

# **Centralised    versus    Distributed computing                              Computing**

## **Sisteme Distribuite**

Mihai Zaharia

# **Bucătărie**

- proful --->
- Examenul --> vezi mai jos

# Criterii folosite în evaluarea activității

- **Participarea:** la orele de curs și de laborator:
  - Neparticiparea la mai mult de 50% din laboratoare conduce la refacerea disciplinei.
  - neparticiparea la curs conduce la probleme la examenul teoretic
- **Laboratoare:** pentru a putea lua 10 la laborator trebuie ca studentii să fie capabili la intrebarile asistenților cu privire la conținutul cursului curent (și pentru care a fost creat laboratorul) - 30% - Atentie asistentii nu stau să repredea!! ci numai noteaza corectitudinea raspunsului și trec mai departe
- **Examen final 70%** (este o singura notă) defalcată astfel:
  - Proba de **laborator** – **ELIMINA-TORIE 40 %** cu bilete și două ore maxim la dispoziție. Un subiect din două trebuie să fie îndeplinit integral pentru a se putea nota (min 5).
  - Proba **teoretică** – "**PICĂ-TORIE**"**40%** - test docimologic - conține și întrebări cu caracter practic specifice laboratorului (min 5)
  - **Teme acasă: semi** - "**PICĂ-TORIE**" la majoritatea laboratoarelor 20% (atentie se poate să aveți 5 la lab și teorie și să picăti din motive de teme nepredate)
- În caz de variantă on line (SARS-CoV-II)
  - testul practic - se aleg automat două probleme din pool și se trimit
  - testul teoretic - oral cu întrebări selectate automat din pool

# **Cum este cu înregistrările????**

- No way!!!
- Why?

# **Cum se utilizează cursul ?????**

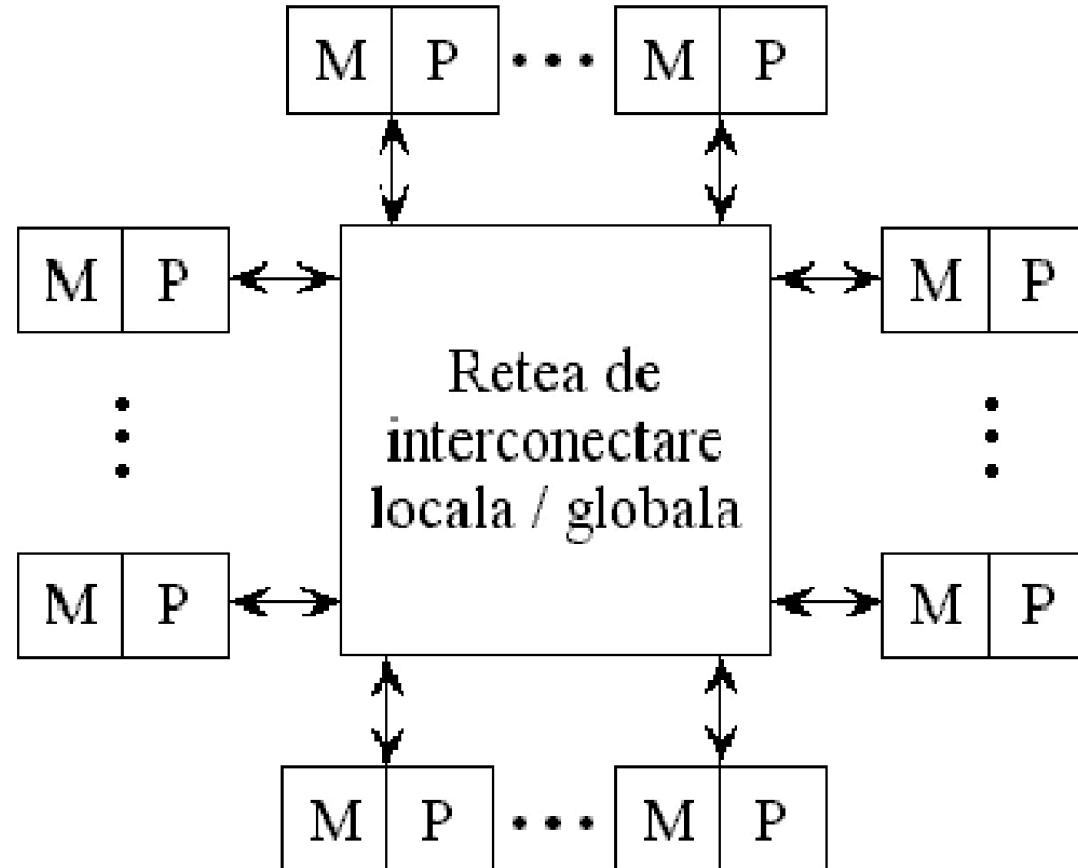
1. se iau notițe! (chiar dacă te doare mâna, neuronul etc)
2. nu este permisă înregistrarea sub nici o formă a titularului - oricare ar fi el - (nu pentru că se supără ci pentru că se încalcă simultan trei legi în vigoare!)
3. te duci acasă și citești despre termenii din notițe și slide-uri până te lămurești apoi te uiți din nou pe slide-uri.
4. dacă mai ramân după atâta citit unii termeni mai neclar există profesor și asistenți
5. prezentarea la laborator fără efectuarea pașilor anteriori anulează 80% din efectul combinat curs+lab asupra pregătirii voastre!

