

M. Caramihai, © 2022

**PROGRAMAREA
ORIENTATA
OBIECT**

CURS 5

Introdurre in UML (2)



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]*

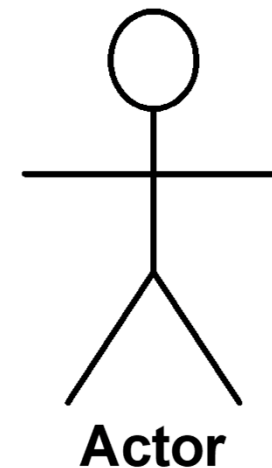
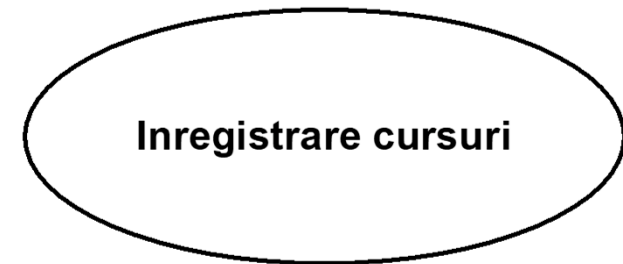
- "An *actor* is an idealization of an external person, process, or thing interacting with a system, subsystem, or class. An actor characterizes the interactions that outside users may have with the system."

- *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
- Reprezentarea interactiunilor sistem-utilizator

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

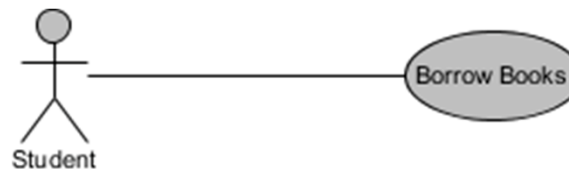
Observatie: CU este initiat de catre un actor si descrie o suite de interactiuni dintre acesta si entitatea (informatica), interactiuni reprezentate printr'un schimb de mesaje

Identificarea CU

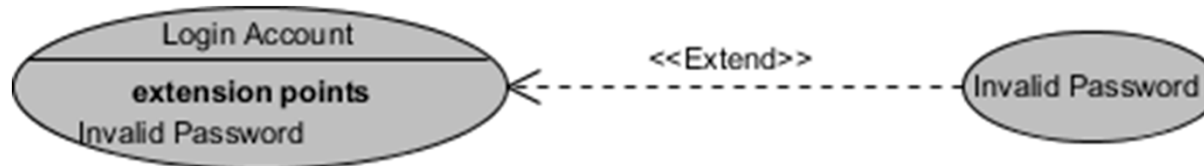
- Identificarea **CU** și apoi procesul de evoluție bazat pe scenarii se bazează pe întrebarea: ce valoare vizibilă externă și observabilă își dorește fiecare actor, de ex:
 - Ce funcții își va dori actorul de la sistem?
 - Sistemul stochează informații? Ce actori vor crea, vor citi, vor actualiza sau vor șterge aceste informații?
 - Este necesar ca sistemul să notifice un actor despre schimbările din starea internă?
 - Există evenimente externe despre care trebuie să știe sistemul? Ce actor informează sistemul despre aceste evenimente?

Relatii intre CU

- **Asociere:** participarea unui actor la un CU



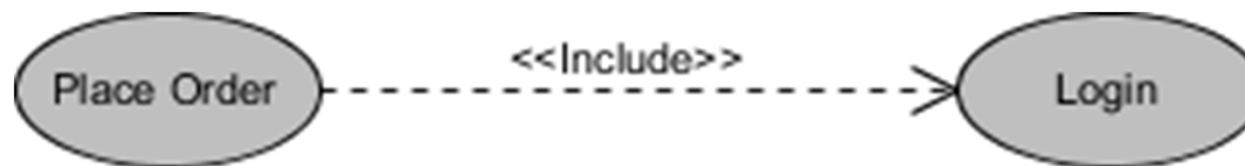
- **Extensie:** un CU poate fi extins cu un comportament definit de un alt CU



Indică faptul că un caz de utilizare „Invalid password” poate include (sub rezerva specificarii în extensie) comportamentul specificat de cazul de utilizare de bază „Login account”.

Relatii intre CU

- **Incluziune:** o instanta a unui CU cuprinde si comportamentul specificat printr'un alt CU



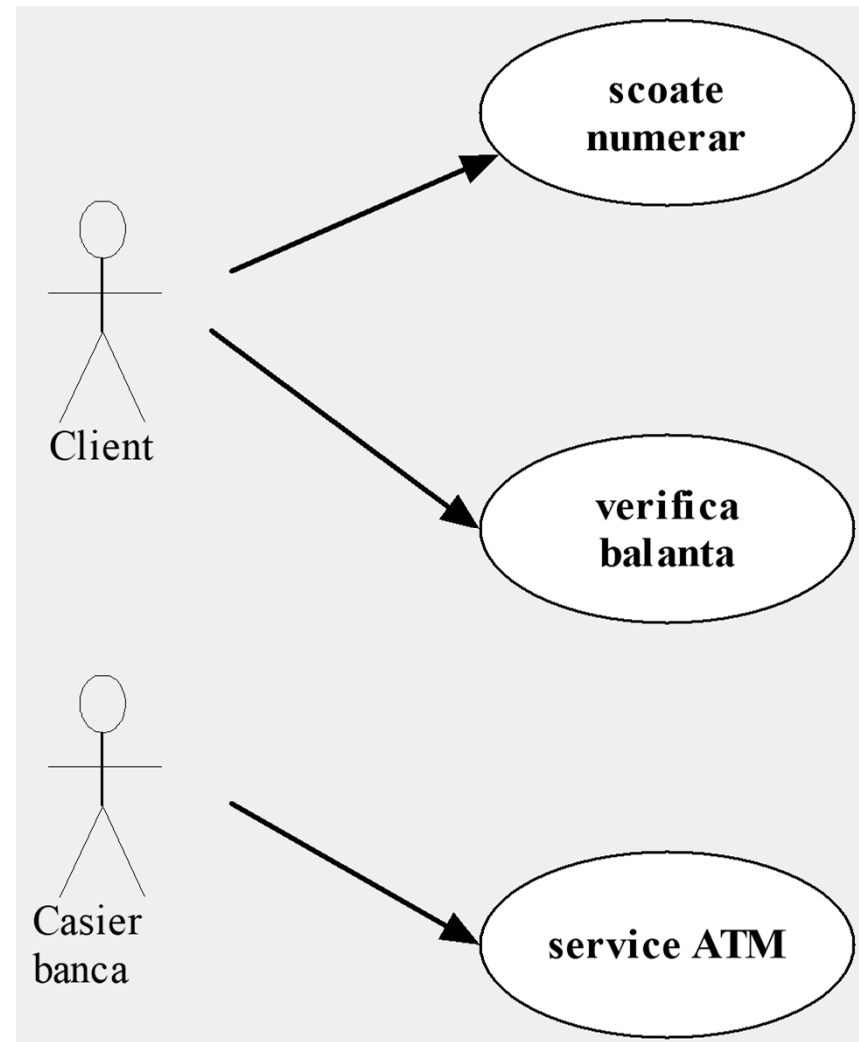
Când un caz de utilizare este descris ca utilizând funcționalitatea altui caz de utilizare, relația dintre cazurile de utilizare este denumită ca **include** sau **use**.

- **Generalizare:** Cazul de utilizare copil reprezinta o îmbunătățire a cazului de utilizare părinte.



CU - exemplu

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



Diagramele UML

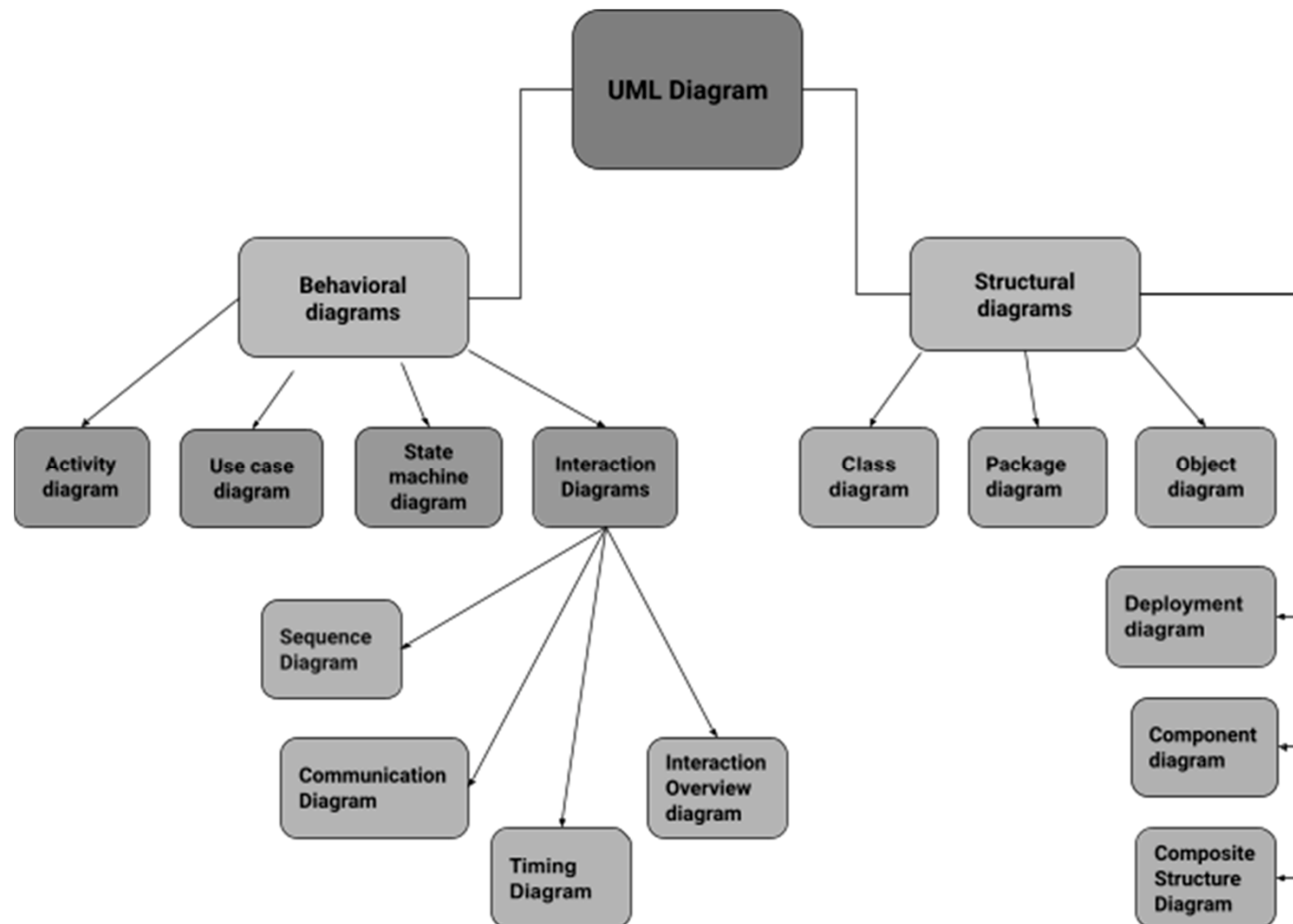


Diagrama de secvente (1)

