdeveniments externs *nouDirector* i *canviDepartament* és:

EI = { tipusInteracció(nouDirector,departament,1),
tipusInteracció(canviDepartament,treballador,1),
tipusInteracció(nouCapDepartament,departament,2),
tipusInteracció(baixaTreballadorDept,departament,2)}

Podem veure aquest Esquema representat gràficament a la figura 3.1.6. Les transaccions de tipus TTR que segueixin l'Esquema d'Interacció EI es processen en dos subestats. Entre l'estat en què es produeix una transacció de tipus TTR i l'estat 1 s'envien tots els esdeveniments externs de les classes *nouDirector* i *canviDepartament* a objectes de les classes *departament* i *treballador,* respectivament. I entre l'estat 1 i l'estat 2, i definitiu, s'envien tots els esdeveniments de les classes *nouCapDepartament* i *baixaTreballadorDept* a objectes de la classe *departament*. Encara que nosaltres considerem el processament d'una transacció dividit en subestats, recordem que des del punt de vista del sistema d'informació només es pot observar l'estat de la base d'informació abans de la transacció i l'estat després de la transacció. ♦

Cal notar, per acabar, que en un Esquemes d'Interacció EI per a un cert tipus de transacció TTR, tots els esdeveniments que TTR indueix potencialment d'una certa classe d'esdeveniments s'envien sempre a una mateixa classe d'objectes en un mateix estat. Això fa que mai no trobem dos tipus d'interacció de la mateixa classe d'esdeveniments a la mateixa classe d'objectes assignats en diferents estats d'un mateix Esquema d'Interacció. Formalment:

$$\forall \text{ EI, } TI_1, TI_2, CE, CO, S_1, S_2$$
$$(TI_1 \in \text{ EI} \wedge TI_2 \in \text{ EI} \wedge \text{què}(TI_1,CE) \wedge \text{què}(TI_2,CE) \wedge$$
$$\text{on}(TI_1,CO) \wedge \text{on}(TI_2,CO) \wedge$$
$$\text{estat}(TI_1,S_1) \wedge \text{estat}(TI_2,S_2) \rightarrow S_1 = S_2)$$

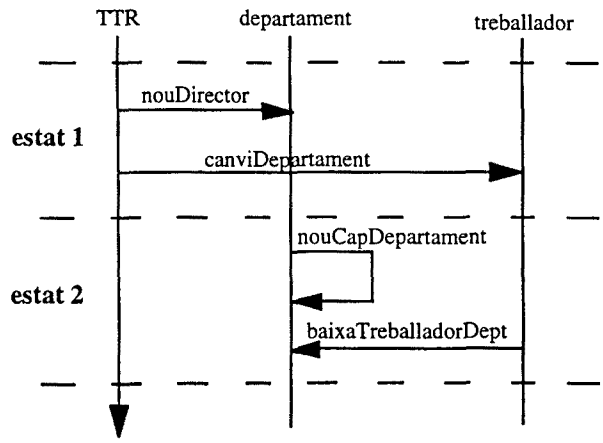


Figura 3.1.6 TTR = {nouDirector,canviDepartament}

Classe d'esdeveniments computada

El concepte de classe d'esdeveniments estructurals computada és necessari per determinar l'estat dins d'un Esquema d'Interacció en què es poden produir les interaccions d'un cert tipus. Un tipus d'interacció no es pot produir en qualsevol estat del processament d'un tipus de transacció. La idea és que, donat un tipus de transacció, no hi pot haver un tipus d'interacció per enviar els esdeveniments d'una certa classe fins que no es coneguin, estiguin computats, tots els esdeveniments d'aquesta classe ocorreguts com a conseqüència del tipus de transacció.

Una classe d'esdeveniments estructurals CEs està computada en un estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si CEs està parcialment computada (vegeu més endavant) a l'estat S respecte de cadascuna de les regles que s'hi refereixen.

Exemple 3.1.10

Tornem a l'Esquema d'Interacciópер al processament del tipus de transacció TTR={nouDirector, canviDepartament} de la figura 3.1.6. Suposem que volem saber si la classe d'esdeveniments *noTreballaDepartament* està computada a l'estat 1 d'EI per al processament de TTR.

Saber si *noTreballaDepartament* està computada, correspondrà a saber si està parcialment computada a l'estat 1 d'EI per al processament de TTR respecte de l'única regla que s'hi refereix, regla r_3. Malgrat que encara no hem definit el concepte de

parcialment computada, intuïtivament podem dir que no hi estarà. Només observant la figura, veiem que en l'estat 1 encara no s'han enviat als objectes de la classe *departament* els esdeveniments de les classes d'esdeveniments *nouCapDepartament* i *baixaTreballadorDept*. És a dir, en l'estat 1 encara no s'han enviat els esdeveniments de les classes d'esdeveniments causa de la regla r_3.

Suposem ara que estem interessats a saber si la mateixa classe d'esdeveniments està computada a l'estat 2 d'EI per al processament de TTR. També intuïtivament, podem dir que sí que hi estarà, ja que a l'estat 2 l'enviament dels esdeveniments ocorreguts de les classes d'esdeveniments *nouCapDepartament* i *baixaTreballadorDept* a la classe *departament* ja s'han fet. ♦

El predicat computada(CEs,S,EI,TTR) indica que la classe d'esdeveniments estructurals CEs està computada a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR.

En lògica de primer ordre, la definició del predicat *computada* és la següent:

$$\forall\ CEs, S, EI, TTR$$
$$(computada(CEs,S,EI,TTR) \leftarrow \forall\ R\ (R \in Regles(CEs) \rightarrow parcialmentComputada(CEs,S,EI,TTR,R))).$$

Una classe d'esdeveniments estructurals CEs està parcialment computada en un estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR respecte d'una regla d'esdeveniments R, si R és una regla d'esdeveniments que es refereix a CEs, CEs és una de les classe d'esdeveniments estructurals de la classe d'objectes CO i es compleix una de les condicions següents:

- Hi ha alguna classe d'esdeveniments CE en el conjunt de classes d'esdeveniments causa de la regla R que no és induïda potencialment per les classes d'esdeveniments en el tipus de transacció TTR i que no és una de les classes en el tipus de transacció TTR. Si passa això, vol dir que en cap cas es produiran esdeveniments de la classe CEs a través de la regla R i, per tant, que CEs estarà computada respecte d'aquesta regla.
- Cadascuna de les classes d'esdeveniments CE en el conjunt de classes d'esdeveniments causa de la regla R és, o bé induïda potencialment per les classes d'esdeveniments en el tipus de transacció TTR o bé una classe d'esdeveniments externs en el tipus de transacció TTR, i en EI hi ha un tipus d'interacció per enviar esdeveniments de les classes d'esdeveniments CE a la classe d'objectes CO en algun estat S_1 anterior o igual a S.

Exemple 3.1.11

Tornem a la primera qüestió plantejada a l'exemple 3.1.10. És a dir, volem saber si la classe d'esdeveniments *noTreballaDepartament* està parcialment computada a l'estat 1 d'EI per al processament de TTR respecte a la regla r_3.

Tenint en compte que r_3 és una regla que es refereix a la classe d'esdeveniments *noTreballaDepartament* i que les classes d'esdeveniments causa de la regla r_3 són *nouCapDepartament* i *baixaTreballadorDept*, podem afirmar que la primera condició no es compleix, ja que totes dues classes d'esdeveniments causa són induïdes potencialment per TTR.

La segona condició en principi sí que es compleix, ja que cadascuna de les classes d'esdeveniments causa de la regla r_3 són induïdes per TTR. Ara bé, després s'exigeix que en algun estat anterior o igual a l'estat 1 en l'Esquema EI hi hagi algun tipus d'interacció per enviar els esdeveniments de cada classe d'esdeveniments causa a la classe d'objectes *departament* (classe d'objectes de la qual *noTreballaDepartament* és classe d'esdeveniments estructural). Com podem observar a la figura 3.1.6, a l'estat 1 (no hi ha cap estat previ) d'EI, no hi ha tipus d'interacció assignats per enviar esdeveniments de les classes *nouCapDepartament* i *baixaTreballadorDept* a la classe *departament*. Així, podem afirmar que la segona condició tampoc no es compleix. ♦

El predicat parcialmentComputada(CEs,S,EI,TTR,R) indica que la classe d'esdeveniments estructurals CEs està parcialment computada a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR respecte a la regla d'esdeveniments R.

En lògica de primer ordre, la definició del predicat *parcialmentComputada* és la següent:

$$
\begin{aligned}
&\forall\ CEs, S, EI, TTR, R, CO \\
&\quad (parcialmentComputada(CEs,S,EI,TTR,R) \leftarrow R \in Regles(CEs) \wedge \\
&\qquad CEs \in EsdEstructurals(CO) \wedge \\
&\qquad ((\exists\ CE\ (CE \in Causa(R) \wedge \neg\ inducció(TTR,CE) \wedge CE \notin TTR)) \vee \\
&\qquad (\forall\ CE\ (CE \in Causa(R) \rightarrow (inducció(TTR,CE) \vee CE \in TTR) \wedge \\
&\qquad\qquad \exists\ TI, S_1\ (TI \in EI \wedge què(TI,CE) \wedge \\
&\qquad\qquad\qquad on(TI,CO) \wedge estat(TI,S_1) \wedge \\
&\qquad\qquad\qquad S_1 \leq S))))).
\end{aligned}
$$

Cal observar que les definicions dels predicats *computada* i *parcialmentComputada*, a part d'usar les funcions *Regles(CEs)* i *Causa(R)* definides al capítol anterior, també utilitzen la funció *EsdEstructurals(CO)*.

Regla d'esdeveniments computable

Aquest concepte ens interessa per saber si els esdeveniments d'una classe d'esdeveniments estructurals a què es refereix una regla, produïts per mitjà d'aquesta regla, es poden computar en un estat d'un Esquema d'Interacció per al processament d'un tipus de transacció. Si és així, tenint en compte aquesta regla, els esdeveniments de les seves classes d'esdeveniments causa, es podran enviar, en aquest estat, a la classe d'objectes corresponent de la classe d'esdeveniments estructurals a què es refereix la regla.

Una regla d'esdeveniments R és computable en un estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR si es compleixen les dues condicions següents:

- Cadascuna de les classes en el conjunt de classes d'esdeveniments causa de la regla R està o bé computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament de TTR o bé és una de les classes d'esdeveniments externs en TTR.
- Cadascuna de les classes en el conjunt de classes d'esdeveniments prerequisit de la regla R que és induïda potencialment pel tipus de transacció TTR està computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament de TTR.

El predicat computable(R,S,EI,TTR) indica que la regla d'esdeveniments R és computable a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

En lògica de primer ordre, la definició del predicat *computable* és la següent:

$$
\begin{aligned}
&\forall\ R, S, EI, TTR \\
&\quad (\text{computable}(R,S,EI,TTR) \leftarrow \\
&\qquad (\forall\ CE\ \ CE \in \text{Causa}(R) \rightarrow (\text{computada}(CE,S\text{-}1,EI,TTR) \vee \\
&\qquad\qquad (CE \in \text{EsdExterns} \wedge CE \in TTR))) \wedge \\
&\qquad (\forall\ CE\ \ CE \in \text{Prerequisit}(R) \wedge \text{inducció}(TTR,CE) \\
&\qquad\qquad \rightarrow \text{computada}(CE,S\text{-}1,EI,TTR))).
\end{aligned}
$$

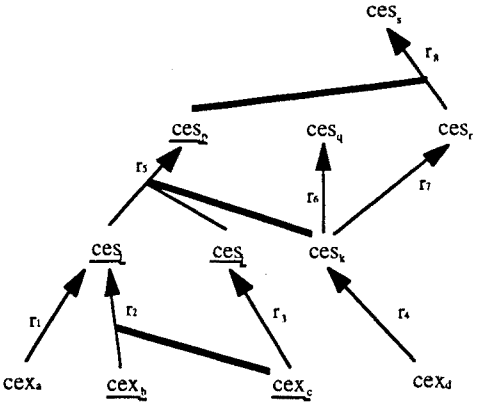


Figura 3.2.1

3.2 Esquemes d'Interacció vàlids

Per a un tipus de transacció, hi pot haver més d'un Esquema d'Interacció vàlid. En aquesta secció donem la definició declarativa de quan un Esquema d'Interacció és vàlid per a un tipus de transacció. Com veurem més endavant, perquè un Esquema d'Interacció EI sigui vàlid per a un tipus de transacció TTR, l'Esquema EI ha de ser complet i correcte per a aquest tipus de transacció.

Que un Esquema d'Interacció sigui complet, vol dir que conté els tipus d'interacció necessaris per al tipus de transacció. Que sigui correcte vol dir que l'Esquema no conté tipus d'interacció sense sentit per al tipus de transacció i que no hi ha problemes amb l'assignació dels diferents tipus d'interacció a estats.

A partir de les definicions declaratives d'aquesta secció, es pot obtenir un procediment per comprovar si un Esquema d'Interacció és vàlid. Una implementació en Prolog d'aquest procediment, es pot trobar a l'annex 1 del capítol.

Esquema d'Interacció complet

Un Esquema d'Interacció EI és complet per a un cert tipus de transacció TTR, si tota classe d'esdeveniments estructurals induïda potencialment per TTR està computada en algun estat de l'Esquema d'Interacció EI. La idea és que amb els tipus d'interacció que agrupa EI, n'hi ha prou per poder computar cadascuna de les classes d'esdeveniments induïdes potencialment pel tipus de transacció TTR. És a dir, que EI permet calcular tots els canvis d'estat que es puguin produir en els objectes de les diferents classes com a conseqüència d'una transacció del tipus TTR.

El predicat *complet(EI,TTR)* indica que l'Esquema d'Interacció EI és complet per a transaccions del tipus TTR.

En lògica de primer ordre, la definició del predicat *complet* és la següent:

$$\forall \text{ EI, TTR } (\text{complet(EI,TTR)} \leftarrow \forall \text{ CEs (inducció(TTR,CEs)} \rightarrow \exists \text{ TI, S (TI} \in \text{EI} \wedge \text{estat(TI,S)} \wedge \text{computada(CEs,S,EI,TTR)))).}$$

Exemple 3.2.1

A la figura 3.2.1 hi ha novament representades les regles d'esdeveniments utilitzades en alguns exemples de la secció anterior, amb classes d'esdeveniments externs EsdExterns = $\{ces_a, ces_b, ces_c, ces_d\}$ i classes d'esdeveniments estructurals corresponents a les classes d'objectes co_α, co_β i co_χ:

$EsdEstructurals(co_\alpha) = \{ces_i, ces_j\}$

$EsdEstructurals(co_\beta) = \{ces_p, ces_k\}$

$EsdEstructurals(co_\chi) = \{ces_q, ces_r, ces_s\}$

A la figura veiem subratllades les classes d'esdeveniments induïdes potencialment per un tipus de transacció TTR = $\{cex_b, cex_c\}$. Un Esquema d'Interacció complet per a TTR serà aquell que amb els seus tipus d'interacció pugui arribar a calcular tots els esdeveniments que es puguin produir de les classes d'esdeveniments estructurals induïdes potencialment per TTR, és a dir, aquell Esquema que en el seu darrer estat, les classes ces_i, ces_j i ces_p estiguin computades.

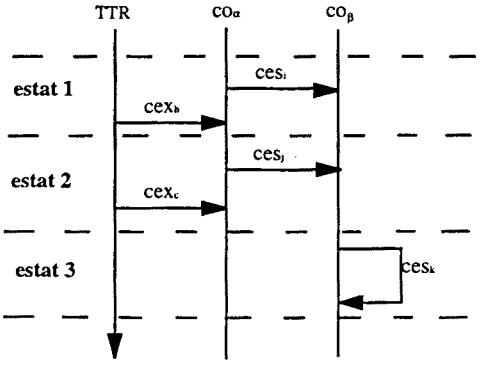


Figura 3.2.2 Esquema d'Interacció E_1. TR = $\{cex_b, cex_c\}$

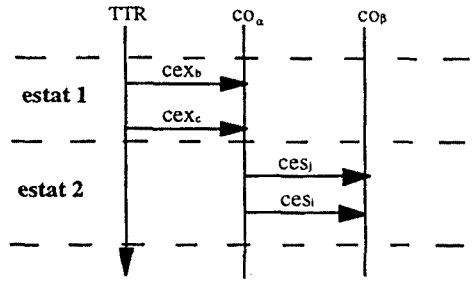


Figura 3.2.3 Esquema d'Interacció E_2. TTR = $\{cex_b, cex_c\}$

Dos Esquemes d'Interacció complets per a TTR són els següents:

EI_1 = {tipusInteracció(ces_i,co_β,1), tipusInteracció(cex_b, co_α,1), tipusInteracció(cex_c, co_α,2), tipusInteracció(ces_j, co_β,2), tipusInteracció(ces_k, co_β,3)}

EI_2 = {tipusInteracció(cex_b, co_α,1), tipusInteracció(cex_c, co_α,1), tipusInteracció(ces_i,co_β,2), tipusInteracció(ces_j, co_β,2)}

Una representació gràfica de cadascun la trobem a les figures 3.2.2 i 3.2.3. ♦

Esquema d'Interacció correcte

Un Esquema d'Interacció EI és correcte per a un tipus de transacció TTR, si cada tipus d'interacció en l'Esquema d'Interacció EI per al processament del tipus de transacció TTR és possible.

El predicat *correcte(EI,TTR)* indica que l'Esquema d'Interacció EI és correcte per al tipus de transacció TTR.

En lògica de primer ordre, la definició del predicat *correcte* és la següent:

$$\forall EI, TTR\ (correcte(EI,TTR) \leftarrow \forall TI\ (TI \in EI \rightarrow interaccióPossible(TI,EI,TTR))).$$

Un tipus d'interacció TI per enviar esdeveniments d'una classe CE a objectes d'una classe CO en un estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR és possible, si existeix una classe d'esdeveniments estructurals CEs i una regla R tal, que:

- R es refereix a CEs
- CE és una de les classes d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R.

- CEs és una classe d'esdeveniments estructurals de CO que és induïda potencialment per les classes d'esdeveniments en TTR, i que no està parcialment computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR respecte a la regla R.

- tota regla d'esdeveniments R_1, que es refereix a una classe d'esdeveniments estructurals CEs_1 de CO i amb un conjunt de classes d'esdeveniments causa al què pertany la classe d'esdeveniments CE, és computable a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

El predicat interacciónPossible(TI,EI,TTR) indica que el tipus d'interacció TI de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR és possible.

En lògica de primer ordre, la definició del predicat *interacciónPossible* és la següent:

$$
\begin{aligned}
&\forall\ TI, EI, TTR, CO, CE, S \\
&\quad (\text{interacciónPossible}(TI,EI,TTR) \leftarrow TI \in EI \wedge \\
&\qquad \text{què}(TI,CE) \wedge \text{on}(TI,CO) \wedge \text{estat}(TI,S) \wedge \\
&\qquad \exists\ CEs\ \exists\ R\ (R \in \text{Regles}(CEs)\ \wedge \\
&\qquad\quad CE \in \text{Causa}(R) \wedge CEs \in \text{EsdEstructurals}(CO) \wedge \\
&\qquad\quad \text{inducció}(TTR,CEs) \wedge \neg\ \text{parcialmentComputada}(CEs,S\text{-}1,EI,TTR,R) \wedge \\
&\qquad\quad (\forall\ CEs_1\ \forall\ R_1\ CEs_1 \in \text{EsdEstructurals}(CO)\ \wedge R_1 \in \text{Regles}(CEs_1)\ \wedge \\
&\qquad\quad CE \in \text{Causa}(R_1) \rightarrow \text{computable}(R_1,S,EI,TTR))))))\text{.}
\end{aligned}
$$

Exemple 3.2.2

Tornem als Esquemes d'Interacció EI_1 i EI_2 de l'exemple 3.2.1. Allà veiem que ambdós són complets. Ara bé, si mirem si són també correctes, ens adonem que només ho és EI_2. En l'Esquema EI_1 hi ha diversos tipus d'interacció que no són possibles per al processament de TTR = $\{cex_b, cex_c\}$. Concretament, no són possibles els tipus d'interacció següents:

- tipusInteracció(ces_i,coβ,1). Existeix una regla, la regla r_5, que es refereix a una classe d'esdeveniments estructurals de la classe d'objectes *coβ* per la qual *ces_i* és una de les classes d'esdeveniments causa. El conjunt de classes d'esdeveniments causa de la regla r_5 és $\{ces_i, ces_j\}$ i el seu conjunt de classes d'esdeveniments prerequisit és $\{ces_k\}$. Ara bé, el problema és que les classes d'esdeveniments causa *ces_i* i *ces_j* i no estan computades a l'estat 0 de l'Esquema EI_1, per al processament de TTR.

