Modelarea arhitecturii software în UML

Prof. univ. dr. ing. Florica Moldoveanu

Curs *Ingineria programelor* – UPB, Automatică şi Calculatoare
2020-2021

UML pentru modelarea arhitecturii software

Reprezentarea arhitecturii

- *Diagrame de componente* – pentru redarea relatiilor structurale dintre componentele software

- *Diagrama de distributie* – pentru a reda repartitia artefactelor software pe noduri hardware, la momentul executiei.

Reprezentarea fluxului global al controlului, procese în sistem

- *Diagrame de activitate*

Gruparea elementelor de modelare în subsisteme

- *Pachete UML*

Componente UML(1)

In UML 1.x

O componenta este un element software fizic din componenţa unui sistem: fişier executabil, componentă binară, document, fişier cod sursa, fisier date, tabelă a unei baze de date.

➢ O componentă binară este o parte fizică substituibilă a unui sistem, care realizează si este în conformitate cu un set de interfețe.

▪ Componentele binare sunt independente de limbajul de programare în care au fost codificate iar utilizarea lor se bazeaza exclusiv pe interfețe.

▪ Tehnologii folosite pentru crearea de componente binare: COM+, DCOM, CORBA, Java Beans, .NET.

Reprezentarea grafica a unei componente în UML 1.x

Componente UML(2)

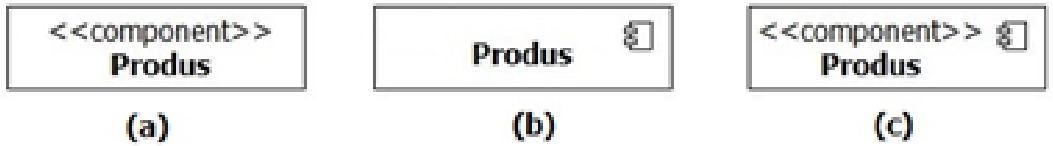
In UML 2.x

❖ **O componentă este o construcţie logică definită la proiectarea sistemului.**

❖ O implementare a unei componente poate fi uşor reutilizată sau înlocuită cu o altă implementare, deoarece componenta "încapsulează" un comportament expus prin interfeţele sale.

O componenta poate fi:

- **Un subsistem**
  - care nu are un corespondent la executie - de ex. subsistemul care reprezinta un nivel într-o arhitectură ierarhică
  - care are un corespondent la executie – de ex. un server de baze de date
- **O componentă binara**



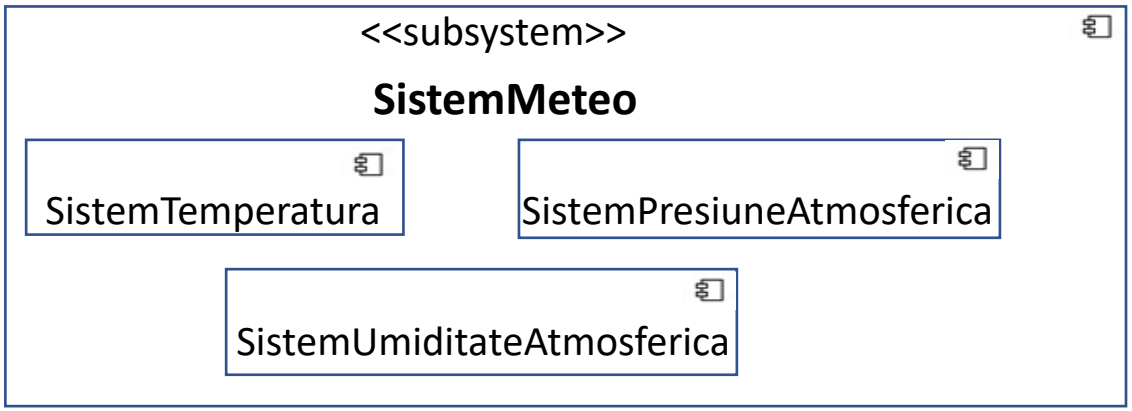
Reprezentarea grafica (de nivel inalt) a unei componente în UML 2.x.

Componente UML(3)

Tipul unei componente poate fi precizat printr-un stereotip:

«subsystem»

- O componentă ”subsistem” reprezintă o unitate de decompoziţie a unui sistem software de dimensiune mare. Un subsistem poate grupa mai multe componente şi este parte aproape independentă a unui sistem.



«process»

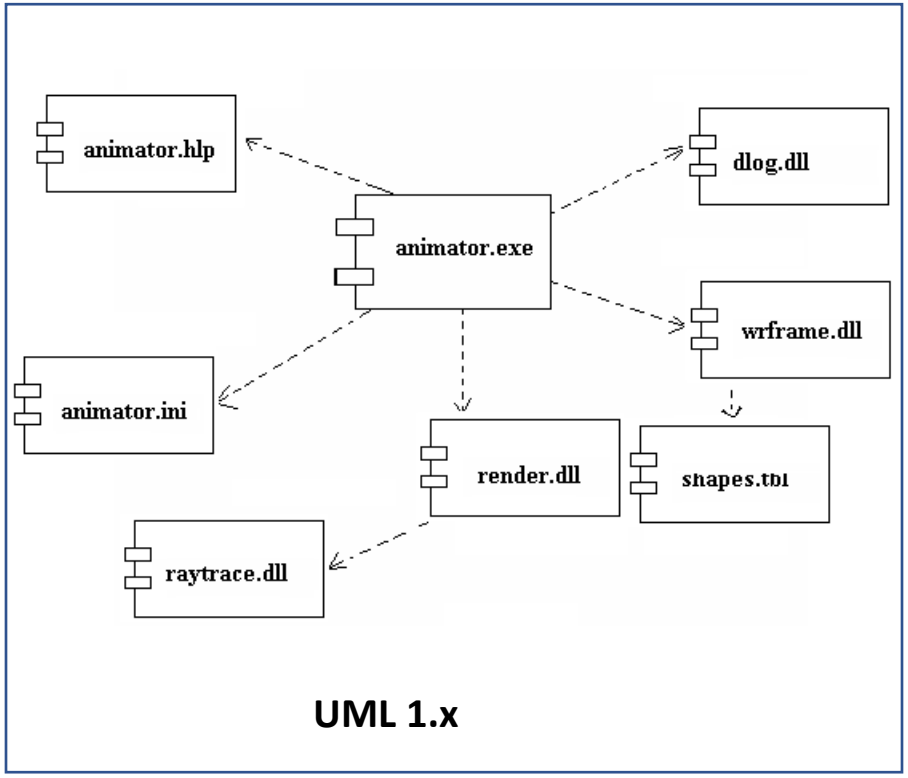
- Componentă bazată pe tranzacţii.

