M. Caramihai, © 2020

PROGRAMAREA ORIENTATA OBIECT

CURS 4

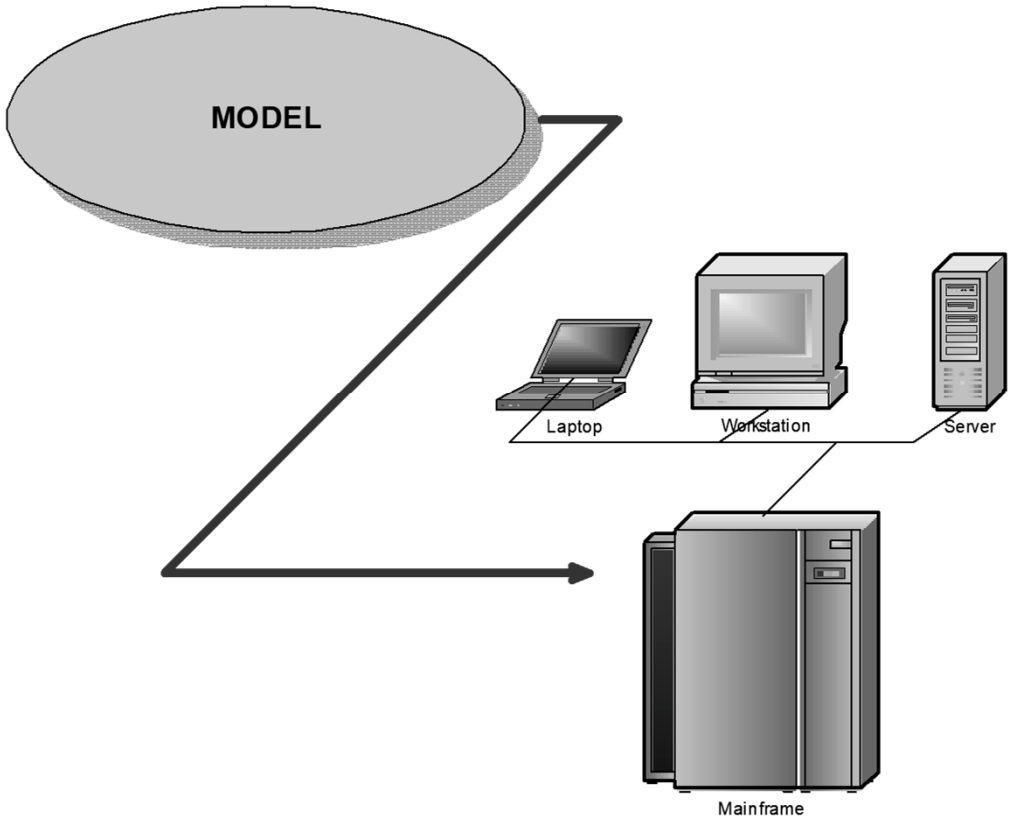
Introducere in UML (1)

Ce este modelarea ?

- Aspecte generale privind modelarea
- Trasaturile modelarii:
  - Simplificare (rezultatul abstractizarii)
  - Subordonare unui scop
  - Reprezentarea unei realitati
  - Divizare
  - Ierarhizare
  - Comunicare
- Grupuri tinta: client/utilizatori si membrii echipa proiect

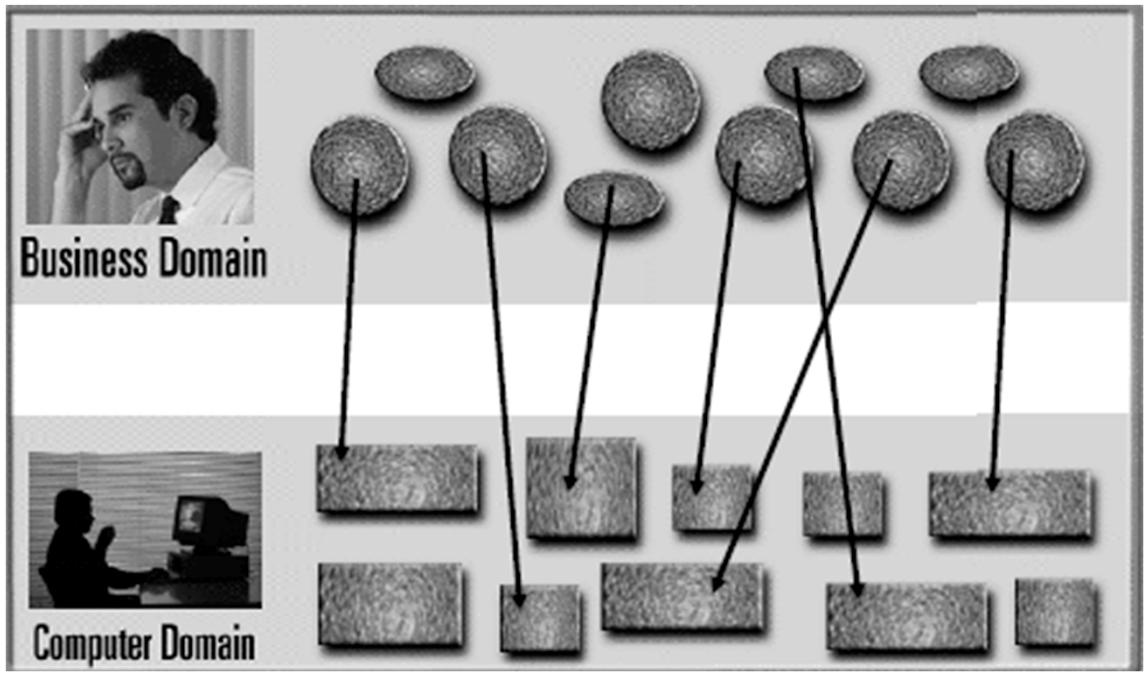
Ce este modelarea vizuala (1) ?

- Modelarea vizuala permite modelarea prin notatii grafice standard
- Principii:
  - Modul de creare tine de scopul utilizarii modelului
  - Orice model poate fi conceput la diferite niv de abstractizare
  - Orice sistem real poate fi reprezentat printr'o suita de modele



Ce este modelarea vizuala (2) ?