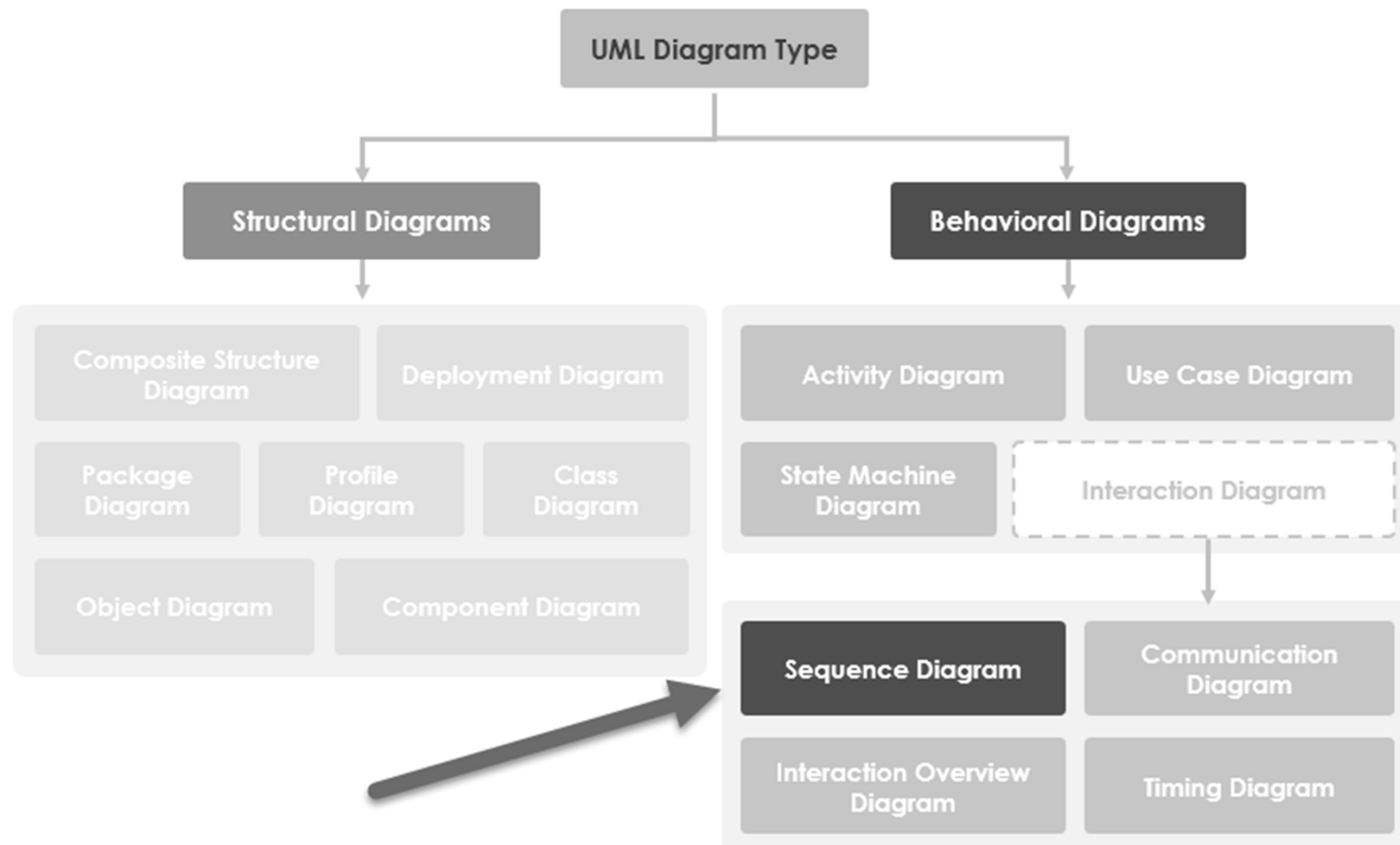


Diagrama de secvente (2)

- **Diagrama de secvente** este o diagrama de interactiuni care reda ordonarea in timp a mesajelor.
- Ea prezinta un set de obiecte si mesajele trimise si primite de acele obiecte.
- Din punct de vedere grafic, diagrama de secvente este un tabel ce prezinta obiectele pe *orizontala* si mesajele (ordonate in sensul parcuregerii timpului) pe *verticala*.

Continut

- **Obiecte:** Schimba mesaje unul cu altul
- **Mesaje:**
 - *Sincrone:*
 - Uzual: apel operație – se trimite mesajul și si se suspenda execuția în așteptarea răspunsului
 - reprezentate prin "sageata plina"; durata trebuie sa fie indicata prin bara de activare sau sageata de intoarcere

Diagrama de secvente (3)

➤ *Asincrone:*

- Se trimite mesajul și si se continua imediat fără a aștepta valoarea returnată
- reprezentata prin "jumătate de sageata"

- Pot exista mesaje de tip «create» si «destroy»

● Tipuri speciale

- **Lost message:** mesajul nu ajunge niciodata la destinatie (d.e. *ping*)



- **Found message:** mesajul a carui origine nu este cunoscuta (d.e. *exception handling*)



Reprezentare

- Un obiect este redat ca o *cutie* din care coboara o linie intrerupta.
- Linia poarta numele de **linia de viata** a obiectului (reprezinta durata de viata a unui obiect pe un anumit interval de timp).
- Mesajele sunt redat cu sageti orizontale ("coboara" odata cu trecerea timpului)
- Conditile (d.e. [check = "true"]) indica faptul ca un mesaj a fost transmis.
- Ordinea obiectelor nu este semnificativa
- Un indicator de iteratie (d.e. * sau [i = 1..n] , indica faptul ca un mesaj va fi repetat de mai multe ori (corespunzator valorii precizate)

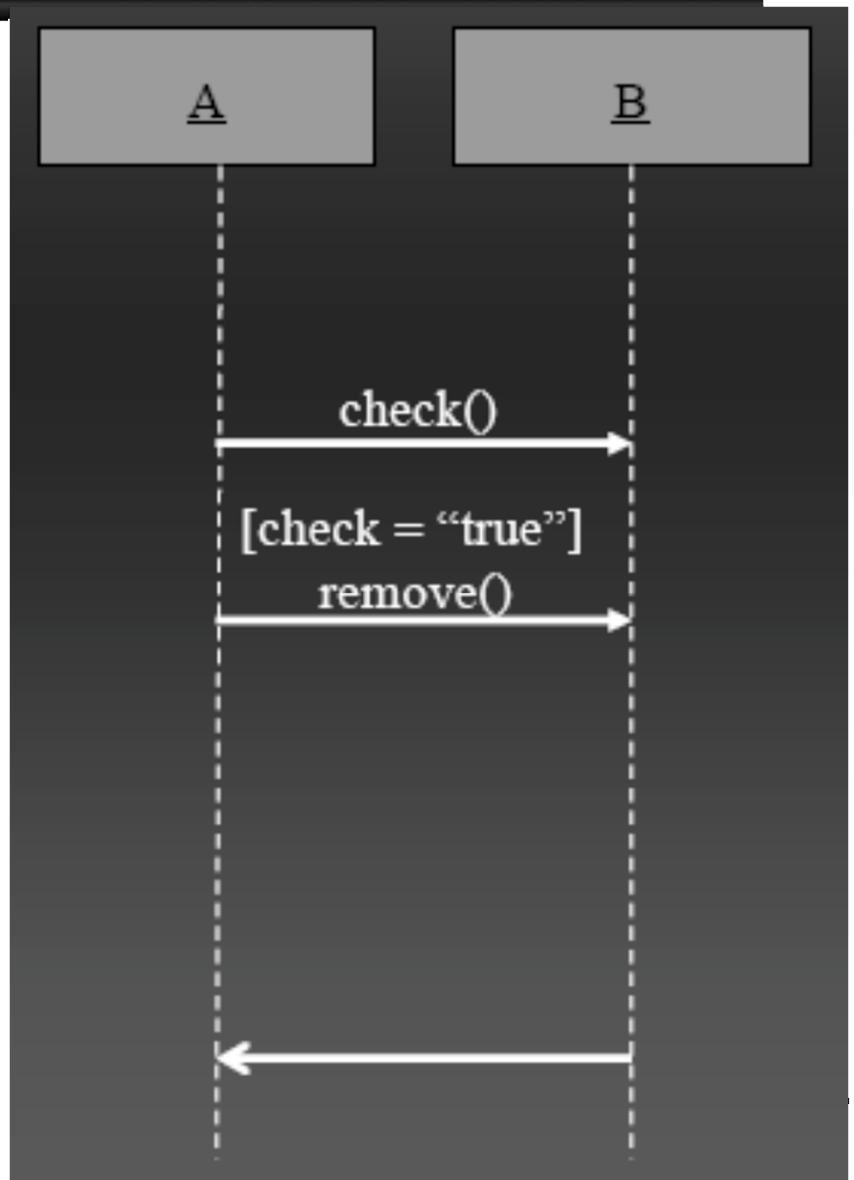


Diagrama de secvente - observatii

- **Diagramade secvente** reprezinta evolutia sistemului in raport cu interactiunile.
- Reprezinta o complementaritate a diagramei de clase (care reprezinta structura sistemului informatic)
- Foarte utila in identificarea obiectelor implicate in diverse activitati.
- Buget mare de timp pentru implementare.
- Principiul de realizare: **KISS** (***K**eep **I**t **S**mall & **S**imple*)

Diagrama de activitati (1)

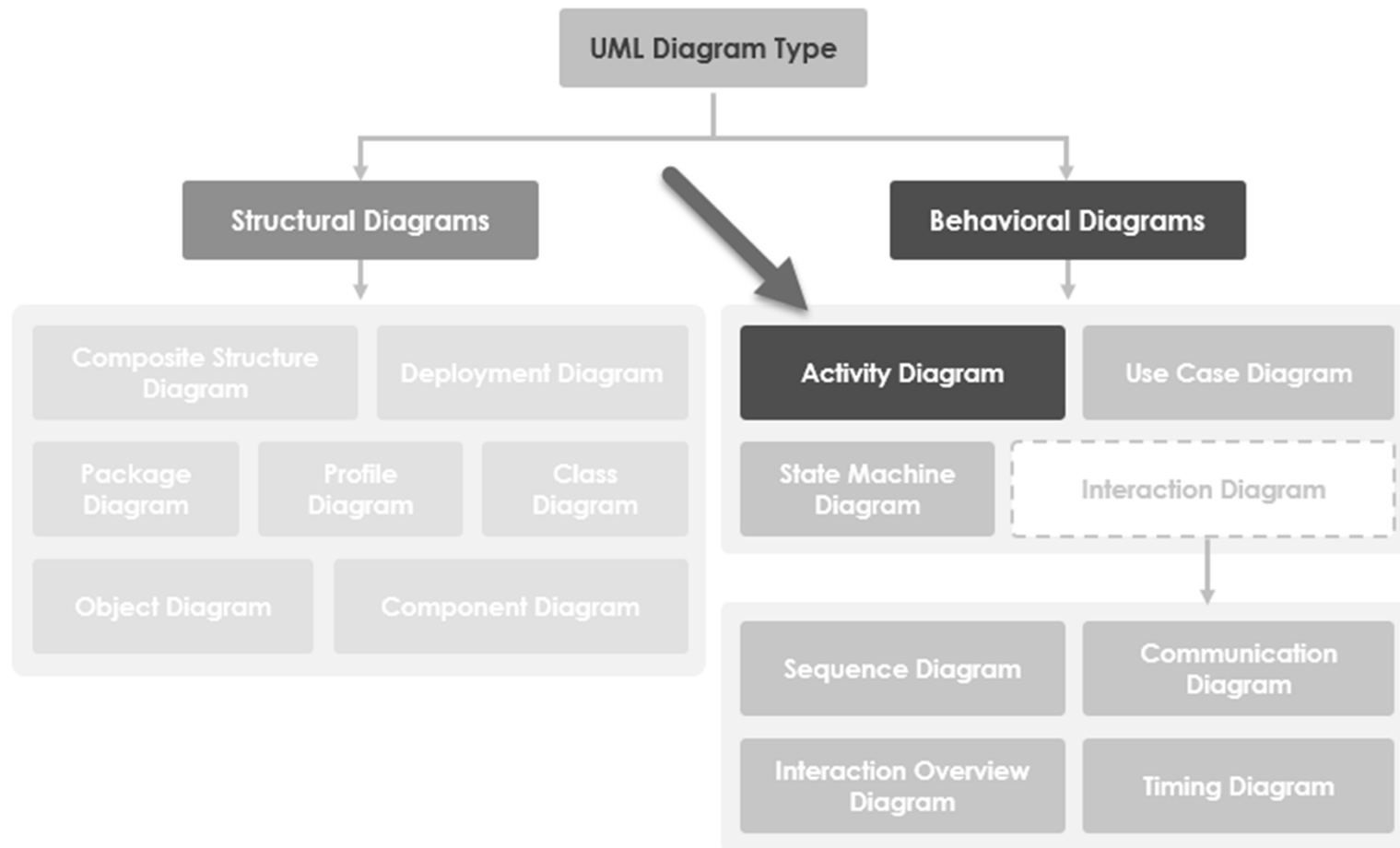


Diagrama de activitati (2)

