Ingineria Programării

Cursul 3 – 2 Martie

adiftene@info.uaic.ro

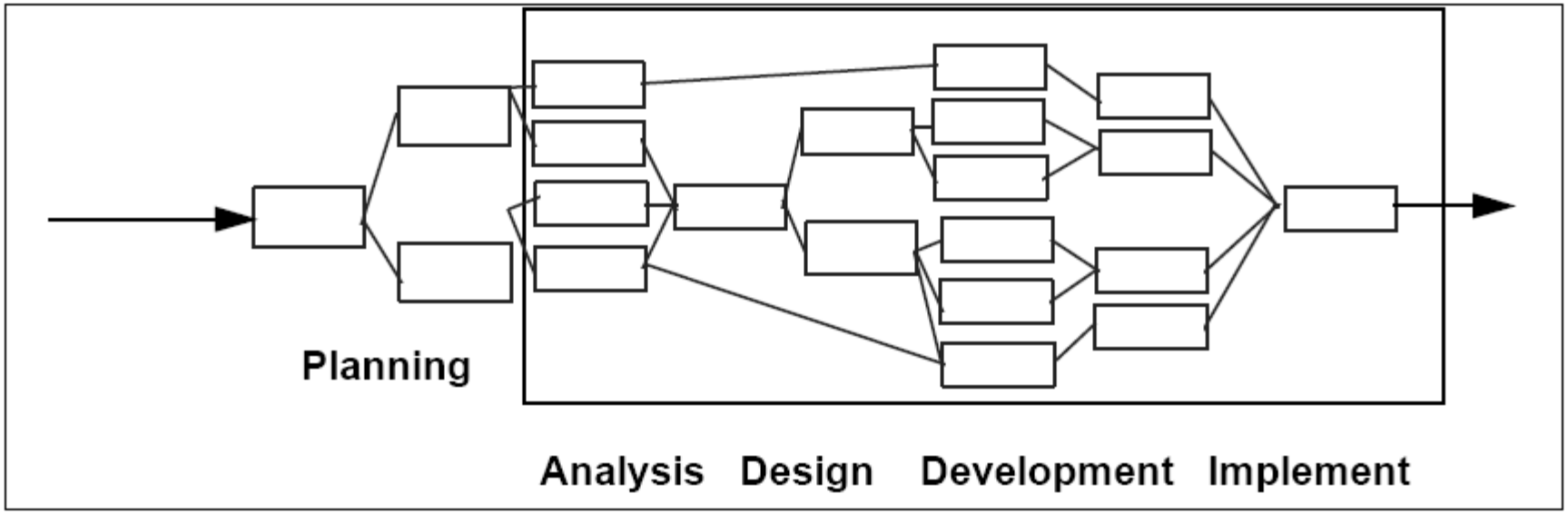
Cuprins

- Din Cursurile 1, 2…
- Modelarea
- Limbaje de Modelare
  - Limbaje Grafice
- UML – Istoric
- UML – Definiţie
- UML – Tipuri de Diagrame
- UML – Diagrame Use Case
- UML – Diagrame de Clase

Din Cursurile 1, 2

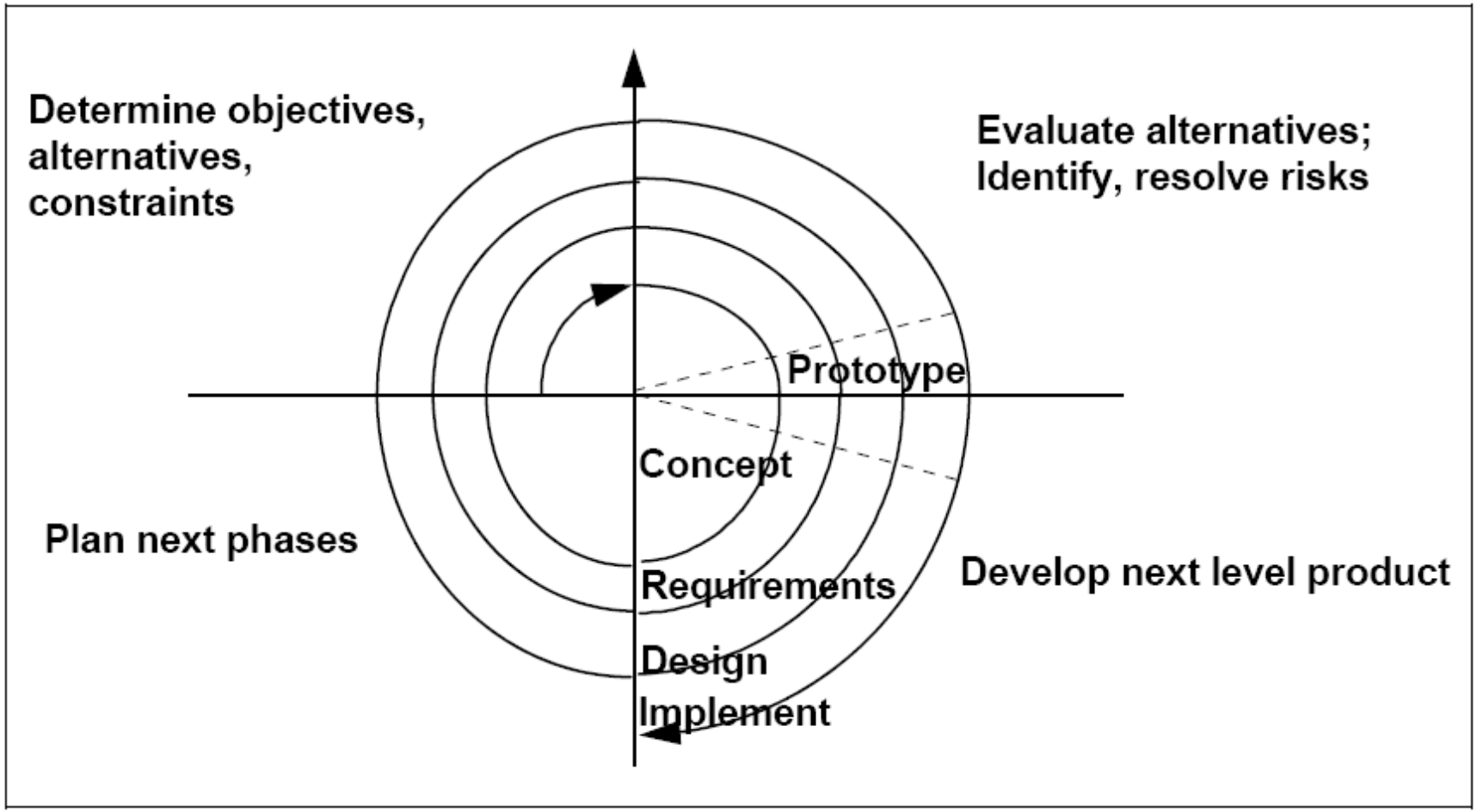
- Ingineria programării (Software engineering)
- Etapele dezvoltării programelor
- Modele de dezvoltare
- Ingineria cerinţelor

Din cursul 1



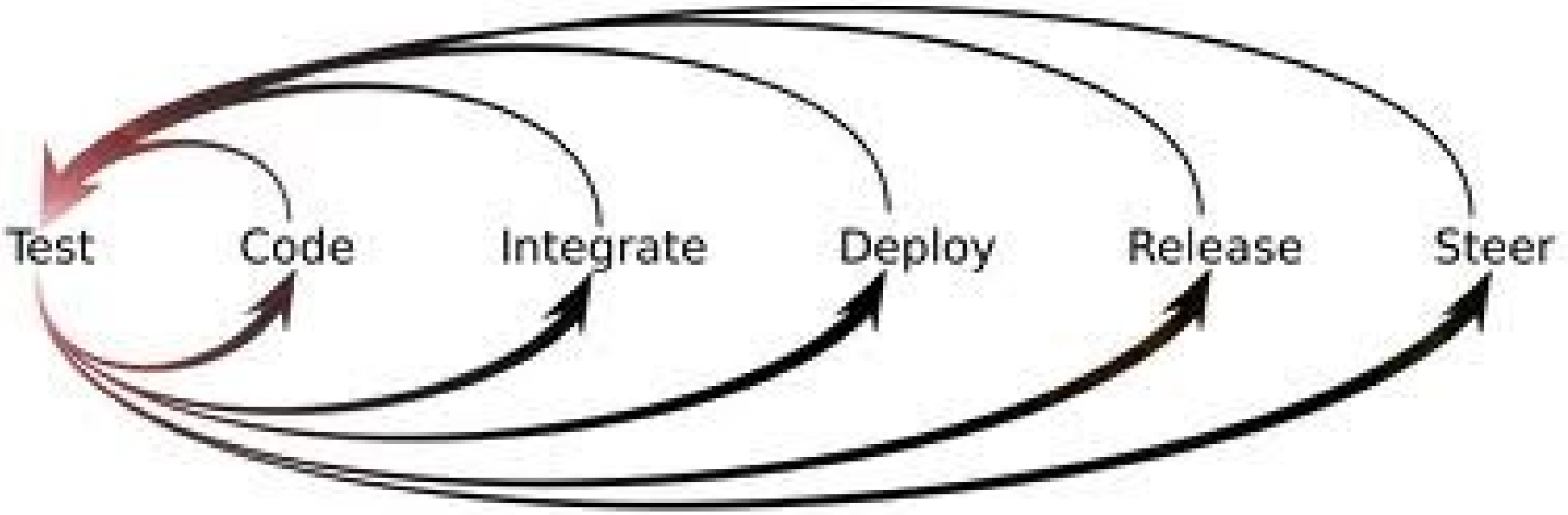
Modelul în cascadă...