«service»

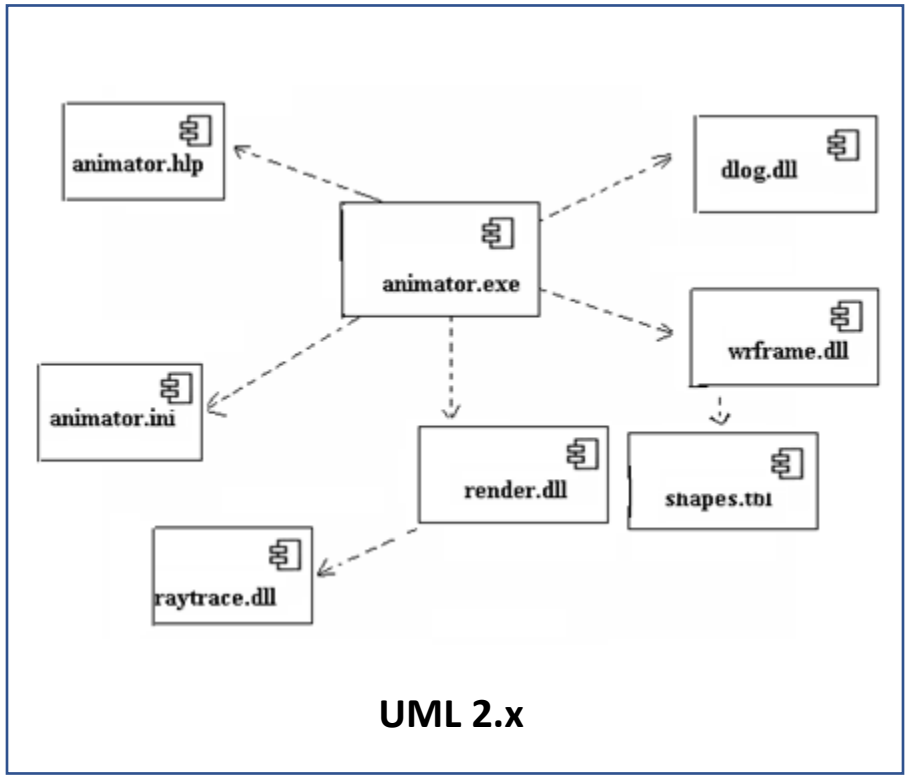
- Componentă funcţională, fără stare.

Diagrame de componente

- Redau relațiile structurale dintre componentele software ale unui sistem.



UML 1.x

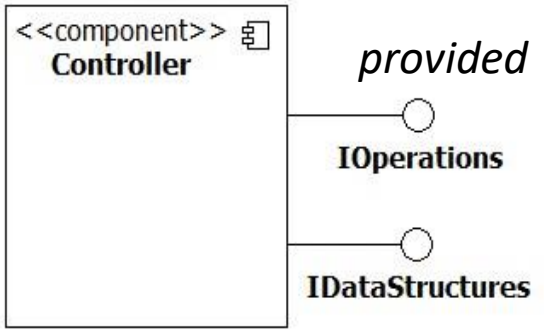


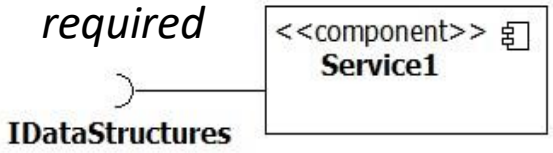
UML 2.x

Interfeţe furnizate şi interfeţe necesare

O interfaţă **furnizată** (*provided*) de o componentă poate fi:

- realizată (implementată) de componenta însăşi, sau
- realizată de alte componente, care *realizează* componenta.





Componenta Service1 utilizează (necesită) serviciile oferite de interfața IDataStructures

Componenta ***Controller*** furnizeaza interfețele ***Ioperations*** si ***IDataStructures***.

O interfaţă **necesară** (*required*) unei componente este una care conţine funcţii:

- necesare implementării componentei, sau

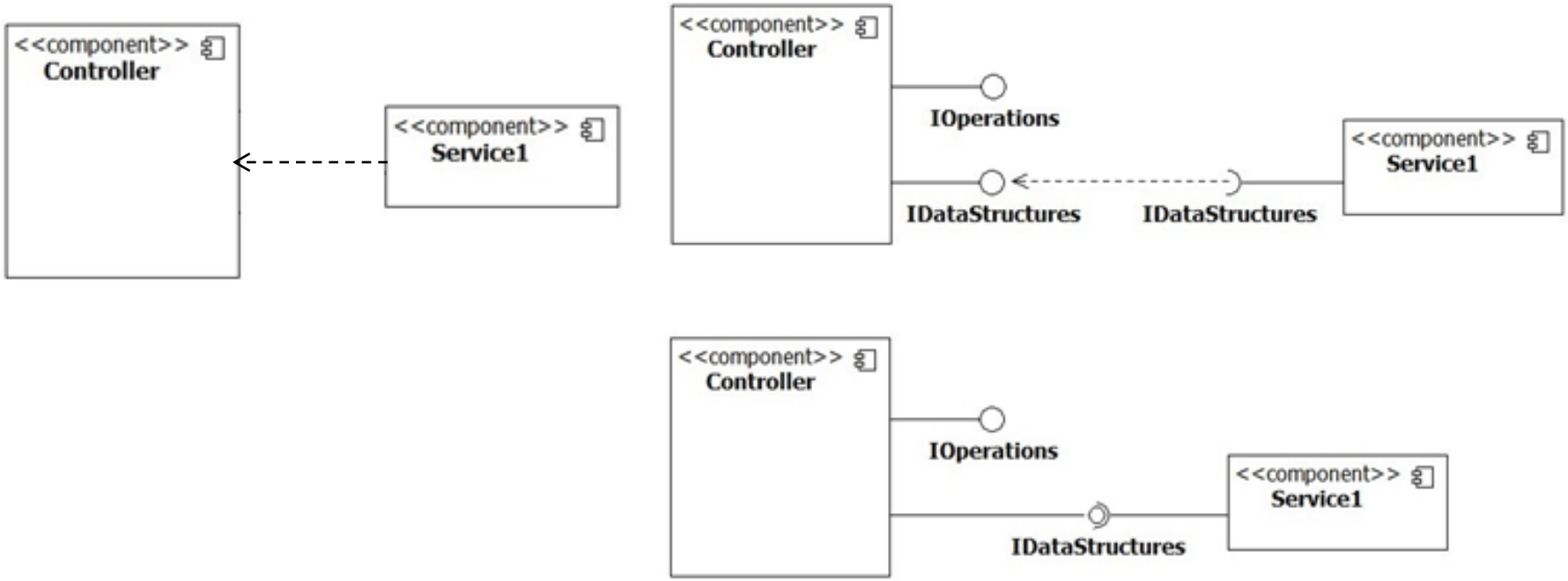
- necesare entităţilor care *realizează* componenta.

Relaţii între componente (1)

Dependența

- O componentă care necesită o interfaţă *depinde* de componenta care implementează interfaţa respectivă.

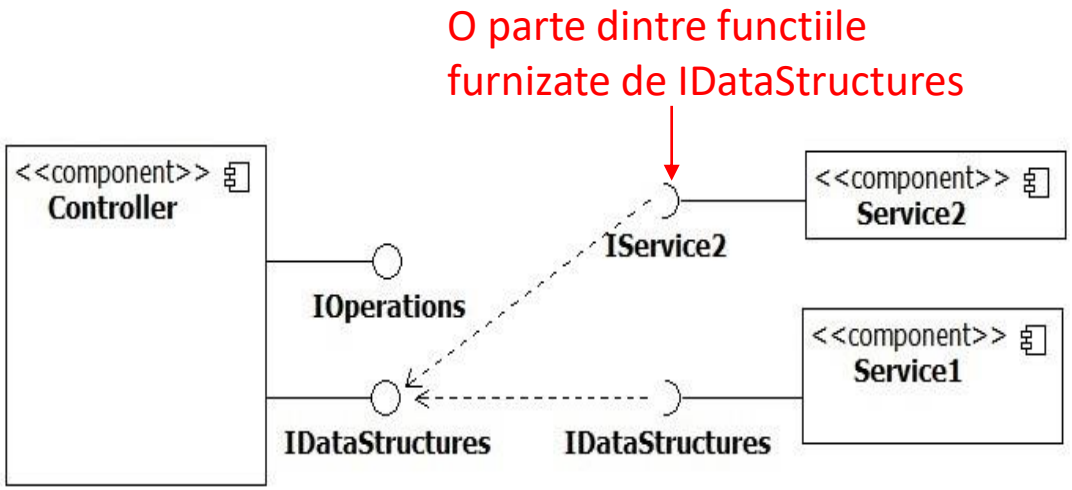
Reprezentări pentru relația de dependenţă



Relaţii între componente (2)

Dependența (cont)