- **Diagrama de activitati (DA)** este un *flowchart* si indica fluxul de control de la o activitate la alta.
- Diagrama de activitati specifica, dezvolta si documenteaza dinamica unei "societati de obiecte"
- **Diagrama de interactiuni** descrie fluxul de control de la un obiect la altul; **diagrama de activitati** descrie fluxul de control de la o activitate la alta.
- Exista doua feluri de stari:
 - *Actiuni (Action state):*
 - Nu pot fi descompuse
 - Sunt "instantanee" (in raport cu nivelul de abstractizare din model)
 - *Activitati (Activity state):*
 - Pot fi descompuse
 - Activitatea este modelata de o alta diagrama de activitati

Diagrama de activitati (2)








Utilizari:



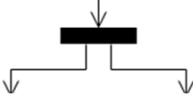
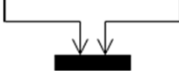
- Identificare cazuri de utilizare candidate, prin examinarea fluxurilor de lucru în afaceri
- Identificare condițiile pre- și post (contextual) pentru cazurile de utilizare
- Modelare fluxuri de lucru între/în cadrul cazurilor de utilizare
- Modelare fluxuri de lucru complexe în operațiuni pe obiecte
- Modelare în detaliu activități complexe într-o activitate de nivel înalt

Caracteristici

- Seamana foarte mult cu diagramele clasice
- Identifica activitatile din cadrul unui sistem (si se bazeaza pe cazurile de utilizare)
- Identifica *tranzitiile intre activitati*
- Descrie *comportamentul* unei *clase* ca raspuns la calculele interne.
- Orientata mai mult catre *business process* decat catre OO
- Foarte utila in faza de testare
- Diagrama de activitati descrie *fluxul* din interiorul unui system
- Diagrama de activitati este un caz special al diagramei de stari in care starile sunt inlocuite cu activitati (functii).

Notatii

Activitate Este folosit pentru a reprezenta un set de acțiuni	
Acțiune O sarcină de îndeplinit	
Controlul fluxului Afișează secvența de execuție	
Fluxul de obiecte Arată fluxul unui obiect de la o activitate (sau acțiune) la o altă activitate (sau acțiune).	
Nodul Inițial Prezintă începutul unui set de acțiuni sau activități	
Nodul final al activității Opriți toate fluxurile de control și fluxurile de obiecte într-o activitate (sau acțiune)	
Nod obiect Reprezentați un obiect care este conectat la un set de fluxuri de obiecte	

Nodul de decizie Reprezentați o condiție de testare pentru a vă asigura că fluxul de control sau fluxul de obiecte merge doar pe o singură cale	
Merge Node Reunește diferite căi de decizie care au fost create folosind un nod de decizie.	
Nodul de furcă Împărțiți comportamentul într-un set de fluxuri paralele sau concurente de activități (sau acțiuni)	
Alăturăți-vă la Node Reunește un set de fluxuri paralele sau concomitente de activități (sau acțiuni).	

Exemplu

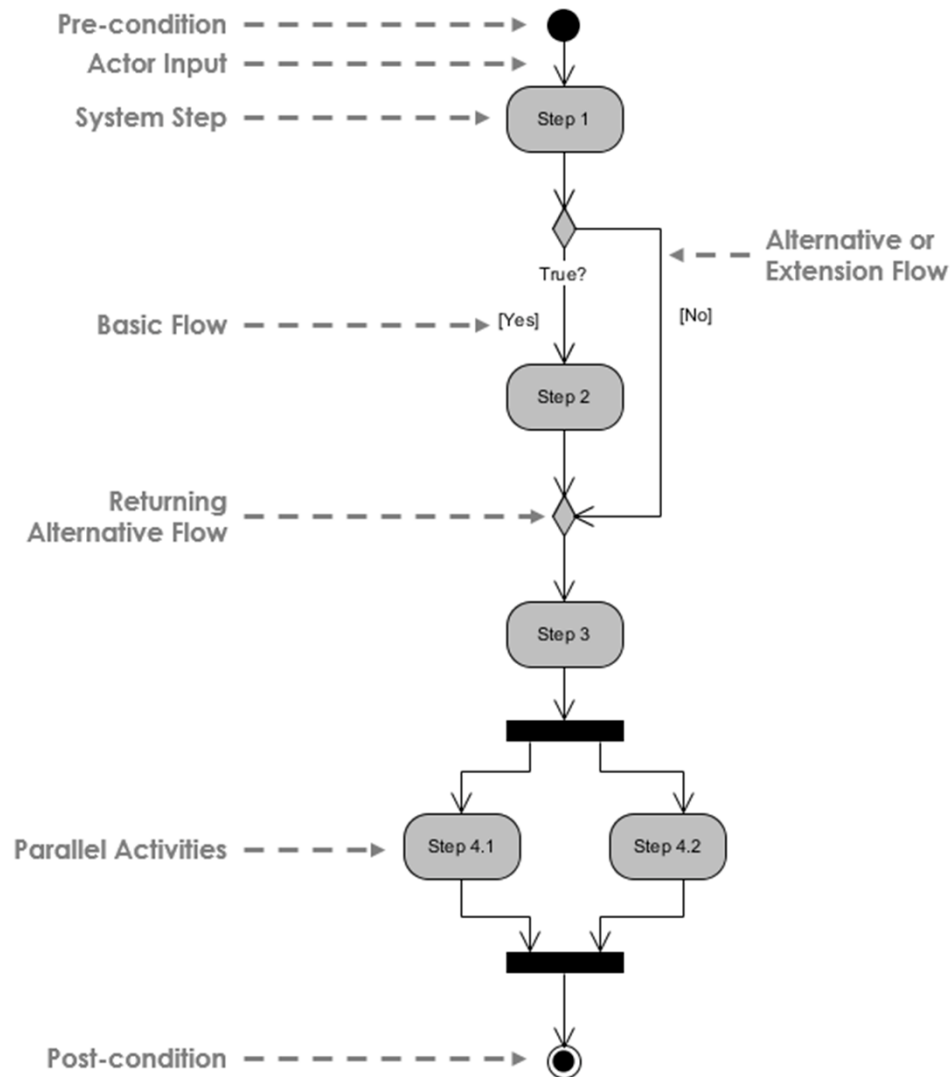


Diagrama de colaborari (1)

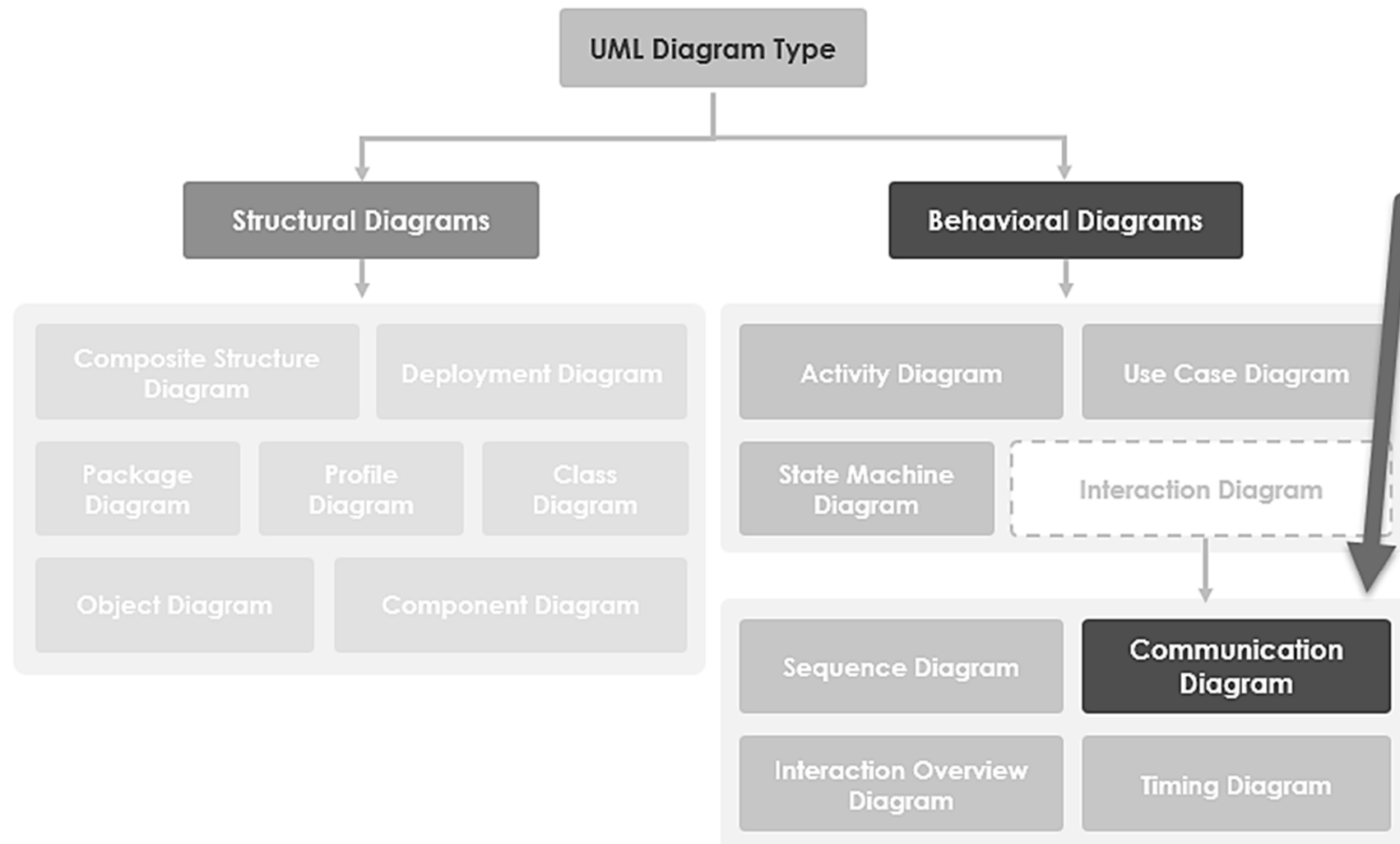


Diagrama de colaborari (2)