# **Termeni și concepte**

sisteme strâns cuplate

sisteme slab cuplate

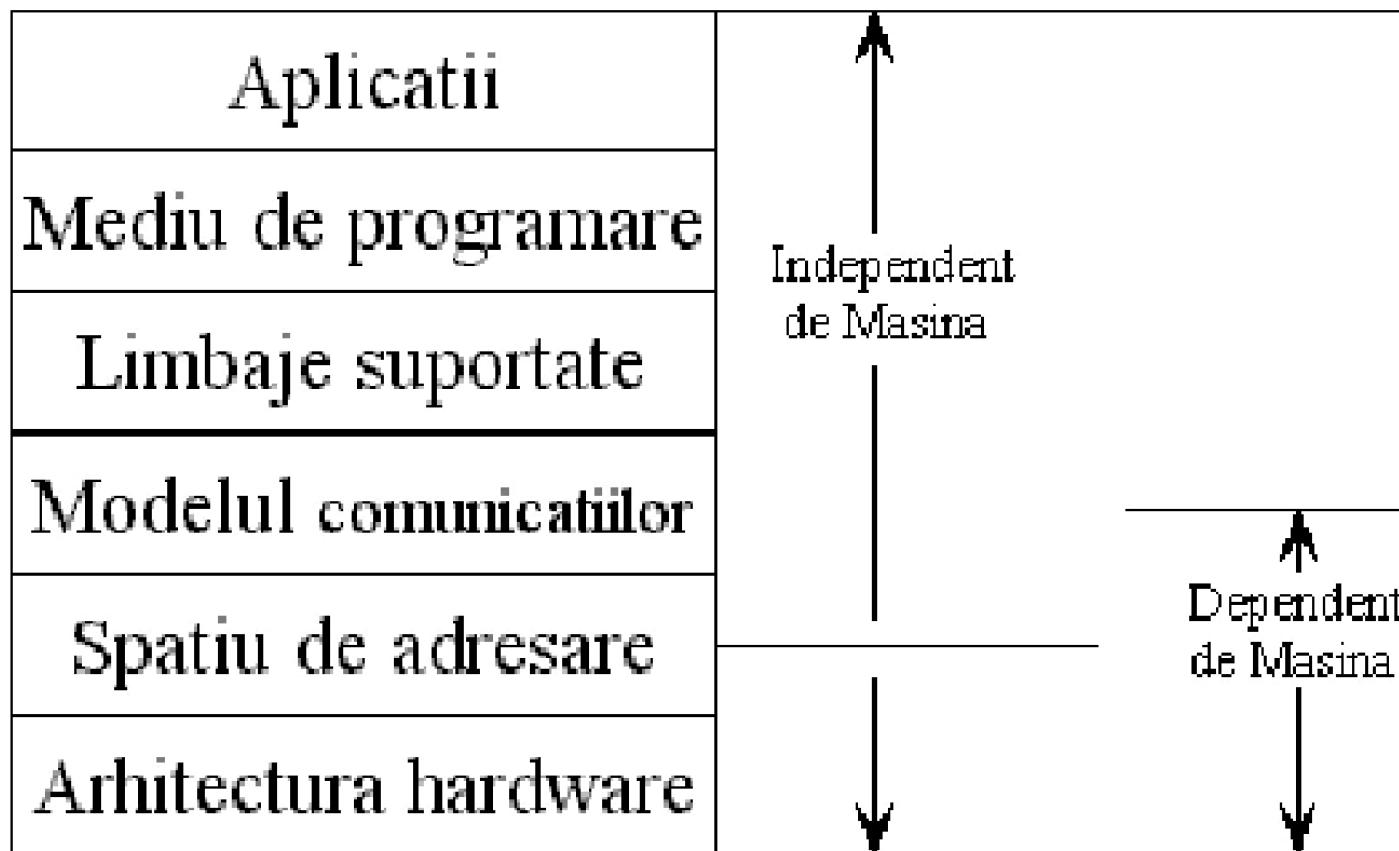
# Modelul sistemelor HW slab cuplate



M = Memorie; P = Procesor

*Multicalcătoare cu memorie distribuită*

# **Modelul Li al unui sistem distribuit(hw+sw)**

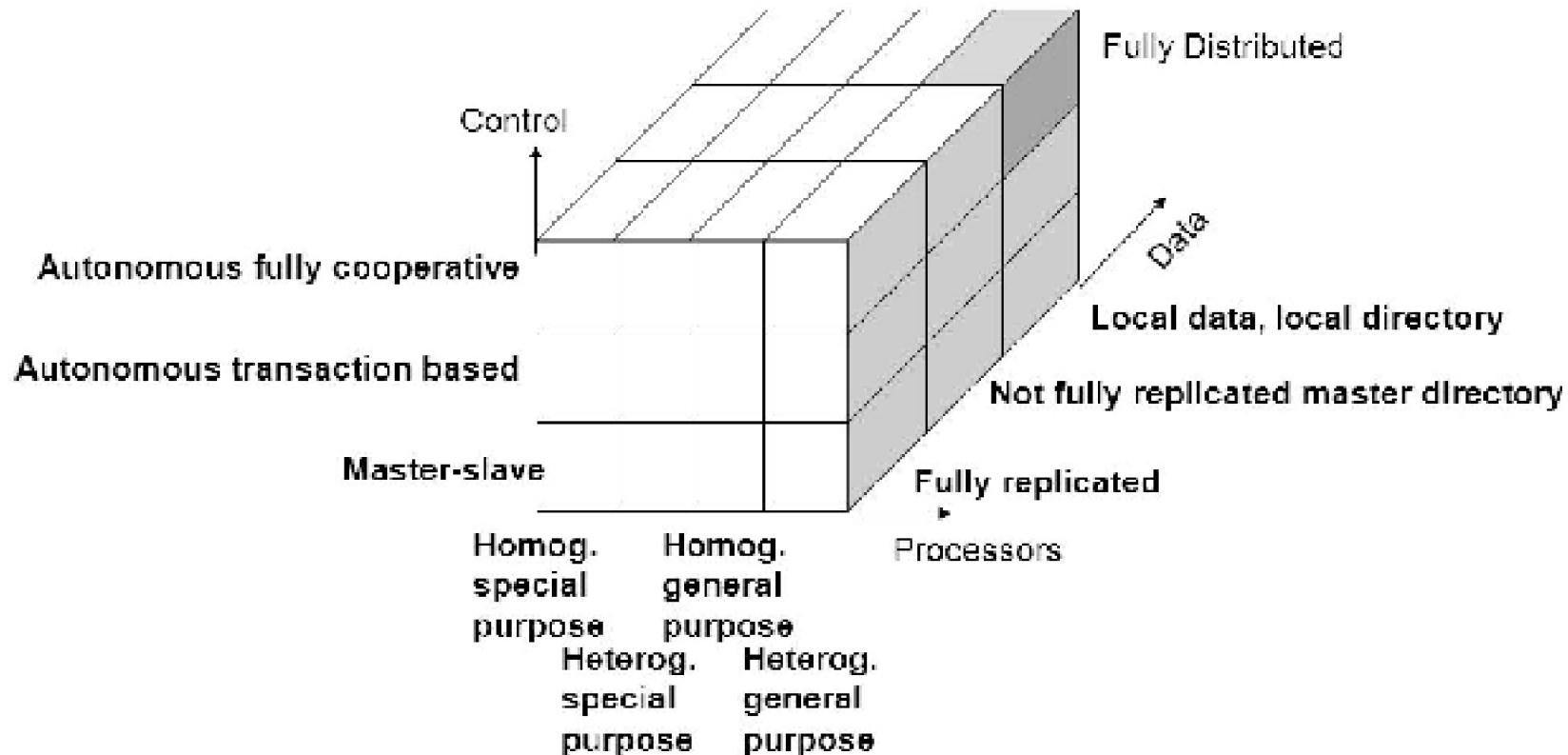


# **Termeni și concepte**

arhitecturi software strâns  
cuplate - monolit

arhitecturi software slab  
cuplate -  
servicii/microservicii

# System distribuit Enslow 1978

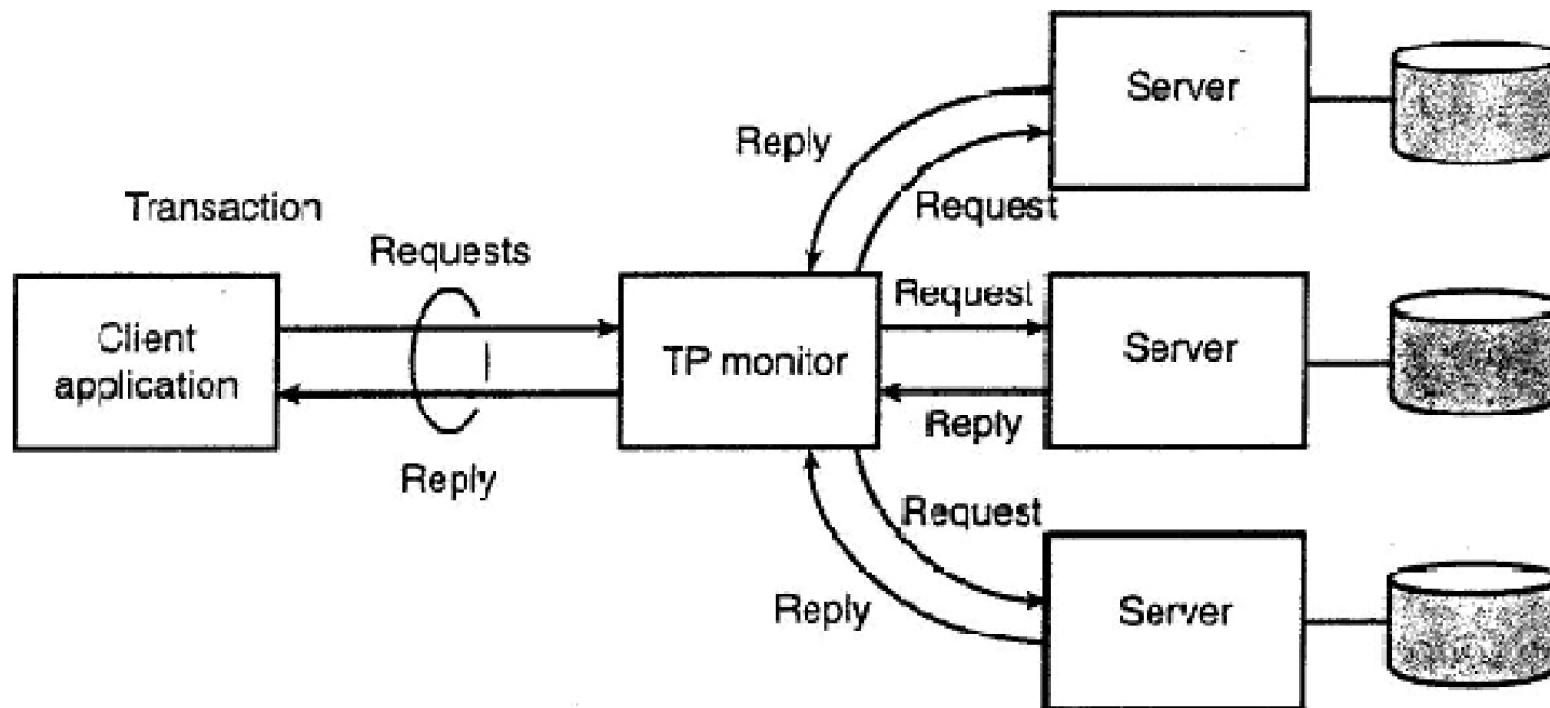


# **Hardware distribuit fizic/geografic**

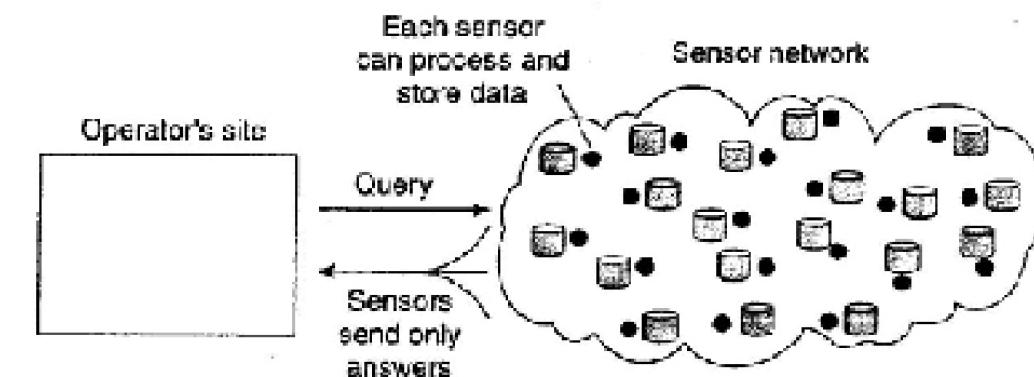
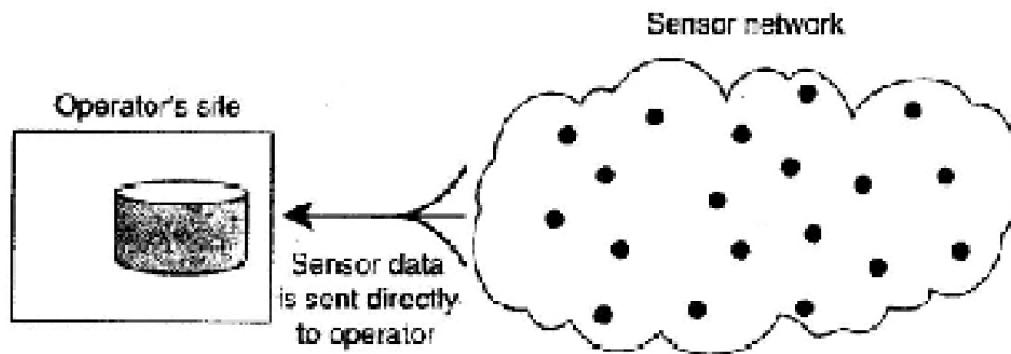
# **Control Distribuit**

