

Ingineria Programării

Cursul 3 – 4 Martie
adiftene@info.uaic.ro

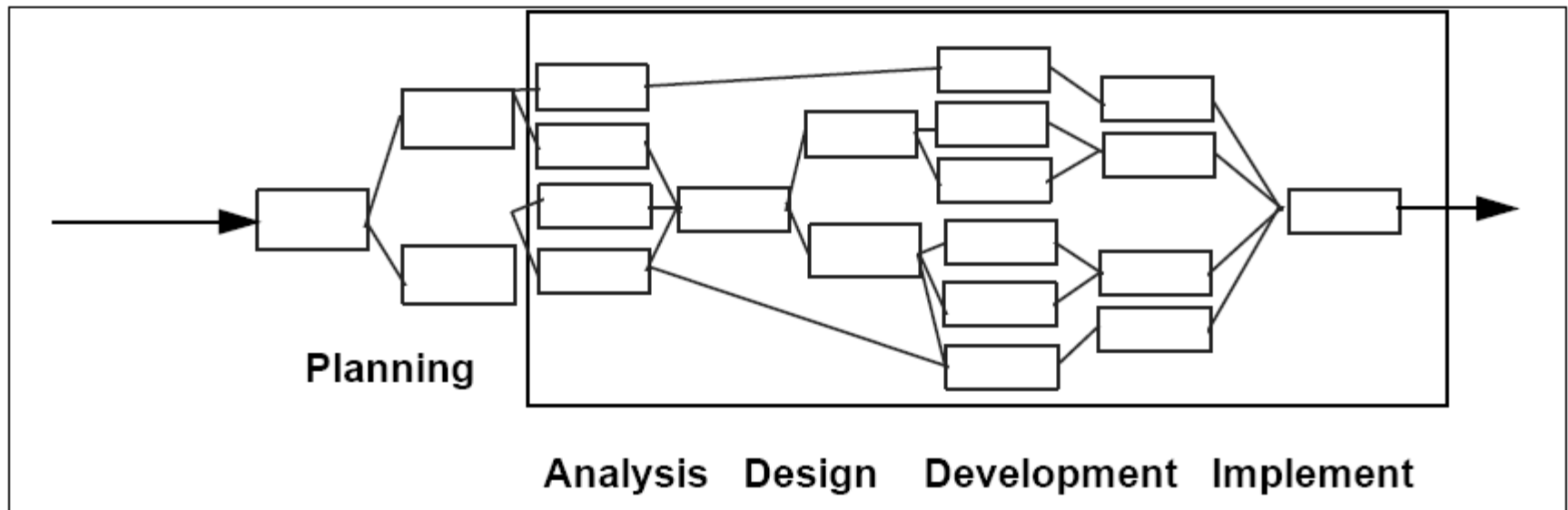
Cuprins

- ▶ Din Cursurile 1, 2...
- ▶ Modelarea
- ▶ Limbaje de Modelare
 - Limbaje Grafice
- ▶ UML – Istoric
- ▶ UML – Definiție
- ▶ UML – Tipuri de Diagrame
- ▶ UML – Diagrame Use Case
- ▶ UML – Diagrame de Clase

Din Cursurile 1, 2

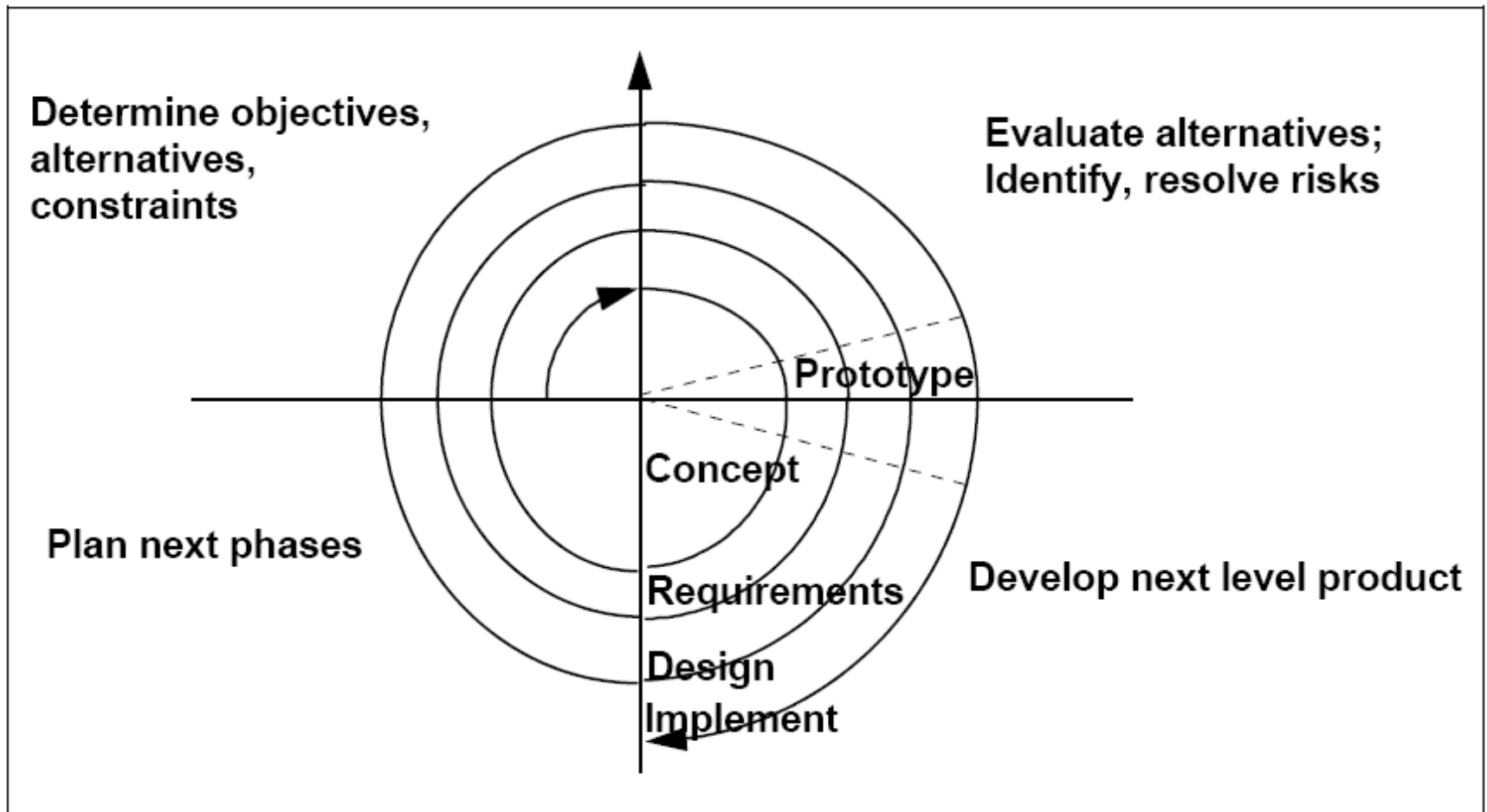
- ▶ Ingineria programării (Software engineering)
- ▶ Etapele dezvoltării programelor
- ▶ Modele de dezvoltare
- ▶ Ingineria cerințelor

Din cursul 1



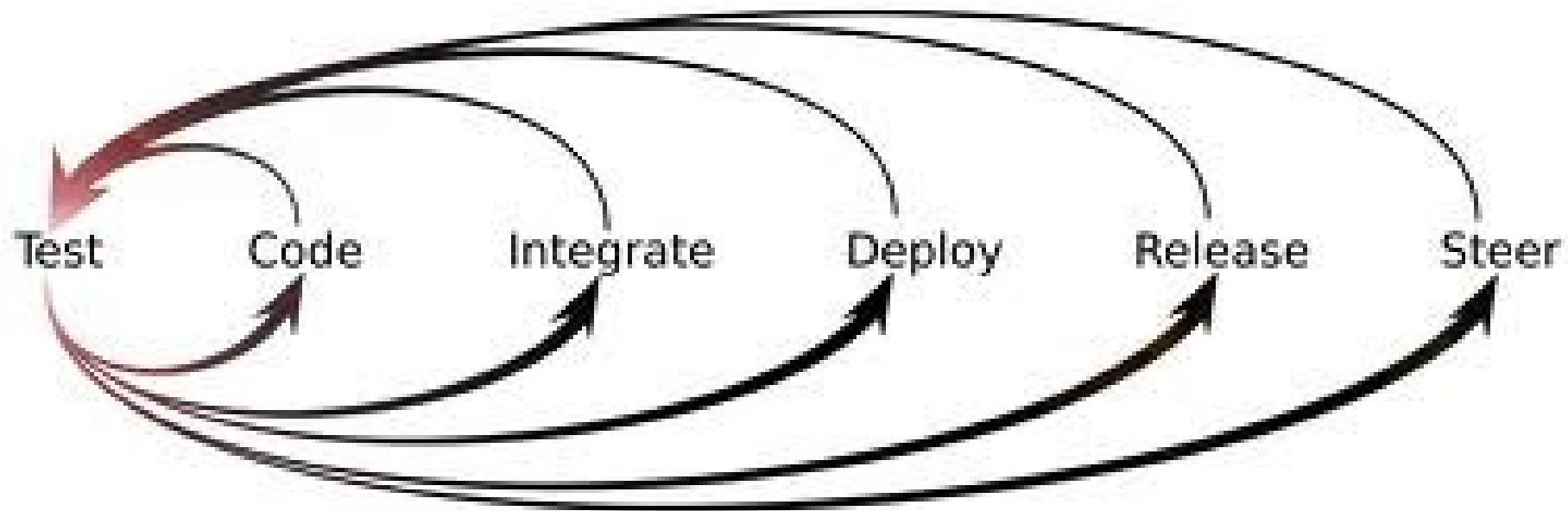
Modelul în cascadă...

Din cursul 1



Modelul în spirală

Din cursul 2

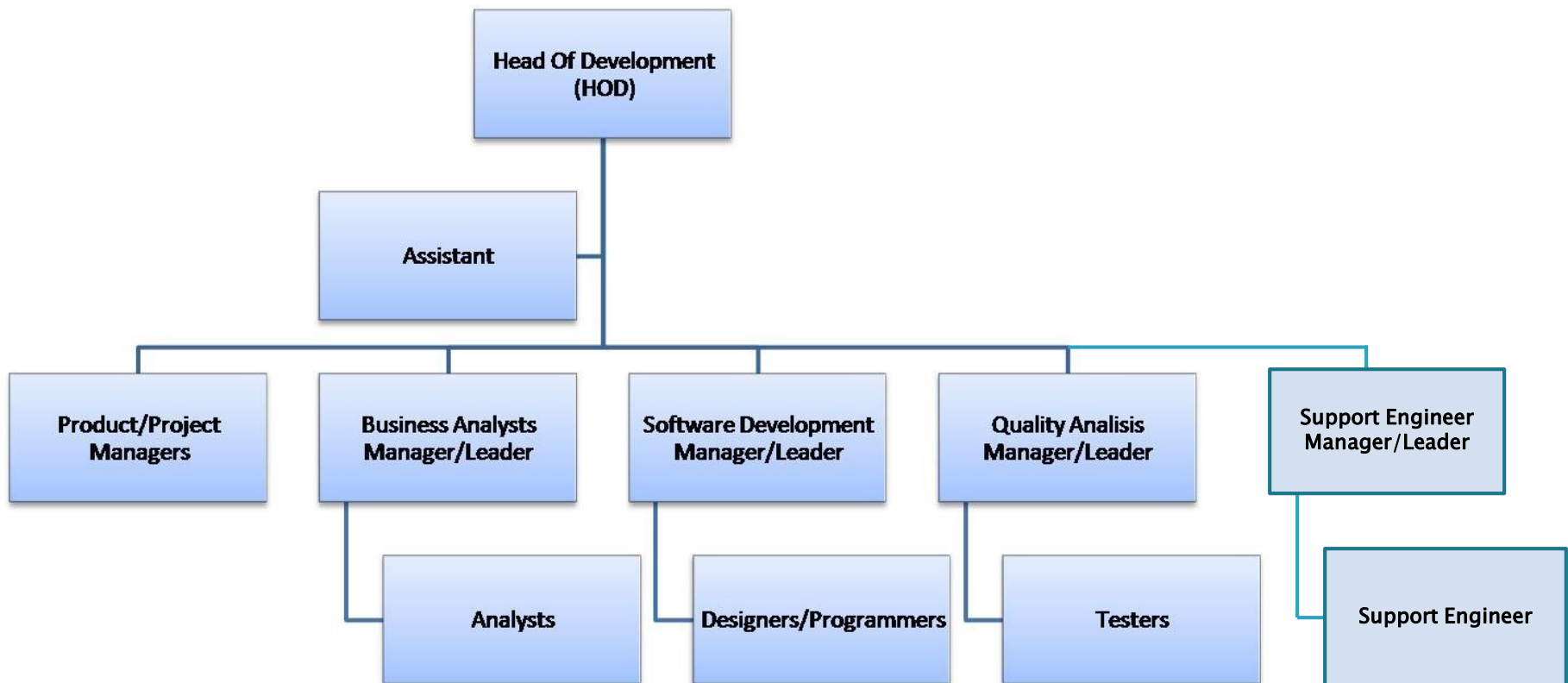


- ▶ XP, TDD...