- tipusInteracció(ces_j, co_β,2). En aquest cas, la regla problemàtica és també la regla r_5. La diferència és que les classes d'esdeveniments en els conjunts causa i prerequisit no computades a l'estat 1 de l'Esquema d'Interacció EI_1 s'ha reduït a la classe ces_j.

- tipusInteracció(ces_k, co_β,3). Aquí el problema és un altre. Es tracta que no hi ha cap regla que es refereixi a una classe d'esdeveniments estructurals de la classe d'objectes co_β per la qual ces_k sigui una de les classes d'esdeveniments causa. Per tant, aquest tipus d'interacció no té sentit en un Esquema d'Interacció per a transaccions de tipus TTR. ♦

Esquema d'Interacció vàlid

El predicat vàlid(EI,TTR) indica que l'Esquema d'Interacció EI és vàlid per al tipus de transacció TTR.

En lògica de primer ordre, la definició del predicat *vàlid* és la següent:

$$\forall \text{ EI, TTR (vàlid(EI,TTR)} \leftarrow \text{complet(EI,TTR)} \wedge \text{correcte(EI,TTR))}.$$

Exemple 3.2.3

Atès que l'Esquema d'Interacció EI_2, com s'ha vist als exemples anteriors, és complet i correcte per al tipus de transacció TTR = {cex_b, cex_c}, podem dir que és un Esquema d'Interacció vàlid per a aquest tipus de transacció. ♦

3.3 Un procediment concret per a la determinació d'un Esquema d'Interacció vàlid

En aquesta secció donem un procediment per obtenir un Esquema d'Interacció vàlid per a un cert tipus de transacció. El resultat d'aplicar aquest procediment a partir de la síntesi d'un model i per a un cert tipus de transacció TTR és un Esquema d'Interacció vàlid per al processament de TTR. En aquest sentit, podem dir que es tracta d'un procediment limitat, ja que sabem que per a un tipus de transacció hi pot haver més d'un Esquema d'Interacció vàlid.

La diferència bàsica entre els diferents Esquemes d'Interacció vàlids per a un tipus de transacció no són les interaccions que hi puguin haver, sinó la distribució dels tipus d'interacció en estats. El procediment que aquí es presenta obté Esquemes d'Interacció en què els tipus d'interacció s'assignen sempre a l'estat amb el número més baix que sigui possible. És a dir, obté Esquemes d'Interacció en què els tipus d'interacció estan distribuïts en el nombre mínim possible d'estats.

El nostre procediment necessita com a entrada el tipus de transacció pel qual es vol obtenir l'Esquema d'Interacció, TTR, i dóna com a resultat un Esquema d'Interacció vàlid EI.

És bàsicament un bucle que va construint l'Esquema d'Interacció per estats. Cada iteració del bucle troba els tipus d'interacció que poden assignar-se a un estat de l'Esquema d'Interacció. En la primera iteració s'han de determinar els tipus d'interacció que s'assignaran a l'estat 1, en la segona iteració els que s'han d'assignar a l'estat 2, i així anar fent. La condició d'acabament és que després de determinar els tipus d'interacció corresponents a un estat S del processament d'un tipus de transacció TTR i després que aquests tipus d'interacció ja s'hagin incorporat a l'Esquema d'Interacció EI, les classes d'esdeveniments estructurals que són induïdes potencialment per TTR estiguin totes computades a l'estat S de l'Esquema d'Interacció EI. Formalment, la condició d'acabament és:

$$\forall \text{ TTR, EI, S}$$
$$\text{(acabament(TTR,EI,S)} \leftarrow \forall \text{ CEs (inducció(TTR,CEs)} \rightarrow \text{computada(CEs,S,EI,TTR)))}.$$

En cada iteració del bucle, el primer que cal fer és obtenir les regles d'esdeveniments rellevants.

Una regla d'esdeveniments R és rellevant per un l'estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si es refereix a una classe d'esdeveniments estructurals CEs induïda potencialment per TTR, que no està computada a l'estat S d'EI, i que no està parcialment computada a l'estat S d'EI per al processament de TTR respecte a la regla R.

El predicat rellevant(EI,TTR,R,S) indica que la regla d'esdeveniments R és rellevant per a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

$$\forall \text{ EI, TTR, R, S (rellevant(EI,TTR,R,S)} \leftarrow \exists \text{ CEs (R} \in \text{ Regles(CEs)} \wedge \text{ inducció(TTR,CEs)} \wedge \neg \text{ computada(CEs,S,EI,TTR)} \wedge \neg \text{ parcialmentComputada(CEs,S,EI,TTR,R)))}.$$

Un cop es tenen les regles rellevants, s'ha de trobar el subconjunt de regles rellevants que són possibles.

Una regla d'esdeveniments R és possible en un estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si es compleixen les dues condicions següents:

- R és rellevant per a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR
- R és computable a l'estat S de l'Esquema EI per al processament del tipus de transacció TTR.

El predicat possible(EI,TTR,R,S) indica que la regla d'esdeveniments R és possible a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

$$\forall \text{ EI, TTR, R, S (possible(EI,TTR,R,S)} \leftarrow \text{rellevant(EI,TTR,R,S)} \wedge \text{computable(R,S,EI,TTR)).}$$

A partir de les regles possibles es poden obtenir les regles avaluables, que són les que passaran a convertir-se en tipus d'interacció per afegir a l'Esquema d'Interacció.

Una regla d'esdeveniments R és avaluable en un estat S d'un Esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si es compleixen les dues condicions següents:

- R és possible a l'estat S de l'Esquema EI per al processament de TTR
- En el cas que existeixi alguna de classes d'esdeveniments CE en el conjunt de classes d'esdeveniments causa de la regla R, que sigui també classe d'esdeveniments causa d'alguna regla d'esdeveniments R_1 rellevant però no possible, les classes d'esdeveniments estructurals a què es refereixen R i R_1 són de classes d'objectes diferents.

El predicat avaluable(EI,TTR,R,S) indica que la regla d'esdeveniments R és avaluable a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

$$\begin{aligned}\forall \text{ EI, TTR, R, S (avaluable(EI,TTR,R,S)} \leftarrow \text{possible(EI,TTR,R,S)} \wedge \\ \exists \text{ CE, } R_1, CEs_1 \text{ (CE} \in \text{Causa(R)} \wedge \text{CE} \in \text{Causa}(R_1) \wedge \\ \text{rellevant(EI,TTR,}R_1\text{,S)} \wedge \\ \neg \text{ possible(EI,TTR,}R_1\text{,S)} \wedge \\ \text{R} \in \text{Regles(CEs)} \wedge R_1 \in \text{Regles}(CEs_1) \rightarrow \\ \neg \exists \text{ CO (CEs} \in \text{EsdEstructurals(CO)} \\ \wedge CEs_1 \in \text{EsdEstructurals(CO))))).}\end{aligned}$$

Un cop es tenen les regles que són avaluables, ja tenim les regles que es transformaran en tipus d'interacció per afegir a l'Esquema d'Interacció. El pas següent serà afegir aquests tipus d'interacció a EI. L'estat a què s'assignen aquests tipus d'interacció serà el que correspon a l'actual iteració del bucle, S. Per a cada regla avaluable R s'afegirà un tipus d'interacció per a cada classe d'esdeveniments causa CE de la regla R. La classe d'objectes destinació CO serà la classe d'objectes a què pertany la classe d'esdeveniments estructurals a què es refereix la regla R.

El predicat tipusInteracciónAfegir(EI,TTR,S,TI) indica que TI és un dels tipus d'interacció que cal afegir a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

$$\forall\ EI, TTR, S, TI\ (tipusInteracciónAfegir(EI,TTR,S,TI) \leftarrow$$
$$\exists\ R\ \exists\ CE\ \exists\ CO\ \exists\ CEs\ (avaluable(EI,TTR,R,S)\ \wedge$$
$$CE \in Causa(R) \wedge R \in Regles(CEs) \wedge$$
$$Objectes(CO) \wedge CEs \in EsdEstructurals(CO) \wedge$$
$$què(TI,CE) \wedge on(TI,CO) \wedge estat(TI,S))).$$

A l'annex 2 d'aquest capítol, hi ha la implementació en Prolog del procediment aquí descrit.

Annex 1

Aquest annex dóna una implementació en Prolog del procediment per saber si un Esquema d'Interacció és vàlid per a un tipus de transacció, "validador.pl". Aquest procediment, és una adaptació al Prolog de les definicions declaratives de les seccions 3.1 i 3.2 d'aquest capítol. Els grups de regles Prolog corresponents a cadascuna de les definicions estan il•lustrats amb comentaris.

Cal notar que en carregar aquest procediment el primer que es fa és carregar també la síntesi dels elements d'un model necessària per a la validació. Aquesta síntesi ha d'estar al fitxer "sintesi.pl" en el format que s'indica a l'annex 3.

El predicat Prolog que permet fer la validació és *valid(EI,TTR).* Quan es formuli la pregunta, ambdues variables han d'estar instanciades, tal com passa a: valid([tipusInteraccio(nouSou,empleat,1)], [nouSou])?. La resposta és "yes" en el cas que l'Esquema d'Interacció sigui vàlid per al tipus de transacció i "no" en el cas contrari.

```
/**********************************
validador.pl
**********************************/
:- no_style_check(single_var).
:- no_style_check(discontiguous).
:- unknown(_,fail).
:- [sintesi].

/***  ESQUEMA D'INTERACCIÓ VÀLID
Un Esquema d'Interacció EI és vàlid per a un tipus de transacció TTR si EI és
correcte i complet per a aquest tipus de transacció.
****/
valid(EI,TTR) :- complet(EI,TTR),correcte(EI,TTR).
```

```
/***  ESQUEMA D'INTERACCIÓ COMPLET
```

Un Esquema d'Interacció EI és complet per a un tipus de transacció TTR, si a l'últim estat de l'Esquema EI, no hi ha cap classe d'esdeveniments estructurals induïda potencialment per TTR que no estigui computada per al processament de TTR.

```
****/
complet(EI,TTR) :- ultimestat(S,EI),not(algunnocomputat(EI,S,TTR)).

algunnocomputat(EI,S,TTR) :- induccio(TTR,CEs),
                             not(computada(CEs,S,EI,TTR)).

ultimestat(S,EI) :- estat(S,EI),not(altreestat(S,EI)),!.
estat(S,EI) :- in(X,EI),X=..[tipusInteraccio,CE,CO,S].
altreestat(S,EI) :- in(X,EI),X=..[tipusInteraccio,CE,CO,S1],S<S1.

/***  ESQUEMA D'INTERACCIÓ CORRECTE
```

Un Esquema d'Interacció EI és correcte per a un tipus de transacció TTR, si no hi ha cap tipus d'interacció en EI que no sigui un tipus d'interacció possible per al processament de TTR.

```
****/
correcte(EI,TTR) :- not(algunnopossible(EI,TTR)).

algunnopossible(EI,TTR) :- in(TI,EI),not(interaccioPossible(TI,TTR,EI)).
```

```
/***  TIPUS D'INTERACCIÓ POSSIBLE
```

Un tipus d'interacció TI de l'Esquema d'Interacció EI per al tipus de transacció TTR, que correspon a l'enviament d'esdeveniments de la classe CE a objectes de la classe CO en un estat S, és possible, si es compleixen les condicions següents:

- existeix una regla R que es refereix a una classe d'esdeveniments estructurals CEs.
- CE és una de les classes d'esdeveniments causa de la regla R.
- CEs és una classe d'esdeveniments estructurals de la classe d'objectes CO induïda potencialment per TTR, que no està parcialment computada a l'estat S-1 de l'Esquema EI per al processament de TTR respecte a la regla R.
- no hi ha cap regla d'esdeveniments R_1, amb CE en el seu conjunt de classes d'esdeveniments causa i que es refereixi a una classe d'esdeveniments CEs_1 de la classe d'objectes CO, que no sigui computable a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

```
****/
interaccioPossible(TI,TTR,EI) :- in(TI,EI), TI=..[tipusInteraccio,CE,CO,S],
        esdev(R,CEs),causa(R,ConjCA),in(CE,ConjCA),
        esdEstructurals(CO,CEs),induccio(TTR,CEs),S1 is S-1,
        not(parcialmentComputada(CEs,S1,EI,TTR,R)),
        not(algunaReglaNoComputable(CE,CO,S,TTR,EI)).

algunaReglaNoComputable(CE,CO,S,TTR,EI) :-
        causa(R1,ConjCA),in(CE,ConjCA),
        esdEstructurals(CO,CEs1),esdev(R1,CEs1),
        not(computable(R,S,EI,TTR)).
```

/* REGLA D'ESDEVENIMENTS COMPUTABLE**

Una regla d'esdeveniments R és computable a l'estat S de l'esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si és compleixen les dues condicions següents:

- no hi ha cap classe en el conjunt de classes d'esdeveniments causa de la regla R que no estigui computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR i que no sigui una classe d'esdeveniments externs en TTR.
- no hi ha cap classe en el conjunt de classes d'esdeveniments prerequisit de la regla R induïda potencialment pel tipus de transacció TTR i que no estigui computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

```
****/
computable(R,S,EI,TTR) :- causa(R,ConjCA),S1 is S-1,
        not(algunaCausaNoComputada(ConjCA,S1,EI,TTR)),
        prerequisit(R,ConjPR),
        not(algunPrerequisitNoComputat(ConjPR,S1,EI,TTR)).

algunaCausaNoComputada(ConjCA,S,EI,TTR) :- in(CE,ConjCA),
        not(computada(CE,S,EI,TTR)),not(in(CE,TTR)).
algunPrerequisitNoComputat(ConjPR,S,EI,TTR) :- in(CE,ConjPR),
        induccio(TTR,CE),not(computada(CE,S,EI,TTR)).
```

/* CLASSE D'ESDEVENIMENTS COMPUTADA**

Una classe d'esdeveniments estructurals CEs està computada a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR si no hi ha

cap regla que es refereix a CEs per la qual CEs no estigui parcialment computada a l'estat S d'EI per al processament de TTR.

****/

```
computada(CEs,S,EI,TTR) :- not(algunnoparcialment(CEs,S,EI,TTR)).

algunnoparcialment(CEs,S,EI,TTR) :- esdev(R,CEs),
        not(parcialmentComputada(CEs,S,EI,TTR,R)).
```

/* CLASSE D'ESDEVENIMENTS PARCIALMENT COMPUTADA**

Una classe d'esdeveniments estructurals CEs està parcialment computada a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR per una regla R, si és compleix una de les condicions següents:

- R es refereix a CEs i hi ha una classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que no és induïda potencialment per TTR i que no està en TTR.

- R es refereix a CEs, CEs és una classe d'esdeveniments estructurals de la classe d'objectes CO, no hi ha cap classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que o bé no sigui induïda potencialment per TTR o bé no estigui en TTR, i no hi ha cap classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que no s'hagi enviat mitjançant un tipus d'interacció de l'Esquema EI a la classe d'objectes CO en un estat anterior o igual a l'estat S.

****/

```
parcialmentComputada(CEs,S,EI,TTR,R) :- esdev(R,CEs),
        causa(R,ConjCE),in(X,ConjCE),
        not(induccio(TTR,X)),not(in(X,TTR)).
parcialmentComputada(CEs,S,EI,TTR,R) :- esdev(R,CEs),
        esdEstructurals(CO,CEs),causa(R,ConjCE),
        not(algunnoinduccioTTR(TTR,ConjCE)),
        not(algunnointeraccio(ConjCE,CO,S,EI)).

algunnoinduccioTTR(ConjCE,ConjC) :-in(X,ConjC),
                    not(induccio(ConjCE,X)),not(inTTR(TTR,X)).

inTTR(TTR,X) :- esdExterns(CEx),in(X,CEx),in(X,TTR).

algunnointeraccio(ConjCE,CO,S,EI) :- in(X,ConjCE),not(enviat(X,CO,S,EI)).

enviat(X,CO,S,EI) :- in(Y,EI),Y=..[tipusInteraccio,X,CO,S1],S1=<S.
```

/*** **INDUCCIÓ POTENCIAL**

Una classe d'esdeveniments CEs és induïda potencialment per un conjunt de classes d'esdeveniments ConjCE, si existeix una regla R que es refereix a CEs, tal que no hi ha cap classe d'esdeveniments en el seu conjunt de classes d'esdeveniments causa que no sigui induïda potencialment pel conjunt ConjCE i que no estigui en ConjCE.

****/

```
induccio(ConjCE,CEs) :- esdev(R,CEs),causa(R,ConjC),
        not(algunnoinduccio(ConjCE,ConjC)).

algunnoinduccio(ConjCE,ConjC) :- in(X,ConjC),not(induccio(ConjCE,X)),
        not(in(X,ConjCE)).
```

/*** **ESDEVENIMENTS ESTRUCTURALS**

Aquest predicat ens permet a partir d'una classe d'objectes CO obtenir una de les classes d'esdeveniments estructurals de CO. També ens permet, a partir d'una classe d'esdeveniments estructurals CEs saber de quina classe d'objectes és. I finalment, ens permet comprovar si CEs és una classe d'esdeveniments estructurals de la classe d'objectes CO.

****/

```
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),insercio(CO,CEs).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),esborrat(CO,CEs).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),
        modificacio(A,ConjCE),in(CEs,ConjCE),
        atributs(CO,ConjA),in(A,ConjA).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),
        generacio(G,CEs),generacions(CO,ConjG),in(G,ConjG).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),
        restriccio(Ic,CEs),restriccions(CO,ConjIc),in(Ic,ConjIc).
```

/*** **REGLA D'ESDEVENIMENTS - CLASSE D'ESDEVENIMENTS ESTRUCTURALS**

Aquest predicat ens permet saber a partir d'una regla R a quina classe d'esdeveniments estructurals es refereix. També ens permet a partir d'una classe d'esdeveniments estructurals CEs obtenir una regla R que s'hi refereix. I per acabar, ens permet comprovar si R és refereix a la classe d'esdeveniments estructurals CEs.

****/

```
esdev(R,CEs) :- regles(CEs,ConjR),in(R,ConjR).
```

/* PREDICATS AUXILIARS**

El predicat "not(X)" és necessari per afegir la negació a la versió de Prolog que hem utilitzat.

El predicat "in(X,L)" permet comprovar si un element X està en la llista L.

```
****/
not(X):- X, !, fail.
not(X).

in(Y,[]) :- fail.
in(Y,[Y|E]).
in(Y,[Z|E]) :- in(Y,E).
```

Annex 2

Aquest annex dóna una implementació en Prolog del procediment per determinar un Esquema d'Interacció vàlid per a un tipus de transacció, "determinador.pl". Aquest procediment, és una adaptació al Prolog de les definicions declaratives de les seccions 3.1 i 3.3 d'aquest capítol. Els grups de regles Prolog corresponents a cadascuna de les definicions estan il•lustrats amb comentaris.

Cal notar que, encara que hi pot haver més d'un Esquema d'Interacció vàlid per a un tipus de transacció, el procediment que aquí es presenta n'obté un de sol. Concretament, obté l'Esquema d'Interacció en què els tipus d'interacció estan distribuïts en el nombre mínim possible d'estats.

En carregar aquest procediment, el primer que es fa és carregar també la síntesi dels elements d'un model necessària per a la determinació d'Esquemes. Aquesta síntesi ha d'estar al fitxer "sintesi.pl" en el format que s'indica a l'annex 3.

