UML (II): el model dinàmic i d'implementació

Benet Campderrich Falgueras
Recerca Informàtica, SL

P00/05007/00301

UOC
www.uoc.edu

Índex

Introducció

Hem vist que el model estàtic d'UML consisteix essencialment en un sol diagrama, el de classes, que s'ha d'elaborar en tot projecte de programari. El model dinàmic i d'implementació, en canvi, comprèn diversos diagrames, en part relacionats entre si i també amb el diagrama estàtic, i alguns d'aquests diagrames només es fan servir en determinats casos.

Els aspectes dinàmics o de comportament del sistema es poden modelar amb els diagrames següents: el diagrama de casos d'ús, el diagrama d'estats i transicions, el diagrama d'activitat i els diagrames d'interacció, que són el de seqüències i el de col·laboració. La implementació es pot modelar amb el diagrama de components i el de desplegament.

Objectius

L'objectiu principal d'aquest mòdul és conèixer la notació i la semàntica de les eines diagramàtiques que UML ofereix per a modelar els aspectes dinàmics i d'implementació d'un programari que es vol desenvolupar de manera orientada a objectes, i saber-les aplicar. Aquest objectiu general es descompon en aquests altres:

1. Aprendre el diagrama d'estats.

2. Adquirir els conceptes de cas d'ús i d'actor i saber identificar els diferents casos de relacions entre casos d'ús.

3. Aprendre a descriure les interaccions mitjançant diagrames de col·laboració i de seqüència i saber-los aplicar a la descripció de casos d'ús i d'operacions complexes.

4. Aprendre què és un diagrama d'activitats i entendre les diferències entre aquest i el diagrama d'estats.

5. Aprendre a interpretar i fer servir els diagrames de components i de desplegament.

1. El diagrama d'estats

A vegades hi ha objectes el comportament dels quals pot variar al llarg del temps; quan passa això es diu que l'objecte té estats. Hi ha alguns tipus d'aplicacions, com les de temps real, per a les quals el modelatge d'estats és especialment important.

Exemples d'estats

Hi ha objectes als quals no se'ls pot demanar qualsevol de les seves operacions en qualsevol moment, o bé objectes per als quals algun dels seus atributs només pot tenir un valor no nul en circumstàncies determinades.

Informació redundant, però aclaridora

La informació que conté el diagrama d'estats és essencialment redundant, ja que els canvis d'estat són resultat de la dinàmica del sistema, i, per tant, de l'execució de les operacions de la mateixa classe o d'altres; però tot i així representa un altre punt de vista sobre la dinàmica d'una part del sistema que pot contribuir decisivament a comprendre-la millor.

En el **diagrama d'estats** o **diagrama dinàmic** hem de distingir el següent:

- Les situacions diferents en què es pot trobar un objecte (els estats).
- Quins canvis d'estat són possibles (transicions).
- Quin fet els produeix (esdeveniments).

En les especificacions d'UML es parla de *màquines d'estats* per a distingir-les dels *diagrames d'estat clàssics* o *de Harel*, amb els quals hi ha diferències; tanmateix nosaltres farem servir el terme *diagrama d'estats* perquè sempre representem les màquines d'estat en forma de diagrama; no hi ha confusió possible, ja que en aquesta assignatura no es tractarà dels diagrames de Harel.

El diagrama d'estats es fa servir normalment per a descriure objectes del domini de l'usuari i es documenta normalment en l'etapa d'anàlisi.

1.1. Conceptes bàsics

A continuació explicarem un seguit de conceptes bàsics:

1) Un **estat** és una situació determinada dins la vida d'un objecte o la durada d'una interacció durant la qual compleix alguna condició, duu a terme alguna acció o espera que es produeixi un esdeveniment. Un estat no correspon a un instant en el temps, sinó que l'objecte o interacció hi roman un temps finit.

Vegeu les interaccions en l'apartat 3 d'aquest mòdul didàctic.

2) Una **transició simple** consisteix en el fet que l'objecte o interacció passa d'un estat (**estat d'origen**) a un altre (**estat de destinació**), que podria tornar a ser el mateix. En el cas més general, aquest pas s'engega quan es produeix un esdeveniment determinat i alhora es compleix una condició especificada (**condició de guarda**); llavors es poden executar unes accions i es poden enviar missatges a objectes individuals o a conjunts d'objectes.

Un estat pot tenir **transicions d'arribada**, una de les quals serà l'estat de destinació, i **transicions de sortida**, una de les quals serà l'estat d'origen.

3) Les **transicions internes** són pseudotransaccions en què no hi ha canvi d'estat; serveixen per a especificar accions que s'han d'executar en resposta a un esdeveniment que no provoca cap canvi d'estat en l'objecte o interacció en qüestió.

Observació

Convé no confondre una transició interna amb una *autotransició*, que és una transició ordinària en la qual l'estat d'origen i l'estat de destinació són el mateix.

4) Una **acció** és l'especificació d'un procés atòmic, és a dir, que o bé no s'executa o bé s'executa fins al final. A conseqüència d'una transició es pot executar una acció o més. Una acció es pot descriure mitjançant un procediment o una màquina d'estats.

5) Els objectes, per mitjà de missatges, poden rebre o bé peticions d'operacions o bé senyals. Els **senyals**, a diferència de les operacions, no fan cap procés i l'únic efecte directe que poden tenir és produir esdeveniments (els quals sí que poden provocar transicions i la consegüent execució d'accions).

6) Els **esdeveniments*** són fets que quan es produeixen poden provocar transicions d'un estat a un altre en objectes i interaccions i/o l'execució de determinades accions. Un esdeveniment no va lligat a cap objecte o classe en particular, sinó que és un fet que pot afectar en general els elements del paquet dins el qual està definit. Cada esdeveniment té un nom que l'identifica dins el paquet i pot tenir paràmetres.

*** En anglès, *events*.**

Normalment, un esdeveniment ha de ser tractat* en el moment en què es produeix; si no es provoca la transició perquè l'objecte o interacció no està en l'estat corresponent, es perd l'esdeveniment a menys que es declari com a **esdeveniment diferit**.

*** Per exemple, si es provoca una transició.**

Tipus d'esdeveniments

Ara veurem diferents tipus d'esdeveniments:

- **De crida**: es produeixen quan es crida una operació de l'objecte al qual correspon el diagrama.
- **De senyal**: representen la recepció d'un senyal per l'objecte al qual correspon el diagrama.
- **De canvi**: representen una notificació que una condició ha esdevingut certa. Aquesta condició no s'ha de confondre amb una condició de guarda, ja que una guarda s'avalua un cop s'ha presentat l'esdeveniment corresponent a una transició per a determinar si es provoca la transició o no, mentre que aquesta condició produeix l'esdeveniment quan esdevé certa.
- **De temps:** representen la notificació que o bé ha passat un període de temps des que s'ha produït un esdeveniment determinat (per exemple, que s'ha entrat en l'estat d'origen de la transició), o que és una hora determinada.

Esdeveniments interns

Són pseudoesdeveniments, ja que estan lligats a un estat en comptes d'una transició i serveixen per a engegar accions que no van associades a cap canvi d'estat concret. Hi ha tres tipus d'esdeveniments interns:

1) D'entrada: es produeixen quan l'objecte entra en l'estat corresponent. No tenen ni guarda ni paràmetres, perquè es cridaran implícitament quan hi hagi una transició d'entrada a l'estat (inclosa una autotransició), però no pas quan hi hagi una transició interna, ja que llavors no hi haurà canvi d'estat. S'especifiquen explícitament i tenen com a nom la paraula clau *entry*.

2) De sortida: són anàlegs als d'entrada, llevat que es produeixen a la sortida de l'estat. S'especifiquen explícitament i tenen com a nom la paraula clau *exit*.

3) D'acció: especifiquen accions que s'executen quan s'arriba a l'estat en qüestió i acaben o bé per si mateixes o bé quan se surt de l'estat. S'especifiquen explícitament i tenen com a nom la paraula clau *do*.

1.2. Notacions bàsiques

La representació més senzilla d'un estat és un rectangle amb els vèrtexs arrodonits, tal com podem veure en la figura següent:

A la pràctica, és imprescindible que cada estat tingui un nom i que aquest nom no es repeteixi en cap altre estat del mateix diagrama; això darrer no impedeix que l'objecte o interacció pugui arribar al mateix estat diverses vegades al llarg de la seva vida.

Les transicions es representen per fletxes de punta plena que van de l'estat de sortida a l'estat d'arribada. Amb la fletxa hi ha una expressió (anomenada **cadena de la transició**) que té la sintaxi formal següent:

signatura '[' guarda ']' '/' acció '^' tramesa

Tant d'accions com de trameses, n'hi pot haver diverses o no haver-n'hi cap, i poden estar barrejades; l'ordre en què apareixen és l'ordre en què s'han d'executar.

A continuació veurem una explicació de cadascun dels elements de la cadena de transició:

- **Signatura**: té un format dependent del tipus d'esdeveniment; en el cas d'un esdeveniment de crida o de senyal, es defineix així:

  nom_esdeveniment '(' nom_paràmetre ':' expressió_tipus ',' ... ')'

  Si l'esdeveniment és de temps, la signatura té una d'aquestes formes:

  'after(' expressió_de_temps ')'

  on *expressió_de_temps* consisteix en una durada a partir d'un origen, o bé:

  'when(' hora o data ')'

  Finalment, si l'esdeveniment és de canvi tindrem el següent:

  'when(' expressió_booleana ')'

- **Guarda**: és una expressió que pot prendre el valor *cert* o *fals*, escrita en pseudocodi o en el llenguatge de programació que es fa servir.

- **Acció**: és l'especificació d'una acció, en pseudocodi o en el llenguatge de programació que es fa servir. En el cas d'un esdeveniment diferit, cal que hi hagi la paraula clau *defer* com a primera acció.

- **Tramesa**. Aquest element té la forma següent:

  destinació '.' missatge '(' argument ',' ... ')'

en què *destinació* ha d'identificar un objecte o grup d'objectes i *missatge* pot ser una operació de l'objecte de destinació o bé un senyal.

Els senyals es poden definir com a classes amb l'estereotip *signal*; no tenen operacions i en el compartiment dels atributs tenen els paràmetres del senyal; poden constituir jerarquies d'herència, però no poden tenir relacions de cap mena amb classes. Els senyals als quals han de ser sensibles els objectes d'una classe es poden especificar en un compartiment addicional del símbol de la classe.