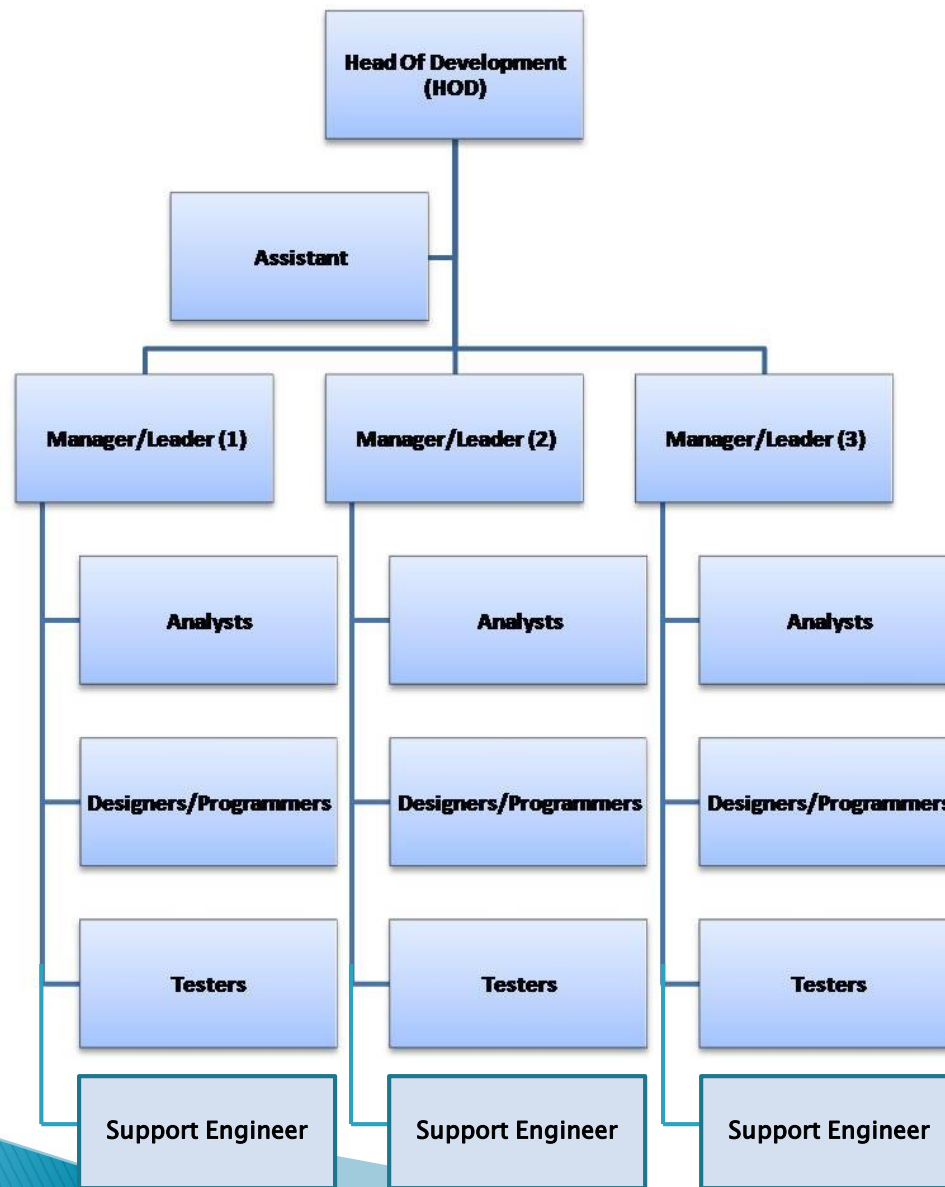
Din cursul 2

- ▶ Ingineria cerințelor:
 - Actori
 - Scenarii

Ierarhia dintr-o firmă de software – Compartimente

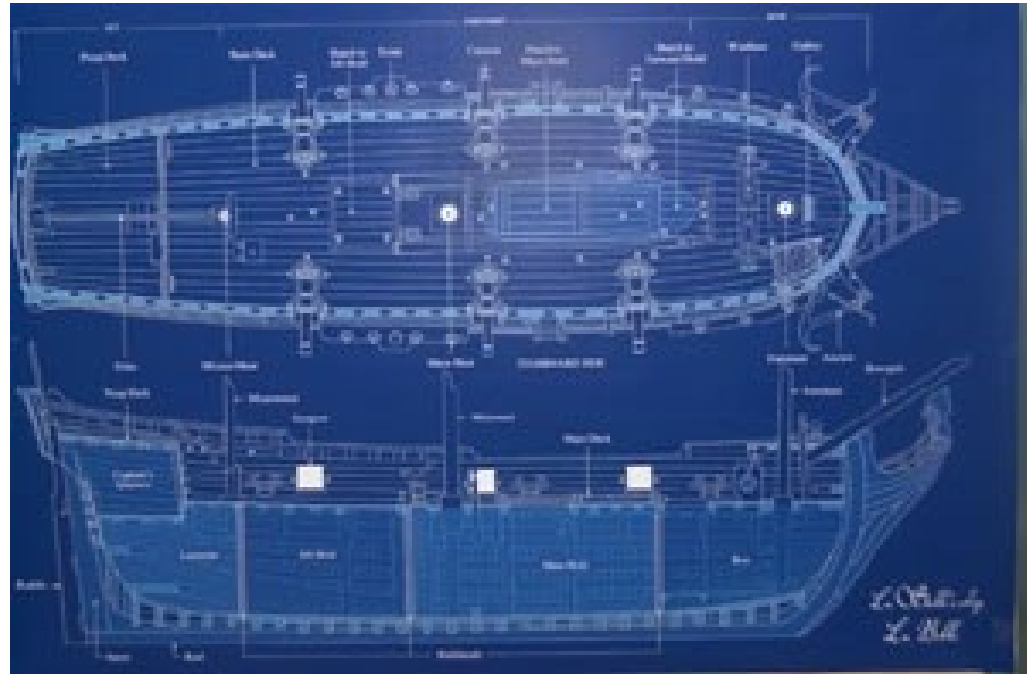


Ierarhia dintr-o firmă de software – Proiecte



Modelarea – De ce?

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



- ▶ De ce modelăm?
 - Pentru a înțelege 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 și/sau a comportării
- ▶ Oferirea unui șablon care să ajute la construcție
- ▶ Documentarea deciziilor luate

Modelarea Arhitecturii

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

Principiile Modelării

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

Limbaje de Modelare

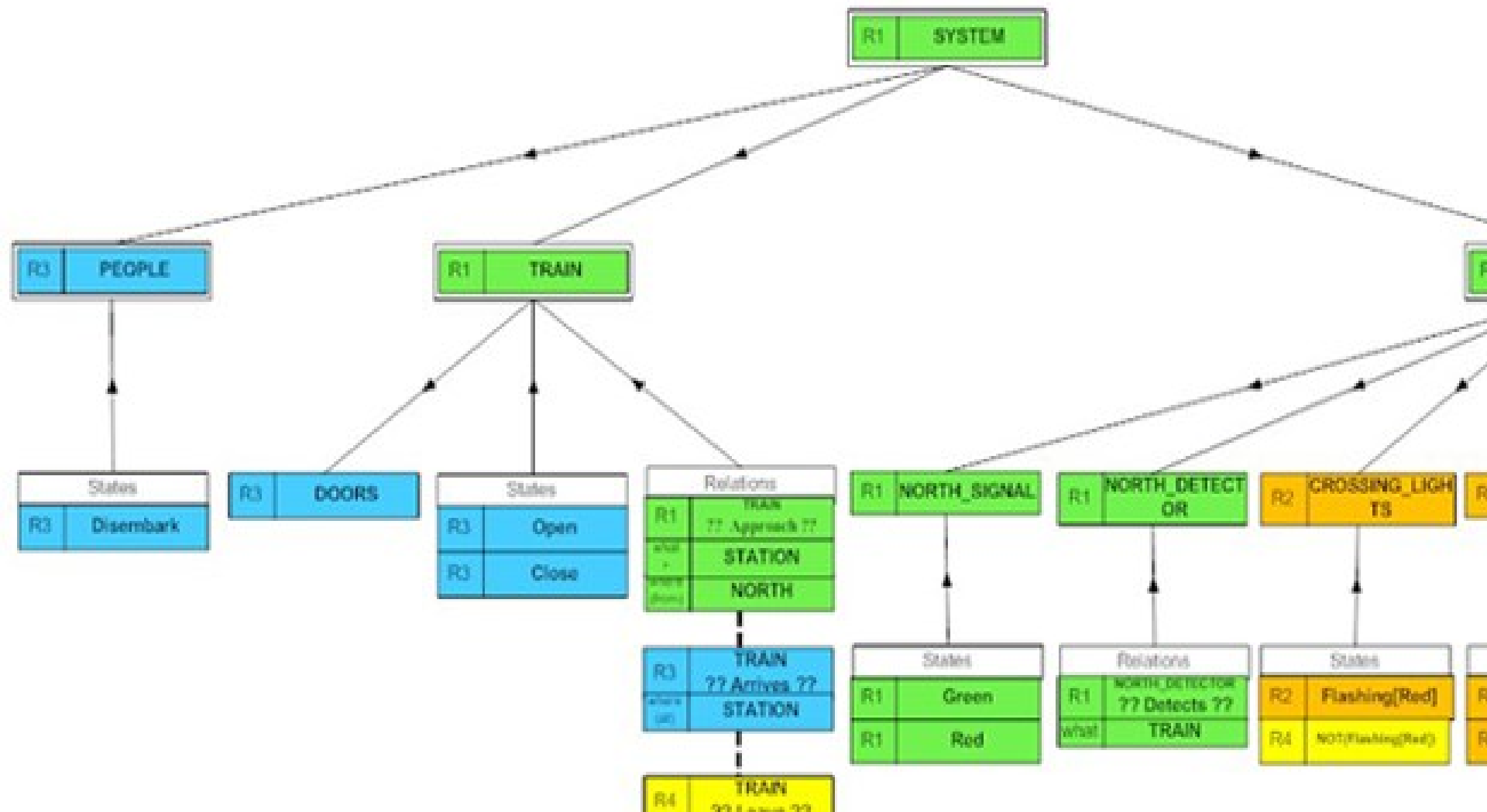
- ▶ Analiza și proiectarea unui proiect trebuie făcute înainte de realizarea codului
- ▶ În prezent, se acordă o atenție deosebită acestei etape, deoarece de ele depind **producerea și refolosirea de software**
- ▶ Pentru analiza și proiectarea programelor s-au creat **limbajele de modelare**
- ▶ **Limbaj 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), rețele Petri, **diagrame UML**
- ▶ **Limbaje Specifice:** modelare algebrică (AML) (pentru descrierea și rezolvarea problemelor de matematică ce necesită putere computațională mare), modelarea domeniilor specifice (DSL), modelarea arhitecturilor specifice (FSML), modelarea obiectelor (object modeling language), modelarea realității virtuale (VRML)

