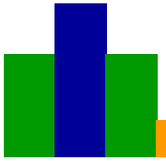


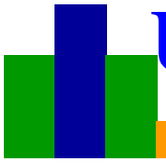
Lecţia 5:

UML: Modelare structurală şi comportamentală (I)

v1.0

Gheorghe Stefanescu — Universitatea Bucureşti

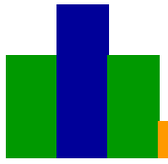
Metode de Dezvoltare Software, Sem.2
Februarie 2007— Iunie 2007



UML: Modelare structurală şi comportamentală

Cuprins:

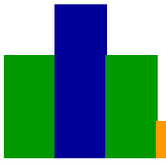
- *Generalităţi*
- Modelare structurală
- Modelare comportamentală
- Concluzii, diverse, etc.



Generalităţi

UML (The Unified Modeling Language):

- UML este un limbaj *grafic* pentru *vizualizarea, specificarea, construcţia*, şi *documentaţia* necesare pentru dezvoltarea de sisteme software complexe
- UML 1.1 a fost adoptat ca standard de OMG (Object Management Group) in 1997. Curent s-a ajuns la versiunea 2.0.
- A fost elaborat cu eforturi multiple, părinţii fiind consideraţi G.Booch, I.Jacobson, şi J.Rumbaugh.
- Noi folosim sursa: *G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeing Language UML, Addison Wesley, 1999*



Modelare

Ce este un model?

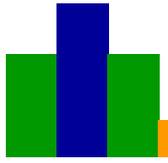
- Un model este o *simplificare a realităţii*.

De ce folosim modele?

- pentru *a înţelege mai bine* sistemul pe care îl dezvoltăm
- pentru a aproxima *sisteme complexe* pe care *nu le putem înţelege* în totalitate

Modelare OO

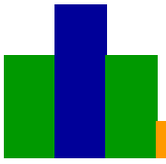
- UML este fundamental legat de *metodologia OO* (Object-Oriented) folosită pentru dezvoltarea de software.



..Modelare

Principii de modelare

- Alegerea unui model are implicaţii profunde asupra modului în care se *atacă* şi *rezolvă* problema.
- Orice model poate fi prezentat cu diverse grade de *precizie*.
- Cele mai bune modele sunt cele care sunt bine ancorate în *realitate*.
- De regulă, nu este suficient un singur model - sistemele netriviale necesită o abordare care cere *construcţia mai multor modele aproximante*, aproape independente.



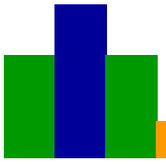
UML

UML ca limbaj pentru vizualizare:

- In esenţă, dezvoltarea unui sistem necesită *scrierea de cod* (text); *modelarea vizuală ajută* la:
  — comunicarea cu alţii;
  — întelegerea unor lucruri care transcend codul
  (ierarhie de clase, modalităţi de plasare a executabilelor, migrări de obiecte, etc);
  — scrierea de documentaţie pentru reutilizarea codului

UML ca limbaj pentru specificare:

- în fiecare fază de analiză, proiectare, implementare, se pot face specificaţii UML quasi-complete pentru respectiva activitate



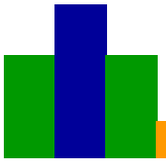
..UML

UML ca limbaj pentru construcţie:

- UML *nu este limbaj pe programare*, dar este *direct conectat* cu astfel de limbaje (Java, C++, Vizual Basic, limbaje de baze de date, etc.)
- *Separaţie*: *unele lucruri* se exprimă mai uşor *grafic* în UML, *altele textual*, în limbajul de programare folosit pentru dezvoltarea aplicaţiei

UML ca limbaj pentru documentaţie:

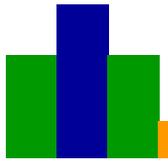
- UML permite elaborarea documentaţiei, care, de regulă, include informaţii despre: *cerinţele aplicaţiei, arhitectură, proiectare, codul sursă, planurile proiectului, teste, prototipuri, lansări*



..UML

UML - domenii de utilizare:

- sisteme informatice în companii
- servicii financiar-bancare
- telecomunicaţii
- transport
- apărare/spaţiu
- vânzări
- sisteme medicale
- aplicaţii ştiinţifice
- servicii web distribuite
- etc.



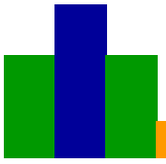
Structura UML: Blocuri de bază

Blocuri de bază: In UML, blocurile de bază sunt *lucruri (things), relaţii, diagrame*.

Lucrurile (things) sunt de 4 tipuri: *structurale, comportamentale, de grupare, de adnotare*

Lucruri structurale

- clase
- interfeţe
- colaborări
- use cases (moduri de utilizare)
- clase active
- componente
- noduri



..Structura UML: Blocuri de bază

Lucruri coportamentale

- interacţii
- maşini de stări

Lucruri de grupare

- pachete

Lucruri de adnotare

- note

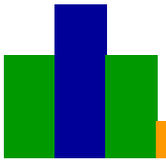
Relaţiile sunt de 4 tipuri

Dependenţe

Asociaţii

Generalizări

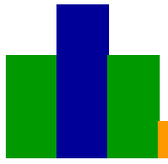
Realizări



..Structura UML: Blocuri de bază

Diagramele sunt de 9 tipuri

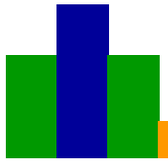
- **Diagrame de clase** (class diagram)
- **Diagrame de obiecte** (object diagram)
- **Diagrame de utilizare** (use case diagram)
- **Diagrame de secvenţe** (sequence diagram)
- **Diagrame de colaborări** (collaboration diagram)
- **Diagrame de statechart-uri** (statechart diagram)
- **Diagrame de activităţi** (activity diagram)
- **Diagrame de componente** (component diagram)
- **Diagrame de desfăşurare** (deployment diagram)



Structura UML: Roluri, extensii

Roluri:

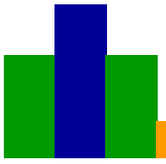
- Există modele UML *bine-formate*, ca şi modele *parţiale*.
- In cele bine formate, există reguli semantice clare pentru
  - nume
  - “scope”
  - vizibilitate
  - integritate date
  - execuţie



Structura UML: Roluri, extensii

Mecanisme comune in UML:

- **Specificaţii**
- **Modificări de notaţii** (adornments)
- **Diviziuni comune**
- **Mecanisme de extensie**
  - **Stereo-tipuri**
  - **Valori ataşate** (tagged values)
  - **Constrângeri**



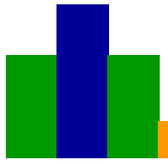
Arhitectura sistemelor software

Arhitectura sistemelor software conţine deciziile importante legate de:

- organizarea sistemului software
- selecţia elementelor structurale şi a interfeţelor
- comportamentul rezultat din colaborarea elementelor
- compunerea elementelor structurale şi comportamentale în sisteme din ce în ce mai mari
- stilul de organizare a elementelor statice şi dinamice, a interfeţelor, colaborărilor, şi a compunerii lor

Există *5 vederi (ori proiecţii) majore*:

(1) Design view, (2) Implementation view, (3) Process view, (4) Deployment view, (5) Use-case view.



Fazele de dezvoltare

Fazele de dezvoltare: UML este, în mare, *independent de metodologiile concrete* folosite curent în dezvoltarea de software. Procesul de dezvoltare din UML este:

A: condus de modul de utilizare a sistemului

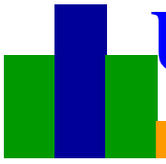
- important este ce face sistemul (*specificare, proiectare, testare*)

B: axat pe arhitectură

- arhitectura este folosită pentru probleme legate de *concepţie, construcţie, management*, şi *dezvoltare*

C: iterativ, incremental

- dezvoltarea trece prin 4 faze: *concepţie, elaborare, construcţie*, şi *tranziţie* (spre utilizatori)



UML: Modelare structurală şi comportamentală

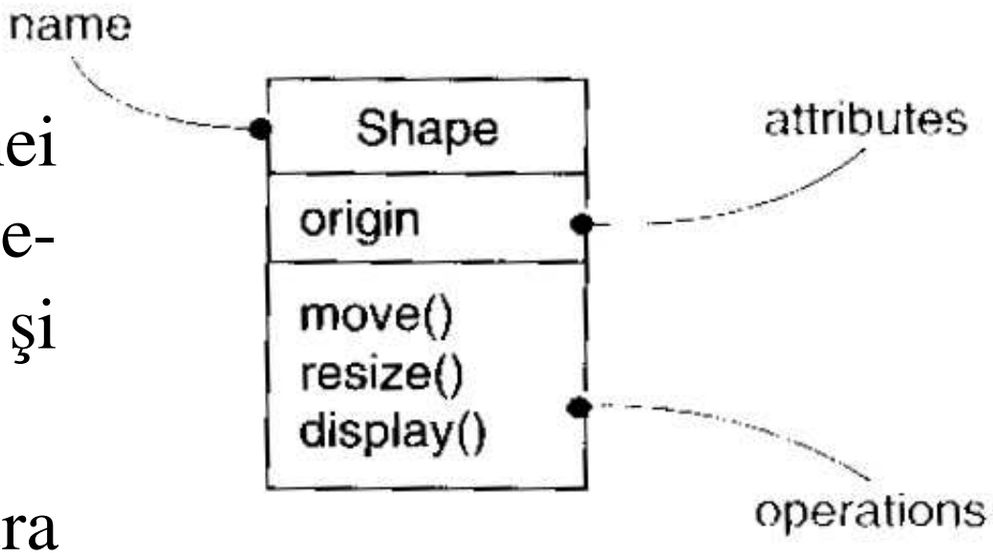
Cuprins:

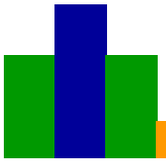
- Generalităţi
- *Modelare structurală*
  - *Clase*
  - Relaţii
  - Mecanisme comune
  - Diagrame
  - Diagrame de clase
- Modelare comportamentală
- Concluzii, diverse, etc.

Clase

Clase:

- o clasă este o descriere a unei mulţimi de obiecte care au aceleasi *atribute, operaţii, relaţii*, şi *semantică*
- este reprezentată ca în figura alăturată
- de regulă, are 4 zone pentru:
  (1) nume, (2) atribute, (3) operaţii, şi *(4) responsabilităţi*





..Clase

Nume

- poate fi *simplu*, ori *cu cale*;
  Exemple: `Wall`, `java::awt::Rectangle`

Atribute

- se pot defini *tipul* şi *valori iniţiale*;
  Exemple: `hight:Float`, `isOld:Boolean = false`

Operaţii

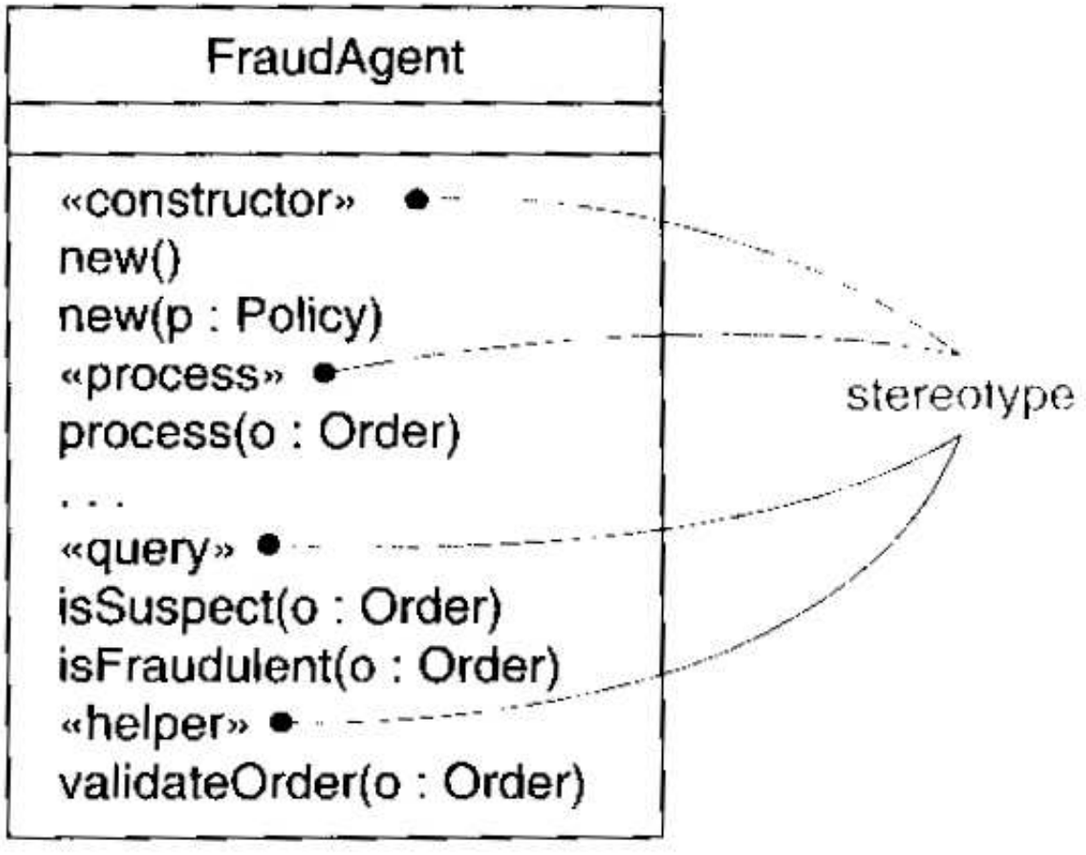
- includ *specificaţia signaturii*;
  Exemple: `reset()`, `setAlarm(t:Temperature)`

..Clase

Organizarea atributelor şi operaţiilor

- se pot folosi *stereptipuri* ca în figură pentru a le organiza

  (stereptipurile sunt extensii ale blocurilor de bază folosite de utilizatori)

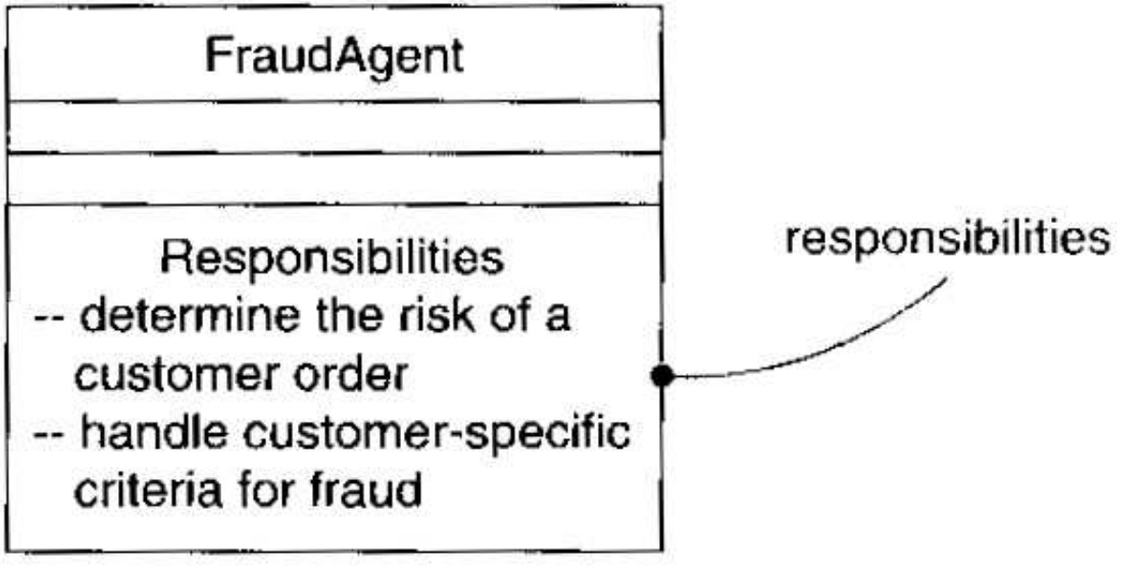


- Exemple:
  “<<id>>”, “...”, etc.

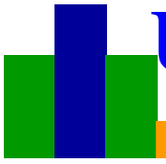
..Clase

Responsabilităţi:

- o *responsabilitate* este un *contract* ori o *obligaţie* a clasei
- se descriu într-o zonă separată, ca în desen



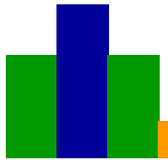
- ele pot fi definite mai *informal*, ori complet *formal*
- responsabilităţile au valoare semantică, anume *restricţionează comportamentul* obiectelor din clasă



UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- *Modelare structurală*
  - Clase
  - *Relaţii*
  - Mecanisme comune
  - Diagrame
  - Diagrame de clase
- Modelare comportamentală
- Concluzii, diverse, etc.