Nota terminològica

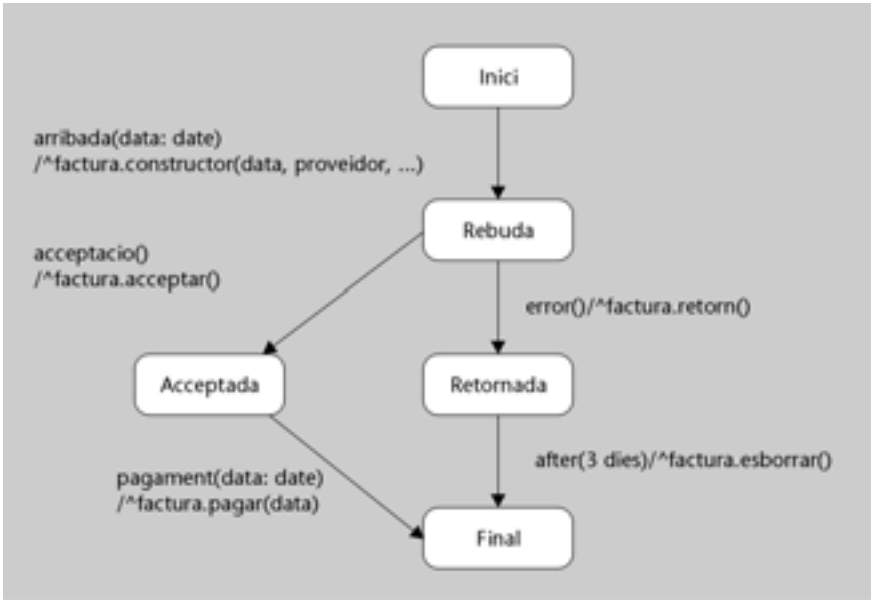
Respecte a l'ús dels termes *paràmetre* i *argument* en relació amb una crida, cal tenir present que *paràmetre* correspon al punt de vista de la cosa cridada i *argument*, al punt de vista d'allò que fa la crida.

Exemples de diagrames d'estat

En els exemples següents veurem diagrames d'estats amb transicions, estats, esdeveniments, autotransició, etc.

1) Exemple d'estats, transicions i esdeveniments

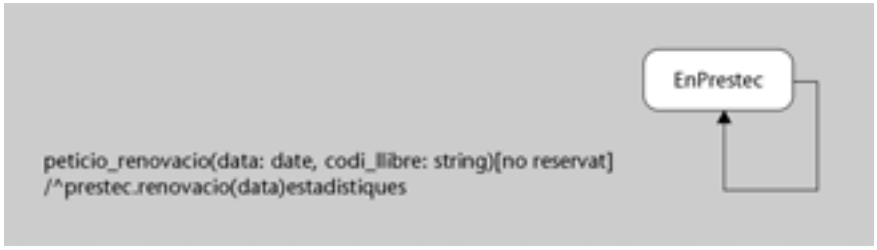
Aquest diagrama correspon a una classe de factures, el nom de la qual no hi figura:



Tot comença quan arriba una factura, fet que descrivim com l'esdeveniment *arribada*, el qual té com a paràmetre la data; aquest esdeveniment demana l'operació constructor de la classe en qüestió en relació amb un objecte de la classe que és la factura arribada (identificada per factura) i provoca la transició cap a l'estat *Rebuda*; *data*, i *proveidor* són paràmetres de *constructor*. Després la factura és verificada (això no apareix explícitament), cosa que pot donar com a resultat dos esdeveniments alternatius: *acceptació* –que demana l'operació *acceptar* i provoca la transició cap a l'estat *Acceptada*–, i *error* –que demana l'operació retorn i provoca la transició a l'estat *Retornada*. Si la factura està en l'estat *Acceptada* i es produeix l'esdeveniment *pagament*, la factura passa a l'estat *Final* i es demana l'execució de l'operació *pagar* en relació amb la factura en qüestió. Si la factura està en l'estat *Retornada*, la transició cap a l'estat *Final* és provocada per un esdeveniment de temps consistent en el fet que hagin passat tres dies (com que no s'especifica des de quan, s'entén que és des de l'entrada en l'estat d'origen), i aleshores es demana l'operació *esborrar*.

2) Exemple d'autotransició, acció i guarda

El diagrama següent representa part del diagrama d'estats dels objectes d'una classe de llibres d'una biblioteca pública:



Quan un lector té un llibre en préstec pot demanar renovacions del préstec, que li seran concedides si no és que el llibre està reservat per a un altre lector. Els efectes de l'esdeveniment *peticio_renovacio* són l'autotransició i, en aquest ordre, la crida a l'operació *renovacio* i l'execució de l'acció *estadistiques*.

3) Exemple d'esdeveniment de senyal

Un lector té un llibre en préstec; quan el torna, si un altre lector el té reservat, s'envia un missatge a l'objecte *reserva* amb el senyal tornat, que ha d'estar definit dins el compartiment de senyals de la classe a què pertany *reserva*.

1.3. Transicions complexes

Un objecte o interacció pot estar en més d'un estat al mateix temps, i hi pot haver transicions que surtin de més d'un, estat o vagin a parar a més d'un, o totes dues coses alhora; són les anomenades **transicions complexes**.

Una transició complexa amb diversos estats d'origen només té lloc si l'objecte o interacció està en tots aquests alhora i, a més, es produeix l'esdeveniment corresponent a la transició (i es compleix la guarda, si n'hi ha) i, per tant, fa una funció de sincronització. Anàlogament, quan es produeix una transició complexa amb diversos estats de destinació l'objecte o interacció passa a tots aquests estats alhora.

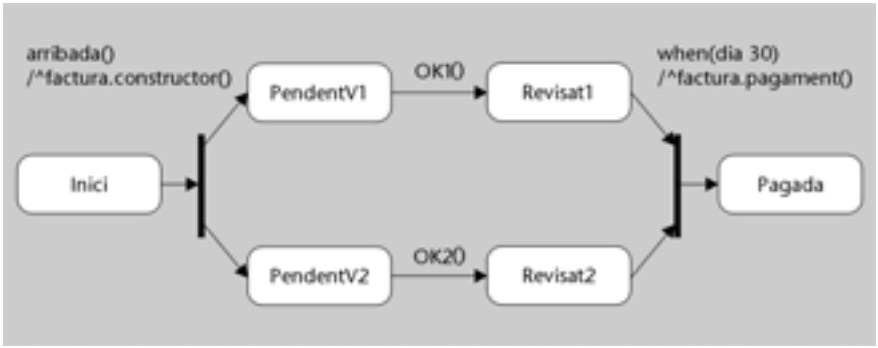
Una transició complexa es representa fent servir un pseudoestat intermedi; a aquest pseudoestat van a parar diverses transicions (pseudoestat de sincronització, *join pseudostate*) o en surten diverses (pseudoestat de bifurcació, *fork pseudostate*), o totes dues coses alhora. Aquest tipus de pseudoestat es representa per mitjà d'una barra vertical curta i gruixuda que és origen i destinació de transicions en què es descompon la transició composta.

Pseudoestat

Un pseudoestat és un símbol que figura en una posició del diagrama on normalment hi hauria un estat i no representa cap concepte, sinó que té una funció purament gràfica. En el subapartat 1.4, veurem els altres tipus de pseudoestats.

Exemple de transicions complexes

Considereu les transicions complexes següents:



Quan arriba una factura d'una compra de material, passa a estar pendent de la verificació comparant-la amb la comanda (V1) i amb el que s'ha rebut (V2), per mitjà d'una transició complexa de bifurcació; si cada verificació és conforme (esdeveniments OK1 i OK2) es passa a l'estat *Revisat1* i *Revisat2*, respectivament. Quan arriba el dia 30, es paga la factura només si estava alhora en aquests dos estats, per mitjà d'una transició complexa de sincronització.

1.4. Estats compostos

Hem vist que mentre un objecte o interacció roman en un estat, es troba en una situació determinada; però de vegades aquesta situació és genèrica, i mentre s'hi està –i només en aquest cas– pot estar en diferents moments en allò que podríem considerar variants o matisos de la situació genèrica.

Una situació genèrica com aquesta es representa mitjançant un **estat compost**, dins el qual hi ha diversos **subestats** possibles, cadascun dels quals pot ser un estat compost o no. Els estats no compostos –tots els que hem considerat fins ara– es diuen **estats simples**.

A un estat compost correspon un **diagrama de subestats.** Els subestats d'un estat poden ser concurrents (és a dir, que es poden presentar simultàniament), o seqüencials (incompatibles entre si); en el cas més general, el diagrama de subestats consta de diverses seqüències de subestats en paral·lel; cada seqüència comença en un pseudoestat inicial i acaba en un pseudoestat final.

Un estat compost es representa pel mateix símbol que un estat simple, però a dintre hi haurà almenys dos compartiments separats per una línia: a dalt el del nom i a sota el que conté el seu diagrama de subestats. Cada seqüència de subestats ocuparà una franja horitzontal separada per una línia discontínua de les franges adjacents. Un subestat inicial es representa amb una rodona plena, i un subestat final amb una rodona plena envoltada per una circumferència. Les transicions entre subestats de la mateixa seqüència es representen igual que entre estats; les transicions entre estats que pertanyen a seqüències diferents fan servir **pseudoestats de sincronització** de la manera que veurem en l'exemple corresponent.

De transicions que tenen per estat d'origen o de destinació un objecte compost, n'hi ha de diverses menes:

- Transicions que entren directament a un subestat o en surten.

- Transicions que entren o surten de l'estat compost, considerat com una unitat: si hi entra, és com si la transició tingués com a estats de destinació els estats inicials de totes les seqüències; si en surt, és com si la transició tingués com a estats d'origen els estats finals de totes les seqüències.

- Transicions que tenen com a destinació un **indicador d'història dels estats**, que és un pseudoestat que indica que en una transició de tornada a l'estat compost es torna, dins aquest, al mateix subestat (dins la seqüència continguda en la mateixa franja que l'indicador d'història en qüestió) del qual es va sortir l'última vegada que es va sortir de l'estat compost. Si el subestat pot ser un estat compost, hi ha una opció que indica que es torni tam-

bé a un subestat d'aquest, seleccionat de la mateixa manera, i això per a tants nivells de subestats com hi hagi. De l'indicador d'història pot sortir una transició cap a un subestat que serà el subestat de destinació en el cas que la transició tingui lloc sense que anteriorment s'hagués estat cap vegada a l'estat compost.

- Transicions *stubbed*. Aquestes transicions tenen com a estat de destinació (origen) algun subestat d'un estat compost del qual no s'especifica el diagrama de subestats; aquest subestat no pot ser inicial (final), ja que en el cas contrari podrien tenir com a estat de destinació (origen) l'estat compost, segons s'ha dit abans.

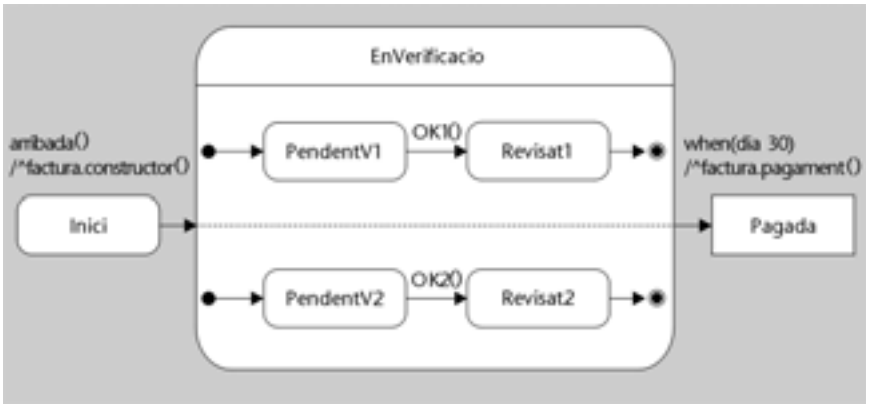
Exemples de transicions que tenen per estat d'origen o destinació un estat compost

Vegeu l'"Exemple de transicions complexes" en el subapartat 1.3 d'aquest mòdul didàctic.