Din cursul 1



Modelul în spirală

Din cursul 2

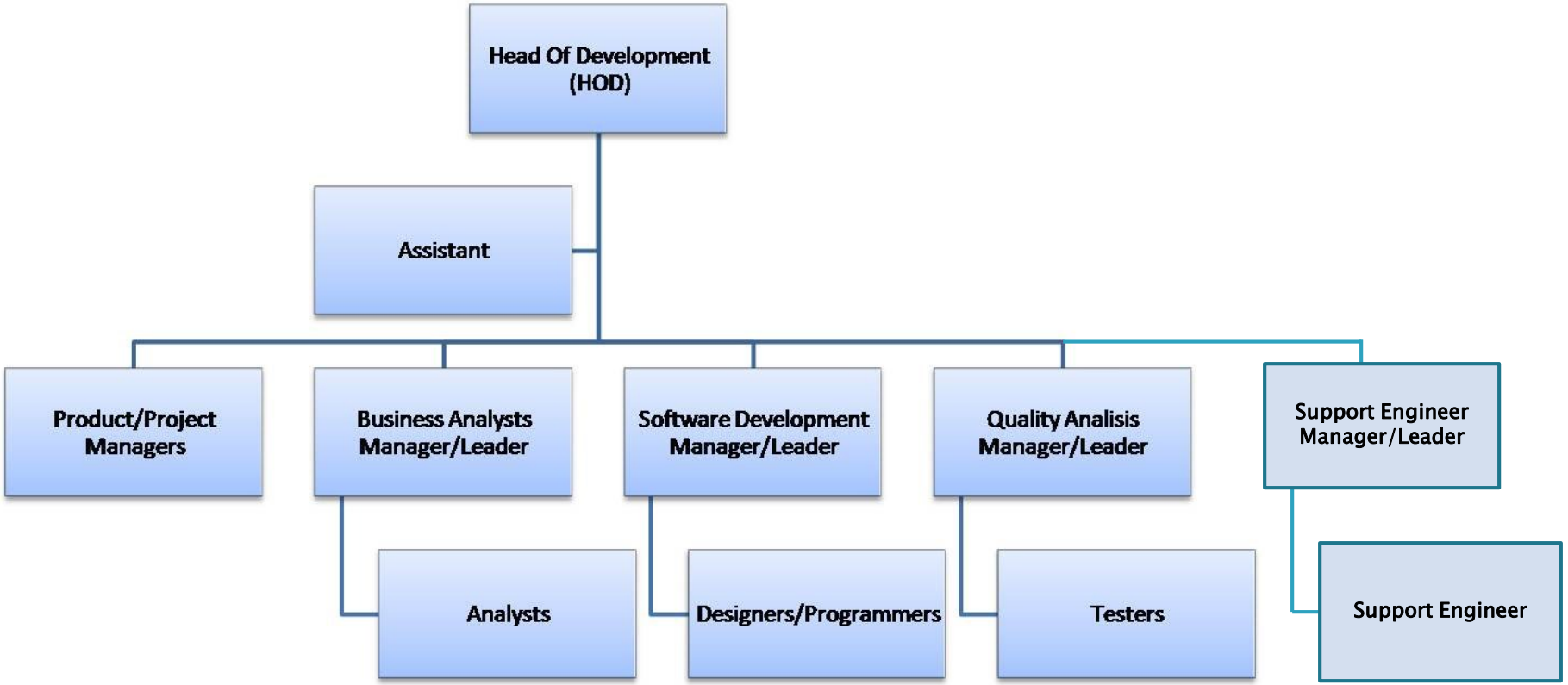


- XP, TDD…

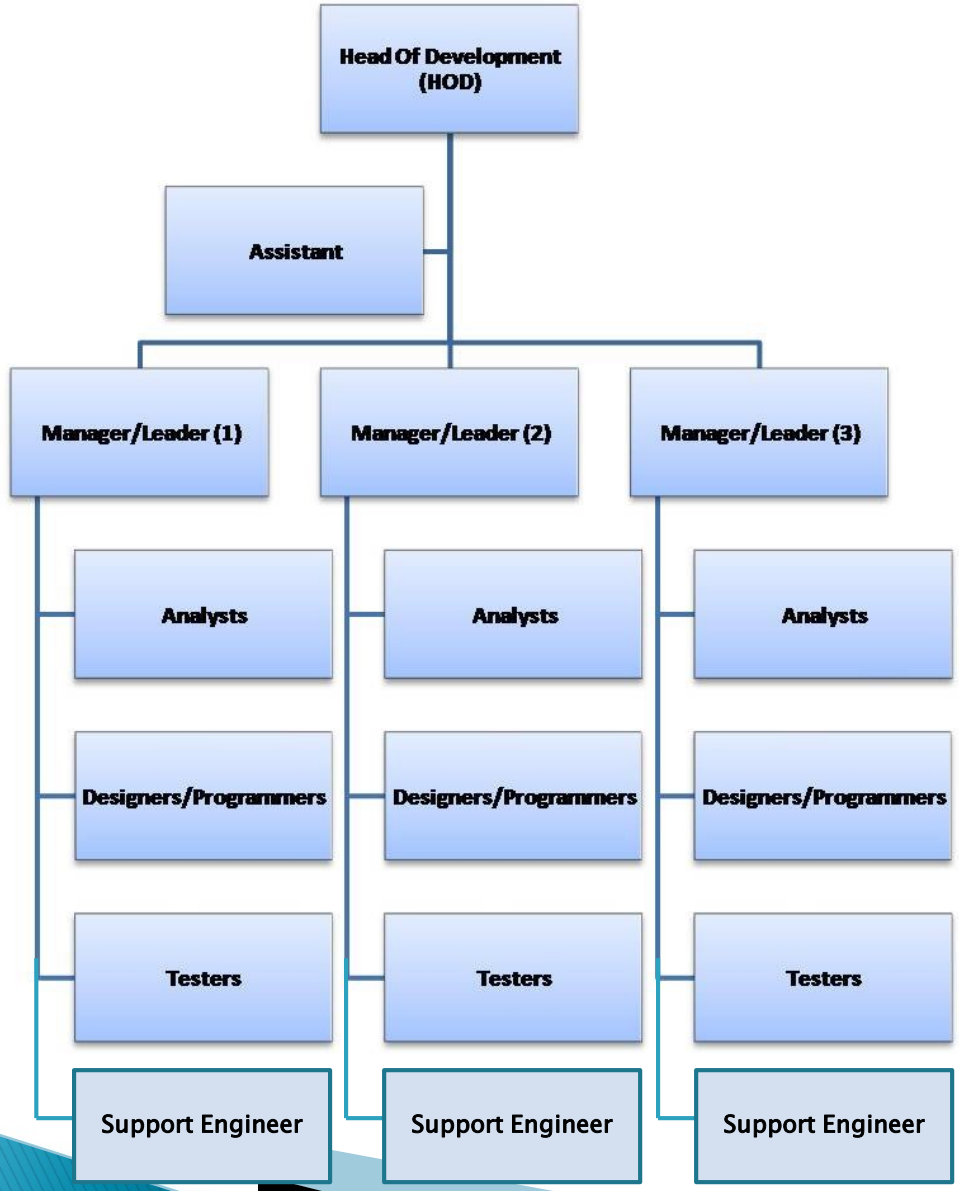
Din cursul 2

- Ingineria cerințelor:
  - Actori
  - Scenarii

Ierarhia dintr-o firmă de software – Compartimente

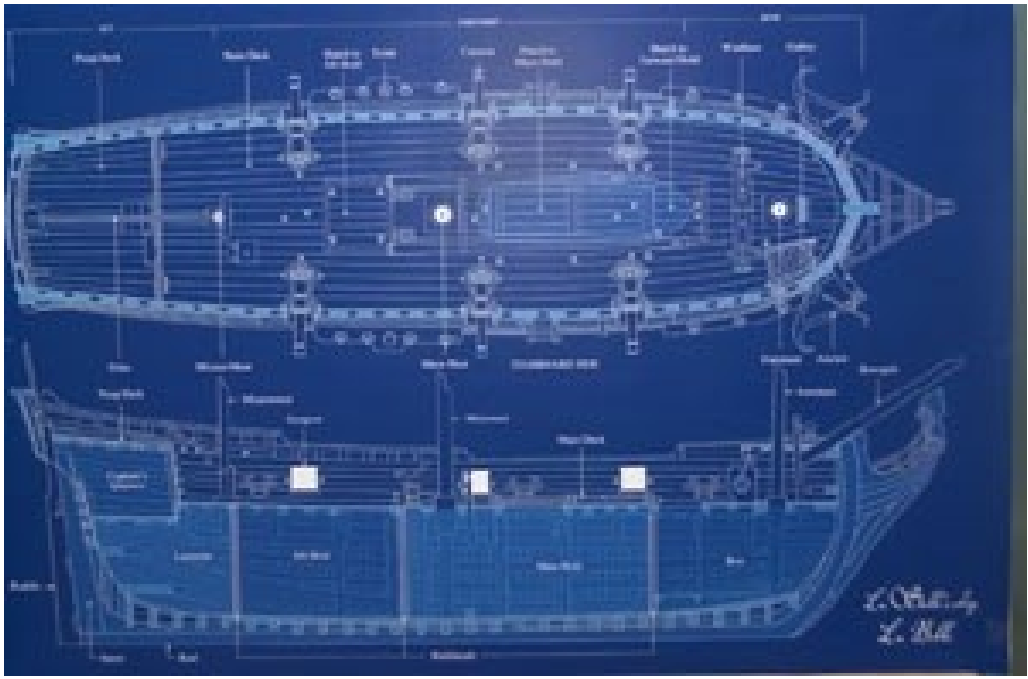


Ierarhia dintr-o firmă de software - Proiecte



Modelarea – De ce?

- Ce este un model?
  - Simplificarea realităţii
  - Planul detaliat al unui sistem (blueprints)



- De ce modelăm?
  - Pentru a înţelege mai bine ce avem de făcut
  - Pentru a ne concentra pe un aspect la un moment dat
- Unde folosim modelarea?

Scopurile Modelării

- Vizualizarea unui sistem
- Specificarea structurii sale şi/sau a comportării
- Oferirea unui şablon care să ajute la construcţie
- Documentarea deciziilor luate

Modelarea Arhitecturii

- Cu ajutorul **Use case**-urilor: pentru a prezenta cerinţele
- Cu ajutorul **Design**-ului: surprindem vocabularul şi domeniul problemei
- Cu ajutorul **Proceselor**: surprindem procesele şi thread-urile
- Cu ajutorul **Implementării**: avem modelarea aplicaţiei
- Cu ajutorul **Deployment**: surprindem sistemul din punct de vedere ingineresc

Principiile Modelării

- Modelele influenţează soluţia finală
- Se pot folosi diferite niveluri de precizie
- Modelele bune au corespondent în realitate
- Nu e suficient un singur model

Limbaje de Modelare

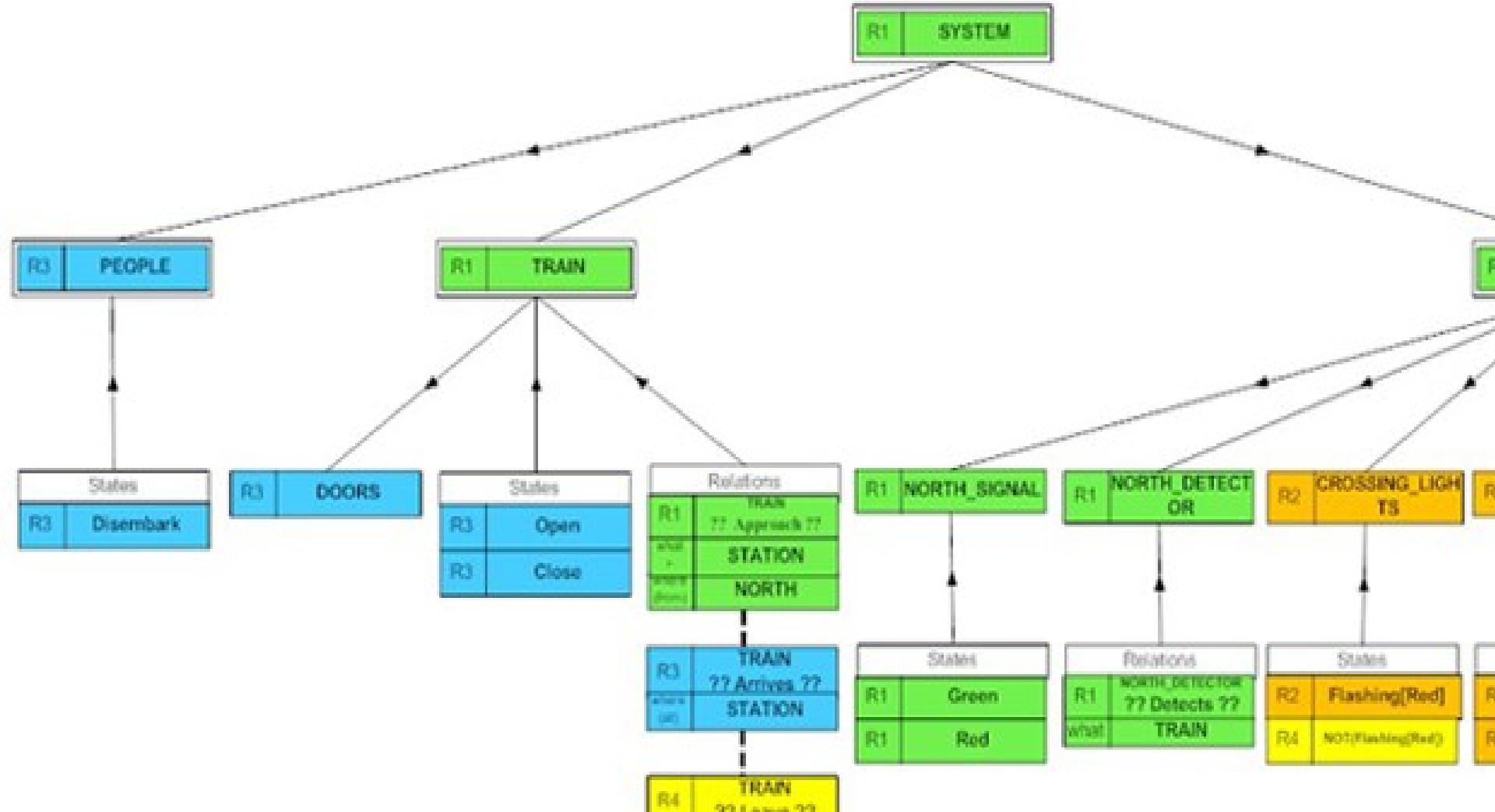
- Analiza şi proiectarea unui proiect trebuie făcute înainte de realizarea codului
- În prezent, se acordă o atenţie deosebită acestei etape, deoarece de ele depind **producerea** şi **refolosirea** de software
- Pentru analiza și proiectarea programelor s-au creat **limbajele de modelare**
- **Limbaj de modelare este** un limbaj artificial care poate fi folosit să exprime informații sau cunoaștere sau sisteme

Tipuri de Limbaje de Modelare

- **Limbaje Grafice**: arbori comportamentali, modelarea proceselor de business, EXPRESS (modelarea datelor), flowchart, ORM (modelarea rolurilor), reţele Petri, **diagrame UML**
- **Limbaje Specifice:** modelare algebrică (AML) (pentru descrierea şi rezolvarea problemelor de matematică ce necesită putere computaţională mare), modelarea domeniilor specifice (DSL), modelarea arhitecturilor specifice (FSML), modelarea obiectelor (object modeling language), modelarea realităţii virtuale (VRML)

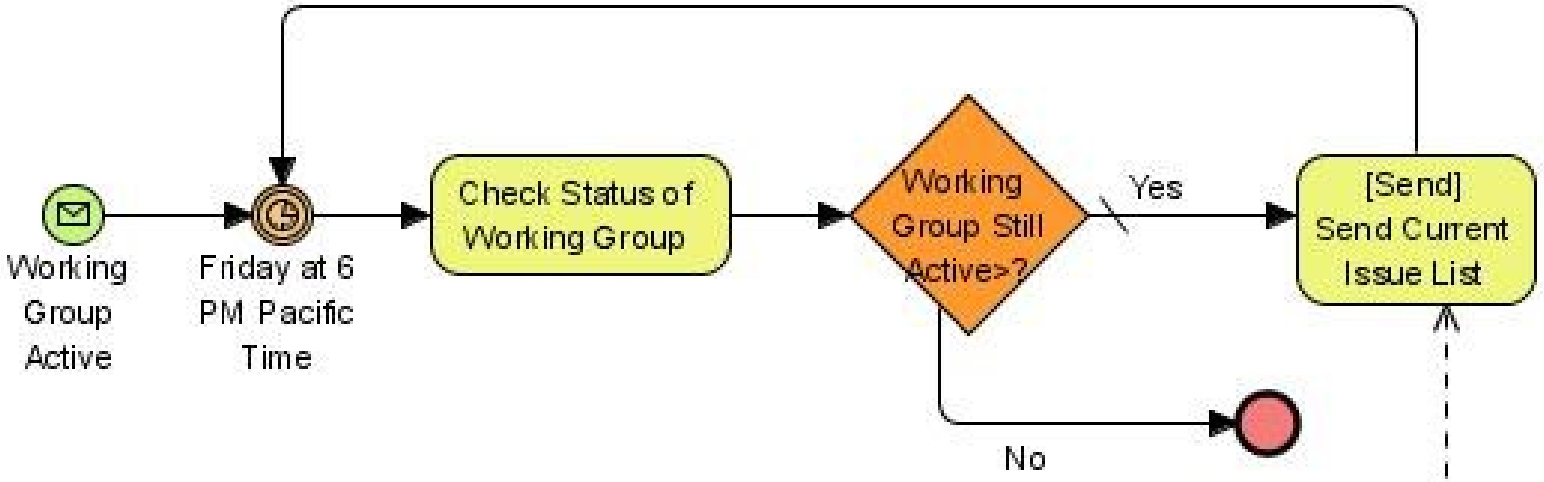
Limbaje Grafice 1

- Arbori comportamentali

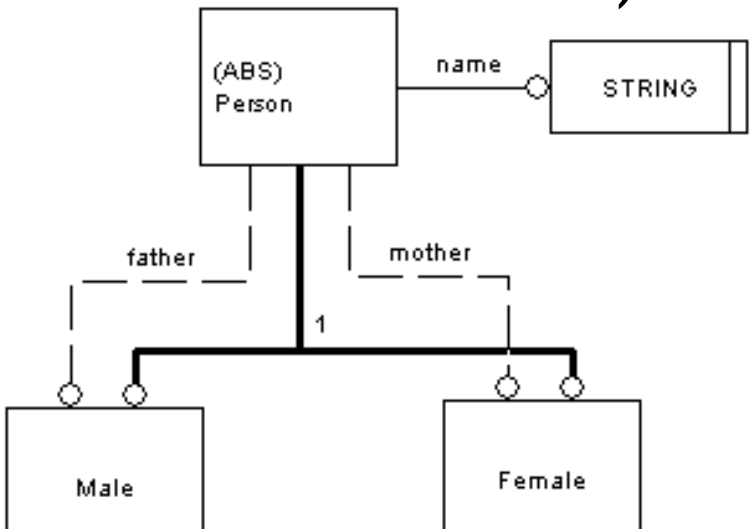


Limbaje Grafice 2

- Modelarea proceselor de business

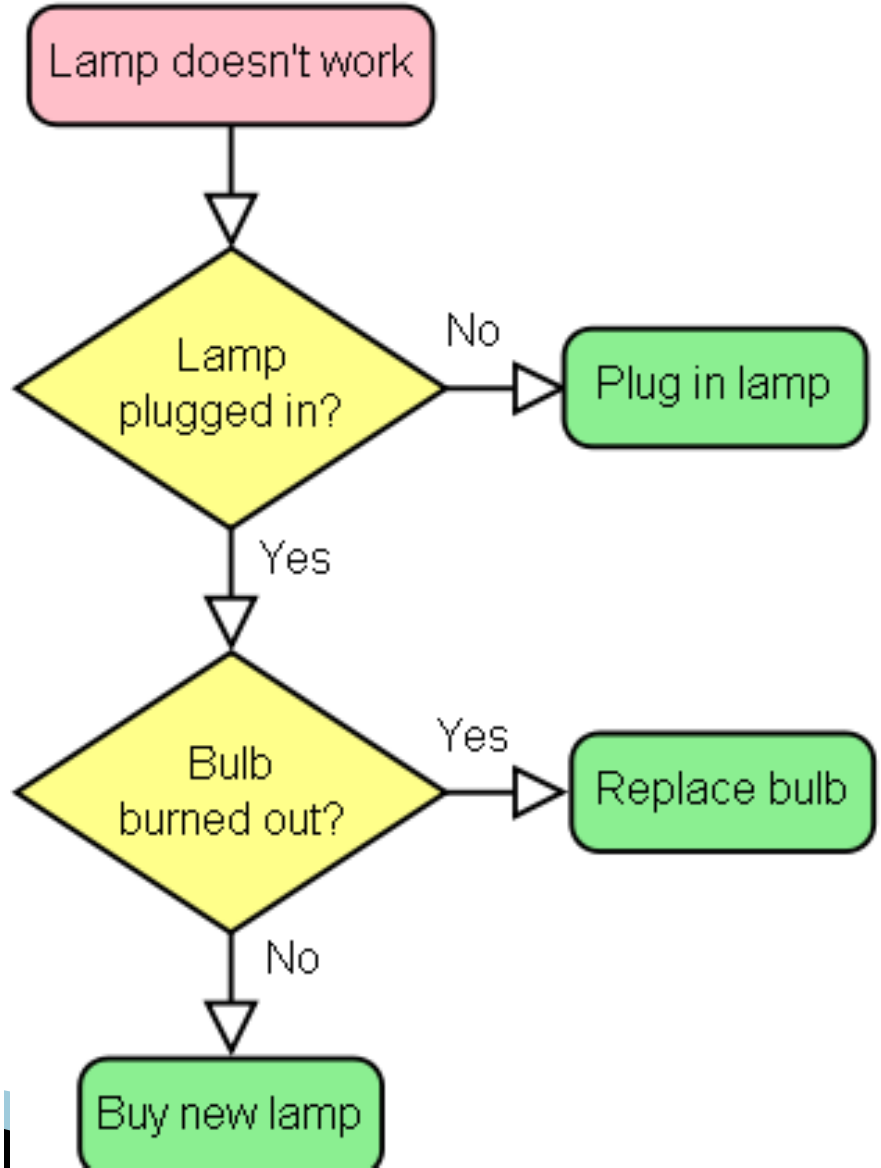


- EXPRESS (modelarea datelor)