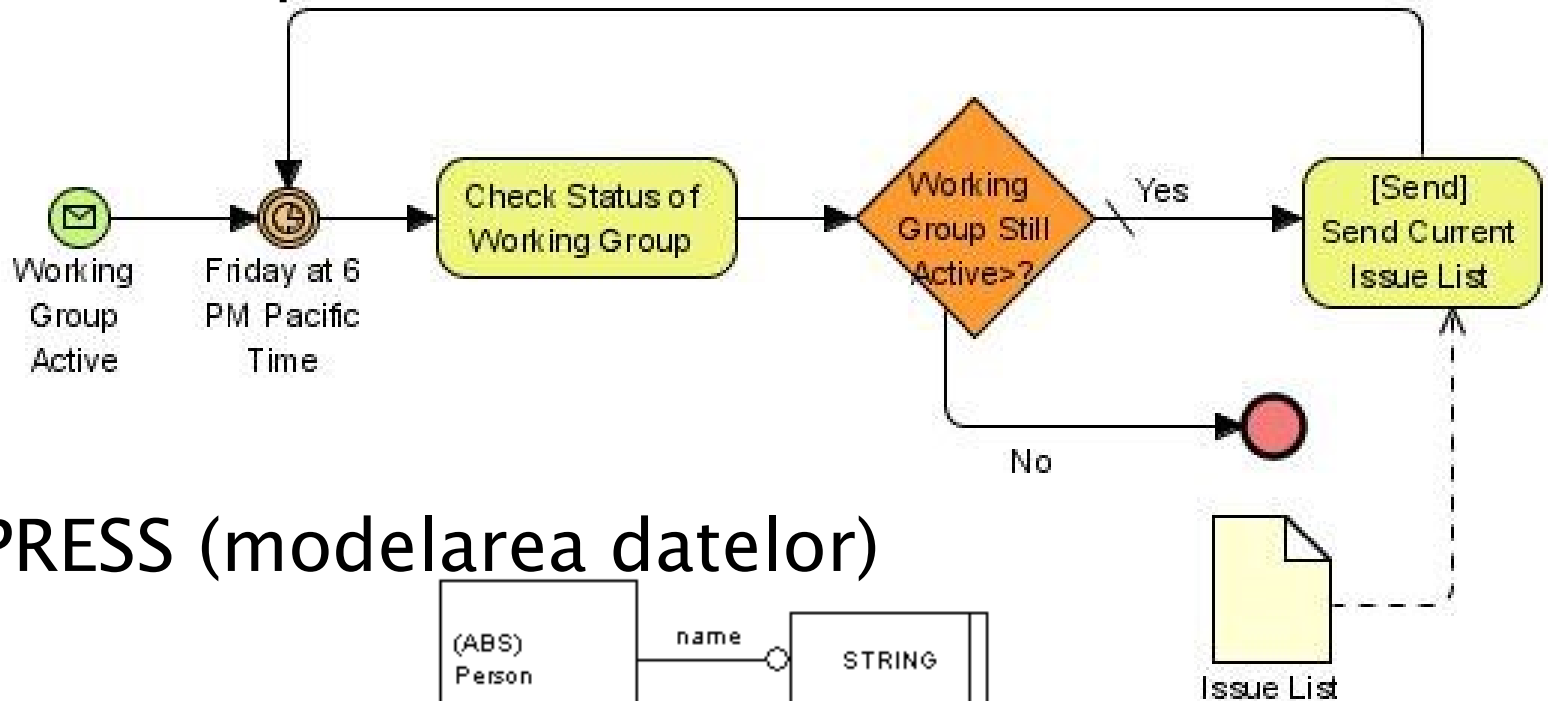
Limbaje Grafice 1

► Arbori comportamentali

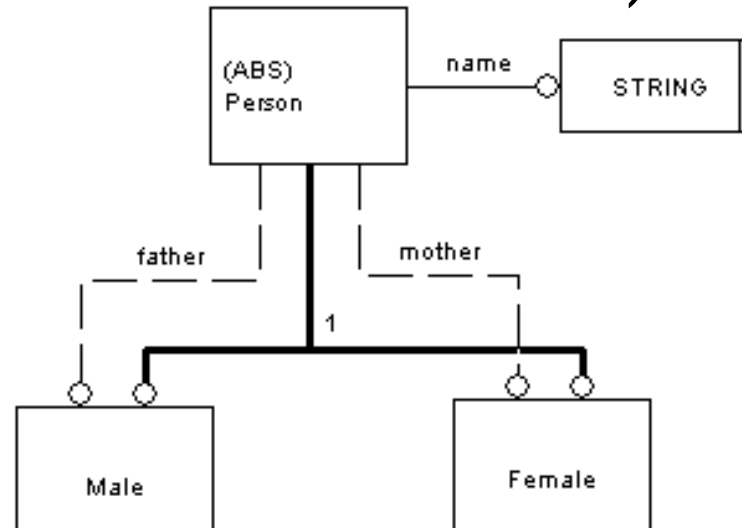


Limbaje Grafice 2

► Modelarea proceselor de business

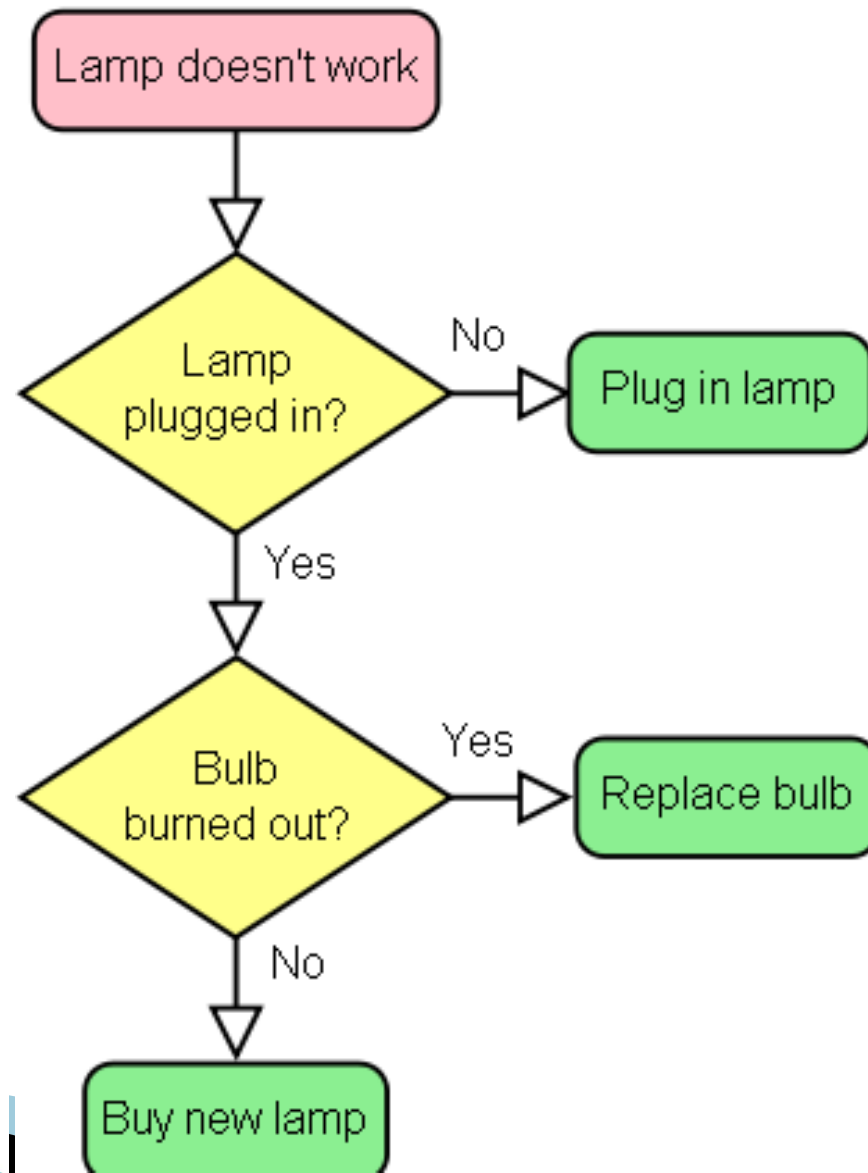


► EXPRESS (modelarea datelor)



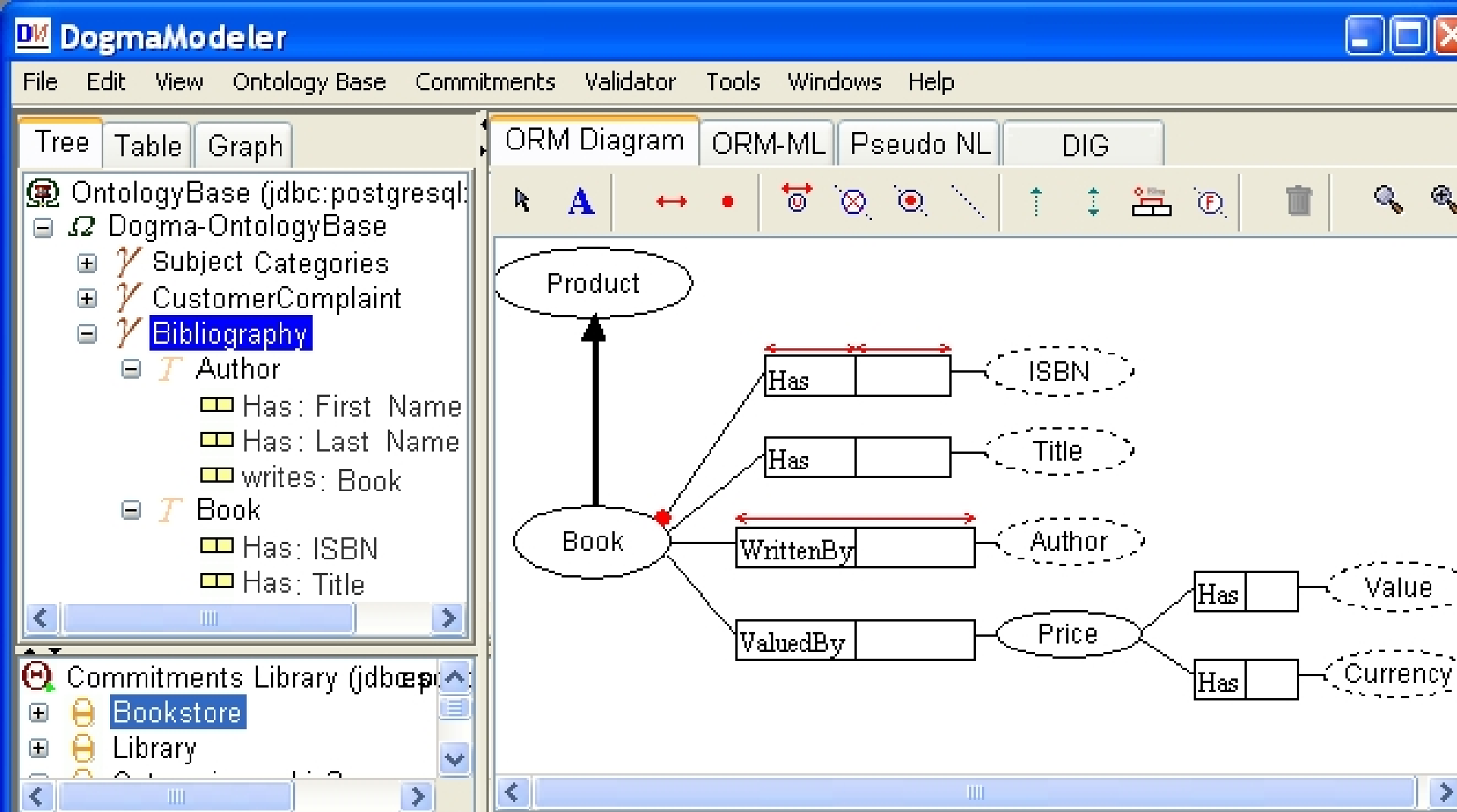
Limbaje Grafice 3

► Flowchart



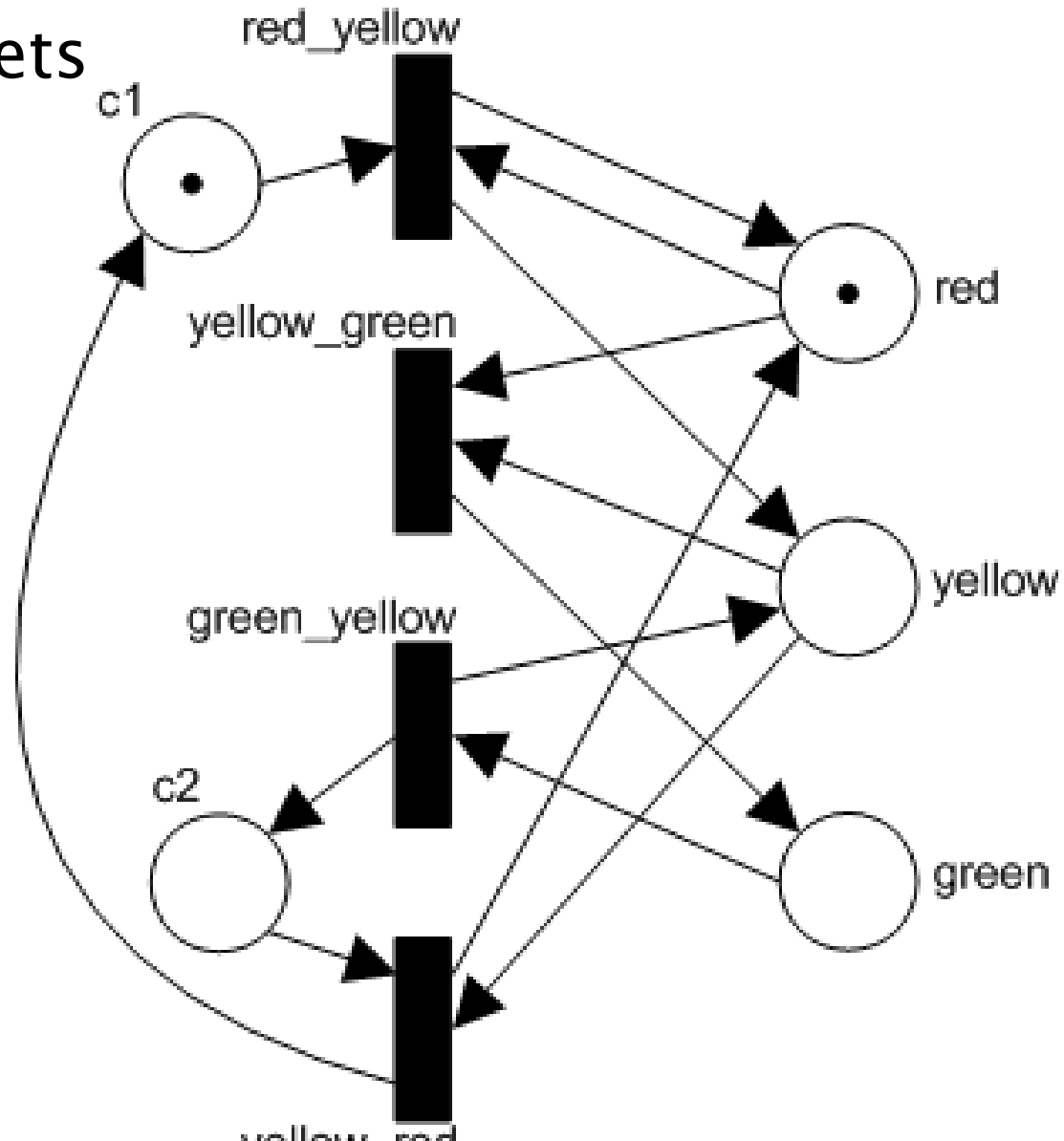
Limbaje Grafice 4

► ORM (Object Role Modeling)