Prin modelarea vizuala se pot abstractiza
obiectele & logica proceselor de afaceri



Prin modelarea vizuala poata fi
analizata & proiectata o aplicatie (informatica)

Ce este UML ?

UML este prescurtarea de la **Unified Modeling Language**

- Definitie: limbaj de modelare pentru specificarea, vizuualizarea, constructia si documentarea componentelor unei aplicatii informatice
- UML reprezinta o sinteza a:
  - Conceptele *Data Modeling* (*Entity Relationship Diagrams*)
  - *Business Modeling* (*work flow*)
  - Modelarea obiectelor
  - Modelarea componentelor
- UML este limbajul standard pentru "visualizing, specifying, constructing, and documenting the artifacts of a software intensive system" [Booch].
- *Focus* pe elementele **conceptuale** si **fizice** ale reprezentarii unui sistem.

Istoria dezvoltarii UML

Conceptele UML

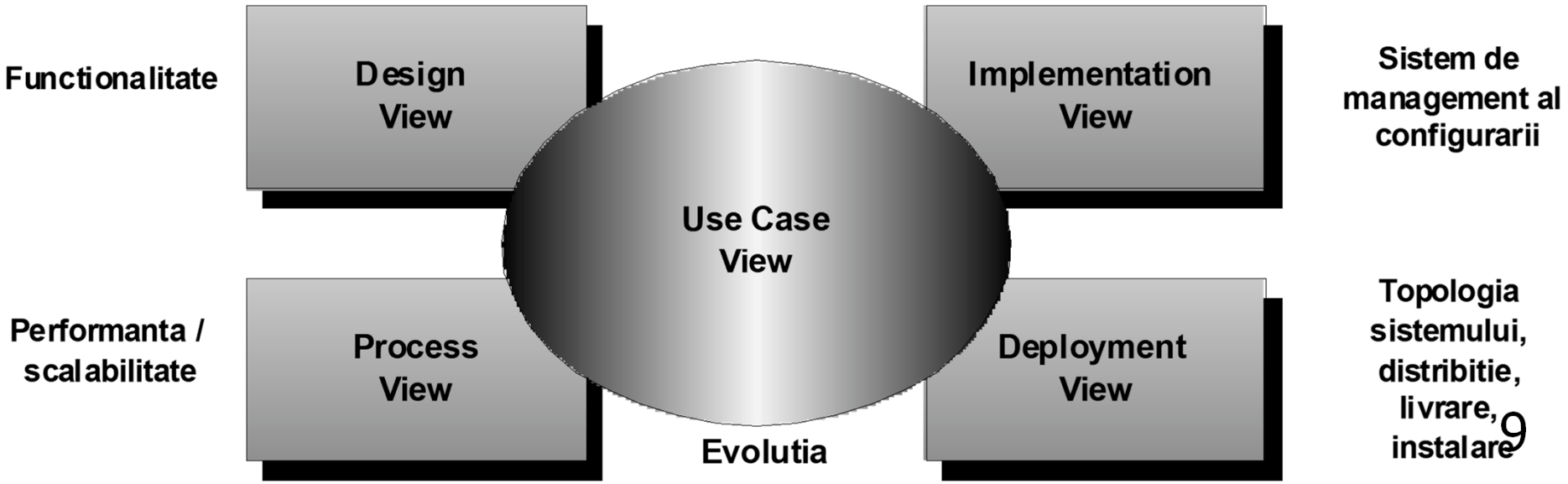
UML poate fi utilizat pentru:

- Descrierea limitelor unui sistem si a functiilor sale principale (*use cases & actori*) → comportamentul functional al sistemului
- Ilustrarea implementarii cazurilor de utilizare prin *diagramele de interactiuni*
- Reprezentarea structurii statice a unui sistem prin *diagrama de clase* (utilizarea claselor+interfetelor pt model. entitatilor statice din sistem) si diagrama de obiecte (arata instantele claselor + legaturi)
- Modelarea evolutiei obiectelor prin *diagrama de tranzitii*
- Structurarea implementarii fizice a arhitecturii sistemului cu *diagramele de componente si operationale* → arata org. elementelor din sistem
- *Diagrama de pachete:* tip special de diagrame de clase; focus pe gruparea claselor
- *Diagrama de activitati:* descr flux de evolutie ale activitatilor
- *Diagrama d secvente:* surprinde tipul si ordinea mesajelor

Arhitectura UML (1)

Arhitectura = setul de decizii privind:

- Organizarea sistemului *software*.
- Selectia elementelor structurale & interfete din care un sistem este format.
- Evolutia & colaborarea elementelor.
- Structura si evolutia elementelor.
- Arhitectura sistemului.



Arhitectura UML (2)

Use Case View

- Analiza Cazurilor de utilizare (CU) este o tehnica de "captura" a proceselor din perspectiva utilizatorului.
- Include evolutia "vazuta" de utilizatori, analisti si testatori.
- Specifica "fortele" ce influenteaza arhitectura.
- Aspectele statice sunt reprezentate in diagramele CU.
- Aspectele dinamice sunt reprezentate in diagramele de interactiuni, de activitati si de stari

Design View

- Cuprinde clasele, interfetele si colaborarile ce definesc "vocabularul" unui sistem.
- Definesc necesitatile functionale ale sistemului.
- Aspectele statice sunt reprezentate in diagrama de clase si diagrama de obiecte

Arhitectura UML (3)

Process View

- Defineste cozile si procesele concurentiale si de sincronizare.
- Reprezinta performanta si scalabilitatea.
- Aspecte statice si dinamice reprezentate la fel ca in *Design View*.

Implementation View

- Defineste componentele si fisierele utilizate in realizarea unui sistem fizic.
- Reprezinta managementul configuratiei.
- Aspectele statice sunt reprezentate in diagrama de componente.
- Aspectele dinamice sunt redate in diagramele de interactiuni, de activitati si de stari.