# **Datele distribuite**

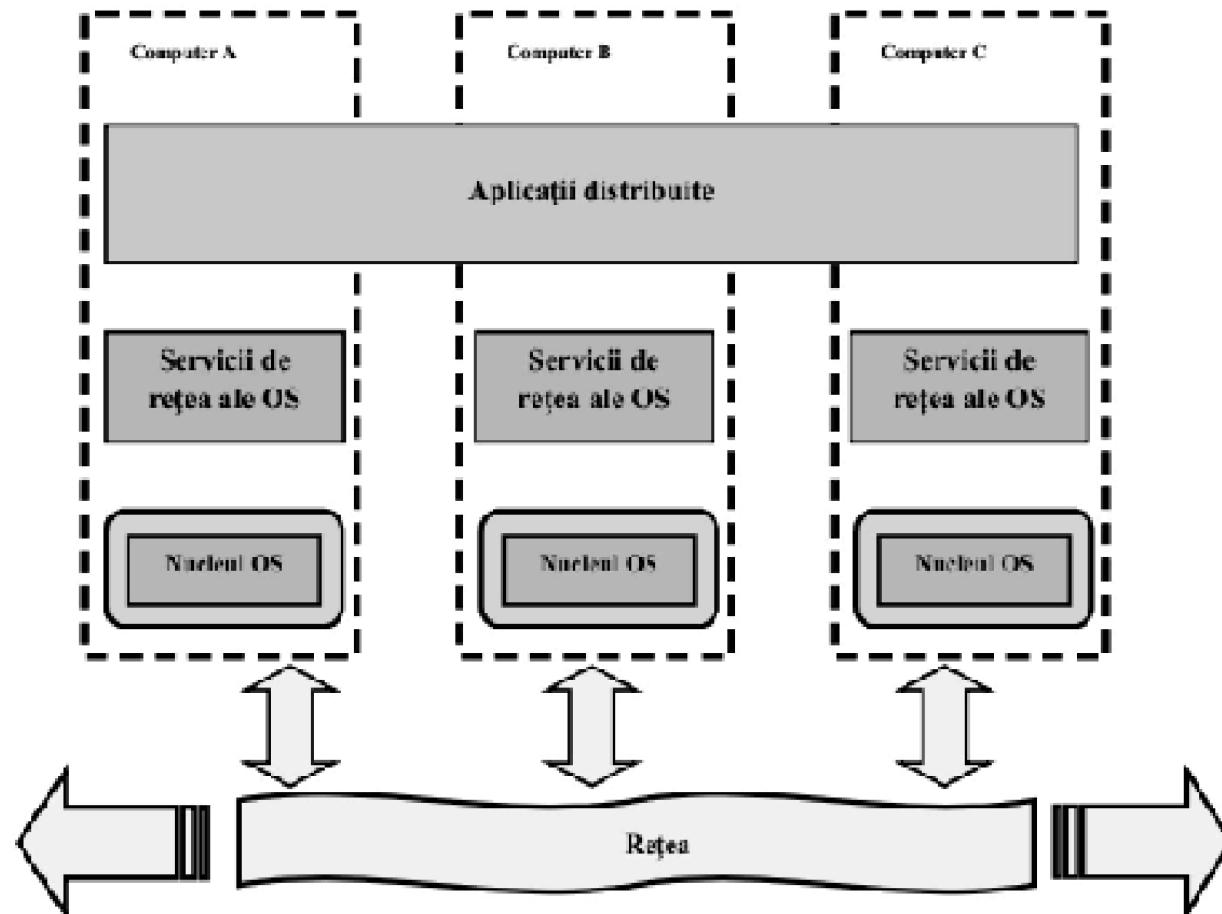
# Exemple



# Exemple

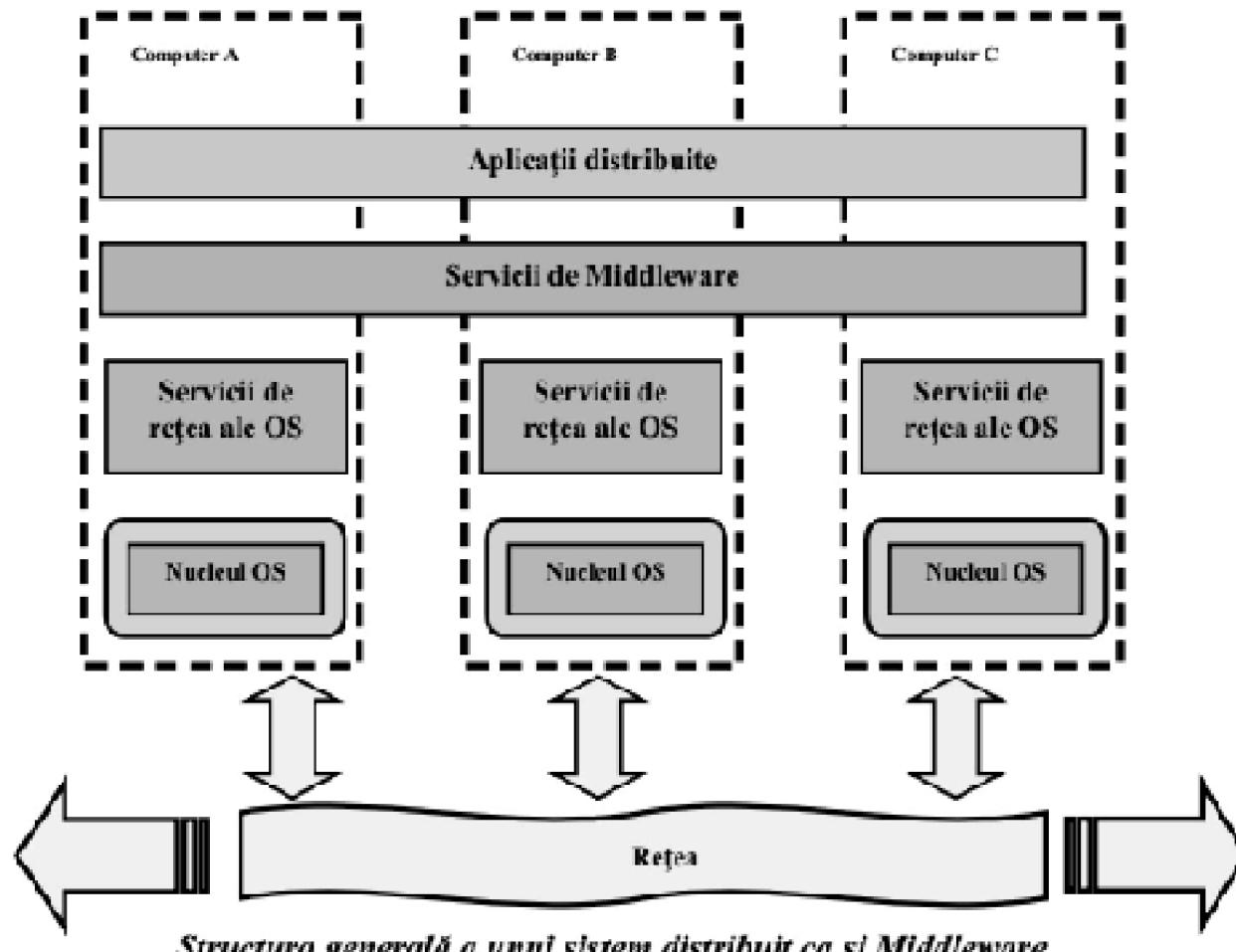


# Ce suport ne oferă sistemele de operare?

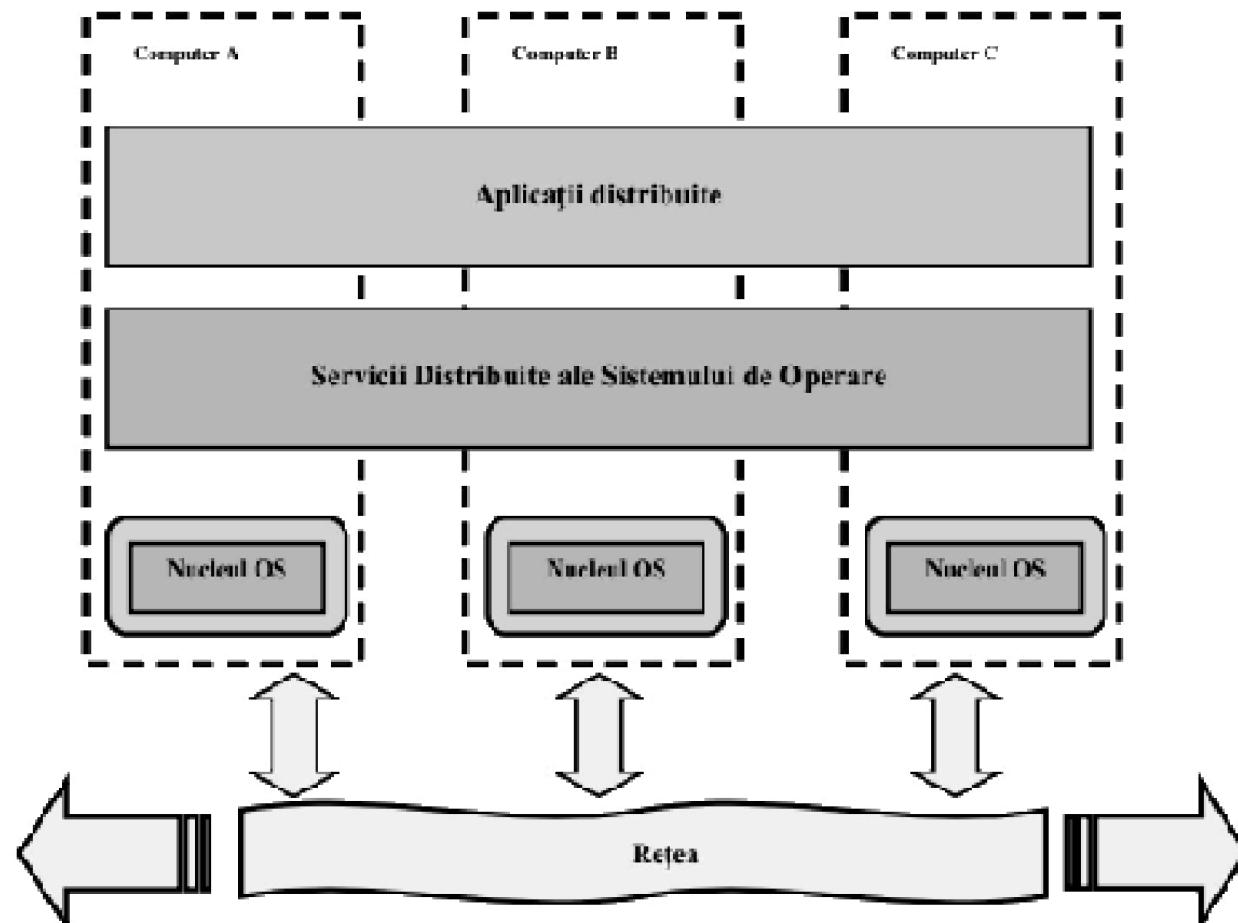


*Strucura generală a unui sistem de operare în rețea*

# Ce suport ne oferă sistemele de operare?

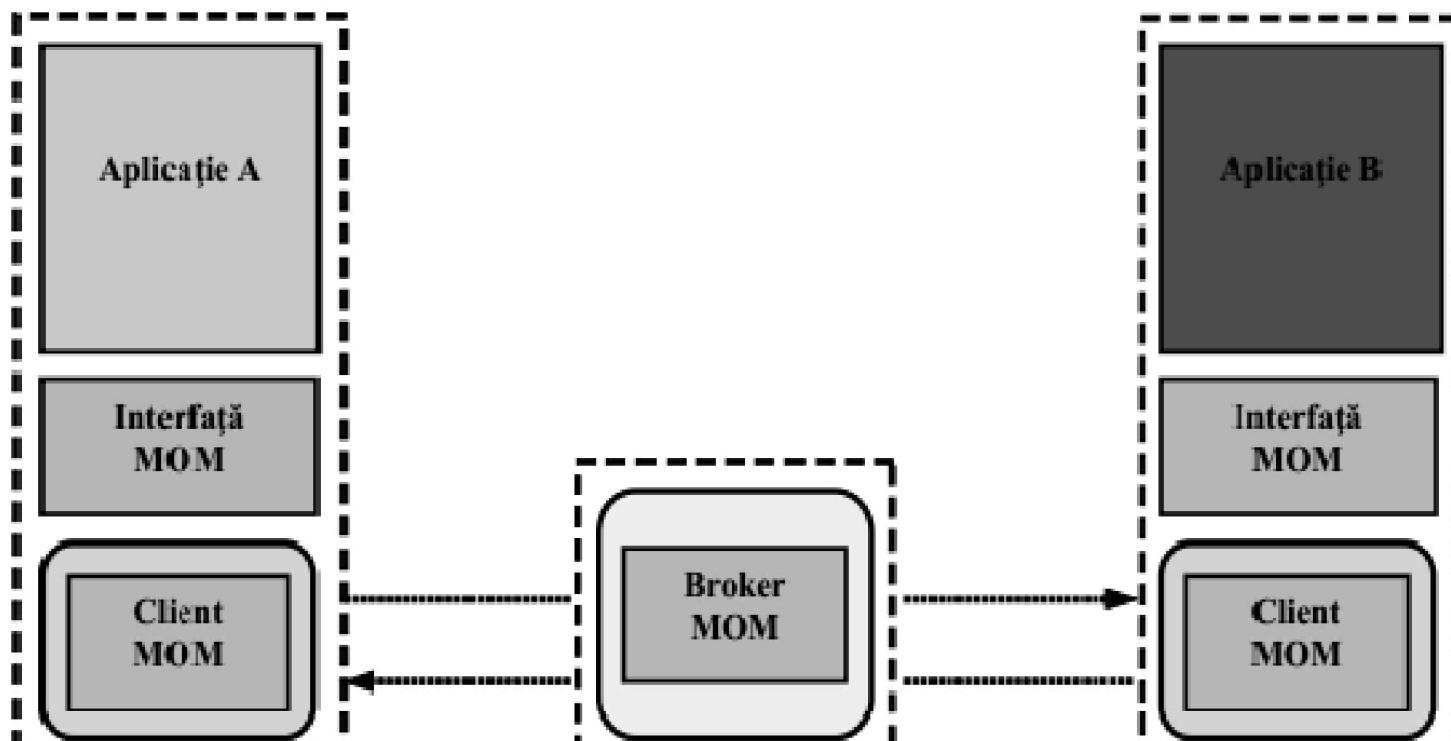


# Ce suport ne oferă sistemele de operare?



*Structura de ansamblu a unui sistem de operare multicompiler*

# Ce suport ne oferă sistemele de operare?



*Comunicarea între aplicații prin intermediul unui middleware orientat pe mesaje*

# **SWOT - SD - Avantaje**

**costuri reduse**

**SWOT - SD - Avantaje**

**modularitate și flexibilitate**

# **SWOT - SD - Avantaje**

- Fiabilitate și integritate**

# **SWOT - SD - Avantaje**

**performanță**

# **SWOT - SD - Dezvantaje**

**Lipsa cunoștințelor despre starea globală**

# **SWOT - SD - Dezvantaje**

**Lipsa unui timp global**