Ara veurem tres exemples d'aquestes transicions:

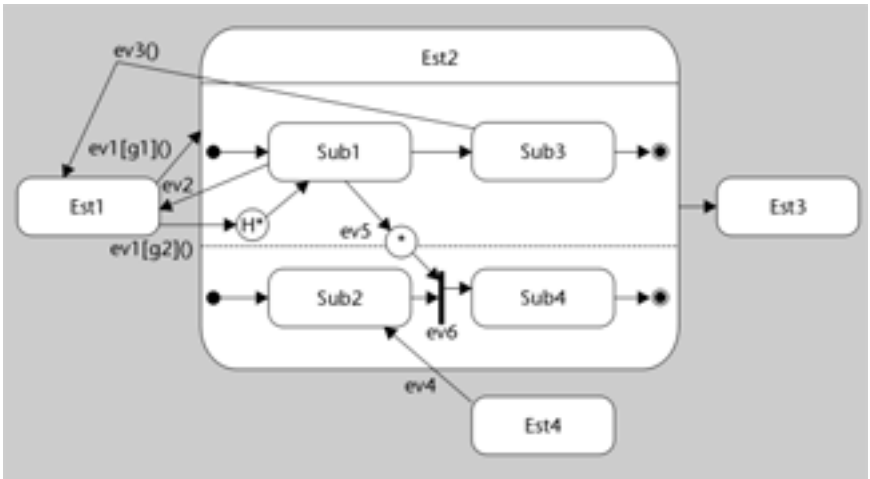
1) Exemple d'estat compost

Aquest exemple descriu el mateix cas que l'exemple de transicions complexes, però ara per mitjà d'un estat compost (de nom *EnVerificacio*). Aquest estat no té compartiment de transicions internes (vegeu el subapartat 1.5).



2) Exemple de transicions amb subestats, indicador d'història i pseudoestat de sincronització

Observem la figura següent:

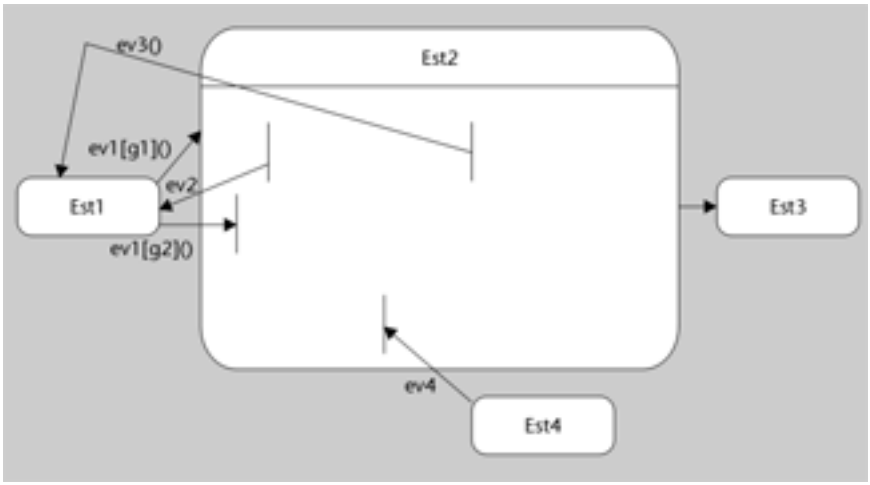


Des de l'estat *Est1* es va a l'estat compost *Est2* en general (és a dir, a tots els seus estats inicials) si es produeix l'esdeveniment *ev1* i es compleix la guarda *g1*; en canvi, si amb el mateix esdeveniment es compleix la guarda *g2*, llavors es passa al subestat d'*Est2* on s'estava

abans de sortir d'*Est2* (observeu que es pot sortir tant des de *Sub1* com des de *Sub3*), i si encara no s'havia estat mai a *Est2* llavors es passa a *Sub1*; si *Sub1* o *Sub3* tinguessin subestats, s'aniria al subestat d'aquests en què s'estava quan es va sortir d'*Est2* per última vegada (això ho indica el fet que l'*H* de l'indicador d'història vagi acompanyada d'un asterisc). Es pot passar de *Sub1* a *Sub4* –que pertany a una altra seqüència– per mitjà d'un pseudoestat de sincronització i una transició complexa de sincronització; l'asterisc dins el símbol de l'estat de sincronització denota que no hi ha límit màxim al nombre de vegades que es pot disparar la transició cap a *Sub4* a conseqüència del fet que es produeixi repetidament *ev6* mentre s'està ininterrompudament en *Sub1* (si hi hagués un valor màxim, es posaria aquest en lloc de l'asterisc). De l'estat *Est4* es passa directament al subestat *Sub2*.

3) Exemple de transicions *stubbed*

Aquest exemple descriu el mateix cas que l'exemple anterior, però ara les transicions que entren i surten de subestats no especifiquen de quins. Tanmateix, les transicions que entren i surten de l'estat compost en el seu conjunt s'indiquen de la mateixa manera que abans.



1.5. Notació ampliada de l'estat

Igual que la classe, l'estat té dues representacions gràfiques: una que només té el nom, que és la que hem emprat fins ara, i una altra amb diversos compartiments. Els compartiments són aquests:

- **Compartiment del nom**, que conté només el nom de l'estat.
- **Compartiment de les transicions internes i els esdeveniments interns**: conté una llista de les cadenes corresponents als esdeveniments i pseudoesdeveniments corresponents. En el cas dels pseudoesdeveniments *entry* i *exit*, no hi pot haver ni condicions de guarda ni paràmetres.
- **Compartiment del diagrama de subestats**, que ja hem vist, en el cas d'estats compostos.

En aquest punt, convé que feu els exercicis d'autoavaluació d'1 a 5 i després l'activitat 1 d'aquest mòdul didàctic.

2. El diagrama de casos d'ús

Els diagrames de casos d'ús* serveixen per a mostrar les funcions d'un sistema de programari des del punt de vista de les seves interaccions amb l'exterior i sense entrar ni en la descripció detallada ni en la implementació d'aquestes funcions.

* En anglès, *use case*.

Els casos d'ús es faran servir tant a nivell de recollida i documentació de requisits com d'anàlisi.

2.1. Actors

La finalitat d'un programari* és proporcionar informació a persones, màquines i dispositius o programaris de l'exterior, el conjunt de tots els quals anomenarem *entitats exteriors*; també hi ha entitats exteriors –les mateixes o d'altres– que demanen funcions al programari o bé li subministren informació per a tractar.

* Finalitat compartida amb la infraestructura de maquinari, programari bàsic i xarxa damunt la qual s'executa.

Un **actor** és un conjunt de papers d'una entitat exterior en relació amb el sistema de programari considerat.

Per tant, un actor no és l'entitat exterior en si, sinó només aquells aspectes que tenen a veure amb la seva interrelació amb el sistema de programari; podríem dir que un actor és la visió que el programari té d'una entitat exterior. Des d'aquest punt de vista, un actor és un conjunt de papers, ja que es considera que l'actor fa un paper diferent en cada interacció (cada cas d'ús, en UML) que té amb el programari. A més, si una entitat exterior fa dos conjunts de papers amb poca relació entre ells, li correspondran dos actors diferents (es podria dir que en aquest cas el programari no pot saber si els dos actors són o no la mateixa entitat exterior).

Sovint hi ha discussions sobre si una entitat exterior en particular és o no un actor en relació amb un programari; nosaltres considerarem que per a ser actor una entitat exterior ha de complir aquestes dues condicions:

- Ser autònoma respecte al programari, és a dir, que l'activitat en la qual fa servir la informació subministrada pel programari no estigui subordinada a la d'aquest.

- Tenir relació directa amb el programari o amb entitats exteriors que fan tasques subordinades a l'activitat del programari.

Exemples d'actors

Alguns exemples ens ajudaran a aclarir el concepte d'actor aplicant les regles que acabem d'indicar.

Un motor elèctric que és engegat o parat per un programari de temps real és un actor, ja que té la seva activitat autònoma (la seva rotació serveix per a accionar algun dispositiu mecànic), mentre que una impressora no té activitat autònoma perquè es limita a imprimir la informació que li envia un sistema de programari (o diversos), i, per tant, la seva activitat està subordinada a la generació de llistats pel programari.

Si la persona A recull un llistat de la impressora i el lliura a una persona B que en fa un resum manual de la informació i el dóna a una altra persona C, només B serà actor, ja que A no té una activitat autònoma i C no rep la informació directament del programari, i, per tant, el programari no "coneix" la seva existència. Naturalment, si A tingués alguna activitat autònoma i aquesta no tingués res a veure amb el programari –per exemple, escombrar l'oficina– no per això seria actor del programari.

Un actor és un classificador (però tot i això no és cap estereotip estàndard del classificador); si hi ha diverses entitats exteriors que fan el mateix conjunt de papers en relació amb el programari, un sol actor les representa totes, de la mateixa manera que una classe representa tots els seus objectes possibles (o bé, dit d'una altra manera, aquestes entitats exteriors serien instàncies de l'actor).

L'exemple dels bibliotecaris (I)

Si en una biblioteca pública tots els bibliotecaris poden fer servir les mateixes funcions del programari de suport a la gestió de la biblioteca, definirem un sol actor per a tots; però si hi ha un bibliotecari que pot fer algunes funcions que la resta no pot (com ara variar la durada dels préstecs), aquestes funcions s'assignarien a un altre actor. Hi ha dues maneres de fer-ho; vegeu una mica més avall la segona part d'aquest exemple.

Entre actors, es poden establir relacions d'especialització/generalització amb herència, igual que entre classes; un actor A que és una especialització d'un altre B fa almenys tots els papers de B.

L'exemple dels bibliotecaris (II)

En l'exemple dels bibliotecaris, el fet que hi hagi alguns bibliotecaris (diguem-ne els caps de biblioteca) que poden fer totes les funcions (casos d'ús, recordem-ho) que fan la resta de bibliotecaris, més d'altres de pròpies, fa que puguem definir un actor *Bibliotecari* i un altre *CapDeBiblioteca*, i aquest heretarà d'aquell. Però això mateix es podria expressar d'una altra manera: fent que l'actor *CapDeBiblioteca* només comprengués els papers que tenen els caps de biblioteca i no tenen la resta de bibliotecaris; en aquest cas, evidentment, no hi hauria herència, i als caps de biblioteca els correspondrien dos actors, *Bibliotecari* i *CapDeBiblioteca*.

2.2. Concepte de cas d'ús

Un cas d'ús documenta una interacció entre el programari i un actor o més. La interacció en qüestió ha de ser en principi una funció autònoma dins el programari.

Cas especial

Si un procés s'ha d'engegar, automàticament o no, en una data i/o hora determinada, es considera que hi ha un actor fictici, per exemple, que fa aquest paper.