Deployment View

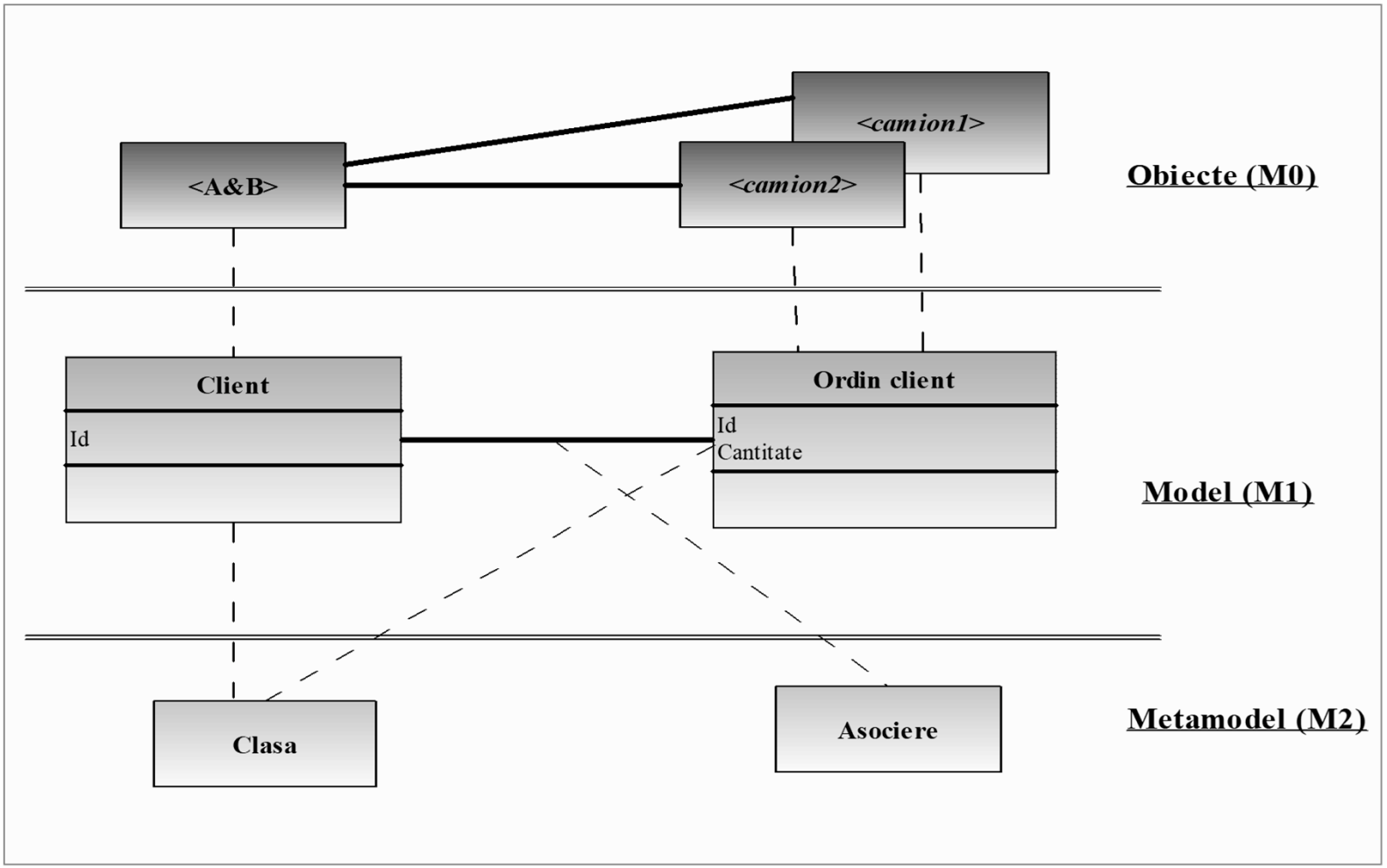
- Defineste nodurile ce formeaza topologia hardware.
- Reprezinta distributia si instalarea sistemului.
- Aspectele statice sunt reprezentate in diagrama de distributii.
- Aspectele dinamice sunt redate in diagramele de interactiuni, de activitati si de stari.

Arhitectura UML (4)