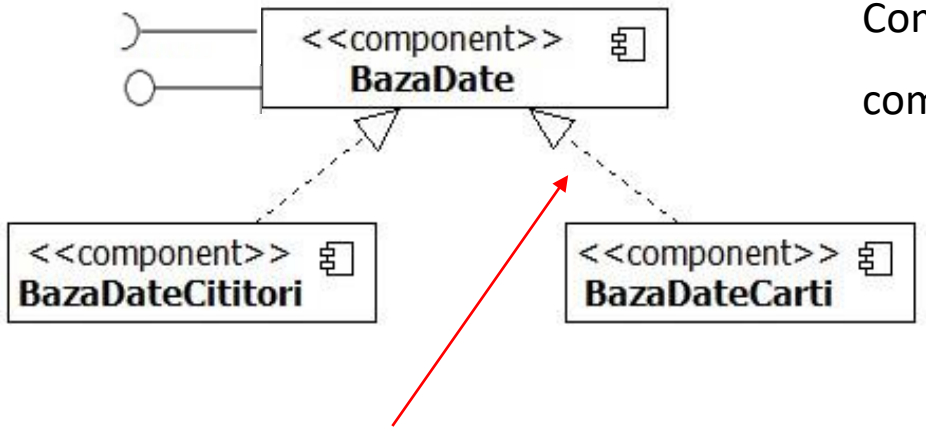
Daca interfaţa necesară unei componente conţine un subset dintre funcţiile furnizate de o interfaţă, numele interfeţei necesare poate fi diferit de cel al interfeţei care furnizează funcţiile.



Relaţii între componente (3)

Realizarea

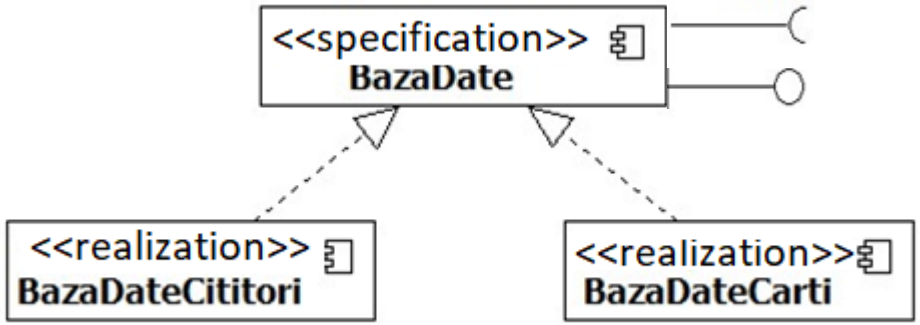
- **O componentă poate fi doar o specificaţie a unui comportament**, prin intermediul interfeţelor furnizate şi necesare.
- Comportamentul poate fi realizat (implementat) de alte componente, caz în care între componenta specificaţie şi componentele care o realizează există o **relaţie de realizare**.



Reprezentarea relaţiei de realizare

Componenta **BazaDate** este realizata de componentele **BazaDateCititori** si **BazaDateCarti.**

Se pot folosi adnotari:



Relaţii între componente (4)

«specification»

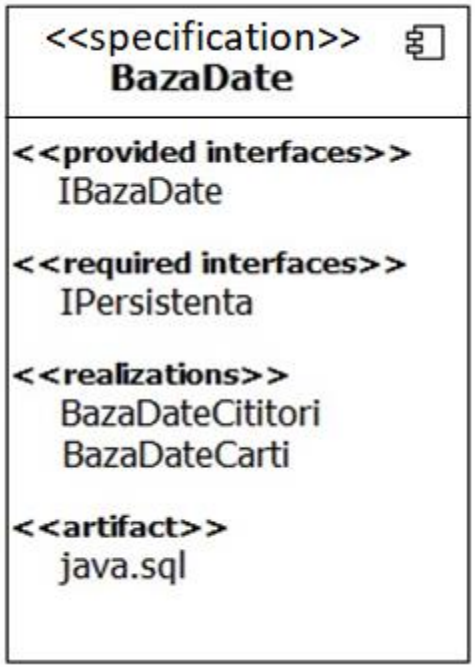
- Se foloseşte pentru a adnota o componentă care reprezintă numai o specificaţie a unui comportament, prin interfeţele furnizate şi cele necesare. Definiţia sa nu precizează entităţile care realizează specificaţia. Acestea pot fi ataşate componentei de tip specificaţie printr-o *relaţie de realizare*.
- O componentă specificație este o abstractie a unui comportament care poate avea mai multe realizari (implementari)

«realization»

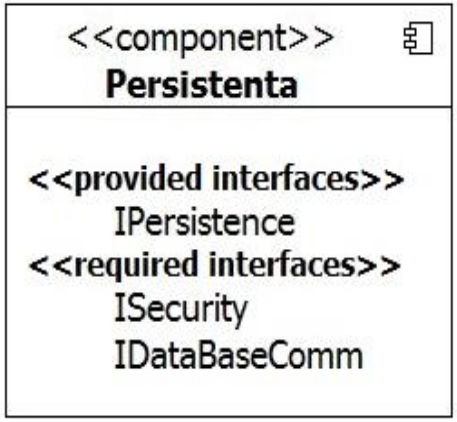
- Se foloseşte pentru a adnota o componentă care implementează comportarea specificată printr-o altă componentă.
- Stereotipurile «specification» şi «realization» sunt utilizate pentru a modela componente cu definiţii separate pentru specificaţie şi realizare, unde o specificaţie poate fi realizata de mai multe componente.

Diferite vederi ale unei componente

Vederea "black box" – prin intermediul simbolurilor interfeţelor furnizate/necesare:



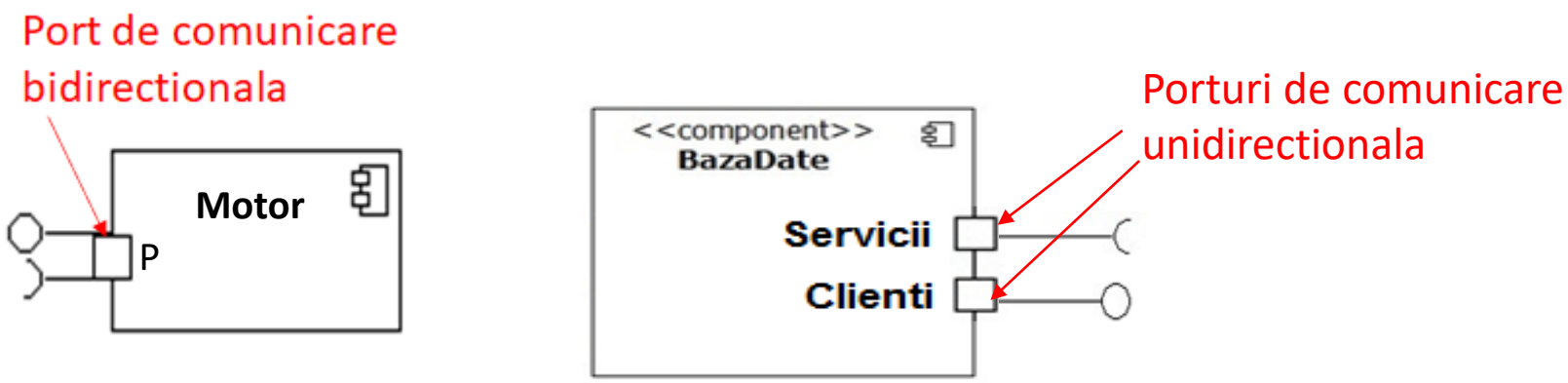
Vederea "white box":



Componenta BazaDate este reprezentată fizic (la executia sistemului) de *artefactul* java.sql.