# **SWOT - SD - Dezvantaje**

## **Nedeterminismul**

**SWOT - SD - Dezvantaje**

**comunicațiile**

# **SWOT - SD - Dezvantaje**

**securitatea**

# **Arhitectura multinivel-multistrat**

# **Decuplarea funcționalităților**

# **Controlul invers**

# **Injectarea dependențelor**

**Abordarea clasică**

# **Injectarea dependențelor**

abordare modernă

# **Componente software**

## **Problema OOP**

“Object orientation has failed but component software is succeeding” Udell, 1994

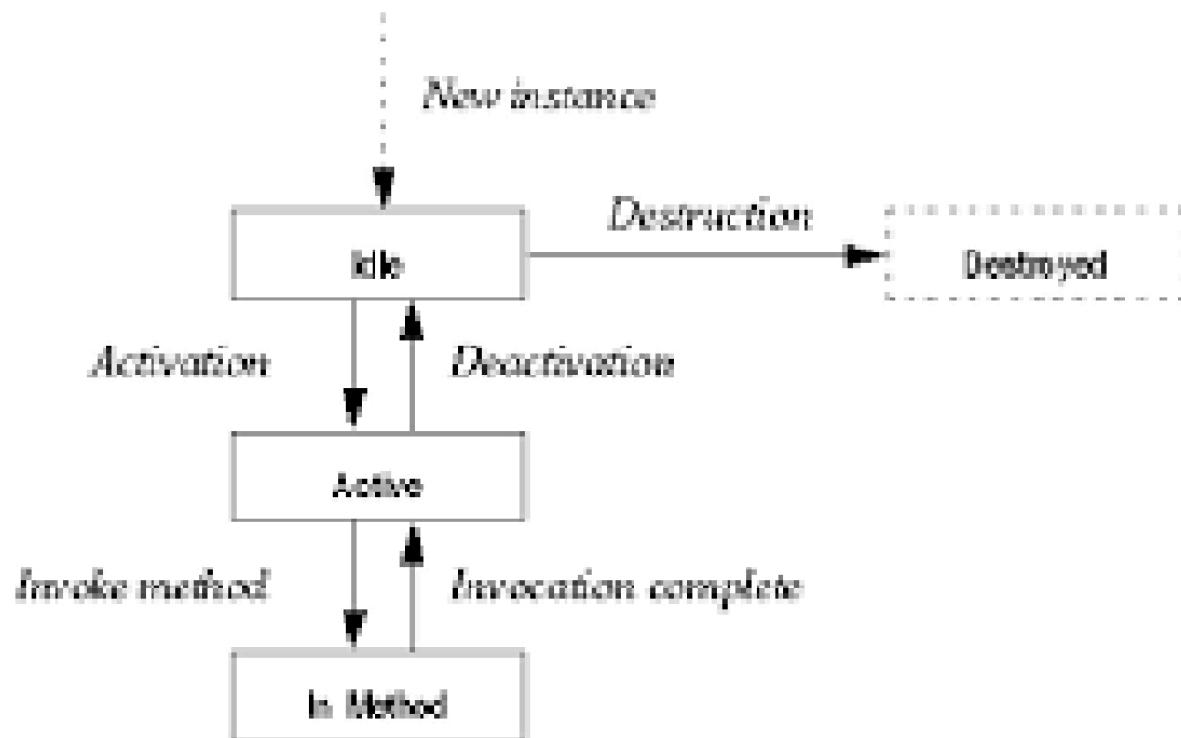
# **Paradigma post OOP: CBSE**

**evolutia posibilă a OO-ler**

# **Procedural vs OOP vs COP**

**Compozabilitatea & Interschimbarea**

# Fazele dezvoltării și utilizării unei componente



# **Tehnologia Java Beans**

# **Bobul de fasole al lui Sun**

```
<jsp:useBean id="studenti"  
            class="lahnieFasoleRosie">  
<jsp:setProperty name="studenti" property="Numele"  
                 value="BautorulCuMobil"/>
```