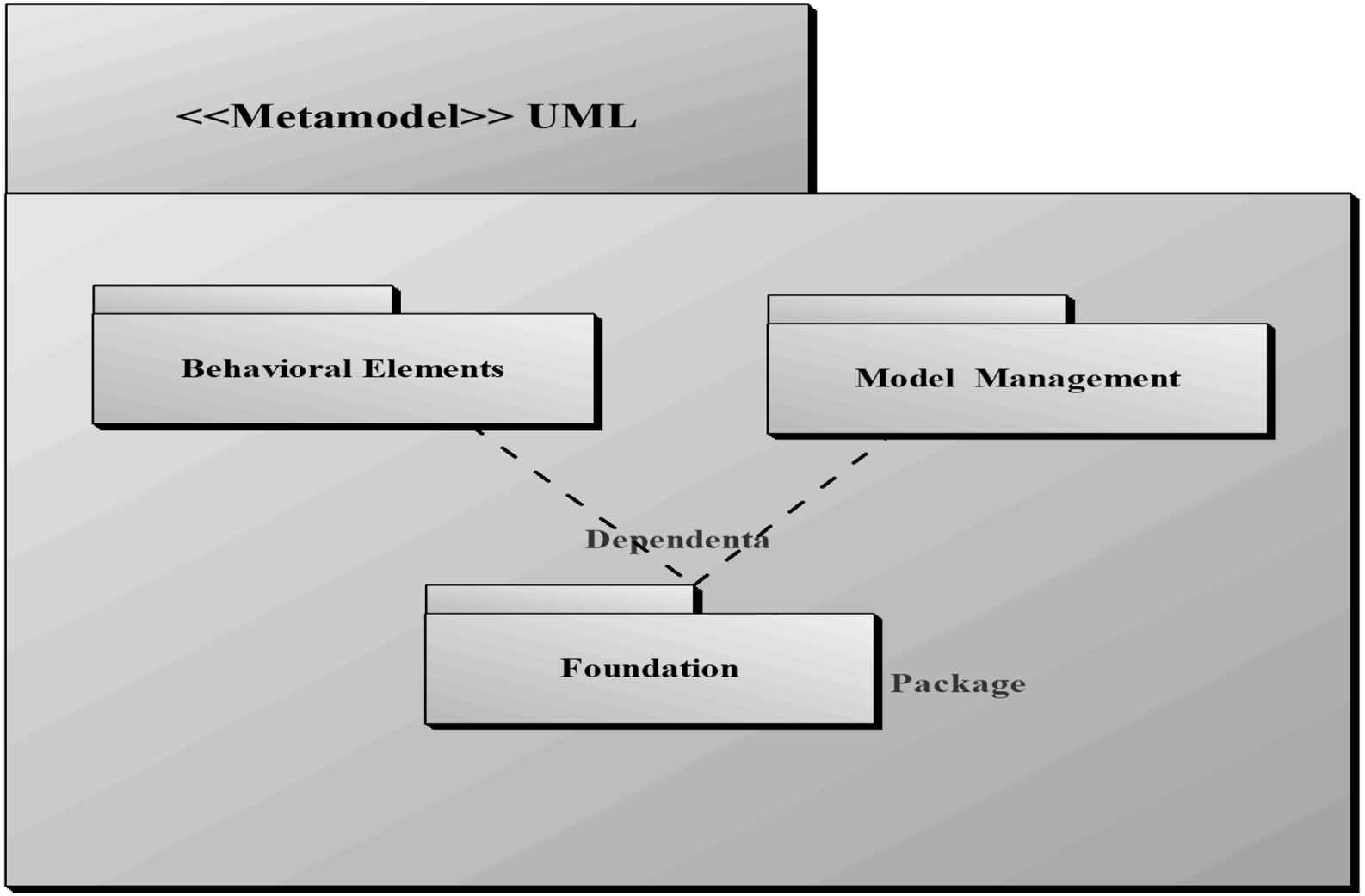
Arhitectura structurata

- Meta-metamodel: limbaj specific metamodelului
- Metamodel: limbaj specific modelului
- Model: limbaj specific domeniului
- Obiecte: instante ale unui model

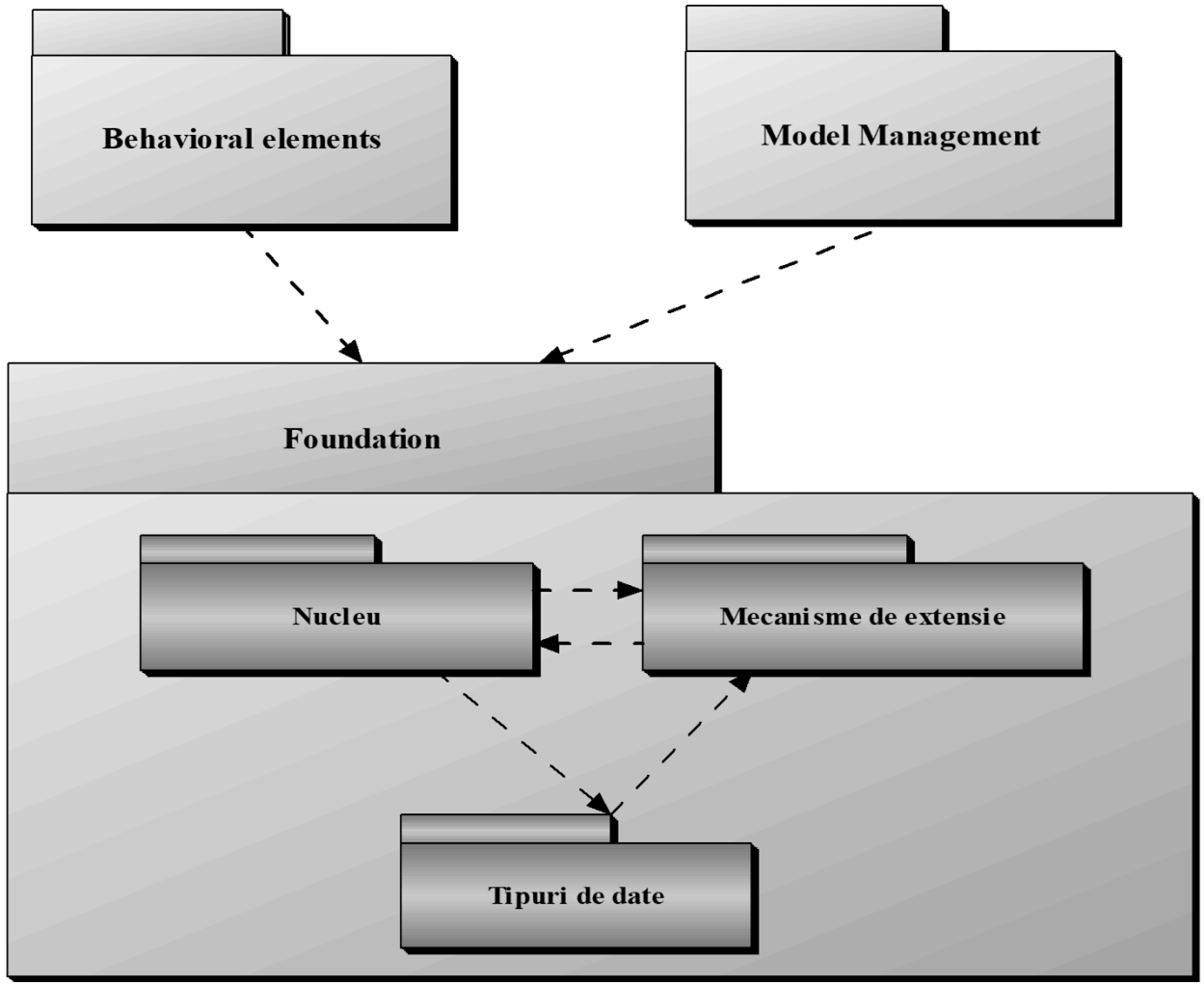
Metamodelul UML (1)