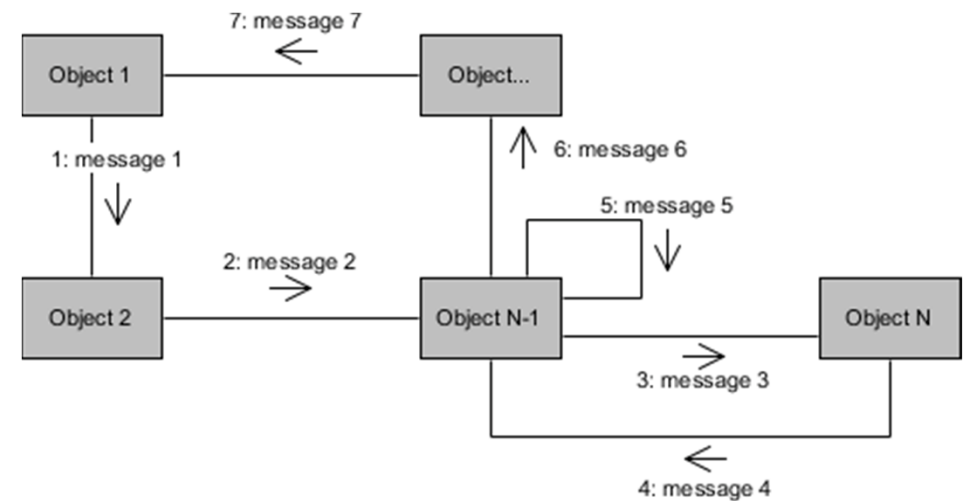
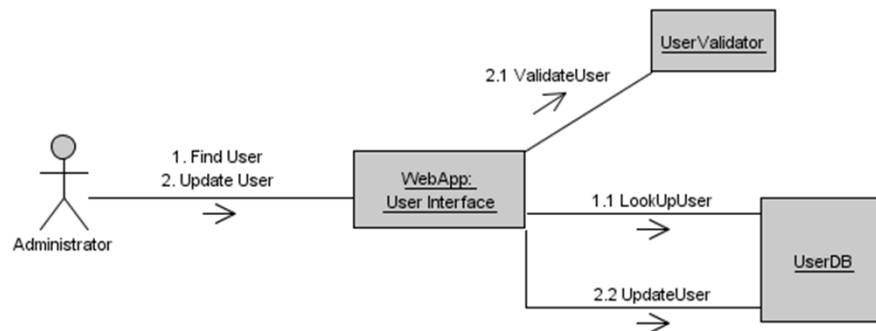
- Prezinta modul in care obiectele interactioneaza unele cu altele (tinand cont de unitatile organizationale)
- Secventa de mesaje este determinata prin numere:
 - 1, 2, 3, 4,
 - 1, 1.1, 1.2, 1.3, 2, 2.1, 2.1.1, 2.2, 3 (prezinta modul in care o operatie apeleaza o alta operatie)

Observatie: DC – **Diagrama de comunicare** (in UML2)

- Continut
 - Obiecte
 - Schimba mesaje unele cu altele
 - Mesaje
 - **Sincrone:** reprezentate prin sageata completa
 - **Asincrone:** "semnale" reprezentate prin sageata incompleta
 - Mesaje de tip «create» sau «destroy»
 - Mesajele sunt numerotate si pot avea "bucle"

Example

- 1. Find User
 - 1.1 LookUpUser
- 2. Update User
 - 2.1 ValidateUser
 - 2.2 UpdateUser



Comparatie DC / DS

- Reprezentarea **diagramei de colaborari** prezinta *conexiunea* dintre obiecte.
- **Diagrama de secvente** permite o buna reprezentare a *fluxului de timp*
- Secventa de mesaje este mai greu de inteles intr'o **diagrama de colaborari**
- *Organizarea obiectelor (si fluxul de control)* sunt cel mai bine identificate in **diagrama de colaborari**
- **Observatii:**
 - Controlul complex este greu de reprezentat printr'o singura diagrama!!!
 - Diagrama de colaborari este mai des folosita decat cea de secvente

Sumar

1. **Diagrama CU**

→modeleaza functionalitatea dpdv user

[Model Functional]

2. **Diagrama de clase**

→modeleaza structura sisemului utilizand obiecte

[Model Obiectual]

3. **Diagrama de interactiuni**

(secvente & colaborari)

→modeleaza mesajele schimbate intre obiecte

[Model Dinamic]

4. **Diagrama de stari**

→modeleaza tranzitiile intre stari

[Model Dinamic]

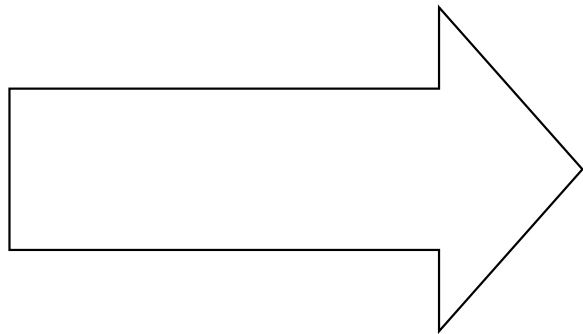
5. **Diagrama de activitati**

→modeleaza fluxul de control (ca tranzitie intre activitati)

[Model Dinamic]

Sisteme de dimensiuni mari

-
- Principiul Roman: ***Divide & impera***
 - ➔ Sistemele mari trebuiesc desfacute in componente mai mici, pentru a putea fi mai usor gestionate
 - Metode structurale: descompunere functionala
 - In OO: gruparea claselor in unitati cu caracteristici comune.



Pachete (*Packages*)

(conceptual: in momentul dezvoltarii sistemului)

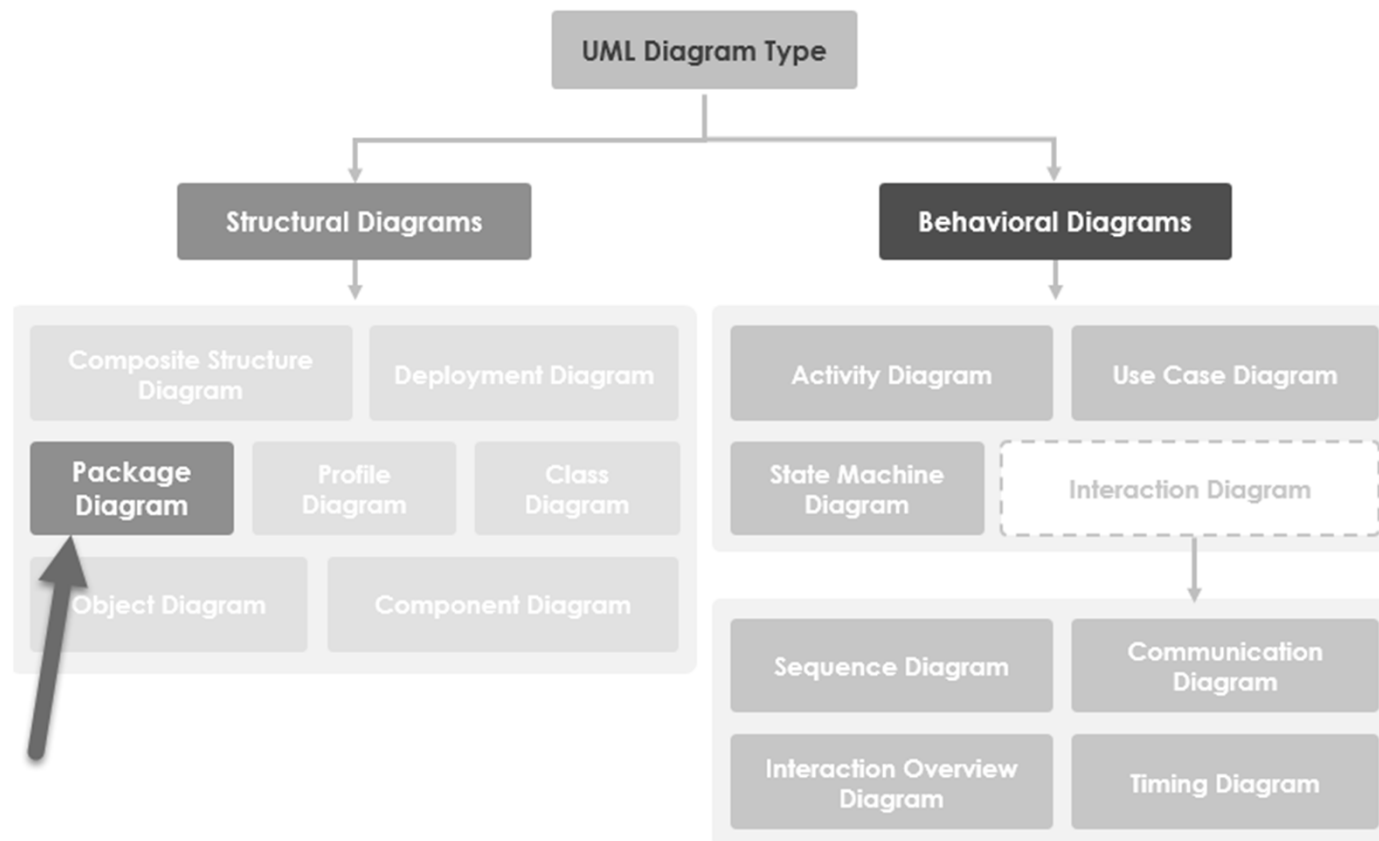
Componente

(fizic: in momentul rularii)

- Diagrama de pachete poate fi folosită pentru a simplifica diagramele de clase complexe (i.e. poate grupa clase în pachete)
- Un pachet reprezinta o colecție de elemente UML legate logic.

Pachete (*Packages*)

Un **pachet** reprezinta o "grupare" de elemente de modelare;
poate contine clase sau alte pachete / diagrame



Elemente

- **Nume** (string)
 - Simple
 - Calificator: numele **P** are ca prefix numele **P** in care se gaseste, d.e. Pachet1::Pachet2
- **Componenta**: un **P** poate contine diferite elemente (d.e. clase interfete, diagrame, alte pachete, etc)

Observatii:

- daca un **P** este distrus, sunt distruse si toate componentele sale
- Orice alement apartine unui singur **P**
- **P** se recomanda a fi pe max 2-3 nivele

- **Vizibilitate**: idem ca la clase (usual: *public*)

Pachete - exemple

Vanzari

Client

Ordin

Depozit

Locatie

Item

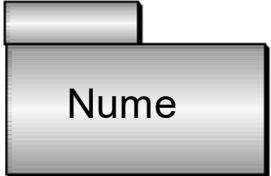


Stoc

Ordin

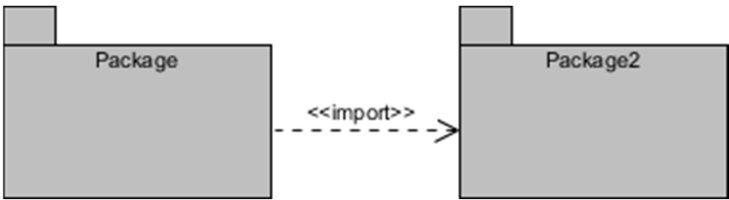
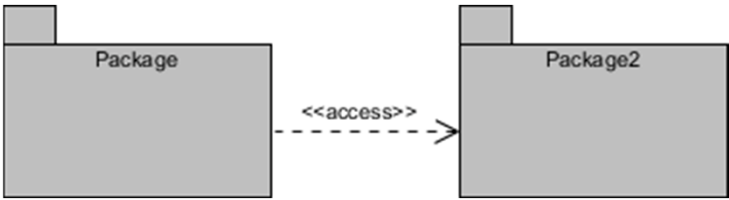
Pachete – caracterizare

- Un **pachet** poate contine diferite tipuri de elemente de modelare
 - Pot fi incluse si alte pachete in vederea dezvoltarii de ierarhii.
- Un **pachet** defineste un “spatiu” pentru elementele ce le contine (similar cu directoarele din DOS / Windows)
- **Pachetele** sunt utilizate in general ca structuri de grupare a unor elemente cu o semantica comuna.
- Diagrama prezinta o “vedere de sus” asupra sistemului.