Entre els actors que participen en un cas d'ús, moltes vegades convé distingir l'**actor primari** del cas d'ús, que és aquell qui l'engega demanant la funció corresponent del programari.

Els casos d'ús són un cas particular dels classificadors; una instància d'un cas d'ús és una execució d'aquest amb intervenció de casos particulars dels actors involucrats. Els casos d'ús poden tenir atributs i operacions, que poden servir per a descriure'n el procés (que també es pot descriure d'altres maneres, com text ordinari i diagrames d'estats i d'activitat).

Es poden fer descripcions més formals i detallades dels casos d'ús durant l'anàlisi i també de la seva implementació suprimir per mitjà de diagrames de col·laboració o de seqüències. Vegeu els subapartats 3.3 i 3.4 d'aquest mòdul didàctic.

Entre els casos d'ús i els actors que intervenen hi ha associacions (cosa que no té res d'especial, ja que uns i altres són classificadors); el significat d'aquestes associacions és el paper de l'actor en relació amb el cas d'ús, però no representen ni la direcció ni el contingut d'un eventual flux de dades entre el programari i l'actor.

2.3. Relacions entre casos d'ús

Entre els casos d'ús hi pot haver tres tipus de relació:

1) Relacions d'extensió: es diu que el cas d'ús A estén el B si dins B s'executa A quan es compleix una condició determinada; A pot estar dividit en fragments, situats en llocs concrets d'A anomenats **punts d'extensió**. A ha de ser un cas d'ús que també es pugui executar separadament de B i ha de tenir el mateix actor primari que aquest.

2) Relacions d'inclusió: un cas d'ús A està inclòs dins els casos d'ús B, C, etc., si és una part de procés comuna a tots aquests. A no és un cas d'ús autònom, en el sentit que no tindrà actor primari, sinó que sempre serà engegat per un o altre dels casos d'ús que l'inclouen; tanmateix la seva implementació no pot dependre d'aquests*. Per tant, la inclusió de casos d'ús és essencialment una forma de reutilització.

*** Per exemple, no en pot fer servir de variables.**

3) Relacions de generalització/explotació: un cas d'ús A és una especialització d'un altre cas d'ús B si A fa tot el procés de B més algun procés específic.

A més, hi ha una forma de relació no tipificada (semblant a les agregacions entre classes) entre casos d'ús que consisteix a definir, per exemple, un cas d'ús que correspon a tot el sistema, diversos que corresponen a tots els seus subsistemes, etc.

2.4. Notació

Tant per als casos d'ús com per als actors no es fa servir el símbol dels classificadors, sinó símbols especials com els següents:

- Els casos d'ús es representen mitjançant el·lipses de traç continu. De vegades s'agrupen totes les el·lipses dins un rectangle que representa tot el programari.

- Els actors es representen mitjançant una figura humana esquemàtica.

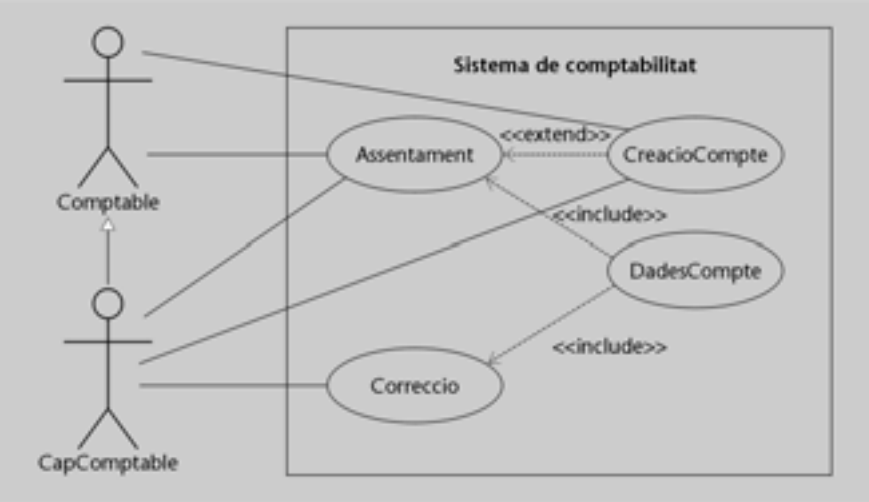
- Les relacions d'especialització entre actors i entre casos d'ús es representen mitjançant el mateix tipus de fletxa que en el cas de classes.

- Les relacions d'extensió i d'inclusió entre casos d'ús són estereotips de la dependència entre classificadors, i es representen mitjançant les paraules clau *extend* i *include*, respectivament.

Vegeu les relacions de dependència entre classificadors al subapartat 6.3 del mòdul "UML(I): el model estàtic" d'aquesta assignatura.

Exemple de casos d'ús i actors amb diferents relacions

Considerem l'exemple següent:



Els usuaris corresponents a l'actor *Comptable* només poden intervenir en el casos d'ús *Assentament* i *CreacioCompte*, mentre que l'actor *CapComptable* pot fer també el cas d'ús *Correccio*; per tant, *CapComptable* és una especialització de *Comptable*. El cas d'ús *CreacioCompte* estén el cas d'ús *Assentament* perquè quan s'intenta fer un assentament amb un compte inexistent cal crear aquest compte. El cas d'ús *DadesCompte* representa l'accés a les dades d'un compte des de dins els casos d'ús *Assentament* i *Correccio*; a diferència de *CreacioCompte*, no és un cas d'ús que es pugui dur a terme de manera directa i independent per un actor, i, per tant, la relació amb els casos que l'utilitzen és d'inclusió i no pas d'extensió.

Ara convé que feu els exercicis d'autoavaluació 6 i 7 i l'activitat 2.

3. Els diagrames d'interacció

L'execució d'un programari orientat a objectes consisteix en un encadenament d'operacions i de canvis d'estat d'objectes; aquest encadenament consisteix en el fet que durant l'execució d'una operació o durant una transició es criden operacions damunt altres objectes (o el mateix) i s'envien senyals que provoquen d'altres transicions. Així, es pot descriure el funcionament dels casos d'ús i també d'operacions complexes; en UML es fa mitjançant els anomenats **diagrames d'interacció**.

3.1. Interaccions i col·laboracions

Per començar, hem de definir els conceptes *d'interacció* i *col·laboració*.

3.1.1. Interaccions

Una interacció és l'especificació del comportament d'un cas d'ús o operació en termes de seqüències d'intercanvi de missatges entre objectes (o, més exactament, entre instàncies de classificadors). Aquests missatges contenen **estímuls**, que poden ser peticions d'execució d'operacions o bé senyals.

Un fil d'execució* és una seqüència de missatges tal que el primer conté un estímul que provoca l'enviament del segon, etc. Si el missatge A ha estat la causa que s'emetés el missatge B, es diu que A és el predecessor de B i B, el successor d'A. Que un missatge tingui diversos successors o diversos predecessors vol dir que hi ha una bifurcació o confluència de fils, respectivament.

* En anglès, *thread*.

3.1.2. Col·laboracions

De la mateixa manera que perquè puguin circular missatges entre ordinadors cal que els ordinadors estiguin units per enllaços de comunicacions, també cal una certa "infraestructura" per a la circulació de missatges entre objectes (a diferència del cas de les xarxes, aquí no es tracta d'una necessitat física, sinó d'una necessitat de coherència entre els diferents diagrames d'UML que descriuen un programari); aquesta infraestructura està formada per les classes o classificadors i les associacions entre aquestes, definides en el model estàtic, i també les associacions entre actors i objectes que s'obtenen quan es descriuen casos d'ús mitjançant interaccions.

D'acord amb això, per a cada interacció cal indicar quina part del model estàtic fa servir. Si un objecte de la classe A ha de demanar una operació d'un objecte de la classe B, cal certament que hi hagi aquestes dues classes i que B tingui definida l'operació que es demana; però cal també que hi hagi una associació entre aquestes dues classes i que tingui especificada la possibilitat de navegar almenys des de la classe A cap a la B.

Per tant, en una col·laboració damunt la qual ha de tenir lloc una interacció determinada cal que figurin tots els classificadors i associacions entre aquests que es faran servir en la interacció; però, de la mateixa manera que en un cas d'ús de les entitats exteriors no ens n'interessen tots els aspectes, sinó només el paper que fan en relació amb aquell cas d'ús, també en una col·laboració, més que classes o objectes, el que en sortirà seran llurs papers en relació amb la interacció en qüestió, i el mateix val per a les associacions.

Observació

Convé no confondre aquests papers amb els papers que fan els classificadors respecte a una associació que els relaciona.

Resumint, una **col·laboració** és un conjunt de papers de classificadors o instàncies i de papers d'associacions entre aquells que intervenen en una interacció. Una col·laboració es pot definir pel que fa a classificadors o pel que fa a instàncies.

La sintaxi dels noms dels papers de classificadors és la que presentem a continuació:

'/' nom_paper ':' nom_classificador '[' nom_estat ']'

o bé aquesta altra:

nom_instància '/' nom_paper ':' nom_classificador '[' nom_estat ']'

i en el cas d'una associació, tenim l'expressió següent:

'/' nom_paper ':' nom_associació

El *nom_estat* correspon a un dels estats que poden tenir els objectes del classificador en qüestió, i és opcional, juntament amb els claudàtors que l'envolten. La primera forma és aplicable tant a classificadors com a les seves instàncies.

La representació gràfica dels papers és igual que la dels classificadors i associacions corresponents, per bé que només cal especificar aquells elements que estan modificats –sempre en sentit restrictiu– en la col·laboració*. En una col·laboració poden figurar diversos papers del mateix classificador, amb noms de paper diferents.

*** Com ara les cardinalitats, que poden tenir valors més baixos, tant la màxima com la mínima.**

Els **multiobjectes** representen un conjunt d'objectes d'un paper amb cardinalitat més gran que 1 dins una associació.

Es representen amb dos rectangles sobreposats, dels quals el de davant està lleugerament desplaçat cap a un angle per a donar la sensació que hi ha una pila d'objectes.

Els multiobjectes necessiten dos missatges per a fer una operació en cadascun dels seus objectes: un serveix per a seleccionar el conjunt d'enllaços de l'associació que corresponen als objectes, i l'altre missatge s'envia separadament a cada objecte individual per mitjà de l'enllaç respectiu; aquests dos missatges poden estar combinats dins un que inclogui tant la iteració com l'aplicació de l'operació a cada objecte individual.

Exemple de col·laboració

Observem la figura següent:

El diagrama estàtic de partida seria aquest (potser amb altres classes i associacions que no fan al cas):

Veiem que en l'exemple de col·laboració no es fan servir tots els aspectes que tenen les dues classes i l'associació en el diagrama estàtic: només es fa servir la navegació en un sentit, i el paper *prestecs* i *reserves* de la classe *Llibre* se substitueix pel paper més restrictiu *prestecs*, cosa que es reflecteix en la cardinalitat màxima. S'ha definit una col·laboració entre objectes (més exactament, entre un objecte i un multiobjecte) i al paper que fa l'objecte *unLector* se li ha donat el nom *prestatari*, mentre que ni el multiobjecte de la classe *Llibre* ni el seu paper no tenen nom.