Porturi si conectori (1)

- **Un port** desemnează un **punct de comunicare între o componentă și mediul său extern.**
- **Un port este asociat cu una sau mai multe interfete.**
- O componenta poate fi utilizata in orice context care satisface constrangerile reprezentate prin porturile sale.
- Un port are asociat un nume și se reprezintă ca un dreptunghi plasat pe una dintre laturile dreptunghiului care încadrează componenta.



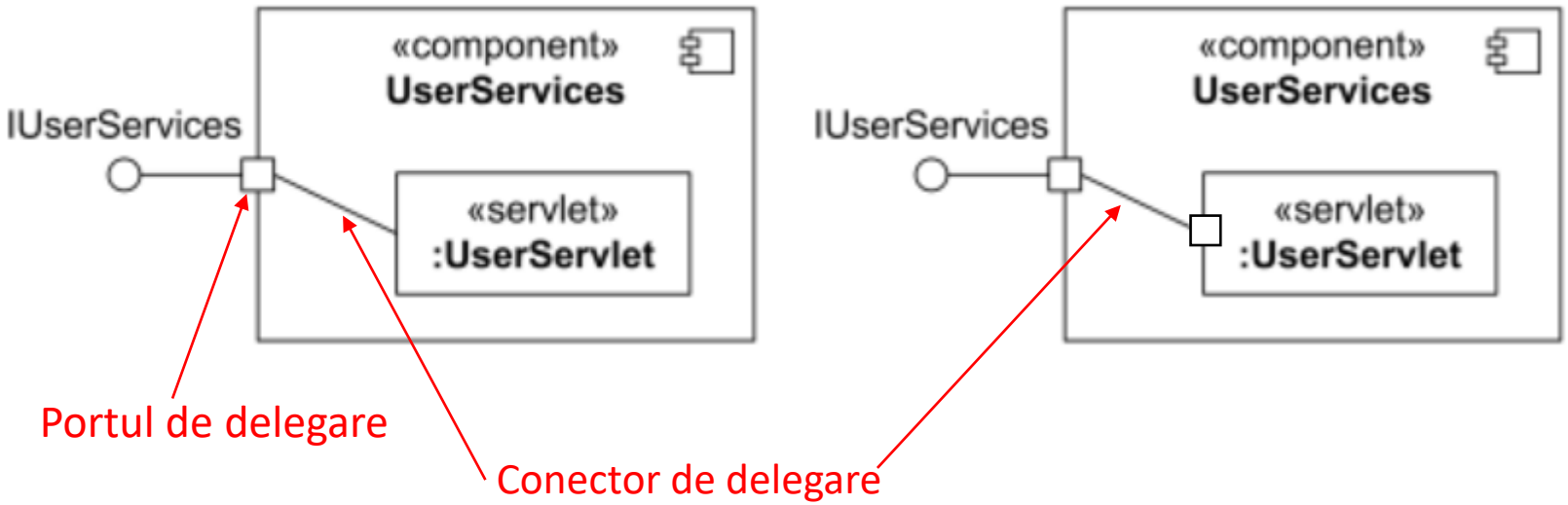
Porturi si conectori (2)

- **Un conector** este **o legătură care reprezintă comunicarea la momentul executiei dintre părti interne ale unei componente sau dintre o parte internă si mediul extern** .
- O instanta a unei componente: entitate prin care este reprezentata fizic componenta la momentul executiei.
- Conectorii sunt de două tipuri:

  - **Conectori de delegare**
    - Leaga un port al unei componente cu o instanta a unei subcomponente
    - Reprezinta transmiterea evenimentelor din exteriorul componentei în interiorul său şi invers, printr-un port
    - «Deleaga» comportamentul extern reprezentat printr-un port unei părți interne
  - **Conectori de asamblare**
    - reprezinta comunicarea dintre partile interne ale unei componete

Porturi si conectori (3)

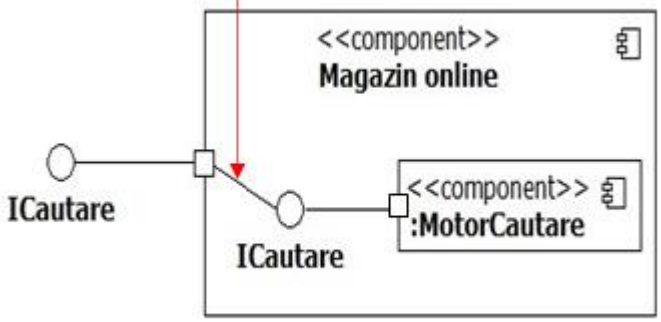
- Un conector de delegare leaga portul de delegare de partea interna care implementeaza functionalitatea portului de delegare.



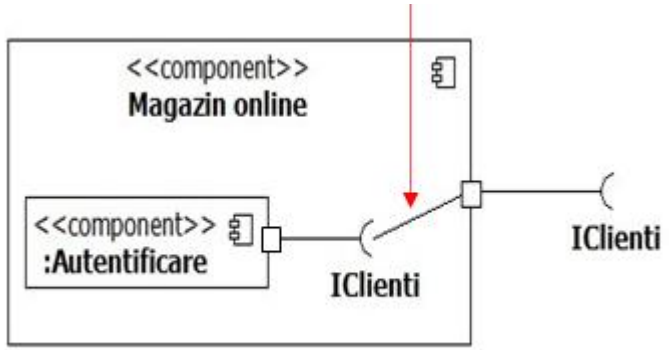
Operatiile expuse de interfata *IUserServices* sunt implementate de o instanta a componentei *:UserServlet*

Porturi si conectori (4)

Conector de delegare de la portul de delegare la un port simplu



Conector de delegare de la un port simplu la portul de delegare



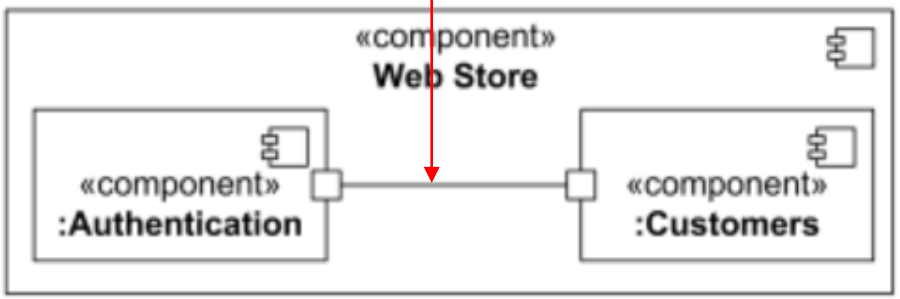
- Mesajele primite de componenta *Magazin online* prin interfața *ICautare* sunt redirecționate către o instanță a subcomponentei *MotorCautare* (care implementeaza operatiile expuse de componenta *Magazin online* prin interfața *ICautare*)

- Mesajele transmise de o instanta a subcomponentei *Autentificare* prin interfața *IClienti* sunt redirecționate către interfata *IClienti* a componentei *Magazin online.*

Porturi si conectori (5)

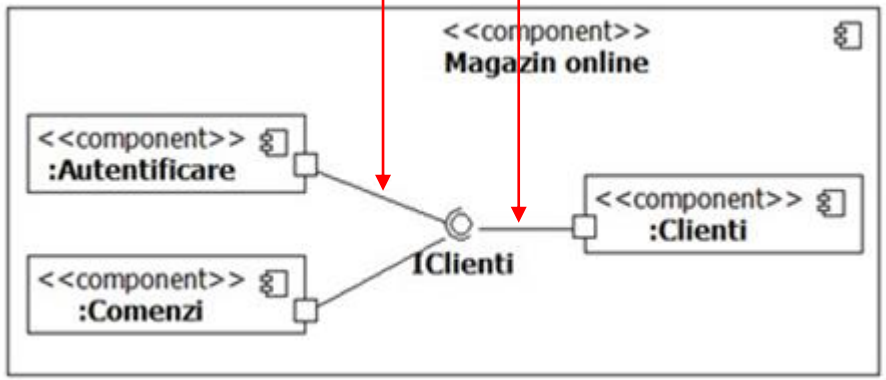
- **Un conector de asamblare** conectează două sau mai multe instanțe ale subcomponentelor unei componente.
- Se reprezinta ca o legatura între 2 porturi sau o legatura între un port şi o interfață.