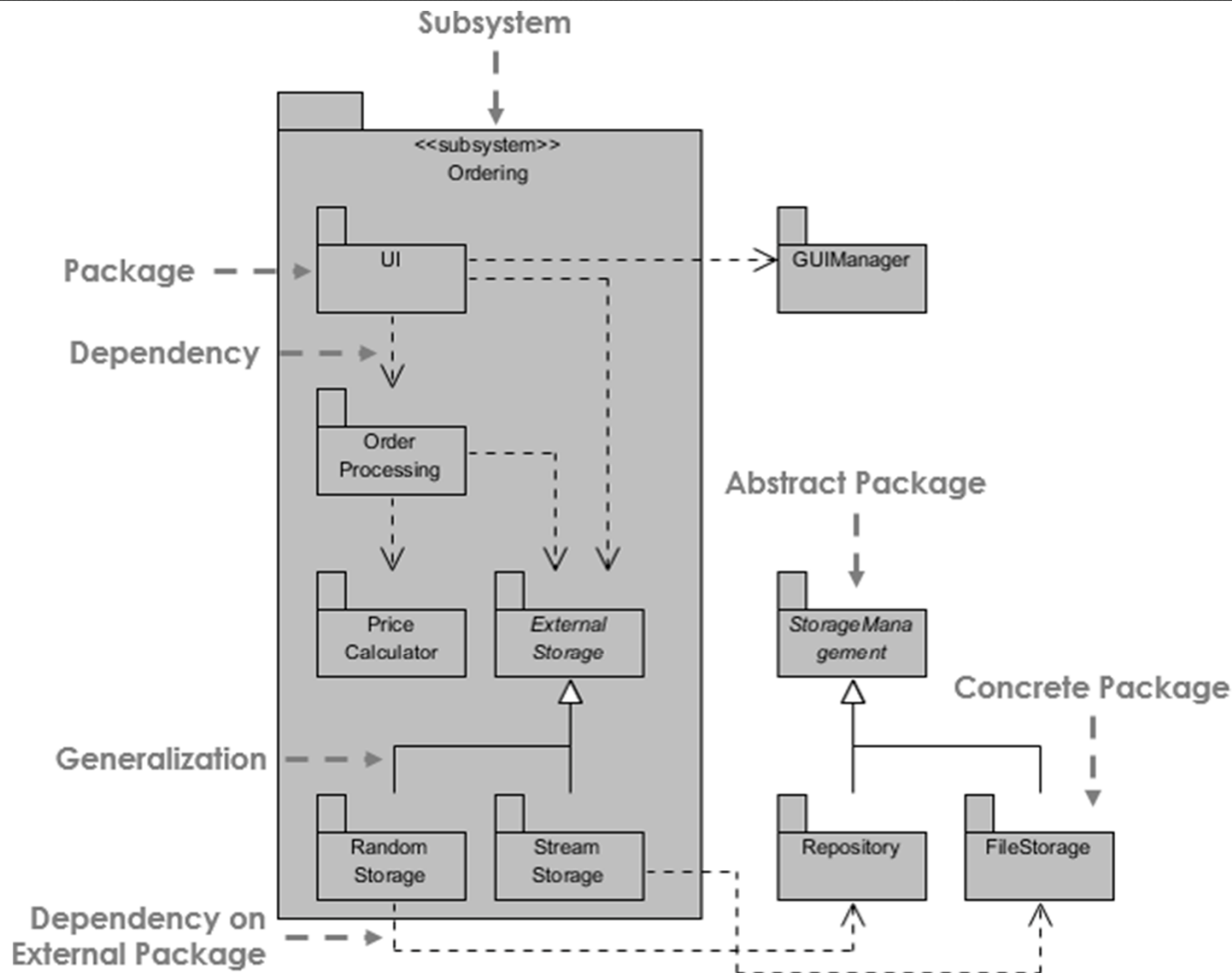
Concepte de baza

<i>Package</i>	Grup de elemente de modelare.	
<i>Import</i>	Relatie de dependenta; indica faptul ca elementele de continut ale pachetului tinta sunt adaugate spatiului pachetului sursa (extinde spatial de lucru al P importator)	<<import>> 
<i>Access</i>	Relatie de dependenta; indica faptul ca elementele de continut ale pachetului tinta pot fi accesate in raport cu spatiul pachetului sursa (permite utilizarea elementelor dintr'un alt P prin specificarea caii de acces la acestea).	<<access>> 

Concepte de baza

<i>Package</i>		
<i>Import</i>	Un pachet importa functionalitatile celui alt pachet	 <p>The diagram shows two package boxes, 'Package' on the left and 'Package2' on the right. A dashed arrow points from 'Package' to 'Package2' with the label '<<import>>' above it.</p>
<i>Access</i>	Un pachet necesita un "ajutor" din partea celui alt pachet pentru a functiona	 <p>The diagram shows two package boxes, 'Package' on the left and 'Package2' on the right. A dashed arrow points from 'Package' to 'Package2' with the label '<<access>>' above it.</p>

Relatii intre pachete – exemplu



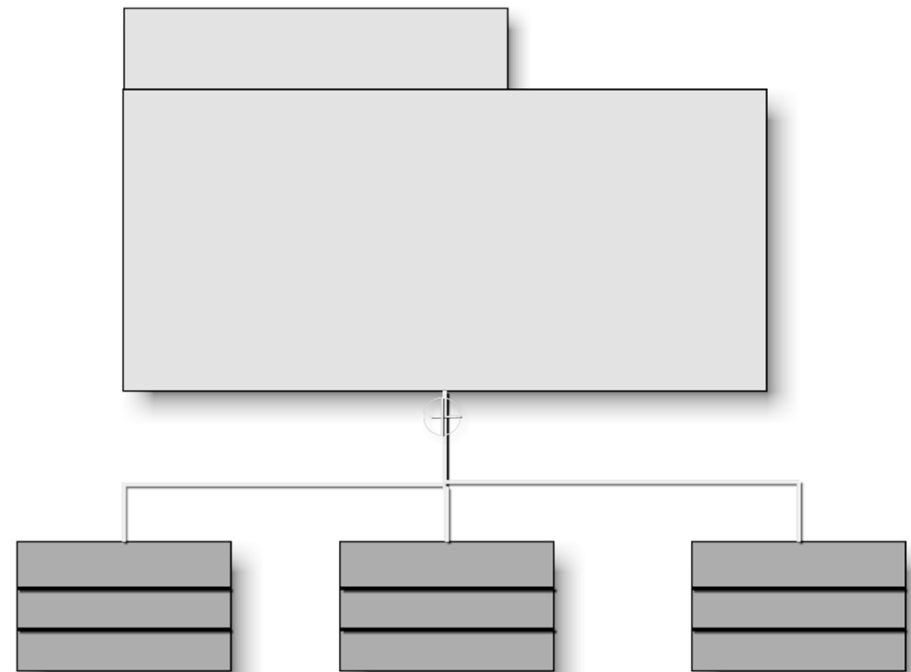
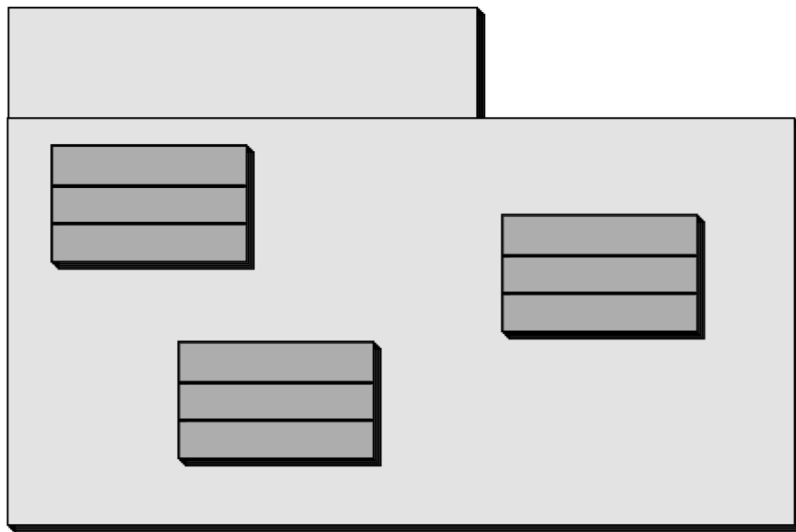
Mostenirea

Un pachet aflat in relatie de generalizare cu un alt pachet va **mosteni** elementele publice / protejate ale acestuia daca:

- Elementele apartin pachetului de la nivelul ierarhic superior
- Elementele sunt importate de catre pachetul de la nivelul ierarhic superior

Reprezentarea pachetelor (continut)

- Pachetele sunt reprezentate prin intermediul diagramelor statice
- Exista doua modalitati echivalente de reprezentare:



Utilizarea pachetelor

- Cerinte

- Grd cat mai inalt de coeziune interna
- Nivel cat mai redus de cuplare externa
- Unitatea scopului

Utilizari

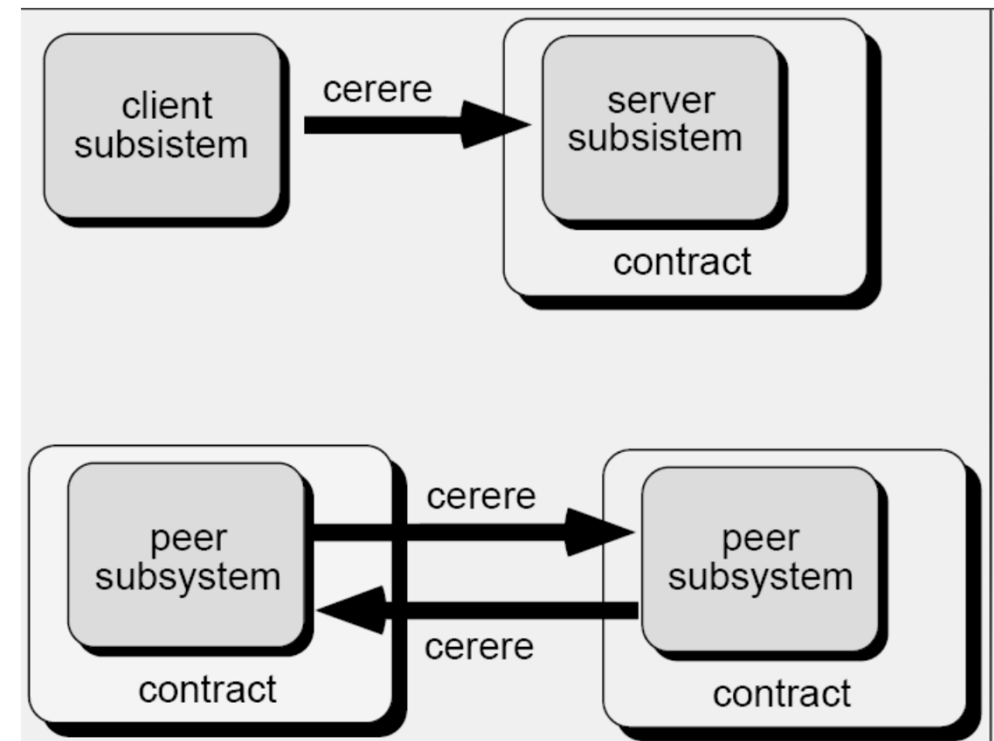
- Crearea unei “vederi de ansamblu” asupra unui mare numar de elemente de modelare
- Organizarea modelelor de mari dimensiuni
- Gruparea elementelor interconectate
- Separarea spatiilor de lucru