3.1.3. Patrons

Una col·laboració parametritzada constitueix la part estructural d'un patró orientat a objectes.

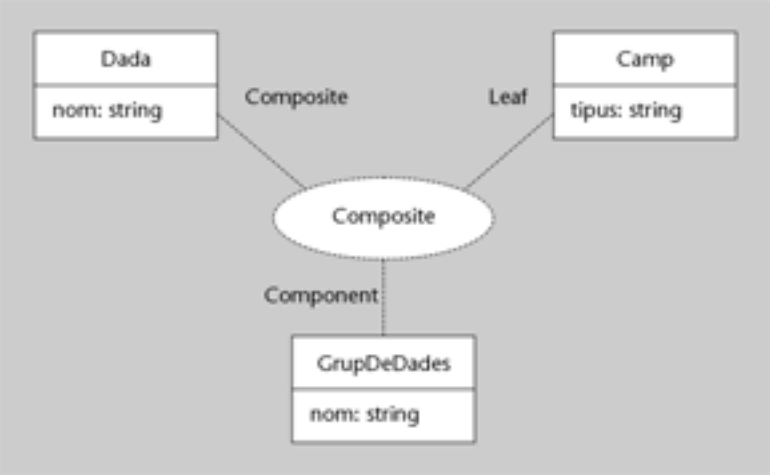
Els patrons són receptes formals per a resoldre problemes de disseny que es presenten sovint, i la descripció de la solució que proposen té una part estructural i una part dinàmica.

Vegeu els patrons al mòdul "Disseny orientat a objectes" d'aquesta assignatura.

Els patrons es representen com a el·lipses amb el contorn discontinu; una aplicació d'un patró a un cas particular es representa connectant el símbol del patró mitjançant línies discontínues a les classes o objectes que substitueixen les del patró en aquesta aplicació.

Exemple d'aplicació d'un patró

Aquest exemple correspon a una aplicació del patró *Composite*, molt conegut, que tracta de la implementació d'un objecte compost, els components del qual poden ser compostos o no.



En aquest cas, els objectes compostos són grups de dades, cada component dels quals pot ser o bé un camp (és a dir, una dada atòmica) o un altre grup de dades. Les classes *Camp*, *Dada* i *GrupDeDades* són els arguments que en aquest cas concret substitueixen respectivament les classes *Leaf*, *Component* i *Composite* que són els paràmetres de la col·laboració que descriu els aspectes estructurals del patró.

UML ofereix dos diagrames per a representar una interacció i la col·laboració en què es basa: el diagrama de col·laboració, que posa èmfasi en la descripció de la col·laboració, i el diagrama de seqüències, que posa èmfasi en la successió temporal dels missatges de la interacció; hi ha una tercera manera de descriure les interaccions, el diagrama d'activitats, que té una altra orientació.

Vegeu el diagrama d'activitats en l'apartat 4 d'aquest mòdul didàctic.

3.2. Diagrama de col·laboració

El diagrama de col·laboració és la representació d'una interacció mitjançant un diagrama estàtic de la col·laboració corresponent damunt el qual es representen els missatges de la interacció.

Per a cada missatge hi ha una especificació amb la sintaxi següent:

predecessors guarda expressió_de_seqüència valors_de_retorn signatura

A continuació, donarem una explicació de cada element del missatge:

- *predecessors* és la llista dels missatges predecessors del missatge en qüestió, en aquesta forma:

  ‘(’ número_de_seqüència ‘,’ número_de_seqüència ‘)’ ... ‘/’

  El número de seqüència del missatge que engega tota la interacció és 1, i els dels missatges que envia en diferents moments el procés que ha engegat directament són 1.1, 1.2, etc., mentre que si el missatge 1.2 engega un procés que envia alhora dos missatges (i es crea, doncs, dos fils d'execució) tindran els números que es distingiran per un nom, com ara 1.2a i 1.2b. Els missatges amb noms com ara 1, 1.3, 1.3.1, 1.3.1.2, etc., es diu que formen una **seqüència**; els predecessors que pertanyen a la mateixa seqüència que el missatge en qüestió no cal esmentar-los.

- *guarda* és la condició que s'ha de complir perquè s'enviï el missatge en qüestió (a més del fet que s'hagin rebut els missatges predecessors).

- *expressió_de_seqüència* té aquesta sintaxi:

  número_de_seqüència ‘[’ recurrència ‘]’ ‘,’ número_de_seqüència ‘[’ recurrència ‘]’ ... ‘:’

  Exemple de clàusula d'iteració

  La *clàusula_d'iteració* podria ser, posem per cas, [x = 1, …, 10].

  La *recurrència* té aquest format:

  ‘*’ ‘[’ clàusula_d'iteració ‘]’

  Si la iteració és en paral·lel en comptes de l'asterisc sol hi ha ‘*||’, o bé:

  ‘[’ clàusula_de_condició ‘]’

  La *clàusula_de_condició* acostuma a servir per a definir branques d'execució; hi pot haver diversos números de seqüència amb les seves clàusules de condició respectives.

- *valors_de_retorn* especifica una llista de noms de valors retornats –si n'hi ha– com a resultat del procés engegat pel missatge. Té el format següent:

  valor_de_retorn ‘,’ valor_de_retorn ... ‘:=’

- *signatura* està composta pel nom de l'estímul i per una llista d'arguments entre parèntesis.

  A més dels missatges, es pot indicar la creació i destrucció d'objectes i enllaços durant la interacció amb les paraules clau *new*, *destroyed* i *transient*, que equival a les altres dues i, per tant, denota que l'objecte o enllaç és creat i destruït durant la interacció.

Tipus de missatges

A continuació, veurem els diferents tipus de missatges:

a) Missatges simples: corresponen a la simple progressió dins un fil d'execució. Es representen amb una fletxa amb la punta oberta, amb la qual s'indica el sentit del missatge.

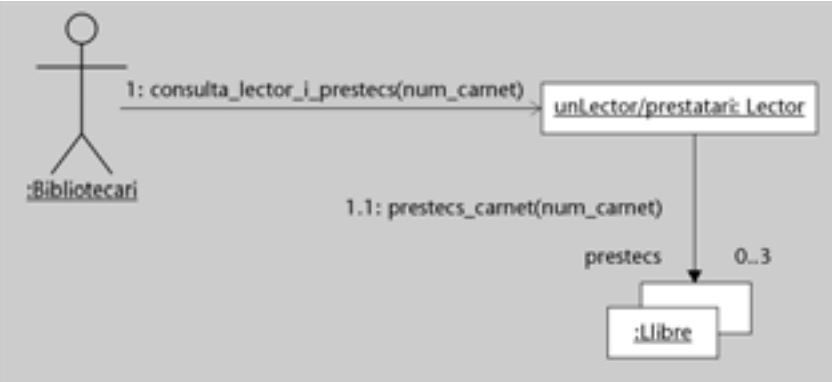
b) Missatges síncrons: només es donen quan el client envia un missatge al subministrador i aquest accepta el missatge; la classe emissora executa el codi fins que envia el missatge i, després, s'espera a rebre el resultat de l'operació associada al missatge. Es representen amb una fletxa amb la punta plena que n'indica el sentit.

c) Missatges asíncrons: la comunicació asíncrona es produeix quan la classe emissora envia un missatge al subministrador i es continua executant sense esperar que arribi el resultat; la classe receptora, per la seva banda, no executa l'operació immediatament, sinó que desa la petició en una cua. Es representa amb una fletxa amb la punta oberta i tallada horitzontalment.

Exemple de diagrama de col·laboració

Si fem servir la col·laboració de l'"Exemple de col·laboració", s'ha descrit una interacció en la qual un bibliotecari demana informació sobre un objecte de la classe *Lector* –identificat pel seu número de carnet– i els seus préstecs, mitjançant un missatge a l'objecte *unLector* que només té el número de seqüència i la signatura de l'operació demanada; durant l'execució d'aquesta, *unLector* envia un missatge al multiobjecte de la classe *Llibre*. Aquest darrer missatge és síncron; del primer no s'indica que sigui síncron o asíncron perquè correspon a una interacció entre un usuari i el sistema.

Vegeu l'"Exemple de col·laboració" al subapartat 3.1.2 d'aquest mòdul didàctic.



3.3. El diagrama de seqüències

A diferència del diagrama de col·laboració, en el diagrama de seqüències no s'hi representen explícitament els papers d'associacions (queden implícits en els missatges) i es representa explícitament l'ordre en el temps, i fins i tot la durada, dels missatges i de les operacions que engeguen.

El diagrama de seqüències està estructurat segons dues dimensions. El temps es representa verticalment i corre cap avall, i no està representat a escala necessàriament. En direcció horitzontal, hi ha franges verticals successives que corresponen als diferents papers de classificadors que participen en la interacció; cada paper de classificador està representat pel símbol habitual, el qual encapçala la seva línia de vida. L'ordre dels classificadors d'esquerra a dreta no és significatiu, si bé la tendència ha de ser que els missatges circulin d'esquerra a dreta i els resultats i respostes de dreta a esquerra.

La **línia de vida** simbolitza l'existència del paper en un cert període de temps. Es representa per una línia discontínua vertical que va des de la creació de l'objecte fins a la seva destrucció.

Si la destrucció no està prevista en el diagrama, llavors la línia de vida anirà fins a baix de tot del diagrama, és a dir, fins més enllà de l'últim missatge que es mostra. Si es vol indicar la destrucció de l'objecte, llavors el final de la línia de vida anirà marcat amb una X. Pot ser que durant una part de la línia de vida hi hagi dues activacions de l'objecte alhora, que no seran concurrents sinó alternatives segons una condició que determina que s'emeti el missatge que engega l'una o el que engega l'altra.

Una **activació** és una part de la línia de vida durant la qual el paper en qüestió executa una acció, o bé altres papers executen altres accions, a conseqüència d'una acció executada pel paper.

Les activacions serveixen per a modelar relacions de control entre classes i es representen mitjançant rectangles allargats verticalment inserits en les línies de vida. L'inici i la fi del rectangle coincideixen amb l'arribada del missatge que engega l'acció i la tramesa del missatge de resposta. Poden tenir etiquetes, que especifiquen l'acció que correspon al missatge. Per a especificar crides recurrents al mateix paper, es representarà una nova activació desplaçada una mica més a la dreta.

Els missatges s'indiquen amb fletxes iguals que les del diagrama de col·laboració, que comencen en una activació (al principi d'aquesta o pel mig) i acaben en una altra. El seu ordre de dalt a baix expressa l'ordre en què es produeixen en el temps; a més, una fletxa inclinada (cap avall, necessàriament) indica un missatge de durada no negligible, altrament la fletxa seria horitzontal.

L'especificació dels missatges és molt semblant a la que hem vist en el diagrama de col·laboració; les diferències principals són les següents:

- No s'indiquen els números de seqüència, ja que queden implícits en l'ordre temporal dels missatges i de les activacions que provoquen.