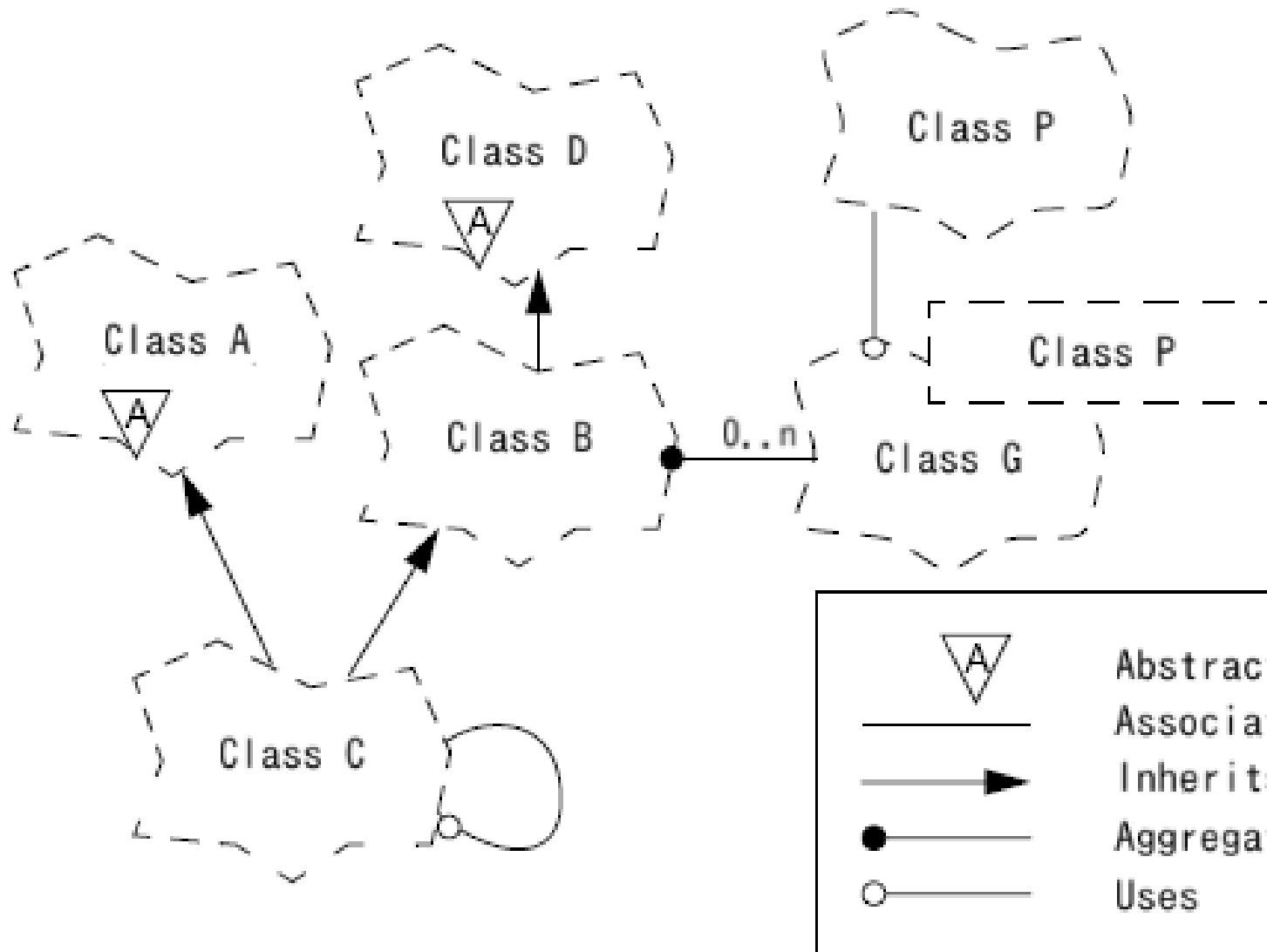
Limbase Grafice 5

► Petri Nets



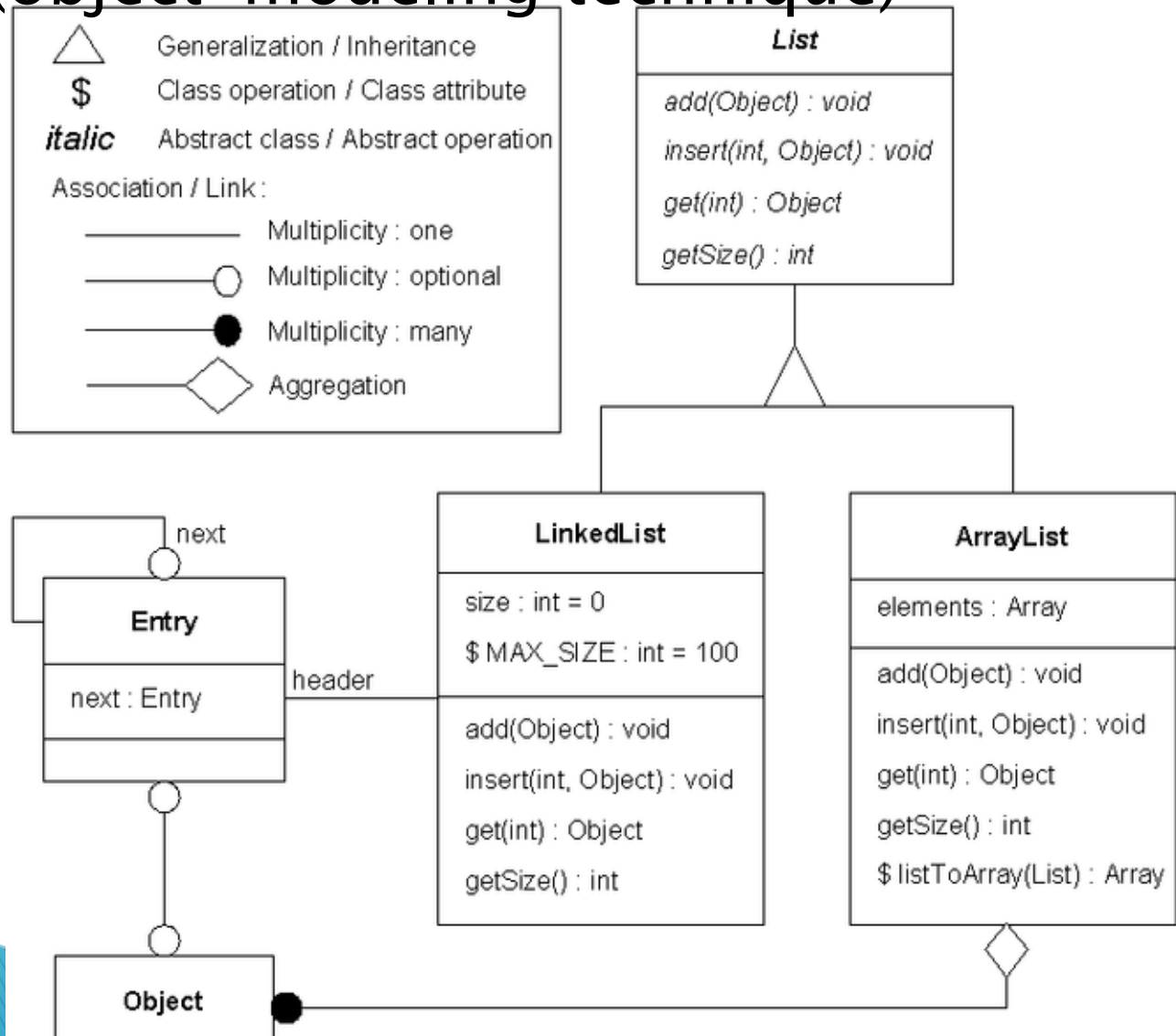
Limbaje Grafice 6

- ▶ Metoda Booch (Grady Booch) – analiza și design oo



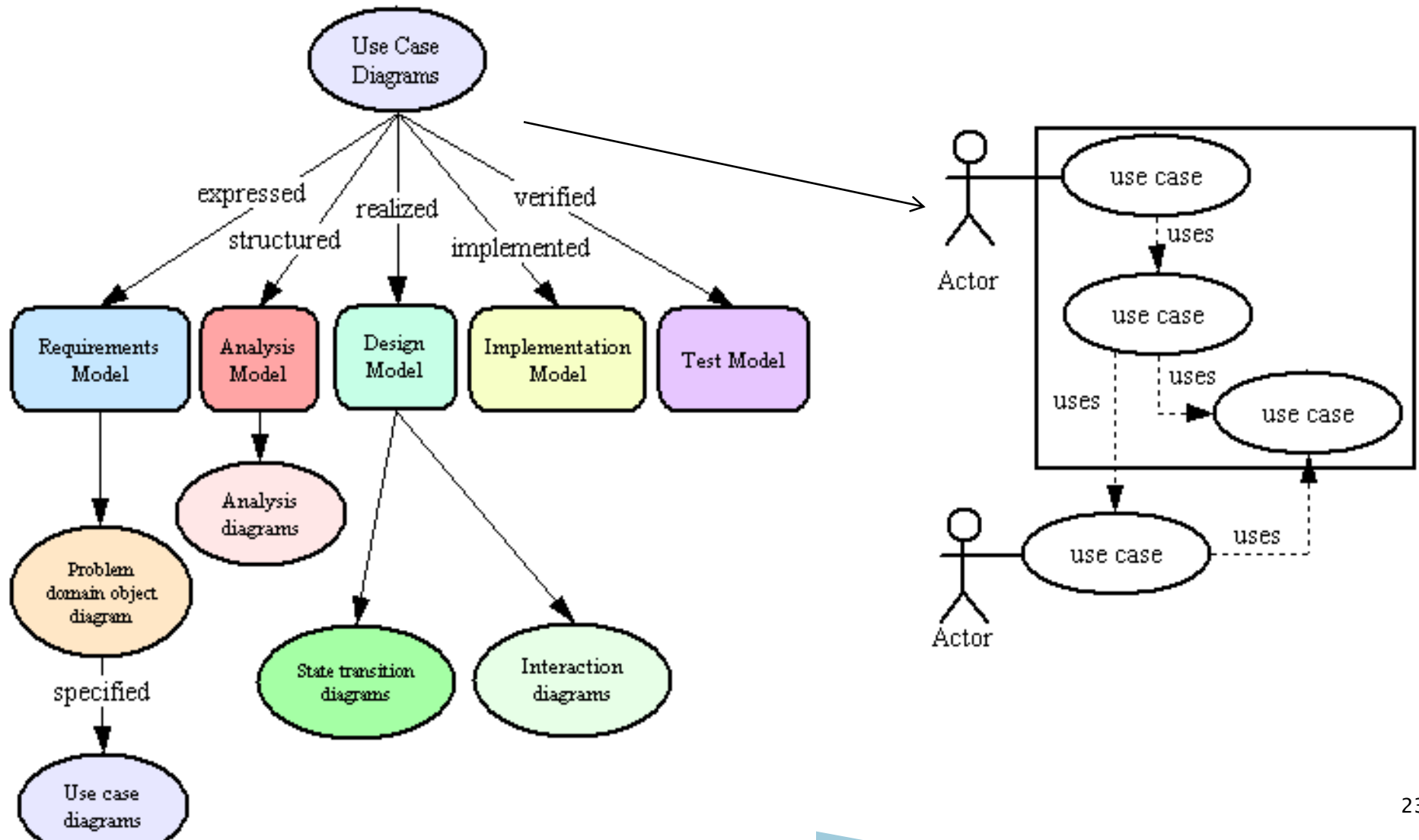
Limbaje Grafice 7

► OMT (object-modeling technique)



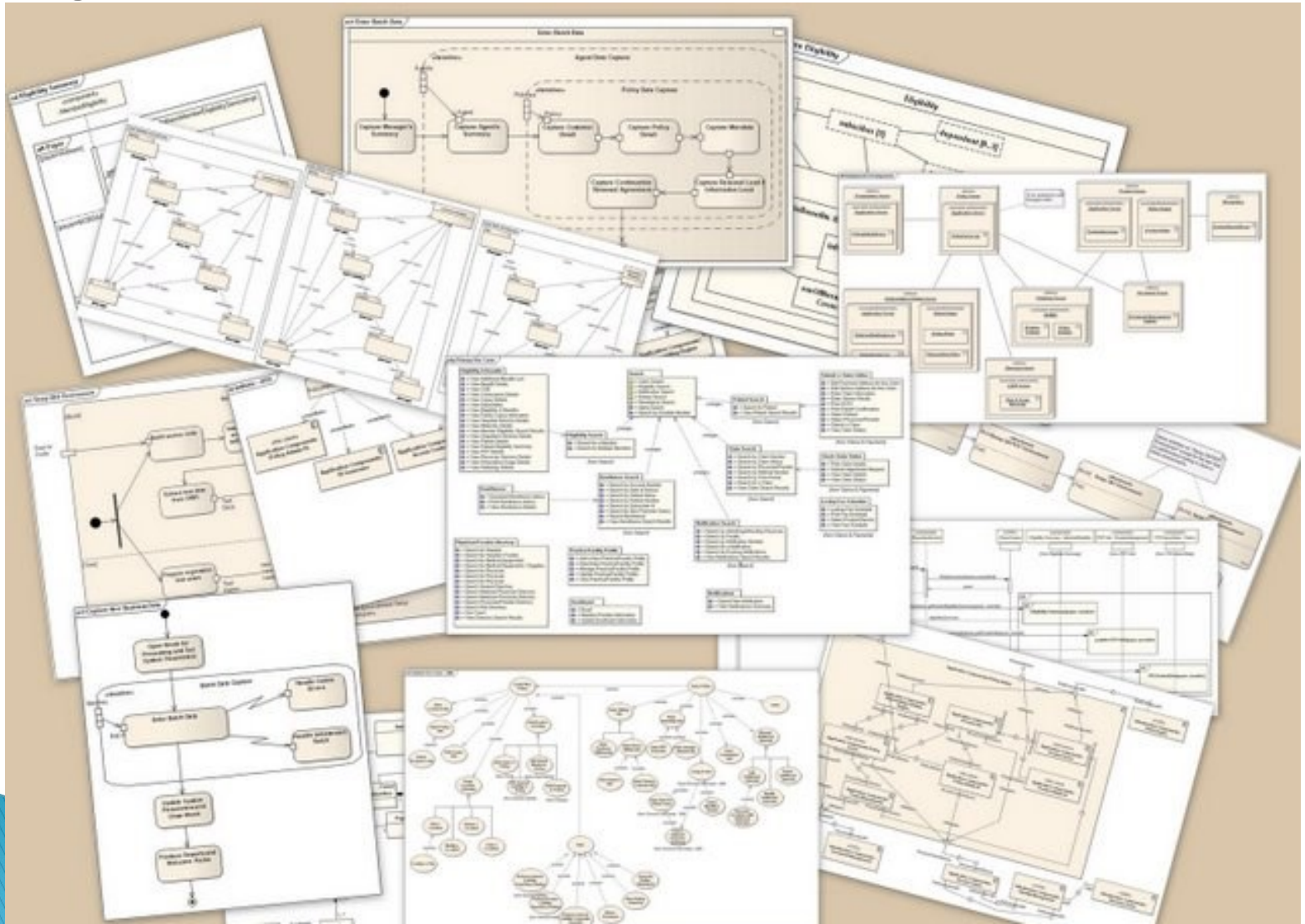
Limbaje Grafice 8

- ▶ OOSE (Object-oriented software engineering)



Limbaje Grafice 9

- Diagrame UML



UML – Introducere

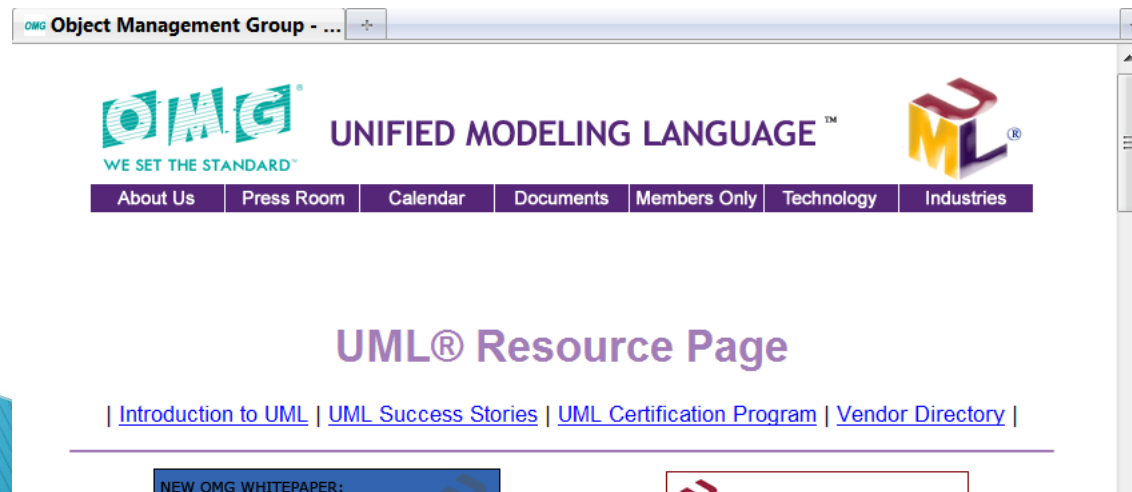
- ▶ UML (Unified Modeling Language) este succesorul celor mai bune trei limbaje OO de modelare anterioare:
 - Booch (Grady Booch)
 - OMT (Ivar Jacobson)
 - OOSE (James Rumbaugh)
- ▶ UML se constituie din unirea acestor limbaje de modelare și în plus are o expresivitate mai mare

UML – Definiție (OMG)

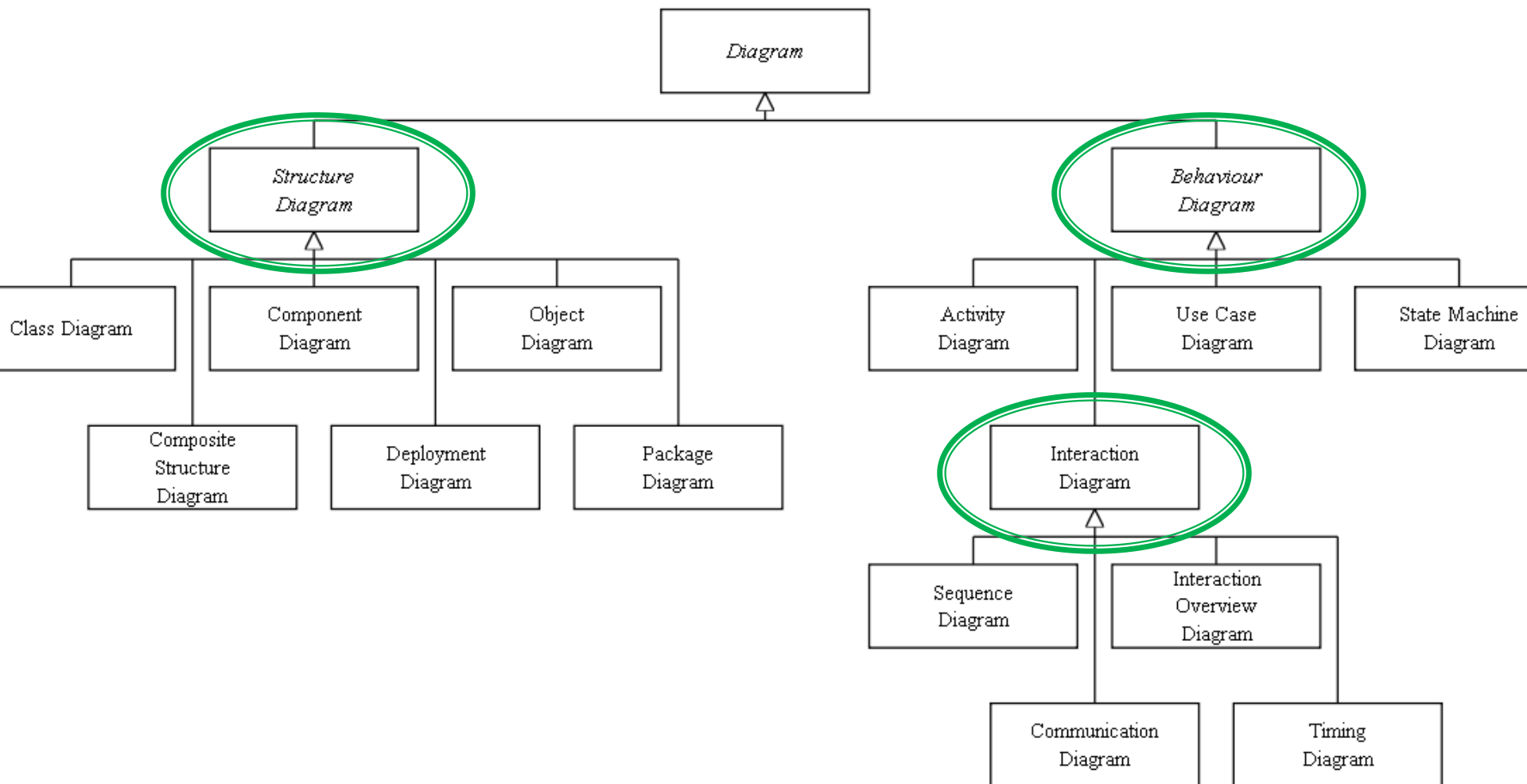
- ▶ *"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.*
- ▶ *The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system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