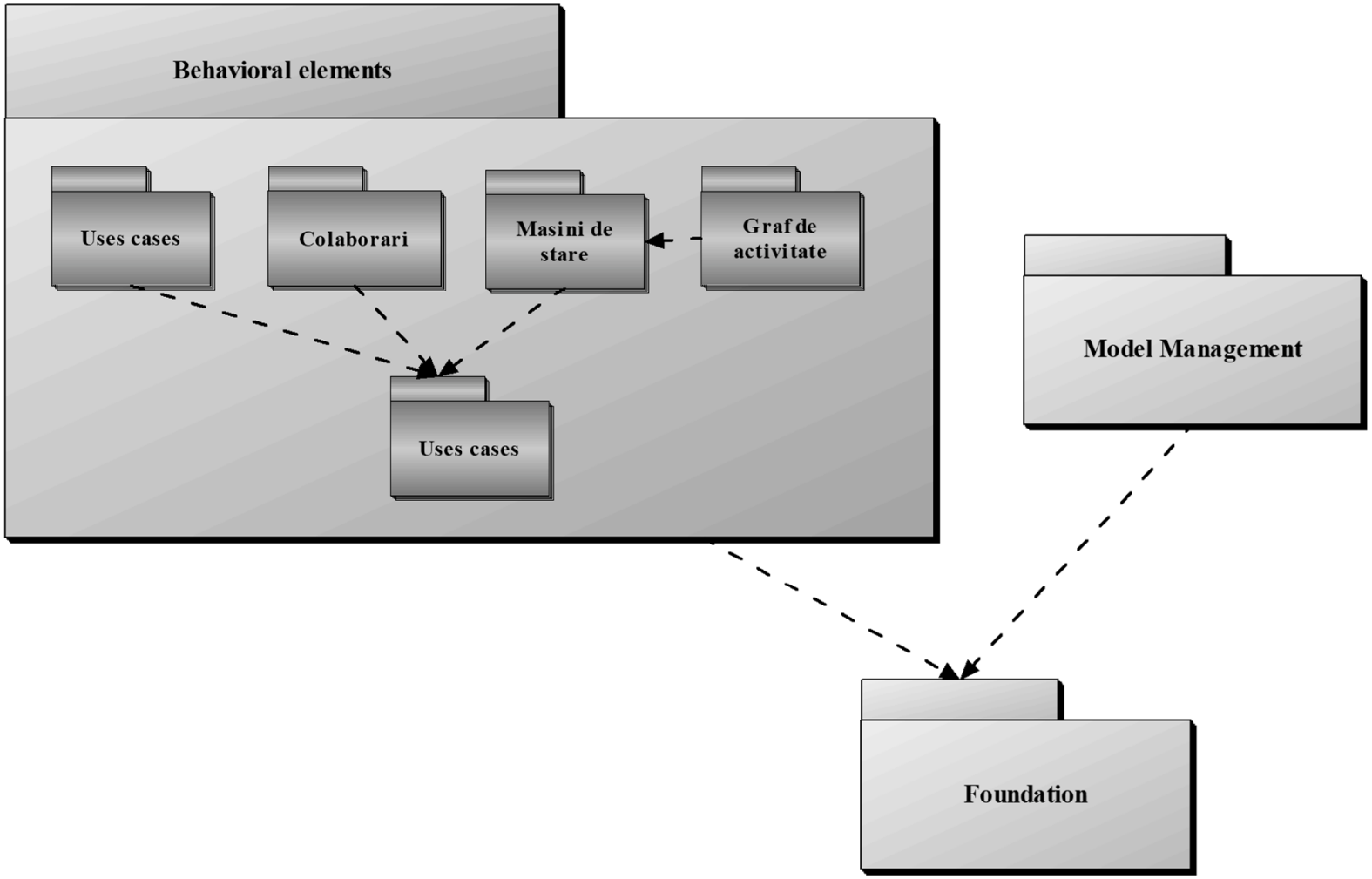
Metamodelul UML (2)



Metamodelul UML (3)



Metamodelul UML (4)



Elemente constructive in UML

Lucruri

- Concept de modelare (elementul de individualitate).

Relatii

- Leaga elementele individuale (I.e. conceptele)