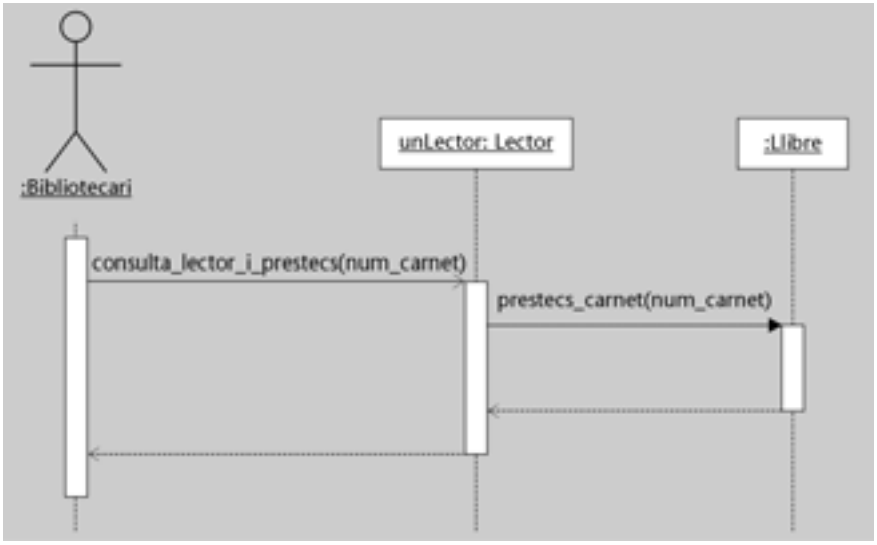
- Si hi ha dues expressions de seqüència per al mateix missatge hi haurà dues fletxes, cadascuna amb la clàusula de condició corresponent, que sortiran del mateix punt de l'activació.

A diferència del diagrama de col·laboració, es poden indicar els missatges de retorn al final d'una activació, en forma de fletxes de línia discontínua i punta oberta.

Exemple de diagrama de seqüències

Diagrama de seqüències equivalent a un diagrama de col·laboració

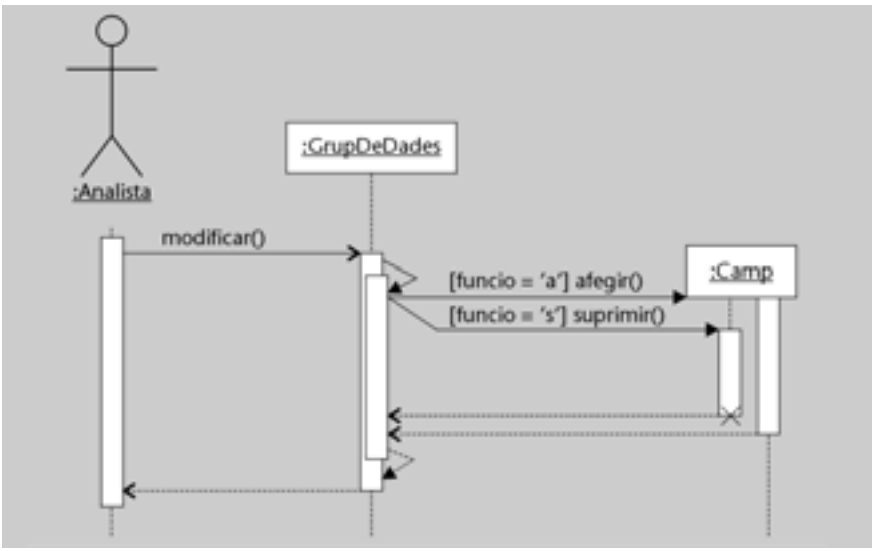
Aquest exemple correspon a l'exemple de diagrama de col·laboració del final del subapartat 3.2:



Observeu que no s'indiquen els papers dels objectes ni els números de seqüència dels missatges, però tot i això darrer queda clar quin missatge és conseqüència de quin altre. Observeu la diferència entre línia de vida i activació: com que els objectes i l'actor existien abans de la interacció i continuaran existint després, les seves línies de vida (les línies verticals discontínues) comencen abans i acaben després de les activacions respectives. El fet que les fletxes dels missatges són horitzontals vol dir que es poden considerar instantanis, però en canvi els processos tenen una durada no negligible, ja que tenen una certa llargada en direcció vertical i els missatges en cascada estan en fletxes cada vegada més avall.

Exemple de diagrama de seqüències amb creació i destrucció d'objectes i activacions alternatives

Considerem el diagrama de seqüències següent:



Es tracta d'afegir o suprimir un camp (dada atòmica) contingut dins un grup de dades, segons que la funció demanada sigui *a* o *s*, respectivament. Com que un grup de dades pot estar contingut dins un altre (una vegada o més) hi ha una activació recursiva d'objectes de la classe *GrupDeDades* (naturalment seran objectes diferents en cada activació, però en el diagrama es representa un objecte en general). Els dos missatges alternatius es produirien en el mateix moment; el fet que un d'aquests es representi amb una fletxa amb una part inclinada és només per raons de dibuix. Fixeu-vos en com es representen la creació i la destrucció del camp; la fletxa del missatge de creació (*afegir*) se suposa que coincideix amb el principi de l'activació, el qual coincideix també amb el principi de la línia de vida). Hi ha una bifurcació de la línia de vida de l'objecte de la classe *camp* des del començament, però les dues parts no es tornen a ajuntar perquè quan es destrueix un objecte (vegeu la X al final d'una de les seves activacions) la seva línia de vida ja no continua.

En aquest punt convé que feu els exercicis d'autoavaluació 8, 9 i 10 i l'activitat 3 d'aquest mòdul didàctic.

4. El diagrama d'activitats

El diagrama d'activitats es pot considerar una variant tant del diagrama d'estats com dels diagrames d'interacció, ja que serveix per a descriure els estats d'una activitat, que és un conjunt d'accions en seqüència i/o concurrents* en el qual intervenen classificadors.

*** Com ara un cas d'ús, una operació, un *workflow*, etc.**

4.1. Elements específics

El **diagrama d'activitats** té molts elements comuns amb els diagrames de col·laboració i d'estats (objectes, estats, transicions simples i complexes i pseudoestats, esdeveniments, senyals, fluxos de control); els seus elements específics són aquests:

1) Estats d'acció, que són un cas particular dels estats, en els quals no hi ha un objecte que roman a l'espera que es produeixi un esdeveniment que el faci sortir, sinó que s'està desenvolupant una acció, que és l'acció d'entrada de l'estat; quan acabi es produirà la transició, cosa que vol dir que un estat d'acció no pot tenir accions lligades a altres esdeveniments que el d'entrada ni, per tant, transicions internes.

Els estats d'acció es representen com un rectangle en el qual els costats dret i esquerre són semicircumferències, amb el nom de l'acció i el nom d'un esdeveniment seguit de "/defer" si durant l'acció es pot produir un esdeveniment que no pot ser tractat fins que acabi.

2) Fluxos d'objectes, que consisteixen en el fet que un objecte és creat o canvia d'estat en una acció i és utilitzat per una altra o unes altres.

Els fluxos d'objectes es representen per fletxes de punta oberta i línia discontínua. Quan un flux de control i un flux d'objecte coincideixen, es representa només el flux d'objecte.

3) Estats de flux d'objecte, que signifiquen que un objecte, probablement en un estat convencional* determinat, ha esdevingut disponible en acabar una acció (és a dir, en sortir d'un estat d'acció).

*** És a dir, un estat dels considerats en el diagrama d'estats.**

L'objecte, amb indicació del seu estat, es representa amb un flux d'objecte que entra i un altre que surt, i el símbol de l'objecte en un estat és el mateix utilitzat en les col·laboracions. Un objecte pot figurar diverses vegades en un diagrama, cada vegada en un estat diferent.

4) Estats de subactivitat, que són un cas particular dels estats compostos, i corresponen a l'execució de tot un subgraf d'activitat.

Els estats de subactivitat es representen com un estat d'acció dins el qual hi ha dos petits estats d'acció amb una transició de l'un a l'altre.

5) ***Swimlanes*** són franges verticals del diagrama les quals, a diferència de les del diagrama de seqüències, no corresponen a papers de classificadors sinó a unitats organitzatives responsables de diferents accions. Cada estat d'acció pertany a un *swimlane*, i hi pot haver transicions i fluxos d'objectes d'un *swimlane* a un altre. Els *swimlanes* duen un nom al capdamunt i estan separats per línies verticals contínues.

6) Icones de control, que representen la tramesa d'un senyal en acabar un estat d'acció i la seva rebuda en un altre com a entrada.

En les transicions hi pot haver bifurcacions i reunificacions posteriors del flux de control basades en condicions de guarda incompatibles –sovint complementàries–; per tant, no expressen cap sincronització, ja que els dos fluxos de control són alternatius; a la bifurcació i a la reunificació es posen rombes en comptes de símbols de pseudoestat (barres gruixudes).

Exemple de diagrama d'activitat

En l'exemple següent es tracta la gestió de la compra d'ordinadors en una empresa.

Si partim de l'estat inicial, el departament peticionari fa l'acció *Preparació proposta*, de la qual resulta un objecte de la classe *Compra* en l'estat *proposta* que entra a l'acció *Revisió pressupost*, que està a càrrec de *Control de gestió*; com a resultat d'aquesta acció; si no hi ha pressupost per a la compra (guarda *no pressupost*) el departament peticionari engega l'acció *Cancel·lació*, amb la qual s'arriba a l'estat final; si la guarda que es compleix és *hi ha pressupost*, l'objecte *Compra* passa a l'estat *pressupostada* i entra dins l'estat de subactivitat *Gestió de compra* (que se suposa que està detallat en un diagrama d'activitat a part). Veieu que alhora s'envia un senyal que fa que un objecte de la classe *ConnexioXarxa* passi a l'estat *activada* i que l'arribada a aquest estat produeix un senyal que, juntament amb l'acompliment successiu de l'activitat *Gestió de compra* i les accions *Enviament* i *Transport*, permet que s'engegui l'acció *Recepció* que porta a l'estat final.

5. Els diagrames d'implementació

Els diagrames d'implementació, a diferència dels estàtics i dels dinàmics, no descriuen la funcionalitat del programari sinó la seva estructura general amb vista a la seva construcció, execució i instal·lació, i són dos:

- El diagrama de components, que mostra quines són les diferents parts del programari.
- El diagrama de desplegament, que descriu la distribució física de les diferents parts del programari en temps d'execució.

Es fan servir en el disseny i la implementació.

5.1. El diagrama de components

El diagrama de components descriu la descomposició física del sistema de programari (i, eventualment, del seu entorn organitzatiu) en components, als efectes de construcció i funcionament.

La descomposició del diagrama de components es fa en termes de components i de relacions entre aquests.

5.1.1. Els components

Els components identifiquen objectes físics que hi ha en temps d'execució, de compilació o de desenvolupament i tenen identitat pròpia i una interfície ben definida.

El components inclouen codi en qualsevol dels seus formats (codi font o bé executable), DLL, imatges, però també poden ser documents manuals quan es descriuen parts no informatitzades d'un sistema d'informació.