UML – Standard Internațional

- ▶ 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>

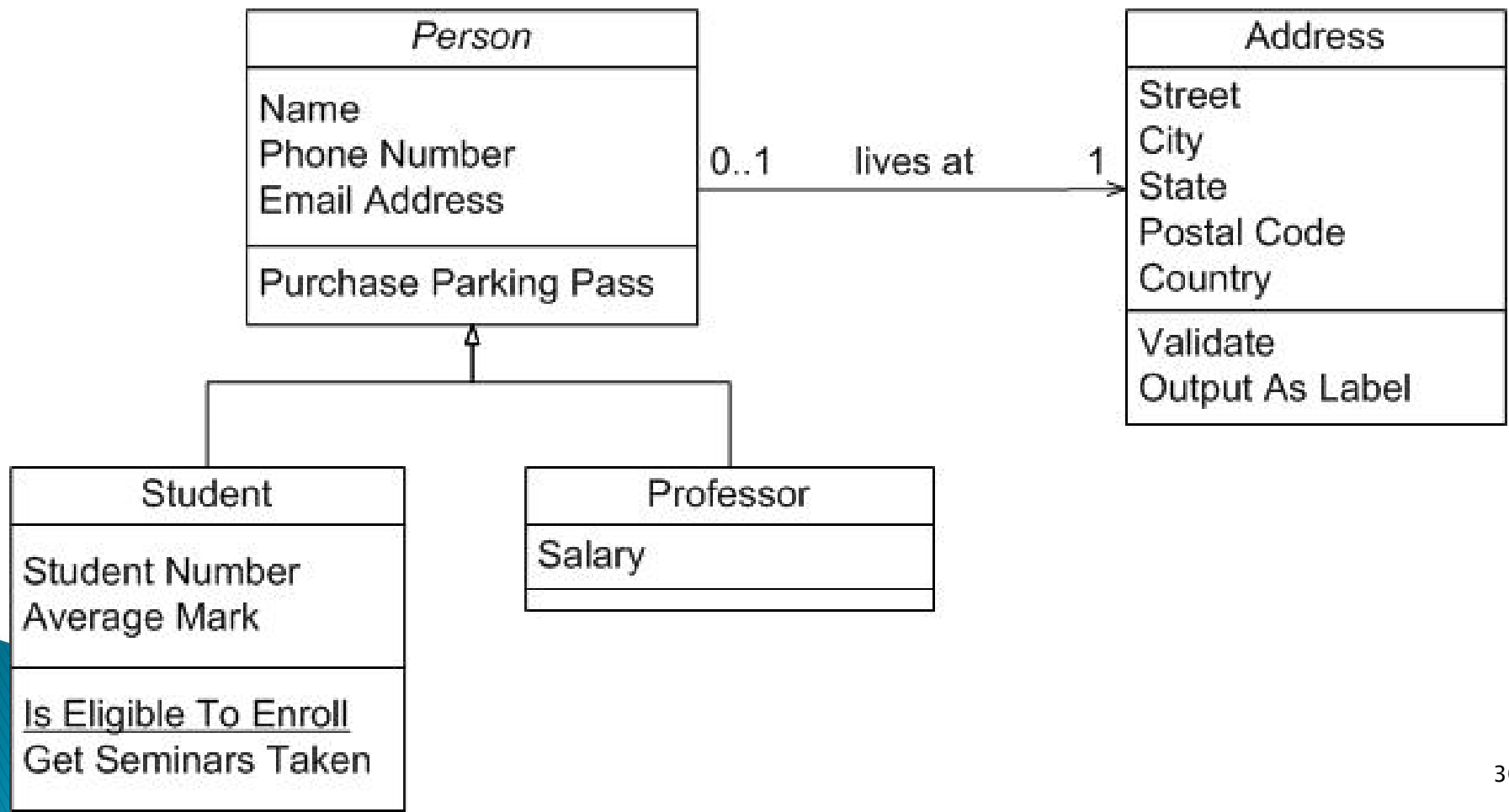


UML2.0 – 13 Tipuri de Diagrame



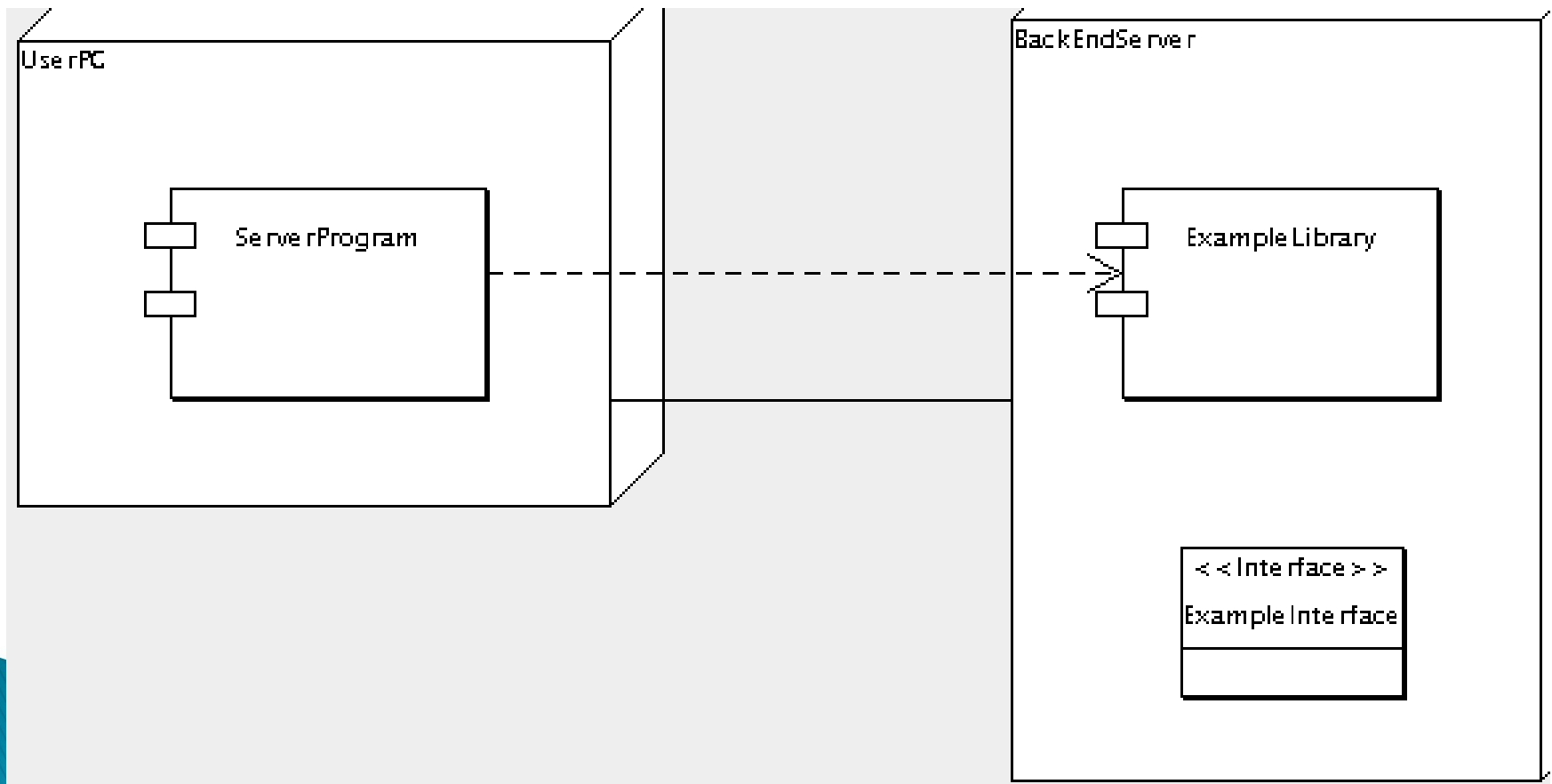
UML2.0 – Diagrame de Structură 1

- ▶ **Diagrame de Clasă:** clasele (atributele, metodele) și relațiile dintre clase



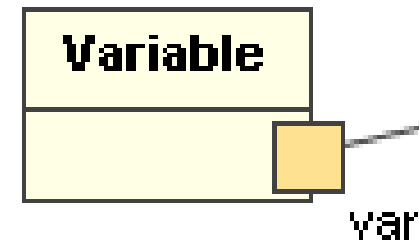
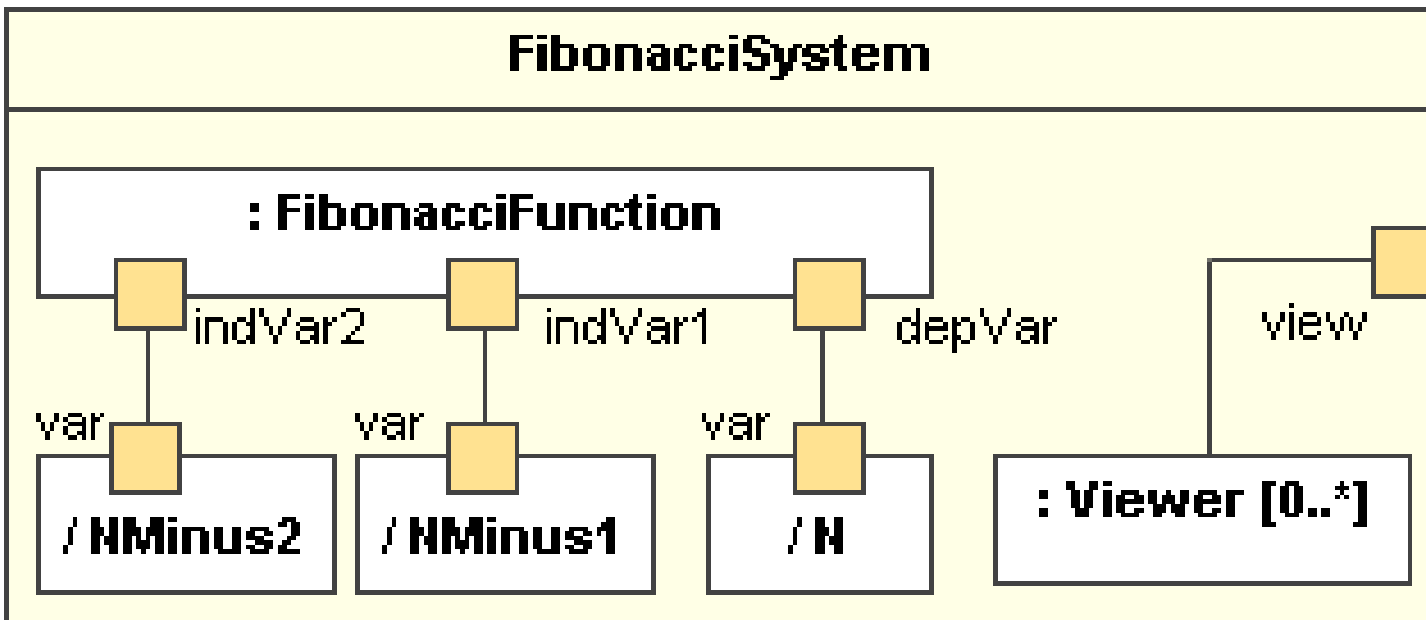
UML2.0 – Diagrame de Structură 2

- ▶ **Diagramă de Componente:** componentele sistemului și legăturile între componente



UML2.0 – Diagrame de Structură 3

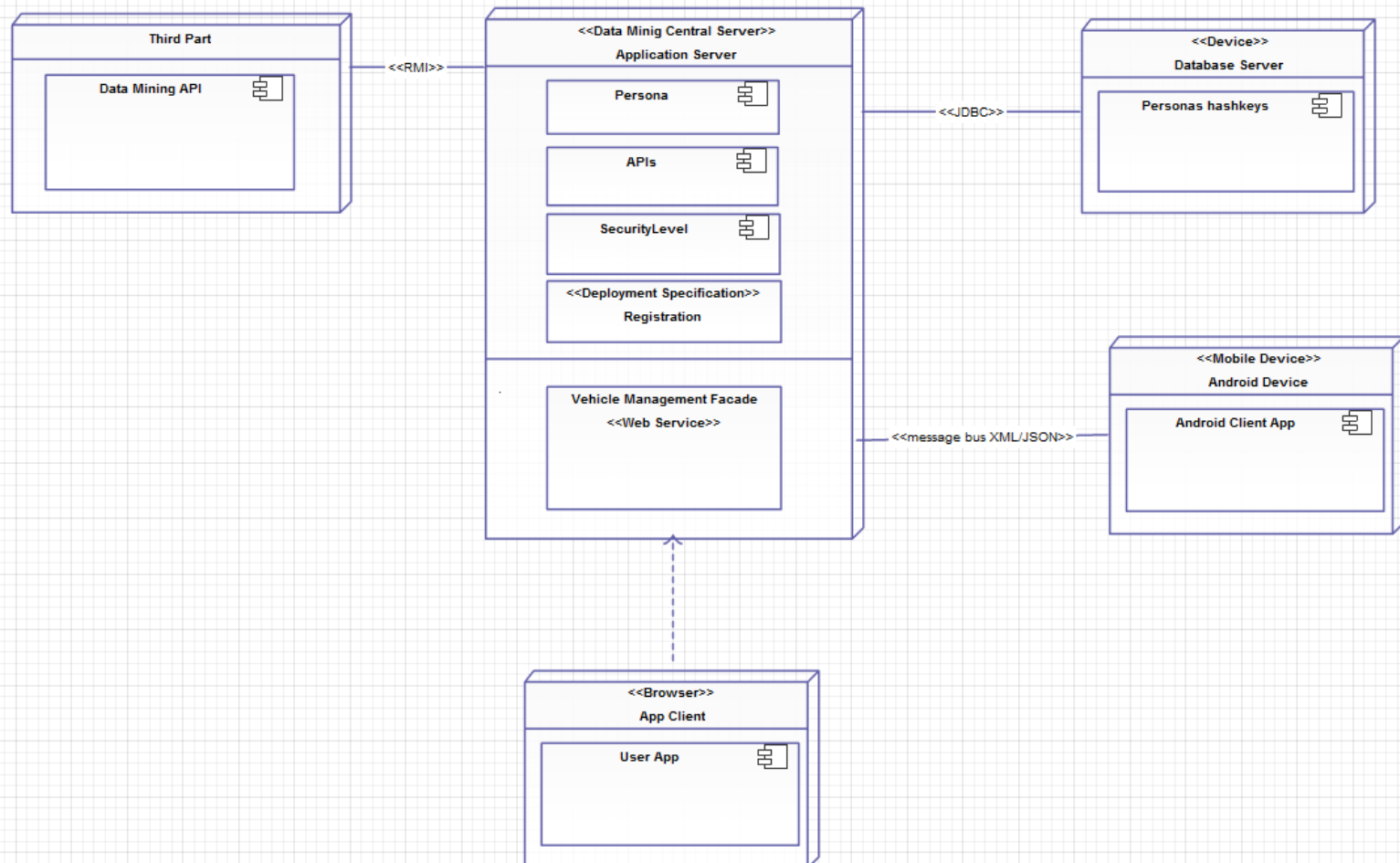
- ▶ Diagrame structură composită: structura internă