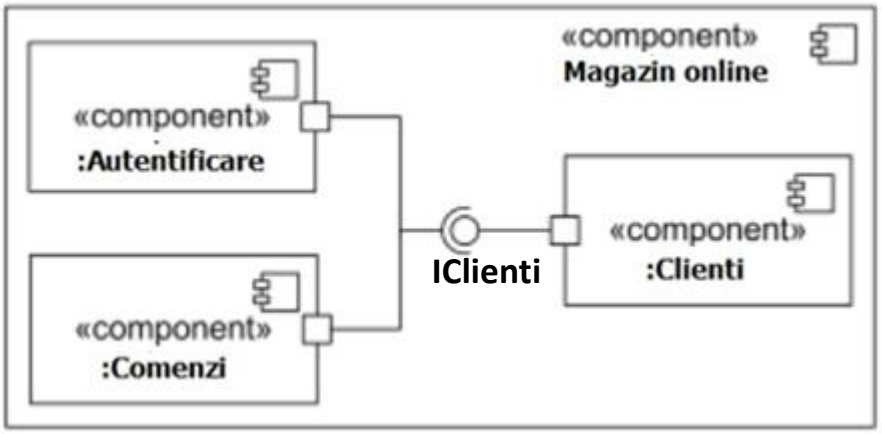
Conector de asamblare



Instantele *:Authentication* şi *:Customers* comunica prin porturile conectate, care pot fi associate cu mai multe interfete

Conectori de asamblare care conecteaza porturi simple

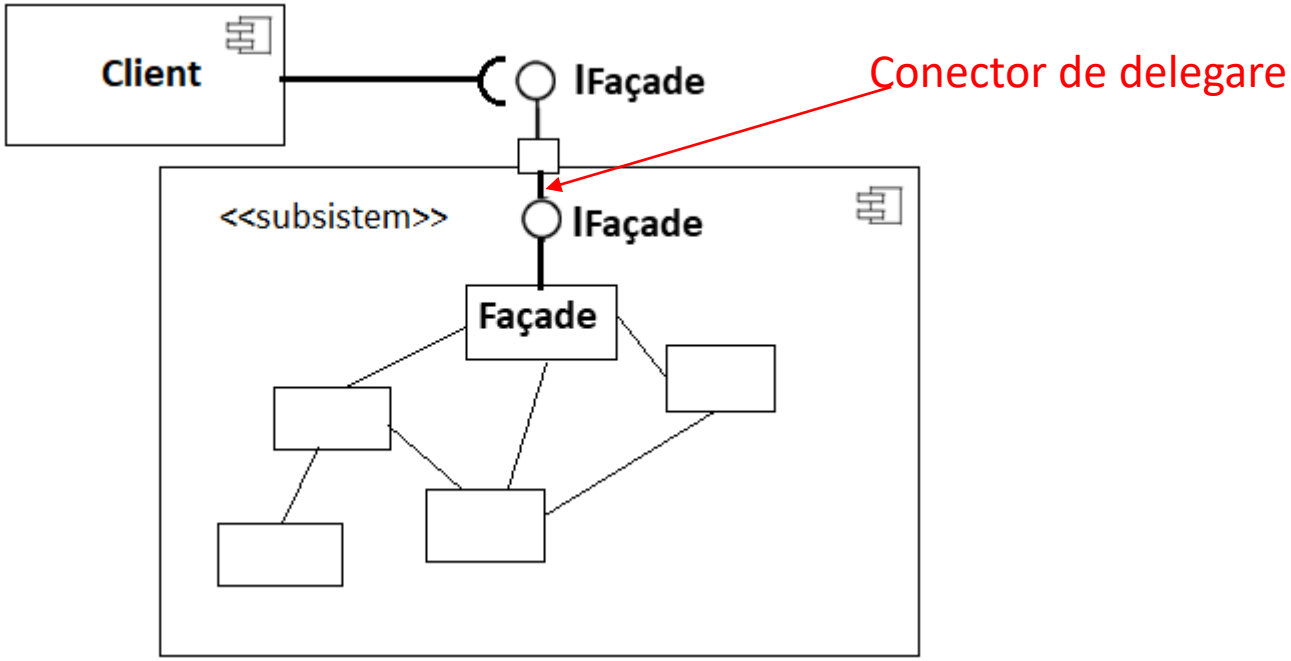




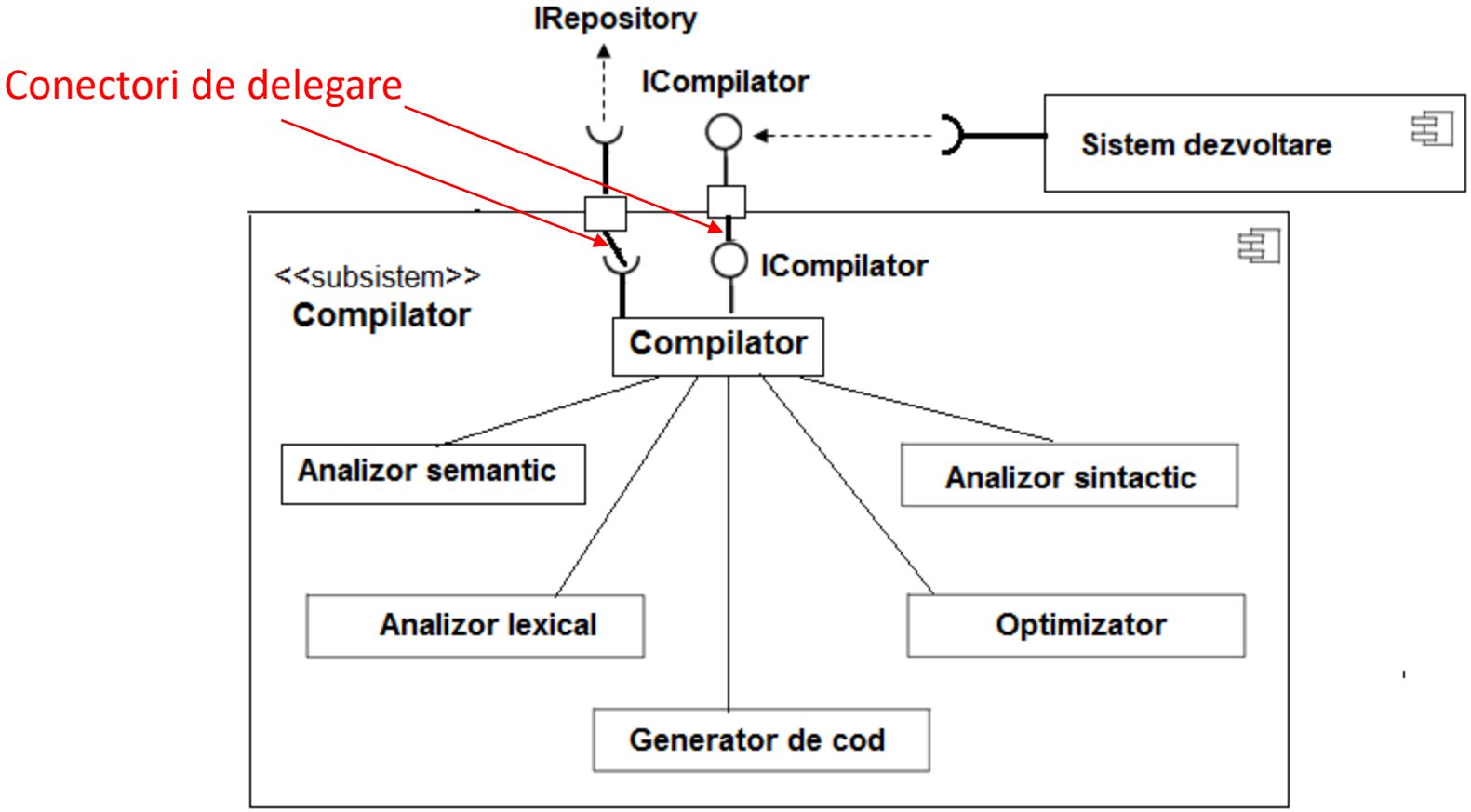
Porturi si conectori (6)

❖ **Sablonul de proiectare Façade** ofera un mecanism adecvat pentru gruparea claselor în subsisteme şi reducerea cuplarii între subsisteme: interfata subsistemului este implementata de o clasa **Façade** care deleaga cererile clientilor catre obiectele claselor subsistemului.