Pachete – reguli de configurare

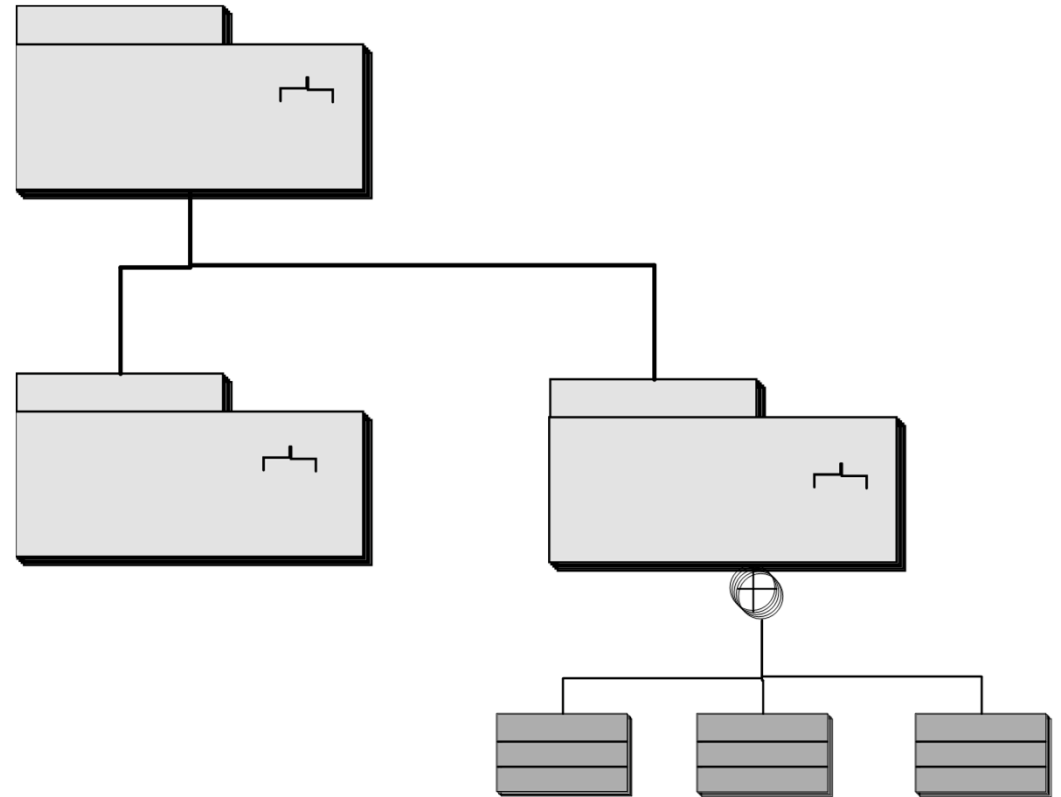
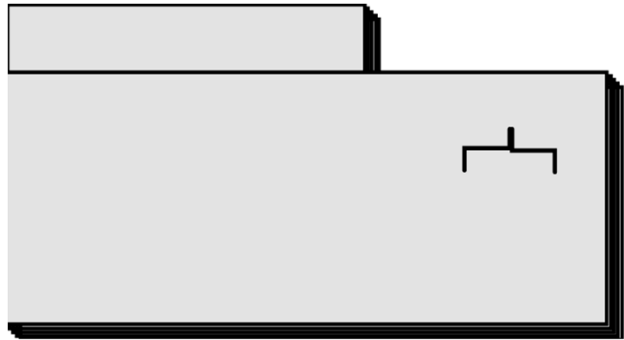
- Elementele de modelare puternic cuplate trebuie sa se gaseasca in acelasi pachet
- Elementele de modelare slab cuplate trebuie sa se gaseasca in pachete diferite.
- Trebuisc evitate relatiile (in special asocierile) dintre elementele de modelare aflate in pachete diferite.
- Un element importat intr'un pachet nu trebuie sa "cunoasca" cum este folosit in acel pachet.

Sisteme & subsisteme

- **Sisteme:** un set e elemente de organizare utilizate in vederea implementarii unui scop
- Legatura dintre sisteme si subsisteme este de tip **compozitie**.
- Arhitecturi C/S
 - Structuri monolitice
 - Pe 2 nivele
 - Pe 3 nivele



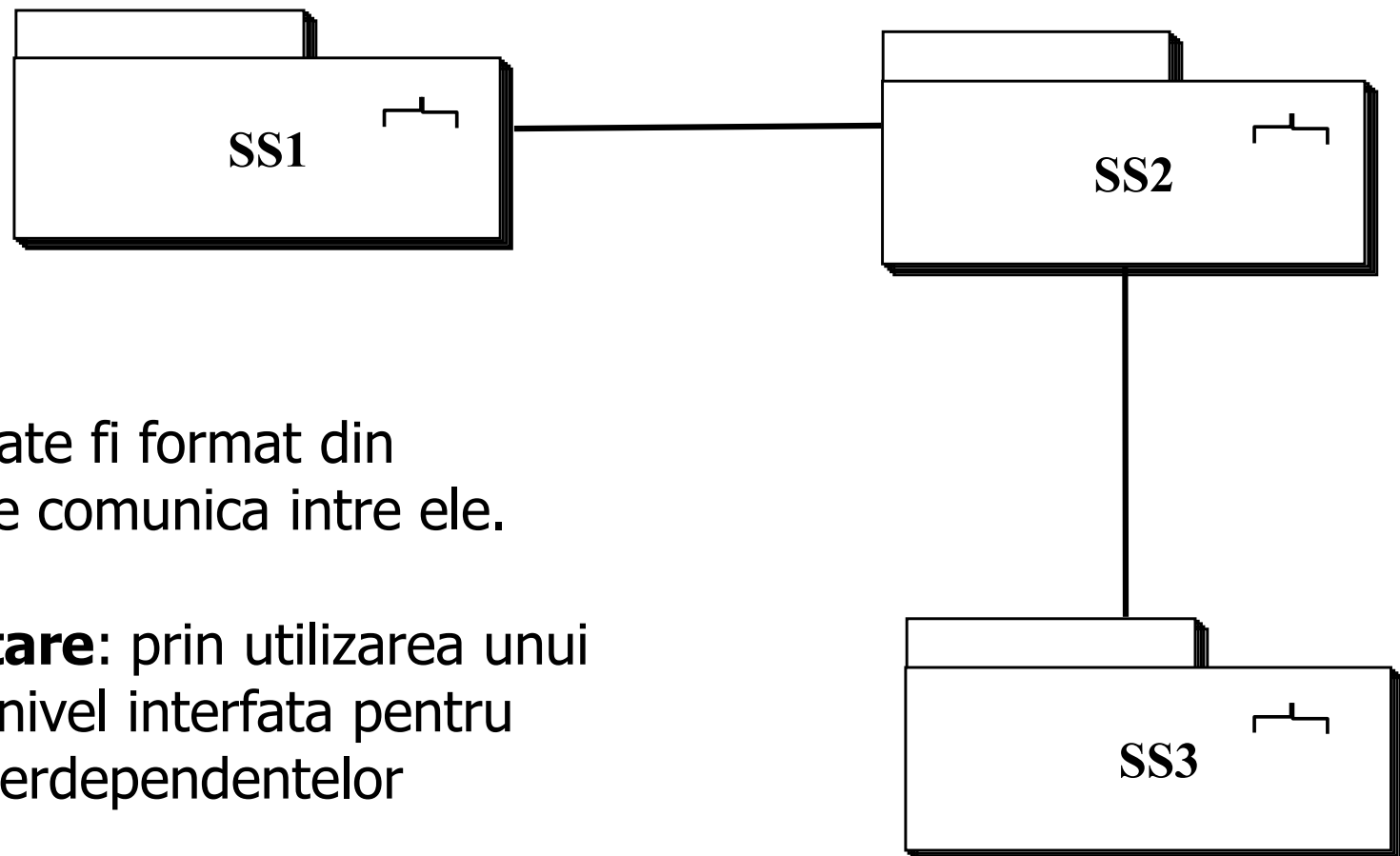
Subsisteme



Subsistemele sunt utilizate in descompunerea sistemelor.

Grup de elemente de modelare ce formeaza impreuna o unitate de comportament in sistem fizic

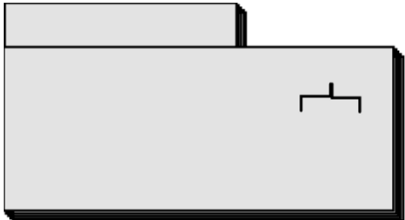
Subsystem – exemplu



Un sistem poate fi format din subsisteme ce comunica între ele.

Interconectare: prin utilizarea unui subsistem la nivel interfață pentru reducerea interdependențelor

Concepte de baza

Subsystem	Un grup de elemente de modelare ce reprezinta o unitate de evolutie intr'un sistem fizic.	
------------------	---	--

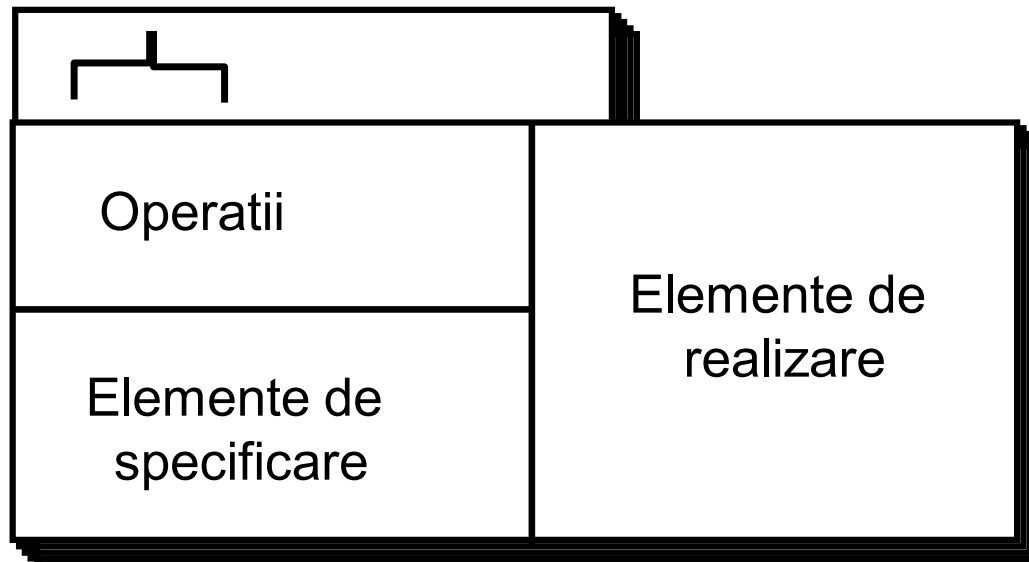
Subsistemele se definesc dpdv: *functional*, *logic* si *avand coeziune fizica*

Subsisteme – componente

- Un subsistem are doua componente:
 - O componenta *externa* – prezinta serviciile furnizate de subsistem (i.e. **specificare**)
 - O componenta *interna* – prezinta configurarea subsistemului (i.e. **realizare**)
- Exista o *mapare* intre cele doua componente.