UML2.0 – Diagrame de Structură 4

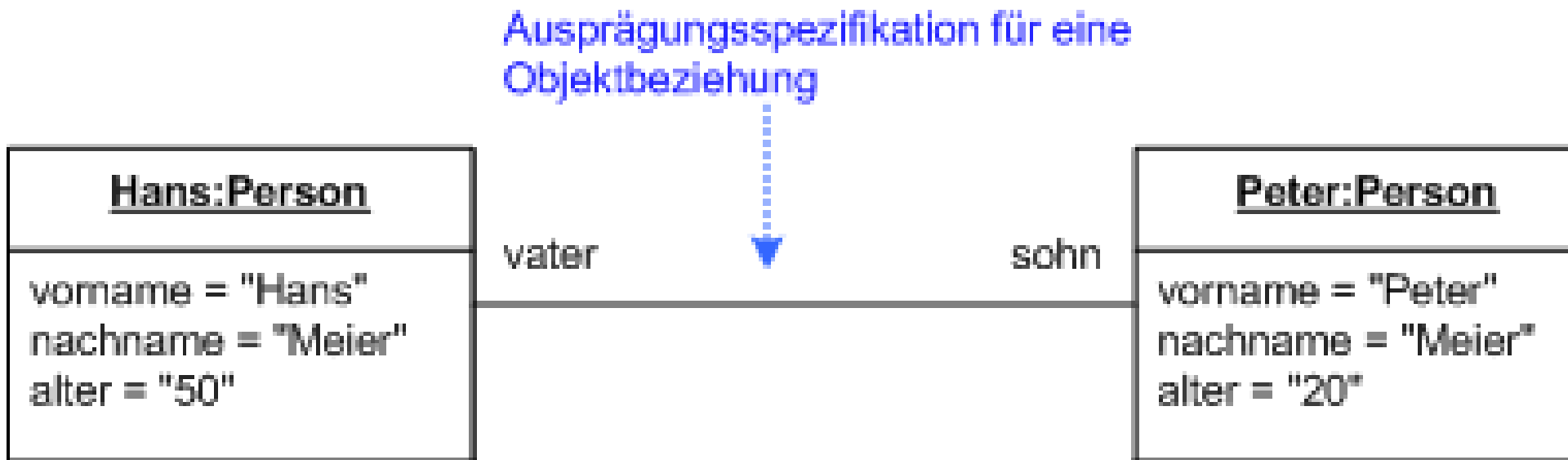
- ▶ Diagramă de Deployment: modelarea structurii hardware

Deployment Diagram For Web DataMining System



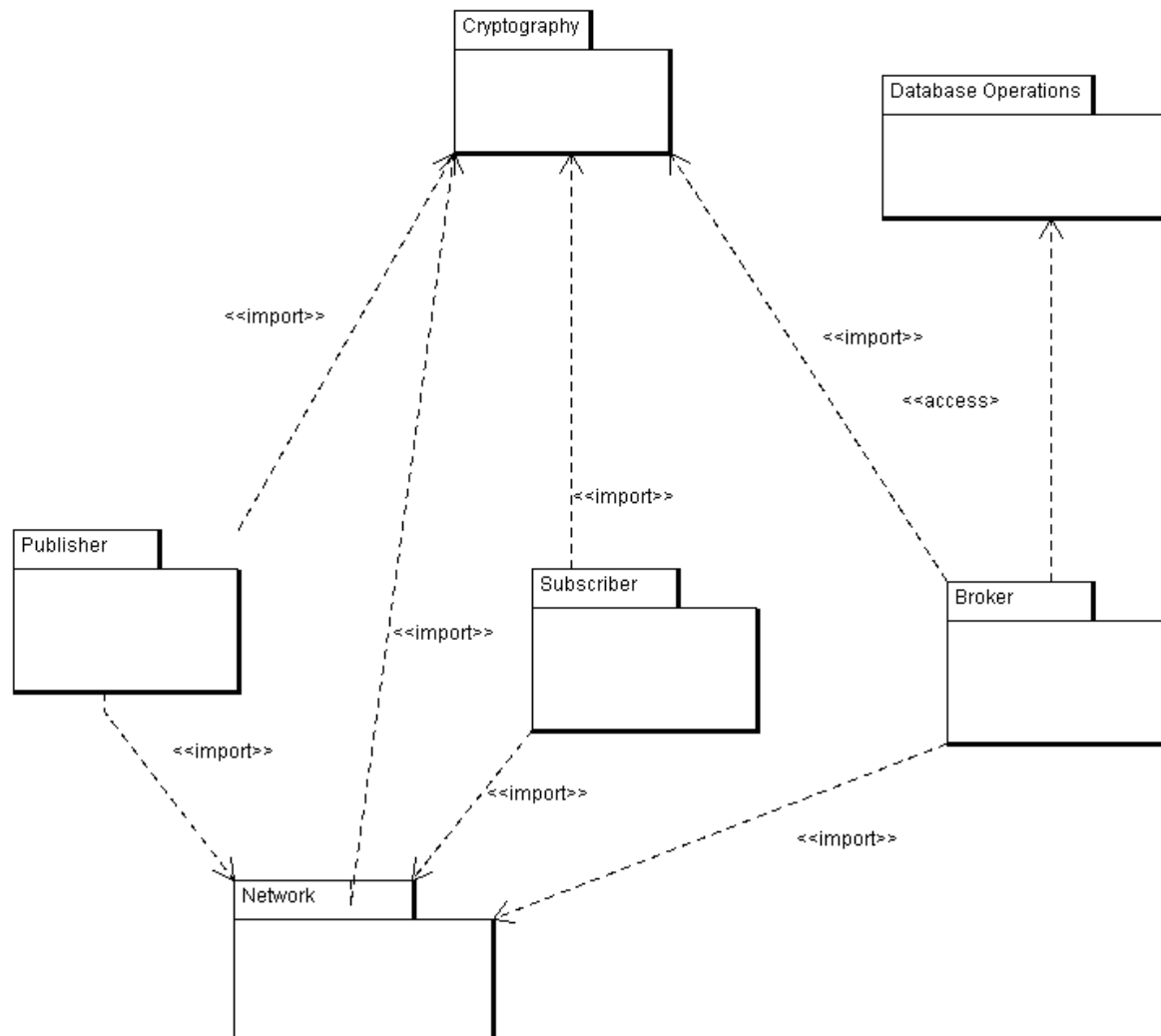
UML2.0 – Diagramme de Structură 5

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



UML2.0 – Diagrame de Structură 6

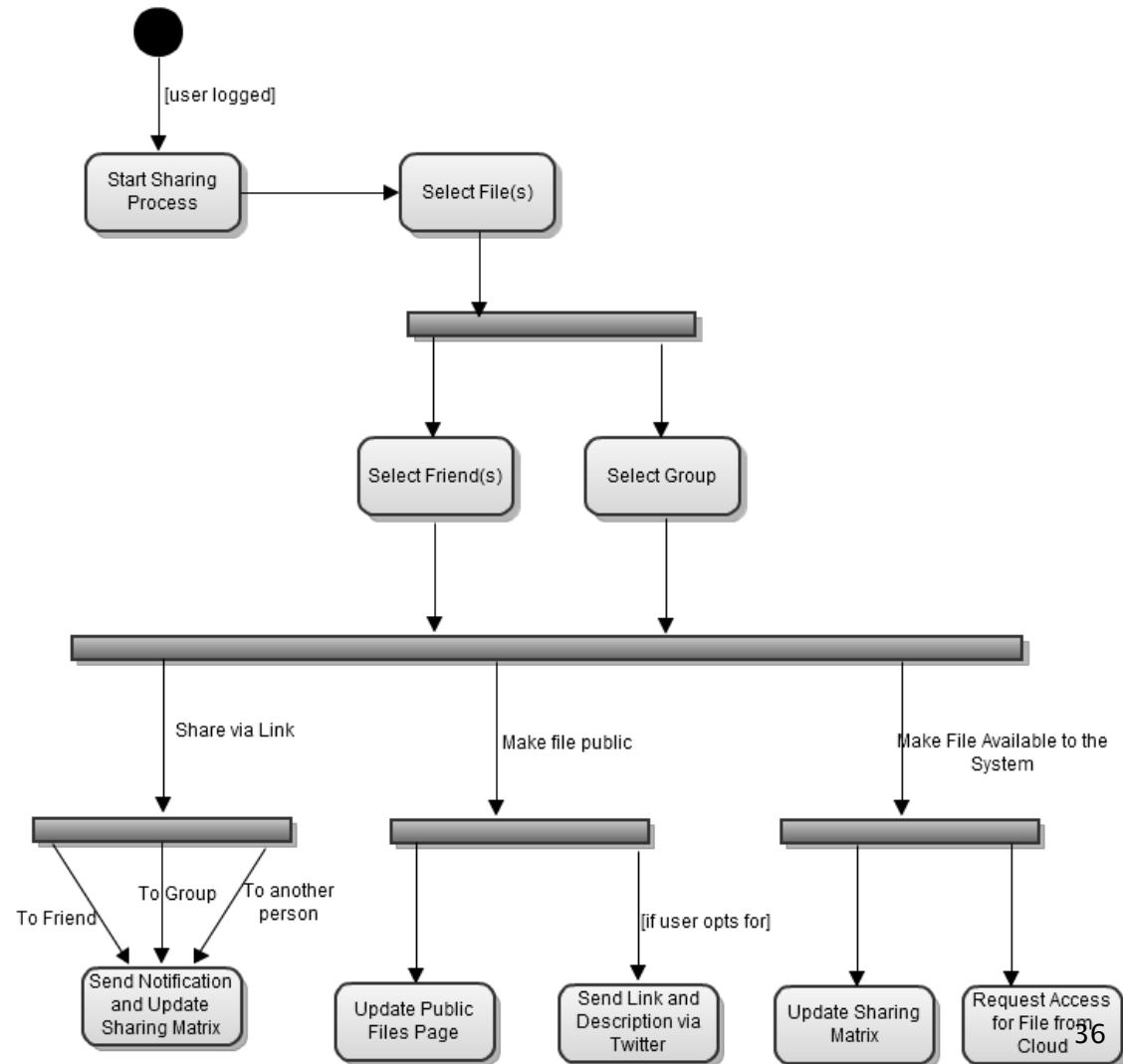
- ▶ Diagramă de pachete: împărțirea sistemului în pachete și relațiile din