El predicat que permet determinar un Esquema d'Interacció és el predicat *esquema(TTR,EI).* Quan es formuli la pregunta, només haurà d'estar instanciada la variable TTR, tal com passa a: esquema([nouSou],EI)?. La resposta dóna l'Esquema d'Interacció resultat a la variable EI, per exemple: EI = [tipusInteraccio(nouSou, empleat,1)].

```
/************************************
determinador.pl
************************************/
:- no_style_check(single_var).
:- no_style_check(discontiguous).
:- unknown(_,fail).
:- [sintesi].
```

```
/***  ESQUEMA D'INTERACCIO
```

El procediment d'obtenció d'un Esquema d'Interacció vàlid EI per a un tipus de transacció TTR té forma de bucle. Cada iteració del bucle troba els tipus d'interacció que s'assignaran a un estat de l'Esquema d'Interacció. En la primera iteració l'Esquema en construcció està buit i l'estat al que correspon és l'1.

```
****/
esquema(TTR,EI) :- bucle([],TTR,EI,1).
```

```
/***  BUCLE
```

La condició d'acabament del bucle és que no hi hagi cap classe d'esdeveniments estructurals induïda potencialment per TTR que no estigui computada a l'estat al que correspon la nova iteració del bucle. En el cas que aquesta condició es compleixi llavors l'Esquema d'Interacció en construcció passa a ser l'Esquema d'Interacció resultat.

En el cas que la condició no es compleixi cal trobar els tipus d'interacció per afegir a l'Esquema per a l'estat al que correspon la nova iteració del bucle, després afegir aquests tipus d'interacció a l'Esquema d'Interacció en construcció i, finalment, incrementar el número d'estat i passar a la següent iteració del bucle.

```
****/
bucle(EII,TTR,EII,S) :- not(algunNoComputat(EII,TTR,S)).
bucle(EII,TTR,EI,S) :- setof(TI,tipusInteraccioAfegir(EII,TTR,S,TI),ConjTI),
        append(EII,ConjTI,EIN),S1 is S + 1,bucle(EIN,TTR,EI,S1).

algunNoComputat(EI,TTR,S) :- induccio(TTR,CEs),
        not(computada(CEs,S,EI,TTR)).
```

```
/***  AFEGIR INTERACCIONS
```

El tipus d'interacció TI per enviar esdeveniments de la classe d'esdeveniments CE a objectes de la classe CO en un estat S, s'ha d'afegir a un Esquema d'Interacció en construcció EI per al processament d'un tipus de transacció TTR, si existeix una regla R que sigui avaluable a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR tal que, CE és una de les classes d'esdeveniments en el seu conjunt de classes d'esdeveniments causa i la classe d'esdeveniments estructurals a què R es refereix és CEs i CEs és una classe d'esdeveniments estructurals de la classe d'objectes CO.

```
****/
```

```
tipusInteraccioAfegir(EI,TTR,S,TI) :- avaluable(EI,TTR,R,S),
        causa(R,ConjCA),in(CE,ConjCA),esdev(R,CEs),
        esdEstructurals(CO,CEs),TI =.. [tipusInteraccio,CE,CO,S].
```

/* REGLES AVALUABLES**

Una regla R és avaluable a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si:

- R és possible per a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR

- no hi ha cap classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que estigui en el conjunt de classes d'esdeveniments causa d'una regla R1, rellevant i no possible, amb classes d'esdeveniments estructurals a què es refereixen R i R1 de la mateixa classe d'objectes.

```
****/
avaluable(EI,TTR,R,S) :- possible(EI,TTR,R,S),
        not(algunaCausaNoPossible(EI,TTR,R,S)).

algunaCausaNoPossible(EI,TTR,R,S) :- causa(R,ConjCA),in(CA,ConjCA),
        causa(R1,ConjCA1),in(CA,ConjCA1),
        rellevant(EI,TTR,R1,S),not(possible(EI,TTR,R1,S)),
        esdev(R,CEs),esdev(R1,CEs1),
        esdEstructurals(CO,CEs),esdEstructurals(CO,CEs1).
```

/* REGLES POSSIBLES**

Una regla R és possible a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si R és rellevant per a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR i és computable a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR.

```
****/
possible(EI,TTR,R,S) :- rellevant(EI,TTR,R,S),computable(R,S,EI,TTR).
```

/* REGLES RELLEVANTS**

Una regla R és rellevant per a l'estat S de l'Esquema d'Interacció EI per al processament d'un tipus de transacció TTR si R es refereix a una classe d'esdeveniments estructurals induïda potencialment per TTR que no està computada a

l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR i que no està parcialment computada en el mateix estat per la regla R.

```
****/
rellevant(EI,TTR,R,S) :- esdev(R,CEs),
                         induccio(TTR,CEs),not(computada(CEs,S,EI,TTR)),
                         not(parcialmentComputada(CEs,S,EI,TTR,R)).
```

/* REGLA D'ESDEVENIMENTS COMPUTABLE**

Una regla d'esdeveniments R és computable a l'estat S de l'esquema d'Interacció EI per al processament d'un tipus de transacció TTR, si és compleixen les dues condicions següents:

- no hi ha cap classe en el conjunt de classes d'esdeveniments causa de la regla R que no estigui computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR i que no sigui una classe d'esdeveniments externs en TTR.
- no hi ha cap classe en el conjunt de classes d'esdeveniments prerequisit de la regla R induïda potencialment pel tipus de transacció TTR i que no estigui computada a l'estat S-1 de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR.

```
****/
computable(R,S,EI,TTR) :- causa(R,ConjCA),S1 is S-1,
                          not(algunaCausaNoComputada(ConjCA,S1,EI,TTR)),
                          prerequisit(R,ConjPR),
                          not(algunPrerequisitNoComputat(ConjPR,S1,EI,TTR)).

algunaCausaNoComputada(ConjCA,S,EI,TTR) :- in(CE,ConjCA),
        not(computada(CE,S,EI,TTR)),not(in(CE,TTR)).
algunPrerequisitNoComputat(ConjPR,S,EI,TTR) :- in(CE,ConjPR),
        induccio(TTR,CE),not(computada(CE,S,EI,TTR)).
```

/* CLASSE D'ESDEVENIMENTS COMPUTADA**

Una classe d'esdeveniments estructurals CEs està computada a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR si no hi ha cap regla que es refereix a CEs per la qual CEs no estigui parcialment computada a l'estat S d'EI per al processament de TTR.

```
****/
computada(CEs,S,EI,TTR) :- not(algunnoparcialment(CEs,S,EI,TTR)).

algunnoparcialment(CEs,S,EI,TTR) :- esdev(R,CEs),
        not(parcialmentComputada(CEs,S,EI,TTR,R)).
```

/* CLASSE D'ESDEVENIMENTS PARCIALMENT COMPUTADA**

Una classe d'esdeveniments estructurals CEs està parcialment computada a l'estat S de l'Esquema d'Interacció EI per al processament del tipus de transacció TTR per una regla R, si és compleix una de les condicions següents:

- R es refereix a CEs i hi ha una classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que no és induïda potencialment per TTR i que no està en TTR.

- R es refereix a CEs, CEs és una classe d'esdeveniments estructurals de la classe d'objectes CO, no hi ha cap classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que o bé no sigui induïda potencialment per TTR o bé no estigui en TTR, i no hi ha cap classe d'esdeveniments en el conjunt de classes d'esdeveniments causa de la regla R que no s'hagi enviat mitjançant un tipus d'interacció de l'Esquema EI a la classe d'objectes CO en un estat anterior o igual a l'estat S.

****/

```
parcialmentComputada(CEs,S,EI,TTR,R) :- esdev(R,CEs),
        causa(R,ConjCE),in(X,ConjCE),
        not(induccio(TTR,X)),not(in(X,TTR)).
parcialmentComputada(CEs,S,EI,TTR,R) :- esdev(R,CEs),
        esdEstructurals(CO,CEs),causa(R,ConjCE),
        not(algunnoinduccioTTR(TTR,ConjCE)),
        not(algunnointeraccio(ConjCE,CO,S,EI)).

algunnoinduccioTTR(ConjCE,ConjC) :-in(X,ConjC),
                        not(induccio(ConjCE,X)),not(inTTR(TTR,X)).

inTTR(TTR,X) :- esdExterns(CEx),in(X,CEx),in(X,TTR).

algunnointeraccio(ConjCE,CO,S,EI) :- in(X,ConjCE),not(enviat(X,CO,S,EI)).

enviat(X,CO,S,EI) :- in(Y,EI),Y=..[tipusInteraccio,X,CO,S1],S1=<S.
```

/* INDUCCIÓ POTENCIAL**

Una classe d'esdeveniments CEs és induïda potencialment per un conjunt de classes d'esdeveniments ConjCE, si existeix una regla R que es refereix a CEs, tal que no hi ha

cap classe d'esdeveniments en el seu conjunt de classes d'esdeveniments causa que no sigui induïda potencialment pel conjunt ConjCE i que no estigui en ConjCE.

```
****/
induccio(ConjCE,CEs) :- esdev(R,CEs),causa(R,ConjC),
        not(algunnoinduccio(ConjCE,ConjC)).

algunnoinduccio(ConjCE,ConjC) :- in(X,ConjC),not(induccio(ConjCE,X)),
        not(in(X,ConjCE)).
```

/* ESDEVENIMENTS ESTRUCTURALS**

Aquest predicat ens permet a partir d'una classe d'objectes CO obtenir una de les classes d'esdeveniments estructurals de CO. També ens permet, a partir d'una classe d'esdeveniments estructurals CEs saber de quina classe d'objectes és. I finalment, ens permet comprovar si CEs és una classe d'esdeveniments estructurals de la classe d'objectes CO.

```
****/
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),insercio(CO,CEs).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),esborrat(CO,CEs).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),
        modificacio(A,ConjCE),in(CEs,ConjCE),
        atributs(CO,ConjA),in(A,ConjA).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),
        generacio(G,CEs),generacions(CO,ConjG),in(G,ConjG).
esdEstructurals(CO,CEs) :- objectes(ConjCO),in(CO,ConjCO),
        restriccio(Ic,CEs),restriccions(CO,ConjIc),in(Ic,ConjIc).
```

/* REGLA D'ESDEVENIMENTS -**
CLASSE D'ESDEVENIMENTS ESTRUCTURALS

Aquest predicat ens permet saber a partir d'una regla R a quina classe d'esdeveniments estructurals es refereix. També ens permet a partir d'una classe d'esdeveniments estructurals CEs obtenir una regla R que s'hi refereix. I per acabar, ens permet comprovar si R és refereix a la classe d'esdeveniments estructurals CEs.

```
****/
esdev(R,CEs) :- regles(CEs,ConjR),in(R,ConjR).
```

```
/***  PREDICATS AUXILIARS
```

El predicat "not(X)" és necessari per afegir la negació a la versió de Prolog que hem utilitzat.

El predicat "in(X,L)" permet comprovar si un cert element X està dins la llista L.

El predicat "restaCJT(CJ1,CJ2,CJR)" permet obtenir a CJR una llista amb tots els elements que hi ha a la llista CJ1 i que no estan a la llista CJ2.

El predicat "noen(X,CJ1,CJ2)" permet obtenir a X un element que està a la llista CJ1 i que no està a la llista CJ2.

El predicat "interseccioCJT(CJ1,CJ2,CJR)" permet obtenir a CJR una llista amb tots els elements que hi ha a la llista CJ1 i que també estan a la llista CJ2.

El predicat "en(X,CJ1,CJ2)" permet obtenir a X un element que està a la llista CJ1 i que també està a la llista CJ2.

El predicat "totsEn(L1,L2)" permet comprovar que tots els elements de la llista L1 estan també a la llista L2.

El predicat "treureRepetits(L1,L2)" permet obtenir a L2 una llista com L1 però en què s'han eliminat els elements repetits.

```
****/
not(X):- X, !, fail.
not(X).

in(Y,[]) :- fail.
in(Y,[Y|E]).
in(Y,[Z|E]) :- in(Y,E).

restaCJT(CJ1,CJ2,CJR) :- findall(X,noen(X,CJ1,CJ2),CJR),!.

noen(X,CJ1,CJ2) :- in(X,CJ1),not(in(X,CJ2)).

interseccioCJT(CJ1,CJ2,CJR):- findall(X,en(X,CJ1,CJ2),CJR),!.

en(X,CJ1,CJ2) :- in(X,CJ1),in(X,CJ2).

totsEn([],L).
totsEn([X],L) :- in(X,L).
totsEn([X|RX],L) :- totsEn(RX,L),in(X,L).

treureRepetits([],[]).
treureRepetits([S|LS],LL) :- in(S,LS), treureRepetits(LS,LL).
treureRepetits([S|LS],[S|LL]) :- treureRepetits(LS,LL).
```

Annex 3

Tot seguit trobem el format que ha de tenir el fitxer amb la síntesi d'elements d'un model conceptual per comprovar si un Esquema d'Interacció és vàlid per a un cert tipus de transacció, o bé per determinar un Esquema d'Interacció vàlid per a un cert tipus de transacció. El nom del fitxer és "sintesi.pl" i que es carrega quan es carrega en el Prolog el programa validador, "validador.pl", o bé el programa determinador, "determinador.pl".

```
/**********************************
sintesi.pl
**********************************/

/***     ESDEVENIMENTS EXTERNS      ****/
esdExterns(llistaClassesEsdevenimentsExterns).

/***     OBJECTES     ****/
objectes(llistaClassesObjectes).

/***     ATRIBUTS     ****/
atributs(classeObjectes,llistaAtributs).

/***     RESTRICCIONS D'INTEGRITAT     ****/
restriccions(classeObjectes,llistaRestriccionsIntegritat).

/***     ESDEVENIMENTS GENERATS     ****/
generacions(classeObjectes,llistaClassesEsdevenimentsGenerats).

/*** CLASSES D'ESDEVENIMENTS ESTRUCTURALS     *****/
/***                D'INSERCIÓ                ****/
insercio(classeObjectes,classeEsdevenimentsEstructuralsInserció).

/*** CLASSES D'ESDEVENIMENTS ESTRUCTURALS     *****/
/***                D'ESBORRAT                ****/
esborrat(classeObjectes,classeEsdevenimentsEstructuralsEsborrat).
```

```
/*** CLASSES D'ESDEVENIMENTS ESTRUCTURALS *****/
/***          MODIFICACIÓ                 ****/
modificacio(nomAtribut,llistaClassesEsdevenimentsEstructuralsModificació).

/*** CLASSES D'ESDEVENIMENTS ESTRUCTURALS *****/
/***          GENERACIÓ                   ****/
generacio(nomClasseEsdGeneració,classeEsdevenimentsEstructuralsGeneració).

/*** CLASSES D'ESDEVENIMENTS ESTRUCTURALS *****/
/***          RESTRICCIÓ                  ****/
restriccio(nomClasseEsdRestricció,classeEsdevenimentsEstructuralsRestricció).

/***          REGLES D'ESDEVENIMENTS      *****/
regles(nomClasseEsdevenimentsEstructurals, llistaReglesQueShiRefereixen).

/*** CONJUNTS CLASSES D'ESDEVENIMENTS CAUSA ****/
causa(reglaEsdeveniments,llistaClassesEsdevenimentsCausa).

/***CONJUNTS CLASSES D'ESDEVENIMENTS PREREQUISIT***/
prerequisit(reglaEsdeveniments,llistaClassesEsdevenimentsPrerequisit).
```

4. Aplicació al Syntropy

En aquest capítol descriurem com aplicar el mètode per determinar Esquemes d'Interacció a models escrits en el llenguatge Syntropy [CoD94]. Concretament veurem com fer la síntesi dels elements rellevants per a la determinació d'Esquemes d'Interacció en un model conceptual escrit en Syntropy i mostrarem alguns Esquemes obtinguts mitjançant el nostre mètode a partir de la síntesi d'elements en un model en Syntropy.

El capítol es divideix en cinc seccions. A la secció 4.1 donarem una visió general del llenguatge d'especificació Syntropy. A la secció 4.2 veurem un exemple de l'ús d'aquest llenguatge per modelar un sistema concret. A la secció 4.3 mostrarem com fer la síntesi dels elements necessaris per a la determinació d'Esquemes d'Interacció. I per acabar, a la secció 4.4 donarem alguns exemples d'Esquemes d'Interacció, obtinguts mitjançant el nostre mètode, per al cas del model exemple de la secció 4.2.

4.1 Introducció al Syntropy

El mètode Syntropy proposa tres models per arribar a determinar l'arquitectura d'un sistema software: el model Essencial, el model Especificació i el model Implementació.

El model Essencial és una descripció d'una situació real o imaginària, que pot o no contenir software. La majoria de les vegades aquest model es fa per desenvolupar tot seguit un sistema software, encara que si es vol es pot usar simplement per arribar a entendre una situació. Els blocs de construcció del model són objectes, classificats en tipus d'objectes, i esdeveniments, classificats en tipus d'esdeveniments. Els esdeveniments són observables a tot arreu, poden contenir informació i són instantanis.

Cada objecte sap si un esdeveniment determinat l'afecta i quin canvi provoca en el seu estat.

El model Especificació és una descripció d'un sistema software a un nivell d'abstracció alt. S'hi tracta d'especificar el que farà el sistema software. Com en el model Essencial, els blocs de construcció d'aquest model són esdeveniments i objectes. La diferència entre ambdós és que els objectes en el model Especificació poden generar esdeveniments. Per arribar al model Especificació a partir del model Essencial, cal marcar els límits que separen el software del seu entorn.

El model Implementació és una descripció de l'arquitectura del sistema software. Descriu els objectes en el software i com es comuniquen entre si. Els seus blocs de construcció són objectes que es comuniquen enviant-se missatges. Encara que es tracta d'un model independent de qualsevol llenguatge de programació, semànticament és molt proper als llenguatges Smalltalk, C++ i Eiffel. La diferència fonamental amb el model Especificació és que hi ha pas explícit de missatges entre objectes i que els objectes tenen operacions amb les quals processen els missatges i hi responen.

A nosaltres ens interessa el model Especificació, ja que és el que correspon a l'especificació dels requeriments d'un sistema. Descriurem el llenguatge que s'utilitza en un d'aquests models en dues subseccions. En la primera, veurem com es modela la part estàtica o d'estructura d'un sistema, i en la segona, com es modela la seva part dinàmica o de comportament.

4.1.1 Estructura

El Syntropy modela la part estàtica d'un sistema software amb un diagrama de tipus. Els elements que componen aquests diagrames són: tipus d'objectes i associacions.

Tipus d'objectes

Un tipus d'objecte descriu els objectes d'un cert tipus. De cada tipus, se n'especifica el nom, les propietats i els invariants que han de satisfer els seus objectes.

Exemple 4.1.1

A la figura 4.1.1 podem veure un tipus d'objecte *Acció.* ♦

Les propietats que apareixen en la descripció d'un tipus no són totes les que té, sinó únicament aquelles que prenen un valor d'un "tipus de valor" (Number, Integer, String, Date, Time,...). Les que prenen com a valor referències a objectes apareixen representades com a associacions entre tipus. Tant les unes com les altres poden ser:

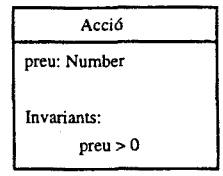


Figura 4.1.1 Tipus d'objecte

- Simples o parametritzades. Una propietat parametritzada és una propietat el valor de la qual depèn d'un o més paràmetres. S'indicarà que una propietat és parametritzada donant el nom i/o el tipus de valor dels seus paràmetres. Per exemple, si el sobresou d'un empleat d'una empresa depèn de la seva edat, aleshores *sobresou* és una propietat parametritzada del tipus d'objecte *Empleat*, el valor de la qual depèn de l'edat de l'empleat.

- Univaluades o multivaluades. Una propietat multivaluada és una propietat que pot prendre un conjunt de valors, un grup de valors o una seqüència de valors. Per dir que una propietat és multivaluada es posa la particula *"set of"*, *"bag of"* o *"seq of"* abans del tipus en que pren el seu valor.

Els invariants per a un tipus d'objecte poden ser:

- Invariants de tipus lògic. Els termes en aquests invariants fan referència al valor de propietats del mateix tipus d'objecte i de propietats d'altres tipus d'objectes a les quals es pot accedir navegant per associacions. La majoria dels invariants d'aquest tipus corresponen a restriccions, encara que també s'usen per definir de manera declarativa el valor de les propietats dels objectes del tipus en funció del seu estat o del d'altres objectes.