- Subsistemul este slab cuplat cu celelalte subsisteme
- Modificarea subsistemului nu afecteaza subsistemele care utilizeaza I**Façade**



Porturi si conectori (7)



Subsistemul **Compilator** implementeaza interfata I**Compilator** si utilizeaza interfata **IRepository.**

Clasa “faţadă” **Compilator** deleaga:

- cererile de servicii adresate prin interfata **ICompilator** catre obiectele claselor subsistemului
- cererile obiectelor din clasele subsistemului pentru serviciile oferite de interfata **IRepository**

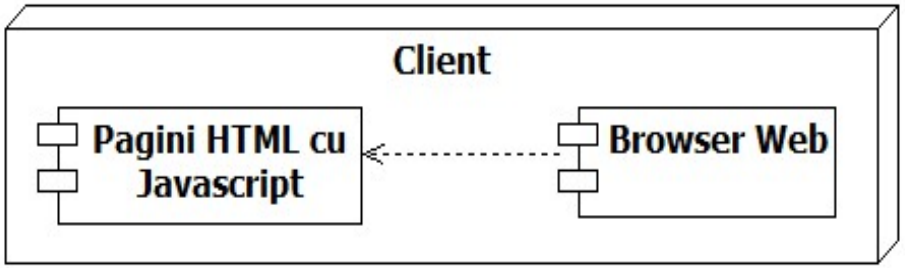
Utilitatea diagramelor de componente

- Furnizează o vedere arhitecturală de nivel înalt a unui sistem software, facilitând luarea deciziilor în privinţa asignării tascurilor.
- Permit modelarea componentelor software de nivel înalt şi a interfeţelor acestora.
- Odata definite interfeţele, se poate repartiza mult mai bine efortul de dezvoltare între subechipele care dezvolta componentele.
- Pe parcursul dezvoltării, interfeţele pot fi modificate pentru a reflecta noi cerinţe, schimbări ale cerinţelor sau ale arhitecturii produsului software.
- Sunt mijloace utile de comunicare între participanţii cheie în proiect şi echipa de implementare.

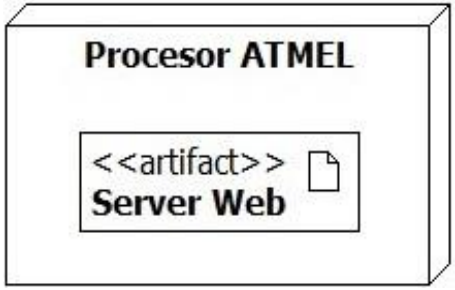
Diagrame de distribuţie (Deployment diagrams)

O diagramă de distribuţie reprezintă arhitectura unui sistem prin *distribuţia artefactelor software* pe echipamentele mediului de operare, reprezentate ca *noduri (cuburi in vedere perspectiva)* în diagramele de distribuţie.

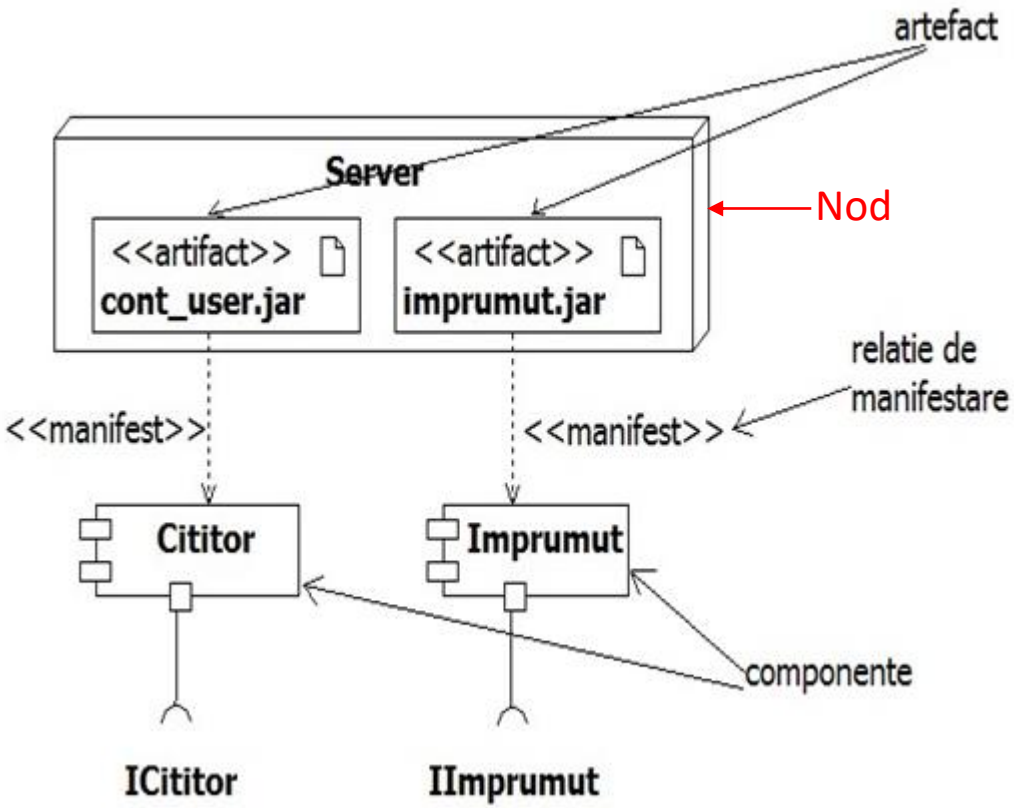
În UML 1.x, componentele sunt distribuite direct pe noduri, ele fiind entităţi fizice din componenţa unui sistem.



În UML 2.x, componentele sunt distribuite indirect pe noduri, prin implementări (*manifestari*) ale lor, numite *artefacte*.



Componente, artefacte, noduri



Cont_user.jar este o implementare a componentei **Cititor**

Imprumut.jar este o implementare a componentei **Imprumut**

Noduri

❖ Nodurile din diagramele de distribuție reprezintă resurse computaționale. Acestea sunt de două tipuri:

- **Nod de tip dispozitiv**, reprezentând un echipament hardware
- **Nod de tip mediu de execuție**, care reprezintă o resursă software ce rulează pe un echipament hardware și este un mediu pentru execuția altor elemente software.