UML 2.0 – Diagrame Comportamentale 1

- ▶ **Diagrame de activitate:** prezentare business și a fluxului de activități

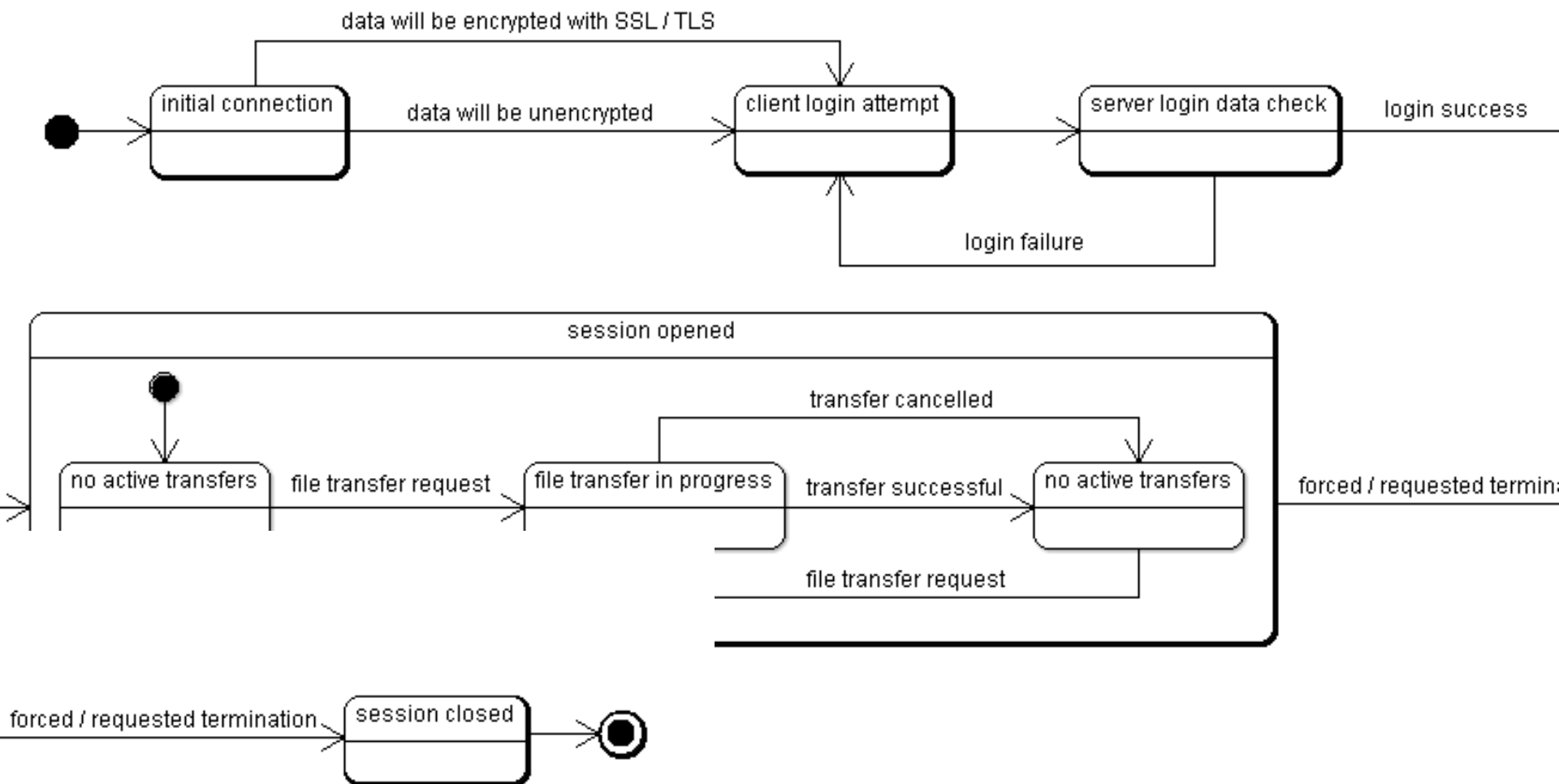
File Sharing: Activity Diagram



UML 2.0 – Diagrame Comportamentale 2

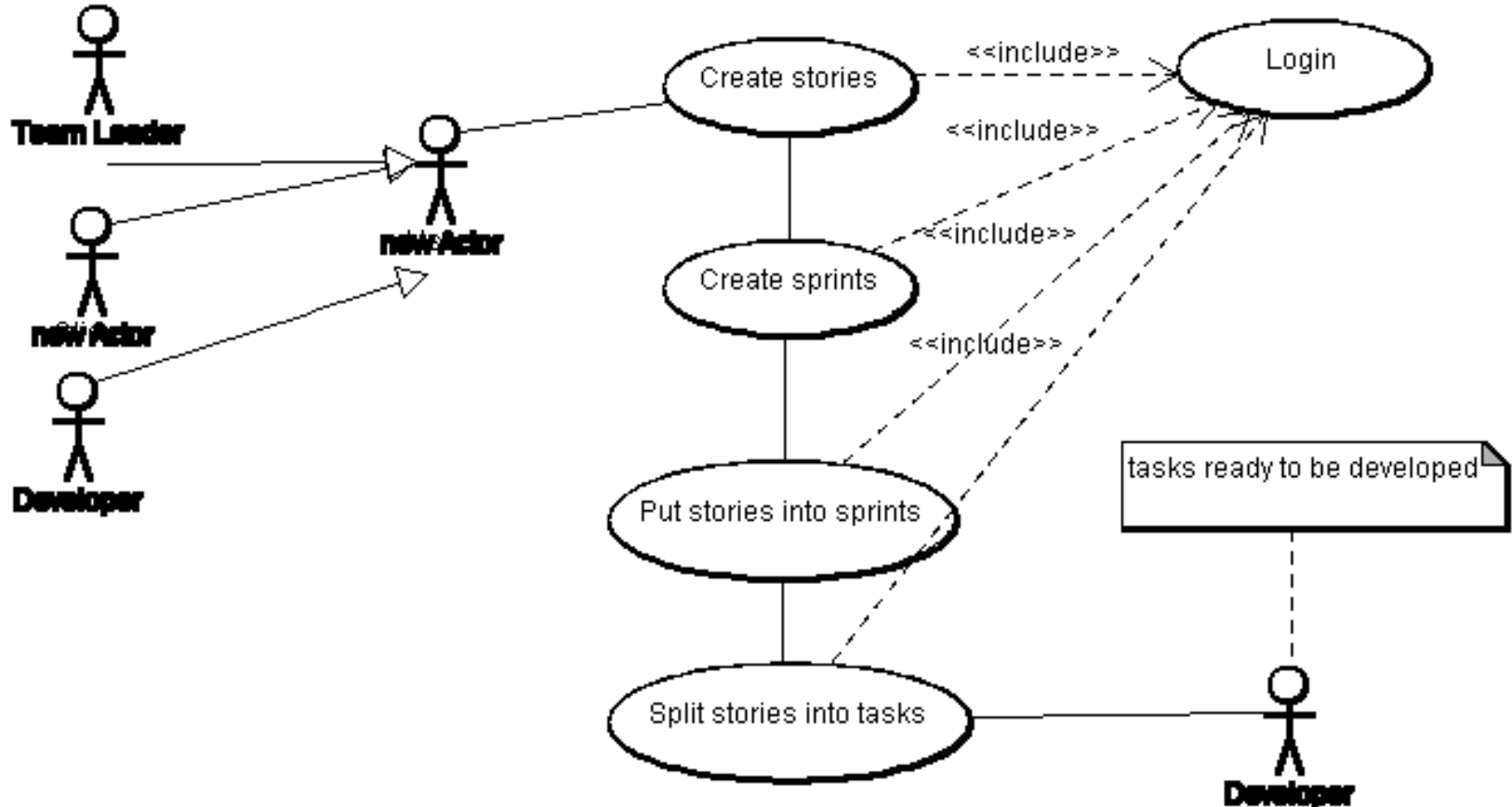
- ▶ **Diagrame de stare:** pentru a prezenta stările obiectelor

State diagram for the FTP protocol



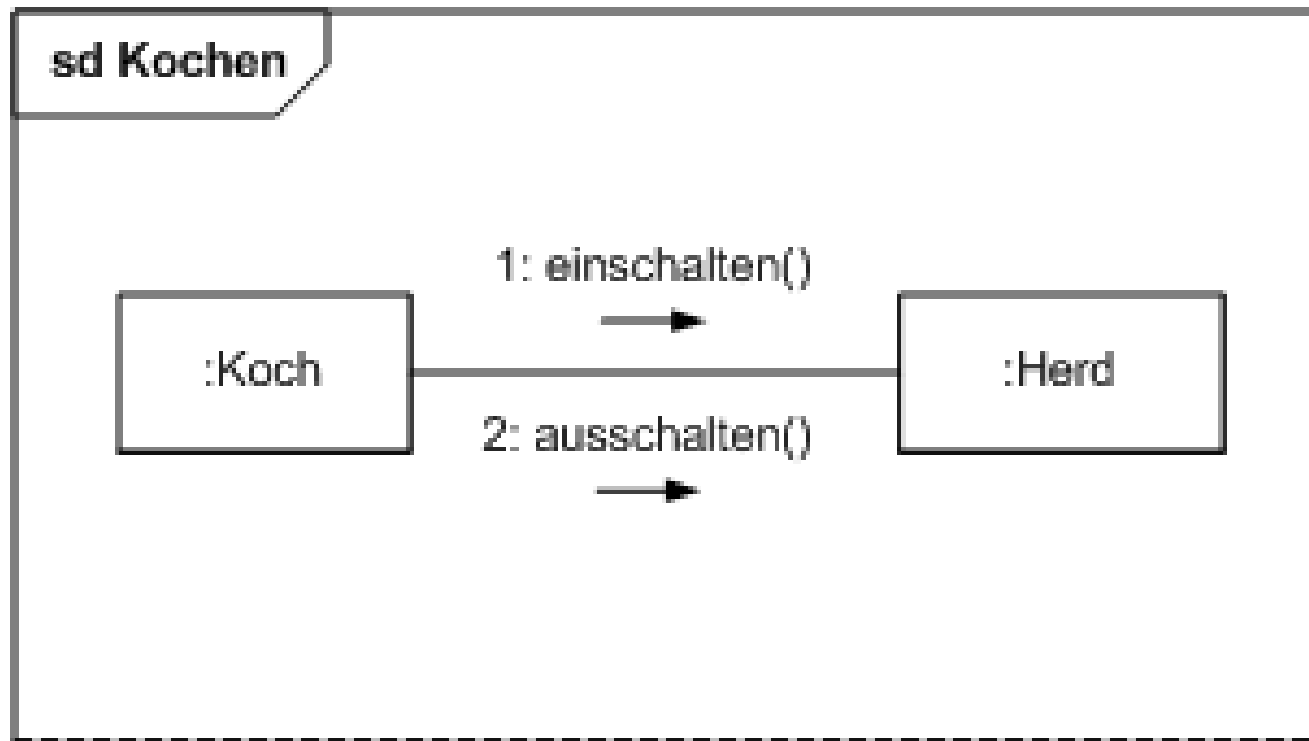
UML 2.0 – Diagrame Comportamentale 3

- ▶ **Diagrame Use Case:** prezintă funcționalitățile sistemului folosind actori, use case-uri și dependențe între ele



UML 2.0 – Diagrame de interacțiuni 1

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



UML 2.0 – Diagrame de interacțiuni 2

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

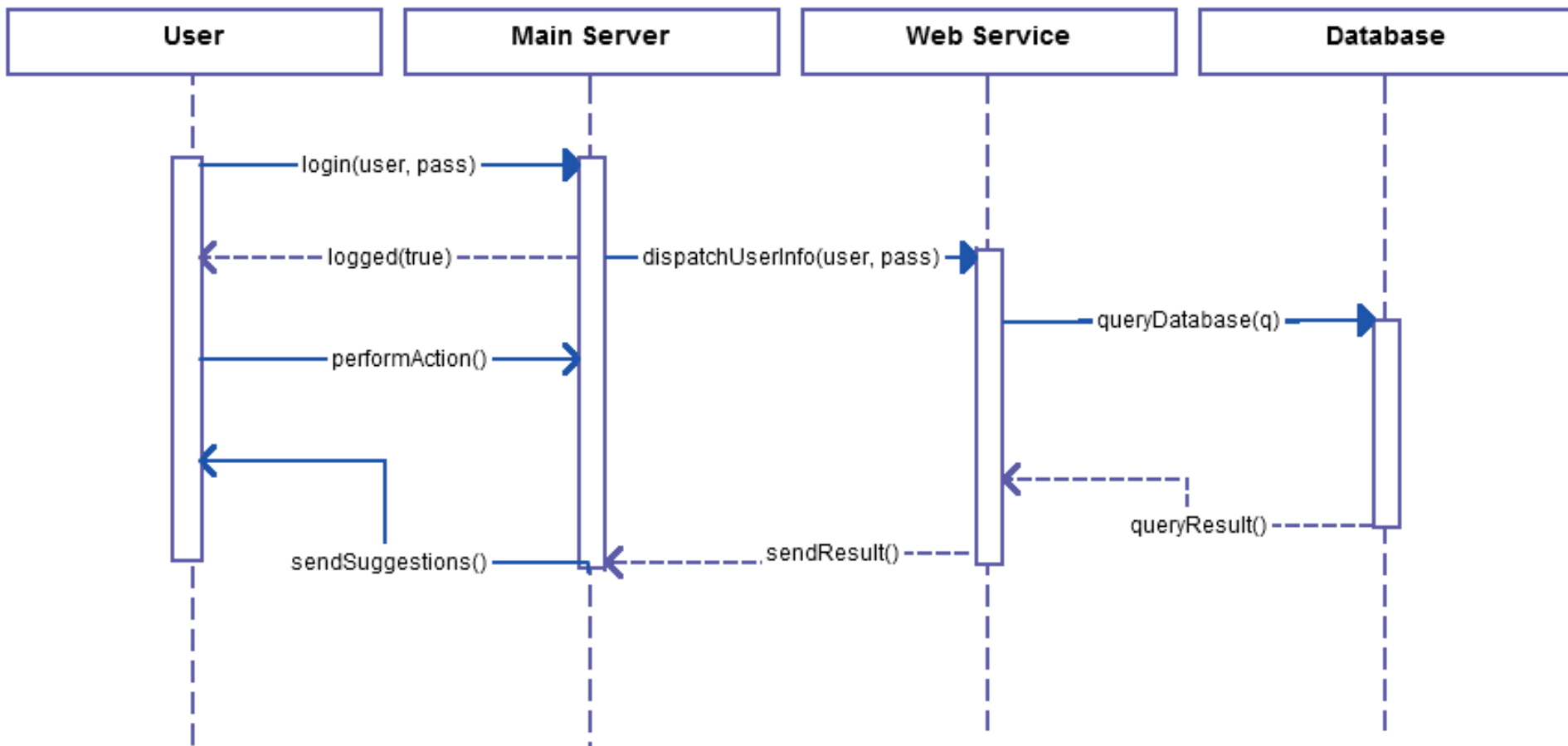


Diagrama de clase – Class Diagram

► Scop:

- Modelează vocabularul sistemului ce trebuie dezvoltat
- Surprinde conexiunile semantice sau interacțiunile care se stabilesc între elementele componente
- Folosită pentru a modela structura unui program

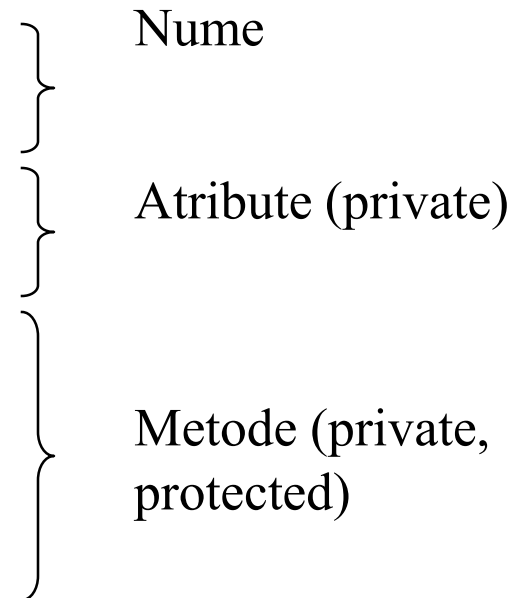
► Conține

- Clase/Interfețe
- Obiecte
- Relații (Asocierie, Agregare, Generalizare, Dependență)

Clase

- ▶ Modelează vocabularul = identifică conceptele pe care clientul sau programatorul le folosește pentru a descrie soluția problemei
- ▶ Elementele unei clase:
 - Nume: identica o clasa
 - Atribute: proprietăți ale clasei
 - Metode: implementarea unui serviciu care poate cerut oricărei instanțe a clasei

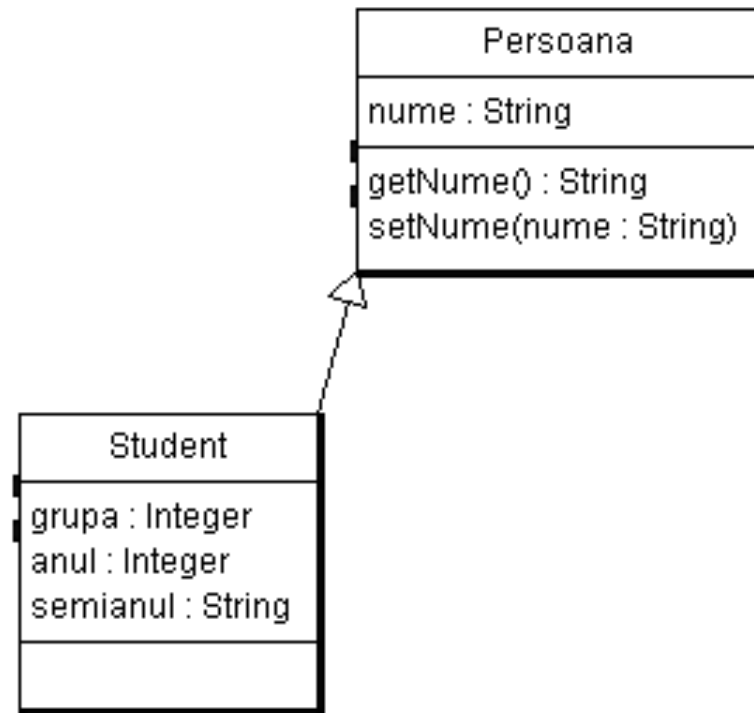
1)