# **Arhitecturi orientate pe component**

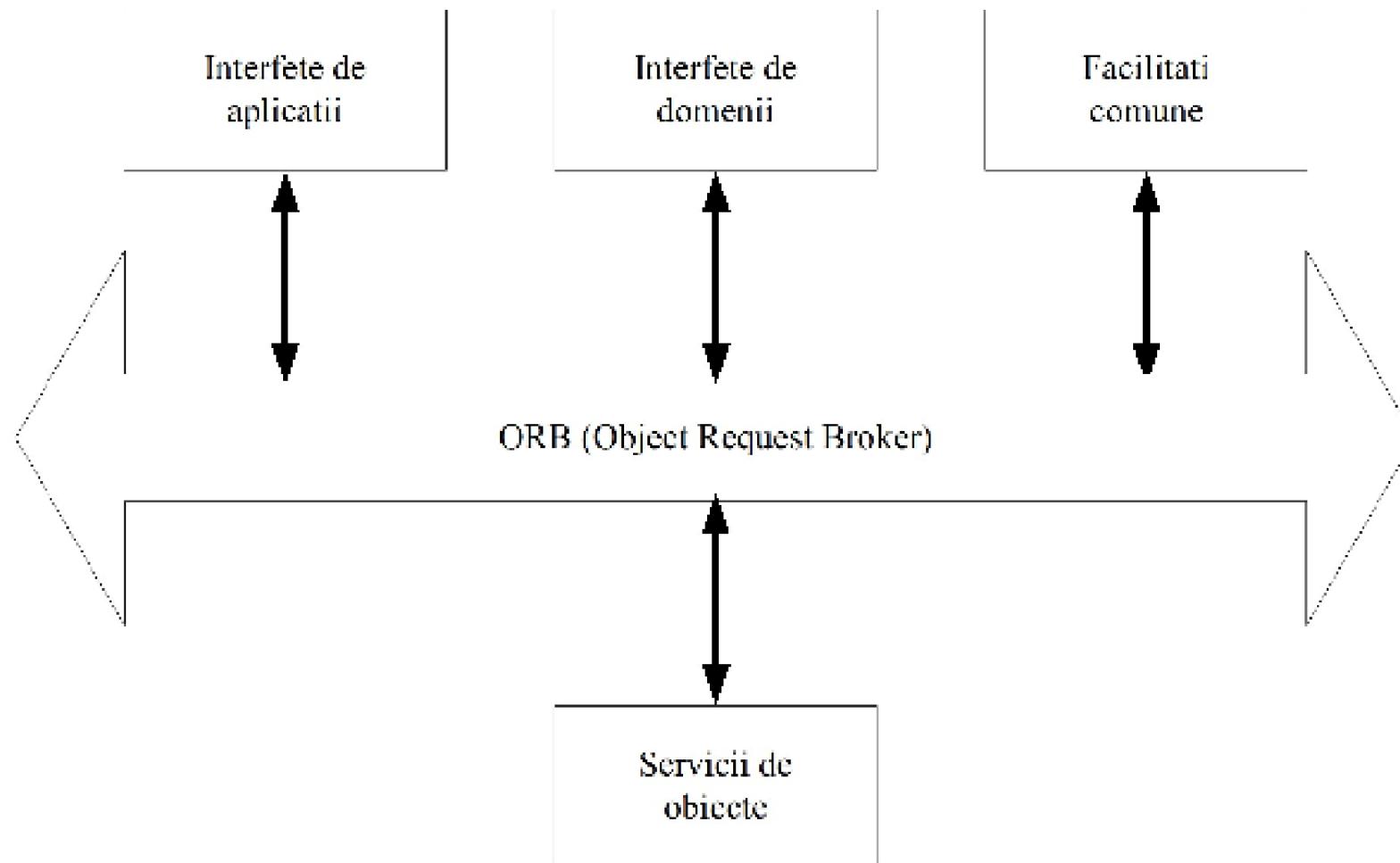
**DCE**

**DCOM**

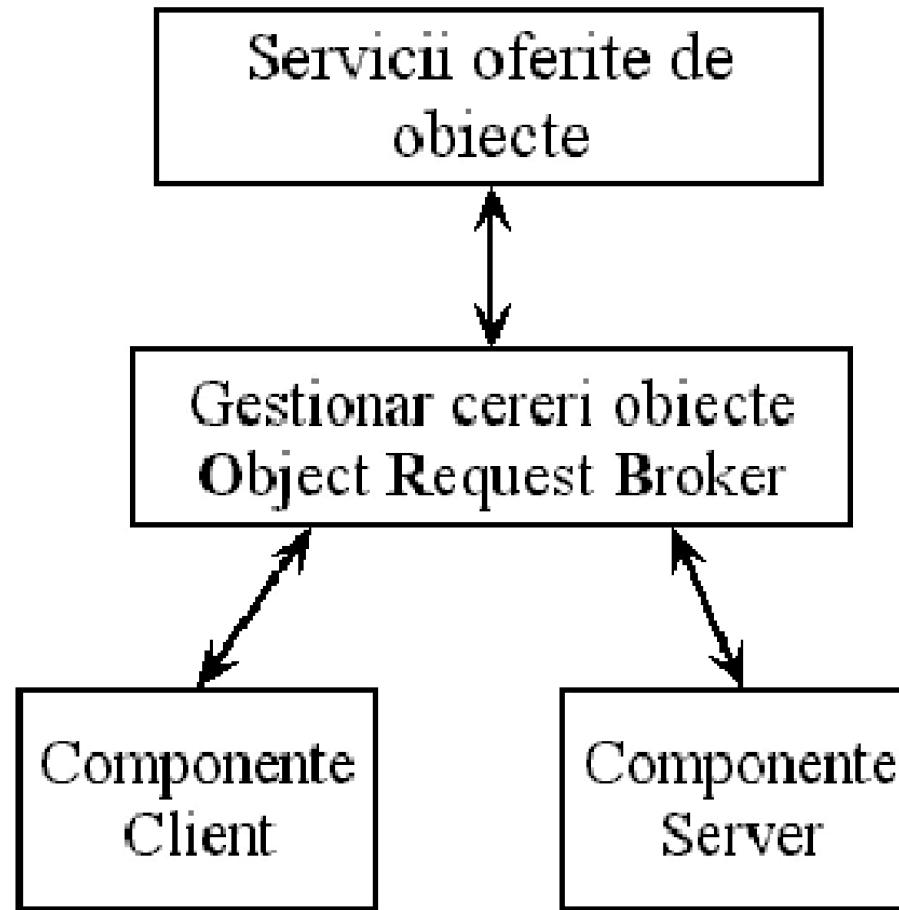
**CORB**

**JBeans for Enterprise**

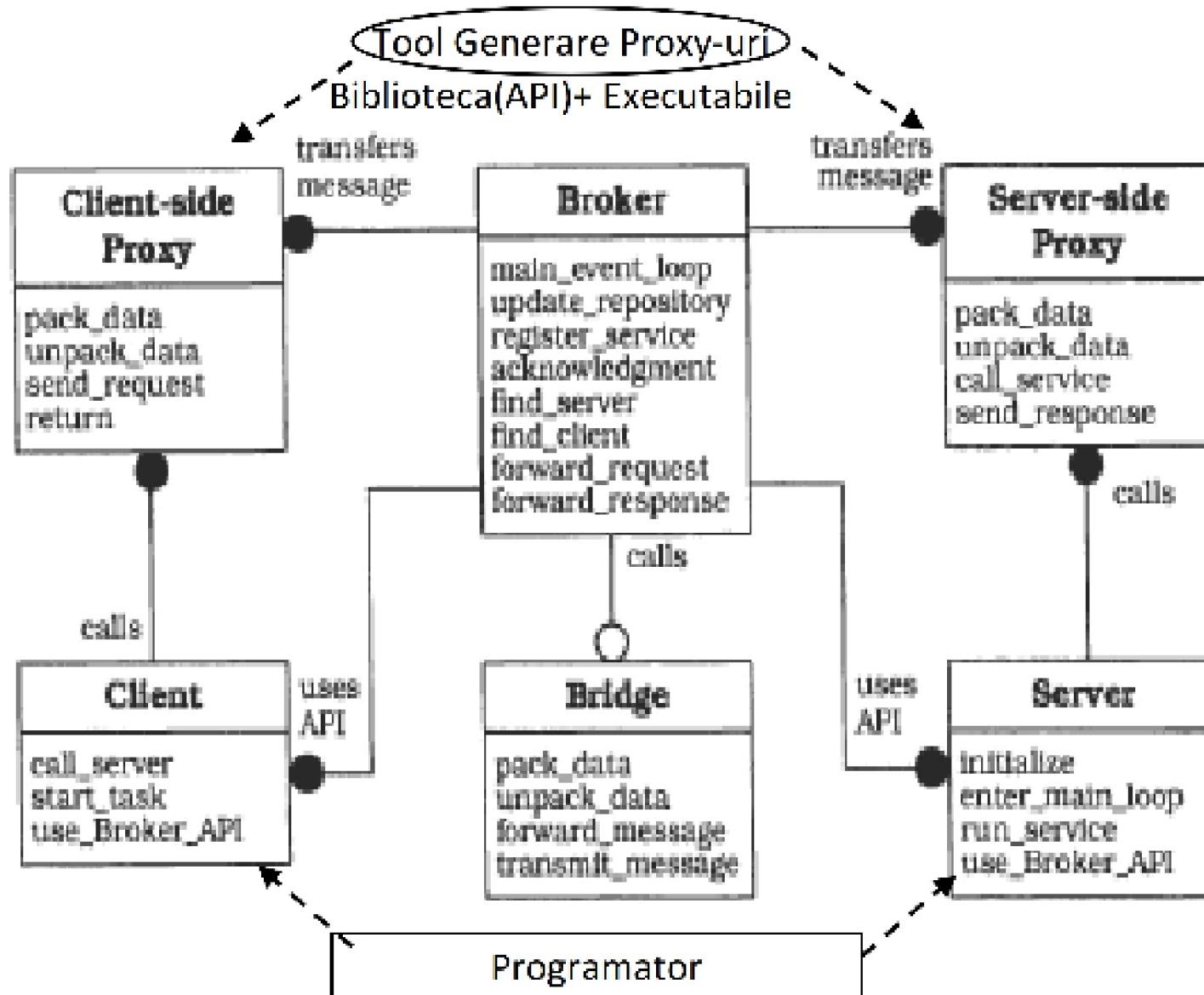
# Arhitectură cu gestionar CORB



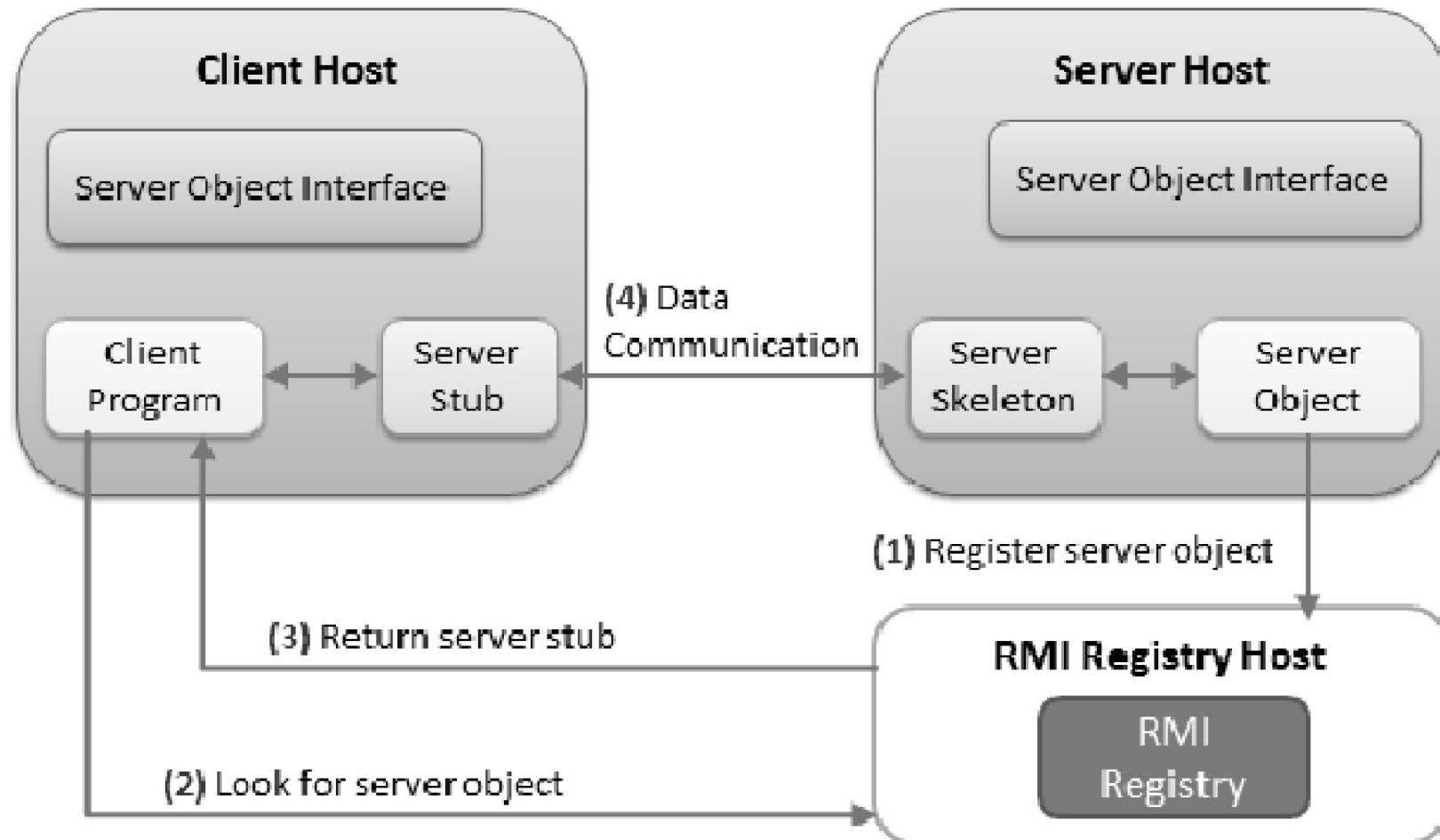
# Arhitectură cu gestionar CORB



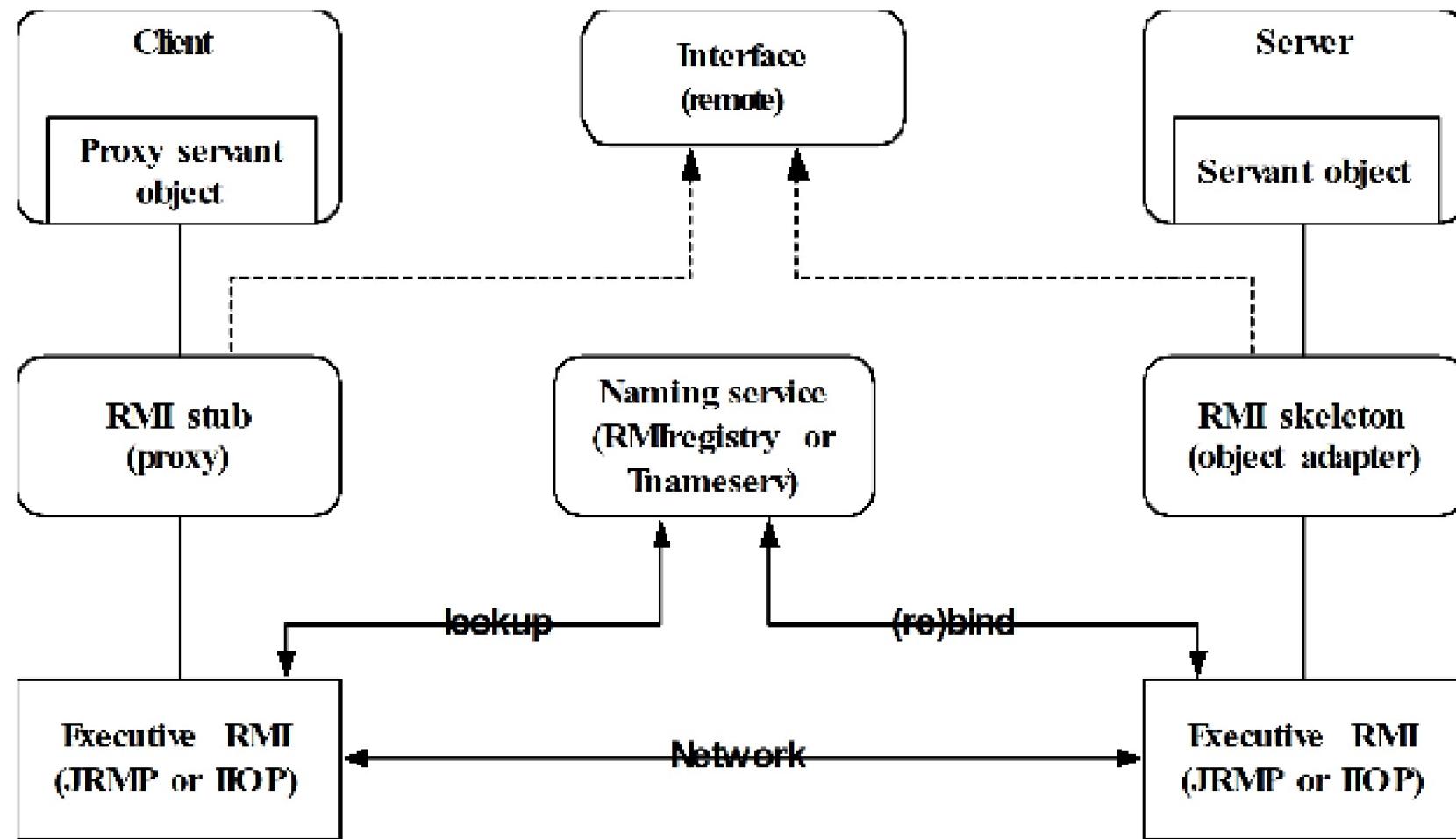