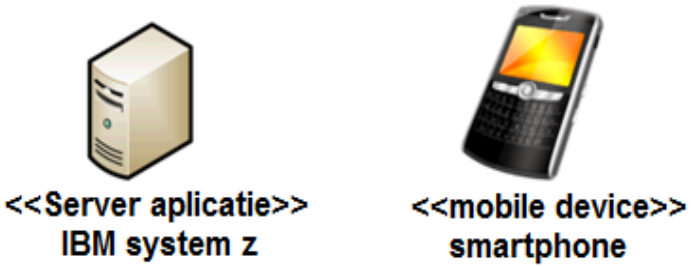
❖ Cele doua tipuri de noduri pot fi marcate prin stereotipurile standard

<<device>> respectiv ***<<executionEnvironment>>***

sau alte stereotipuri nestandard: *<<computer>>, <<cd-rom>>, <<server>>, <<pc>>, <<client>>*, <<application server>>, <<OS>>, <<database system>>, <<web server>>.

Noduri si artefacte

Nodurile de tip <<device>> pot fi reprezentate şi prin simboluri grafice specifice:



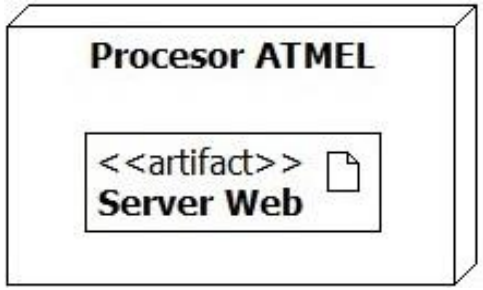
❖ ***Artefactele*** sunt entităţi fizice obținute printr-un proces de dezvoltare software: fişiere executabile, biblioteci, fişiere arhivă, documente, baze de date, etc. Ele **se execută sau sunt stocate pe noduri.**

Reprezentarea artefactelor

Distribuția artefactelor pe noduri

Sunt 2 metode uzuale de a reprezenta distributia unui artefact pe un nod:

- prin amplasarea artefactului pe suprafața nodului



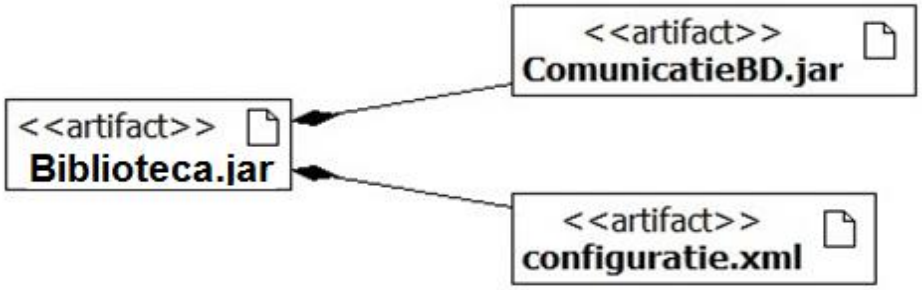
- printr-o relatie de distributie de la artefact catre nod

O alta metoda de reprezentare foloseste un specificator de distributie (a se vedea “UML practic”, Ed MatrixRom, 2014)

Relații între artefacte

❖ Între artefacte se pot stabili relații de asociere şi dependență.

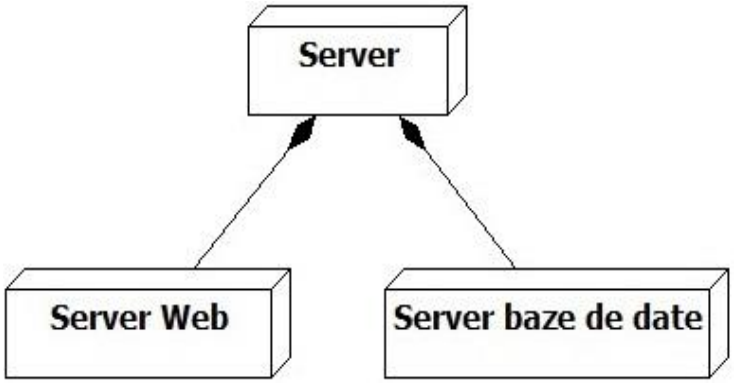
- O ***asociere*** între două artefacte este de tip ***compunere*** sau ***agregare*** – cu aceeasi semnificatie ca relatiile corespunzatoare dintre clase.



- ***Dependenţa***

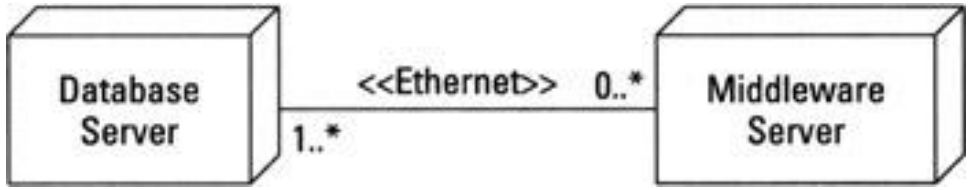
Relații între noduri (1)

Compunerea



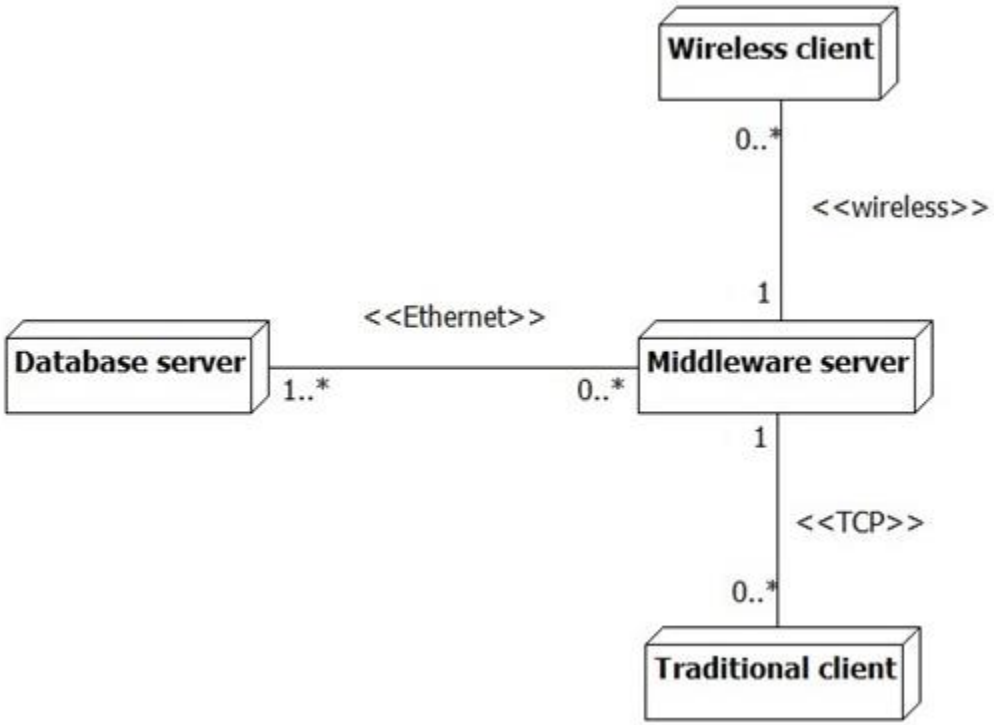
Asocierea

- Reprezintă o cale de comunicare între două noduri și se notează ca şi relaţia de asociere dintre clase.
- Pentru a reprezenta modul în care comunică nodurile se folosesc, ca nume de asociere, stereotipuri care indică modul de comunicare: <<TCP>>, <<UDP>>, <<Ethernet>>, etc.