Caracteristicile subsistemelor

Un **subsistem** are elemente de specificare, respectiv de realizare pentru a putea implementa cele doua componente ale sale

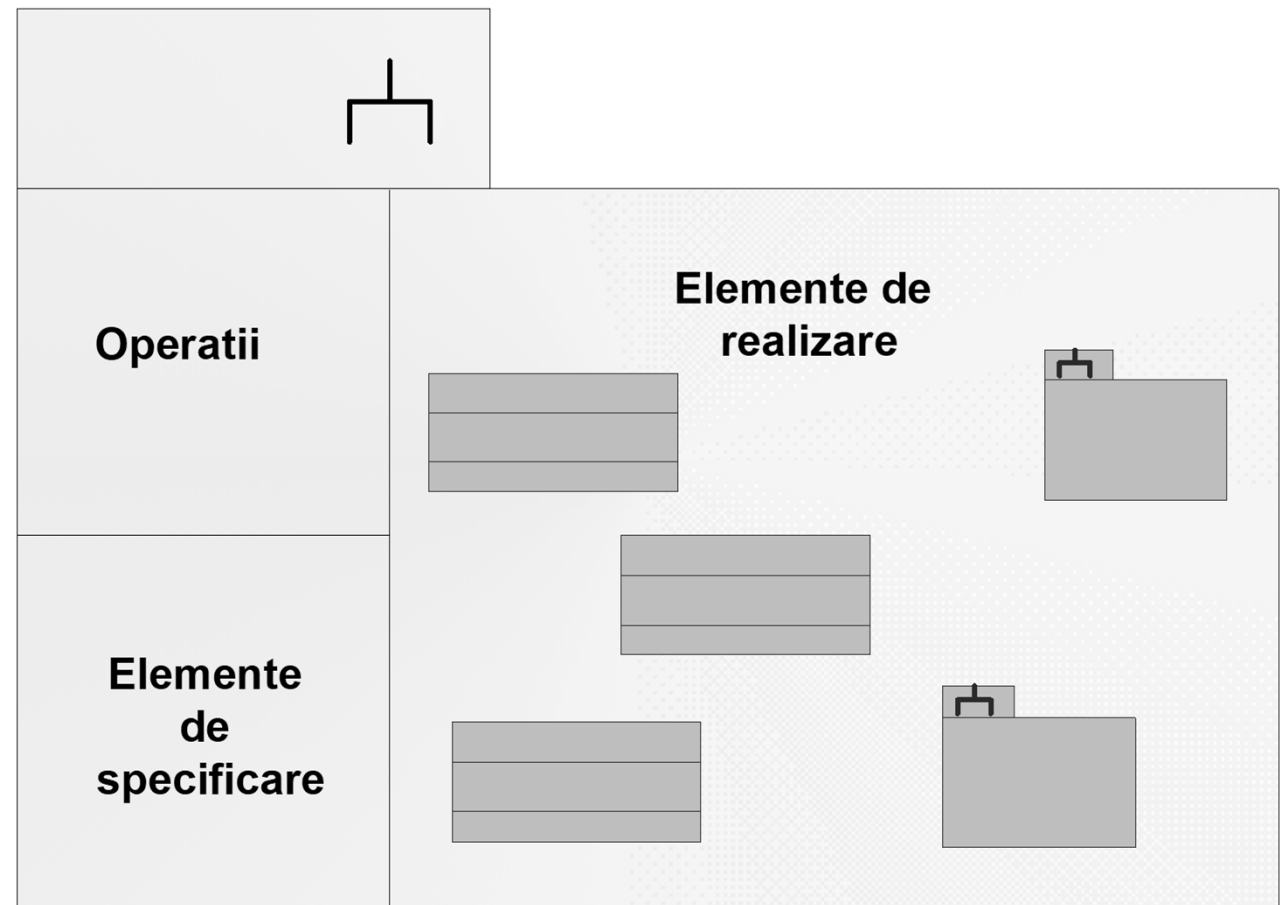


Elemente de realizare

→ Elementele de realizare definesc continutul actual al subsistemului.

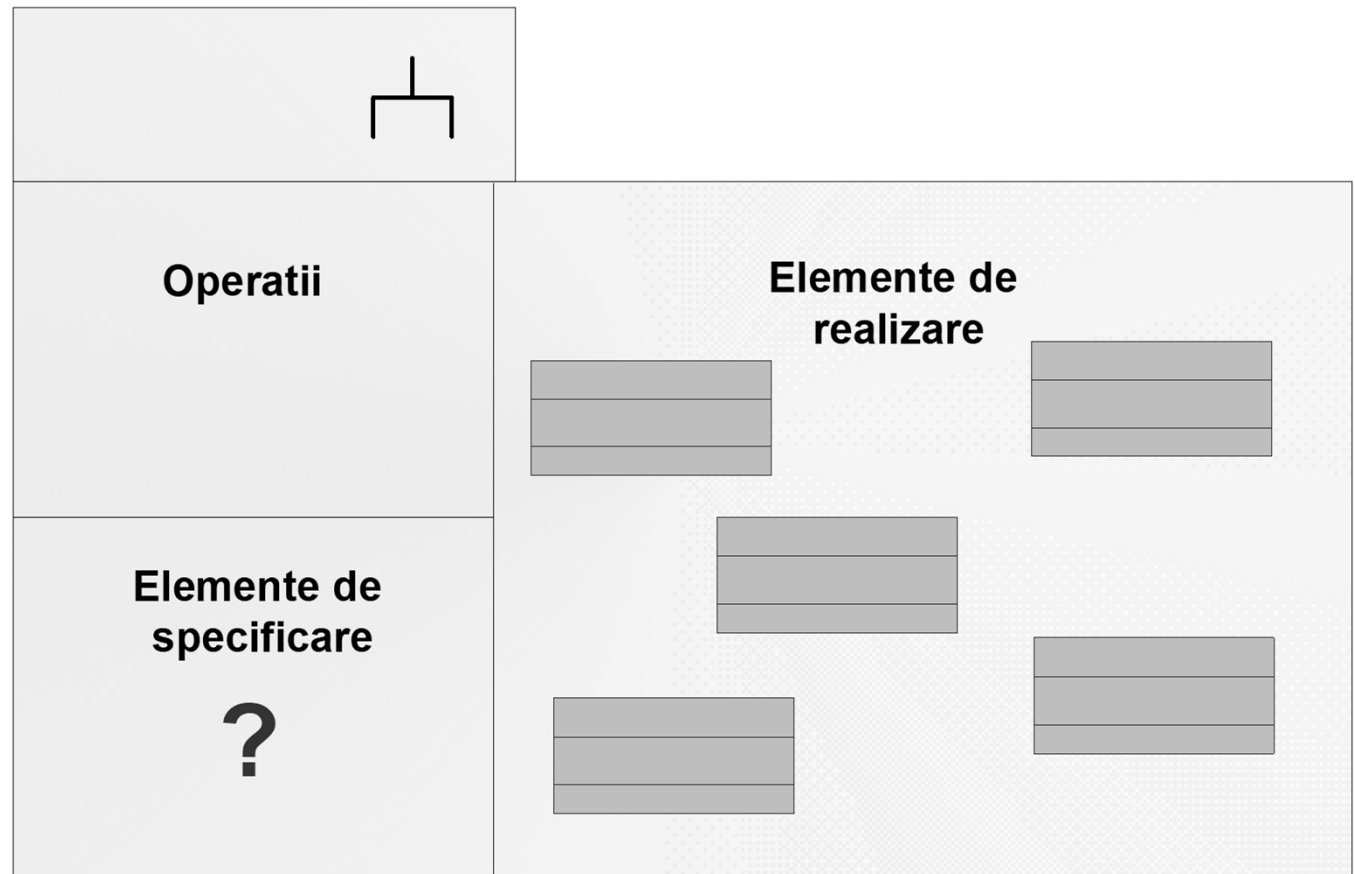
→ Elementele de realizare consta in:

- Clase si relatiile dintre ele
- Ierarhii de subsisteme



Elemente de specificare (1)

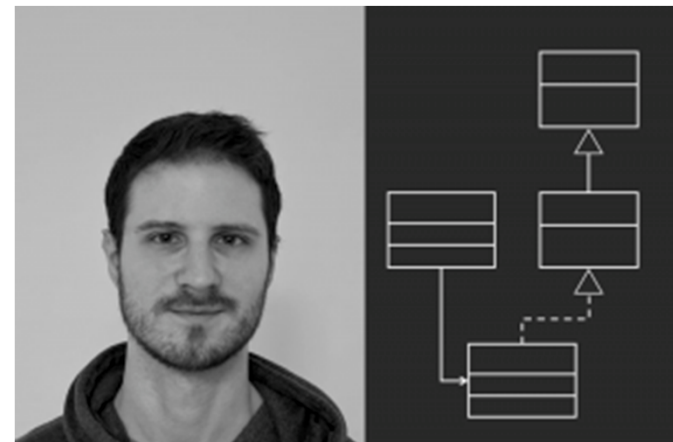
- Elementele de specificare definesc vizibilitatea externa a subsistemului



Elemente de specificare (2)

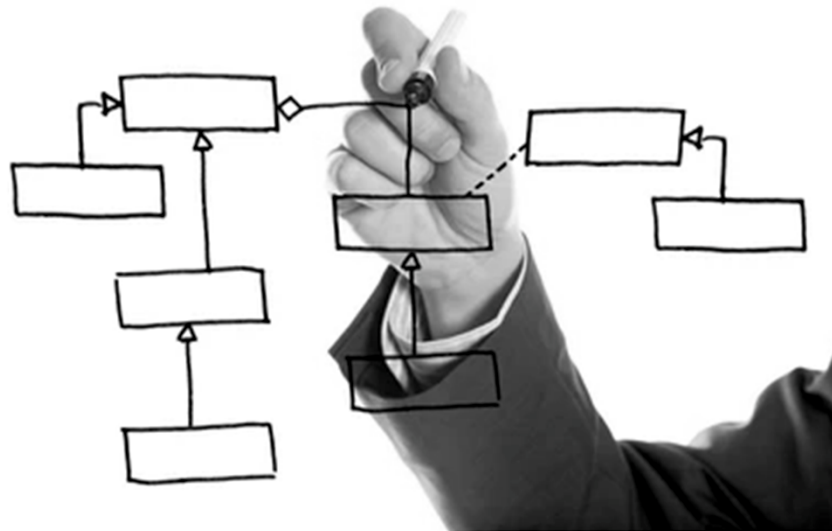
Elementele de specificare:

- Descriu serviciile oferite de subsistem
- Descriu evolutia exterioara a subsistemului
- Nu dau nici o informatie despre structura interna a subsistemului
- Descriu interfata subsistemului



Cand se utilizeaza subsistemele ?