# Exemplu de aplicare broker



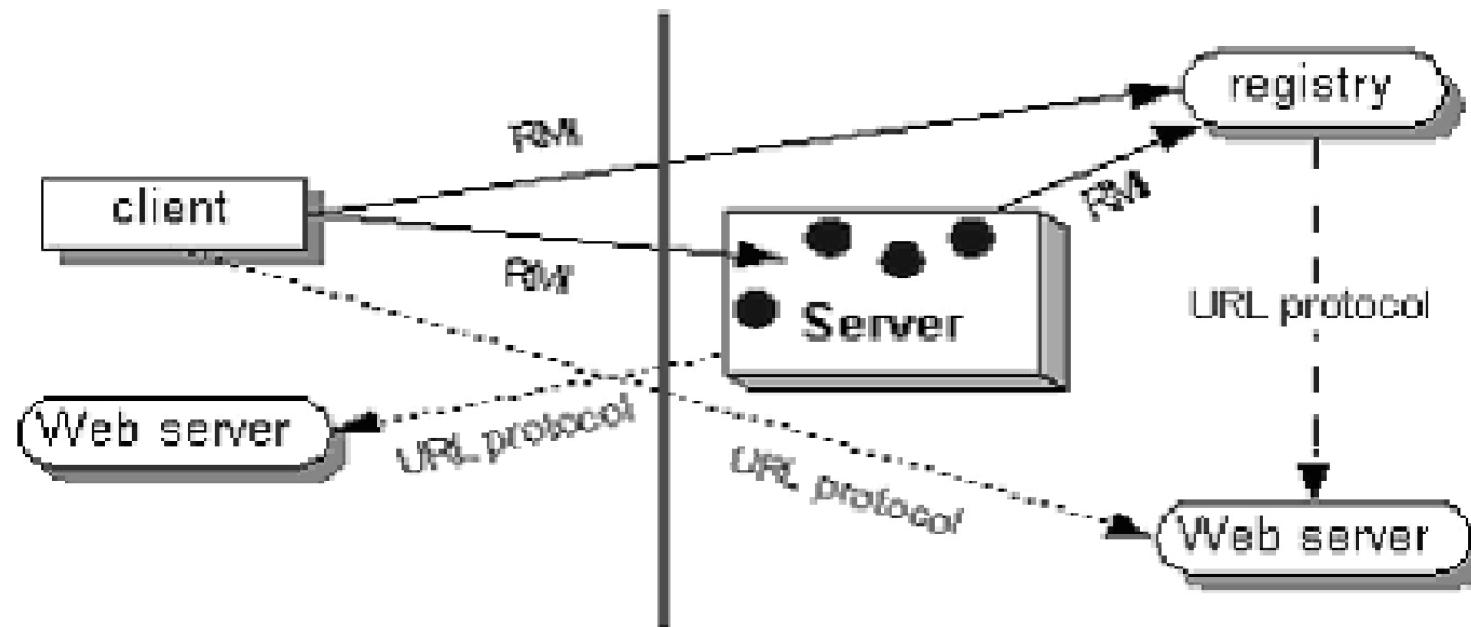
# JAVA RMI



# Detalii utilizare RMI



# Detalii utilizare RMI

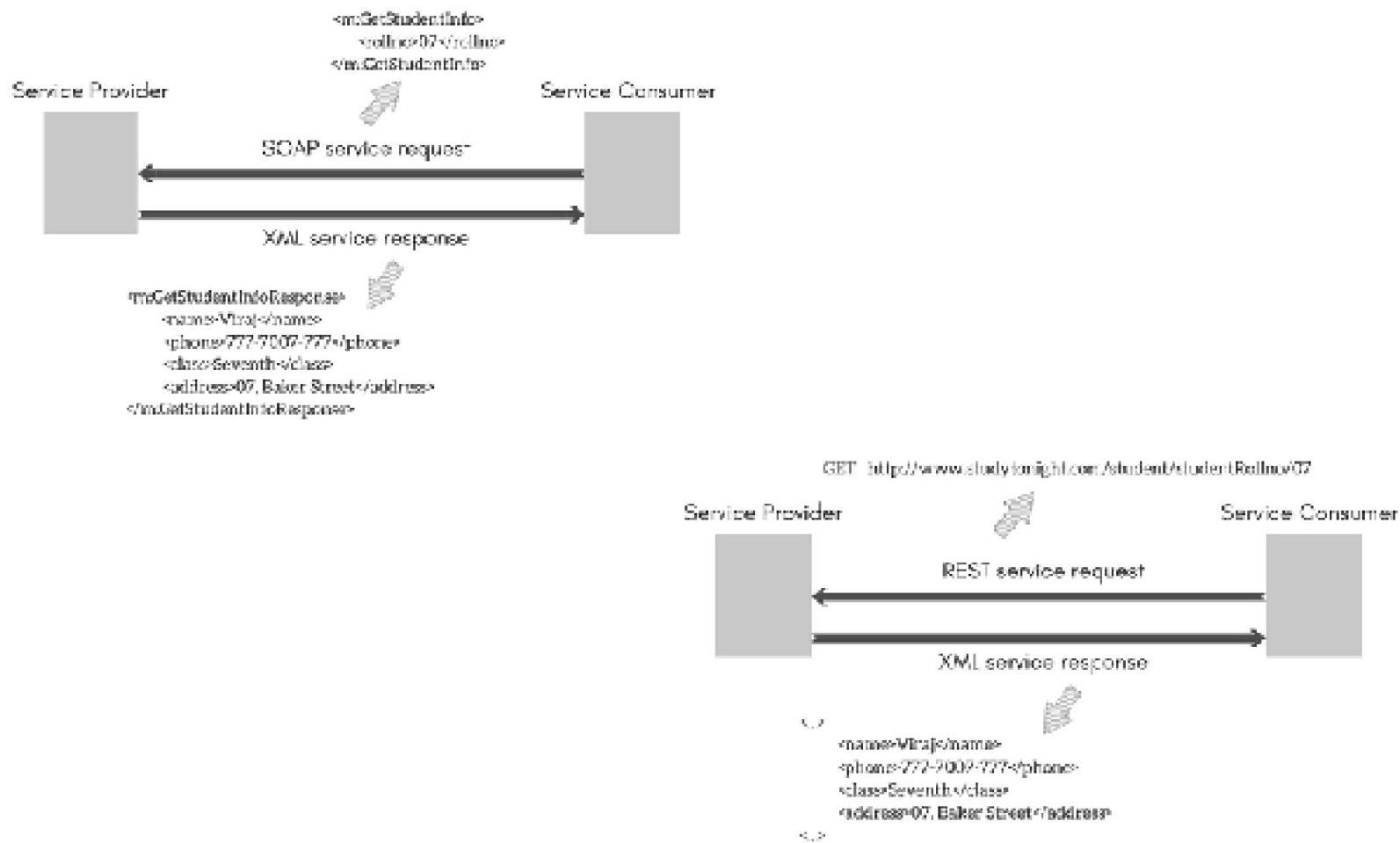


# **Ce sunt serviciile?**

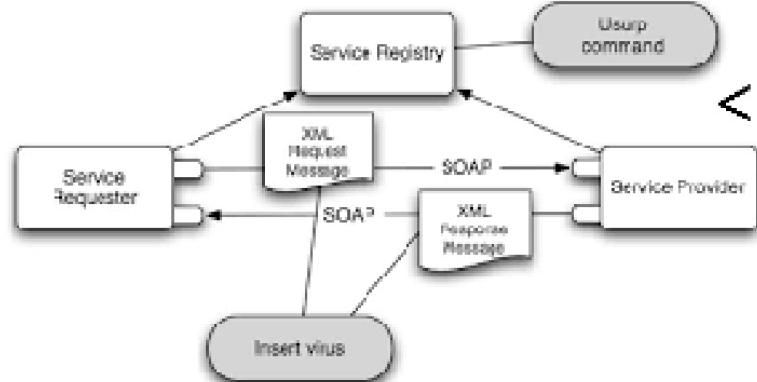
# **Ce sunt Serviciile WEB?**

- cum au apărut?