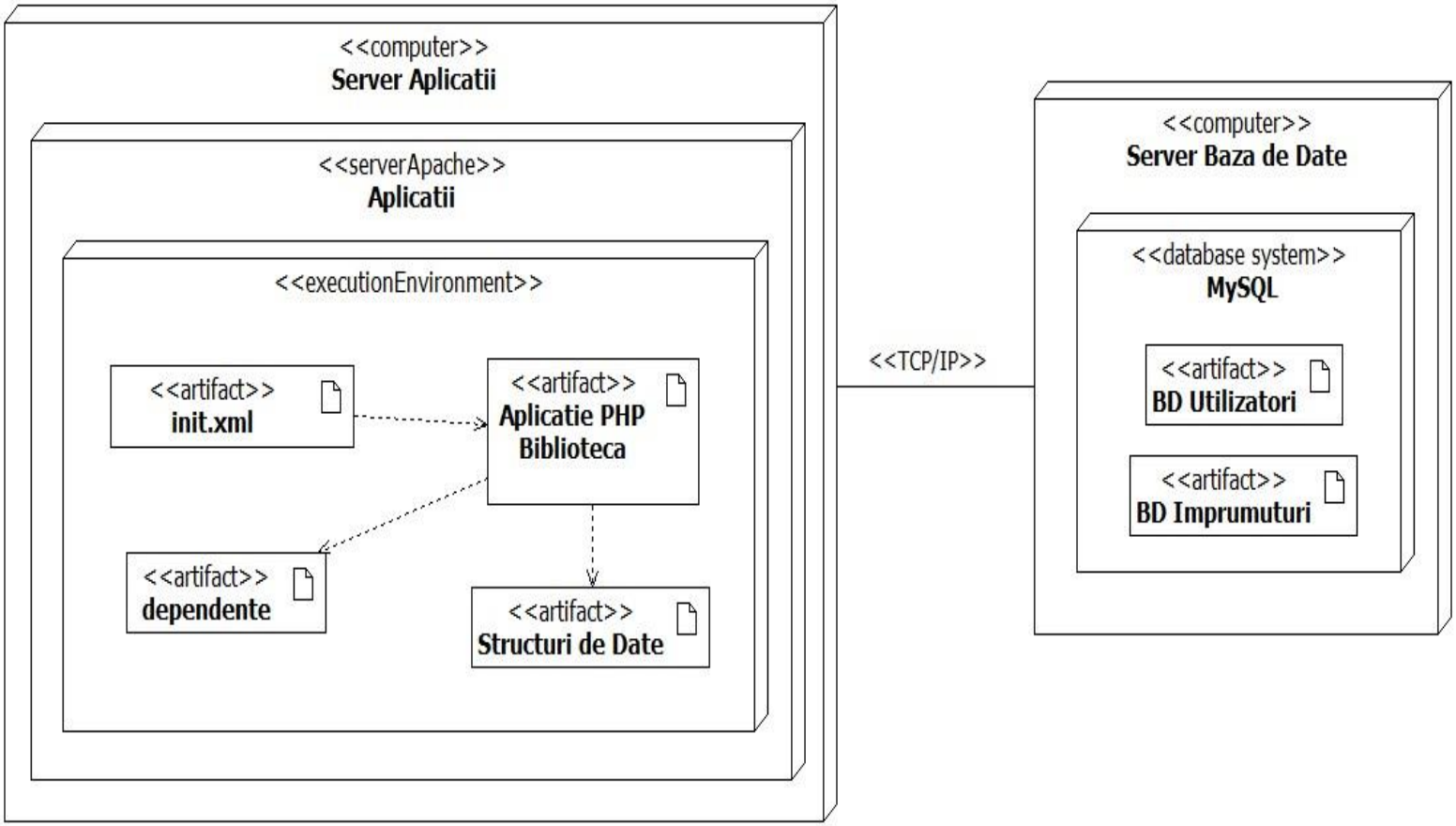


Relatii între noduri (2)

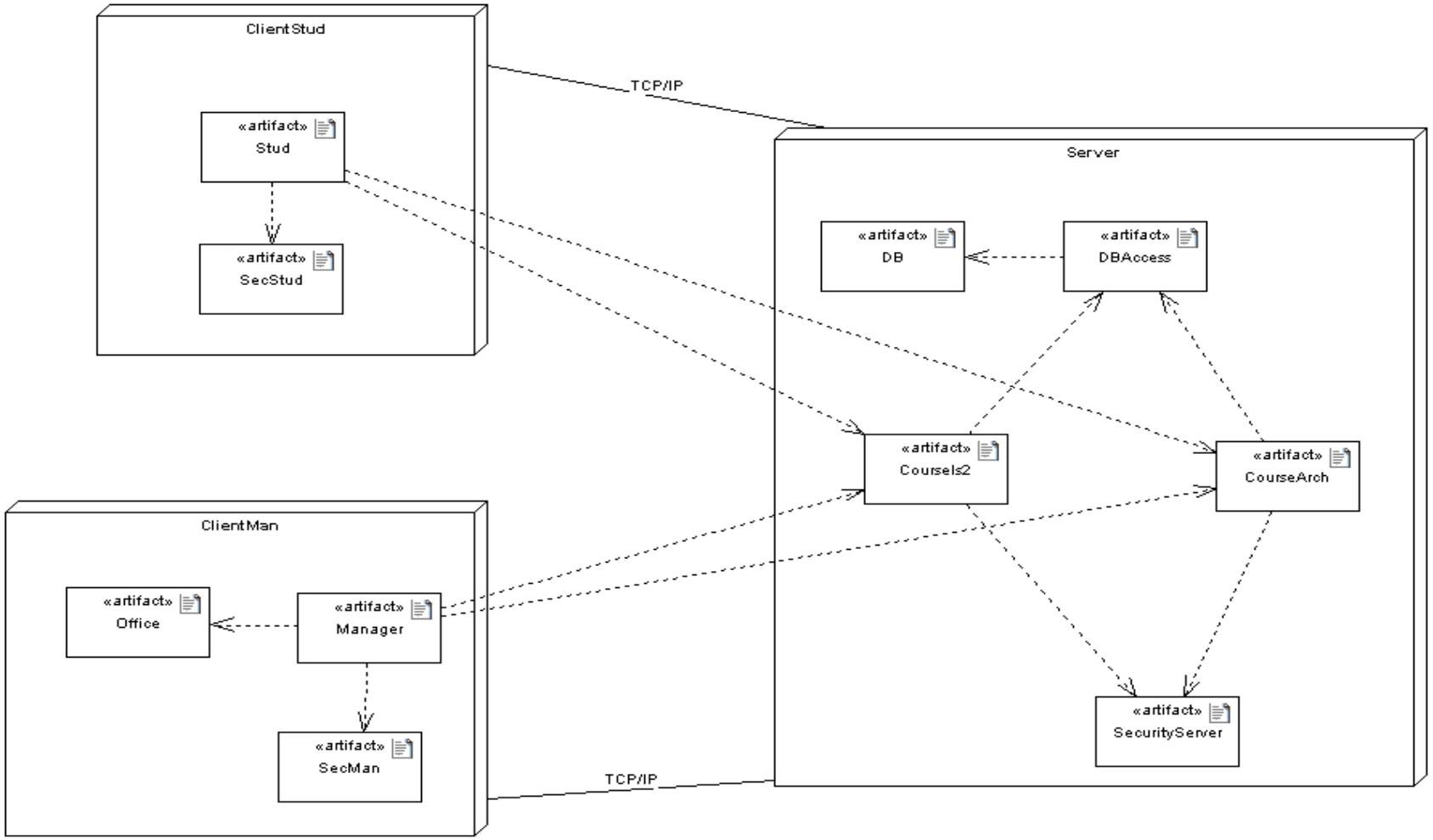
- Capetelor asocierii le pot fi atribuite multiplicităţi, reprezentând numărul de instanțe ce pot fi implicate în comunicare.



Diagramă de distribuție în UML 2.x



Diagramă de distribuție în UML 2.x



Lecturi suplimentare – diagrame de componente si de distributie

https://www.uml-diagrams.org/composite-structure-diagrams/connector.html

https://www.uml-diagrams.org/deployment-diagrams-overview.html

https://www.lucidchart.com/pages/uml-deployment-diagram/#section_5