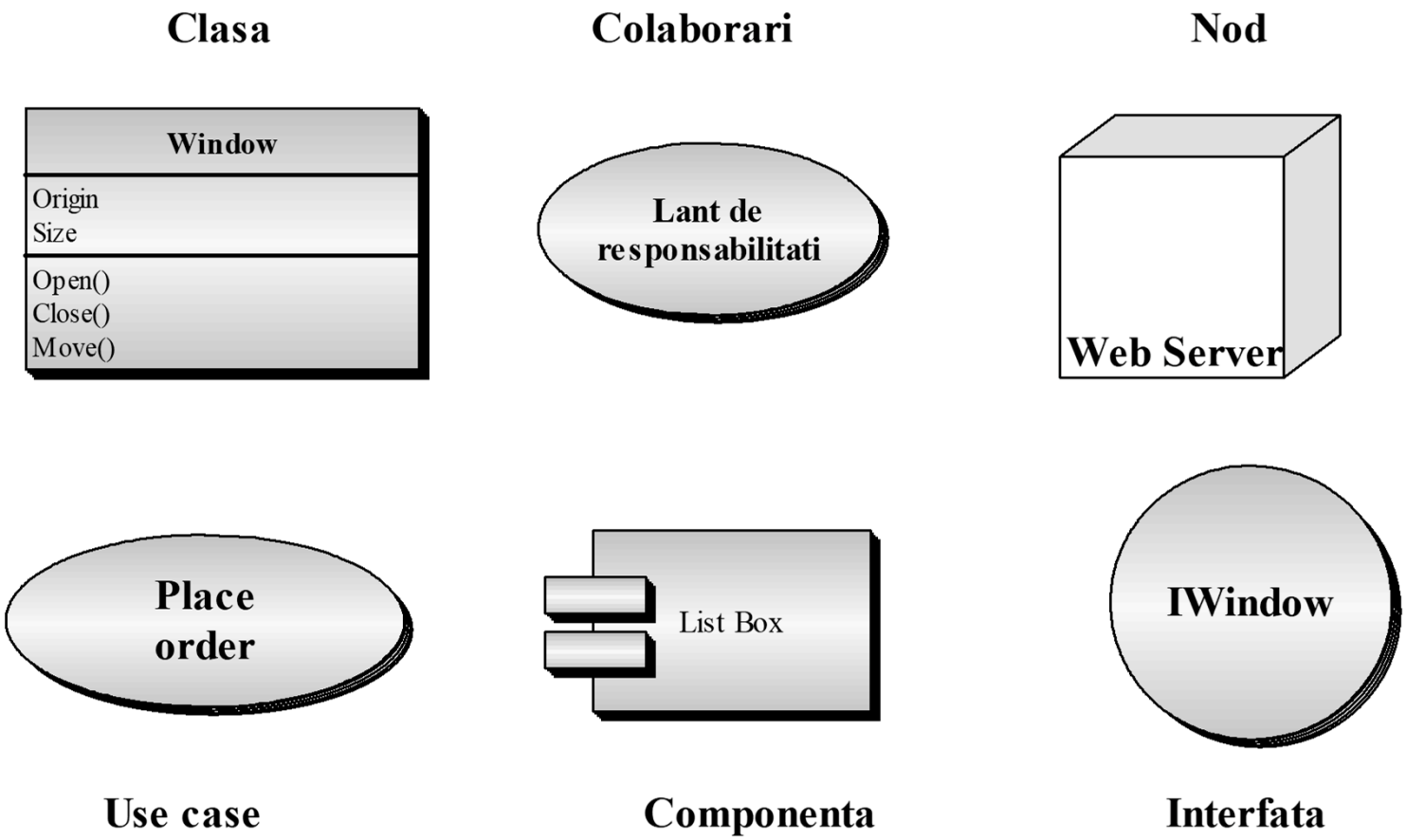
Diagrame

- Grupeaza colectiile organizate de elementele individuale (lucruri / relatii)

Lucruri - componente

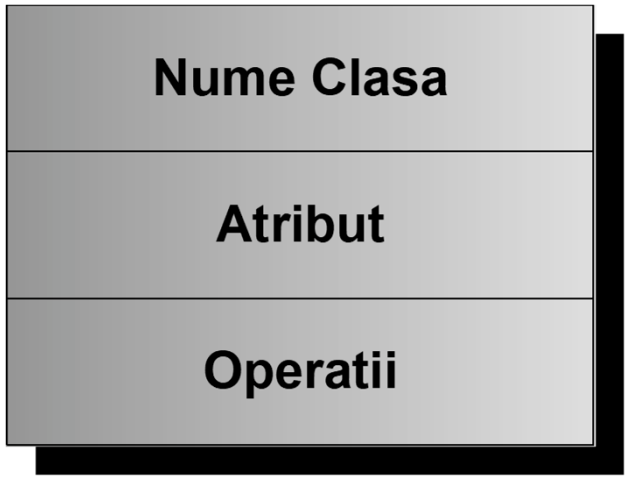
- **Structuri**: "substantivul" modelelor UML
- **Evolutii**: componenta dinamica a modelelor UML
- **Grupari**: componenta organizationala a modelelor UML
- **Adnotari**: componenta explicativa a modelelor UML

Structuri



- Elemente conceptuale sau fizice

Clase (1)

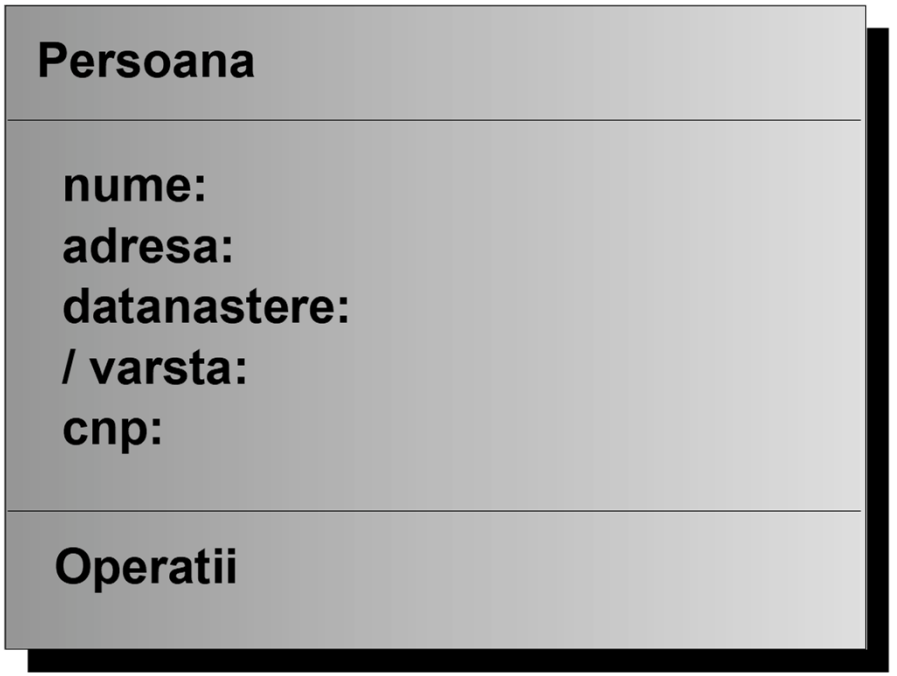


- Din punct de vedere grafic, o clasa este redata ca un dreptunghi;
- Include in mod uzual numele clasei, atribute si operatii (in compartimente separate).
- Numele clasei este obligatoriu
- Atributul este numele propietatii unei clase; forma:

attributeName: Type

Clase (2)

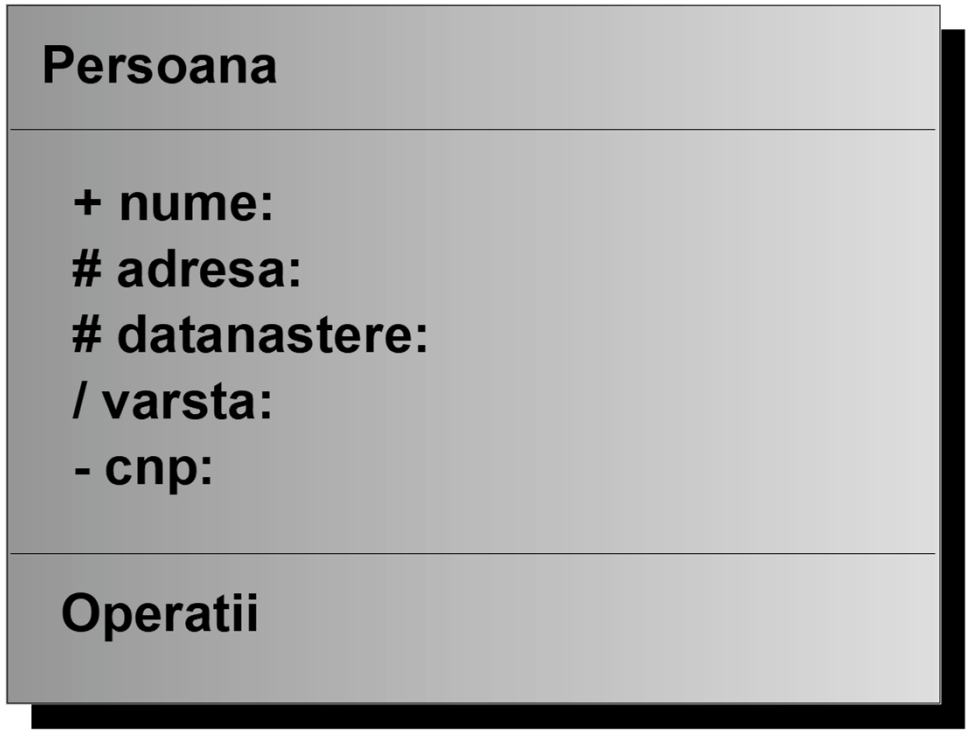
- Un atribut derivat poate fi calculat din alte atribute (el ne-existand in forma directa)
- d.e. varsta unei persoane poate fi calculata in raport de data nasterii si data curenta; reprezentare:



/ varsta: Date

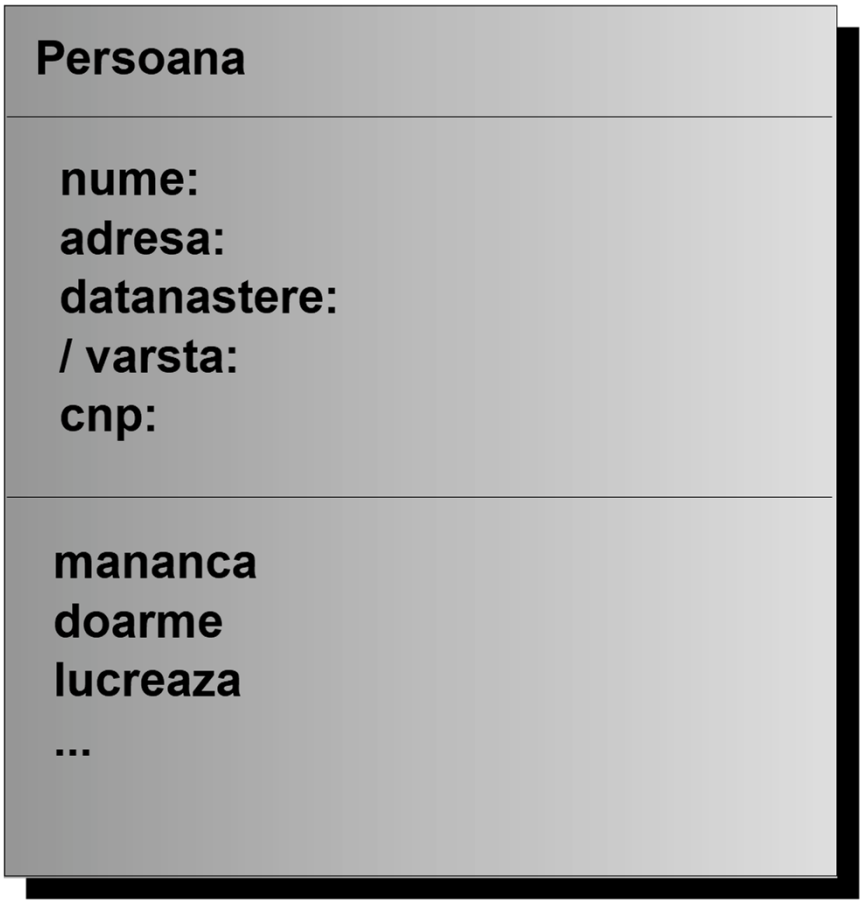
Clase (3)

- Atributele pot fi:
  - + public
  - # protected
  - - private
  - / derived



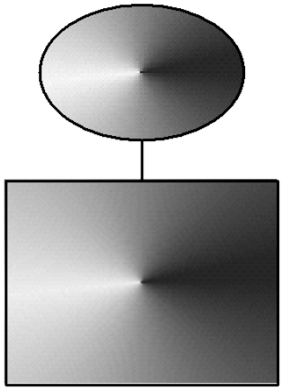
Clase (4)

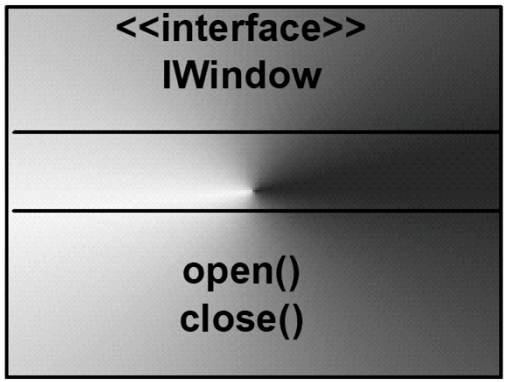
- **Operatiile**: descriu evolutia unei clase



Interfata

- Descrie un set de operatii ce specifica evolutia obiectelor fara a descrie structura lor interna.
- Se reprezinta cu stereotipul <<interface>> in fata numelui





- Interfata nu poate fi instantiata
- Nu au atribute sau stari
- Pot specifica serviciile oferite de o clasa asociata

Cazuri de utilizare (CU)

- "A *use case* specifies the behavior of a system or a part of a system, and is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor."

  -*The UML User Guide, [Booch,99]*

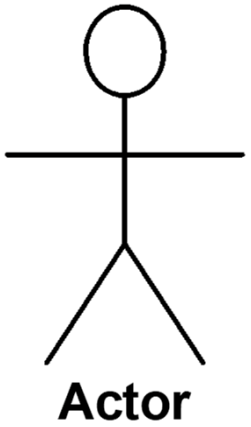
- "An *actor* is an idealization of an external person, process, or thing interacting with a system, subsystem, or class. An actor characterizes the interactions that outside users may have with the system."

  -The UML Reference Manual, [Rumbaugh,99]

Ce este un CU ?