Exemple de component

Un component podria ser, per exemple, un conjunt de classes –ja sigui en forma font o executable– agrupades dins un paquet d'UML o de Java; la interfície corresponent constarà, no pas de totes aquelles operacions d'alguna classe del paquet que poden ser dema-

nades des d'altres classes, sinó sols d'aquelles que poden ser demanades des de fora del paquet. Dins un programari pot ser que en temps de compilació hi hagi diversos paquets de classes fetes anteriorment que estan en forma executable, que facin falta per a compilar uns altres paquets, aquest de classes noves en forma font; després de la compilació pot ser que les classes resultants, ara ja totes en forma executable, s'agrupin en paquets d'una altra manera (amb vista a la comercialització del programari, per exemple) i les classes d'aquests diferents paquets col·laborin en temps d'execució.

Dels components es pot indicar el tipus o bé una instància; els primers s'anomenen **components de tipus** i existeixen en temps de desenvolupament o bé en temps de compilació, i els segons s'anomenen **components d'instància** i existeixen en temps d'execució.

Un component es representa mitjançant tres rectangles, un que conté l'identificador del component (el nom del tipus i opcionalment el de la instància, com en les classes i objectes) i dos de més petits incrustats al costat esquerre del primer.

5.1.2. Relacions entre components

En un diagrama de components es mostren les diferents relacions que hi pot haver entre components i altres components, objectes o processos (objectes actius).

En el cas de components no informàtics el significat de la relació és que un component fa servir la informació continguda en l'altre; en el cas de components de programari es distingeixen dos tipus de relacions, les relacions en temps de desenvolupament i les relacions de crida:

- Les **relacions en temps de desenvolupament** són associacions entre components que modelen dependències que es tindran en compte en temps de compilació o en temps d'enllaçament.

- Les **relacions de crida** són associacions entre components que serveixen per a modelar crides entre components; és a dir, el fet que un component –el client– fa servir serveis d'un altre –el proveïdor–. En temps de desenvolupament, les relacions de crida s'estableixen entre components de tipus, i es representen en els diagrames de components; en temps d'execució són entre dos components d'instància, i es representen en els diagrames de desplegament.

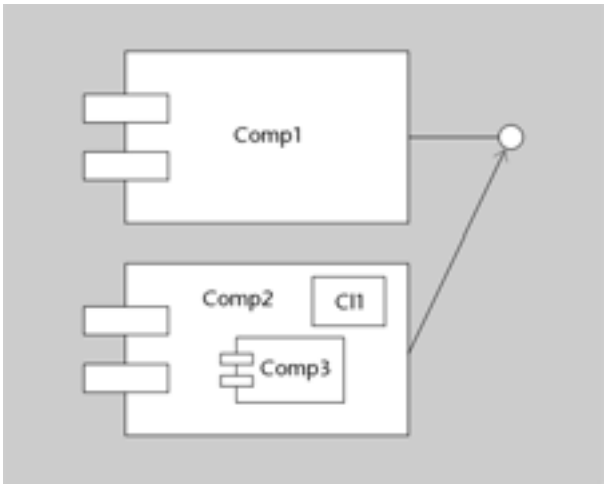
Notació

Ambdós tipus de relacions es representen amb la notació simplificada de les interfícies que heu vist en el subapartat 4.4 del mòdul "UML (I): el model estàtic" d'aquesta assignatura.

A més, un component pot tenir relacions d'agregació i composició amb altres components i amb objectes; en el cas de la composició, els components d'un component es poden representar dins aquest.

Exemple de diagrama de components

Vegeu el diagrama de components següent:



El component *Comp2* empra el *Comp1* i conté el component *Comp3* i un objecte de la classe Cl1.

5.2. El diagrama de desplegament

El diagrama de desplegament* permet de mostrar l'arquitectura en temps d'execució del sistema quant a maquinari i programari.

*** En anglès, *deployment*.**

El diagrama de desplegament es fa servir en el disseny i la implementació. S'hi poden distingir components (com els del diagrama de components) i nodes, i també les relacions entre tots aquests.

És més limitat que el diagrama de components, en el sentit que representa l'estructura del sistema només en temps d'execució, però no en temps de desenvolupament o compilació; però és més ampli en el sentit que pot contenir més tipus d'elements.

5.2.1. Els nodes

Els nodes representen objectes físics existents en temps d'execució, serveixen per a modelar recursos que tenen memòria i capacitat de procés, i tant poden ser ordinadors com dispositius o persones*; és mitjançant aquests que els components participen en els processos.

*** Pròpiament, en comptes de persones, serien els seus papers, com en els casos d'ús.**

Hi pot haver nodes de tipus i nodes d'instància. Els nodes es representen per parel·lelepípedes rectangulars.

5.2.2. Relacions dins el diagrama de desplegament

Entre els nodes hi ha relacions que signifiquen que hi ha comunicació entre ells; es representen mitjançant línies contínues, potser amb un estereotip que indica el tipus de comunicació*.

*** Per exemple, *Ethernet*.**

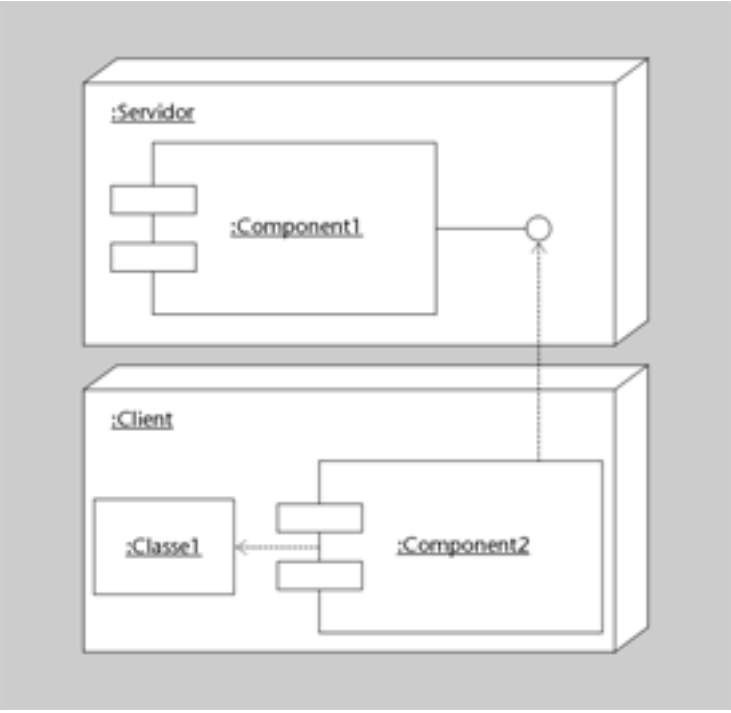
Un component o un objecte es pot executar si es fan servir els recursos d'un node o pot estar contingut en aquest; en el primer cas, hi ha una dependència amb l'estereotip *supports*; en el segon cas, hi ha una relació d'agregació o composició, que es pot representar de les maneres habituals.

Es pot representar que un objecte o component migra d'un node a un altre o es transforma en un altre; en el primer cas es representa l'objecte o component a tots dos nodes, i en tots dos casos la relació entre si és una dependència amb l'estereotip *becomes*. Poden tenir associada una propietat que indiqui el temps en el qual es produirà la migració.

A més, entre components hi pot haver les mateixes relacions mitjançant interfícies que en el diagrama de components (limitades però a relacions en temps d'execució).

Exemple de diagrama de desplegament

Observem el diagrama de desplegament següent:



El node *Servidor* conté el component *Component1* i té comunicació amb el node *Client*, el qual conté el component *Component2* i un objecte de *Classe1*; *Component2* depèn de *Classe1* d'una manera no especificada.

En aquest punt convé que feu els exercicis d'autoavaluació 12, 13 i 14 d'aquest mòdul didàctic.

Resum

En aquest mòdul, hem vist la resta de diagrames que considera l'estàndard UML. Aquests diagrames són els següents:

- diagrama de casos d'ús
- diagrama d'estats
- diagrames d'interacció: diagrama de seqüències i diagrama de col·laboració
- diagrama d'activitat
- diagrames d'implementació: diagrames de components i diagrama de desenvolupament

De tots aquests hem vist detalladament els conceptes i l'ús.

S'han vist també les connexions que hi ha entre aquests diagrames i també entre si i el diagrama estàtic; així, per exemple, els classificadors figuren en el diagrama estàtic, els d'interacció i els d'implementació; els estats en el diagrama d'estats i en el d'activitat; els actors en el diagrama de casos d'ús i en els d'interacció que descriuen la implementació dels casos d'ús, etc.

Activitats

1. Suposem que l'accés a la universitat des del batxillerat es desenvolupa segons el procediment següent (no es pretén que sigui fidel a la realitat):
Quan un alumne ha aprovat el batxillerat es preinscriu a vuit opcions d'estudis universitaris i es pot presentar a les proves d'accés, i si no les aprova s'ha de tornar a examinar l'any següent, però conserva la preinscripció. Si les aprova, entra en el procés d'assignació de places, i segons la seva nota se li assignarà plaça en la 1a. opció, en una opció de la 2a. a la 8a., o no se n'hi assignarà. Si se li ha assignat plaça de la 1a. opció, entra en el procés de matrícula de juliol; si se li ha assignat plaça d'una opció de la 2a. a la 8a., entra en el procés de matrícula de setembre, i si no se li ha assignat plaça, queda en la situació de no admès.
Els processos de matrícula de juliol i de setembre tenen dues etapes successives: matricular-se i pagar. Un alumne que s'hagi de matricular pel juliol i no ho faci, passa a la situació de no admès, igual que un alumne que s'hagi de matricular pel setembre i el 1r. d'octubre no ho hagi fet o no hagi pagat; però un alumne que s'hagi matriculat el juliol i no hagi pagat abans del 1r. d'agost pot fer-ho durant el setembre i ser admès.
Feu el diagrama d'estats de l'alumne.

2. Feu un diagrama de casos d'ús per a aquesta situació:
En un hotel, l'encarregat de les reserves rep peticions de reserva dels clients per telèfon i llavors fa una consulta de les habitacions disponibles els dies compresos en la reserva; si n'hi ha alguna de disponible, la reserva i ho comunica al client; si un client indica que arribarà més tard de les 8 del vespre, es posa una marca a la reserva. Quan arriba un client, un recepcionista comprova la reserva del client o li'n fa una després, si no en tenia, i en tots dos casos introdueix a la reserva la marca de l'arribada del client; quan un client se'n va, s'introdueix la data del dia. A les vuit del vespre s'engega un procés que esborra totes aquelles reserves que no portin la marca que el client arribarà més tard.

3. Feu els diagrames de col·laboració i de seqüència corresponents a l'enunciat de l'activitat 2.

Exercicis d'autoavaluació

1. Quina diferència hi ha entre un subestat i un pseudoestat?

2. Quina diferència hi ha entre una transició interna i una autotransició?

3. Quina diferència hi ha entre una condició de guarda relativa a un esdeveniment i la condició lligada a un esdeveniment de canvi?

4. De quin tipus són els esdeveniments de l'exemple d'estats, transicions i esdeveniments (tret de l'esdeveniment de temps)?

Vegeu l'"Exemple de diferents diagrames d'estats" al subapartat 1.2 d'aquest mòdul didàctic.

5. Hi pot haver més d'una transició amb el mateix estat d'origen i el mateix esdeveniment?

6. És correcta l'afirmació "els actors són classes"?