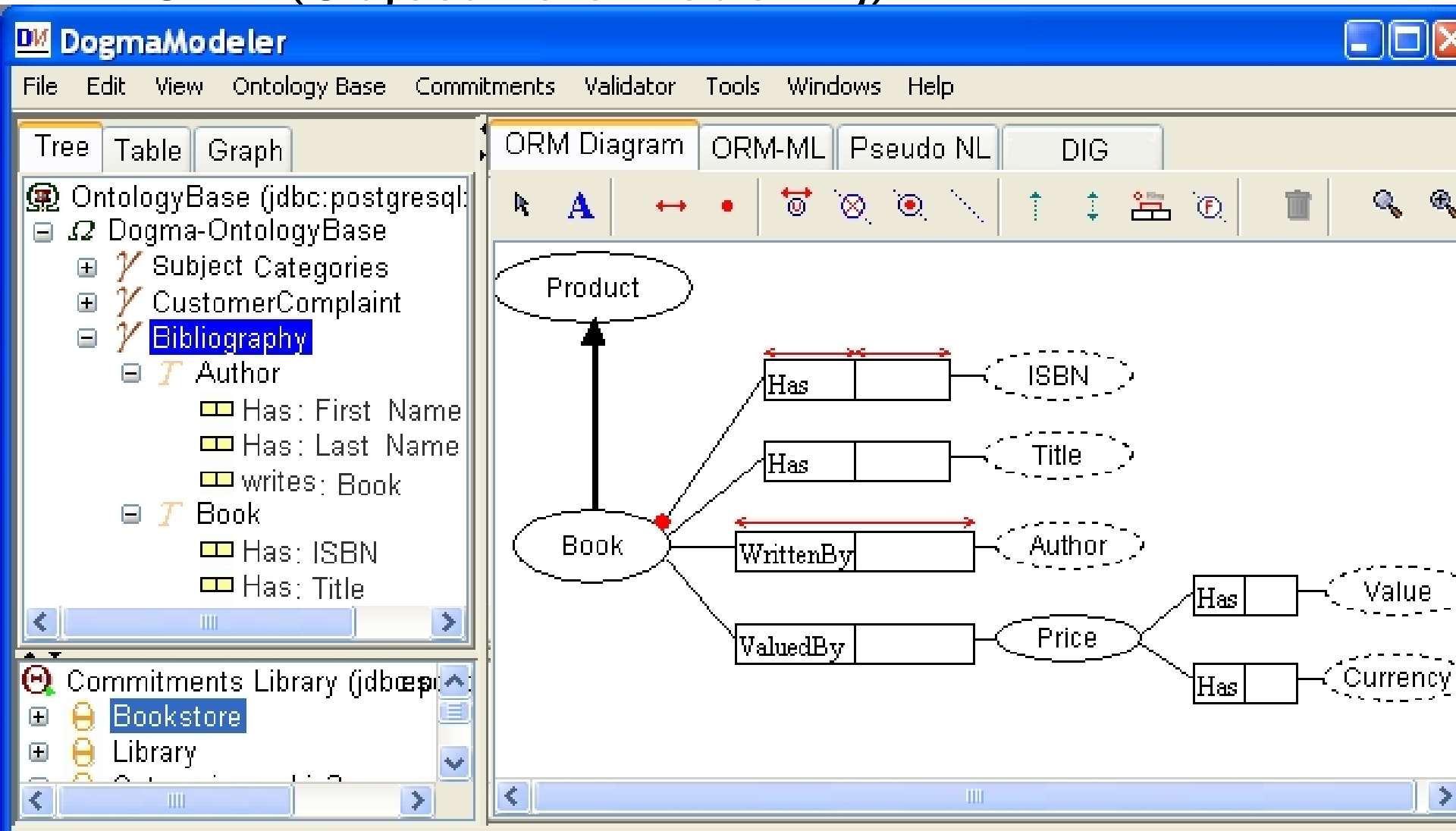
Limbaje Grafice 3

- Flowchart



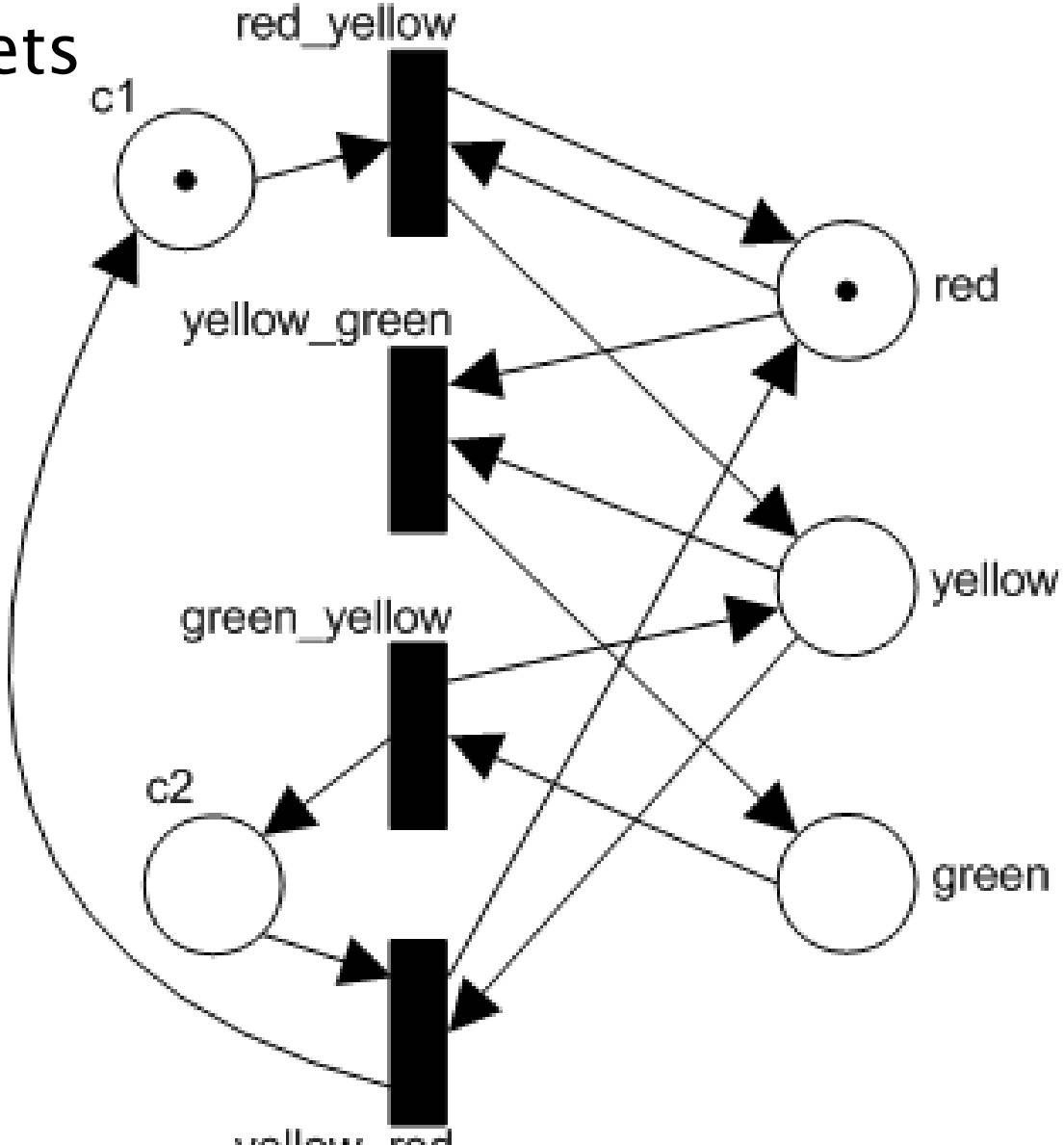
Limbaje Grafice 4

- ORM (Object Role Modeling)