- Invariants de propietats. Amb aquests invariants es pot indicar que una propietat tindrà sempre el mateix valor durant tota la vida dels objectes en un tipus (*"const" nomPropietat*), que una propietat ha de tenir valors diferents per a cadascun dels objectes d'un tipus (*"unique" nomPropietat*), o que una propietat pot tenir valor "nil" (*"optional" nomPropietat*).

- Invariant "abstract". Tal com veurem en parlar de jerarquies d'objectes, quan posem aquest invariant en un tipus voldrà dir que cadascun dels objectes del tipus ha de ser també objecte d'un dels seus subtipus.

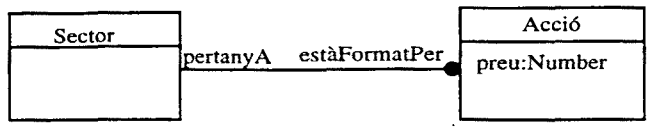


Figura 4.1.2 Associació

Exemple 4.1.2

Un invariant de tipus lògic és el definit per al tipus d'objecte *Acció* de la figura 4.1.1. En aquest cas es correspon a una restricció que indica que el preu d'una acció ha de ser sempre superior a 0. ♦

Associacions

Les associacions són relacions entre objectes. El Syntropy només permet definir associacions binàries, que poden ser reflexives.

En cada costat d'una associació es pot posar una etiqueta, que indica el rol dels objectes de cada tipus en l'associació. Aquestes etiquetes només són obligatòries quan hi ha més d'una associació entre dos tipus d'objectes concrets; en qualsevol altre cas s'agafa com a nom de rol per defecte el del tipus del final de l'associació.

Exemple 4.1.3

A la figura 4.1.2 hi ha un diagrama de tipus amb una associació entre un tipus d'objecte *Sector* i un tipus d'objecte *Acció.* El nom de rol dels objectes del tipus *Acció* és el de l'etiqueta *estàFormatPer.* En el cas de no posar aquesta etiqueta, el nom de rol seria el del mateix tipus, és a dir, *acció.* ♦

Com hem dit en parlar de propietats, cada associació defineix una propietat per a cadascun dels tipus d'objectes que hi intervenen. Es tracta de propietats que prenen com a valor referències a objectes. Els noms d'aquestes propietats són els noms dels rols corresponents.

Exemple 4.1.4

Així, segons la figura 4.1.2 els objectes del tipus *Acció* tenen dues propietats. La primera és la propietat *preu,* que pren un valor de tipus *Number.* La segona és la propietat definida per l'associació, que té per nom *pertanyA* i que prendrà com a valor referències a objectes de tipus *Sector.* ♦

Les propietats definides per una associació també poden ser parametritzades. Això interessa quan es vol que des d'un dels tipus d'objectes, tipus origen, es puguin qualificar els objectes de l'altre tipus, tipus destinació, amb els quals estan relacionats. El qualificador s'explicita juntament amb el tipus d'objecte origen en l'associació.

La multiplicitat d'una associació indica, per a un cert objecte d'un tipus, amb quants objectes de l'altre tipus es pot arribar a relacionar. Les multiplicitats més habituals són:

- de 0 a N. Es representa amb una bombolla negra al final de l'associació.

- de 0 a 1. Es representa amb una bombolla transparent al final de l'associació.
- d'1 a 1. Es representa no posant cap mena de símbol al final de l'associació.

La multiplicitat de 0 a N és pot reduir amb restriccions. Aquestes restriccions apareixen al costat de la bombolla negra. Alguns exemples de restriccions són: *[5..10]*, *[11+]*, *[4]*, que redueixen la multiplicitat a de 5 a 10, més d'11 i exactament 4, respectivament.

Amb la multiplicitat i altres restriccions que es posen en ambdós costats d'una associació, s'especifiquen algunes característiques de les propietats que l'associació defineix:

- La multiplicitat indica si són multivaluades o univaluades, i si és obligatori o no que tinguin valor.
- Les restriccions [bag] i [seq] indiquen, en el cas que una propietat sigui multivaluada, si els valors formen un grup o una seqüència, respectivament. Addicionalment, en aquests casos, es pot indicar un cert ordre entre aquests valors.

Exemple 4.1.5

A la figura 4.1.2, la bombolla negra en un extrem de l'associació vol dir que un objecte *Sector* es pot relacionar amb de 0 a N objectes de tipus *Acció*. D'altra banda, vol dir que la propietat *estàFormatPer*, que pren com a valors referències a objectes de tipus *Acció*, és multivaluada. ♦

Les restriccions entre dues associacions són restriccions direccionals, en què un dels dos tipus que hi participen és font de la restricció, i on sempre es fa referència a associacions en què intervé un mateix objecte del tipus font. El seu format és com el dels invariants de tipus lògic.

Les agregacions són associacions en què els objectes d'un dels tipus, objectes agregats, estan formats per objectes de l'altre tipus, objectes "part de". Es representen posant un rombe al costat del tipus d'objecte agregació. Tot objecte "part de" està permanentment lligat a un objecte agregat, i només es pot eliminar aquest lligam suprimint-lo. D'altra banda, suprimint un objecte agregat se suprimeixen tots els seus objectes "part de".

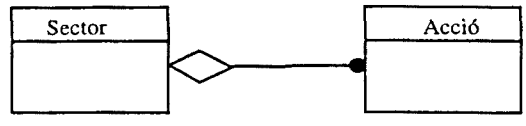


Figura 4.1.3 Agregació

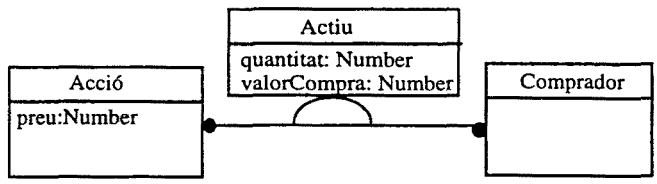


Figura 4.1.4 Tipus associació

Exemple 4.1.6

La relació entre sectors borsaris i accions, que abans s'havia representat com una simple associació, pot ser vista com una agregació a la figura 4.1.3. ♦

Les associacions poden tenir propietats. Quan una associació té més d'una propietat, el Syntropy proposa crear un nou tipus lligat a l'associació, que anomena tipus associació. Aquest tipus pot estar associat amb altres tipus d'objectes, a més dels dos que participen en l'associació.

Exemple 4.1.7

A la figura 4.1.4 podem veure un cas de tipus associació entre els tipus *Acció* i *Comprador*. Les propietats que interessen de l'associació entre objectes d'aquests dos tipus són la quantitat i el valor de compra de l'acció que té el comprador. ♦

Per acabar, les associacions entre dos tipus d'objectes poden ser derivades. Per indicar que una associació és derivada es posa una petita línia que creua l'associació i, al costat, la definició entre claudàtors de com es deriva en termes d'altres associacions i valors de propietats.

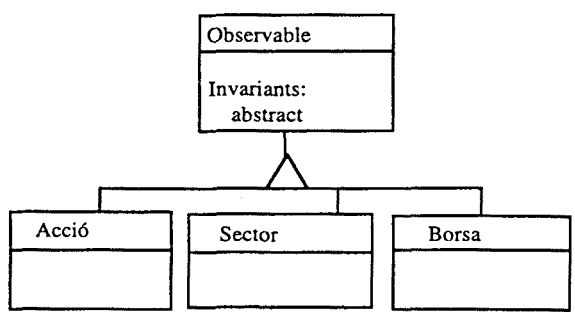


Figura 4.1.5 Jerarquia amb un tipus d'objecte abstracte

Jerarquies de tipus d'objectes

El Syntropy permet establir que un tipus d'objecte és subtipus d'un o més tipus d'objecte. Un subtipus hereta totes les propietats, associacions i invariants definits per al seu o seus supertipus. En general, els objectes en un tipus d'objecte no tenen per què pertànyer a un dels seus subtipus. Tal com hem vist en parlar d'invariants, per indicar que tot objecte d'un tipus ha de ser d'un dels seus subtipus, cal posar l'invariant "abstract".

Exemple 4.1.8

A la figura 4.1.5 podem veure un exemple de tipus d'objecte abstracte. Es tracta del tipus *Observable*. Tot observable ha de ser una acció, un sector o una borsa. ♦

4.1.2 Comportament

La part dinàmica d'un sistema software es modela amb un diagrama d'estats per a cada tipus d'objecte al diagrama de tipus. Un dels elements principals que apareixen en aquests diagrames són els tipus d'esdeveniments.

Tipus d'esdeveniments

Els esdeveniments són fets que ocorren en l'entorn del sistema o que són generats pels objectes del mateix sistema. En cap cas es dirigeixen o s'envien a uns objectes determinats, sinó que es comuniquen al sistema i que cada objecte detecta si el poden afectar o no. Al diagrama d'estats, per a un tipus d'objecte es descriu quins tipus d'esdeveniments interessen als objectes del tipus i com els poden afectar. L'efecte d'un esdeveniment sobre un objecte determinat pot ser un canvi en l'estat de l'objecte i/o la generació d'altres esdeveniments, que seran alhora comunicats al sistema.

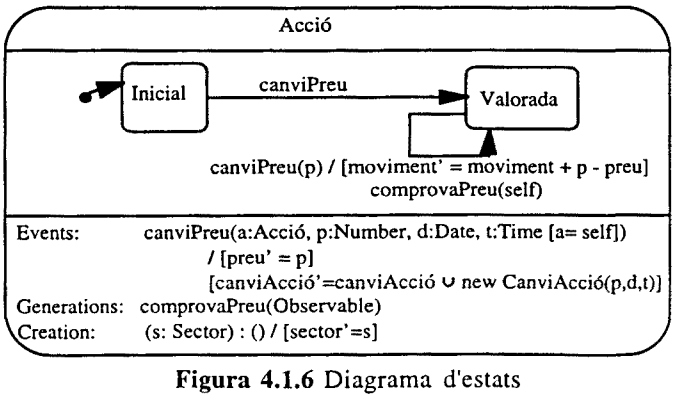


Figura 4.1.6 Diagrama d'estats

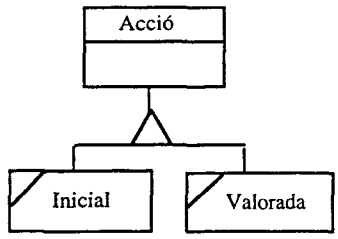


Figura 4.1.7 Diagrama de tipus

En Syntropy no és possible descriure models amb un comportament "espontani". És a dir, qualsevol canvi en l'estat d'un sistema modelat en Syntropy estarà sempre causat directament o indirecta per l'ocurrència d'esdeveniments externs.

Estats

Un diagrama d'estats té tres parts: nom, cos i descripció. El cos està format per una o més màquines d'estat que indiquen per quins estats passen els objectes del tipus durant la seva vida. Cada màquina està formada per un conjunt d'estats i transicions entre aquests estats. S'utilitza més d'una màquina d'estats en el cas de voler modelar els estats pels quals passen els objectes d'un tipus segons més d'un criteri.

Exemple 4.1.9

A la figura 4.1.6 podem veure el diagrama d'estats del tipus d'objecte *Acció*. En aquest diagrama només hi ha una màquina d'estats. Els objectes del tipus *Acció* passen per dos estats durant la seva vida, l'estat *Inicial* i l'estat *Valorada*. Quan es crea un objecte de tipus *Acció,* sempre entra a l'estat *Inicial*. A partir del moment en què se li assigna preu, passa a l'estat *Valorada*. ♦

Per reduir el nombre de transicions en una màquina d'estats, es pot utilitzar el concepte d'estats imbricats. Si un estat E té estats imbricats, les transicions amb origen a l'estat E s'apliquen a tots els objectes en aquest estat, independentment de l'estat imbricat en què estiguin. L'estat imbricat al qual passen els objectes quan se'ls aplica una transició amb destinació E, s'especifica posant una fletxa d'estat inicial que l'assenyali.

Els estats es poden veure com a tipus estat. Un tipus estat representa tots els objectes que hi ha en un estat d'un diagrama d'estats. Aleshores, els estats imbricats dins d'un estat E seran subtipus del tipus estat E. D'altra banda, els estats no imbricats en cap altre estat seran subtipus del tipus objecte que descriuen. El Syntropy proposa, si es vol, estendre el diagrama de tipus reflectint els tipus estat, encara que només se sol fer en el cas que es vulguin mostrar associacions particulars dels objectes en els tipus estat.

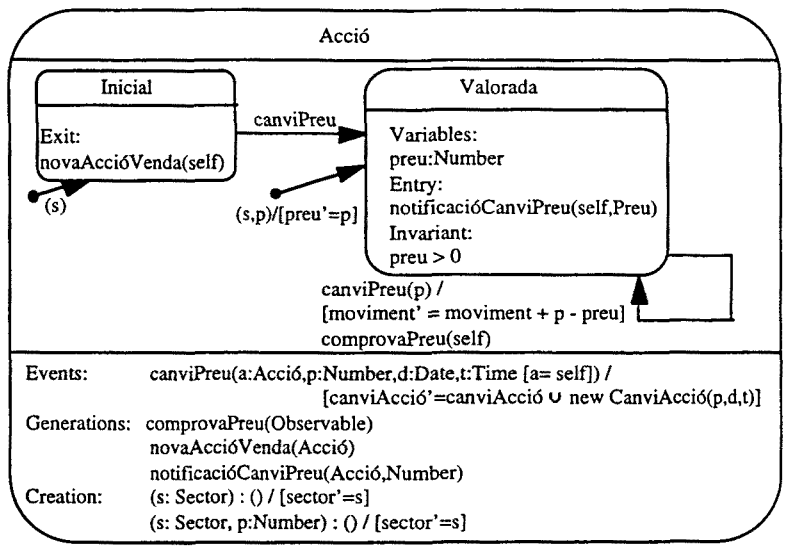


Figura 4.1.8 Diagrama d'estats

Exemple 4.1.10

A la figura 4.1.7 podem veure un diagrama de tipus, estès amb els tipus estats del tipus d'objecte *Acció*.

Per acabar, en un estat d'un diagrama d'estats es poden especificar propietats i invariants que només tenen sentit per als objectes en el tipus estat, esdeveniments que s'han de generar quan un objecte hi entra o en surt i esdeveniments admesos a l'estat (vegeu més endavant).

Exemple 4.1.11

A la figura 4.1.8 hi ha un diagrama d'estats del tipus *Acció*, alternatiu al de la figura 4.1.6. En aquest diagrama s'especifica el preu de les accions com una propietat de les accions en el tipus estat *Valorada*. També s'hi han afegit dos esdeveniments generats, l'esdeveniment *novaAcciόVenda,* que es genera quan una acció surt de l'estat *Inicial,* i l'esdeveniment *notificacióCanviPreu,* que es genera cada vegada que una acció entra a l'estat *Valorada.* ♦

Transicions, esdeveniments i operacions de creació d'objectes

Les transicions apareixen al cos dels diagrames d'estats, concretament entre els estats de les màquines d'estats. Més endavant veurem la sintaxi usada per definir-les.

Els tipus d'esdeveniments apareixen en la descripció dels diagrames. Concretament en una de les parts d'aquesta descripció que s'enumeren tot seguit:

- Una llista d'esdeveniments rellevants, etiquetada amb "Events:", on cada entrada es refereix a un tipus d'esdeveniment que pot afectar l'estat dels objectes del tipus.
- Una llista d'esdeveniments generats, etiquetada amb "Generations:", on cada entrada es refereix a un tipus d'esdeveniment que pot ser generat pels objectes del tipus.
- Una llista d'esdeveniments admesos, etiquetada amb "Allow:", on cada entrada es refereix a un tipus d'esdeveniment que es vol que, en el cas de no causar transicions en els objectes que estan en un cert estat, es consideri que no efectuen cap canvi sobre aquests objectes.

Els diagrames d'estats amb més d'una màquina d'estats poden tenir llistes d'esdeveniments rellevants, generats i admesos particulars per a cada màquina.

Les operacions de creació d'objectes es defineixen també en la descripció dels diagrames, etiquetades amb "Creation:", i només en el cas d'haver d'evitar confusions es completa aquesta definició en les fletxes d'estat inicial del diagrama. Els tipus d'objectes amb més d'una operació de creació distingeixen les diferents operacions pels seus paràmetres. Finalment, en la definició de l'operació de creació d'objectes en un tipus s'invocarà l'operació de creació dels seus supertipus, si és que en té.

La sintaxi utilitzada per descriure transicions, entrades en una llista d'esdeveniments rellevants i operacions de creació té uns elements comuns que veiem a continuació:

- Transicions:

esdeveniment (np_1...np_{tn}) [gd_1]...[gd_{tm}] / [$postc_1$]...[$postc_{tp}$] eg_1...eg_{tq}.

On *esdeveniment* és el nom del tipus d'esdeveniment que pot causar la transició; $np_1...np_{tn}$ són els noms dels paràmetres de l'esdeveniment que apareixen en els guardes, postcondicions i/o esdeveniments generats per la transició; $gd_1...gd_{tm}$ són guardes; $postc_1...postc_{tp}$ són postcondicions, i $eg_1...eg_{tq}$ són esdeveniments generats. L'única part obligatòria en una transició entre estats és el nom del tipus d'esdeveniment.

- Entrades a la llista d'esdeveniments rellevants:

esdeveniment(p_1...p_{rn}[filtre]) [$prec_1$]...[$prec_{rm}$]/[$postc_1$]...[$postc_{rp}$] eg_1...eg_{rq}.

On *esdeveniment* és el nom del tipus d'esdeveniment rellevant; $p_1...p_{rn}$ són els noms de tots els paràmetres de l'esdeveniment amb els seus tipus; $prec_1...prec_{rm}$ són precondicions; $postc_1...postc_{rp}$ són postcondicions, i $eg_1...eg_{rq}$ són esdeveniments generats. Les úniques parts obligatòries en una d'aquestes entrades són el nom del tipus d'esdeveniment i els noms i el tipus dels paràmetres. El *filtre* serveix per indicar quin

és l'objecte o objectes del tipus als què pot afectar l'esdeveniment. Si hi ha un únic paràmetre amb el tipus de l'objecte descrit, llavors no cal filtre, el valor d'aquest paràmetre marcarà l'objecte que pot ser afectat per l'esdeveniment.

- Operacions de creació:

$(p_1...p_{cn})$: $(np_{ci}...np_{cj})$ / $[postc_1]...[postc_{cp}]$.

On $p_1...p_{cn}$ són els noms dels paràmetres de l'operació de creació amb els seus tipus; $np_{ci}...np_{cj}$ són els noms del subconjunt de paràmetres de $p_1...p_{cn}$ que s'usaran per invocar l'operació de creació de l'objecte en el seu supertipus, i $postc_1...postc_{cp}$ són postcondicions. Les postcondicions són l'única part no obligatòria d'aquestes definicions.