7. En una agència de viatges, un client demana a un empleat que li miri per l'ordinador si hi ha places en un vol determinat. Quin actor o actors hi ha?

8. Quina relació hi ha entre *interacció* i *col·laboració*?

9. En un diagrama de col·laboració, dos missatges tenen, respectivament, els números 1.1 i 1.2. Què vol dir això? Com es representaria la mateixa informació en el diagrama de seqüència de la mateixa interacció?

10. Quina diferència hi ha entre *línia de vida* i *activació*? En quins casos coincideixen?

11. Quina diferència hi ha entre un estat i un estat d'acció?

12. Indiqueu les diferències bàsiques entre el diagrama de components i el diagrama de desplegament.

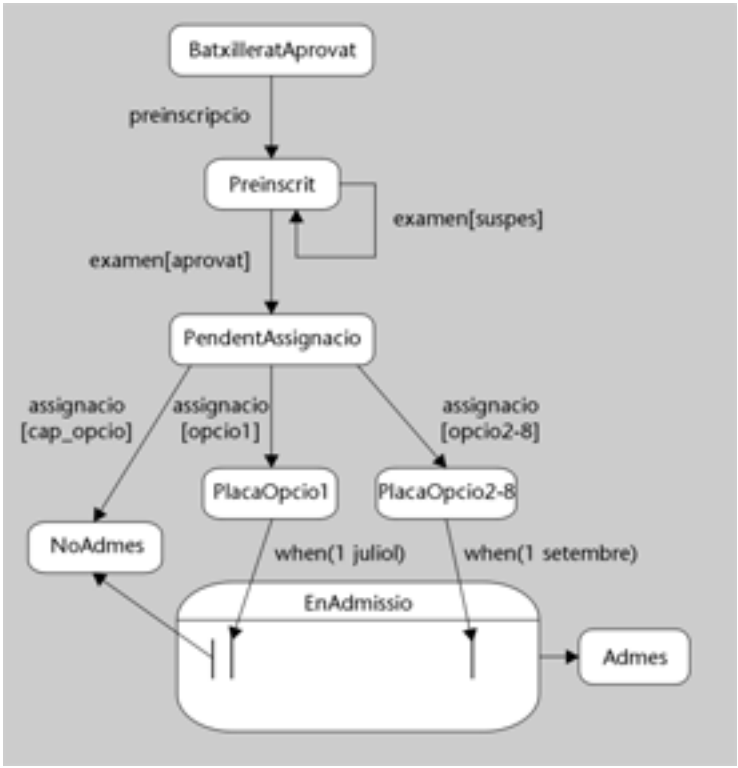
13. Són classes els components? I els nodes?

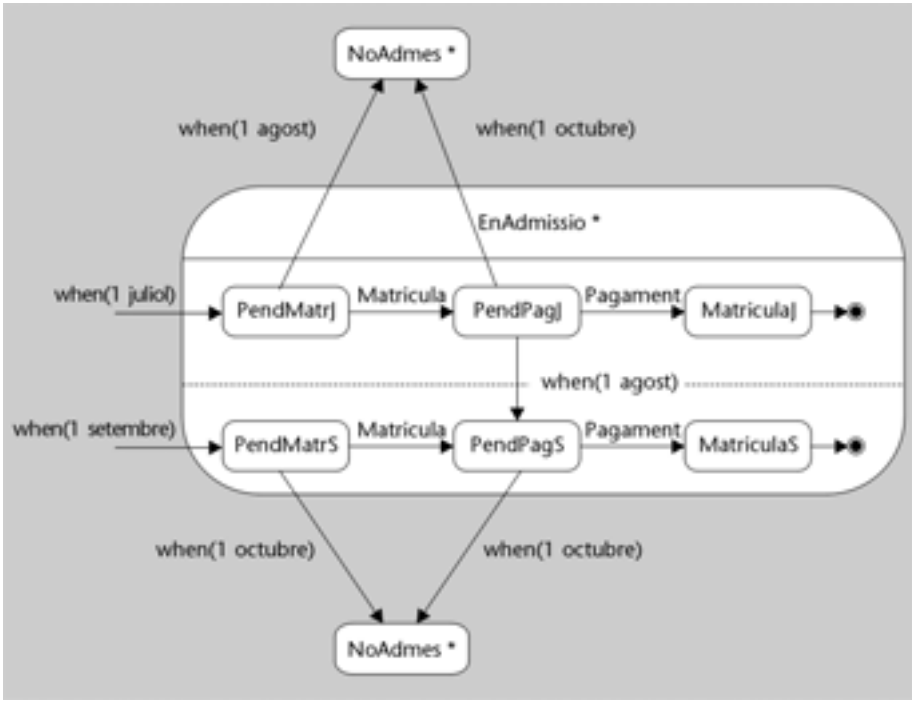
14. Què seria una instància de node? I una de component?

Solucionari

Activitats

1.





Hi ha altres solucions.
Per raons de claredat del diagrama aquells estats que tenen el nom seguit d'un asterisc són duplicats del que té el mateix nom sense asterisc.

2. Hi ha tres actors, *EncarregatReserves*, *Recepcionista* i *Rellotge*; els casos d'ús són *Reserva* (del primer actor), *Arribada* i *Partida* (del segon) i *Cancel·lacio*; *Reserva* estén *Arribada*.

3. Hi ha moltes solucions possibles. Com a orientacions generals, podríem dir que probablement calgui una classe *Habitacio* i una altra *Reserva*, amb una associació d'un a diversos entre elles, i que el procés d'una reserva demanaria almenys dues operacions diferents damunt la classe *Habitacio*: una consulta sobre un multiobjecte que serien les habitacions disponibles en una data determinada, i una altra que crearia una reserva d'aquella habitació.

Exercicis d'autoavaluació

1. Un subestat és un estat dins un estat, mentre que un pseudoestat no és un estat, sinó un recurs de representació gràfica, que no correspon a cap concepte nou.

Vegeu el pseudoestat en el subapartat 1.4 d'aquest mòdul didàctic.

2. En una autotransició, figura que hi ha canvi d'estat i, per tant, s'executen les eventuals accions d'entrada i sortida; en canvi, en una transició interna no es canvia d'estat i, per tant, aquelles accions no s'executen.

Vegeu el subapartat 1.1.

3. Una condició de guarda s'avalua un cop s'ha produït un esdeveniment per a comprovar si ha de tenir lloc una determinada transició; va lligada a aquesta més que a l'esdeveniment i s'avalua després que aquest s'ha produït.
Un esdeveniment de canvi es produeix quan la condició lligada a aquest passa de no complir-se a complir-se, i, per tant, l'avaluació d'aquesta és anterior a l'esdeveniment.

Vegeu el subapartat 1.1.

4. Són esdeveniments de crida, perquè criden operacions o senyals.

5. Sí. D'una banda, hi pot haver diverses transicions lligades a guardes diferents (per exemple, una i la seva contrària); de l'altra, hi pot haver una transició complexa amb diversos estats de destinació.

6. L'afirmació no és correcta. Els actors són classificadors, com les classes, però no classes.

Vegeu el subapartat 1.2.

7. Hi ha un sol actor que és l'empleat, perquè és l'únic que té accés directe al programari.

8. Una col·laboració és un conjunt de papers de classificadors connectats per associacions; una interacció és un intercanvi de missatges entre instàncies que té lloc dins una col·laboració seguint aquestes associacions.

Vegeu el subapartat 3.1.

9. Tots dos missatges són conseqüència del missatge 1 i s'emeten primer un i després l'altre. En el diagrama de seqüències, sortirien tots dos de l'activació engegada pel missatge 1, primer 1.1 i després (és a dir, més avall) 1.2.

Vegeu el subapartat 3.2.

10. La línia de vida representa tota la durada d'un paper d'un classificador, des que es crea fins que es destrueix, mentre que una activació representa el temps en què s'estan executant accions relatives al paper en qüestió durant una interacció.
L'activació coincidiria amb la línia de vida si el paper fos creat i destruït durant la interacció i entremig s'executessin només accions relatives al paper en qüestió.

Vegeu el subapartat 3.3.

11. A un estat s'arriba per mitjà d'una transició provocada per un esdeveniment; a un estat d'acció s'arriba quan acaba l'acció associada a aquest.

12. En un diagrama de components no hi ha nodes, d'una banda, i de l'altra els components no són necessàriament executables, sinó que poden estar en forma font o objecte.

13. És clar que tant els nodes com els components són classificadors, ja que tenen tipus i instàncies. Els nodes són classes, ja que implementen interfícies.

14. Una instància de node pot ser, per exemple, un ordinador concret entre tots aquells que poden contenir o executar components de les mateixes classes; en general, una instància de node és un recurs concret entre els d'una certa mena.
Una instància de component és una execució amb identitat pròpia d'un component (d'un component es poden executar, eventualment, diverses còpies alhora –per exemple, en diferents nodes– i cadascuna té identitat); per això, les instàncies de components només poden aparèixer en diagrames de components en temps d'execució o en diagrames de desplegament.

Glossari

acció
Procés que o bé no s'executa o bé s'executa fins al final.

activació
Part de la línia de vida d'un paper de classificador durant la qual s'executen accions lligades a aquest.

actor
Conjunt de papers que fa una entitat exterior en relació amb un programari; cada paper correspon a un cas d'ús.

cas d'ús
Interacció entre el programari i un actor o més que comporta una o més accions.

component
Objecte físic que existeix en temps de desenvolupament, de compilació o d'execució i té identitat pròpia i interfície.

esdeveniment
Fet que es pot produir en un instant en el temps i que pot provocar una transició.

estat
Situació durant la vida d'un objecte o la durada d'una interacció en què compleix alguna condició, duu a terme alguna activitat o espera algun esdeveniment.

estat d'acció
Estat que correspon al fet que hi hagi en curs una acció determinada.

estímul
Petició d'una operació o comunicació d'un senyal que arriben a un objecte.

interacció
Especificació del funcionament d'una operació o cas d'ús en termes de seqüències de missatges entre instàncies de classificadors que demanen operacions o envien senyals.

línia de vida
Interval de temps durant el qual un paper existeix.

node
Representació d'un objecte físic existent en temps d'execució amb memòria i capacitat de procés.

senyal
Estímul entre dues instàncies de classificadors que pot provocar un esdeveniment.

transició composta
Transició amb diversos estats d'origen i/o de destinació.

transició interna
Recurs per a especificar que quan es produeix un cert esdeveniment que té lloc mentre l'objecte està en un cert estat s'han d'executar determinades accions sense que hi hagi canvi d'estat.

transició simple
Pas d'un estat d'origen a un altre de destinació provocat per un esdeveniment; pot engegar accions.

Bibliografia

Fowler, M.; Scott, K. (1999). *UML gota a gota*. Amsterdam: Prentice Hall.

Rumbaugh, J.; Jacobson, I.; Booch, G. (2000). *UML. El lenguaje de modelado unificado. Manual de referencia*. Addison Wesley.

Booch, G.; Rumbaugh, J.; Jacobson, I. (1999). *UML. El lenguaje de modelado unificado. Guía del usuario*. Addison Wesley.