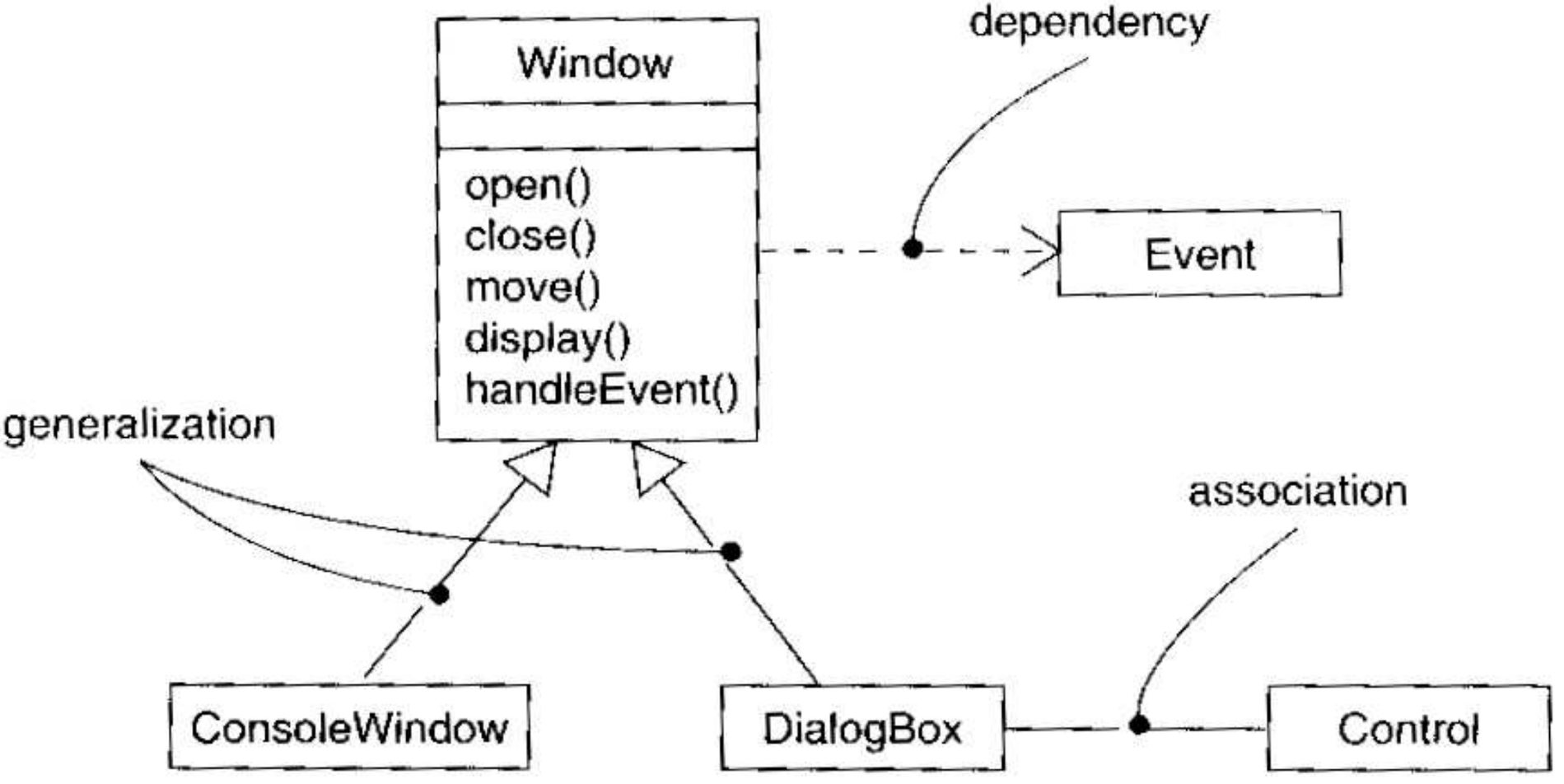


Relaţii

Relaţii:

- *relaţiile* sunt *conexiuni între "lucruri"* (things)
- cele mai importante sunt:
  - *dependenţe* - *relaţii de utilizare*, deci schimbarea unui lucru implică schimbarea celuilalt
  - *generalizări* - relaţii dintre *general şi particular* clase_generalizate-clase_specializate, părinte-fiu, etc.
  - *asociaţii* - *relaţii structurale* între instanţe
- se reprezintă *grafic* prin diverse tipuri de *linii, săgeţi*, etc.
- un exemplu este pe slide-ul următor

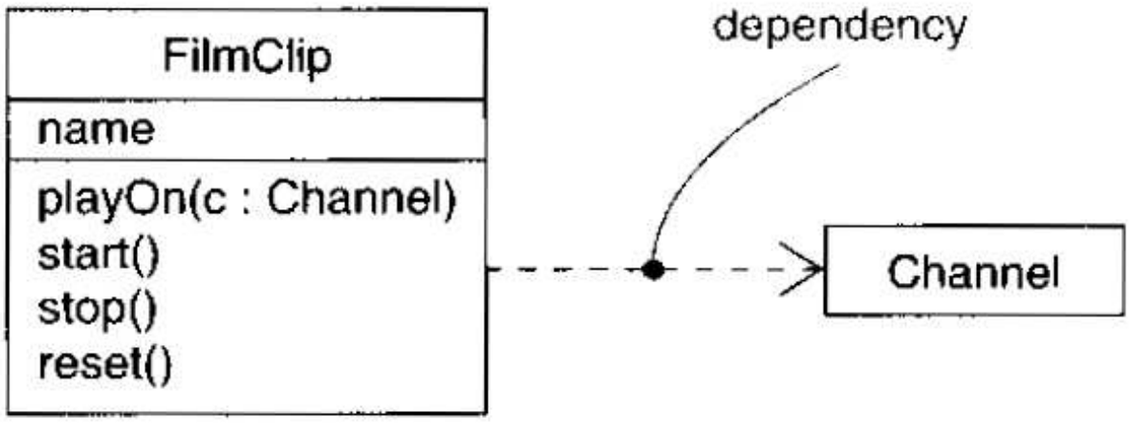
..Relaţii


Window
open()
close()
move()
display()
handleEvent()
dependency
Event
generalization
association
ConsoleWindow
DialogBox
Control

..Relaţii

Dependenţe:

- reprezintă *relaţii de utilizare*
- sunt *direcţionate*: modificarea specificării unuia îl poate afecta pe celălalt, dar nu şi reciproc

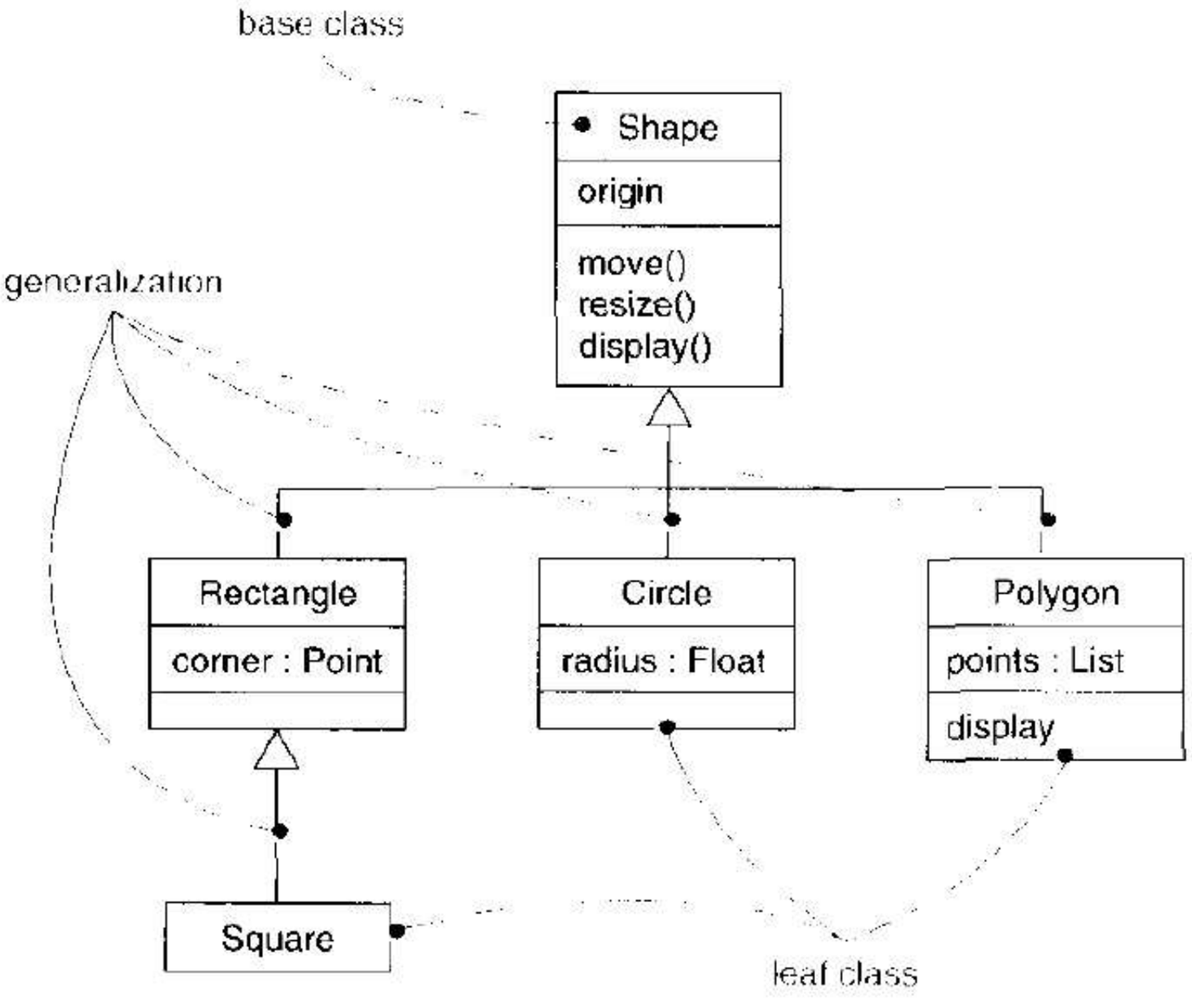


- exemplu frecvent: o clasă *A* are *un agrument* în definirea unei operaţii de timp *B*; atunci, *A* depinde de *B*

..Relaţii

Generalizări:

- relaţii dintre *general şi particular* - numită uneori *"is-a-kind-of" relation*
- lucrul/obiectul fiu poate fi substituit oriunde apare părintele, dar nu reciproc



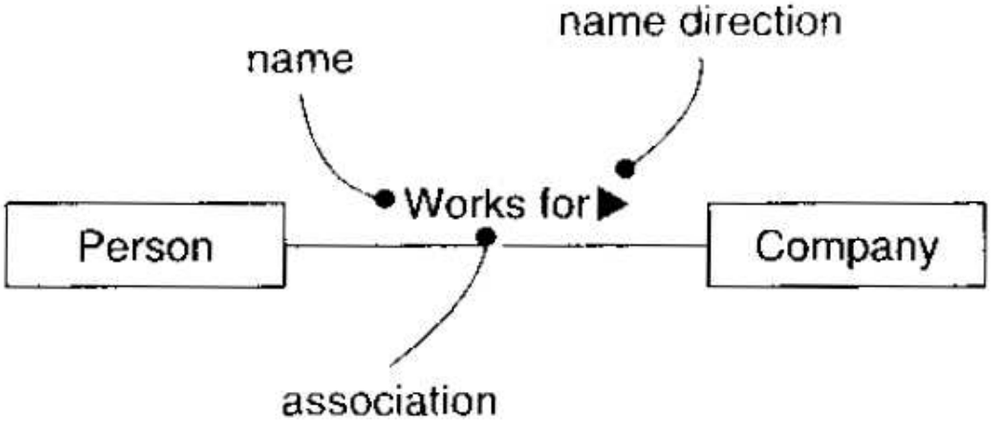
..Relaţii

Asociaţii:

Nume:

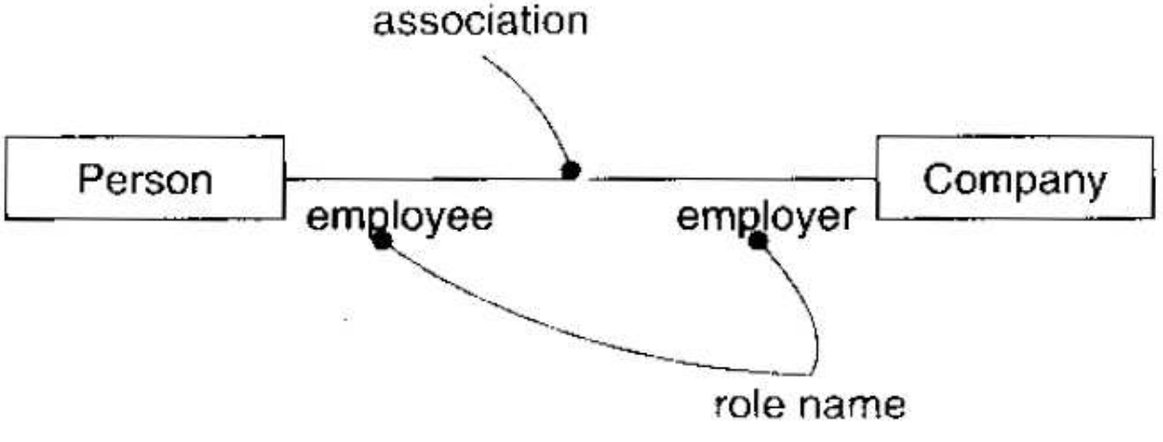
- o asociaţie are un nume

- pentru a evita ambiguitatea, se poate asocia şi o direcţie



Rol:

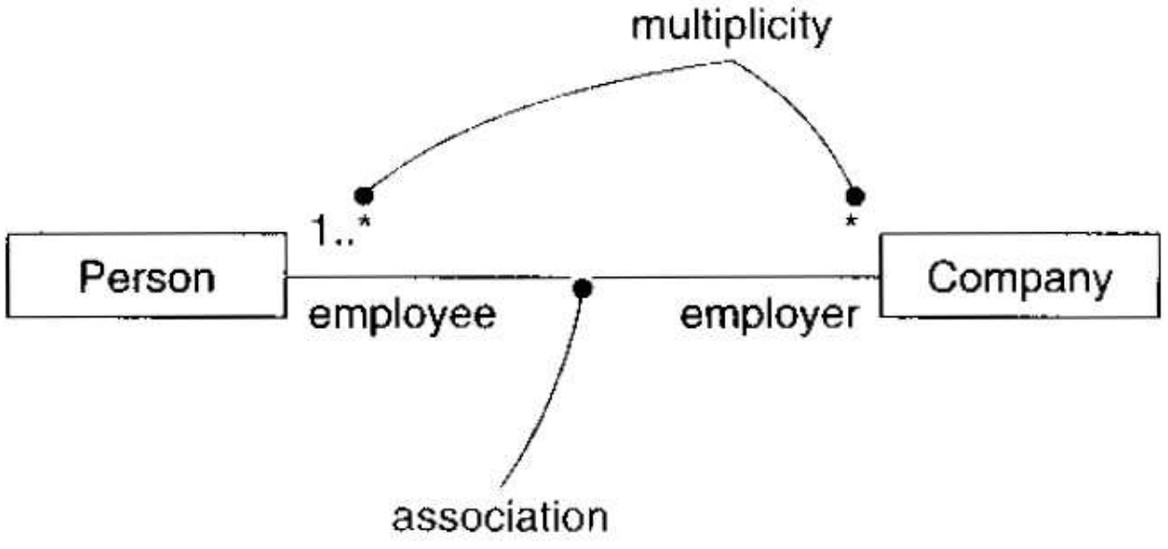
- ca alăturat, se poate explicita rolul fiecărei părţi într-o asociaţie



..Relaţii

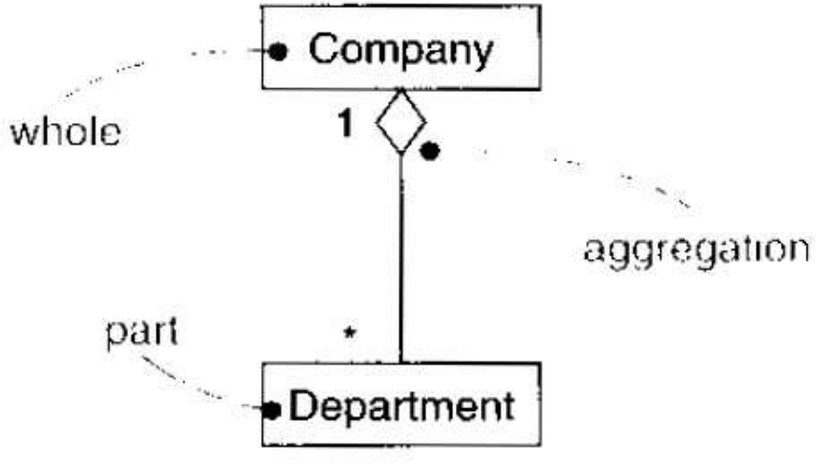
Multiplicitate:

- o asociaţie este o relaţie structurală între obiecte şi putem specifica câte obiecte participă: `*`, `(0..1)`, `(0..*)`, `(1..*)`, `3`



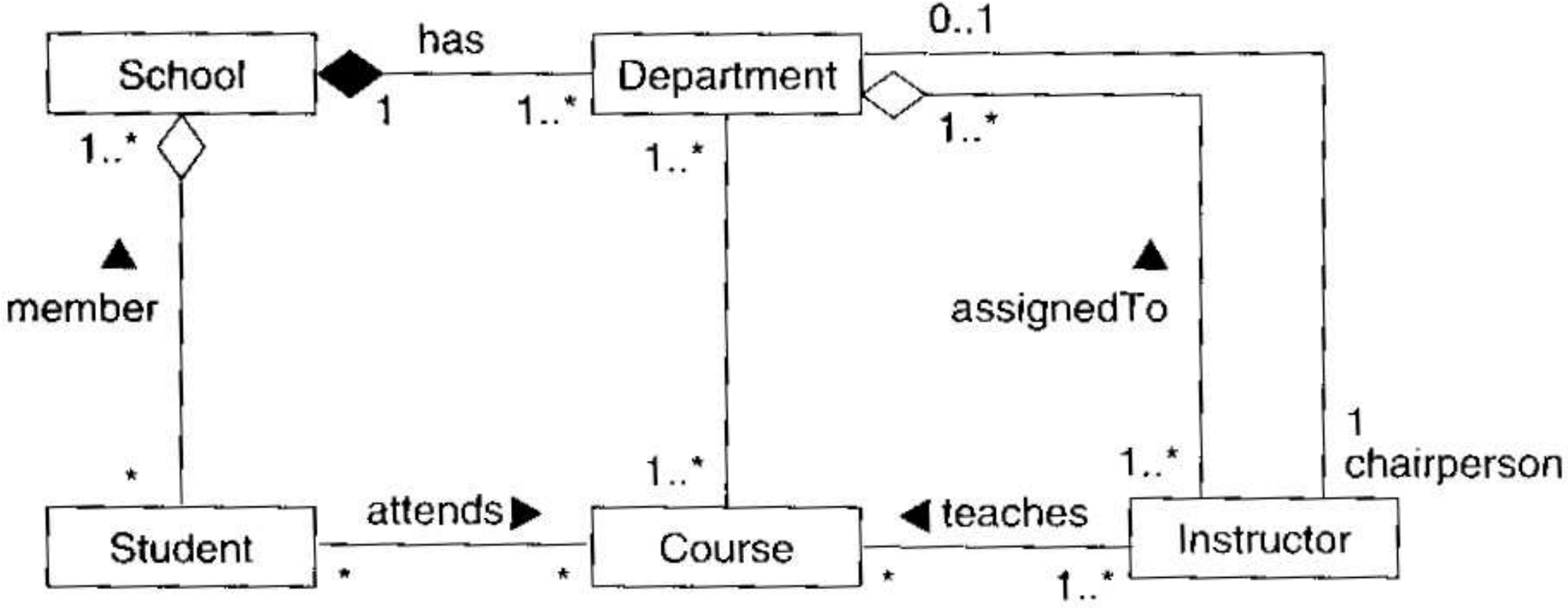
Agregare:

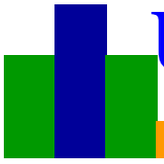
- specifică relaţia dintre *întreg* şi *părţi* (când "*întregul* este *simpla totalitate a părţilor*")



..Relaţii

Exemplu (de modelare de relaţii structurale)





UML: Modelare structurală şi comportamentală

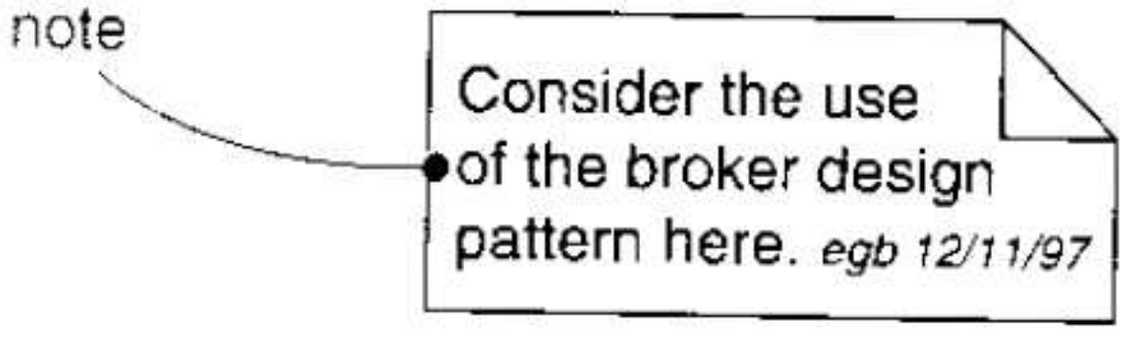
Cuprins:

- Generalităţi
- *Modelare structurală*
  - Clase
  - Relaţii
  - *Mecanisme comune*
  - Diagrame
  - Diagrame de clase
- Modelare comportamentală
- Concluzii, diverse, etc.

Mecanisme comune

Mecanisme comune - reprezintă modalităţi de a extinde limbajul.

Note - simple comentarii, fără valoare semantică



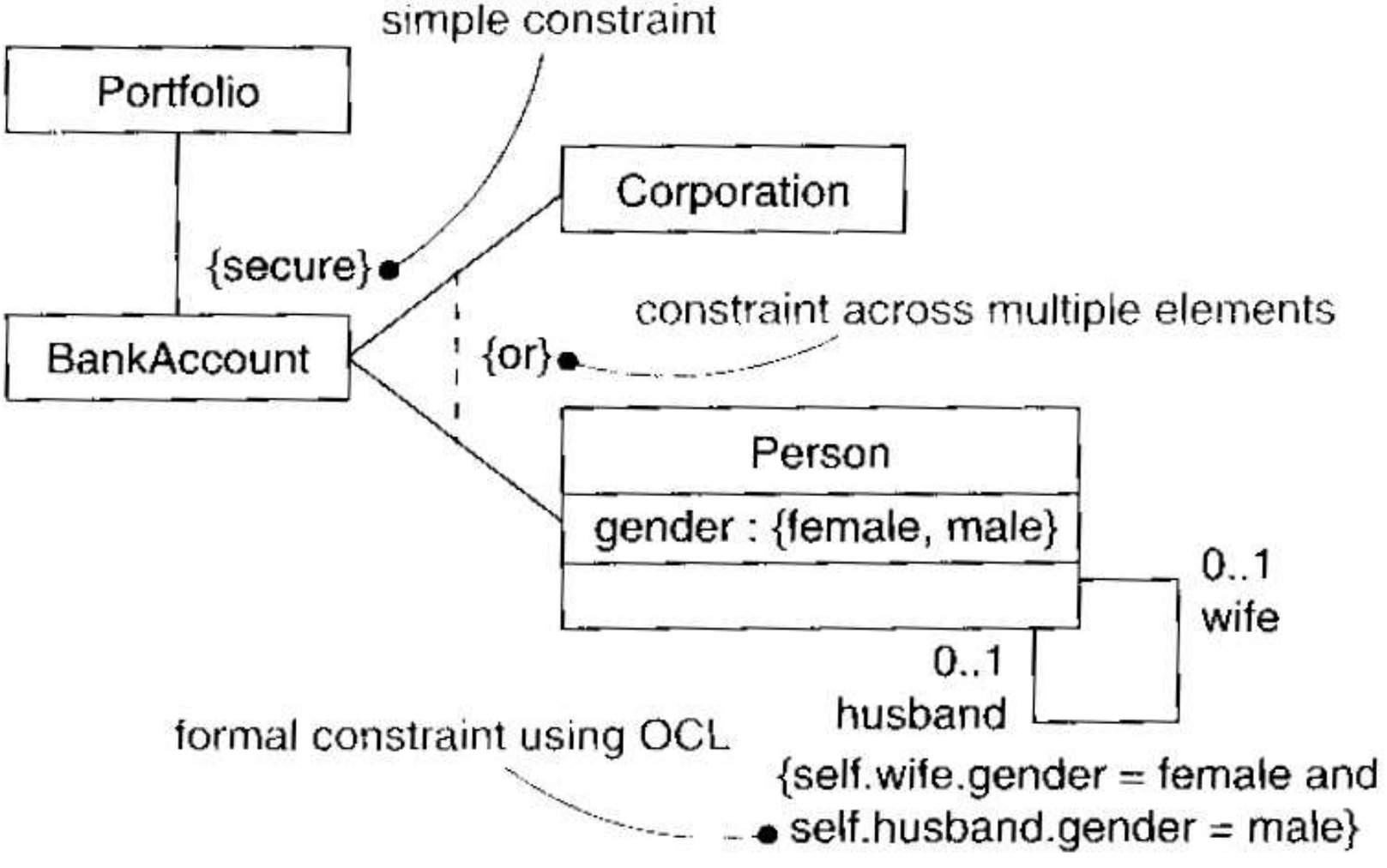
Adnotări - informaţii suplimentare, ca la asociaţii, etc.

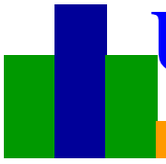
Stereotipuri - introducere de noi blocuri prmitive (meta-tipuri), noi notaţii, fără a intra în conflict cu cele utilizate

Valori ataşate - se adauga noi proprietăţi şi valori ataşate

Mecanisme comune

Constrângeri - se adaugă o semantică nouă prin astfel de constrângeri





UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- *Modelare structurală*
  - Clase
  - Relaţii
  - Mecanisme comune
  - *Diagrame*
  - Diagrame de clase
- Modelare comportamentală
- Concluzii, diverse, etc.

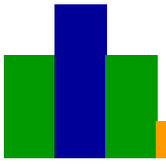
Diagrame

Diagrame:

- diagramele sunt prezentări grafice a unei mulţimi de elemente, cel mai comun reprezentate ca un *graf* cu *noduri (pentru lucruri)* şi *arce (pentru relaţii)*
- UML are 4 tipuri de diagrame pentru partea *statică* a sistemului:

  (1) diagrame de clase, (2) de obiecte, (3) de componente, şi *(4) de desfăsurare* (engl. class diagram, object diagram, component diagram, deployment diagram)
- UML are 5 tipuri de diagrame pentru partea *dinamică*:

  (1) diagrame de utilizare, (2) de secvenţe, (3) de colaborare, (4) de statechart-uri, şi *(5) de activitate* (engl. use-case diagram, sequence diagram, collaboration diagram, statechart diagram, activity diagram)



..Diagrame

Diagrame structurale:

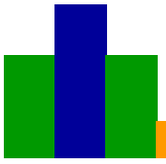
- descriu aspecte statice;
- sunt specializate pentru: (1) *clase, interfeţe, colaborări*; (2) *obiecte*; (3) *componente*; (4) *noduri*

Diagrame de clase:

- descriu clasele, interfeţele, colaborările, şi relaţiile lor
- în genere, descriu aspecte statice
- dacă sunt incluse "clase active" (care conţin procese) pot prinde şi aspecte dinamice

Diagrame de obiecte:

- prezintă o mulţime de obiecte şi relaţiile lor (statice)
- sunt similare cu cele de clase, dar acum lucrurile sunt privite *din prisma obiectelor*



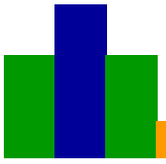
..Diagrame

Diagrame de componente:

- descriu componentele şi relaţiile lor
- în genere, *o componentă* conţine *mai multe clase, interfeţe*, şi *colaborări*

Diagrame de desfăşurare:

- prezintă o mulţime de *noduri* şi *relaţiile lor* (statice)
- ele capturează "arhitectura" sistemului
- în genere, *un nod conţine mai multe componente*



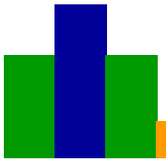
..Diagrame

Diagrame comportamentale:

- descriu aspecte dinamice;
- sunt specializate pentru: (1) organizarea comportamentului; (2) mesaje şi ordinea lor în timp; (3) organizarea structurală şi comunicarea send-receive; (4) evoluţia ca reacţie la semnale; (5) evoluţie din activitate în activitate

Diagrame de utilizare (use-care diagrams)

- descriu actorii şi modalităţile lor de interacţiune
- sunt folosite, în genere, pentru a organiza şi modela comportamentul sistemului



..Diagrame

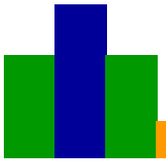
Diagrame de secvenţe:

- în esenţă sunt MSC-uri
- sunt diagrame de interacţie care descriu mulţimi de obiecte care comunica prin mesaje şi *ordinea mesajelor în timp*

Diagrame de colaborare:

- prezintă *organizarea structurală* a obiectelor şi comunicările send-receive dintre ele
- sunt *semantic echivalente* cu diagramele de secvenţe (aici *ordinea în timp* a mesajelor este *ne-explicită*, rezultând din etichetele temporale ataşate mesajelor)

Notă: Diagramele de secvenţe şi cele de colaborări se mai numesc diagrame de interacţie.



..Diagrame

Diagrame de statechart-uri:

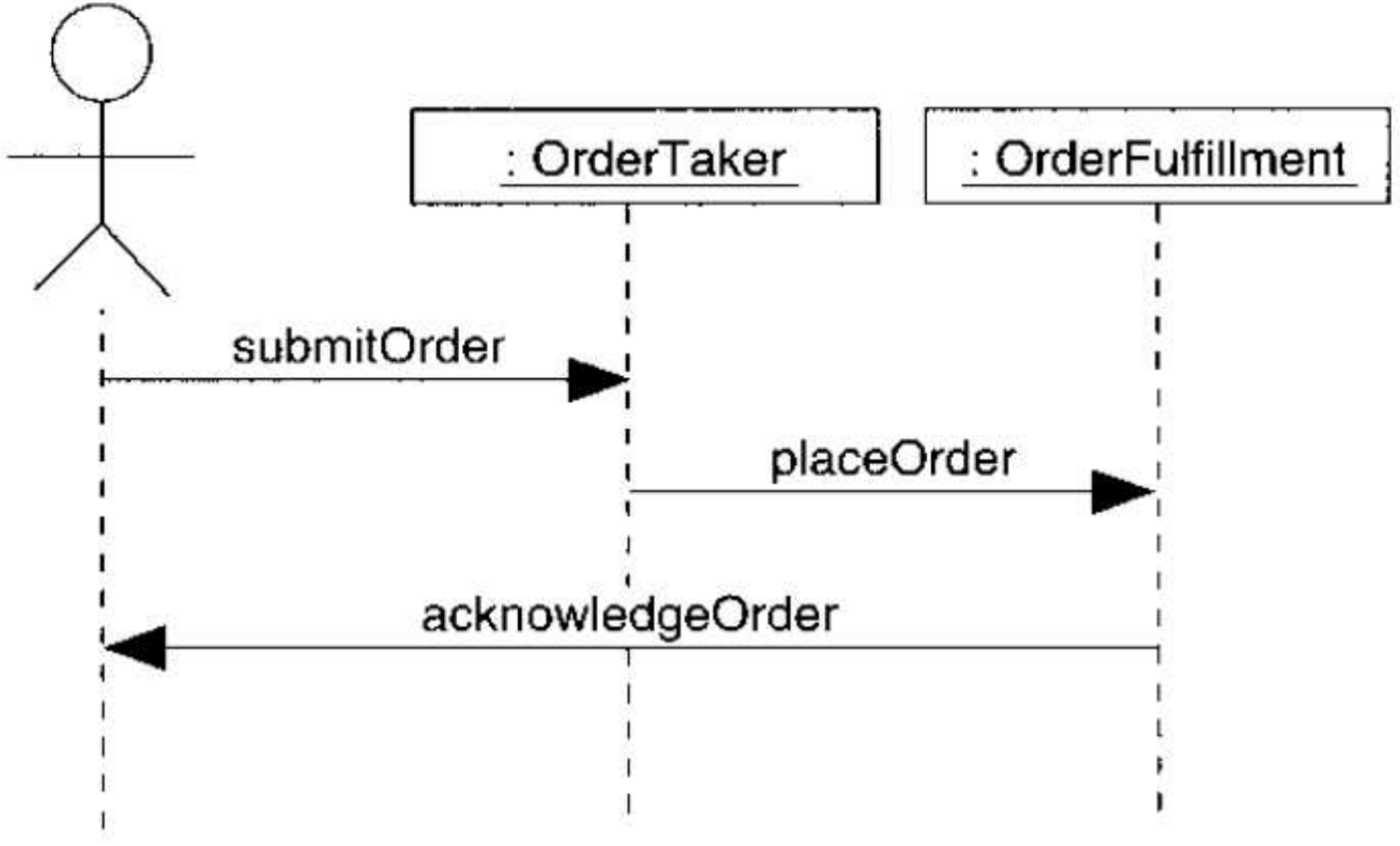
- descriu evoluţia cu *maşini de stări* finite folosind: *tranziţii, evenimente, şi activităţi*
- se folosesc pentru a descrie comportamentul interfeţelor, al claselor, ori al colaborărilor
- descriu evoluţia sistemului *din eveniment în eveniment*

Diagrame de activităţi:

- prezintă comportamentul ca flux *de la o activitate la alta*
- ele prezintă evoluţia secvenţială ori paralelă a activităţilor şi, totodată, obiectele asupra cărora acţionează
- sunt *semantic echivalente* cu diagramele de statechart-uri (dar, aici, accentul se pune pe funcţiile sistemului, nu pe reacţia la evenimente)

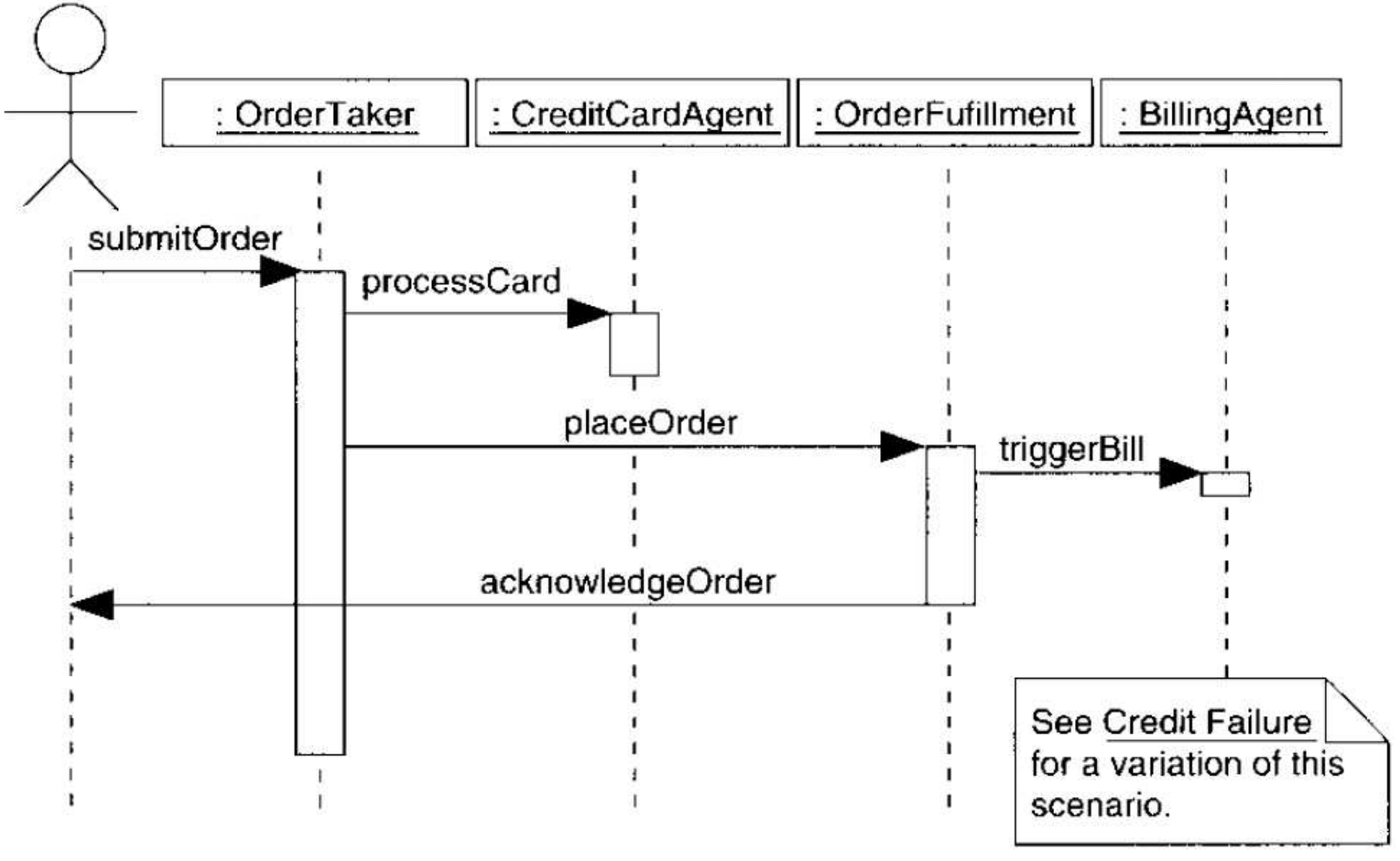
..Diagrame

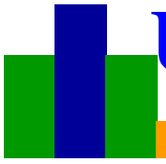
Exemplu - o diagrama de interacţie *mai abstractă*



..Diagrame

... şi o diagrama de interacţie *mai concretă*

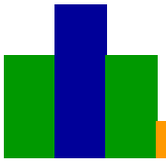




UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- *Modelare structurală*
  - Clase
  - Relaţii
  - Mecanisme comune
  - Diagrame
  - *Diagrame de clase*
- Modelare comportamentală
- Concluzii, diverse, etc.