- Descompunerea sistemelor mari in module / componente mai mici
- Dezvoltarea sistemelor distribuite
- Asamblarea unui set de module intr'un sistem de dimensiuni mai mari
- Dezvoltarea sistemelor bazate pe structuri multiple



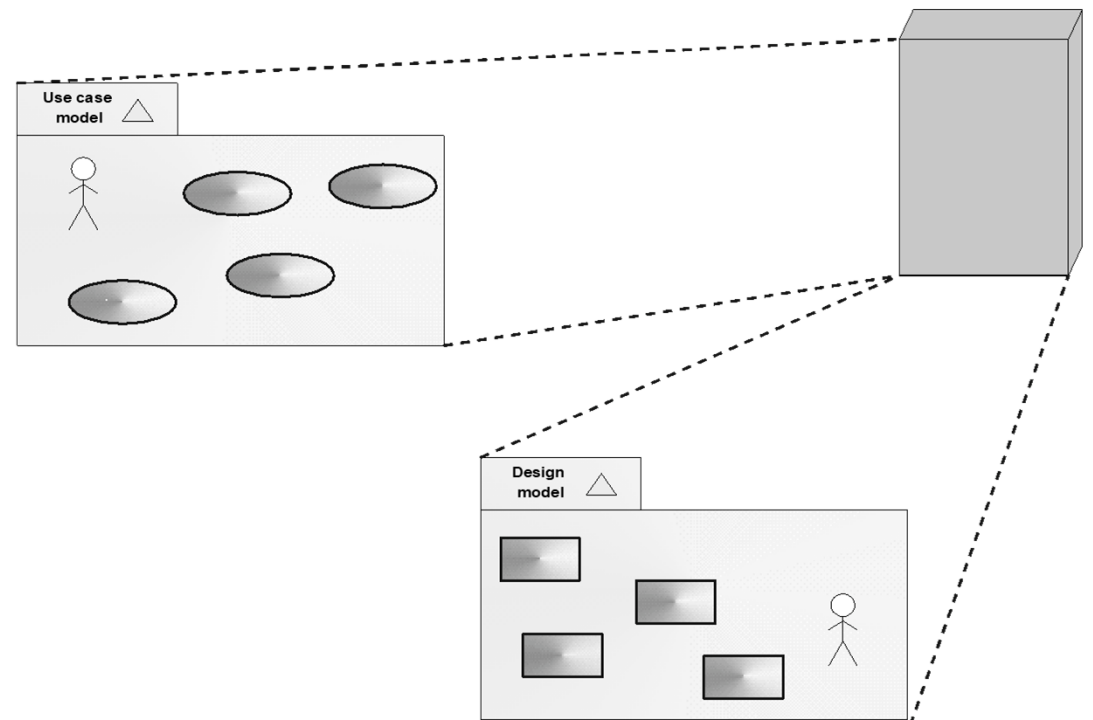
Subsisteme – reguli de configurare

- Un subsistem se definește pentru fiecare parte a unui sistem de mari dimensiuni
- Specificațiile tehnice sunt dependente de tipul sistemului / subsistemului
- Fiecare subsistem trebuie să fie proiectat în mod independent




Modele

- Un model surprinde o “vedere” asupra unui sistem si incearca o “descriere” completa a acestuia
- In **UML 2.5** nu este foarte consistenta definirea domeniului de aplicare a modelului. Într’un loc se spune că un model surprinde un **view** a unui sistem fizic, în timp ce în altul - acel sistem este înțeles în sensul cel mai larg și poate include nu numai software și hardware, ci și organizații și procese.



Concepte de baza

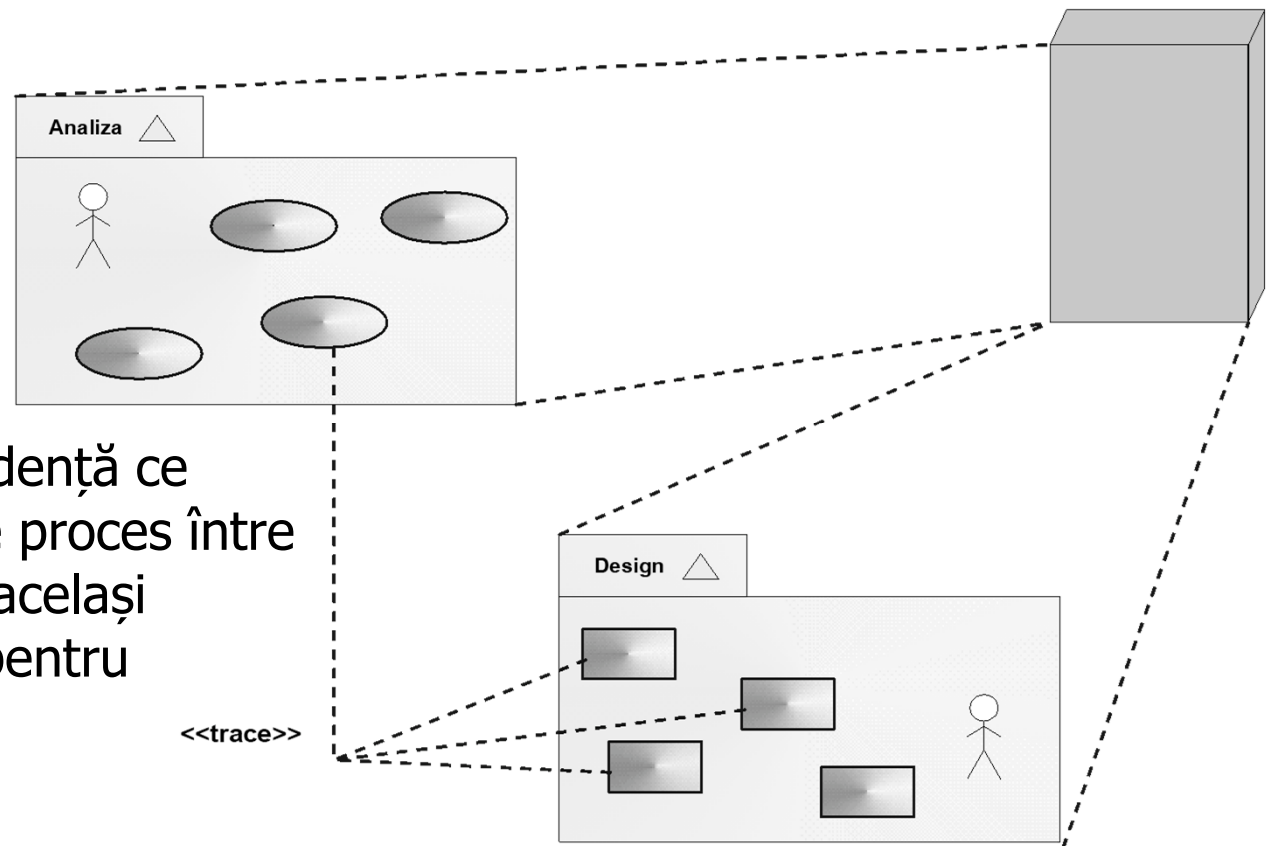
Model	O viziune asupra unui sistem, intr'un anumit scop, in vederea determinarii aspectelor semnificative ale sistemului	
<i>Trace</i>	O conexiune intre elementele de modelare care caracterizeaza acelasi concept in raport cu mai multe modele. (nu exista o relatie directa intre modele ci doar relatii de tip "trace" intre elementele diferitelor modele)	<p><<trace>></p> <p>-----</p>

Trace

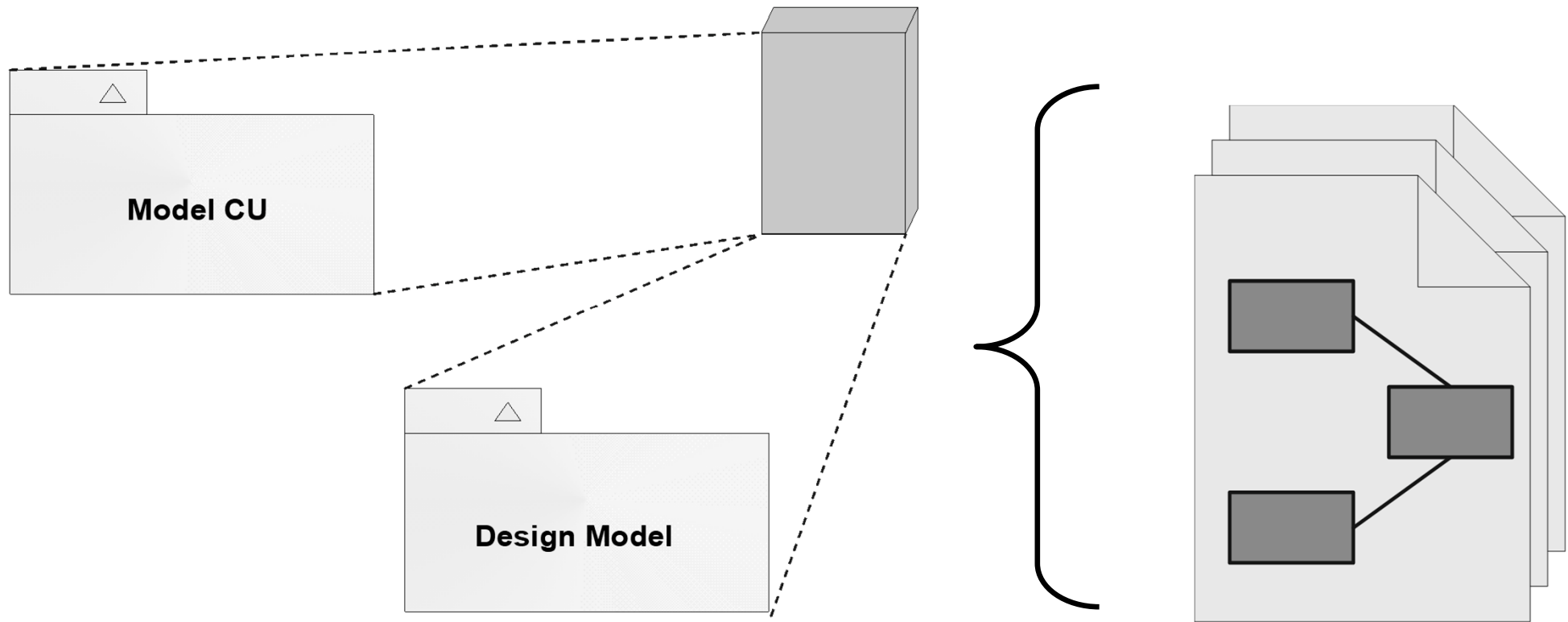
Este un stereotip al relatiei de dependenta

Nu se aplica elementelor ce apartin aceluiasi model

Reprezinta (generic) o dependență ce indică o relație istorică sau de proces între două elemente ce reprezintă același concept fără reguli specifice pentru derivarea unuia din celălalt.



Modele *vs.* diagrame



- Diagramele specifica detaliile tehnice ale unui model.

Utilizarea modelelor

Modelele se folosesc pentru:

- A oferi diferite "viziuni" asupra sistemului pentru categorii diferite de utilizatori
- A sublinia diversele aspecte ref. evolutia sistemului in timp
- A evidentia diversele etape de evolutie a unui sistem informatic