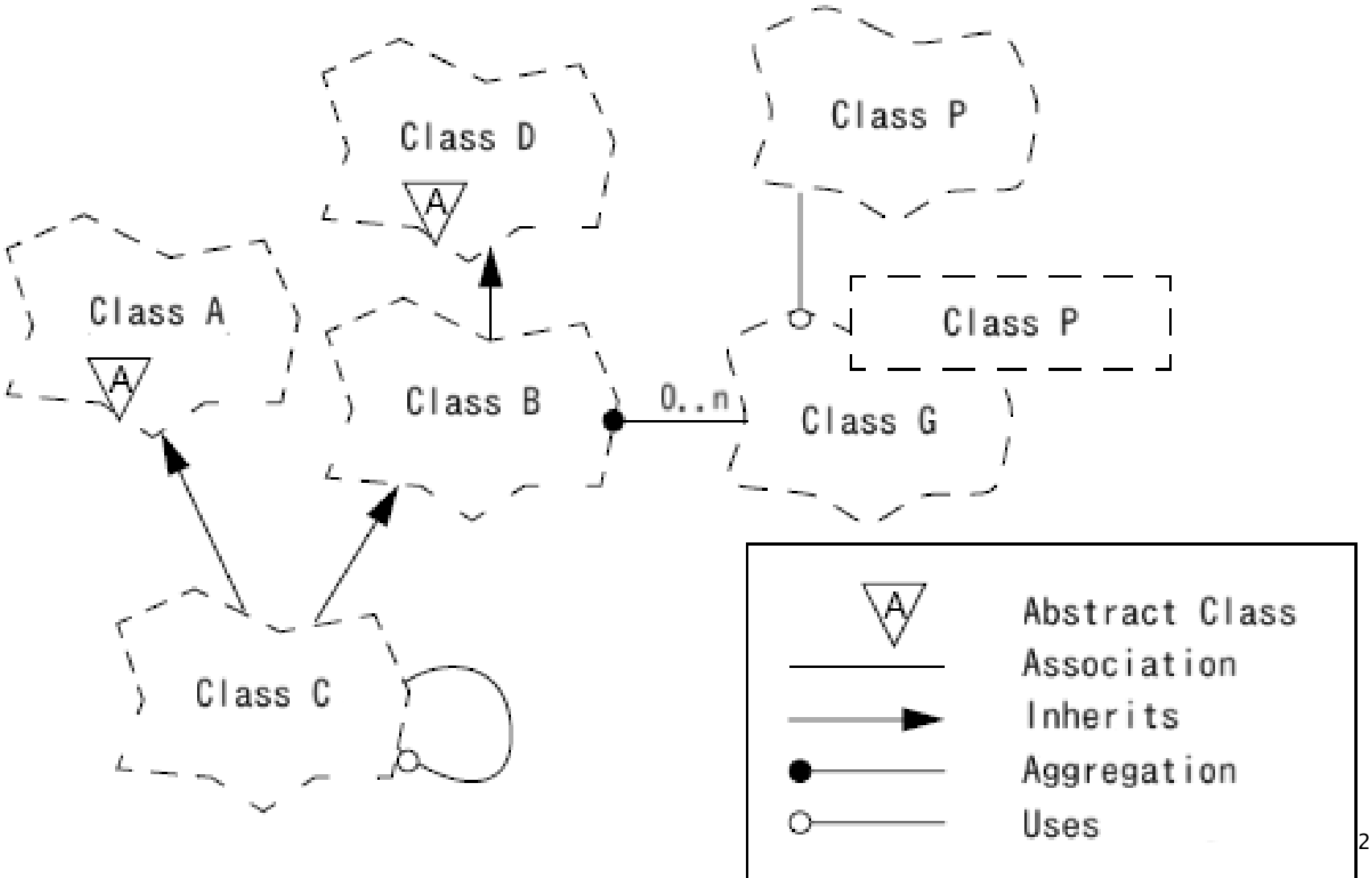
Limbaje Grafice 5

- Petri Nets



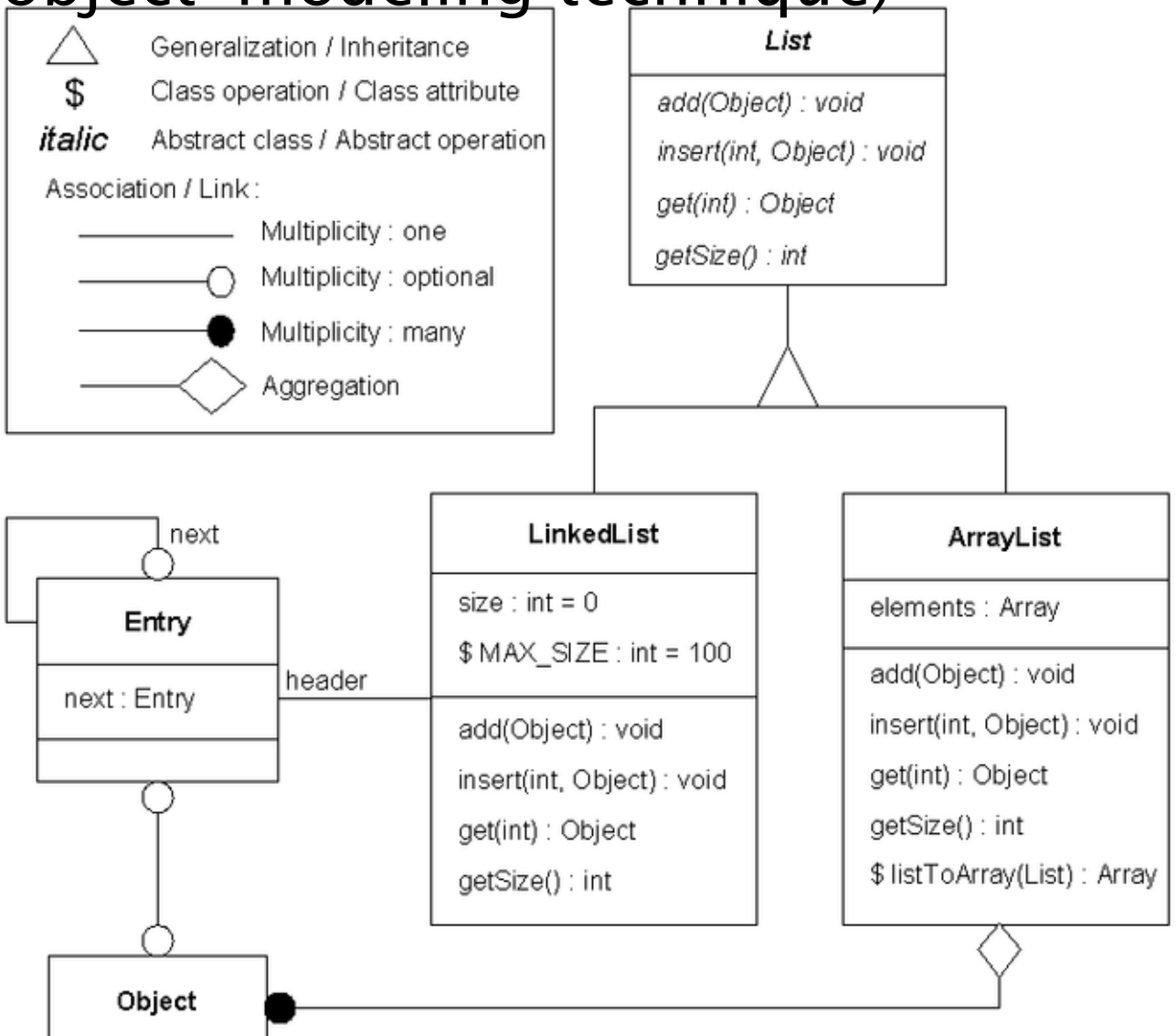
Limbaje Grafice 6

- Metoda Booch (Grady Booch) – analiza şi design oo



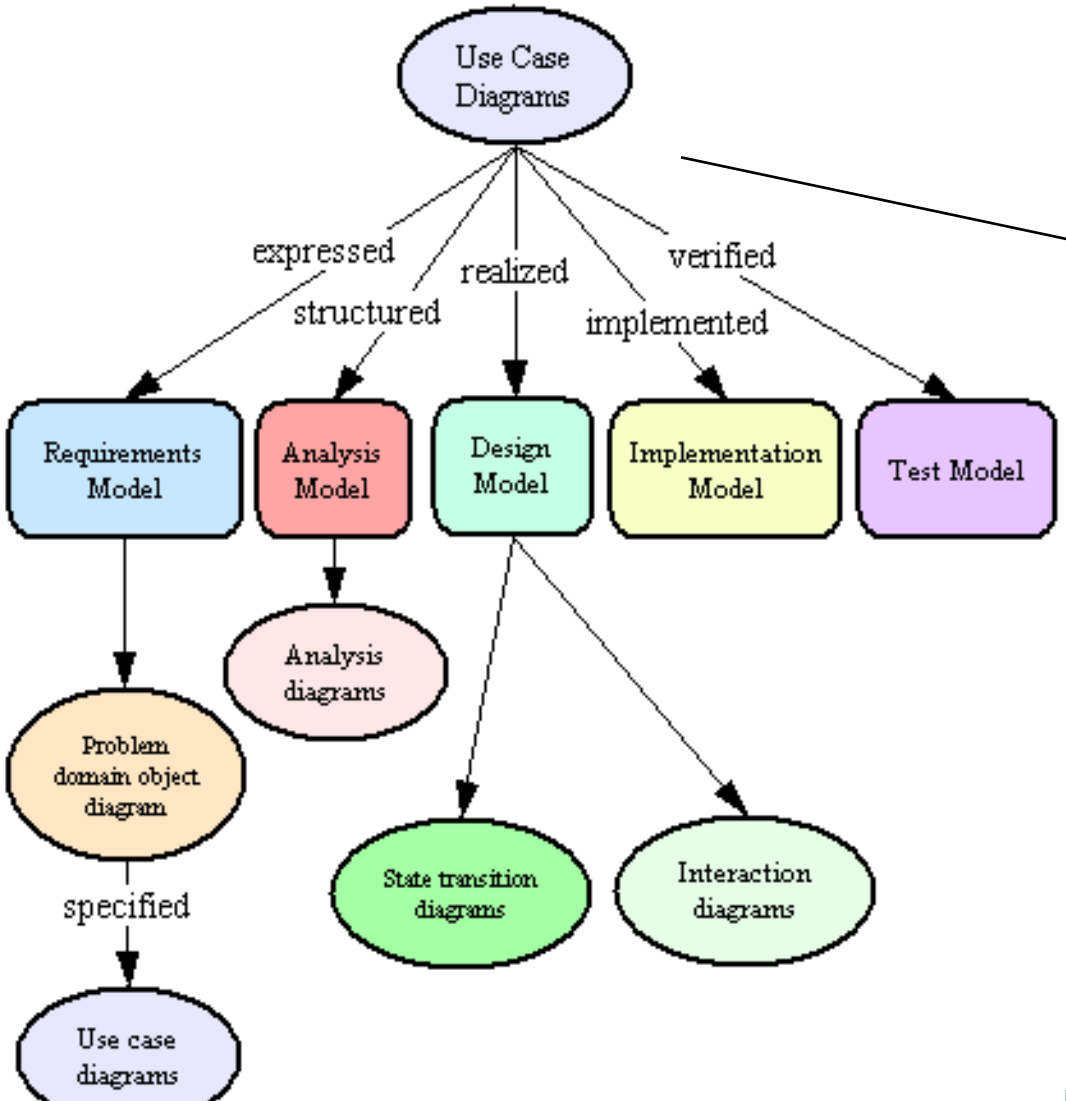
Limbaje Grafice 7

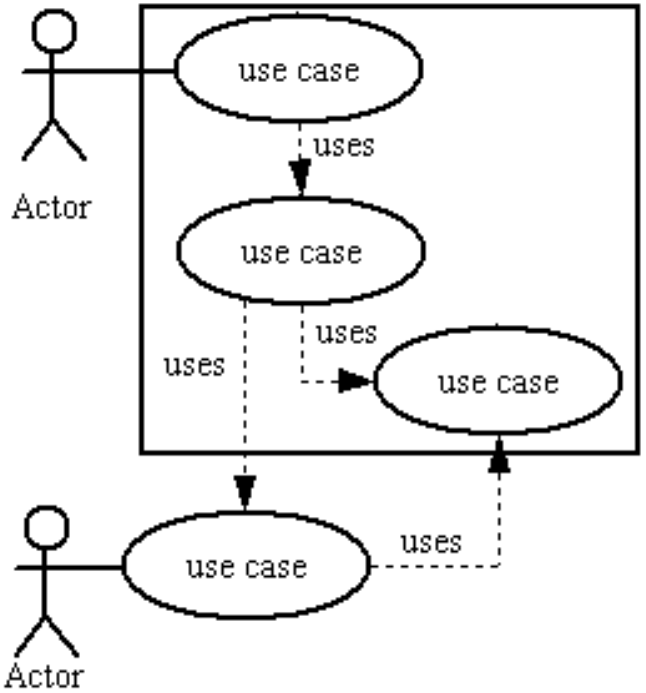
- OMT (object-modeling technique)



Limbaje Grafice 8

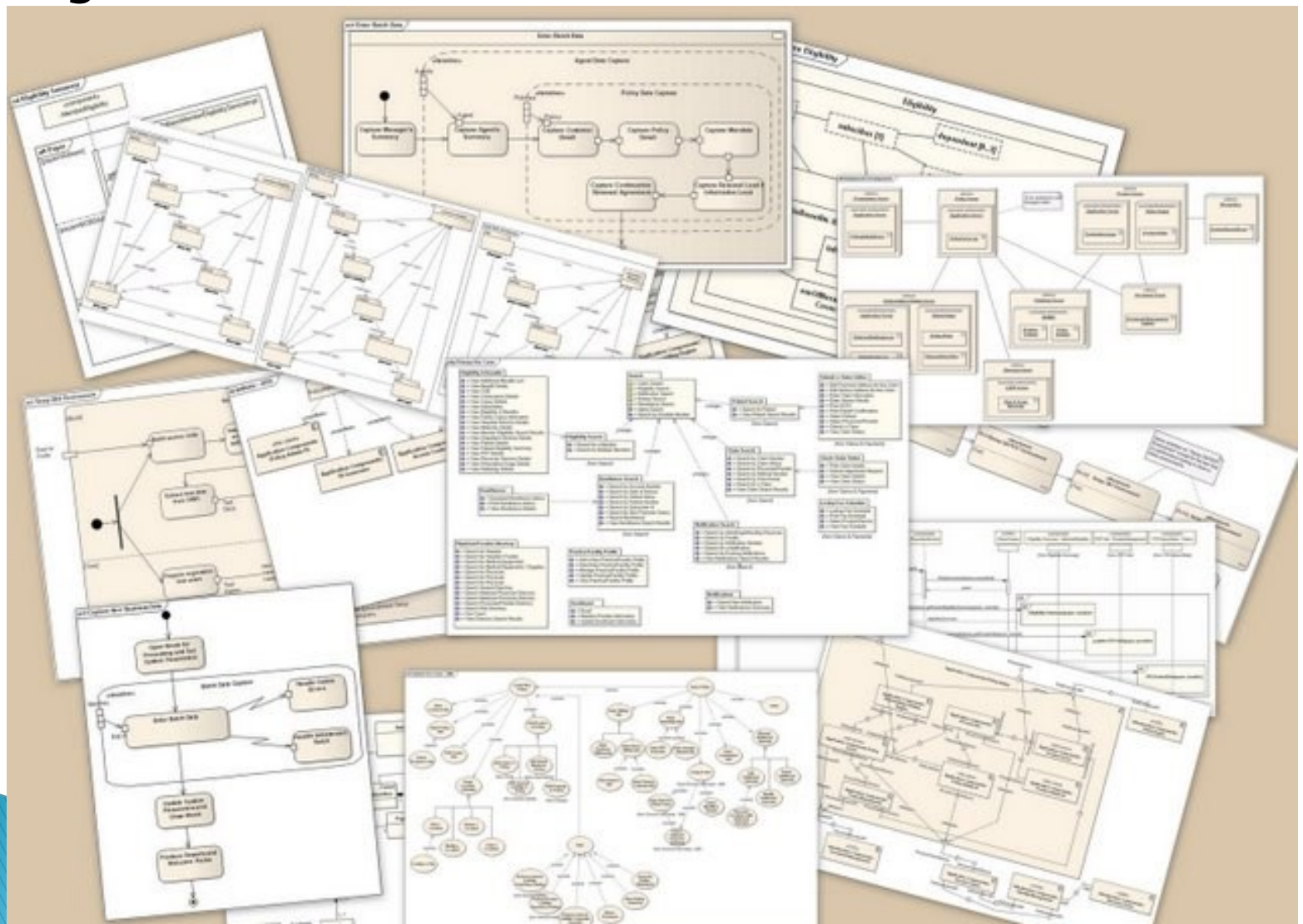
- OOSE (Object-oriented software engineering)





Limbaje Grafice 9

- Diagrame UML

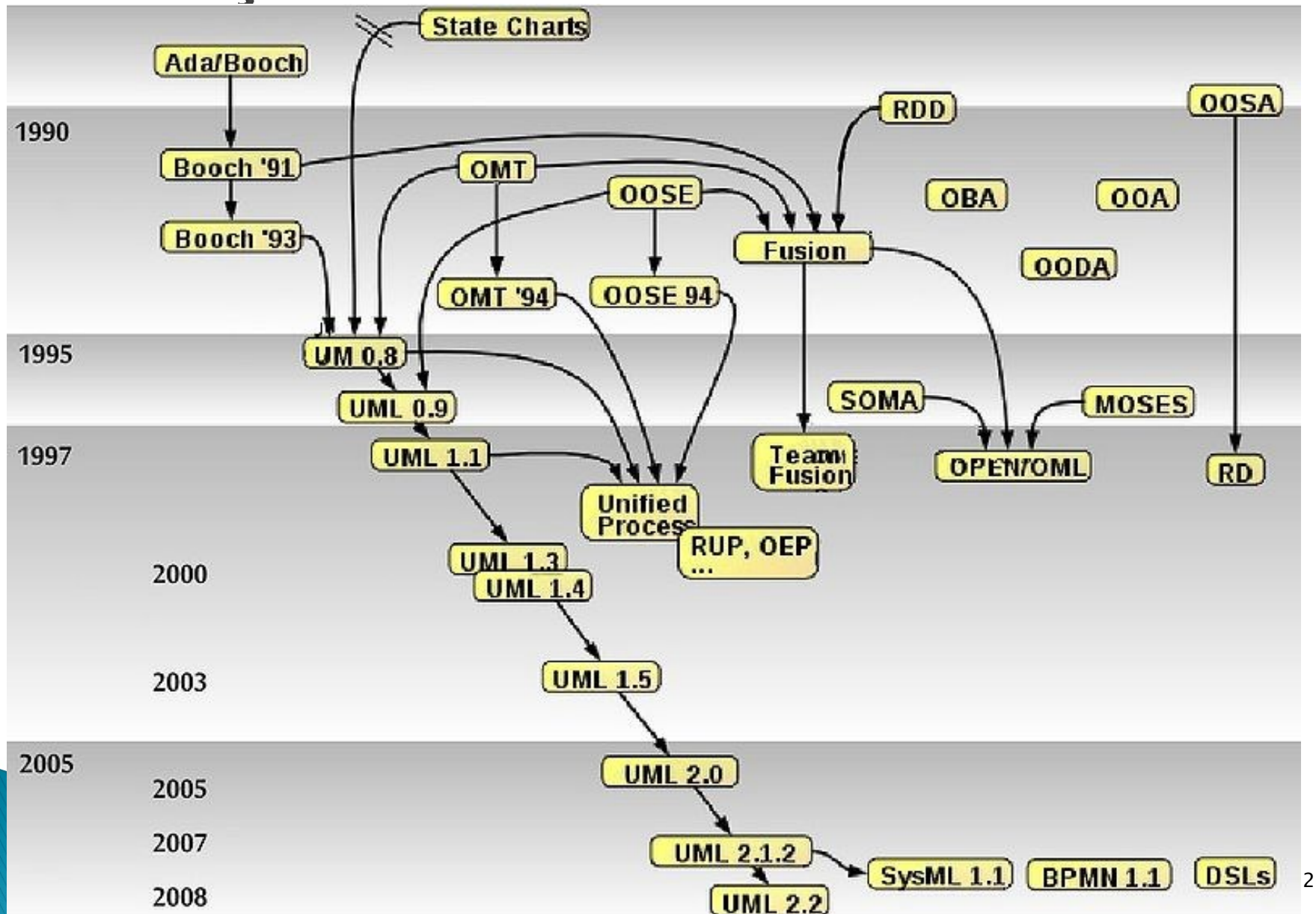


UML – Introducere

- UML (Unified Modeling Language) este succesorul celor mai bune trei limbaje OO de modelare anterioare:
  - Booch (Grady Booch)
  - OMT (Ivar Jacobson)
  - OOSE (James Rumbaugh)

- UML se constituie din unirea acestor limbaje de modelare şi în plus are o expresivitate mai mare

Evoluţie UML

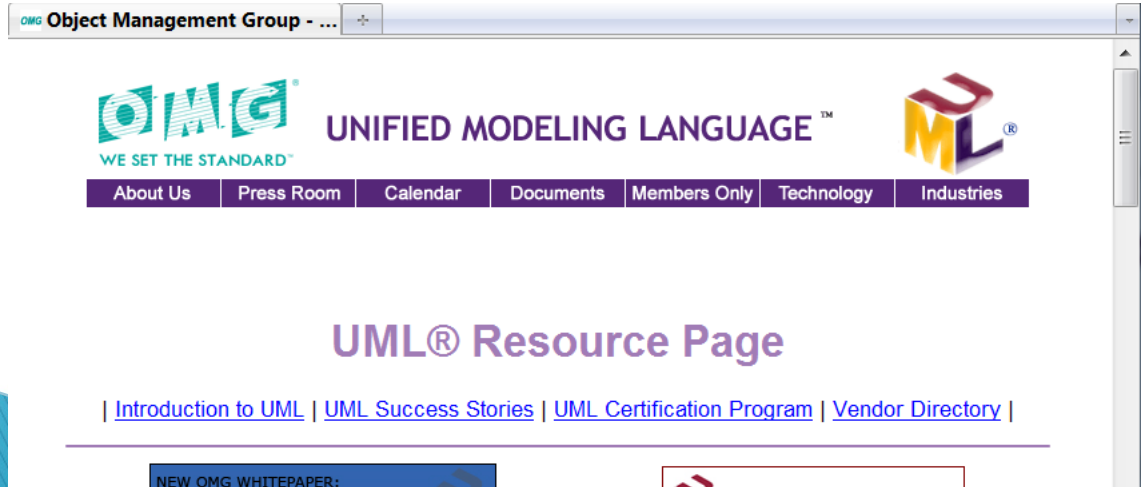


UML – Definiţie (OMG)