# Standarde clasice în serviciile WEB



# Servicii Web bazate pe SOAP



<---Și SOAP este sigur nu-i aşă?



Level 3  
Emerging Standards

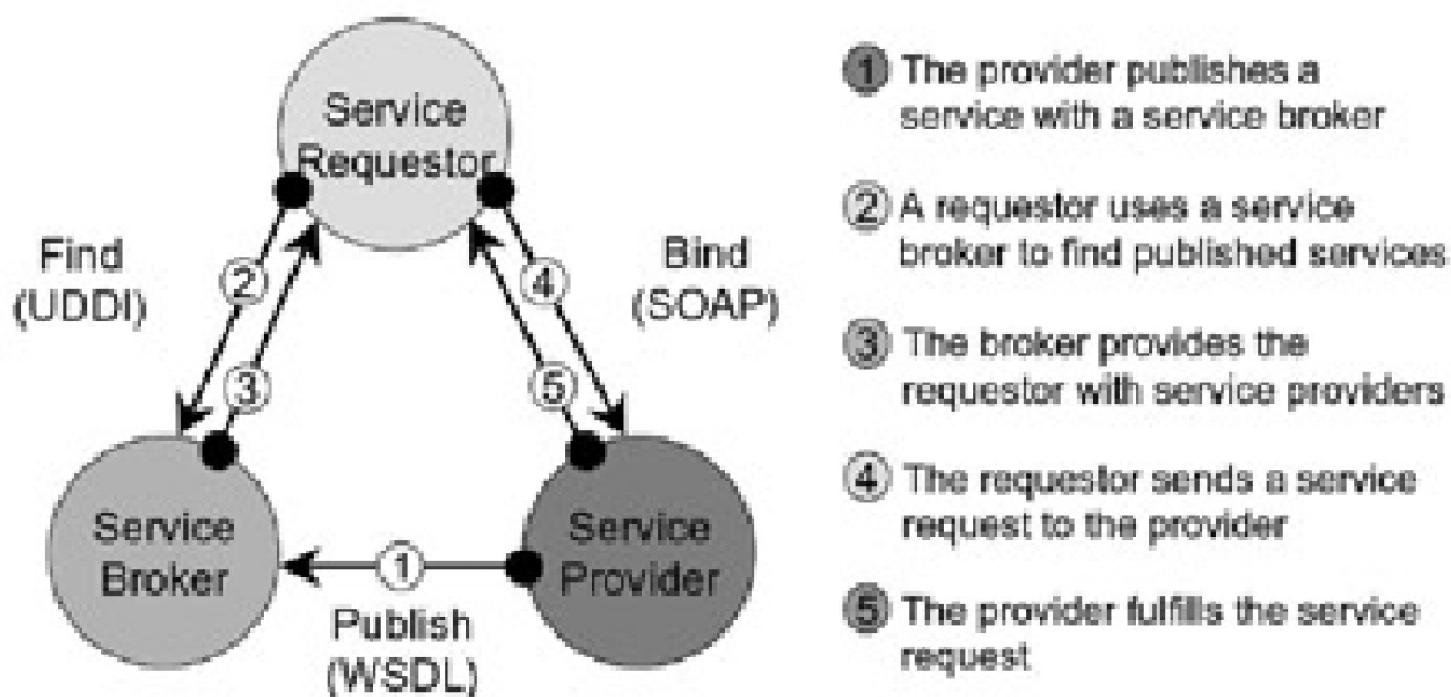
Level 2  
Evolving Standards

Level 1  
Enabling Standards

Stiva tehnologică a serviciilor Web

structura unui mesaj SOAP

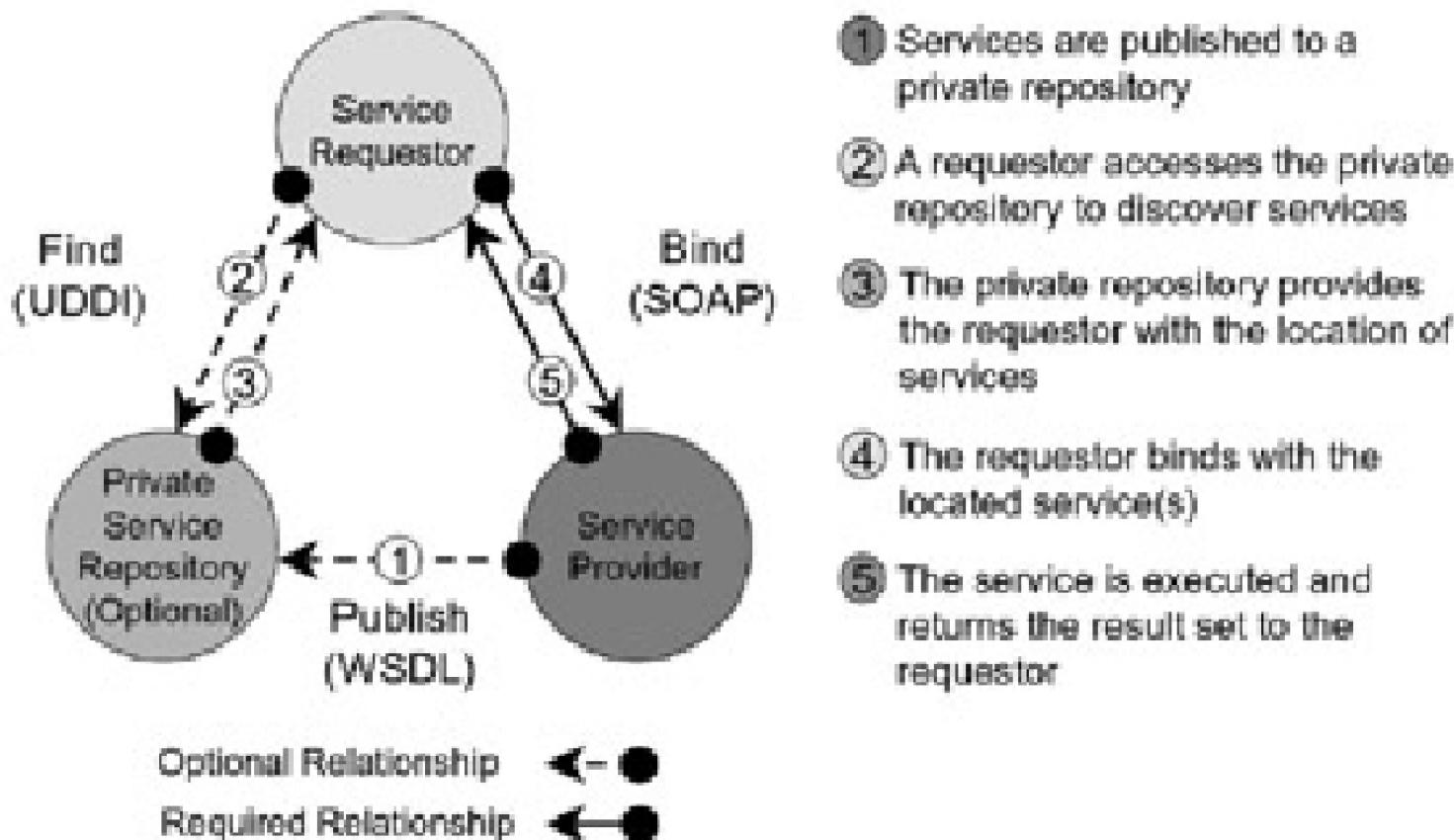
# Abordările moderne în servicii Web



CE S-A DORIT

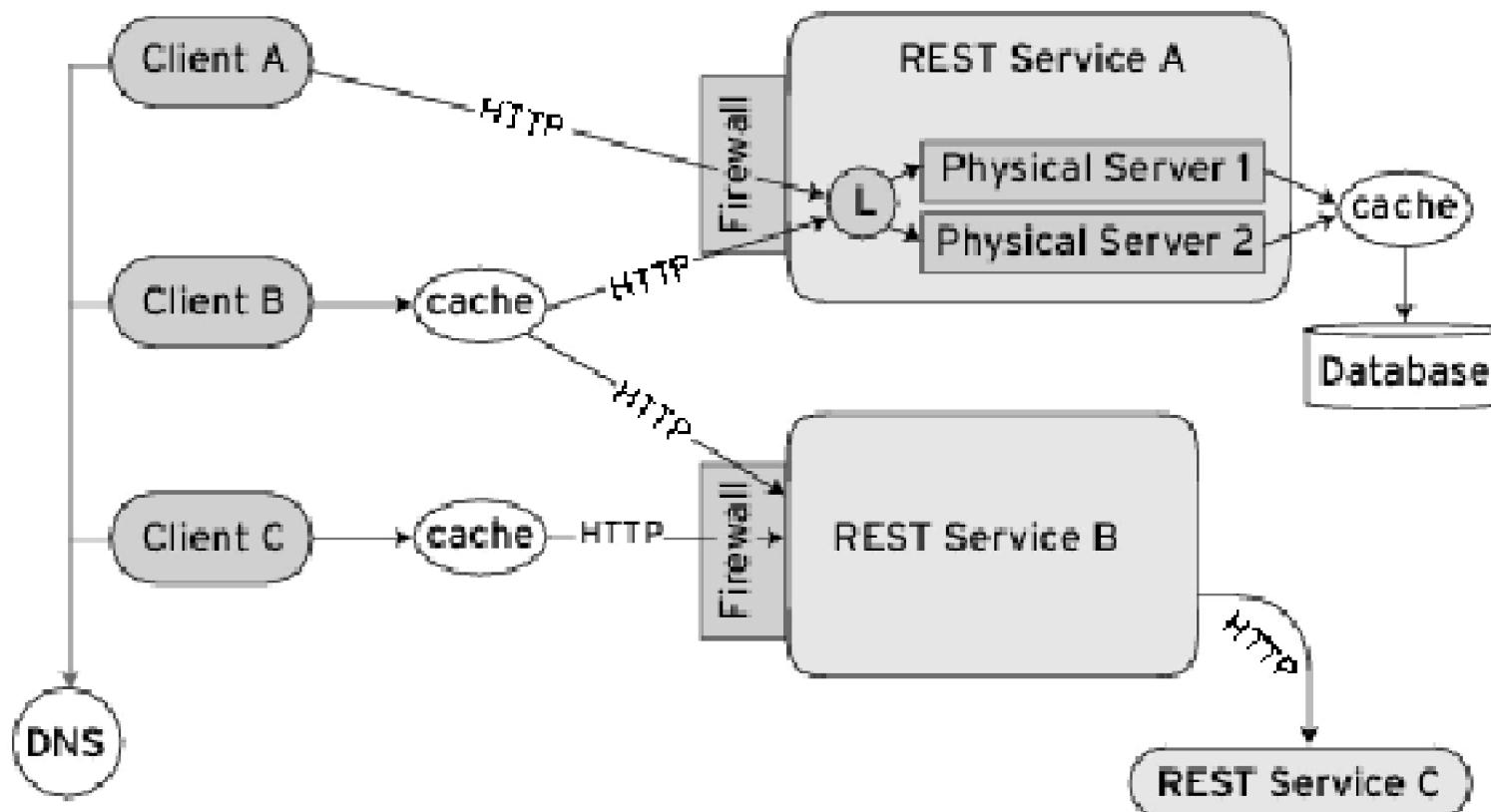
Servicii Web: publish, find & bind.