Exemple 4.1.12

Al diagrama d'estats de la figura 4.1.6 hi ha dues transicions entre estats. L'una, entre l'estat *Inicial* i l'estat *Valorada,* que només indica el nom del tipus d'esdeveniment que causa la transició, i l'altra, que surt i entra de l'estat *Valorada*, que dóna el nom del tipus d'esdeveniment que la causa, *canviPreu*, el paràmetre *p* que s'utilitza després en la postcondició *[moviment' = moviment + p - preu]* i un esdeveniment generat *comprovaPreu(self).* ♦

Exemple 4.1.13

Al mateix diagrama d'estats, trobem una única entrada a la llista d'esdeveniments rellevants. El tipus d'esdeveniment a què es refereix és *canviPreu.* Els noms i tipus dels seus paràmetres són *a:Acció, p:Number, d:Date* i *t:Time.* El filtre *[a=self]* indica que el valor del paràmetre *a* és el que diu quin objecte es pot veure afectat per un esdeveniment *canviPreu.* I finalment hi ha dues postcondicions. La primera indica que la propietat *preu* de l'objecte *a* de tipus *Acció* passarà a tenir el valor del paràmetre *p* com a conseqüència de l'esdeveniment *canviPreu.* La segona estableix que la propietat multivaluada *canviAcció* de l'objecte *a* de tipus *Acció* passarà a tenir un valor més, que serà el corresponent a un nou objecte del tipus *CanviAcció* que es crearà com a conseqüència de l'esdeveniment *canviPreu.* En aquesta entrada no hi ha precondicions ni tampoc generació d'esdeveniments, encara que en la segona postcondició s'invoca l'operació de creació d'objectes en el tipus *CanviAcció.* ♦

Exemple 4.1.14

En les figures 4.1.6 i 4.1.8 veiem dues maneres de situar la definició d'operacions de creació d'objectes. En la primera tota la definició se situa en la part de descripció del diagrama. En la segona la definició es reparteix entre la descripció i el cos. La raó és que en aquest segon diagrama hi ha dues operacions de creació d'objectes del tipus

Acció, cadascuna de les quals condueix a crear les accions en diferents estats inicials, i una té una postcondició més que l'altra. Cal notar que la diferenciació de les dues operacions es fa mitjançant els seus paràmetres. ♦

Procés d'avaluació d'esdeveniments

En Syntropy els esdeveniments es comuniquen al sistema en un cert ordre, tant en el cas dels esdeveniments externs, és a dir, esdeveniments que el sistema rep del seu entorn, com en el cas d'esdeveniments generats pels objectes del sistema. L'ordre en què es reben dos esdeveniments indica que, el segon no ha d'afectar l'estat del sistema, fins que no ho hagi fet el primer, i tots els esdeveniments que aquest pugui generar [pàg. 146, CoD94].

Exemple 4.1.15

En el cas que es rebi un esdeveniment de tipus *canviPreu* i, a continuació, un de tipus *crearAcció*, l'esdeveniment de tipus crearAcció no ha d'afectar l'estat del sistema fins que ho hagi fet l'esdeveniment de tipus canviPreu i tots els esdeveniments que aquest generi. Entre els esdeveniments generats per *canviPreu* també s'establirà un ordre, i per tant entre si també s'aplicarà la regla anterior. ♦

Ara veurem quins són els passos que se segueixen quan un esdeveniment és comunicat al sistema, i que s'han de realitzar abans de processar qualsevol altre esdeveniment. Se suposa que el procés d'aquests passos és instantani.

- Comprovar que es tracta d'un esdeveniment vàlid per a l'estat del sistema. Es pot trobar la definició de quan un esdeveniment d'una successió ordenada d'esdeveniments és vàlid a [CoD94, pàg. 125-126].

- Obtenir quina o quines transicions provoca l'esdeveniment, per a cada grup de màquines d'estats amb una mateixa llista d'esdeveniments rellevants que contingui una entrada per al tipus de l'esdeveniment.

- Establir les postcondicions corresponents a la transició o transicions que s'han trobat i les postcondicions definides a la llista d'esdeveniments rellevants per al tipus d'esdeveniment en qüestió. Això inclou passar els objectes que canviïn d'estat al seu nou estat. Cal tenir en compte que, en establir les postcondicions, també s'han d'establir de manera immediata totes les conseqüències directes i indirectes d'aquestes postcondicions. És a dir, restablir els invariants i fer la comprovació de les restriccions.

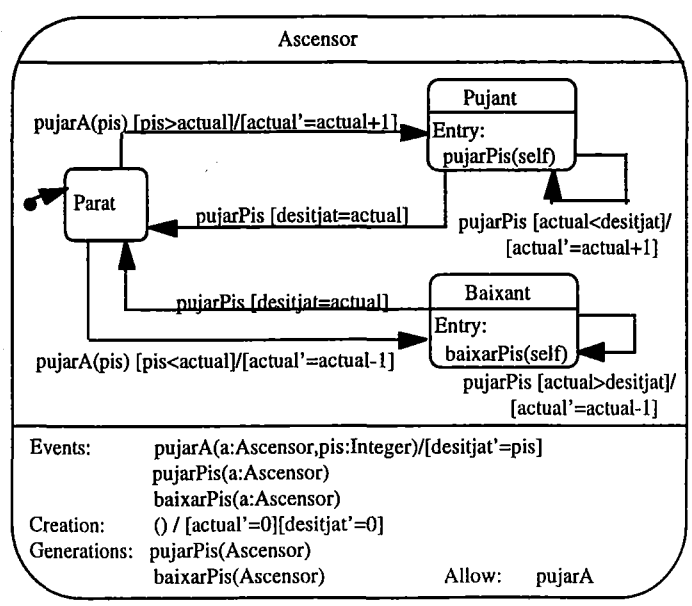
- Generar esdeveniments en un cert ordre. Primer els esdeveniments de sortida de l'estat origen de les transicions, després els definits a la llista d'esdeveniments, després els definits en les transicions i finalment els d'entrada de l'estat destinació de les transicions.

Cal fer notar que els esdeveniments generatsno es notificaran al sistema fins que les postcondicions s'hagin aplicat. Així, podem dir que les precondicions i guardes definides per a un cert tipus d'esdeveniment fan referència a l'estat del sistema en el moment en què ocorren. Aquest, com veurem més endavant, serà un fet important a tenir en compte en el moment de fer la síntesi d'elements necessaris per a la determinació d'Esquemes.

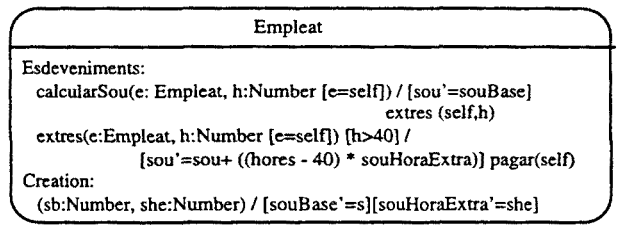
En el cas que un esdeveniment afecti diversos objectes, en què cadascun genera uns certs esdeveniments en un cert ordre, el Syntropy no determina un ordre relatiu entre si. L'únic que assenyala és que totes les postcondicions en els diferents objectes i les conseqüències que puguin tenir s'estableixen abans de generar cap esdeveniment. Això pot portar a especificacions incorrectes. De totes maneres, els autors del Syntropy en són conscients i afirmen que només és així si aquests esdeveniments sense un ordre relatiu poden afectar un mateix objecte, i que això es dóna rarament.

Determinació d'Esquemes d'Interacció en models en Syntropy

La documentació sobre Syntropy de què disposem, l'última que ha sortit fins al moment, és el llibre de Cook i Daniels [CoD94]. Aquest llibre no té la pretensió de ser un manual complet, i això fa que deixi alguns temes oberts. Els models en Syntropy als que podrem aplicar el nostre mètode de determinació d'Esquemes d'Interacció són aquells que es poden especificar a partir del contingut del llibre amb les limitacions que descrivim tot seguit.



4.1.9 Diagrama d'estats Ascensor



4.1.10 Diagrama d'estats Empleat

La primera limitació és que el models no presenti propietats parametritzades, qualificadors en associacions i tipus associació. Aquests elements només allargarien l'explicació de com fer la síntesi dels elements necessaris per a l'anàlisi d'interaccions, però no hi afegirien més dificultat.

Una altra limitació és que en el model, un esdeveniment extern no pugui provocar directament o indirecta més d'un esdeveniment d'un mateix tipus amb un cert ordre d'ocurrència entre si.

Exemple 4.1.16

Un exemple de model que queda exclòs per aquesta limitació és el del diagrama d'estats del tipus *Ascensor* de la figura 4.1.9. Observant aquest diagrama podem veure que un esdeveniment extern del tipus *pujarA* pot causar més d'un esdeveniment del tipus *pujarPis* amb un ordre establert de com aquests esdeveniments han d'anar afectant el sistema. ♦

L'última limitació és que en els models, un esdeveniment extern no pugui provocar directament o indirecta canvis en una part concreta de l'estat del sistema (modificació del valor d'un atribut, inserció d'objectes en una classe,...) causats per esdeveniments de diferents tipus amb un cert ordre d'ocurrència entre si.

Exemple 4.1.17

A la figura 4.1.10 hi ha el diagrama d'estats d'un tipus d'objecte *Empleat.* Un model amb aquest diagrama d'estats queda exclòs per l'última limitació. Observant aquest diagrama podem veure que un esdeveniments extern del tipus *calcularSou* provoca canvis en la propietat *sou* de l'empleat de manera indirecta, a causa d'esdeveniments de diferents tipus, *calcularSou* i *extres*, amb un cert ordre d'ocurrència entre si. ♦

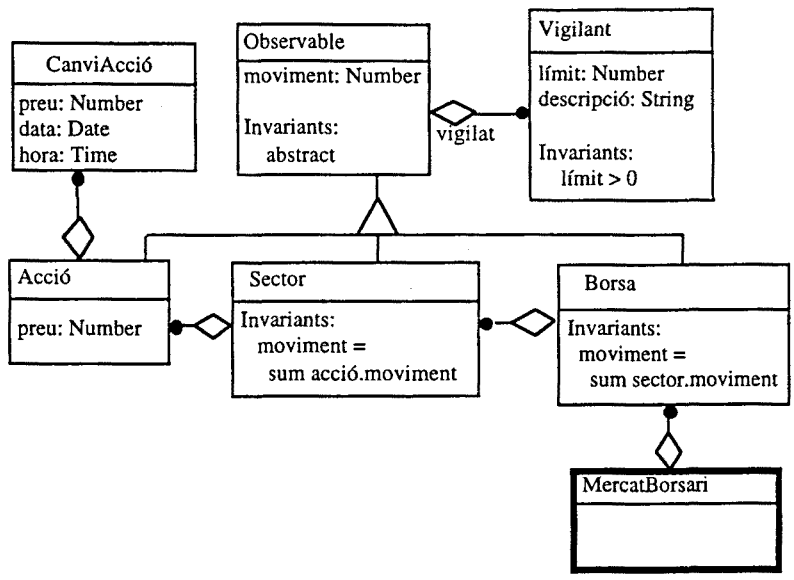


Figura 4.2.1 Diagrama de tipus sistema exemple

El primer pas del mètode de determinació d'Esquemes d'Interacció és fer la síntesi dels elements rellevants en un model. En el cas del Syntropy, se suposa que abans de fer la síntesi s'hauran aplicat al model les regles de conformaça del comportament. A partir d'aquestes regles es pot fer explícit, en un model, tot allò que els tipus d'objectes hereten quant a comportament dels seus supertipus. A la secció 4.2 no s'han explicat aquestes regles perquè allargaria molt la descripció del llenguatge, i es poden consultar a les pàgines 195-215 de [CoD94].

4.2 Sistema exemple

En aquesta secció presentem un model Especificació d'un sistema exemple que anirem usant a la resta del capítol. Es tracta, essencialment, del sistema descrit a [CoD94, pàg. 136 a 156]. La situació en què s'emmarca és el mercat borsari.

A la figura 4.2.1 hi ha el diagrama de tipus del sistema descrit. En aquest, *Acció*, *Sector* i *Borsa* són tipus d'objectes, tots subtipus del tipus d'objecte *Observable*. És a dir, les accions, els sectors i les borses són objectes observables, ja que interessa controlar com incrementen o disminueixen de preu. Aquest increment o disminució es modela amb la propietat *moviment*. Cada *Observable* pot tenir diferents observadors, representats pel tipus d'objecte *Vigilant*, preparats per disparar una alarma en el moment en què l'observable tingui un *moviment*, amb valor absolut, superior a un cert *límit*. Finalment, el tipus d'objecte *CanviAcció* representa els canvis de *preu* d'una acció des que existeix.

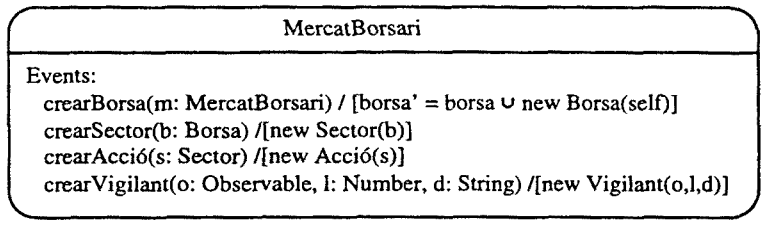


Figura 4.2.2 Diagrama d'estats MercatBorsari

Estructura

Els objectes més rellevants del sistema són les accions, els sectors als quals pertanyen i les borses on es fan operacions amb accions. Tant les accions com els sectors i les borses són objectes observables, ja que interessa tenir-los controlats per saber la quantitat en què n'ha augmentat o ha disminuït el preu des que han entrat al mercat borsari. Es vol que per a cada observable el sistema permeti fixar uns límits, de manera que si el moviment del preu és superior a un d'aquests límits, s'activi una alarma. Finalment, el sistema ha de mantenir un històric dels canvis de preus de les accions durant la seva existència.

El tipus d'objecte *MercatBorsari* té, com es veu, un dibuix amb una línia més gruixuda. La raó és que es tracta del tipus inicial del model. En un model especificació hi ha sempre un tipus inicial amb, com a mínim, un objecte. El tipus MercatBorsari té un únic objecte inicial que representa el conjunt del mercat.

Els invariants que s'han definit en *Sector* i *Borsa* indiquen quin és el valor del moviment d'un sector i d'una borsa. L'invariant en *Observable* assenyala que tot objecte en aquest tipus estarà també en algun dels seus subtipus. L'invariant en el tipus *Vigilant* estableix que el límit de moviment que controla un vigilant serà sempre superior a 0.

Comportament

El comportament del sistema exemple es descriu amb un diagrama d'estats per a cada tipus d'objecte. El diagrama d'estats del tipus inicial *MercatBorsari,* figura 4.2.2, ens permet veure quins són els tipus d'esdeveniments a partir dels quals es començarà a treballar amb el nou sistema. Aquests són esdeveniments que ocorreran a l'entorn del sistema i que causaran que es creïn nous objectes dels tipus *Borsa*, *Sector*, *Acció* i *Vigilant*. L'esdeveniment *crearBorsa*, per exemple, provocarà, tal com es veu a les postcondicions, que la propietat *borsa* de *MercatBorsari* canviï de valor. Cal recordar

que aquesta propietat ve definida per l'associació existent entre els tipus *MercatBorsari* i *Borsa*. El canvi consisteix a afegir-li un nou valor, corresponent a l'associació amb la nova *Borsa* que es crea. Cal notar que *MercatBorsari* no té definida una operació de creació perquè és el tipus inicial del model, i, en aquest, els objectes ja existeixen d'entrada.

A la figura 4.2.3 veiem el diagrama d'estats corresponent al tipus *Borsa*. El més destacable d'aquest diagrama és la llista d'esdeveniments. La seva única entrada indica que l'esdeveniment *canviPreu* d'una acció *a* a un nou preu *p* en una data *d* i una hora *t*, pot afectar els objectes de tipus *Borsa*. El filtre que apareix en aquesta entrada indica que una borsa només es veurà afectada en el cas que l'acció *a* pertanyi a les accions dels sectors associats amb aquesta borsa. Si és així, generarà un esdeveniment *comprovaPreu* amb si mateixa com a paràmetre (self).

El diagrama d'estats corresponent al tipus *Sector*, figura 4.2.4, és molt similar al del tipus *Borsa*. Els sectors també es poden veure afectats pels canvis de preu de les accions. En aquest cas, el sector afectat pel canvi de preu d'una acció *a* és aquell sector al qual pertany l'acció *a* (vegeu filtre en l'única entrada de la llista d'esdeveniments).

El comportament del tipus d'objecte *Acció*, l'hem pogut veure a la figura 4.1.6 de la secció anterior. Aquest diagrama té un cos amb dos estats, l'estat *Inicial*, on entren les accions en crear-les al sistema i l'estat *Valorada*, on passen les accions quan ja se'ls ha assignat un preu. Una acció canvia d'estat quan es produeix un esdeveniment de *canviPreu* de l'acció. En aquest cas s'ha d'assignar un nou valor a la propietat *preu* de l'acció i s'ha d'afegir una associació amb un nou objecte *CanviAcció*, que registra el nou preu en l'històric de canvis de preu.

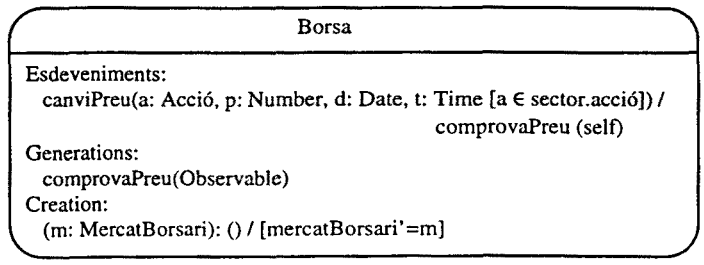


Figura 4.2.3 Diagrama d'estats Borsa

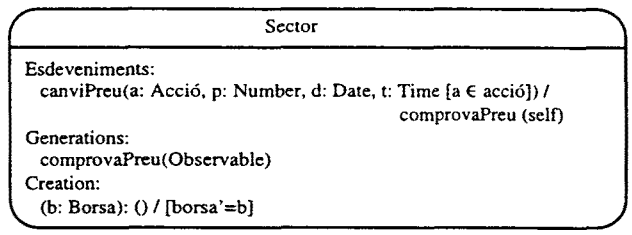


Figura 4.2.4 Diagrama d'estats Sector

Les definicions de les operacions de creació d'objectes en els tipus *Acció*, *Sector* i *Borsa* són molt semblants. En totes s'assenyala que quan es creï un objecte d'un dels tipus s'haurà d'afegir també en el tipus *Observable*, ja que *Acció*, *Sector* i *Borsa* són subtipus d'*Observable*. També en totes es lliga el nou objecte amb un objecte ja existent al sistema, que és un sector, una borsa i el mercat borsari respectivament, i s'assigna un valor a les propietats sector, borsa i mercatBorsari de cada tipus.

El diagrama d'estats del tipus d'objecte *Vigilant*, figura 4.2.5, té un cos amb un únic estat en què estan tots els vigilants actius (en aquest exemple són tots els que existeixen). Alhora, aquest estat té dos estats imbricats, l'estat *Preparat* i l'estat *Activat*. Els vigilants els observables dels quals tenen un moviment inferior al límit estan a l'estat *Preparat*, i aquells que els seus observables tenen un moviment superior al límit, estan a l'estat *Activat*. Quan un vigilant passa de l'estat *Preparat* a l'estat *Activat*, es genera un esdeveniment *alarmaVigilant* (que dispara l'alarma del vigilant). Per acabar, l'esdeveniment *elimina* fa que un vigilant deixi d'existir al sistema.

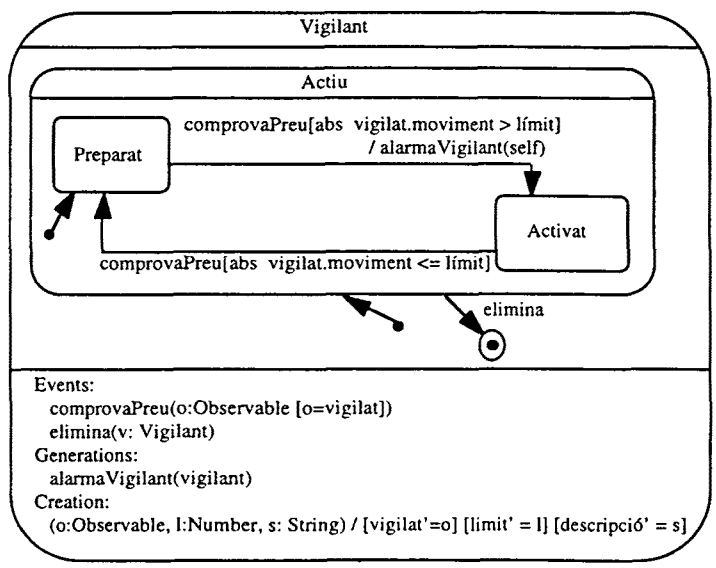


Figura 4.2.5 Diagrama d'estats Vigilant

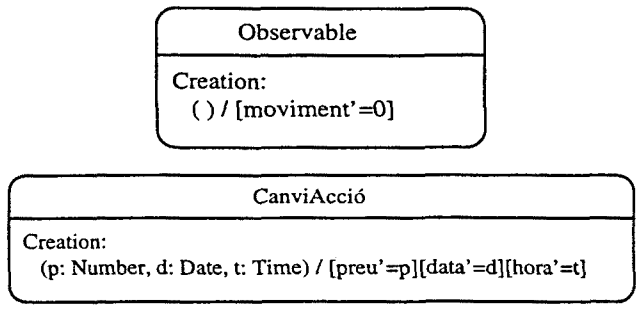


Figura 4.2.6 Diagrames d'estats CanviAcció i Observable

La descripció del comportament associat als tipus d'objectes *Observable* i *CanviAcció* no mereix cap comentari, figura 4.2.6. Potser l'únic remarcable és des d'on es provoca la creació dels objectes en aquests tipus. En el tipus *Observable*, els objectes es creen en el tipus quan es creen en algun dels seus subtipus (vegeu operació de creació d'accions, sectors i borses). En el tipus *CanviAcció,* els objectes es creen en produir-se un canvi de preu d'una acció (vegeu el diagrama d'estats del tipus d'objecte *Acció*).