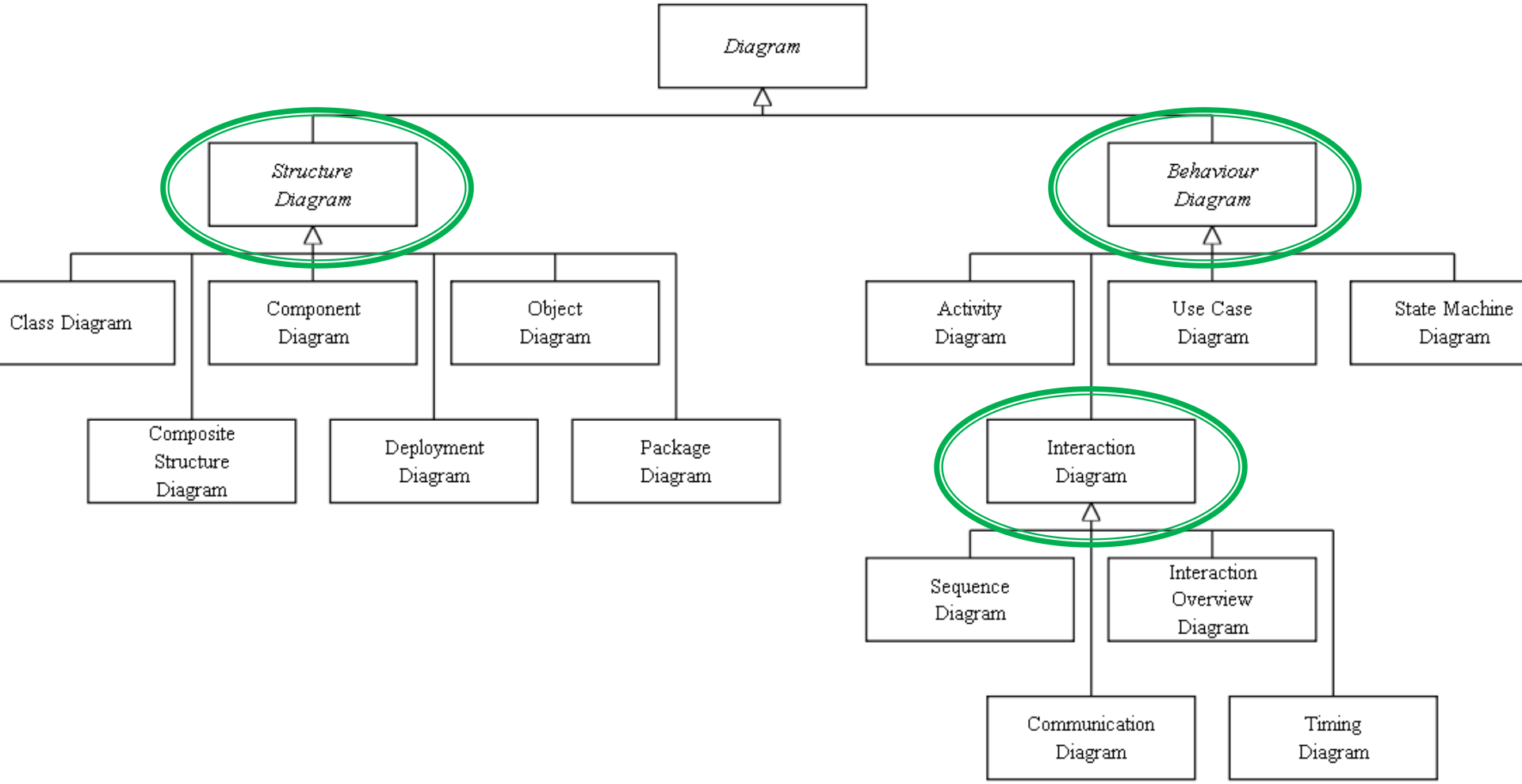
- *"The Unified Modeling Language (UML) is a* ***graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.***
- *The UML offers a* ***standard way to write a system's blueprints****, including conceptual things such as* ***business processes*** *and* ***system functions*** *as well as concrete things such as* ***programming language statements****,* ***database schemas****, and* ***reusable software components****."*

UML – Standard Internaţional

- Ianuarie 1997 – UML 1.0 a fost propus spre standardizare în cadrul OMG (Object Management Group)
- Noiembrie 1997 – Versiunea UML 1.1 a fost adoptată ca standard de către OMG
- Ultima versiune este UML 2.5.1 (Decembrie 2017)
- Site-ul oficial: http://www.uml.org

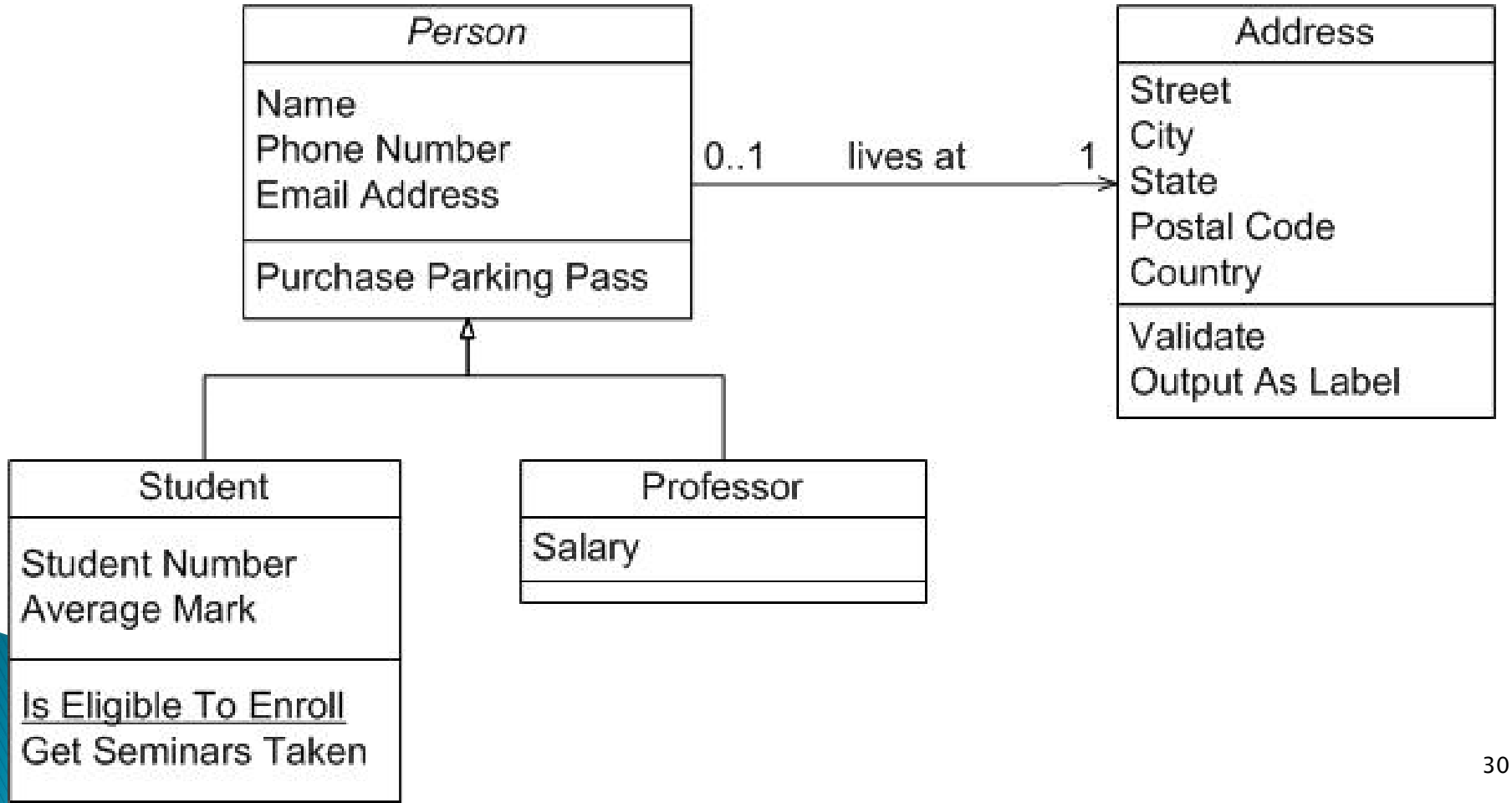


UML2.0 – 13 Tipuri de Diagrame



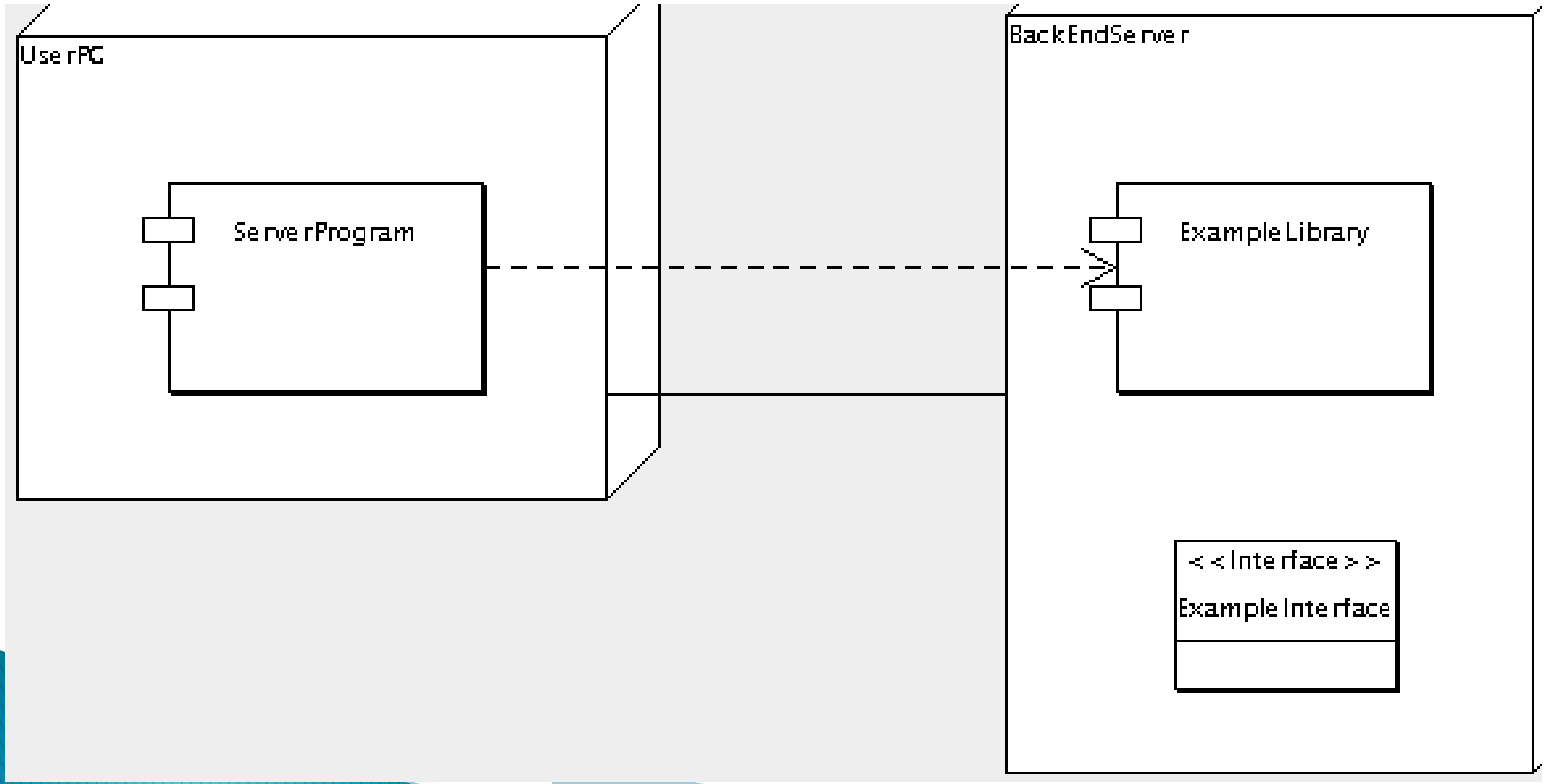
UML2.0 – Diagrame de Structură 1

- **Diagrame de Clasă**: clasele (atributele, metodele) şi relaţiile dintre clase



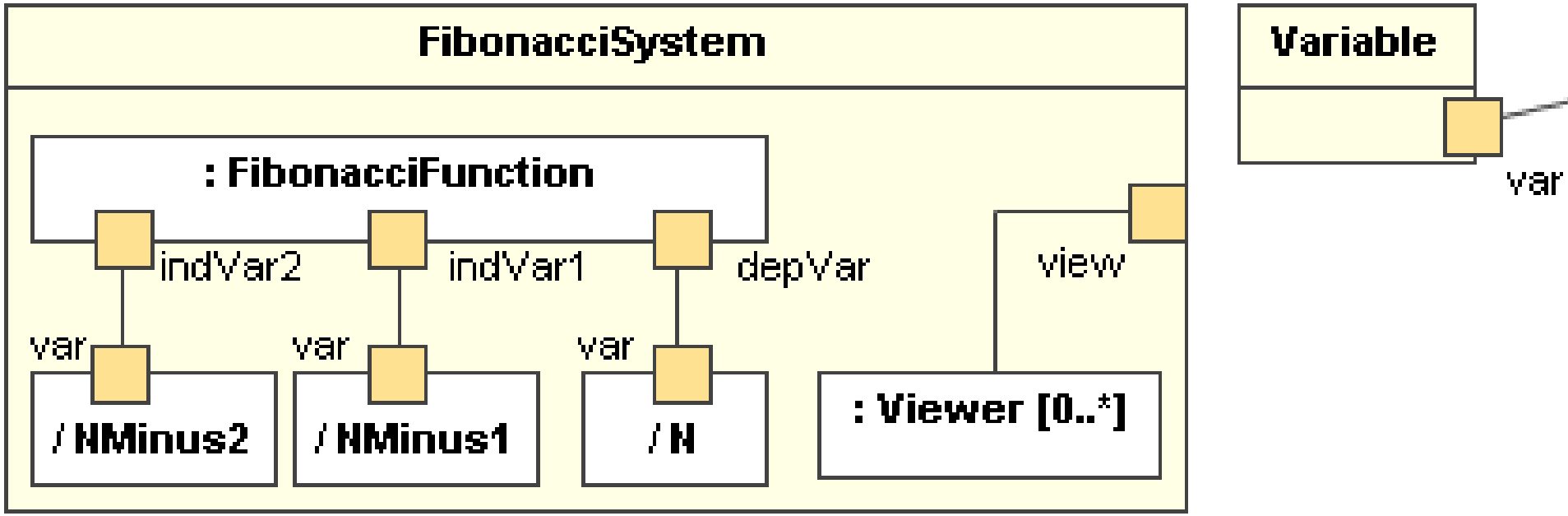
UML2.0 – Diagrame de Structură 2

- **Diagramă de Componente:** componentele sistemului şi legăturile între componente