Diagrame de clase

Diagrame de clase

- sunt diagrame ca şi celelalte, dar se *focalizează pe clase*

Conţin:

- *clase*
- *interfeţe*
- *colaborări*
- *relaţii de dependenţă, generalizare, ori asociere*

Se utilizează pentru:

- a modela *vocabularul sistemului*
- a modela *simple colaborări*
- a modela *baze de date*

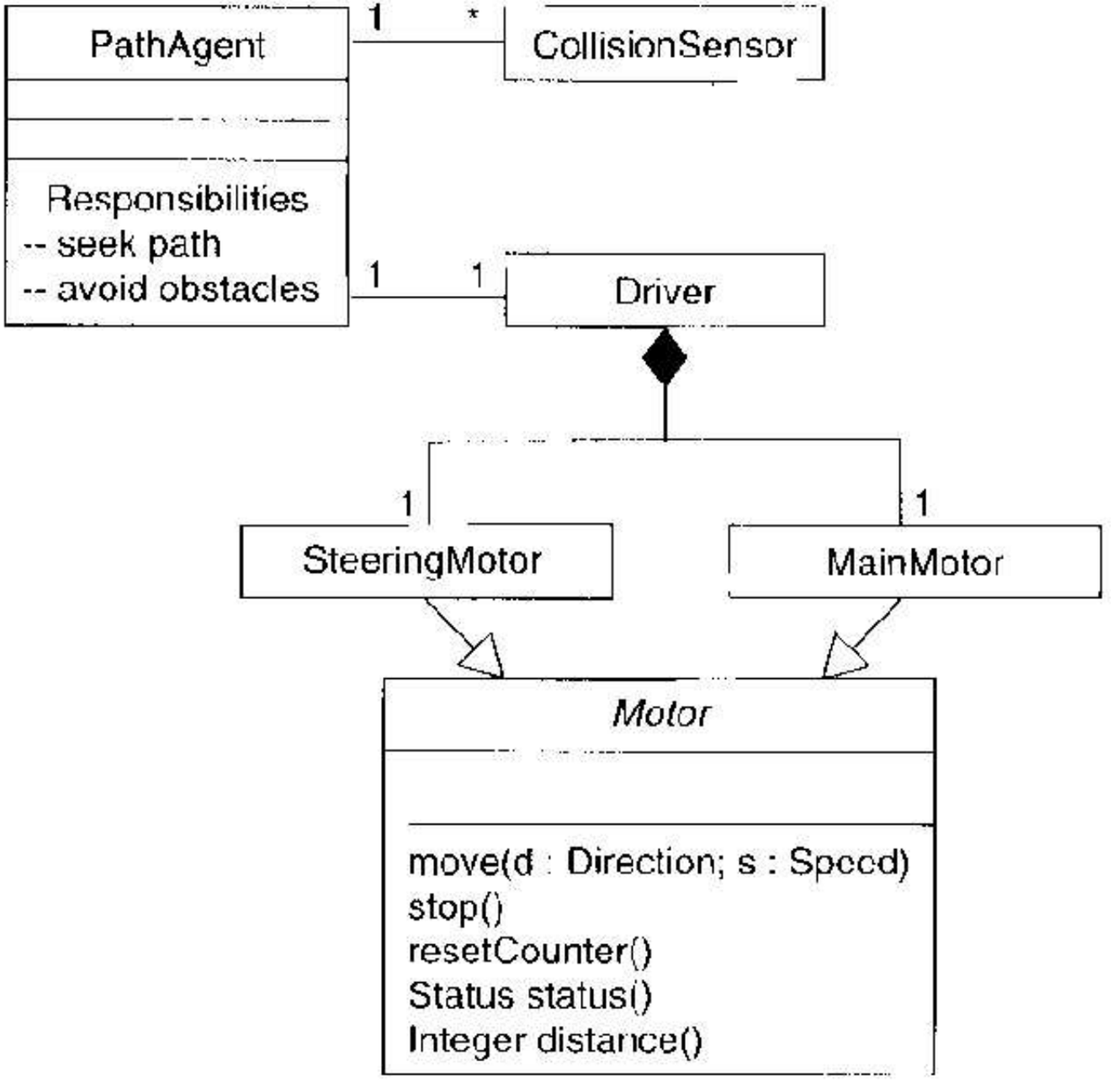
Exemplu:

o diagramă de clase

Diagrame de clase

Exemplu:

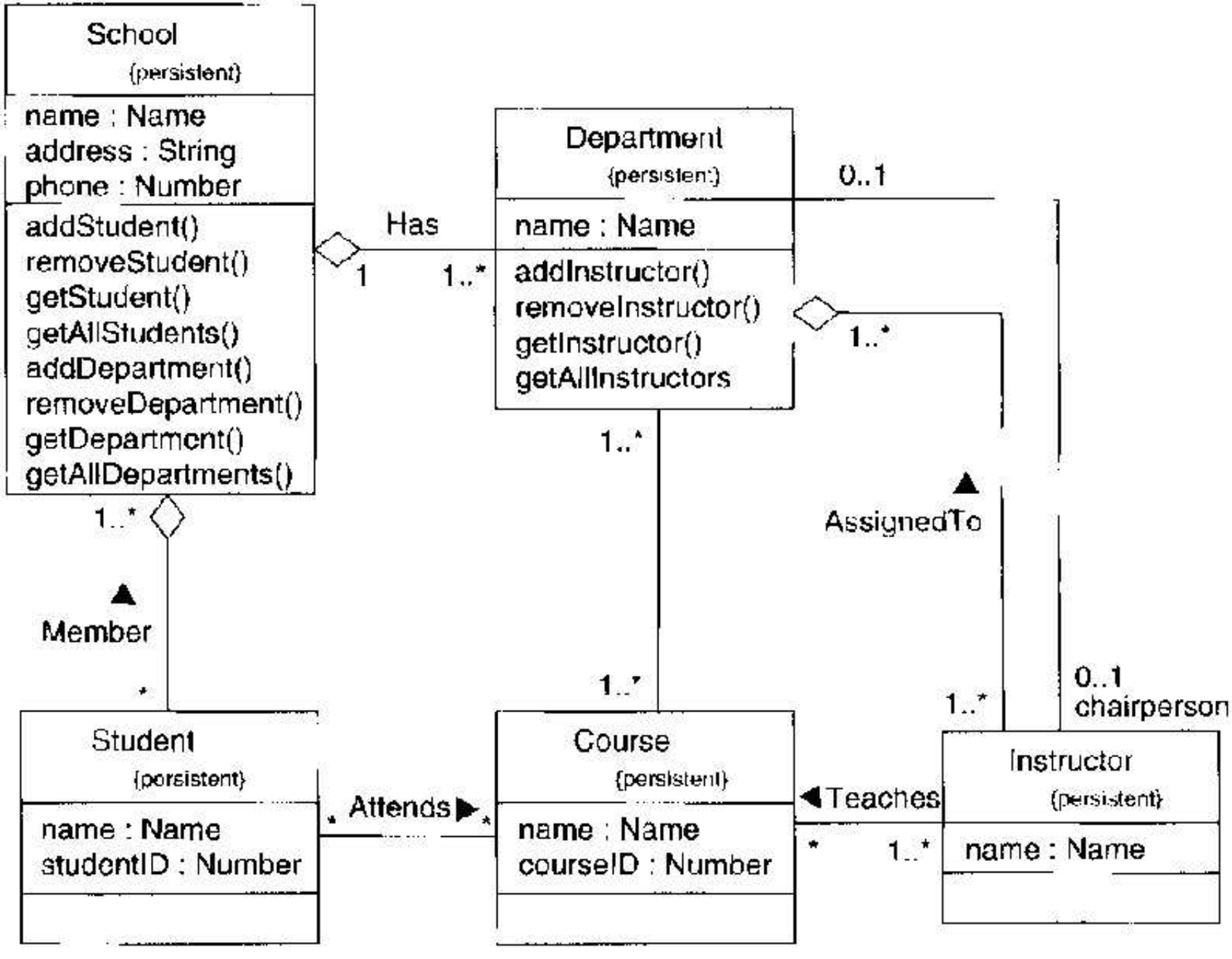
o simplă colaborare

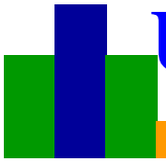


Diagrame de clase

Exemplu:

o schemă logică pentru date





UML: Modelare structurală şi comportamentală

Cuprins:

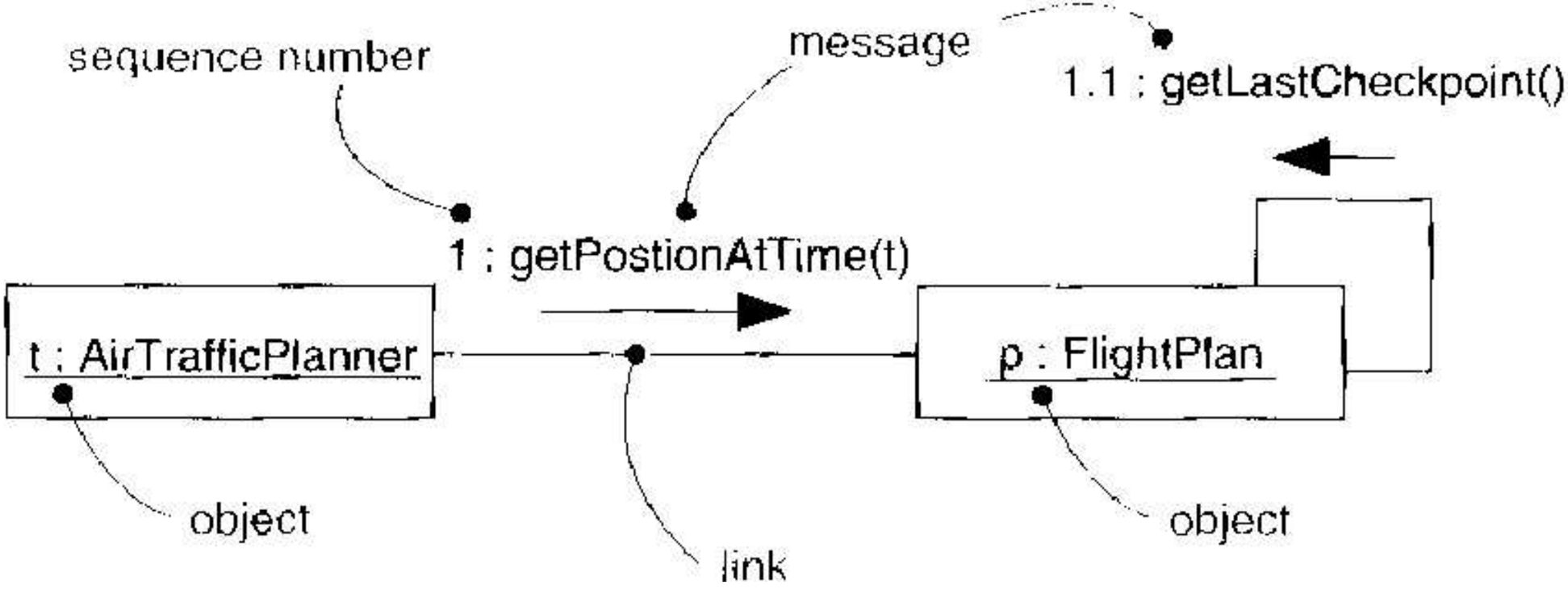
- Generalităţi
- Modelare structurală
- *Modelare comportamentală*
  - *Interacţii*
  - Use cases
  - Diagrame “use cases”
  - Diagrame de interacţie
  - Diagrame de activităţi
- Concluzii, diverse, etc.

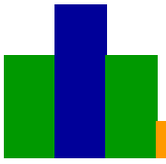
Interacţii

Interacţii:

- elementele principale folosite la modelarea interacţiilor sunt: *(1) mesaje, (2) link-uri*, şi *(3) numere de secvenţializare*

Exemplu:





..Interacţii

Context:

- interacţia depinde de "contextul" în care se studiază (clasă, sub-sistem, sistem)
- poate fi descrisă pe diverse *paliere de abstracţie*

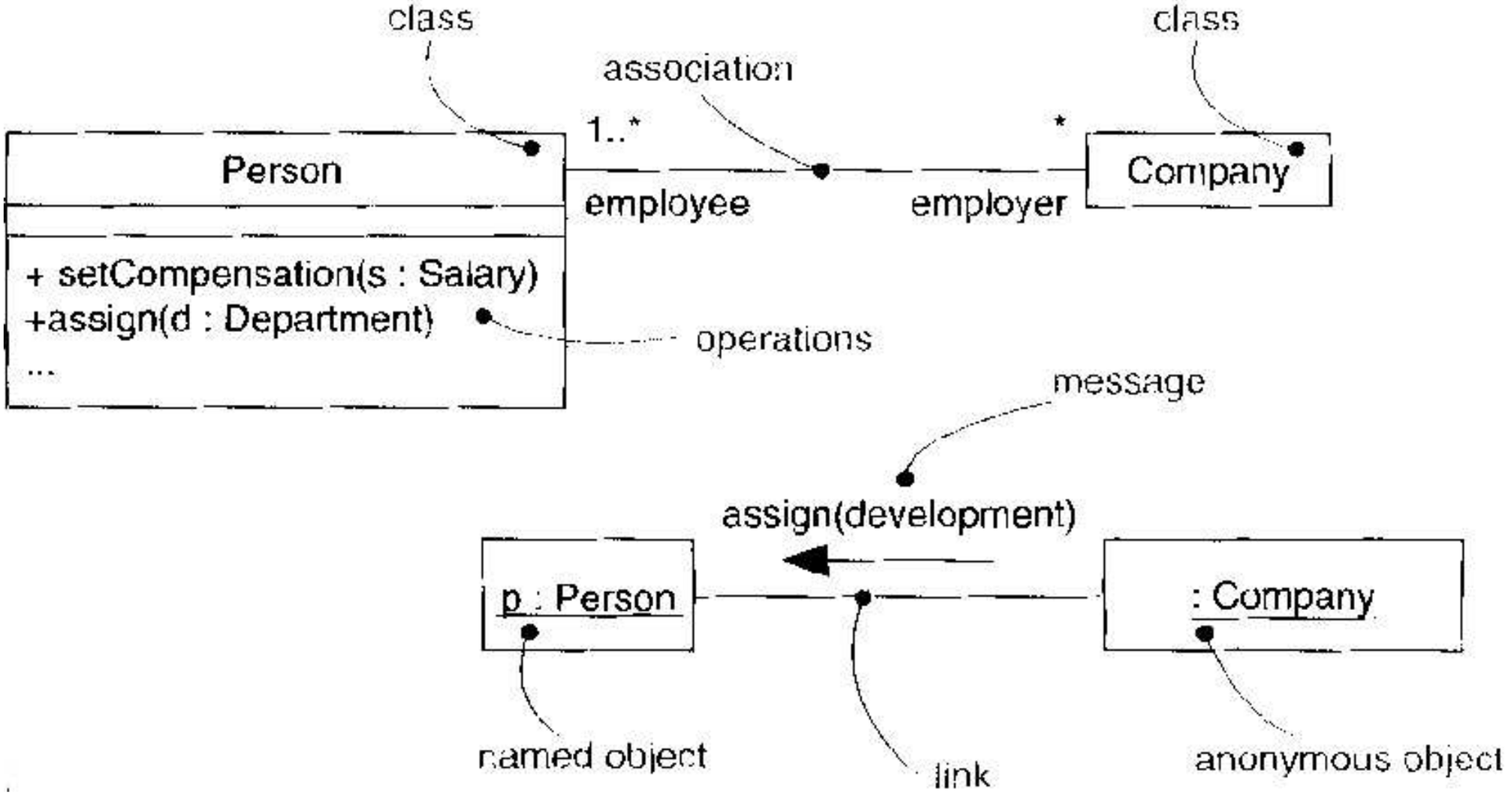
Obiecte şi roluri:

- obiectele care participă pot fi *concrete*, ori *lucruri generice* (e.g., o persoană reprezentativă pentru clasa `Person`)
- apar *instanţe de clase, componente, noduri, ori use-case-uri* (chiar şi "clasele abstracte" pot fi folosite, instanţele lor fiind "generice")

..Interacţii

Link-uri:

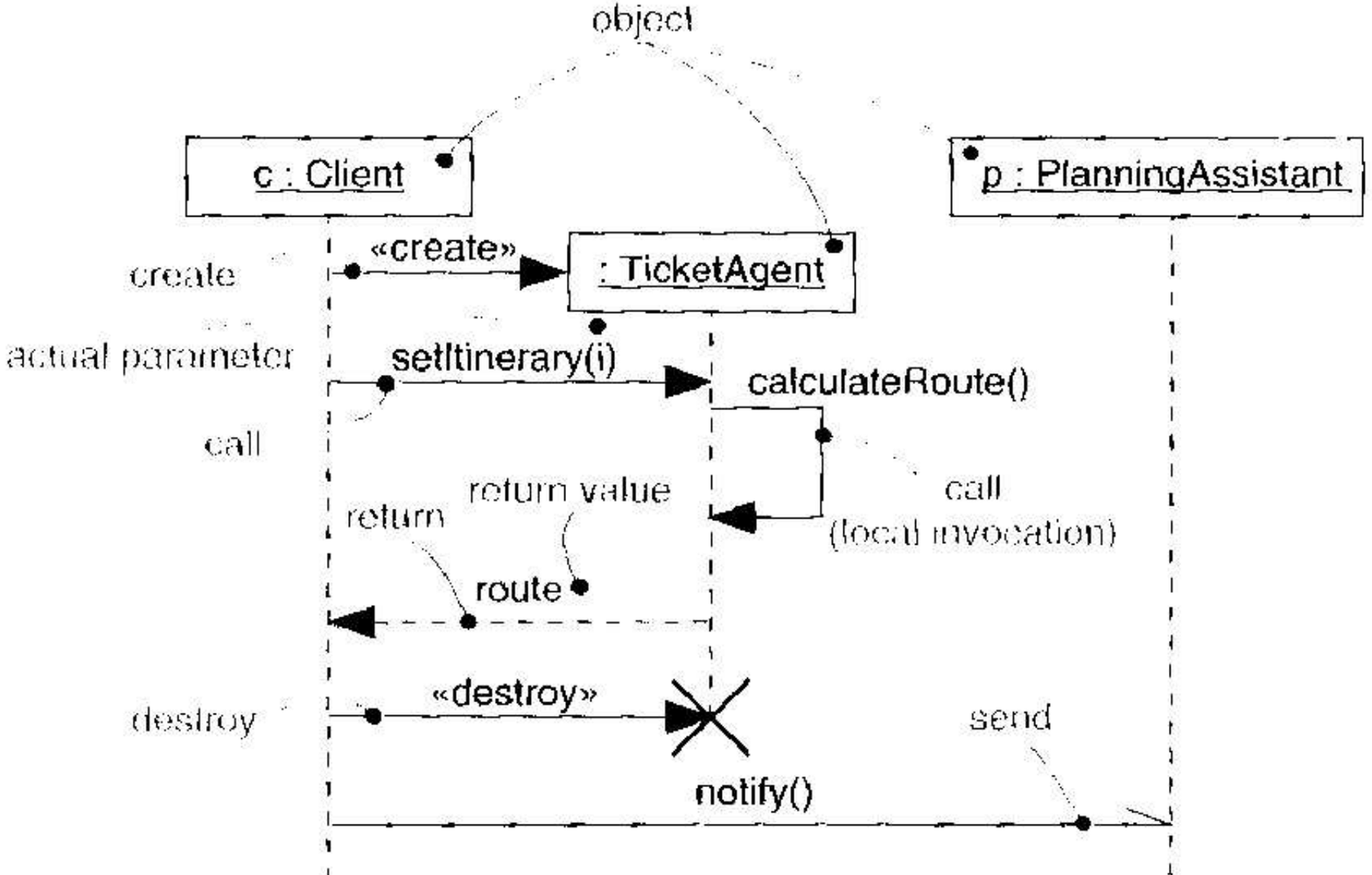
- sunt conexiuni semantice între obiecte; în genere sunt asociaţii



..Interacţii

Mesaje:

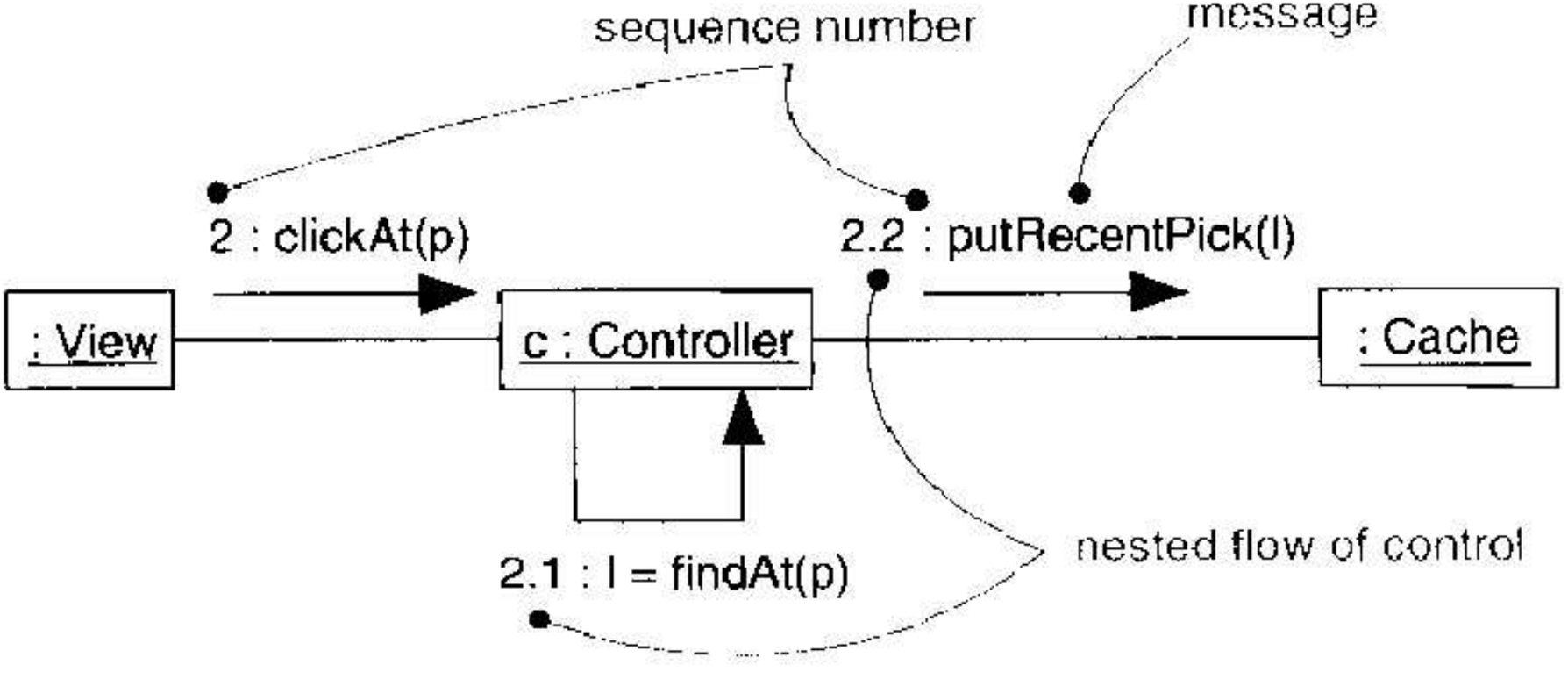
- în afara mesajelor uzuale, pot fi mesaje de: *call, return, send, create, destroy*



..Interacţii

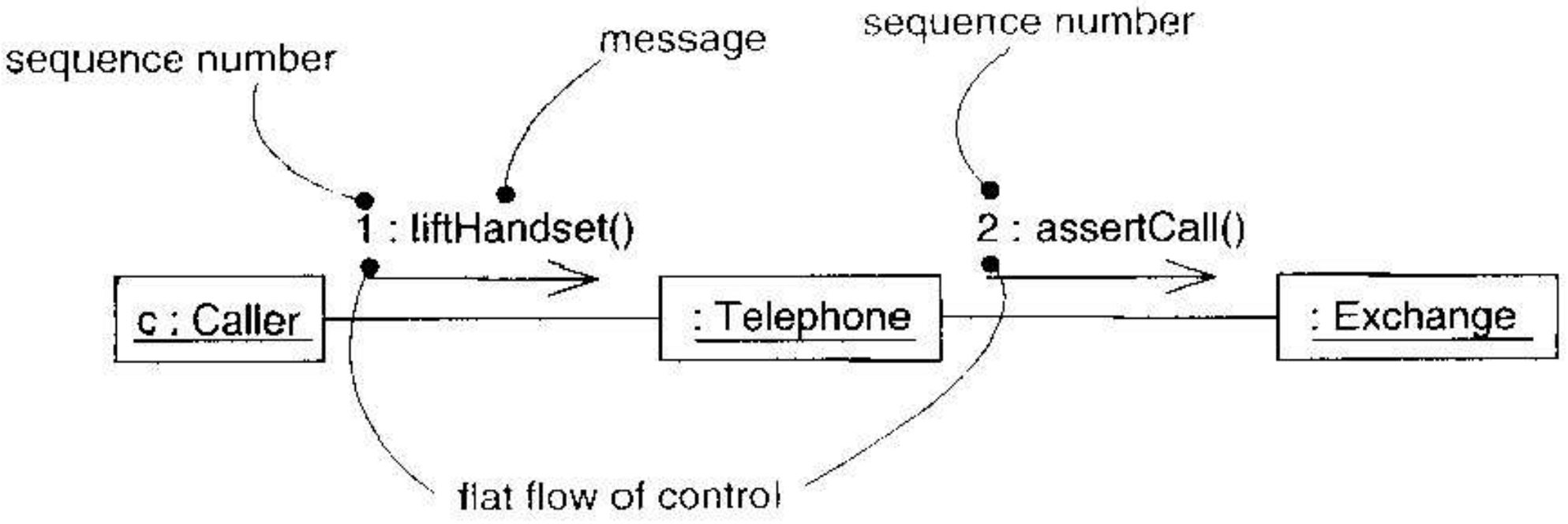
Secvenţializare:

- secvenţializarea este un mecanism de a determina *ordinea mesajelor în timp*
- se pot folosi notaţii *ierarhice* (secvenţe cuibărite/săgeţi cu triunghiuri pline) ori *plate* (numere naturale/săgeţi simple)



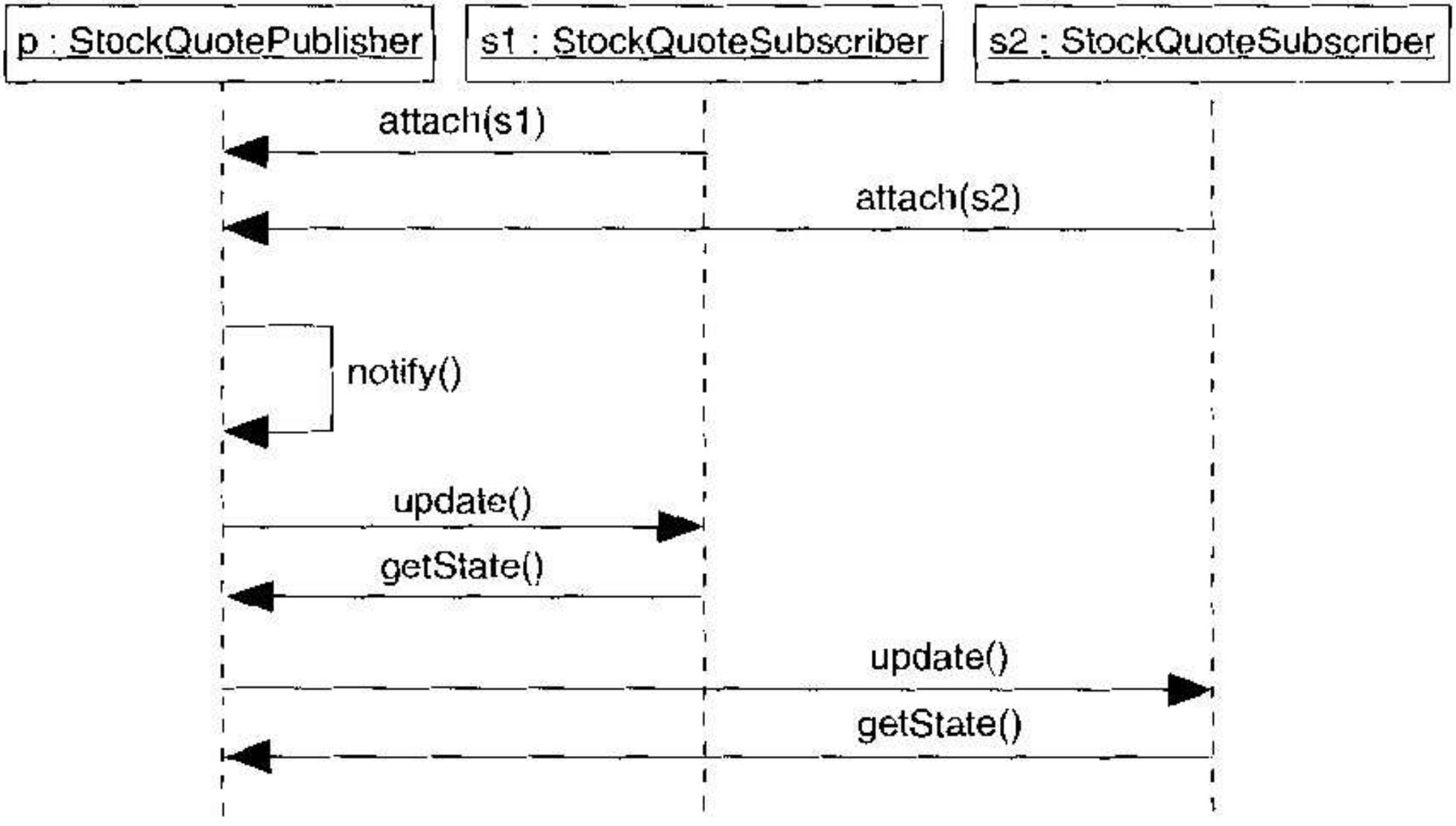
..Interacţii

(exemplu de secvenţializare cu notaţie plată)



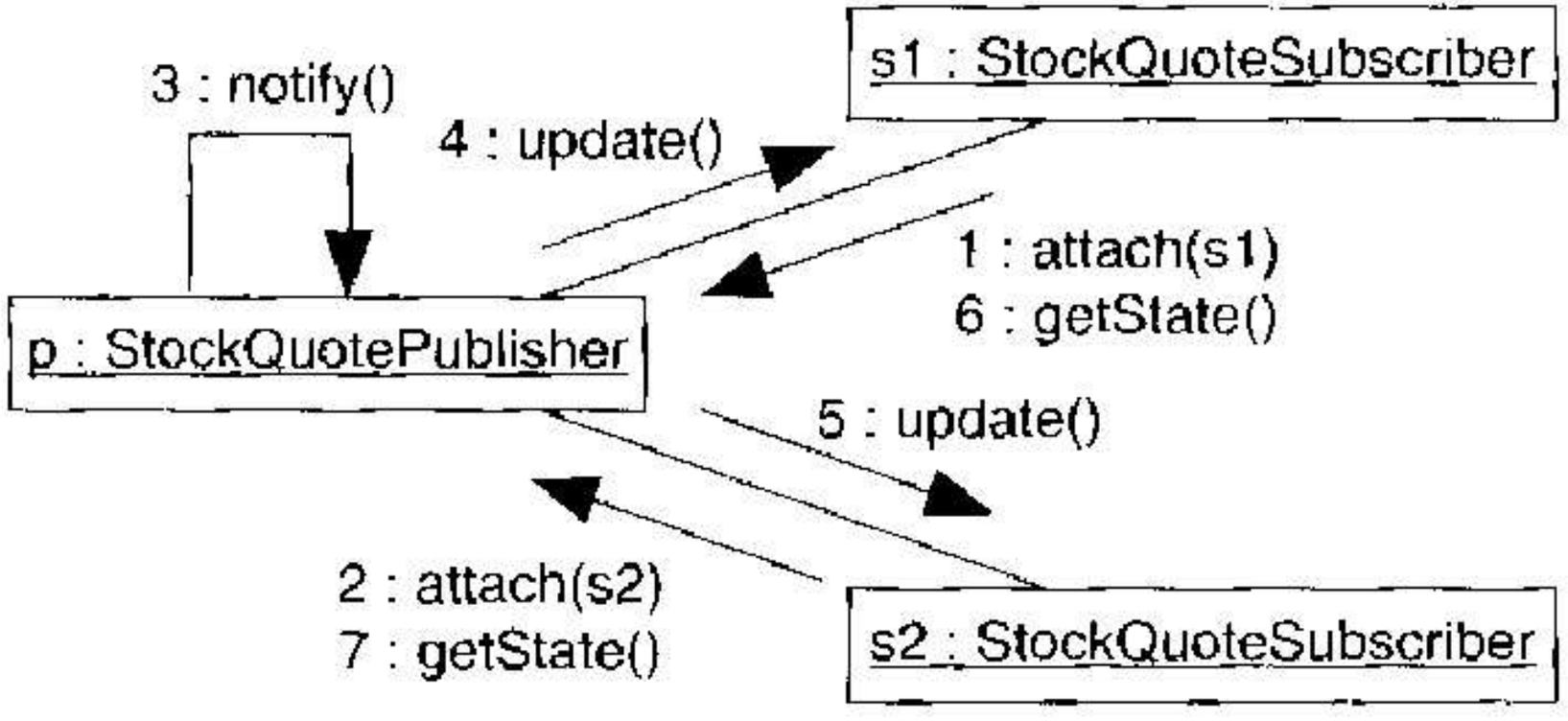
..Interacţii

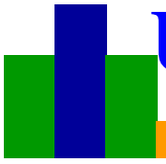
Exemplu: *echivalenţa* între secvenţializarea timp MSC ...