4.3 Síntesi dels elements necessaris per determinar Esquemes d'Interacció

L'objectiu d'aquesta secció és explicar com fer la síntesi dels elements necessaris per a la determinació d'Esquemes d'Interacció, presentats al capítol 2, a partir d'un model Especificació en Syntropy. Aquesta explicació la donarem element per element repartida en dues subseccions. Una primera subsecció en què veurem la síntesi del elements bàsics i una segona subsecció en què veurem la síntesi de les regles d'esdeveniments. Tant a l'una com a l'altre posarem exemples basats en el model de la secció 4.2. La síntesi completa del model exemple es pot trobar a l'annex que s'inclou al final del capítol.

4.3.1 Síntesi d'elements bàsics

Els elements bàsics són els elements relacionats amb classes d'esdeveniments externs, classes d'objectes i classes d'esdeveniments estructurals.

Classes d'esdeveniments externs

En la síntesi ens interessa tenir el conjunt de noms de classes d'esdeveniments externs definides en un model. Aquest conjunt es coneix amb el nom d'*EsdExterns*.

En Syntropy els noms de les classes d'esdeveniments es poden obtenir a partir de les llistes d'esdeveniments rellevants i les llistes d'esdeveniments generats als diagrames d'estats dels diferents tipus d'objectes. Concretament, són aquells noms de tipus d'esdeveniments en alguna entrada d'una llista d'esdeveniments rellevants que no apareixen en cap entrada de cap llista d'esdeveniments generats. És a dir, les classes d'esdeveniments externs en un model són aquells tipus d'esdeveniments que no són generats pel mateix sistema.

Exemple 4.3.1

En el sistema exemple, els noms dels tipus d'esdeveniments en les entrades de les llistes d'esdeveniments rellevants són *crearBorsa*, *crearSector*, *crearAcció*, *crearVigilant*, *canviPreu*, *comprovaPreu* i *elimina*, i els noms dels tipus d'esdeveniments en les entrades de les llistes d'esdeveniments generats són *comprovaPreu* i *alarmaVigilant*. Això vol dir que el conjunt de noms de les classes d'esdeveniments externs és EsdExterns = {crearBorsa, crearSector, crearAcció, crearVigilant, canviPreu, elimina}. ♦

Classes d'objectes

En la síntesi necessitem el conjunt de noms de les classes d'objectes definides en un model. Aquest conjunt es coneix amb el nom d'*Objectes*.

En Syntropy el conjunt de noms de classes d'objectes està format pels noms dels tipus d'objectes i els noms dels tipus estat en un model.

Exemple 4.3.2

Els tipus d'objectes al sistema exemple són *MercatBorsari*, *Borsa*, *Sector*, *Acció*, *CanviAcció*, *Observable* i *Vigilant*, i els tipus estat són *Inicial*, *Valorada*, *Actiu*, *Preparat* i *Activat*. Això vol dir que el conjunt de noms de les classes d'objectes en aquest sistema és Objectes = {mercatBorsari, borsa, sector, acció, canviAcció, observable, vigilant, accióInicial, accióValorada, vigilantActiu, vigilantPreparat, vigilantActivat}. ♦

Atributs. En la síntesi interessa el conjunt de noms d'atributs de cadascuna de les classes d'objectes en un model. Aquest conjunt es coneix amb el nom d'*Atributs(CO)*, on CO és la classe d'objectes.

En Syntropy els atributs d'una classe d'objectes s'especifiquen de maneres diferents segons la classe d'objectes:

- Si la classe d'objectes correspon a un tipus d'objecte, els atributs són les propietats dels objectes en aquest tipus. Cal recordar que aquestes propietats no són només les que apareixen en la definició del tipus al diagrama de tipus, sinó també les que vénen donades per les associacions en què participa i les heretades dels seus supertipus.

- Si la classe d'objectes correspon a un tipus estat, els atributs són les propietats dels objectes en aquest estat. És a dir, les que apareixen a la definició de l'estat al diagrama d'estats, les que vénen donades per associacions en què participa i les heretades dels seus supertipus.

El nom d'un atribut pot estar format pel nom de la classe i el nom de la propietat a què correspon, *nomClasseNomPropietat,* o simplement pel nom de la propietat, *nomPropietat.* El primer cas és necessari quan dues classes tenen un atribut amb el mateix nom.

Exemple 4.3.3

El tipus d'objecte *Acció* té la propietat *preu* en la seva definició al diagrama de tipus, vegeu figura 4.2.1, té dues propietats donades per les associacions en què participa, anomenades *canviAcció* i *sector,* i hereta dues propietats del tipus d'objecte *Observable,* que són *moviment* i *vigilant.* Això vol dir que el conjunt de noms d'atributs de la classe d'objectes *acció* és Atributs(acció) = {accióPreu, accióCanviAcció, accióSector, accióMoviment, accióVigilant}.

Una classe d'objectes corresponent a un tipus estat, és la classe *accióValorada.* En la seva definició no apareix cap propietat, tampoc no té cap associació amb altres tipus, però sí que hereta les propietats del tipus d'objecte *Acció.* Per tant, Atributs(accióValorada) = {accióValoradaPreu, accióValoradaCanviAcció, accióValoradaSector, accióValoradaMoviment, accióValoradaVigilant}.♦

Esdeveniments generats. En la síntesi ens interessa tenir el conjunt de noms de les classes d'esdeveniments generats per a cada classe d'objectes d'un model. Es coneix aquest conjunt amb el nom de *Generacions(CO),* on CO és la classe d'objectes.

Com en el cas dels atributs, les classes d'esdeveniments que poden generar els objectes d'una classe d'objectes s'especifiquen de diferents maneres segons la classe d'objectes:

- Si la classe d'objectes correspon a un tipus d'objecte, les classes d'esdeveniments generats són les que corresponen als tipus d'esdeveniments en cada entrada de la llista o llistes d'esdeveniments generats definides al diagrama d'estats del tipus d'objectes. Ara bé, hi ha altres tipus d'esdeveniments que no estan en aquestes llistes i que poden també

ser considerats com a generats. Són els que es troben en postcondicions d'alguna entrada de les llistes d'esdeveniments rellevants o d'alguna transició i en la definició de les operacions de creació del tipus d'objecte i que van dirigits a crear un objecte en algun altre tipus d'objecte.

- Si la classe d'objectes correspon a un tipus estat, les classes d'esdeveniments generats són les que apareixen a la definició del tipus estat. Concretament, a les llistes d'esdeveniments etiquetades amb "Entry:" i "Exit:". A la figura 4.1.8 de la secció 4.1 hem vist exemples d'aquest tipus de llistes.

El nom d'una classe d'esdeveniments generats pot estar format pel nom de la classe i el nom de la classe d'esdeveniments generats corresponent, *nomClasseNomEsdGenerat*, o simplement pel nom de la classe d'esdeveniments generats, *NomEsdGenerat*. El primer cas és necessari quan un esdeveniment pot ser generat per objectes de més d'una classe.

Exemple 4.3.4

A la llista d'esdeveniments generats del diagrama d'estats del tipus d'objecte *Acció*, figura 4.1.6, hi ha una única entrada corresponent a la classe d'esdeveniments *comprovaPreu*. D'altra banda, en les postcondicions a l'única entrada de la llista d'esdeveniments rellevants hi ha una referència a la creació d'objectes de tipus *CanviAcció*, que és *newCanviAcció*. Finalment, en l'operació de creació hi ha una referència a la creació d'objectes en el supertipus *Observable*, que és *newObservable*. Així, el conjunt de noms de classes d'esdeveniments generats de la classe d'objectes *acció* és Generacions(acció) = {accióComprovaPreu, newCanviAcció, newAccioObservable}. Cal notar que altres classes d'objectes poden generar esdeveniments de les classes *comprovaPreu* i *newObservable*.

Una altra classe d'objectes, en aquest cas corresponent a un tipus estat, és la classe *accióValorada*. En la definició d'aquest tipus no apareix cap llista d'esdeveniments "Entry:" ni "Exit:". El conjunt de noms és, per tant, Generacions(accióValorada) = {}. En canvi, en l'altra versió del diagrama d'estats del tipus *Acció*, de la figura 4.1.8, el conjunt de noms seria Generacions(accióValorada) = {notificacióCanviPreu}. ♦

Restriccions d'integritat. En la síntesi ens interessa tenir el conjunt dels predicats d'inconsistència corresponents a les restriccions d'integritat definides per a cada classe d'objectes d'un model. Aquest conjunt es coneix amb el nom de *Restriccions(CO)*, on CO és la classe d'objectes.

En Syntropy, les restriccions es troben representades com:

- Invariants de tipus lògic.

- Invariants de propietats. Concretament: "unique" i "optional". Cal notar que "const" no s'inclou com a restricció, ja que la comprovació que el valor d'una propietat no varia durant la vida d'un objecte pot fer-se en el moment de la validació sintàctica del model.

- Restriccions sobre el valor d'una propietat donades en la seva definició.

- Restriccions de multiplicitat en les propietats.

- Restriccions de multiplicitat en les associacions. Aquestes restriccions es consideren definides en el tipus d'objecte origen de l'associació.

- Restriccions entre associacions. Aquestes restriccions les considerarem definides en el tipus d'objecte font de la restricció (vegeu associacions en Syntropy, apartat 4.1.1).

Els invariants "abstract" no corresponen tampoc a restriccions pel mateix motiu que els invariants de propietats "const". És a dir, perquè la comprovació que només es puguin crear objectes en un tipus "abstract" com a conseqüència de la invocació d'operacions de creació d'objectes en els seus subtipus es pot fer en el moment de la validació sintàctica del model.

En Syntropy les restriccions no tenen associat un predicat d'inconsistència. Per tractar amb aquestes restriccions usarem el nom genèric de *restricció* amb un subíndex que permetrà diferenciar-les. Aquests noms genèrics simulen els predicats d'inconsistència per a cada restricció. Podem considerar que es dóna un fet d'un d'aquests predicats quan la restricció que representa es viola a conseqüència de l'ocurrència d'un o més esdeveniments externs.

Exemple 4.3.5

Al sistema exemple trobem les restriccions següents:

- En cada tipus d'objecte *Sector*, *Borsa* i *Vigilant* hi ha definit un invariant del tipus lògic. Només l'invariant del tipus d'objecte *Vigilant* és una restricció, a la qual ens referirem com a $restricció_1$.

- Cap propietat té definit un invariant "optional", això vol dir que per a cada propietat del model tenim una restricció equivalent a *nomPropietat* <> *"nil"*. No considerarem aquestes restriccions per simplificar la síntesi del model del sistema exemple. ♦

Classes d'esdeveniments estructurals

En la síntesi ens interessen els tipus d'esdeveniments estructurals següents:

- Esdeveniments estructurals d'inserció d'objectes en una classe. Inserció(CO) és el nom de la classe d'esdeveniments estructurals d'inserció d'objectes de la classe CO.

- Esdeveniments estructurals d'esborrat d'objectes d'una classe. Esborrat(CO) és el nom de la classe d'esdeveniments estructurals d'esborrat d'objectes de la classe CO.

- Esdeveniments estructurals de modificació del valor d'un atribut dels objectes d'una classe d'objectes. El conjunt de noms de les classes d'esdeveniments estructurals de modificació del valor d'un atribut A és Modificació(A).

- Esdeveniments estructurals de violació de restriccions d'integritat. El nom de la classe d'esdeveniments estructurals de violació d'una restricció d'integritat Ic es Restricció(Ic).

- Esdeveniments estructurals de generació d'esdeveniments. El nom de la classe d'esdeveniments estructurals de generació d'esdeveniments d'una classe G es designa Generació(G).

Ara veurem quines són les classes d'esdeveniments estructurals per a una classe d'objectes en un model en Syntropy.

- La classe d'esdeveniments estructurals d'inserció d'objectes en una classe d'objectes no es defineix explícitament en aquests models. Ens referirem a aquesta classe com *inserir_nomClasse.*

- El mateix passa amb la classe d'esdeveniments estructurals d'esborrat d'objectes d'una classe. Ens referirem a aquesta classe com *esborrar_nomClasse.*

- I tampoc no ho estan les classes d'esdeveniments estructurals de modificació dels valors dels atributs. Segons si un atribut és univaluat o multivaluat hi haurà una classe d'esdeveniments estructurals de modificació o dues. En els univaluats, l'únic tipus de modificació és assignar valors a l'atribut, així, ens referirem a aquesta modificació com *assignar_nomAtribut.* En els multivaluats, en canvi, s'afegeixen nous valors i s'eliminen valors que té l'atribut, així, ens referirem a aquests dos tipus de modificacions com *afegir_nomAtribut* i *eliminar_nomAtribut.*

- Com que hi ha una relació biunívoca entre una restricció d'integritat i la classe d'esdeveniments estructurals de violació corresponent, utilitzarem el mateix nom per a la classe d'esdeveniments estructurals de violació que per al predicat d'inconsistència que fa referència a la restricció.

- Com que hi ha una relació biunívoca entre una classe d'esdeveniments generats i la classe d'esdeveniments estructurals de generació corresponent, utilitzarem el mateix nom per a ambdues.

Exemple 4.3.6

Les classes d'esdeveniments corresponents a la classe d'objectes *acció,* són les següents. Cal tenir en compte que aquesta classe no té definida cap restricció d'integritat.

Inserció(acció) = inserir_acció.
Esborrat(acció) = esborrar_acció.
Modificació(accióPreu) = {assignar_accióPreu}
Modificació(accióCanviAcció) = {afegir_accióCanviAcció,
eliminar_accióCanviAcció}
Modificació(accióSector) = {assignar_accióSector}
Modificació(accióMoviment) = {assignar_accióMoviment}
Modificació(accióVigilant) = {afegir_accióVigilant, eliminar_accióVigilant}
Generació(accióComprovaPreu) = accióComprovaPreu
Generació(newCanviAcció) = newCanviAcció
Generació(newAccioObservable) = newAccioObservable ♦

4.3.2 Síntesi de regles d'esdeveniments

En la síntesi de regles d'esdeveniments s'ha de capturar tot el que sigui rellevant quant a comportament en un model en Syntropy, de manera que els Esquemes d'Interacció que es determinin a partir de les regles no vagin en contra d'allò especificat en el model. Així, serà molt important, en fer la síntesi, tenir en compte el procés d'avaluació d'esdeveniments en Syntropy vist a la secció 4.1, principalment en el cas de la síntesi de classes d'esdeveniments prerequisit.

El primer que veurem és com fer la síntesi de les regles i les seves classes d'esdeveniments causa. Per a una classe d'esdeveniments estructurals deduirem tantes regles d'esdeveniments com causes hi hagi. Un cop vist això, mostrarem com obtenir les classes d'esdeveniments prerequisit per a cadascuna de les regles.

És possible que, en algun cas, no es pugui deduir cap regla que es refereixi a una determinada classe d'esdeveniments estructurals. Això passarà quan al sistema modelat mai no puguin ocórrer esdeveniments estructurals de la classe. És a dir, l'operació primitiva d'actualització representada per l'esdeveniment estructural mai no serà utilitzada al sistema.

Regles i causes

La síntesi de regles i causes es fa de manera diferent segons el canvi que representa la classe d'esdeveniments estructural.

- Regles d'esdeveniments que es refereixen a una classe d'esdeveniments estructurals d'inserció d'objectes en una classe.

Per una classe d'objectes *nomClasse* que correspon a un tipus d'objectes, podem dir que hi haurà una regla que es referirà a la classe d'esdeveniments estructurals *inserir_nomClasse*, per a cada classe d'esdeveniments estructurals de generació d'esdeveniments G que provoqui la invocació d'una operació de creació d'objectes en el tipus d'objecte considerat. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments estructurals G.

Per una classe d'objectes *nomClasse* que correspon a un tipus estat, es pot afirmar que hi haurà una regla que es referirà a la classe d'esdeveniments estructurals *inserir_nomClasse* per a cada classe d'esdeveniments estructurals o externs CE corresponent a un tipus d'esdeveniments que pugui provocar una transició que tingui el tipus estat com a estat destinació i amb estat origen diferent de la destinació. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments CE. Cal tenir en compte que en el cas de transicions d'estat inicial, el tipus dels esdeveniments que provoquen el pas dels objectes a aquest estat, no està, la majoria de vegades, explícit al diagrama d'estats. En aquests casos, la classe d'esdeveniments CE que ha d'agafar-se com a causa és la que correspon al tipus d'esdeveniment que provoca que s'insereixin objectes en el supertipus del tipus estat considerat.

Exemple 4.3.7

La inserció d'objectes del tipus d'objecte *Observable*, Inserció(observable) = inserir_observable, es produeix quan s'invoca l'operació de creació *newObservable*. Com que hi ha tres classes d'esdeveniments estructurals que provoquen la invocació d'aquesta operació, les regles i classes que es referiran a la classe d'esdeveniments estructurals d'inserció *inserir_observable* són:

Regles(inserir_observable) = $\{r_1, r_2, r_3\}$
amb causa(r_1) = {newAccioObservable}
amb causa(r_2) = {newSectorObservable}
amb causa(r_3) = {newBorsaObservable}

En el cas de la inserció d'objectes en el tipus estat *Preparat*, *inserir_vigilantPreparat*, els esdeveniments que provoquen el pas d'objectes del tipus *Vigilant* a l'estat *Preparat* es poden deduir de les dues transicions que apareixen al diagrama d'estats d'aquest tipus amb estat destinació *Preparat*, figura 4.2.5. A partir de la transició amb estat origen *Valorat* i amb estat destinació *Preparat*, deduïm el tipus d'esdeveniment *comprovaPreu*. A partir de la transició d'estat inicial, deduïm el tipus d'esdeveniment

newVigilant. Tenint en compte les classes d'esdeveniments estructurals corresponents a cadascun d'aquests tipus d'esdeveniments, s'obtenen les regles següents:

Regles(inserir_vigilantPreparat) = $\{r_{12}, r_{13}, r_{14}, r_{15}\}$
amb causa(r_{12}) = {newVigilant}
amb causa(r_{13}) = {accióComprovaPreu}
amb causa(r_{14}) = {sectorComprovaPreu}
amb causa(r_{15}) = {borsaComprovaPreu} ♦

- Regles d'esdeveniments que es refereixen a una classe d'esdeveniments estructurals d'esborrat d'objectes d'una classe.

Per una classe d'objectes *nomClasse* que correspon a un tipus d'objectes, hi haurà una regla d'esdeveniments que es referirà a la classe d'esdeveniments estructurals *esborrar_nomClasse,* per a cada classe d'esdeveniments estructurals o externs CE corresponent a un tipus d'esdeveniments que pot provocar una transició amb estat origen un dels estats subtipus directe del tipus d'objecte considerat, i sense un estat destinació. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments CE.

Per una classe d'objectes *nomClasse* que correspon a un tipus estat, hi haurà una regla d'esdeveniments que es referirà a la classe d'esdeveniments estructurals *esborrar_nomClasse,* per a cada classe d'esdeveniments estructurals o externs CE corresponent a un tipus d'esdeveniments que pot provocar una transició que té per origen el tipus estat considerat i amb estat destinació diferent del d'origen. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments CE.