Un **caz de utilizare** este:

- Un set de scenarii ce descrie evolutia unui sistem impreuna cu utilizatorii sai
  - Descrie functionalitatea pe care un sistem o pune la dispozitia utilizatorilor sai (oameni / sisteme) si a legaturilor dintre ei.
  - Defineste necesitatile clientilor
- **Actor**: oameni / sisteme ce se gasesc in afara sistemului (interactioneaza cu sistemul)
- **Scenariu**: dialog intre actor si sistem



Scopul CU

- Descrierea cerintelor functionale ale sistemului
- Structurarea unui cadru de referinte comun
- Formarea bazei de testare si verificare

→ **Totalitatea CU**: functionalitatea complete a sistemului

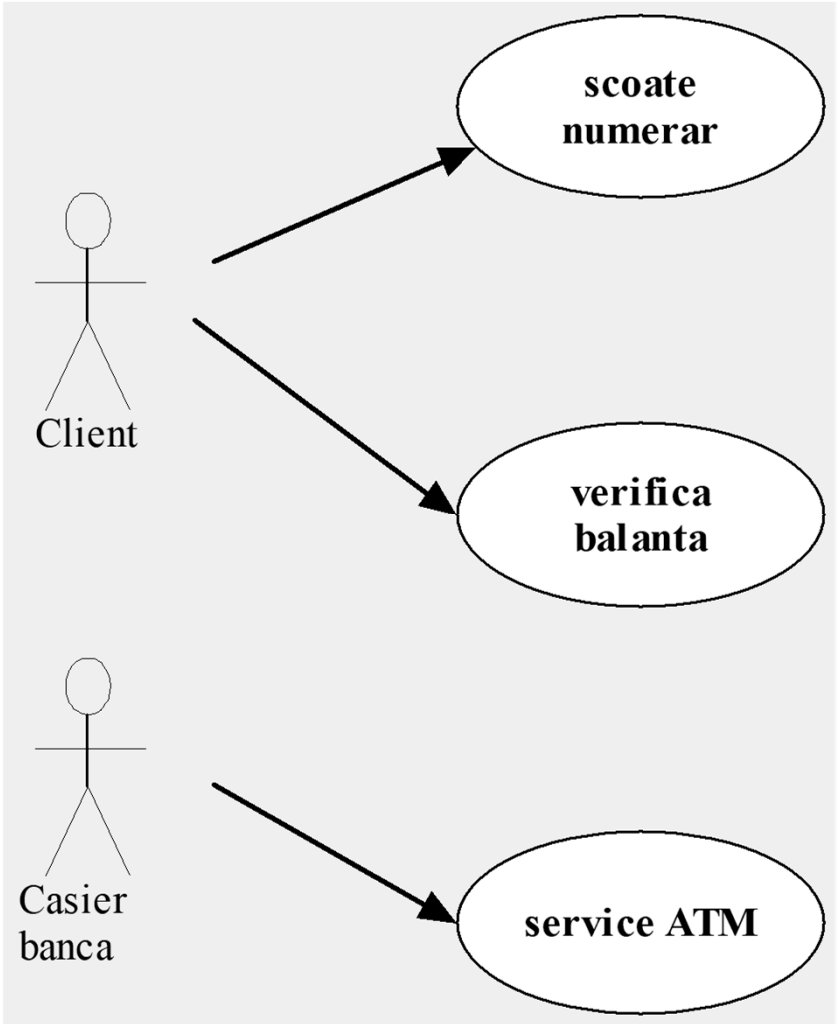
Observatie: CU este initiat de catre un actor si descrie o suita de interactiuni dintre acesta si entitatea (informatica), interactiuni reprezentate printr'un schimb de mesaje

Relatii intre CU

- **Asociere**: participarea unui actor la un CU
- **Extensie**: un CU poate fi extins cu un comportament definit de un alt CU
- **Incluziune**: o instanta a unui CU cuprinde si comportamentul specificat printr'un alt CU
- **Generalizare**

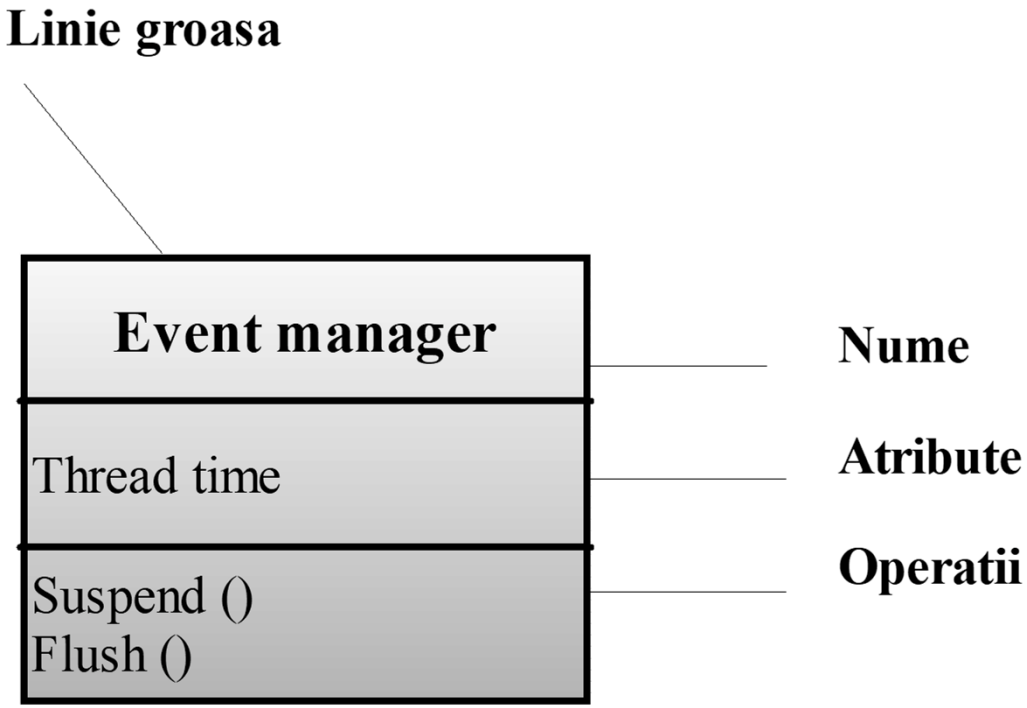
CU - exemplu

- Actorii sunt conectati la cazurile de utilizare numai prin relatii de asociere.



Active Class

- Clasa speciala a carei obiecte gestioneaza unul / mai multe procese / cozi de asteptare
- Poate initia activitati de control.



Nod

- Element ce exista in *run time*.
- Reprezinta o resursa computationala
- In general se refera la memoria si puterea procesorului.