..Interacţii

...şi cea din diagramele de colaborări

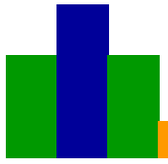




UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- Modelare structurală
- *Modelare comportamentală*
  - Interacţii
  - *Use-cases şi diagrame "use cases"*
  - Diagrame de interacţie
  - Diagrame de activităţi
- Concluzii, diverse, etc.



Use-cases

Use-cases:

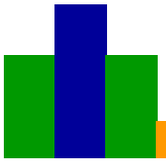
- descriu *secvenţe de acţiuni (cu varinate)* pe care un sistem le poate face rezultând ceva de valoare pentru un *actor*
- un actor poate fi *om, hardware, alt sistem*, etc.

Nume:

- orice use-case are nu nume (simplu, ori cu cale)
- un use-case se desenează ca o *elipsă*

Use-cases vs. actori:

- un actor reprezintă un *rol* (ori un set de roluri) coerent pe care un user îl are în interacţia cu use-case-ul
- un actor *nu* face parte din sistem
- un actor este desenat schematic ca un om



Use-cases

Use-cases vs. flux de evenimente:

- un use-case descrie *ce* se face, *nu cum*
- sunt descrise de use-cases atât *evoluţiile normale*, cât şi cele *excepţionale*

Use-cases vs. scenarii:

- un use-case descrie un *set de scenarii*, nu doar un scenariu

Use-cases vs. colaborări:

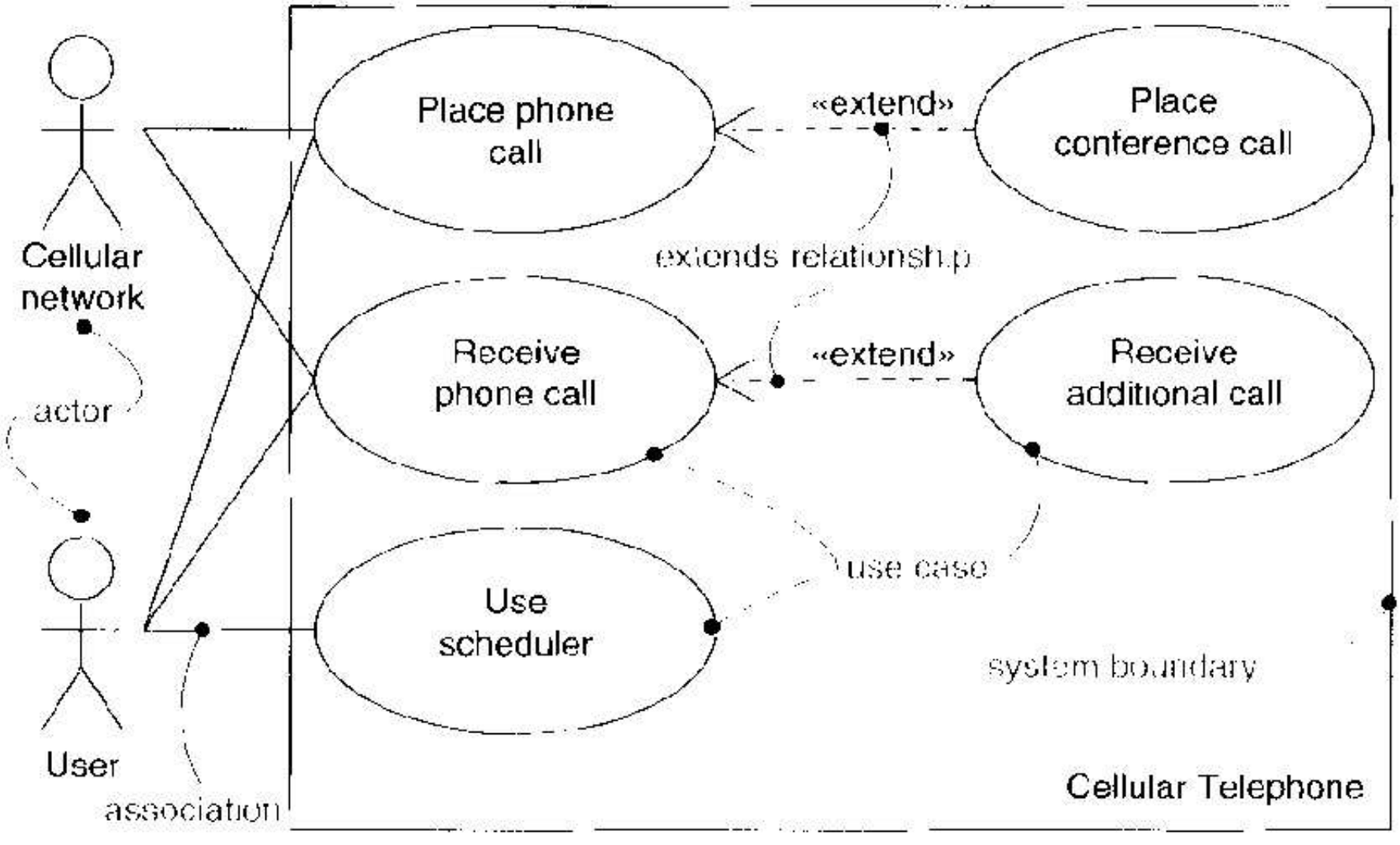
- use-cases descriu ce, nu cum şi sunt *implementate prin colaborări* între diverse elemente

Organizarea use-case-urilor:

- se pot folosi relaţii de *generalizare, includere*, ori *extensie* între use-case-uri

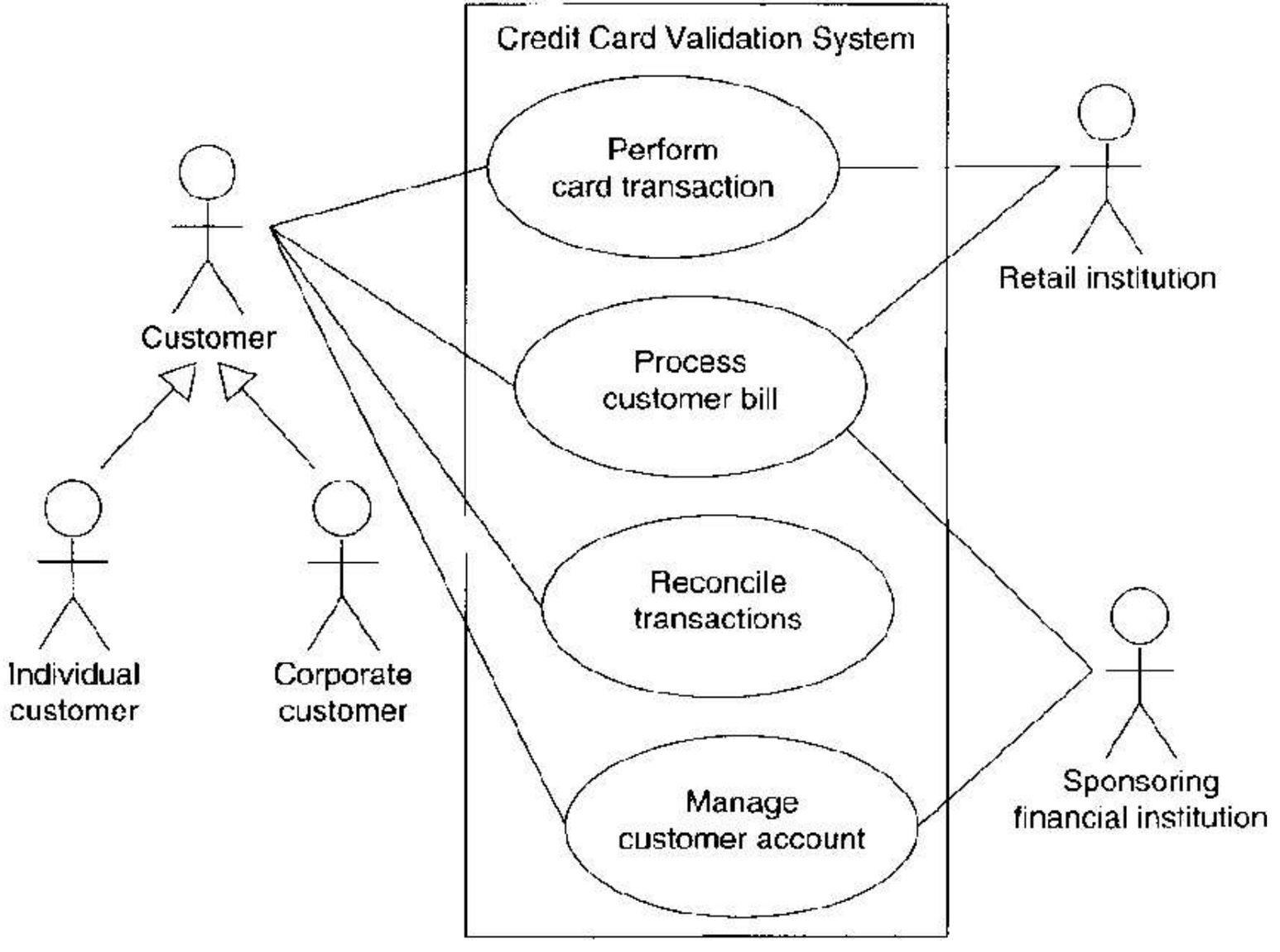
Diagrame use-case

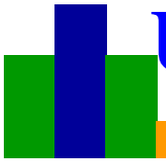
Exemplu - o diagramă use-case



..Diagrame use-case

Exemplu - modelarea contextului de folosire a unui sistem

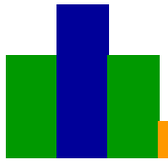




UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- Modelare structurală
- *Modelare comportamentală*
  - Interacţii
  - Use-cases şi diagrame “use cases”
  - *Diagrame de interacţie*
  - Diagrame de activităţi
- Concluzii, diverse, etc.



Diagrame de interacţie

Diagrame de interacţie

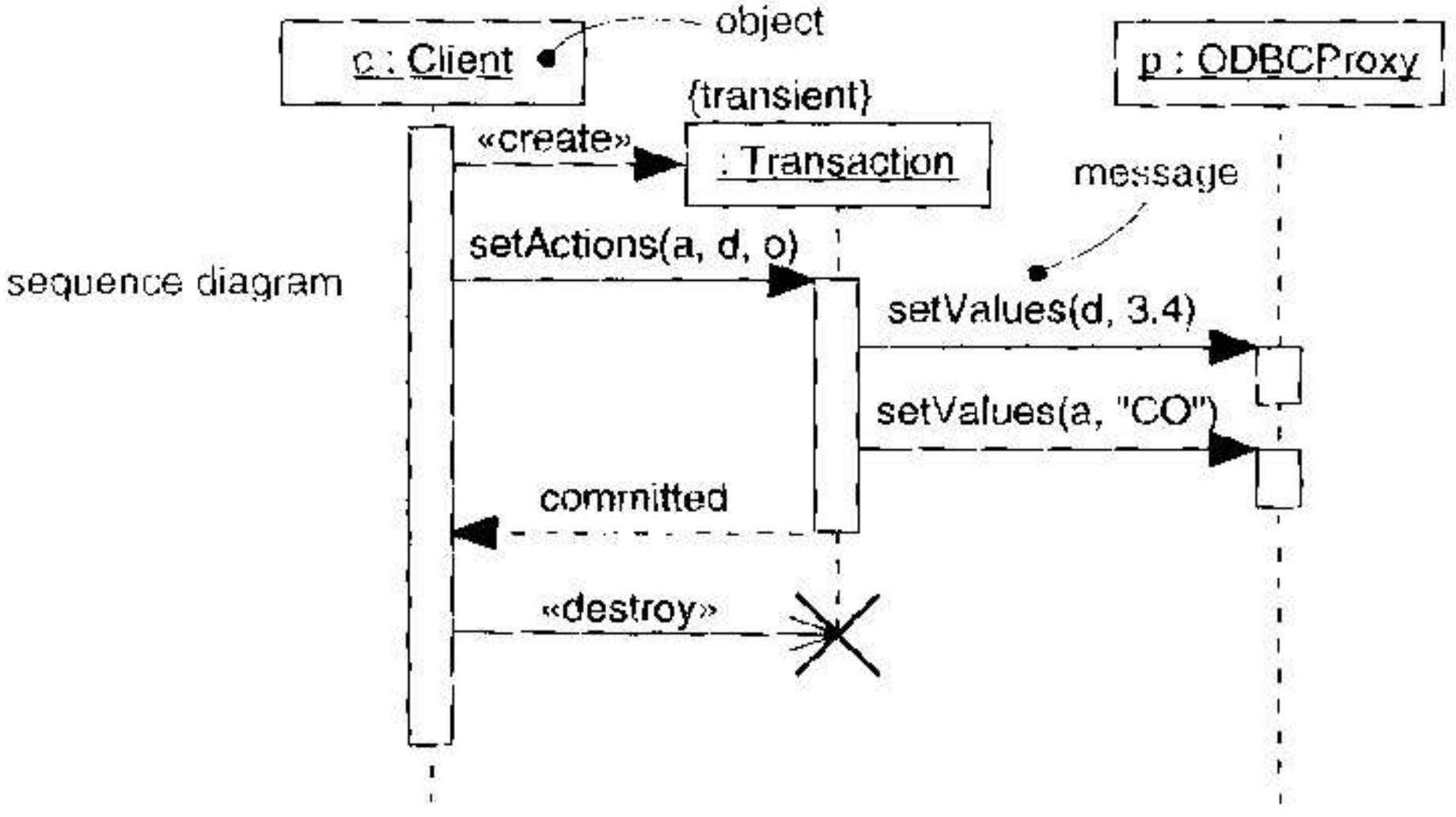
- o *diagramă de interacţie* descrie o interacţie între obiecte, precum şi relaţiile lor, ori mesajele trimise între ele
- sunt de două tipuri: *diagrame de secvenţe* (tip MSC) şi *diagrame de colaborări*

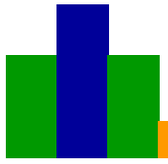
Diagramele de secvenţe

- se pune *accent pe timp*
- ca şi MSC-urile, se reprezintă grafic cu *linii verticale pentru evoluţia în timp* a obiectelor, şi *linii orizontale pentru mesaje*

..Diagrame de interacţie

Exemplu - diagramă de secvenţe





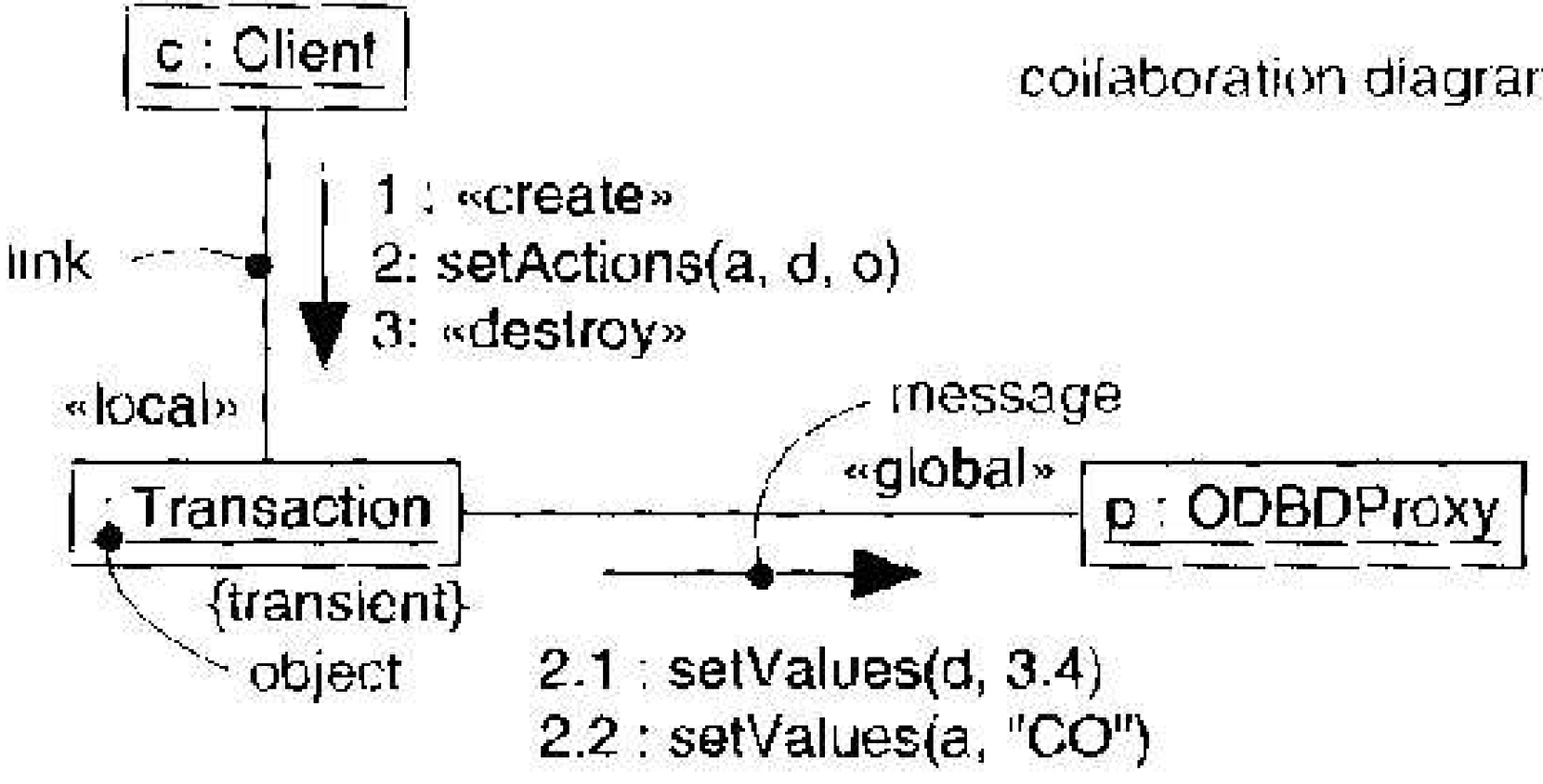
..Diagrame de interacţie

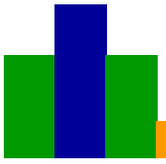
Diagramele de colaborări

- se pune accentul pe *organizarea structurală* a obiectelor care comunică prin mesaje
- se desenează plat cu *noduri pentru obiecte* şi *arce pentru mesaje*, ordinea mesajelor în timp fiind indicată prin *etichete temporale*