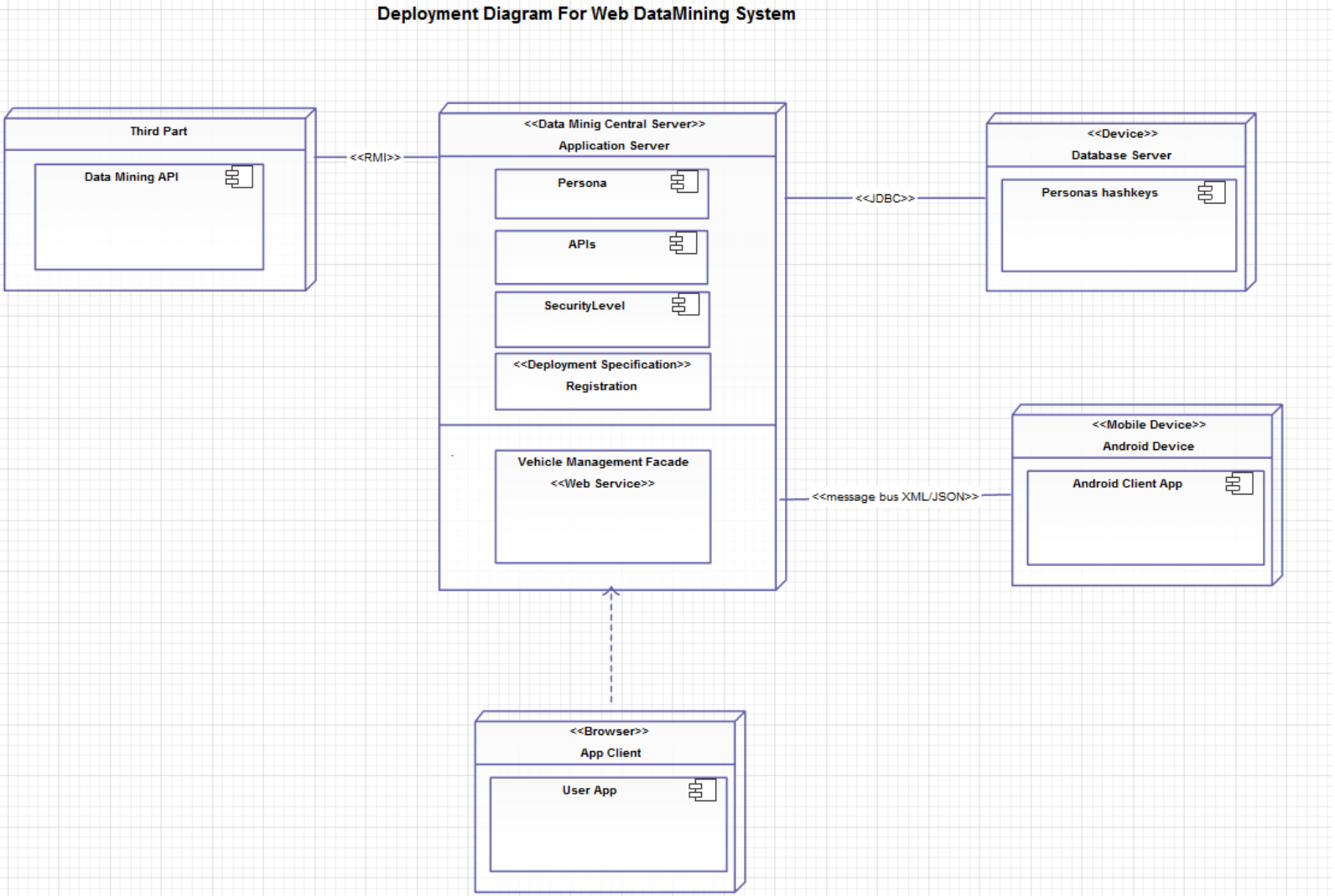
UML2.0 – Diagrame de Structură 3

- **Diagrame structură composită**: structura internă



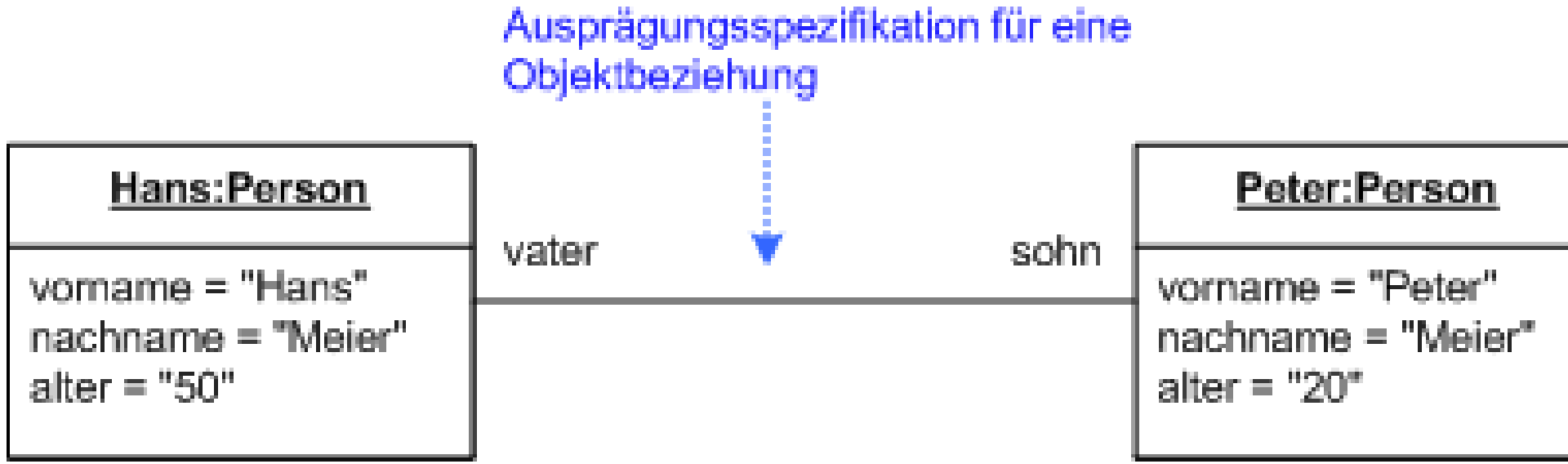
UML2.0 – Diagrame de Structură 4

- **Diagramă de Deployment**: modelarea structurii hardware



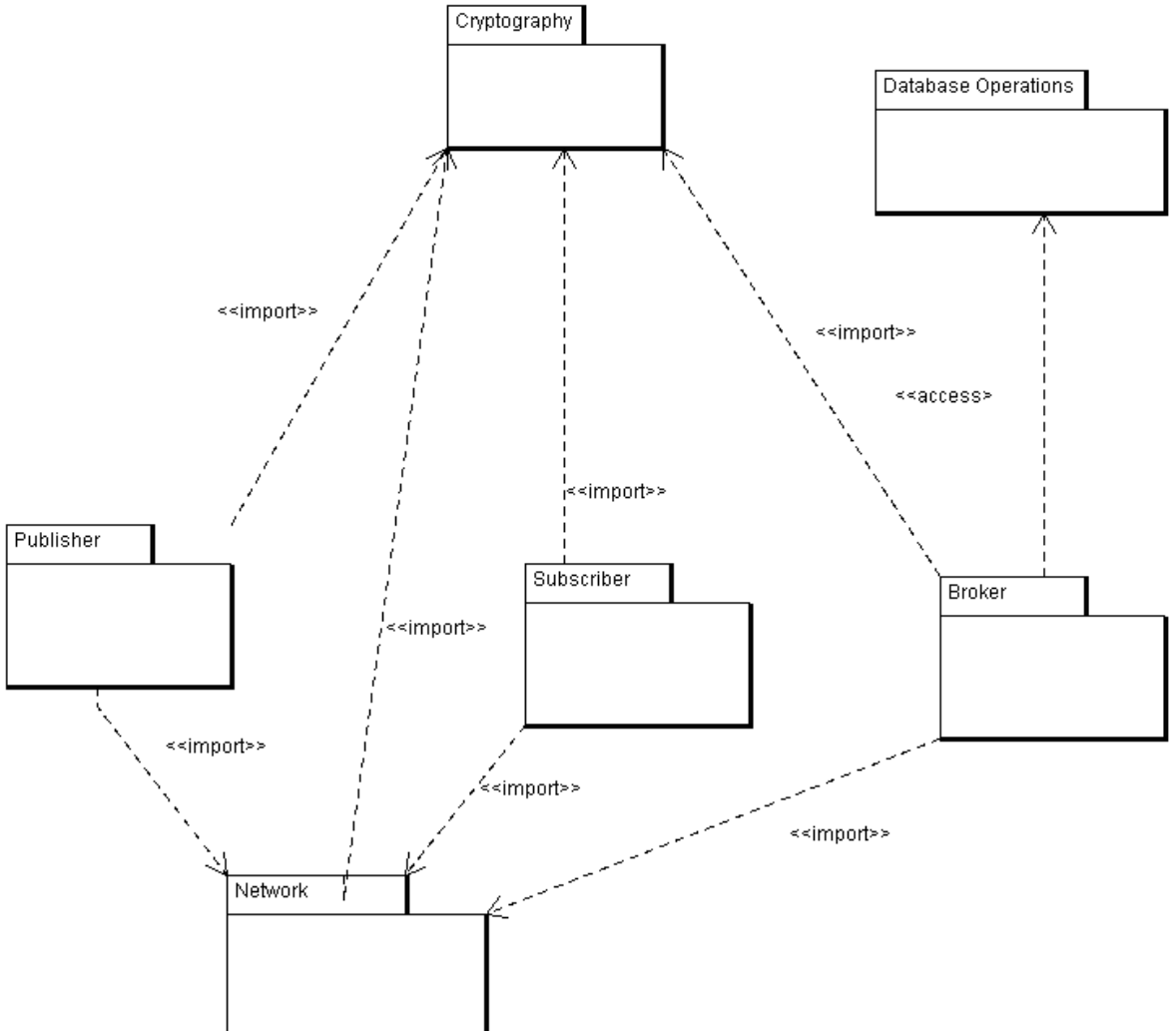
UML2.0 – Diagrame de Structură 5

- **Diagramă de obiecte**: structura sistemului la un moment dat



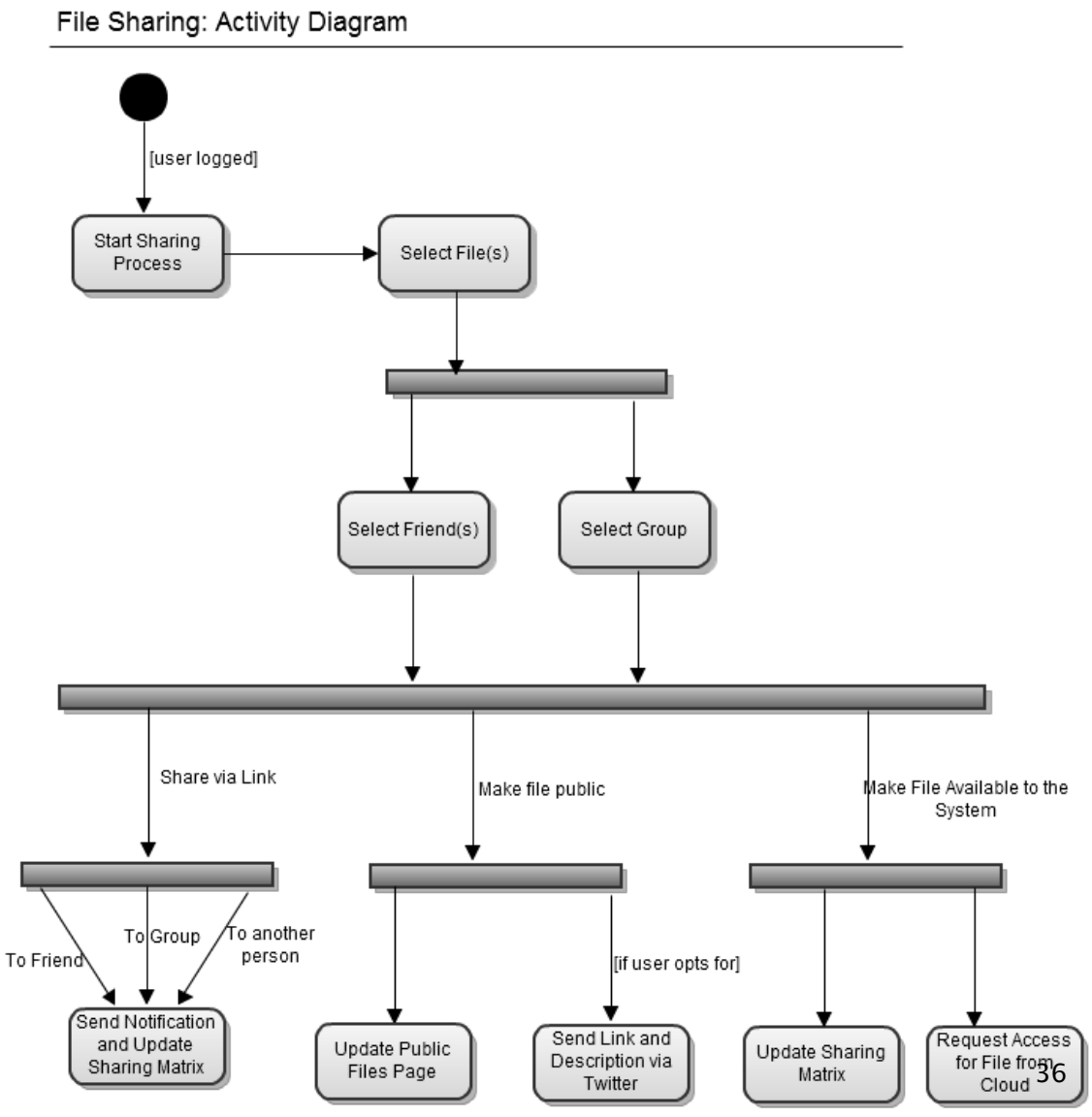
UML2.0 – Diagrame de Structură 6

- **Diagramă de pachete:** împărţirea sistemului în pachete şi relaţiile din



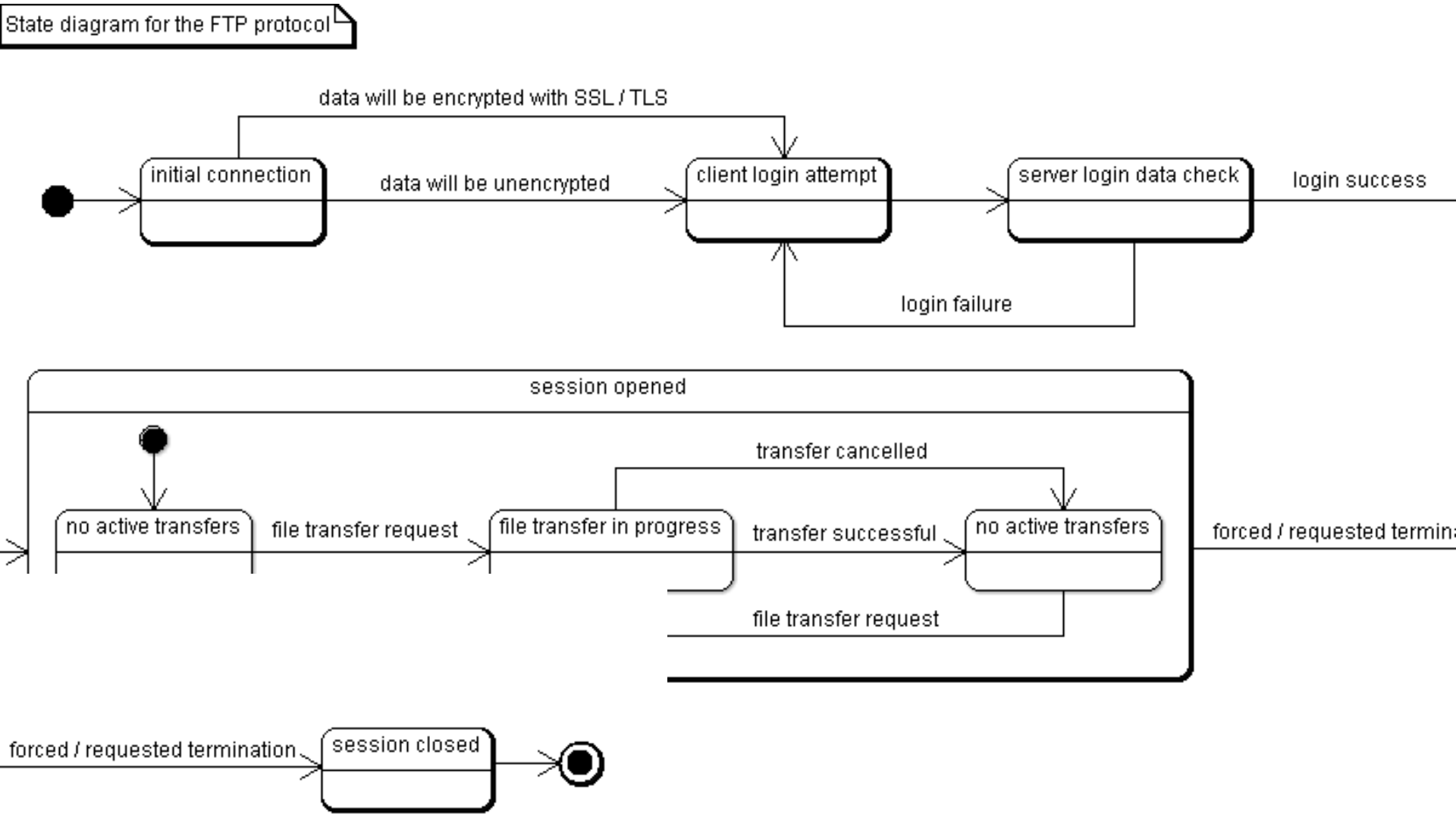
UML 2.0 – Diagrame Comportamentale 1

- **Diagrame de activitate**: prezentare business şi a fluxului de activităţi



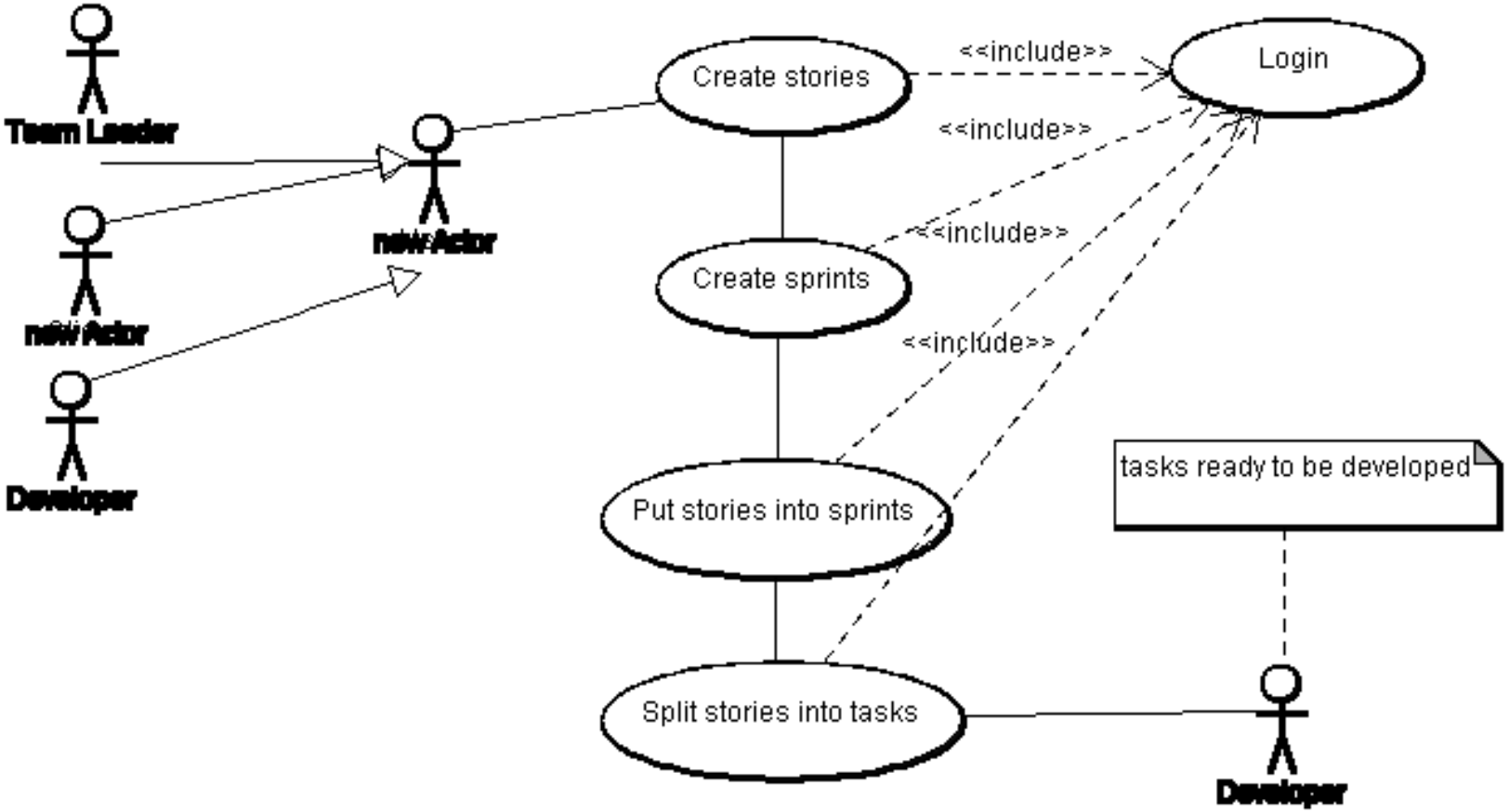
UML 2.0 – Diagrame Comportamentale 2

- **Diagrame de stare**: pentru a prezenta stările obiectelor



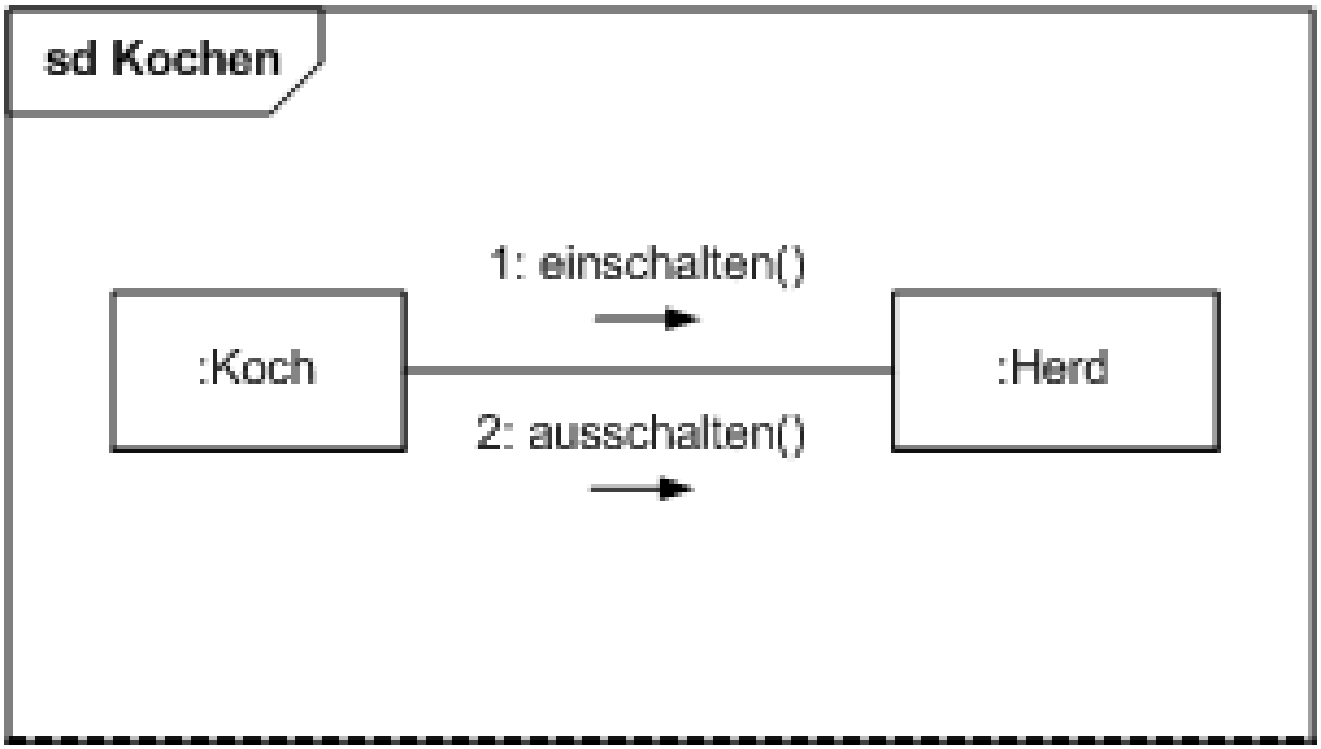
UML 2.0 – Diagrame Comportamentale 3

- **Diagrame Use Case**: prezintă funţionalităţile sistemului folosind actori, use case-uri şi dependenţe între ele



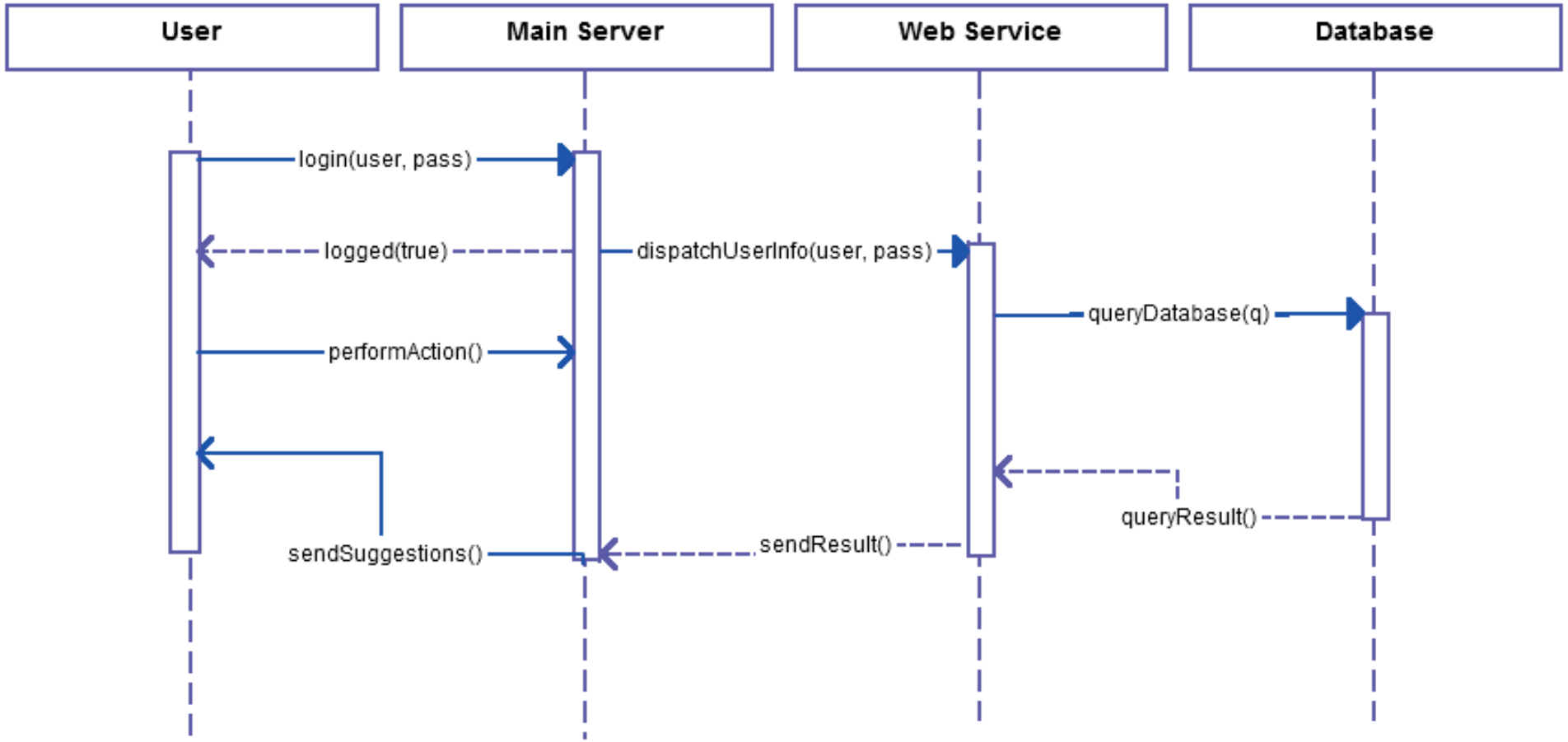
UML 2.0 – Diagrame de interacţiuni 1

- **Diagrama de comunicare**: arată interacţiunile între obiecte (comportamentul dinamic al sistemului) (**actori**: bucătar, aragaz, **acțiuni**: gătirea, aprinderea, deconectarea)