Exemple 4.3.8

Al diagrama d'estats del tipus d'objecte *Vigilant,* figura 4.2.5, només hi ha una transició amb estat origen un dels estats subtipus directe del tipus d'objecte *Vigilant* i sense estat destinació. Es tracta de la transició amb estat origen *Actiu.* Aquesta transició és provocada per esdeveniments del tipus *elimina.* Així, obtenim la regla següent:

Regles(esborrar_vigilant) = $\{r_{20}\}$
amb causa(r_{20}) = {elimina}

Al mateix diagrama d'estats podem veure que hi ha dues transicions que es poden aplicar als objectes en l'estat *Preparat.* La primera transició és la que va de l'estat *Preparat* a l'estat *Activat* i és provocada per esdeveniments del tipus *comprovaPreu.* La

segona és la transició amb estat origen *Actiu* i sense estat destinació, que s'aplica als objectes en qualsevol dels estats imbricats en l'estat *Actiu*, i que és provocada per esdeveniments del tipus *elimina*. Tenint en compte les classes d'esdeveniments externs i estructurals corresponents als tipus d'esdeveniments *comprovaPreu* i *elimina*, podem deduir les regles d'esdeveniments següents:

Regles(esborrar_vigilantPreparat) = $\{r_{22},r_{23},r_{24},r_{25}\}$
amb causa(r_{22}) = {elimina}
amb causa(r_{23}) = {acciócomprovaPreu}
amb causa(r_{24}) = {sectorComprovaPreu}
amb causa(r_{25}) = {borsaComprovaPreu}

Finalment, al diagrama d'estats del tipus d'objecte *Acció*, figura 4.1.6, no hi ha cap transició sense estat destinació. Així, no és possible deduir cap regla d'esdeveniments que es refereixi a la classe d'esdeveniments estructurals *esborrar_acció*. Això vol dir que, tenint en compte el model mai no es podran esborrar objectes del tipus *Acció* del sistema. ♦

- Regles d'esdeveniments que es refereixen a una classe d'esdeveniments estructurals de modificació del valor d'un atribut.

La síntesi es fa de manera diferent segons el lloc del model en què s'especifiquen les modificacions de les propietats d'un tipus d'objectes:

- Si s'especifiquen en les postcondicions d'algun dels elements (transició, entrada d'una llista d'esdeveniments rellevants, operació de creació) en un diagrama d'estats, hi haurà una regla d'esdeveniments per a cada classe d'esdeveniments estructurals o externs CE corresponent a un tipus d'esdeveniments que provoqui la transició, que invoqui la creació o que provoqui l'entrada a la llista d'esdeveniments rellevants en què aparegui la postcondició. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments CE.

- Si s'especifiquen en els invariants de tipus lògic que defineixen el valor de la propietat, hi haurà una regla d'esdeveniments per a cada classe d'esdeveniments estructurals CEs corresponent a una de les propietats del model que aparegui referenciada dins l'invariant (sense tenir en compte la propietat per a la qual l'invariant defineix el valor). Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments estructurals CEs.

- Si s'especifiquen en la definició d'associacions derivades, hi haurà una regla d'esdeveniments per a cada classe d'esdeveniments estructurals CEs corresponent a una

propietat del model que aparegui referenciada dins la definició. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments estructurals CEs.

- Si s'especifiquen en la definició d'operacions de creació de tipus d'objectes que són supertipus directes o indirectes del tipus d'objecte per al qual és definida la propietat, hi haurà una regla d'esdeveniments per a cada classe d'esdeveniments estructurals de generació d'esdeveniments G que provoqui la invocació d'una de les operacions de creació. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments estructurals de generació G.

Exemple 4.3.9

Al model exemple hi ha un únic cas de postcondició en què es modifiqui el valor de la propietat *preu* dels objectes del tipus d'objecte *Acció*, [preu'=p]. Es troba a l'única entrada que hi ha a la llista d'esdeveniments rellevants del diagrama d'estats del tipus d'objecte *Acció*, figura 4.1.6. Així, podem deduir una única regla d'esdeveniments amb una única classe d'esdeveniments causa, que és la classe d'esdeveniments externs, *canviPreu*, que correspon al tipus d'esdeveniment a l'entrada de la llista d'esdeveniments rellevants:

Regles(assignar_accióPreu) = $\{r_{36}\}$
amb causa(r_{36}) = {canviPreu}

El valor de la propietat *moviment* d'un objecte de tipus *Sector* s'especifica, d'una banda, amb l'invariant *moviment = sum acció.moviment* al diagrama de tipus per al tipus d'objectes *Sector*, figura 4.2.1. Tenint en compte que l'única propietat referenciada en l'invariant, a part de la propietat que es defineix, és la propietat *moviment* del tipus d'objecte *Acció*, podem deduir una única regla d'esdeveniments amb classe d'esdeveniments causa *assignar_accióMoviment.*

D'altra banda, la modificació d'aquesta propietat s'especifica en la definició de l'operació de creació d'objectes del tipus *Observable*. Això vol dir que podem deduir una altra regla d'esdeveniments que es refereixi a *assignar_sectorMoviment* amb classe d'esdeveniments causa la classe d'esdeveniments estructurals de generació *newSector*:

Regles(assignar_sectorMoviment) = $\{r_{60}, r_{61}\}$
amb causa(r_{60}) = {assignar_accióMoviment}
amb causa(r_{61}) = {newSector} ♦

En alguns models trobem que el valor d'una de les propietats definides per una associació del model no es defineix explícitament. La raó és que el seu valor es pot

obtenir en funció del valor de l'altre. Es pot entendre aquesta funció com un invariant implícit que defineix el valor d'una propietat a partir del valor de l'altre. Les regles d'esdeveniments que es refereixen a les classes d'esdeveniments estructurals de modificació dels atributs corresponents a aquestes propietats, es deduiran de la mateixa manera que per a qualsevol altra propietat el valor de la qual està definit per un invariant.

Exemple 4.3.10

Exemples d'aquests invariants implícits, al model exemple, ho són els de les propietats corresponents als atributs *canviAccióAcció*, *sectorAcció*, *borsaSector*, *observableVigilant*, *accióVigilant*, *sectorVigilant* i *borsaVigilant*, que ens porten a la definició de les regles r_{37}, r_{58}, r_{59}, r_{64}, r_{65}, r_{34}, r_{35}, r_{41}, r_{42}, r_{62}, r_{63}, r_{69} i r_{70}. Concretant-ho en el cas de la propietat *acció* d'un *Sector*, *sectorAcció*, en el model no s'explícita quin és el valor d'aquesta propietat per a un determinat sector s. Implícitament, sabem que el valor d'aquesta propietat depèn de les accions que tenen s com a valor de la propietat *sector*. A partir d'aquest invariant implícit, podem deduir les regles r_{58} i r_{59}:

Regles(afegir_sectorAcció) = $\{r_{58}\}$
amb causa(r_{58}) = {assignar_accióSector}
Regles(eliminar_sectorAcció) = $\{r_{59}\}$
amb causa(r_{59}) = {assignar_accióSector} ♦

En el cas d'atributs que corresponen a propietats en un tipus estat, l'especificació de modificacions es pot trobar igualment en diferents llocs d'un model. Tot seguit veurem com deduir les regles d'esdeveniments que es referiran a les classes d'esdeveniments estructurals que corresponen a aquestes modificacions segons quin és aquest lloc:

- Si s'especifiquen en les postcondicions de transicions que entren o surten del tipus estat en consideració o que entren o surten d'estats imbricats directament o indirecta el tipus estat en consideració, hi haurà una regla d'esdeveniments per a cada classe d'esdeveniments estructurals o externs CE corresponent a un tipus d'esdeveniments que pot provocar la transició on aparegui la postcondició. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments CE.

- Si s'especifiquen en els invariants de tipus lògic que es defineixin per al tipus estat, la deducció de regles d'esdeveniments es farà tal com hem explicat abans per al cas de la modificació de propietats de tipus d'objectes.

- Si s'especifiquen en la definició d'associacions derivades, la deducció de regles d'esdeveniments es farà tal com hem explicat abans per al cas de la modificació de propietats en tipus d'objectes.

No hem parlat, ni en el cas dels tipus d'objectes ni en el dels tipus estat, del que passa amb les propietats heretades dels seus supertipus. Per al cas dels tipus d'objectes no cal fer cap distinció entre propietats pròpies o heretades, ja que partim de models en què s'han aplicat les regles de conformança de comportament (vegeu l'apartat Determinació d'Esquemes d'Interacció en models en Syntropy, a la secció 4.1). En aquests models s'hauran fet explícits els elements, heretats dels seus supertipus, que descriuen quan i com es modifiquen les propietats d'un tipus d'objectes. Per tant, amb les indicacions donades fins ara es podran obtenir les regles d'esdeveniments tant per a les propietats pròpies com per a les heretades.

En el cas dels tipus estat, no té sentit parlar d'aplicar regles de conformança. Si el valor de la propietat heretada no és explícitament redefinit en el subtipus, existeix un invariant implícit que indica que el valor de la propietat és sempre el mateix valor que el de la propietat en el supertipus. Llavors les regles d'esdeveniments es dedueixen aplicant el que hem vist fins ara per al cas d'invariants explícits. Si el valor és explícitament redefinit, per obtenir les regles d'esdeveniments que es refereixen a les classes d'esdeveniments estructurals de modificació de l'atribut corresponent a la propietat caldrà utilitzar també les indicacions que s'han donat fins ara.

Exemple 4.3.11

Al sistema exemple no hi ha propietats definides per als tipus estat. El que sí que hi ha són tipus estat amb propietats heretades dels seus supertipus. Un exemple d'això és l'atribut *accióValoradaMoviment* corresponent a la propietat *moviment* del tipus estat *Valorada*, heretada del tipus *Acció*, que alhora l'hereta del tipus *Observable*. Com a conseqüència de l'invariant implícit que diu que el valor de la propietat *moviment* d'un objecte en el tipus estat *Valorada* és el valor de la propietat *moviment* del mateix objecte en el tipus d'objecte *Acció*, deduïm la regla r_{54}:

Regles(assignar_accióValoradaMoviment) = $\{r_{54}\}$
amb causa(r_{54}) = {assignar_accióMoviment } ♦

- Regles d'esdeveniments que es refereixen a una classe d'esdeveniments estructurals de generació d'esdeveniments.

Per una classe d'esdeveniments generats G corresponent a un tipus d'esdeveniments generats determinat, hi haurà una regla d'esdeveniments que es referirà a G per a cada classe d'esdeveniments estructurals o externs CE corresponent a un tipus d'esdeveniments que pot provocar una transició, que pot invocar la creació o que pot provocar una de les entrades a la llista d'esdeveniments rellevants en què G apareix com

un dels tipus d'esdeveniments que cal generar. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments CE.

Exemple 4.3.12

Al diagrama d'estats del tipus d'objecte *Acció,* figura 4.1.6, trobem un únic lloc en què esdeveniments del tipus *comprovaPreu* apareixen com a generats. Es tracta de la transició amb estat origen i destinació l'estat *Valorada*. Hi haurà una regla d'esdeveniments amb només una classe d'esdeveniments causa, que és la classe d'esdeveniments externs corresponent al tipus d'esdeveniment que provoca la transició, és a dir el tipus d'esdeveniment *canviPreu.*

Regles(accióComprovaPreu) = $\{r_{96}\}$
amb causa(r_{96}) = {canviPreu} ♦

- Regles d'esdeveniments que es refereixen a una classe d'esdeveniments estructurals de violació d'una restricció d'integritat.

La síntesi d'aquestes de regles d'esdeveniments dependrà de com estiguin representades les restriccions en el model:

- En el cas d'una Ic definida amb un invariant de tipus lògic, hi haurà una regla d'esdeveniments que es referirà a la classe d'esdeveniments estructurals Ic per a cada classe d'esdeveniments estructurals CEs corresponent a una propietat del model que aparegui referenciada a l'invariant. Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments estructurals CEs.

- En el cas d'una Ic definida amb un invariant de propietats del tipus "unique" o "optional", hi haurà una regla d'esdeveniments que es referirà a la classe d'esdeveniments estructurals Ic per a cada classe d'esdeveniments estructurals de modificació del valor d'atributs CEs corresponent a la propietat, en l'invariant en el cas d'"unique" i que no està a l'invariant en el cas d'"optional". Cada regla d'esdeveniments tindrà un conjunt de classes d'esdeveniments causa format per una única classe d'esdeveniments, que serà la classe d'esdeveniments estructurals CEs.

- En el cas d'una Ic definida com una restricció sobre el valor d'una propietat, hi haurà una regla d'esdeveniments que es referirà a la classe d'esdeveniments estructurals Ic per a cada classe d'esdeveniments estructurals de modificació del valor d'un atribut CEs corresponent a la propietat. Cada regla tindrà una única classe d'esdeveniments causa, que serà la classe d'esdeveniments estructurals CEs.

- En el cas d'una Ic definida com una restricció de multiplicitat d'una propietat, la deducció de les regles d'esdeveniments que es referiran a la classe d'esdeveniments

estructurals Ic es fa de la mateixa manera que per a restriccions sobre el valor d'una propietat.

- En el cas d'una Ic representada com una restricció entre associacions, aquesta restricció es pot veure com un invariant de tipus lògic, i la deducció de les regles d'esdeveniments que es referiran a la classe d'esdeveniments estructurals Ic es farà de la mateixa manera que per a les restriccions definides amb un d'aquests invariants.

Exemple 4.3.13

La restricció *restricció*$_1$ ∈ Restriccions(vigilant), *límit* > *0*, està representada en el model com un invariant de tipus lògic. En aquest invariant trobem una única propietat *límit*. Tenint en compte que aquesta propietat té només una classe d'esdeveniments estructural de modificació del seu valor, Modificacions(vigilantLímit) = {assignar_vigilantLímit}, es dedueix la regla d'esdeveniments següent:

Regles(restricció$_1$) = {r_{88}}
amb causa(r_{88}) = {assignar_vigilantLímit} ♦

Exemple 4.3.14

Al sistema exemple no hi ha cap propietat multivaluada sobre la qual s'hagi definit una restricció de multiplicitat. Ara bé, suposem que per a la propietat del tipus d'objecte *Sector*, que defineix l'associació existent entre *Sector* i *Acció*, *sectorAcció*, es defineix una restricció, *restricció*$_i$, on es diu que el nombre d'accions màxim d'un sector és 100. Tenint en compte que l'atribut corresponent a la propietat *acció* té dues classes d'esdeveniments estructurals de modificació del seu valor, Modificacions(sectorAcció) = {afegir_sectorAcció, esborrar_sectorAcció}, obtenim les regles d'esdeveniments següents:

Regles(restricció$_i$) = {r_j,r_k}
amb causa(r_j) = {afegir_sectorAcció}
amb causa(r_k) = {esborrar_sectorAcció}

Cal notar que la regla r_k podria ser eliminada, perquè la propietat només té definit un màxim en la multiplicitat, i no un màxim i un mínim. ♦

Classes d'esdeveniments prerequisit

En estudiar la manera com fer la síntesi de regles d'esdeveniments i les seves classes d'esdeveniments causa hem vist que les regles d'esdeveniments que deduïm a partir d'un model en Syntropy tenen sempre una única classe d'esdeveniments causa. Tot seguit descriurem com trobar el conjunt de classes d'esdeveniments prerequisit d'una regla tenint en compte quina és la seva classe d'esdeveniments causa.

Per fixar el conjunt de classes d'esdeveniments prerequisit d'una regla d'esdeveniments, ens caldrà identificar en quin o quins dels quatre casos que es plantegen a continuació ens trobem. Aquests quatre casos, que ens indiquen quines han de ser les classes d'esdeveniments prerequisit d'una regla d'esdeveniments, s'han identificat a partir de l'estudi del procés d'avaluació d'esdeveniments en Syntropy (secció 4.1).

El primer cas és el d'una regla d'esdeveniments que tingui com a classe d'esdeveniments causa una classe d'esdeveniments externs. Aquesta regla tindrà sempre un conjunt buit de classes d'esdeveniments prerequisit. En un model en Syntropy no hi ha cap manera d'evitar que esdeveniments que provenen de l'entorn del sistema s'enviïn a un objecte d'un cert tipus. Cal recordar que els esdeveniments externs es comuniquen al sistema en un cert ordre (per tant, el sistema els rep d'un en un) i el seu efecte és instantani.

El segon cas és el d'una regla d'esdeveniments que es refereix a una classe d'esdeveniments estructurals *newNomClasse* amb una classe d'esdeveniments causa CE; la classe d'esdeveniments CE és també classe d'esdeveniments causa d'una classe d'esdeveniments estructurals CEs que no és una classe d'esdeveniments estructurals de generació; i tant *newNomClasse* com CEs són classes d'esdeveniments estructurals de la mateixa classe d'objectes.

D'acord amb el procés d'avaluació d'esdeveniments hem d'assegurar que els esdeveniments de la classe CEs no s'envien a altres objectes fins que tots els esdeveniments de la classe *newNomClasse* han estat enviats als objectes que hi puguin estar interessats i totes les conseqüències directes i indirectes d'aquests esdeveniments s'han aplicat. Per assegurar-ho, hem de posar com a classes d'esdeveniments prerequisit de qualsevol regla d'esdeveniments en què CEs és classe d'esdeveniments causa, la classe d'esdeveniments *newNomClasse* i totes aquelles classes d'esdeveniments estructurals amb una regla que s'hi refereix amb classe d'esdeveniments causa directa o indirecta la classe d'esdeveniments *newNomClasse*.

Una representació gràfica d'aquest cas es pot veure a la figura 4.3.1. La classe d'esdeveniments *newNomClasse* i les classes d'esdeveniments estructurals CEs_1 i CEs_2 per a les quals *newNomClasse* és causa directa i indirecta, respectivament, hauran d'estar en el conjunt de classes d'esdeveniments prerequisit de qualsevol regla en què CEs sigui classe d'esdeveniments causa, concretament en el conjunt de classes d'esdeveniments prerequisit de la regla r_t.

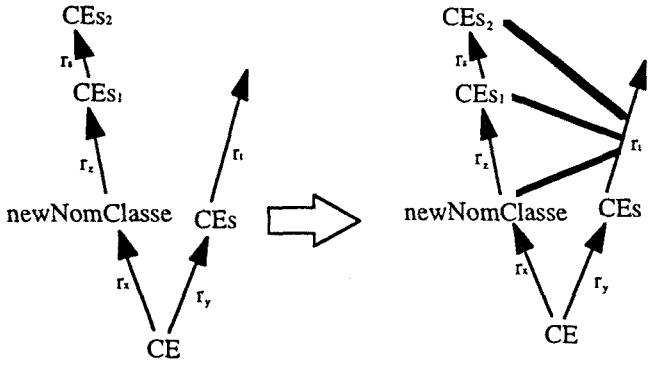


Figura 4.3.1 Síntesi de classes d'esdeveniments prerequisit

Exemple 4.3.15

Agafem el cas de la regla r_{99} que es refereix a la classe *newCanviAcció* i té com a classe d'esdeveniments causa *canviPreu*. La classe d'esdeveniments *canviPreu* és també classe d'esdeveniments causa de la classe d'esdeveniments estructurals *assignar_accióPreu*, i ambdues, *newCanviAcció* i *assignar_accióPreu*, són classes d'esdeveniments estructurals de la classe d'objectes *acció*.

Per assegurar que els esdeveniments que ocorrin de la classe *assignar_accióPreu* no siguin enviats fins que els esdeveniments que ocorrin de la classe *newCanviAcció* no hagin estat enviats i les seves conseqüències directes i indirectes no hagin estat aplicades, hem de posar com a classes d'esdeveniments prerequisit de totes les regles amb classe d'esdeveniments causa *assignar_accióPreu* les classes d'esdeveniments estructurals següents: *newCanviAcció*, *inserir_canviAcció*, *assignar_canviAccióPreu*, *assignar_data* i *assignar_hora*. Aquest és el cas de la regla r_{43}:

Regles(assignar_accióIncialPreu) = $\{r_{43}\}$
amb causa(r_{43}) = {assignar_accióPreu}
amb prerequisit(r_{43}) = {...,newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora, ...} ♦

El tercer cas és el d'una regla d'esdeveniments que es refereix a una classe d'esdeveniments estructurals de generació G que no és del tipus newNomClasse amb una certa classe d'esdeveniments causa CE; la classe d'esdeveniments CE és també classe d'esdeveniments causa d'una classe d'esdeveniments estructurals CEs que o bé és una classe d'esdeveniments estructurals de generació del tipus newNomClasse o bé no és una classe d'esdeveniments estructurals de generació.