..Diagrame de interacţie

Exemplu - diagramă de colaborări





..Diagrame de interacţie

Echivalenţa

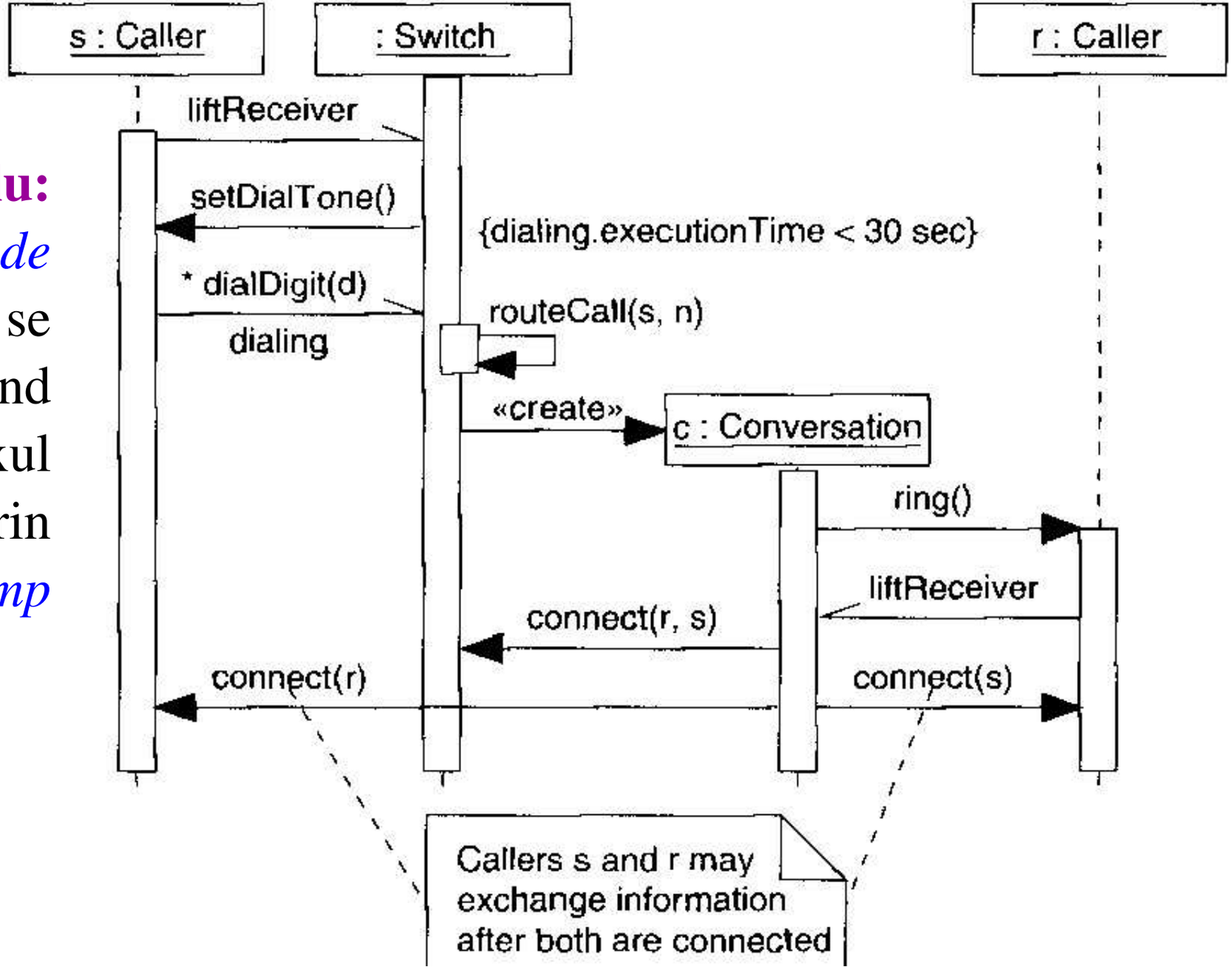
- *Fapt: Diagramele de secvenţe şi diagramele de colaborări sunt semantic echivalente.*

Anume,

- există transformări simple care ne duc dintr-o reprezentare în alta, fără a pierde informaţie

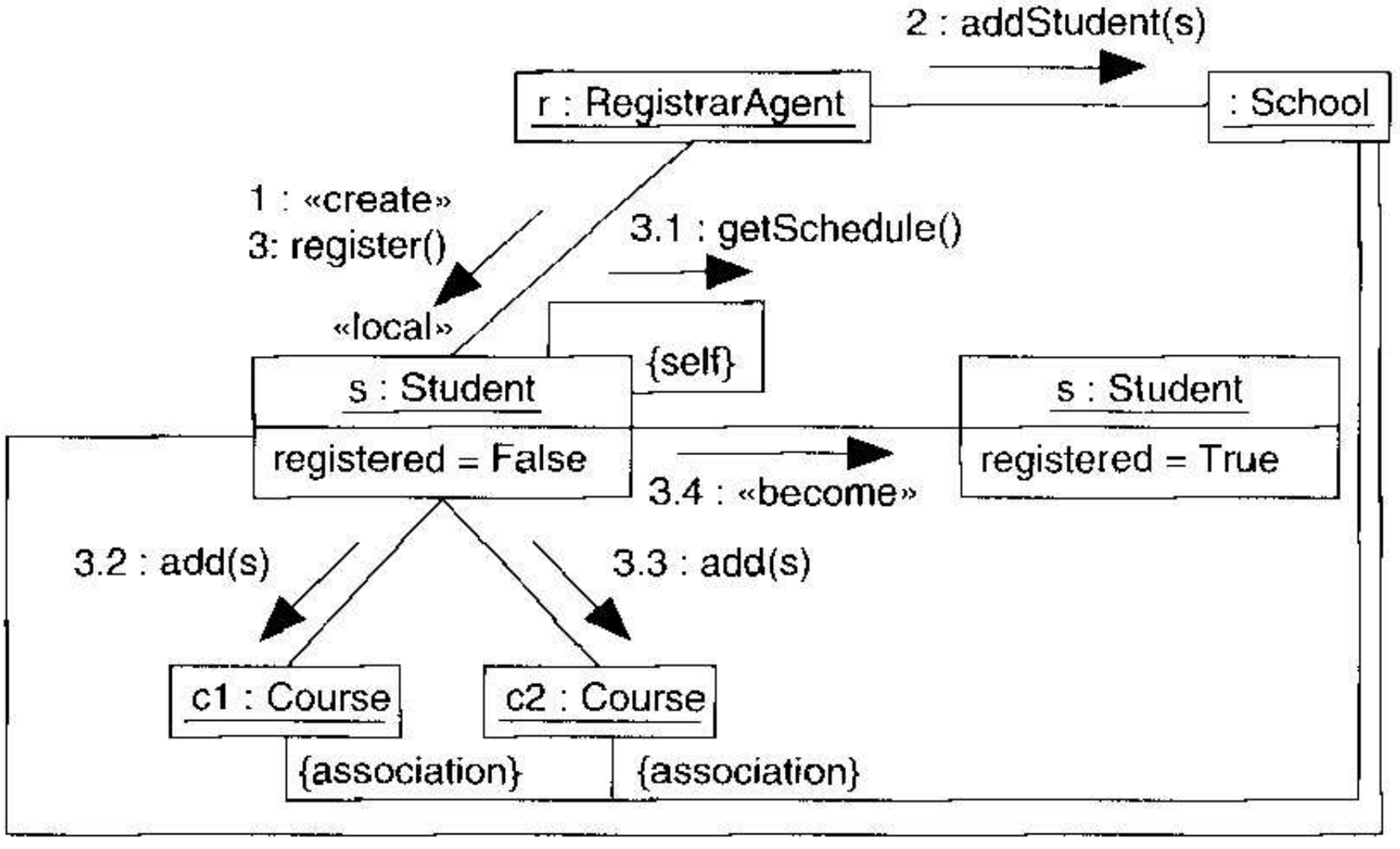
..Diagrame de interacţie

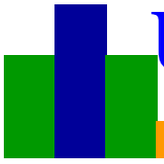
Exemplu:
diagramele de secvenţe se folosesc când modelăm fluxul de control prin *ordinea în timp*



..Diagrame de interacţie

Exemplu - *diagramele de colaborări* se folosesc când modelăm fluxul de control prin *organizarea activităţii*

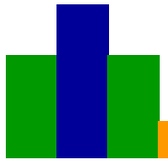




UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- Modelare structurală
- *Modelare comportamentală*
  - Interacţii
  - Use-cases şi diagrame “use cases”
  - Diagrame de interacţie
  - *Diagrame de activităţi*
- Concluzii, diverse, etc.



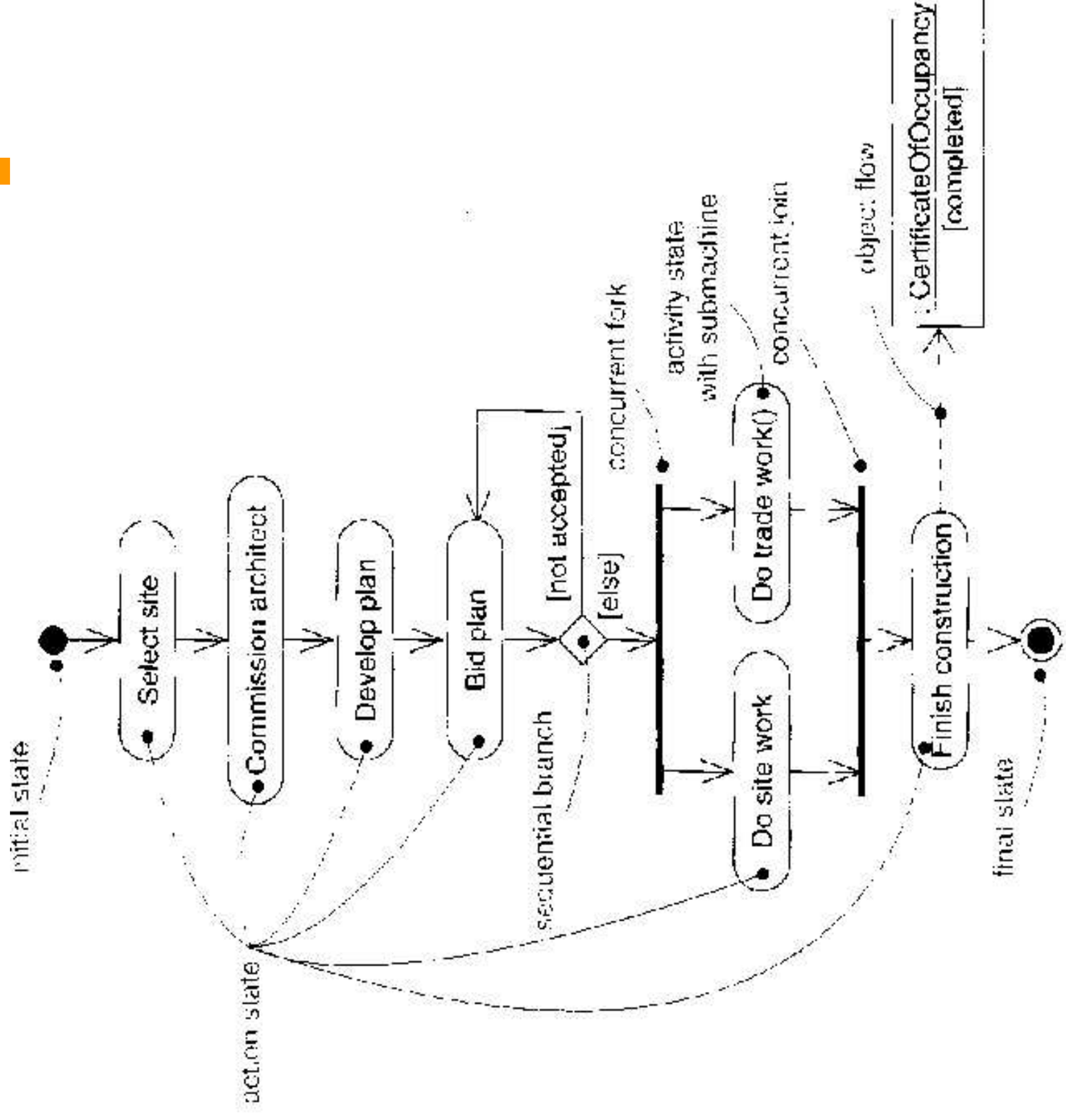
Diagrame de activităţi

Diagrame de activităţi

- o *diagramă de activităţi* descrie fluxul *din activitate în activitate*
- o activitate este o execuţie *ne-atomică, continuă*, constând din *diverse acţiuni*
- *acţiunile* pot fi diverse, precum: *apelul unei alte operaţii, trimitere de semnal, crearea ori distrugerea de obiecte, paşi de calcul local*, etc.
- grafic, o diagramă de activităţi este o colecţie de *noduri şi arce*

Exemplu:

o diagramă de activităţi

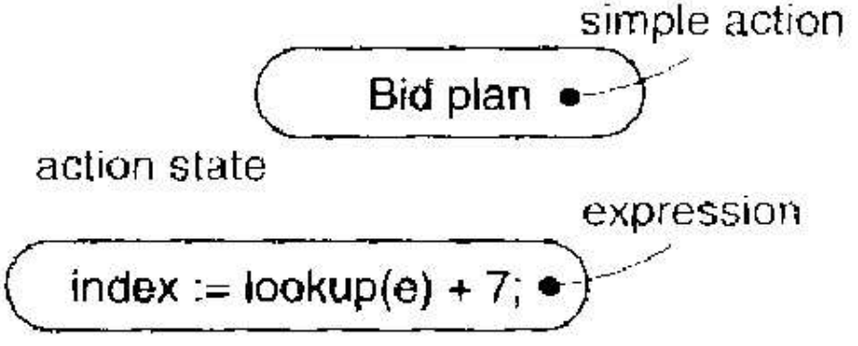


..Diagrame de activităţi

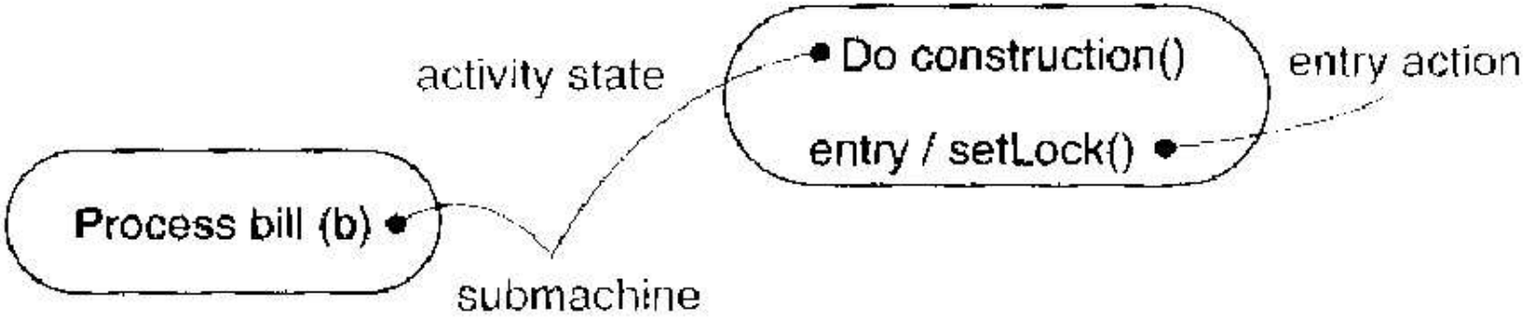
Diagrame de activităţi

Stări de acţiuni şi stări de activităţi

- *stările de acţiuni* sunt *atomice* (acţiunile nu pot fi descompuse ori întrerupte)



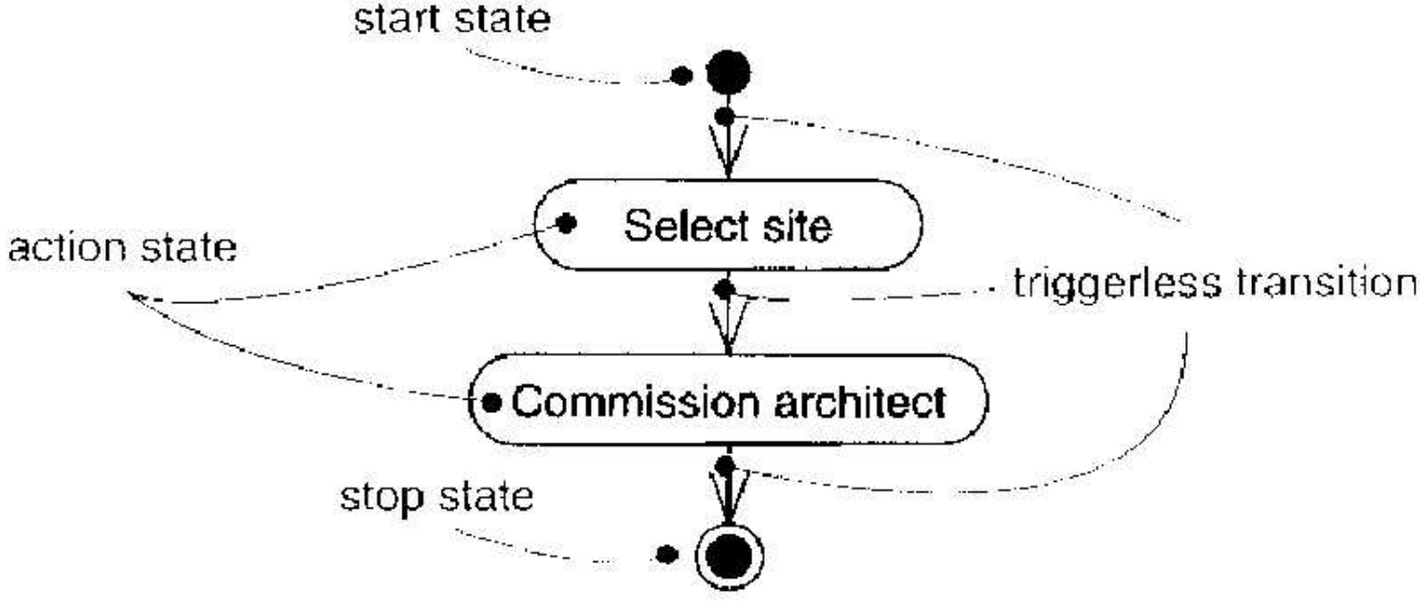
- *stările de activităţi* de descompun, activităţile pot fi întrerupte, durează, se pot include unele în altele, etc.