UML 2.0 – Diagrame de interacţiuni 2

- **Diagramă de secvenţă**: prezintă modul în care obiectele comunică între ele din punct de vedere al trimiterii de mesaje



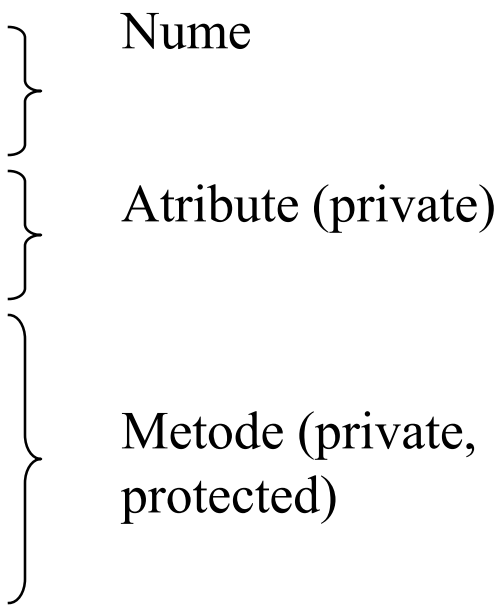
[online diagramming & design] creately.com

Diagrama de clase – Class Diagram

- **Scop**:
  - Modelează vocabularul sistemului ce trebuie dezvoltat
  - Surprinde conexiunile semantice sau interacţiunile care se stabilesc între elementele componente
  - Folosită pentru a modela structura unui program
- **Conţine**
  - Clase/Interfeţe
  - Obiecte
  - Relaţii (Asociere, Agregare, Generalizare, Dependenţă)

Clase

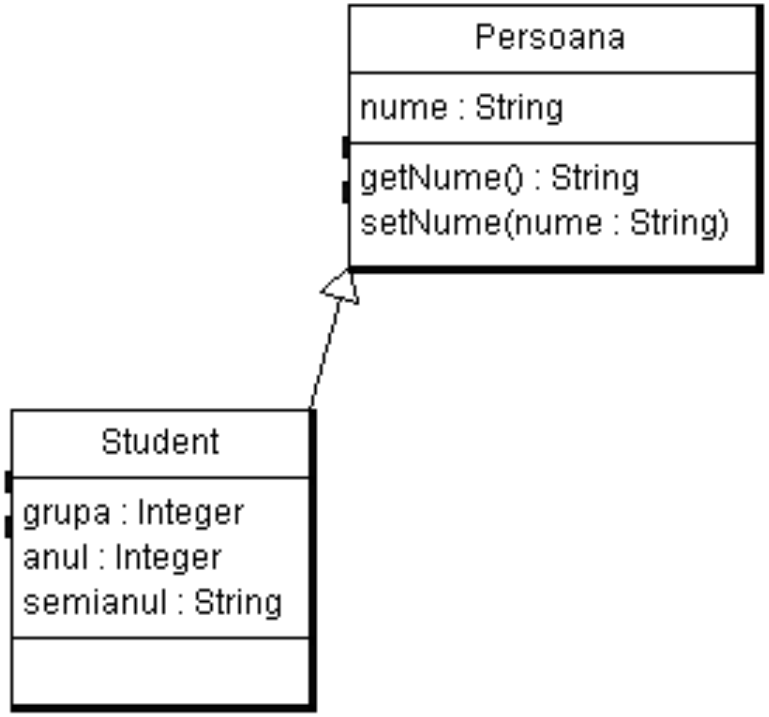
- Modelează vocabularul = identifică conceptele pe care clientul sau programatorul le foloseşte pentru a descrie soluţia problemei
- Elementele unei clase:
  - Nume: identifică o clasă
  - Atribute: proprietăţi ale clasei
  - Metode: implementarea unui serviciu care poate fi cerut oricărei instanţe a clasei



Relaţii – Generalizare – C#

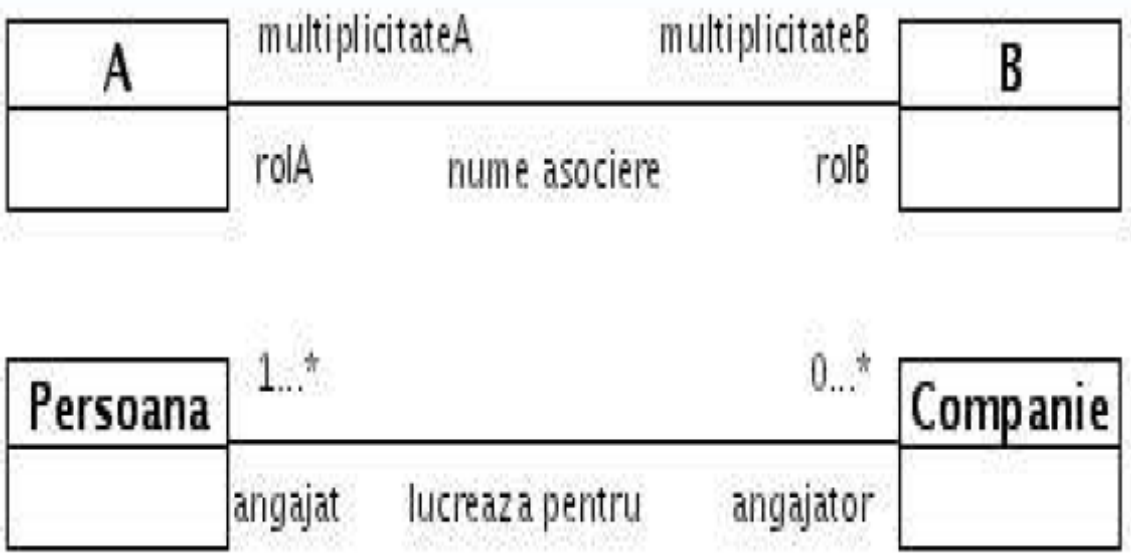
- Modelează conceptul de moştenire între clase
- Mai poartă denumirea de relaţie de tip *is a* (este un/este o)