Relații – Generalizare – C#

- ▶ Modelează conceptul de moștenire între clase
- ▶ Mai poartă denumirea de relație de tip *is a* (este un/este o)

ArgoUML – Relația de generalizare



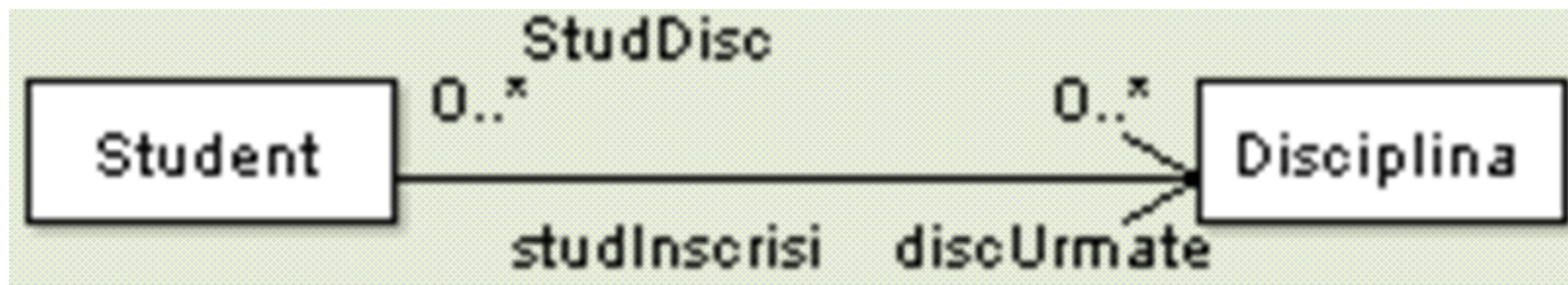
Relații – Asociere

- ▶ Exprimă o conexiune semantică sau o interacțiune între obiecte aparținând diferitelor clase
- ▶ Pe măsura ce sistemul evoluează noi legături între obiecte pot fi create, sau legături existente pot fi distruse
- ▶ O asociere interacționează cu obiectele sale prin intermediul capetelor de asociere
- ▶ Elemente:
 - Nume: descrie relația
 - Capete de asociere
 - Nume = rolul jucat de obiect în relație
 - Multiplicitate = câte instanțe ale unei clase corespund unei singure instanțe ale celeilalte clase



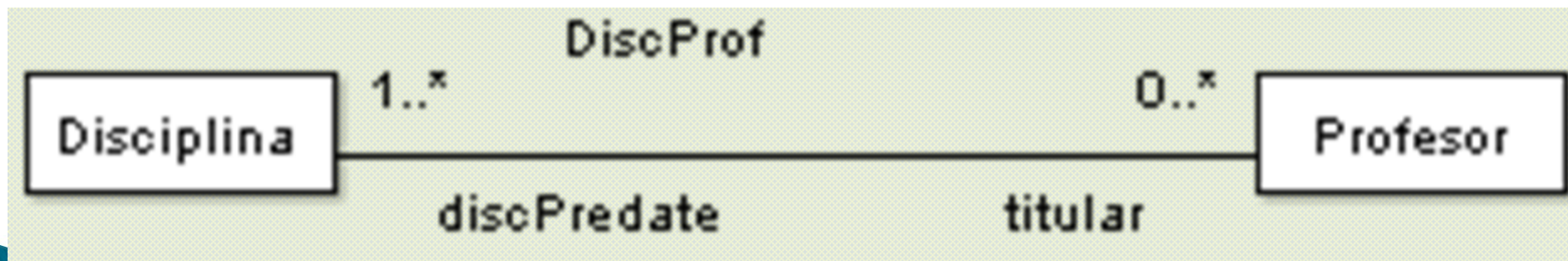
Relația de asociere 1

- ▶ Relația Student – Disciplină
 - **Student**: urmez 0 sau mai multe discipline, cunosc disciplinele pe care le urmez;
 - **Disciplină**: pot fi urmată de mai mulți studenți, nu cunosc studenții care mă urmează



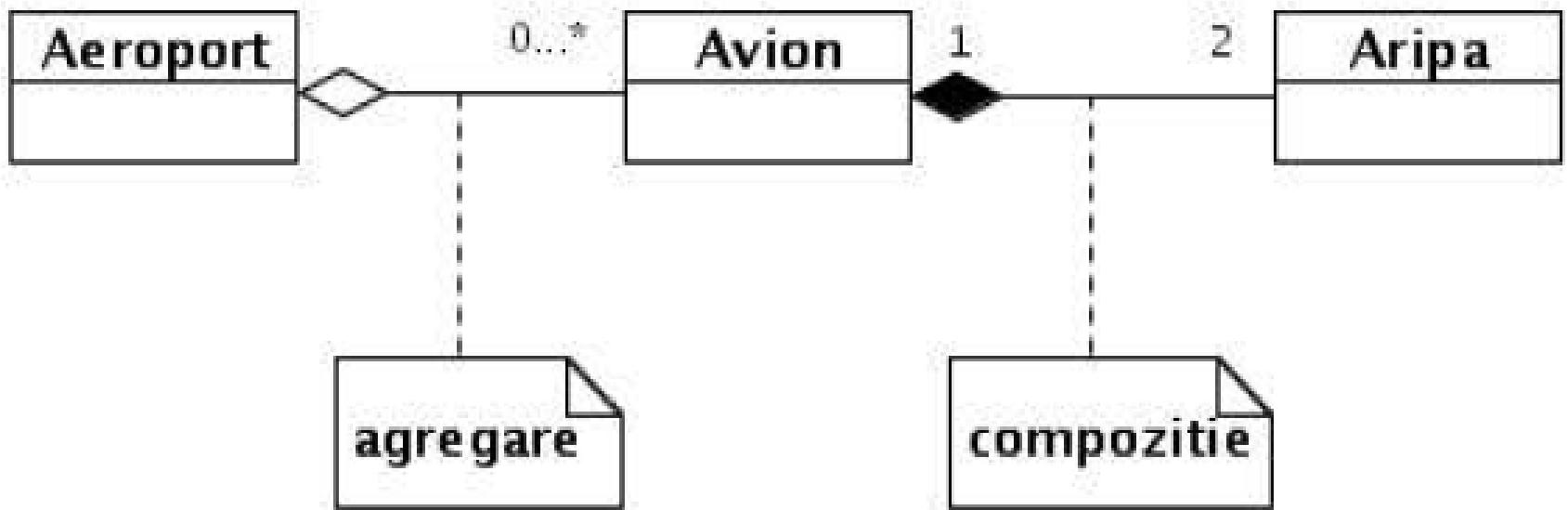
Relația de asociere 2

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

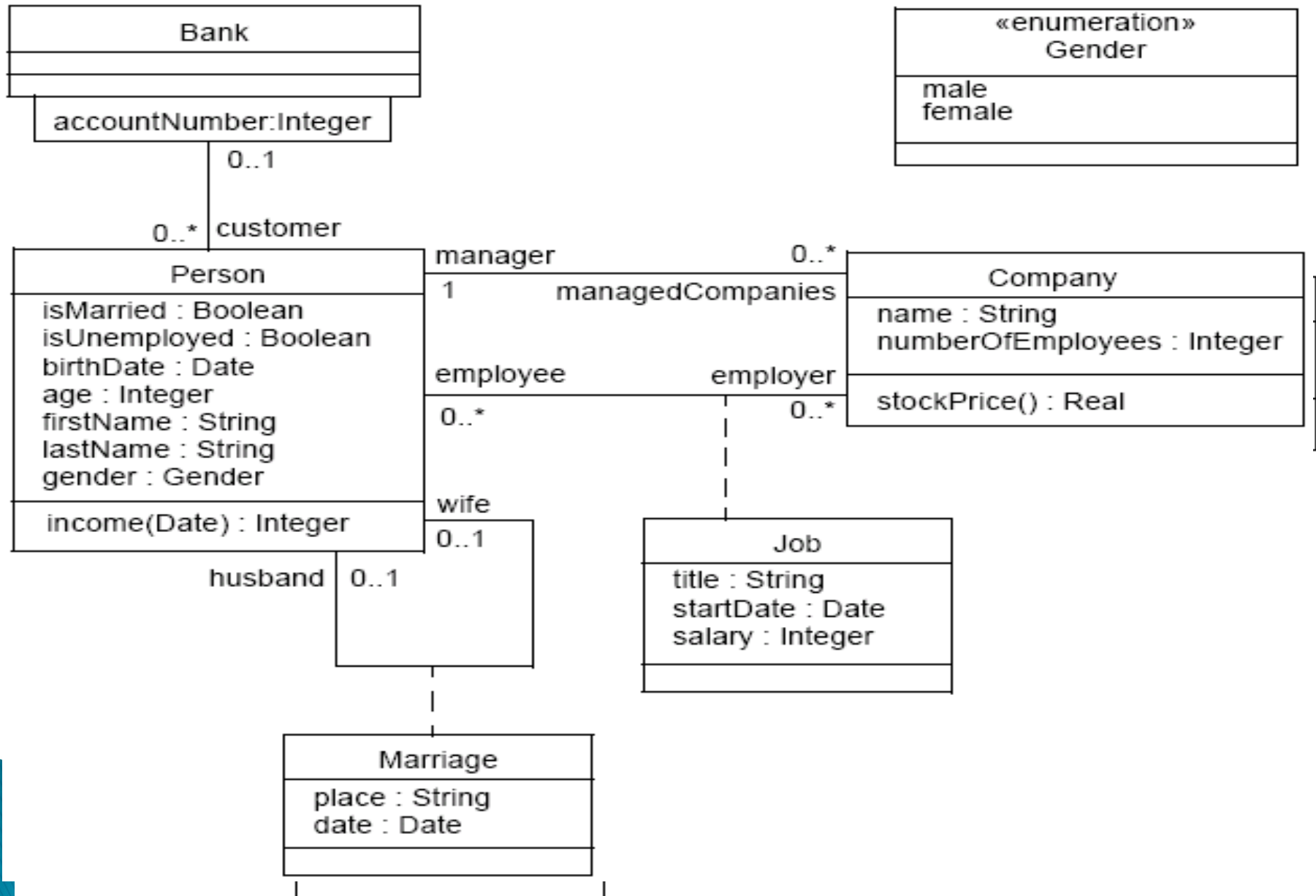


Relații – Agregare

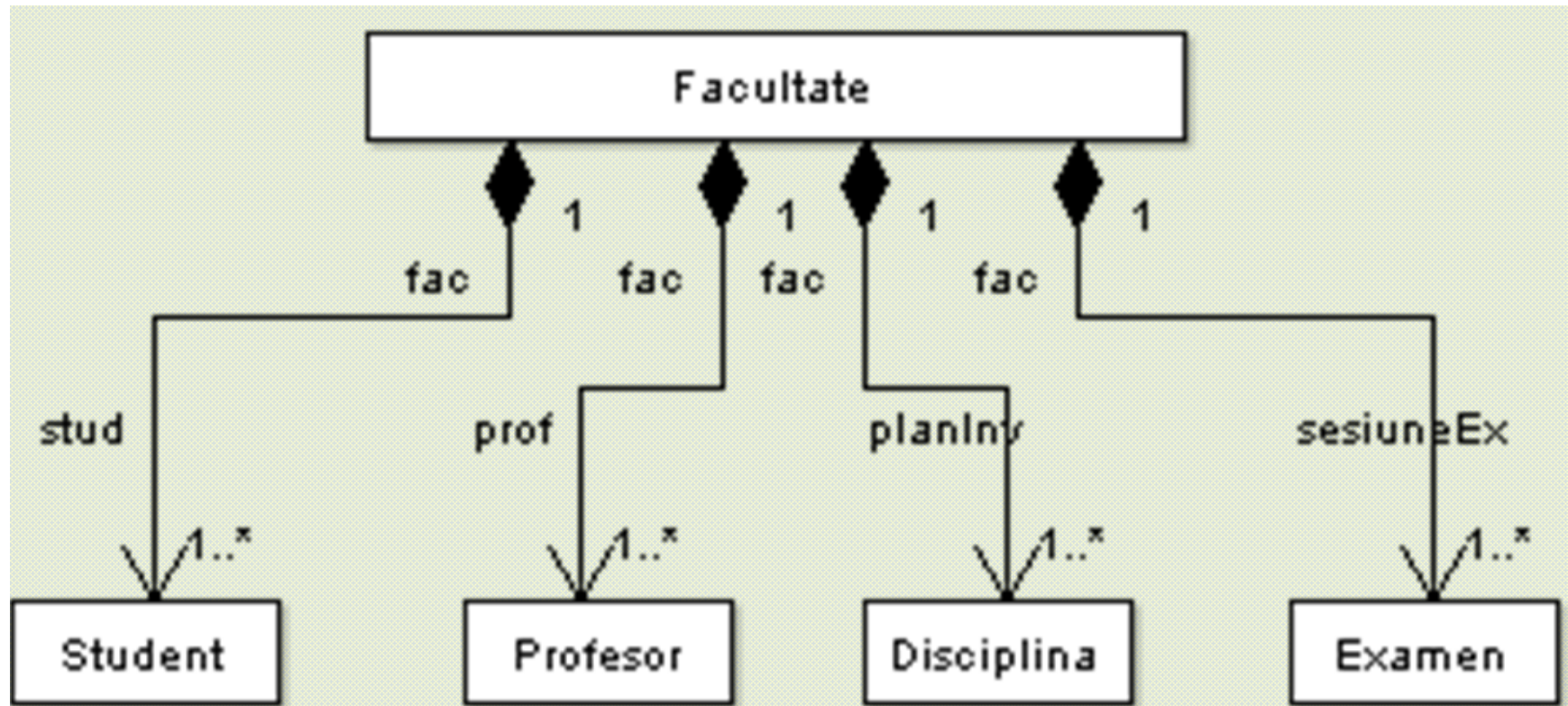
- ▶ Este un caz particular al relației de asociere
- ▶ Modelează o relație de tip parte-întreg
- ▶ Poate avea toate elementele unei relații de asociere, însă în general se specifică numai multiplicitatea
- ▶ Se folosește pentru a modela situațiile între care un obiect este format din mai multe componente.



Exemplul



Relația de Compoziție (“hasA”)



Studiu de Caz

- ▶ Obținerea Studenților Bursieri
 - Actori
 - Scenarii de utilizare
 - Clase

Concluzii

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

Bibliografie

- ▶ **OMG Unified Modeling Language™ (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 Gheorghieș, 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