..Diagrame de activităţi

Tranziţii

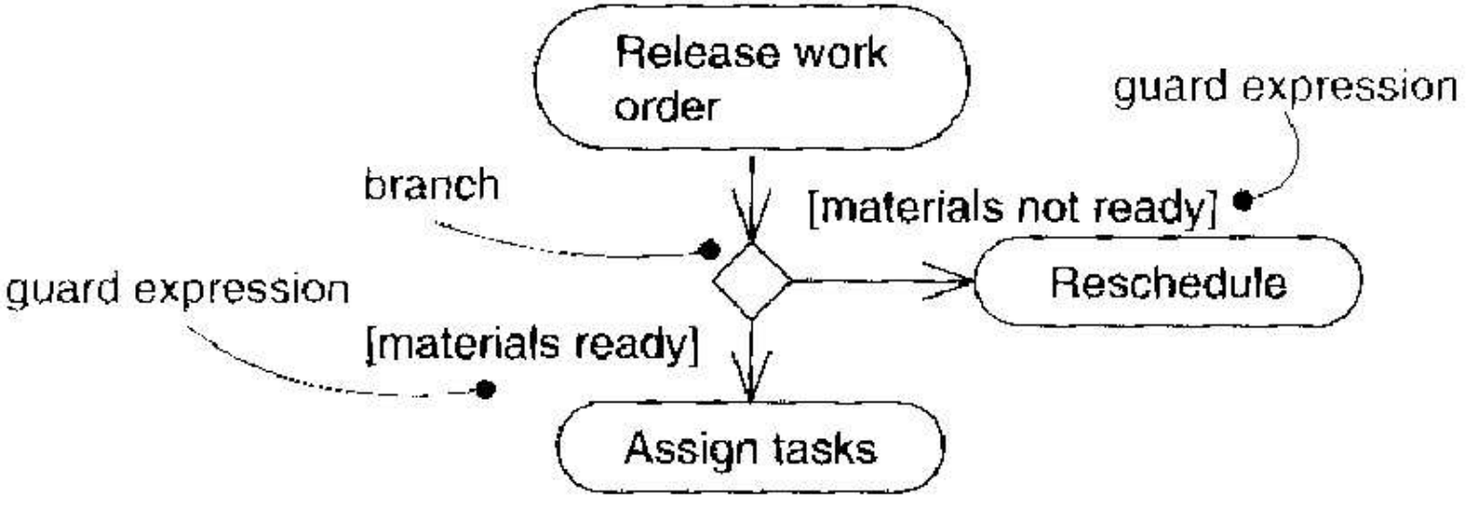
- tranziţiile apar când se trece de la o acţiune/activitate la alta
- pot fi generate de *completarea* acţiunii/activităţii curente, ori *declanşate de evenimente*
- grafic, se reprezintă cu arce direcţionate simple



..Diagrame de activităţi

Ramificaţii

- ca în schemele logice, o diagramă de activităţi poate avea *blocuri condiţionale*, reprezentate cu romburi

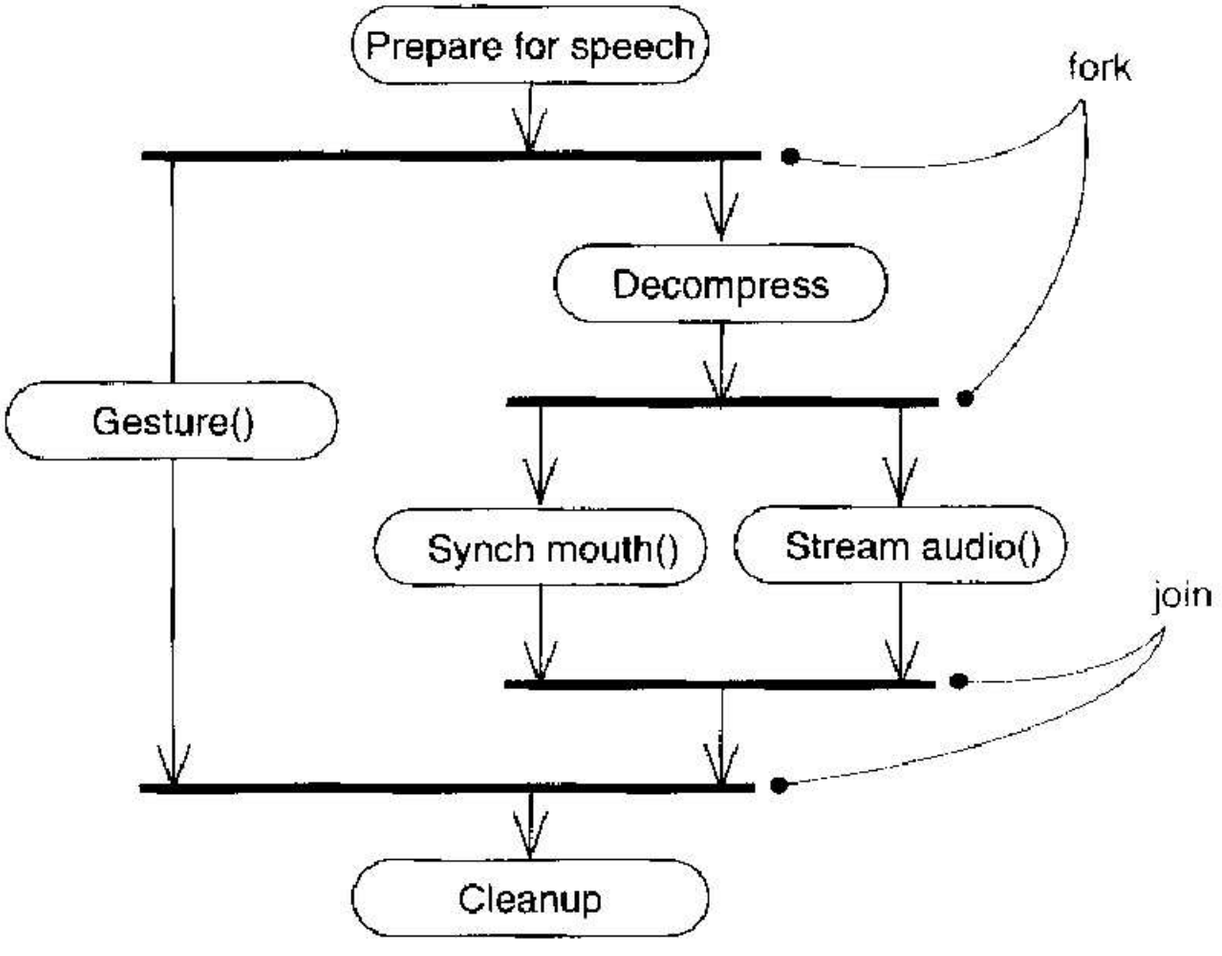


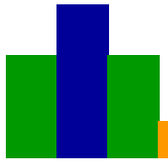
- într-un nod condiţional, condiţiile sunt disjuncte pe fiecare ramură

..Diagrame de activităţi

Fork şi join

- în diagramele de activităţi sunt admise *crearea şi distrugerea* dinamică a *proceselor* prin fork şi join





..Diagrame de activităţi

Swimlanes

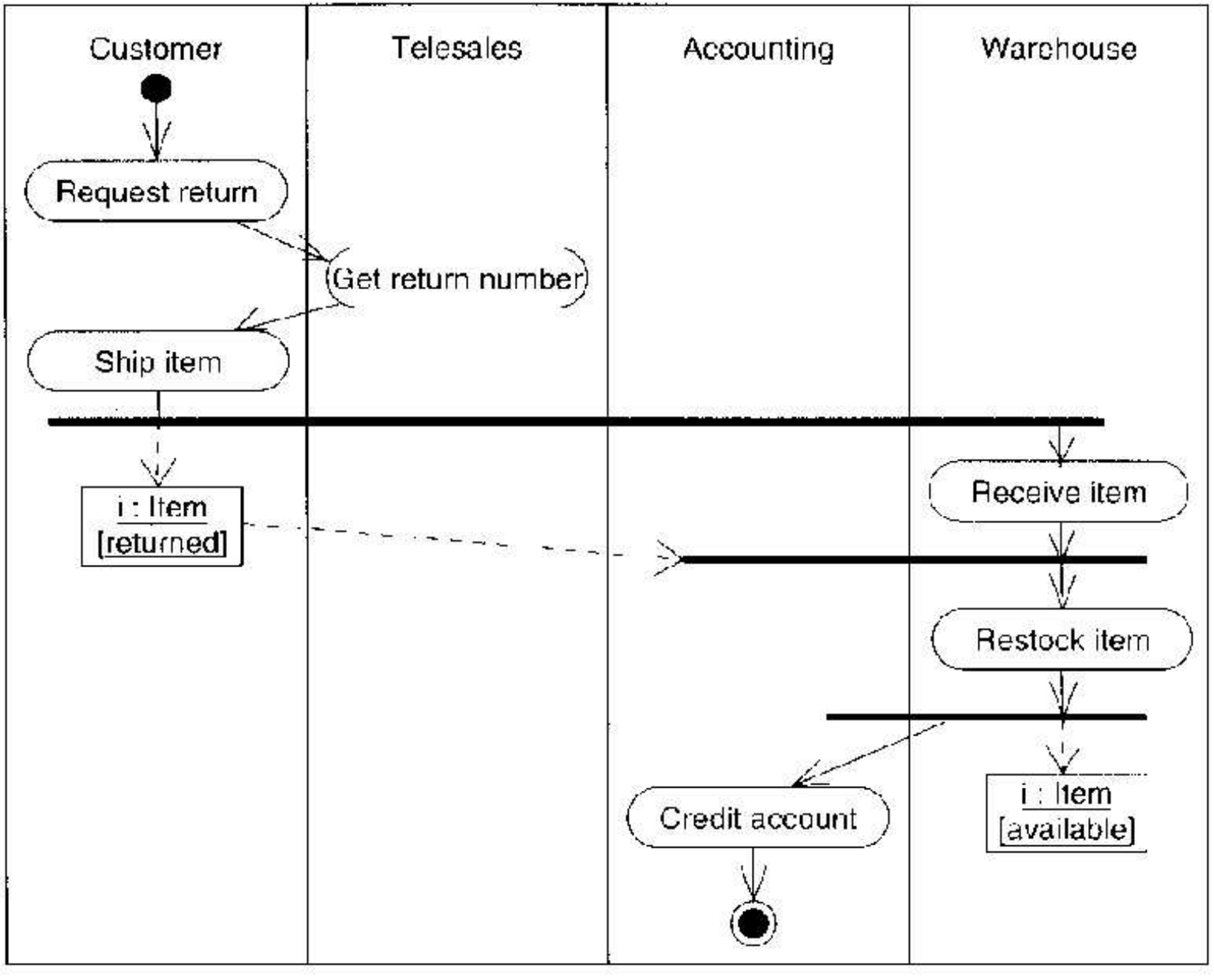
- deseori este utilă *partiţionarea activităţilor pe grupe*, de care sunt responsabile societăţi diferite
- este o împărţire virtuală a diagramei cu linii verticale, fiecare "culoar" având un nume unic

Fluxul obiectelor

- uneori este util să urmărim ce de întâmplă cu anumite obiecte, pasate de la un grup de activităţi la altul
- urmărirea acestui "flux al obiectelor" este utilă în procese manageriale (*workflow*)

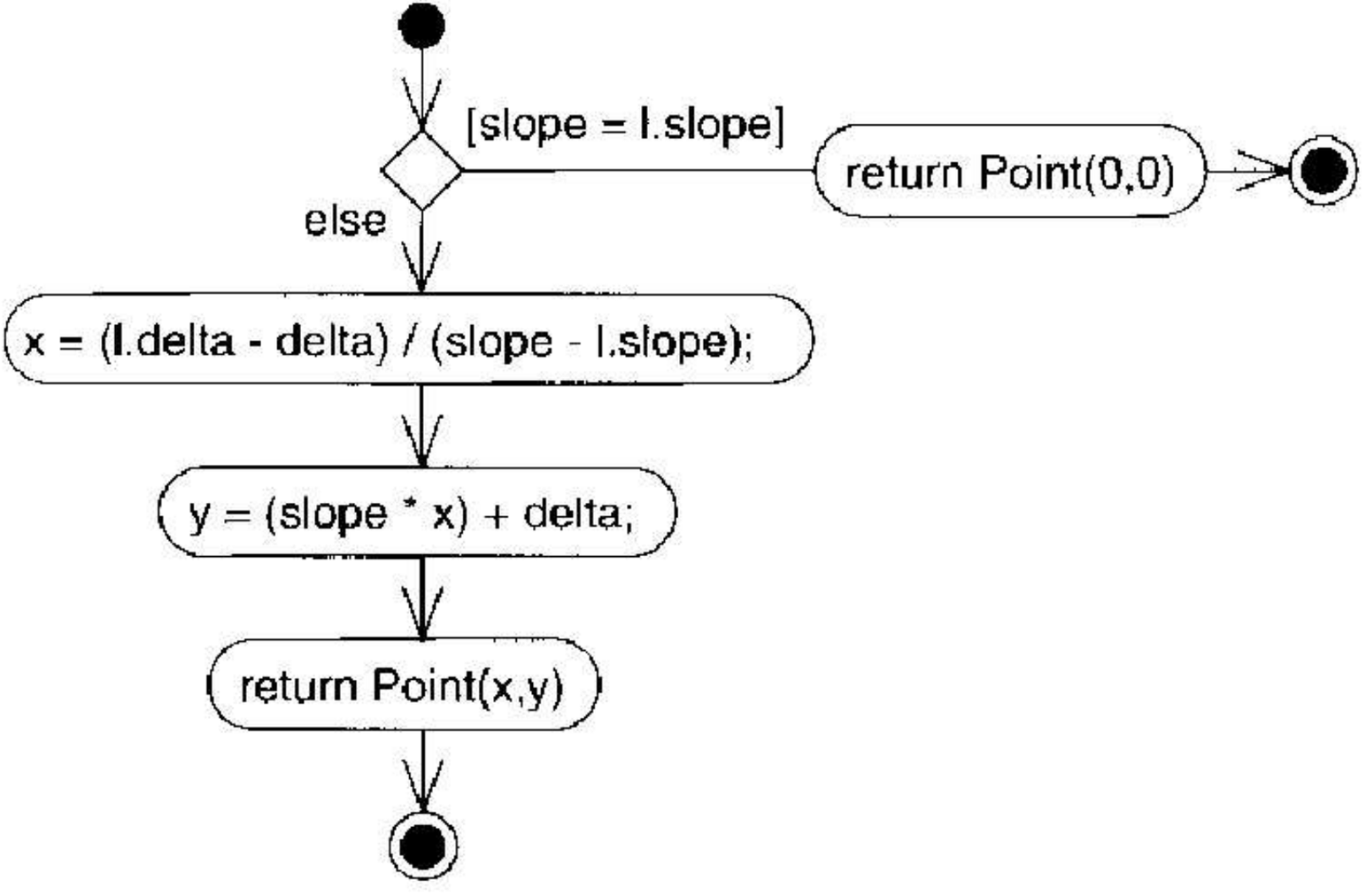
..Diagrame de interacţie

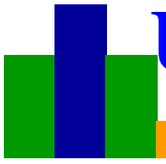
Exemplu - modelarea unui *workflow*



..Diagrame de interacţie

Exemplu - modelarea unei *operaţii* (schemă logică simplă)

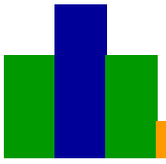




UML: Modelare structurală şi comportamentală

Cuprins:

- Generalităţi
- Modelare structurală
- Modelare comportamentală
- *Concluzii, diverse, etc.*



Concluzii, diverse, etc.

a se insera...