ArgoUML – Relaţia de generalizare



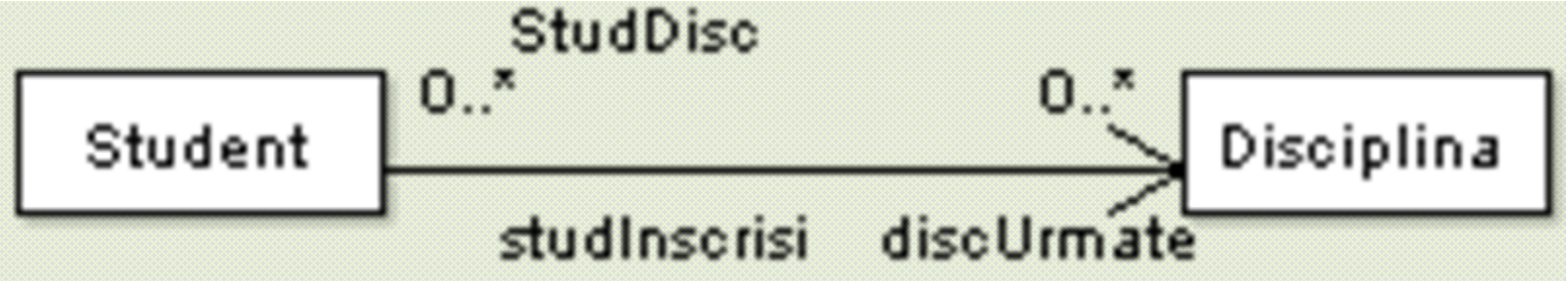
Relaţii – Asociere

- Exprimă o conexiune semantică sau o interacţiune între obiecte aparţinând diferitelor clase
- Pe măsura ce sistemul evoluează noi legaturi între obiecte pot fi create, sau legături existente pot fi distruse
- O asociere interacţionează cu obiectele sale prin intermediul capetelor de asociere
- Elemente:
  - Nume: descrie relaţia
  - Capete de asociere
    - Nume = rolul jucat de obiect în relaţie
    - Multiplicitate = câte instanţe ale unei clase corespund unei singure instanţe ale celeilalte clase



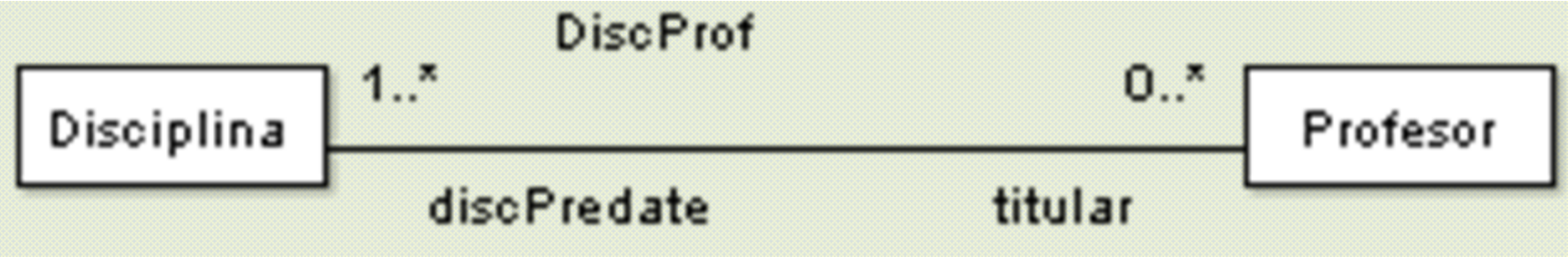
Relaţia de asociere 1

- Relaţia Student – Disciplină
  - **Student**: urmez 0 sau mai multe discipline, cunosc disciplinele pe care le urmez;
  - **Disciplină**: pot fi urmată de mai mulţi studenţi, **nu** cunosc studenţii care mă urmează



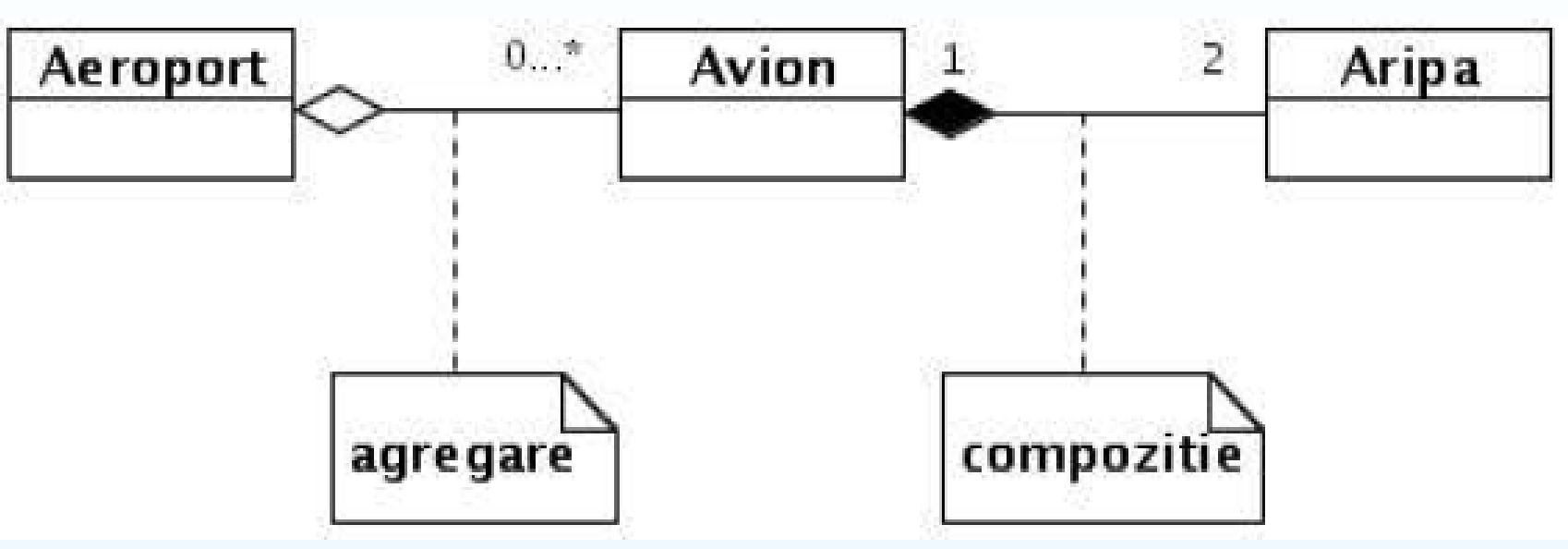
Relaţia de asociere 2

- Relaţia Disciplină – Profesor
  - **Disciplină**: sunt predată de un profesor, îmi cunosc titularul
  - **Profesor**: pot preda mai multe discipline, cunosc disciplinele pe care le predau

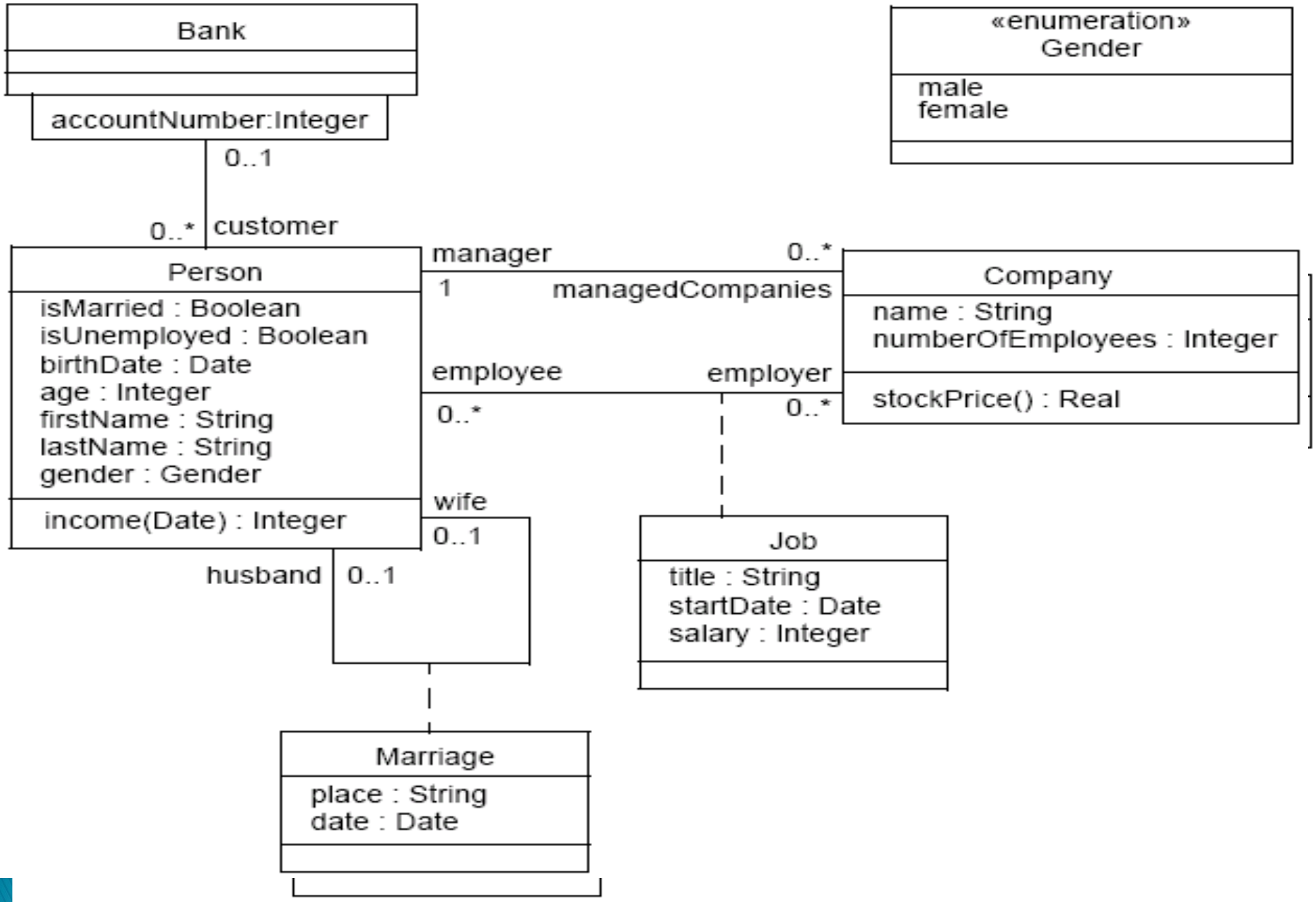


Relaţii – Agregare

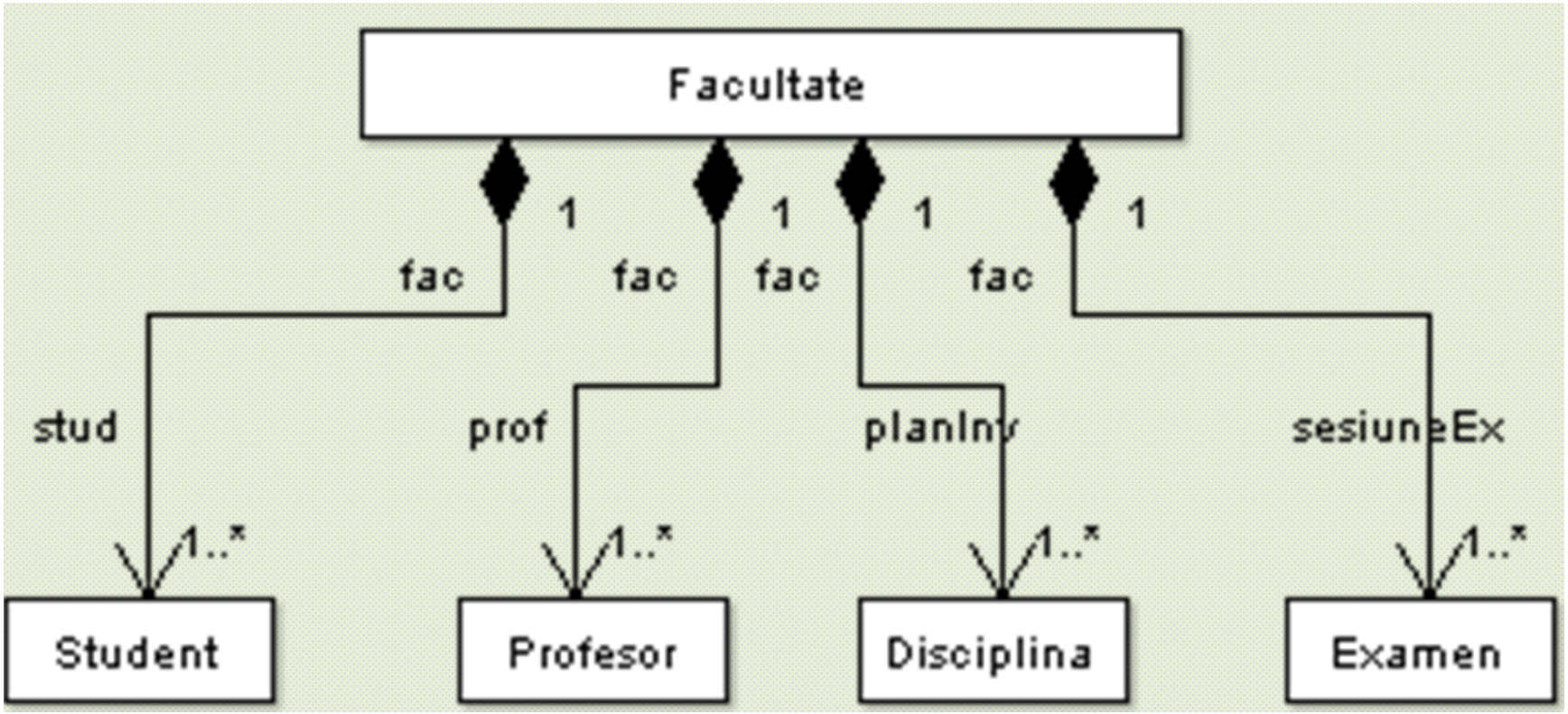
- Este un caz particular al relaţiei de asociere
- Modelează o relaţie de tip parte-întreg
- Poate avea toate elementele unei relaţii de asociere, însă în general se specifică numai multiplicitatea
- Se foloseşte pentru a modela situaţiile între care un obiect este format din mai multe componente.



Exemplul


Bank
accountNumber:Integer
0..1
0..*
customer
«enumeration»
Gender
male
female
Person
isMarried : Boolean
isUnemployed : Boolean
birthDate : Date
age : Integer
firstName : String
lastName : String
gender : Gender
income(Date) : Integer
manager
1
0..*
managedCompanies
Company
name : String
numberOfEmployees : Integer
stockPrice() : Real
employee
0..*
employer
0..*
wife
0..1
husband
0..1
Job
title : String
startDate : Date
salary : Integer
Marriage
place : String
date : Date

Relaţia de Compoziţie (“hasA”)



Studiu de Caz

- Obţinerea Studenţilor Bursieri
  - Actori
  - Scenarii de utilizare
  - Clase

Concluzii

- Modelare – De ce?
- Limbaje grafice
- UML
  - Structurale: clase
  - Comportamentale: use-case
  - De interacțiuni

Întrebări

- 1) Dați exemplu de o situație în care e suficientă doar o diagramă sau o schemă pentru a realiza ceva.
- 2) Dați exemplu de o situație în care e nevoie de mai mult de o diagramă sau o schemă pentru a realiza ceva.
- 3) Care e legătura dintre diagramele de tip use-case și diagramele de clase?

Bibliografie

- OMG Unified Modeling Language**TM (OMG UML), Infrastructure,** Version 2.2, May 2008, http://www.omg.org/docs/ptc/08-05-04.pdf
- **ArgoUML User Manual,** A tutorial and reference description, http://argouml-stats.tigris.org/documentation/printablehtml/manual/argomanual.html
- Ovidiu Gheorghieş, Curs IP, Cursurile 3, 4
- Diagrame UML, Regie.ro

Links

- OOSE: http://cs-exhibitions.uni-klu.ac.at/index.php?id=448
- ArgoUML: http://argouml-stats.tigris.org/nonav/documentation/manual-0.22/
- Wikipedia