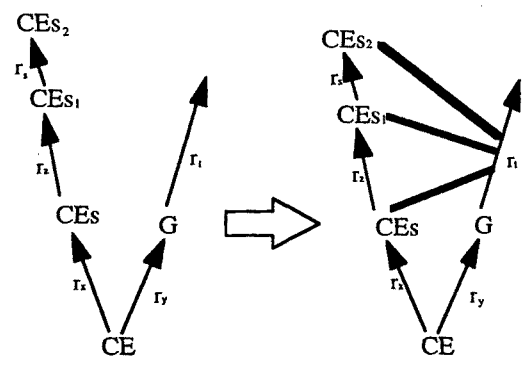


Figura 4.3.2 Síntesi de classes d'esdeveniments prerequisit

D'acord amb el procés d'avaluació d'esdeveniments hem d'assegurar que els esdeveniments de la classe G no s'envien a altres objectes fins que totes les postcondicions definides en el model per al tipus d'esdeveniment corresponent a la classe d'esdeveniments estructurals CE han estat aplicades, i també han estat aplicades totes les seves conseqüències indirectes. Per assegurar-ho, hem de posar com a classes d'esdeveniments prerequisit de qualsevol regla d'esdeveniments per a la qual G és classe d'esdeveniments causa, la classe d'esdeveniments CEs i totes aquelles classes d'esdeveniments estructurals amb una regla que s'hi refereix amb classe d'esdeveniments causa directa o indirecta la classe d'esdeveniments CEs.

Una representació gràfica d'aquest cas es pot veure a la figura 4.3.2. La classe d'esdeveniments CEs i les classes d'esdeveniments estructurals CEs_1 i CEs_2 per a les quals CEs és causa directa i indirecta, respectivament, hauran d'estar en el conjunt de classes d'esdeveniments prerequisit de qualsevol regla en què G sigui classe d'esdeveniments causa, concretament en el conjunt de classes d'esdeveniments prerequisit de la regla r_t.

Exemple 4.3.16

Agafem el cas de la regla r_{96} que es refereix a la classe d'esdeveniments estructurals de generació *accióComprovaPreu.* La classe d'esdeveniments causa d'aquesta regla és *canviPreu* i aquesta classe és també classe d'esdeveniments causa de la regla r_{39} que es refereix a la classe d'esdeveniments estructurals *assignar_accióMoviment.*

Per assegurar que els esdeveniments que ocorrin de la classe *accióComprovaPreu* no siguin enviats fins que les postcondicions definides en el model per *canviPreu* s'hagin aplicat, i també s'hagin aplicat totes les seves conseqüències indirectes, hem de posar com a classes d'esdeveniments prerequisit de totes les regles amb classe

d'esdeveniments causa *acciócomprovaPreu* les classes d'esdeveniments estructurals següents: *assignar_accióMoviment*, *assignar_acciówaloradaMoviment*, *assignar_acciólnicialMoviment*, *assignar_sectorMoviment* i *assignar_borsaMoviment*. Aquest és el cas de la regla r_{23}:

Regles(eliminar_vigilantPreparat) = $\{r_{23}\}$
amb causa(r_{23}) = {acciócomprovaPreu}
amb prerequisit(r_{23}) = {...,assignar_acciómoviment,
assignar_acciówaloradaMoviment, assignar_acciólnicialMoviment,
assignar_sectorMoviment, assignar_borsaMoviment, ...} ♦

El quart i últim cas és el de dues regles d'esdeveniments que es refereixen a dues classe d'esdeveniments estructurals de generació G_1 i G_2 amb una mateixa classe d'esdeveniments causa CE; ni G_1 ni G_2 són del tipus newNomClasse; tant G_1 com G_2 són classes d'esdeveniments estructurals de generació de la mateixa classe d'objecte; i, l'ordre en que s'han de generar segons el model és, primer els esdeveniments de la classe G_1 i després els de la classe G_2.

D'acord amb el procés d'avaluació d'esdeveniments hem d'assegurar que els esdeveniments de la classe G_2 no s'envien a altres objectes fins que tots els generats de G_1 han estat enviats, totes les conseqüències directes i indirectes d'aquests esdeveniments s'han aplicat, els esdeveniments generats directament o indirectament per l'ocurrència d'esdeveniments de la classe G_1 han estat enviats i també han estat aplicades totes les seves conseqüències directes i indirectes. Per assegurar-ho, hem de posar com a classes d'esdeveniments prerequisit de qualsevol regla d'esdeveniments per a la qual G_2 és classe d'esdeveniments causa, la classe d'esdeveniments G_1 i totes aquelles classes d'esdeveniments estructurals amb una regla que s'hi refereix amb classe d'esdeveniments causa directa o indirecta la classe d'esdeveniments G_1.

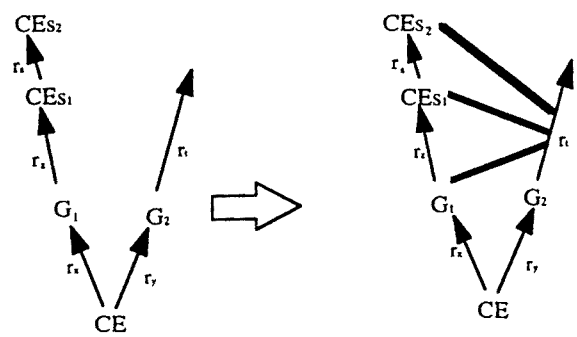


Figura 4.3.3 Síntesi de classes d'esdeveniments prerequisit

Una representació gràfica d'aquest cas es pot veure a la figura 4.3.3. La classe d'esdeveniments G_1 i les classes d'esdeveniments estructurals CEs_1 i CEs_2 per a les quals G_1 és causa directa i indirecta, respectivament, hauran d'estar en el conjunt de classes d'esdeveniments prerequisit de qualsevol regla en què G_2 sigui classe d'esdeveniments causa, concretament en el conjunt de classes d'esdeveniments prerequisit de la regla r_t.

Exemple 4.3.17

En el model del nostre sistema exemple no existeix cap cas de dues regles d'esdeveniments que compleixin aquestes condicions. ♦

A l'annex que hi ha al final d'aquest capítol es pot trobar la síntesi d'elements completa per al cas del model exemple.

Simplificació del conjunt de regles

El conjunt de regles que s'obté de fer la síntesi d'un model es pot simplificar, en alguns casos, reduint el nombre de regles. Concretament, es poden eliminar aquelles regles en què alguna de les seves classes d'esdeveniments causa no té cap regla que s'hi refereix. La raó és que podem afirmar que amb aquestes regles mai no es produiran esdeveniments de les classes a què es refereixen.

Exemple 4.3.18

Al sistema exemple això passa amb les regles r_{45}, r_{52} i r_{75}. Totes tres tenen una mateixa classe d'esdeveniments causa, *eliminar_acciócanviAcció*. Aquesta classe d'esdeveniments estructurals no té cap regla que s'hi refereixi. Podem afirmar, per tant, que mai no es produiran esdeveniments de les classes *eliminar_acciólnicialCanviAcció*, *eliminar_acciówaloradaCanviAcció* i *assignar_canviAcciówcció* mitjançant les regles r_{45}, r_{52} i r_{75}.

4.4 Determinació d'Esquemes d'Interacció per al cas del model exemple

En aquesta secció veurem els Esquemes d'Interacció que s'obtindrien aplicant el procediment de determinació d'un Esquema d'Interacció vàlid, presentat al capítol 3, per a tres tipus de transaccions corresponents al model exemple de la secció 4.2. Cal recordar que aquest procediment només determina un únic Esquema vàlid dels molts possibles. Concretament, determina l'Esquema d'Interacció vàlid en què els tipus d'interaccions estan distribuïdes en el nombre mínim possible d'estats.

D'altra banda, cal fer notar que els tipus de transaccions que aquí prendrem com a exemples no contenen en cap cas més d'una classe d'esdeveniments externs. La raó és que, tal com hem explicat en parlar del procés d'avaluació d'esdeveniments en Syntropy, en els models escrits en aquest llenguatge se suposa que no ocorreran dos esdeveniments externs en un mateix instant, sinó que sempre hi haurà un ordre d'ocurrència entre si.

Començarem primer veient el cas de les transaccions del tipus TTR = {crearAcció}. L'Esquema d'Interacció que obtenim és el que trobem gràficament representat a la figura 4.4.1, i que està format pels tipus d'interaccions següents:

EI = {tipusInteracció(crearAcció,mercatBorsari,1),
tipusInteracció(newAcció,acció,2),
tipusInteracció(newAcció,accióInicial,2),
tipusInteracció(newAccioObservable,observable,3),
tipusInteracció(assignar_accióMoviment,sector,4),
tipusInteracció(assignar_accióSector,sector,4),
tipusInteracció(assignar_accióSector,accióInicial,4),
tipusInteracció(assignar_accióMoviment,accióInicial,4),
tipusInteracció(assignar_accióSector,accióValorada,4),
tipusInteracció(assignar_accióMoviment,accióValorada,4),
tipusInteracció(assignar_accióMoviment,observable,4),
tipusInteracció(assignar_sectorMoviment,borsa,5),
tipusInteracció(assignar_sectorMoviment,observable,5),
tipusInteracció(assignar_borsaMoviment,observable,6)}

En conèixer aquest Esquema d'Interacció ens adonem que una transacció que contingui un esdeveniment extern del tipus *crearAcció* pot arribar a modificar l'estat d'objectes de les classes *mercatBorsari*, *acció*, *borsa*, *observable*, *sector*, *accióInicial* i *accióValorada*. En canvi, en cap cas podrà modificar l'estat d'objectes de les classes *vigilant*, *vigilantActiu*, *vigilantPreparat* i *vigilantActivat*. També ens adonem que com que el *crearAcció* afegirà una nova acció amb un nou valor per a la propietat *moviment* això farà que el valor d'aquesta propietat en el *sector* a què pertany l'*acció* estigui interessat a saber el nou valor de la propietat i que la *borsa* a què pertany el *sector* estigui interessat a saber el nou valor de la propietat *moviment* del *sector*. En el cas que el dissenyador observés un tipus d'interacció que el sorprengués o trobés a faltar-hi un tipus d'interacció, hauria d'anar a estudiar el model i comprovar si és conseqüència d'un error en el model o bé d'alguna cosa que no havia tingut en compte.

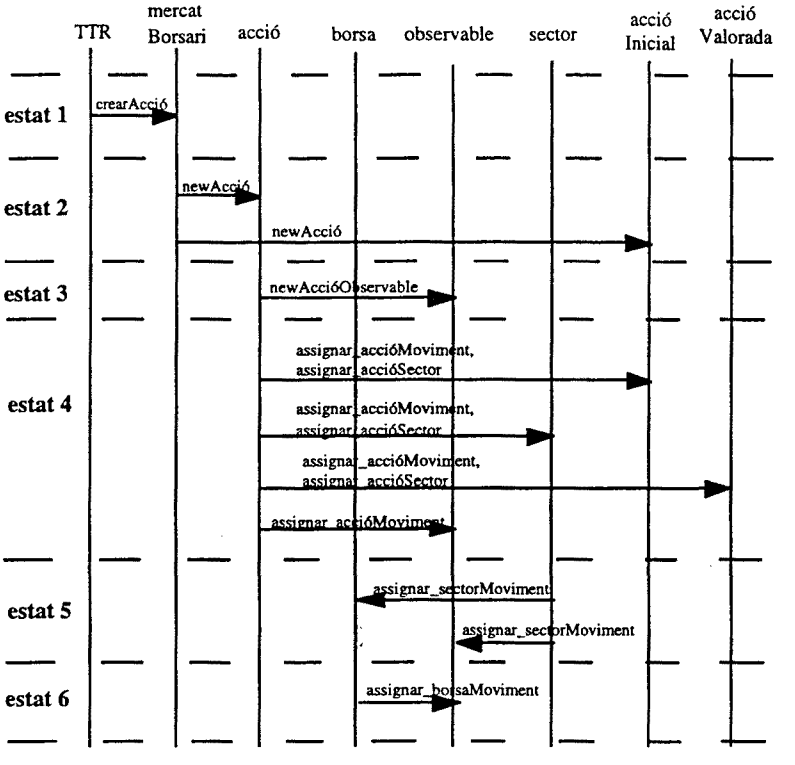


Figura 4.4.1 Esquema d'Interacció TTR = {crearAcció}

Ara veurem el cas de les transaccions del tipus TTR = {crearSector}. L'Esquema d'Interacció que obtenim és el que trobem gràficament representat a la figura 4.4.2, i que està format pels tipus d'interaccions següents:

EI = {tipusInteracció(crearSector, mercatBorsari,1),
tipusInteracció(newSector,sector,2),
tipusInteracció(newSectorObservable,observable,3),
tipusInteracció(assignar_sectorMoviment,borsa,4),
tipusInteracció(assignar_sectorMoviment,observable,4),
tipusInteracció(assignar_sectorBorsa,borsa,4),
tipusInteracció(assignar_borsaMoviment,observable,5)}

D'aquest Esquema d'Interacció podem destacar el tipus d'interacció tipusInteracció(newSectorObservable,observable,3), aquest tipus d'interacció assenyala que, quan es crea un nou objecte a la classe *sector* es genera un esdeveniment de la

classe newSectorObservable que podrà afectar a un o més objectes de la classe *observable*.

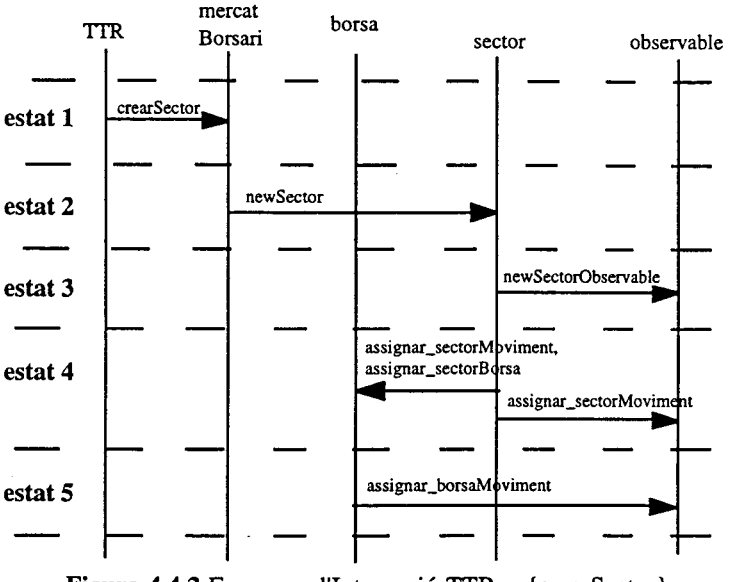


Figura 4.4.2 Esquema d'Interacció TTR = {crearSector}

Suposem per acabar, el cas de les transaccions del tipus TTR = {canviPreu}. L'Esquema d'Interacció que obtenim és el que trobem gràficament representat a la figura 4.4.3, i que està format pels tipus d'interaccions següents:

EI = {tipusInteracció(canviPreu,borsa, 1),
tipusInteracció(canviPreu,sector,1),
tipusInteracció(canviPreu,acció,1),
tipusInteracció(canviPreu,accióValorada,1),
tipusInteracció(canviPreu,accióInicial,1),
tipusInteracció(newCanviAcció,canviAcció,2),
tipusInteracció(afegir_accióCanviAcció,canviAcció,3),
tipusInteracció(assignar_acciómoviment,sector,3),
tipusInteracció(afegir_accióCanviAcció,accióInicial,3),
tipusInteracció(assignar_acciópreu,accióInicial,3),
tipusInteracció(assignar_acciómoviment,accióInicial,3),
tipusInteracció(afegir_accióCanviAcció,accióValorada,3),
tipusInteracció(assignar_acciómoviment,accióValorada,3),
tipusInteracció(assignar_acciópreu,accióValorada,3),
tipusInteracció(assignar_acciómoviment,observable,3),

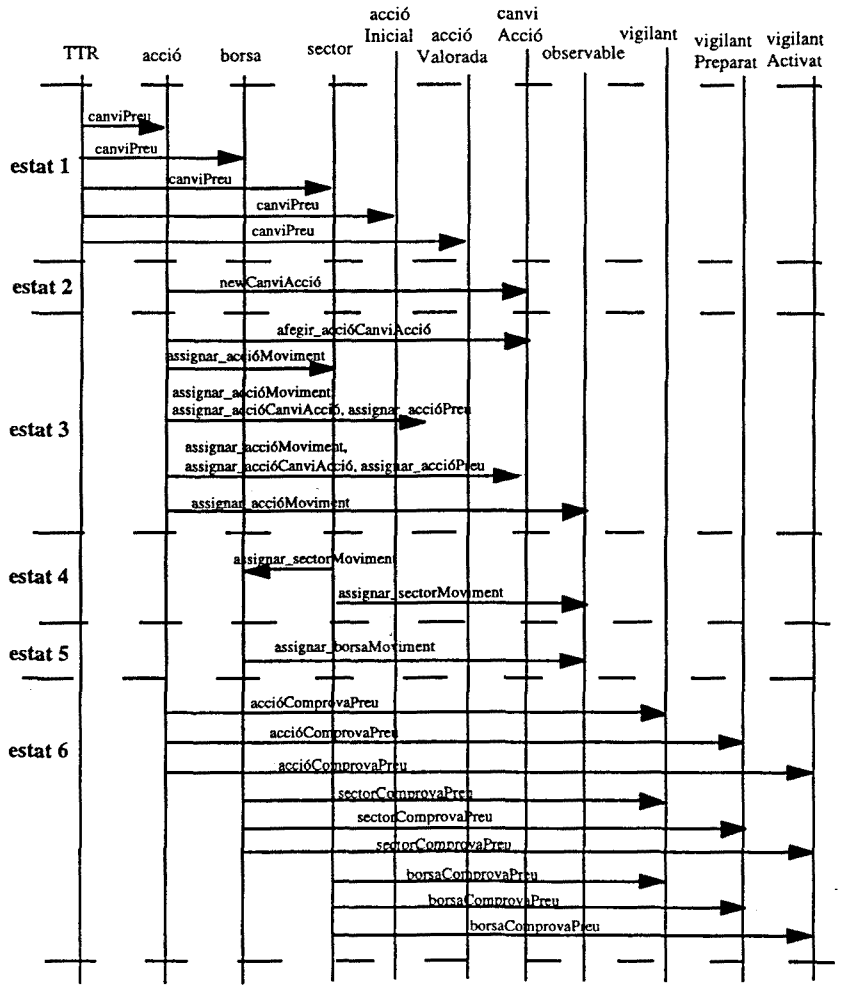


Figura 4.4.2 Esquema d'Interacció TTR = {canviPreu}

tipusInteracció(assignar_sectorMoviment,borsa,4),
tipusInteracció(assignar_sectorMoviment,observable,4),
tipusInteracció(assignar_borsaMoviment,observable,5),
tipusInteracció(accióComprovaPreu,vigilant,6),
tipusInteracció(sectorComprovaPreu,vigilant,6),
tipusInteracció(borsaComprovaPreu,vigilant,6),
tipusInteracció(accióComprovaPreu,vigilantPreparat,6),
tipusInteracció(borsaComprovaPreu,vigilantPreparat,6),
tipusInteracció(sectorComprovaPreu,vigilantPreparat,6),

tipusInteracció(accióComprovaPreu,vigilantActivat,6),
tipusInteracció(borsaComprovaPreu,vigilantActivat,6),
tipusInteracció(sectorComprovaPreu,vigilantActivat,6)}

En aquest Esquema d'Interacció es pot veure que els esdeveniments de les classes *accióComprovaPreu*, *sectorComprovaPreu* i *borsaComprovaPreu* no s'envien als objectes de les classes als que poden afectar fins que en aquests objectes ja s'han aplicat els canvis a l'atribut *moviment* provocats pel tipus de transacció.