# Abordările moderne în servicii Web



CE S-A OBTINUT  
Servicii Web: optional publish & find.

# Servicii Web bazate pe REST



# REST vs SOAP

## REST

REST este o arhitectură software.

REST acronimul lui **Representational State Transfer**

REST poate utiliza SOAP deoarece fiind model de proiectare arhitectural poate utiliza orice protocol ar dori

REST utilizează URI pentru a expune logica de afaceri. Deoarece el folosește cereri HTTP același URI poate fi utilizat pentru diverse tipuri de operații

REST preia/are securitatea protocolului de transport utilizat

REST acceptă diverse formate de date, fișier text, HTML, JSON, XML etc.

## SOAP

SOAP este un protocol sau un set de standarde

SOAP acronimul lui **Simple Object Access Protocol**

SOAP nu poate utiliza rest REST deoarece este un protocol.

SOAP utilizează interfața serviciului pentru a expune logica de afaceri.

SOAP își definește propriul standard de securitate.

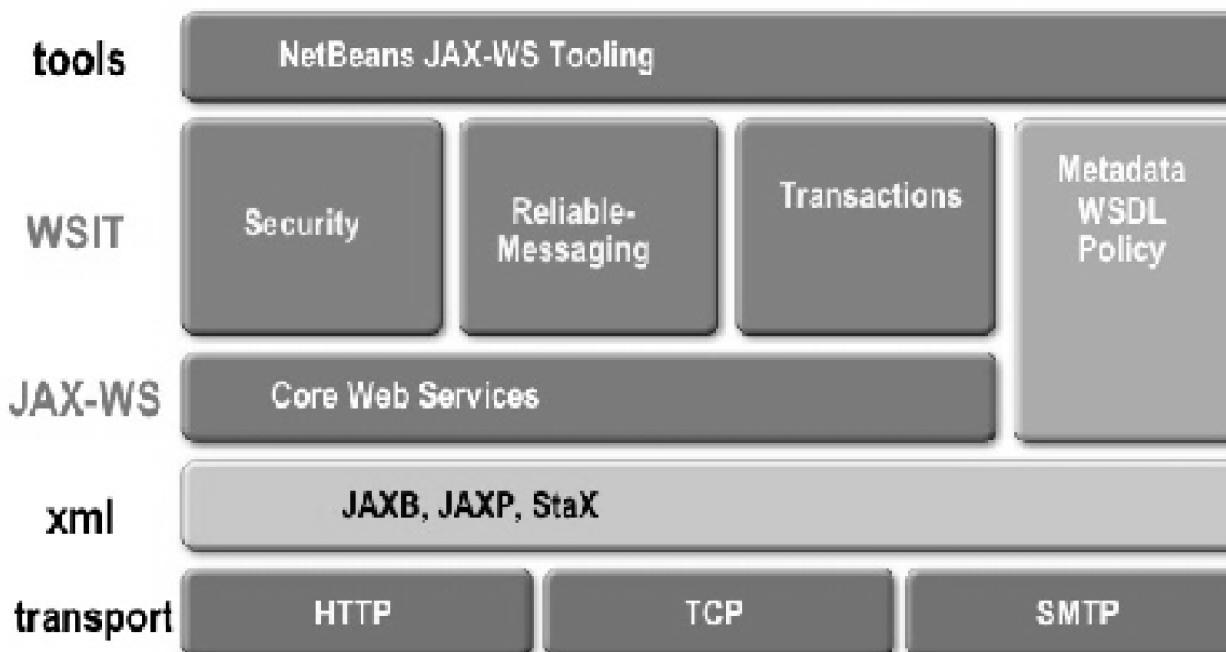
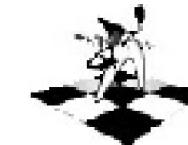
SOAP lucrează numai cu formatul XML.

# Serviciile Web specifice JEE / J2EE



## Sun's Web Services Stack

Metro: JAX-WS , WSIT



JAXB = Java Architecture for XML Binding | JAX-WS = Java APIs for XML Web Services

**Stiva Sun a serviciilor Web**

# **Fasole industriale - Enterprise JavaBeans**

# Java Beans vs Enterprise Java Beans

## EJB

A Java API that allows modular construction of enterprise software

EJB requires an application server or EJB container to run EJB applications

EJB is complex than JavaBeans

Programmer can concern about the business logic as the application server manages services such as transactions and exception handling

## JAVABEANS

Classes that encapsulates many objects into a single object

JavaBeans should be serializable, have a zero argument constructor and allow access to properties using getter and setter methods

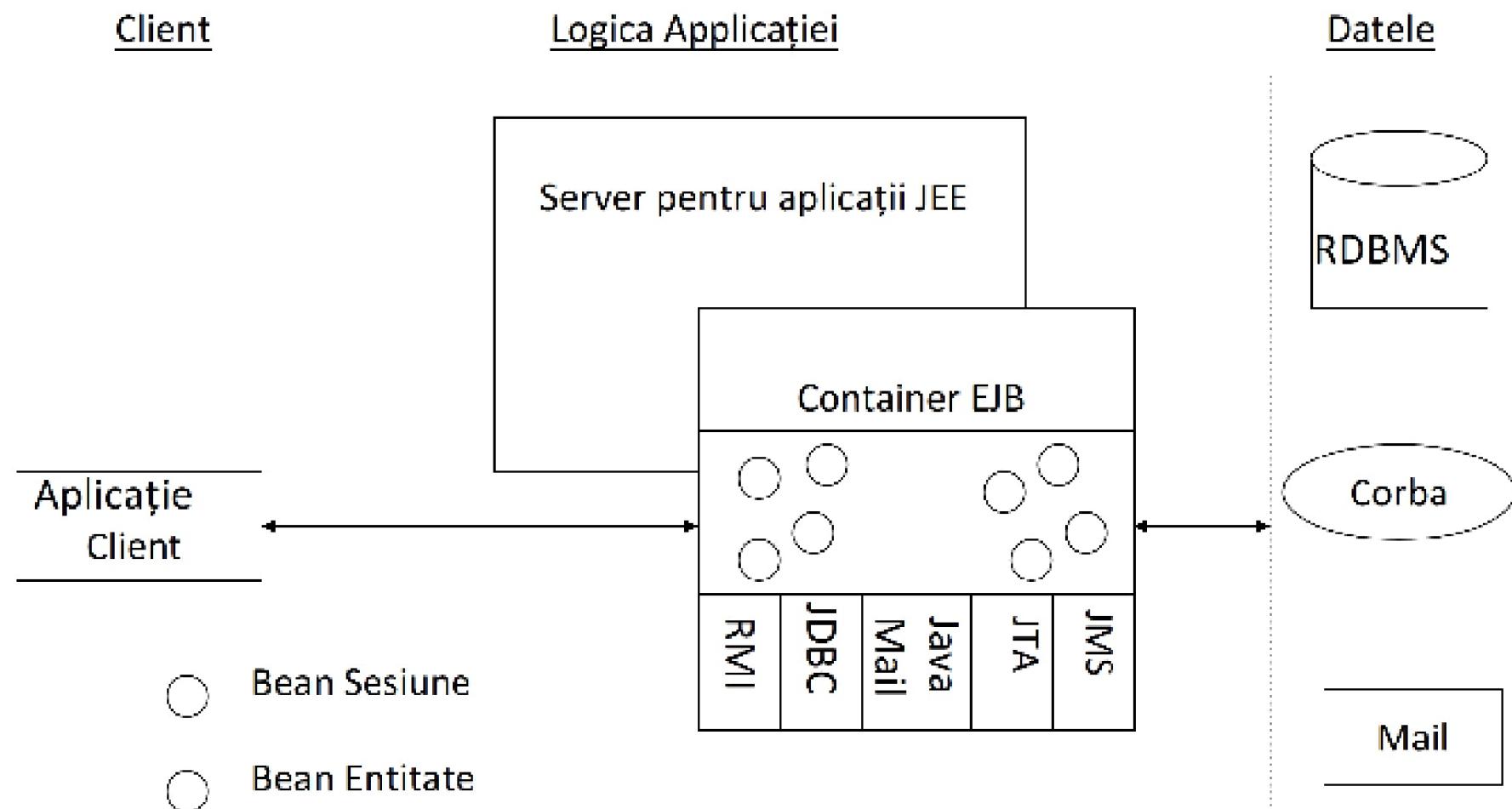
JavaBeans is simpler than EJB

JavaBeans allow another application to use properties and methods of the Bean

# POJO vs Java Beans

POJO	Java Bean
Nu are alte restricții decât cele impuse de limbajul Java	Este un caz particular de POJO care are unele restricții.
Nu permite un control foarte strict al membrilor	Permite controlul total asupra membrilor
Nu poate implementa interfața Serializable	Trebuie să implementeze interfața Serializable
Câmpurile pot fi accesate direct prin numele lor	Câmpurile pot fi accesate numai prin <i>getteri</i> și <i>setteri</i> .
Câmpurile pot avea orice nivel de vizibilitate	Câmpurile pot fi numai <i>private</i>
Este permisă dar <b>nu obligatorie</b> utilizarea unui constructor fără argumente (no-arg)	Este <b>obligatorie</b> utilizarea unui constructor fără argumente (no-arg)
Este recomandat a fi utilizat când nu se dorește nici un fel de restricții asupra membrilor iar utilizatorul poate avea acces complet la entitatea creată	Este utilizat atunci cand se dorește furnizarea unui acces restricționat a utilizatorului la unele părți din entitatea creată (interfață contract etc)

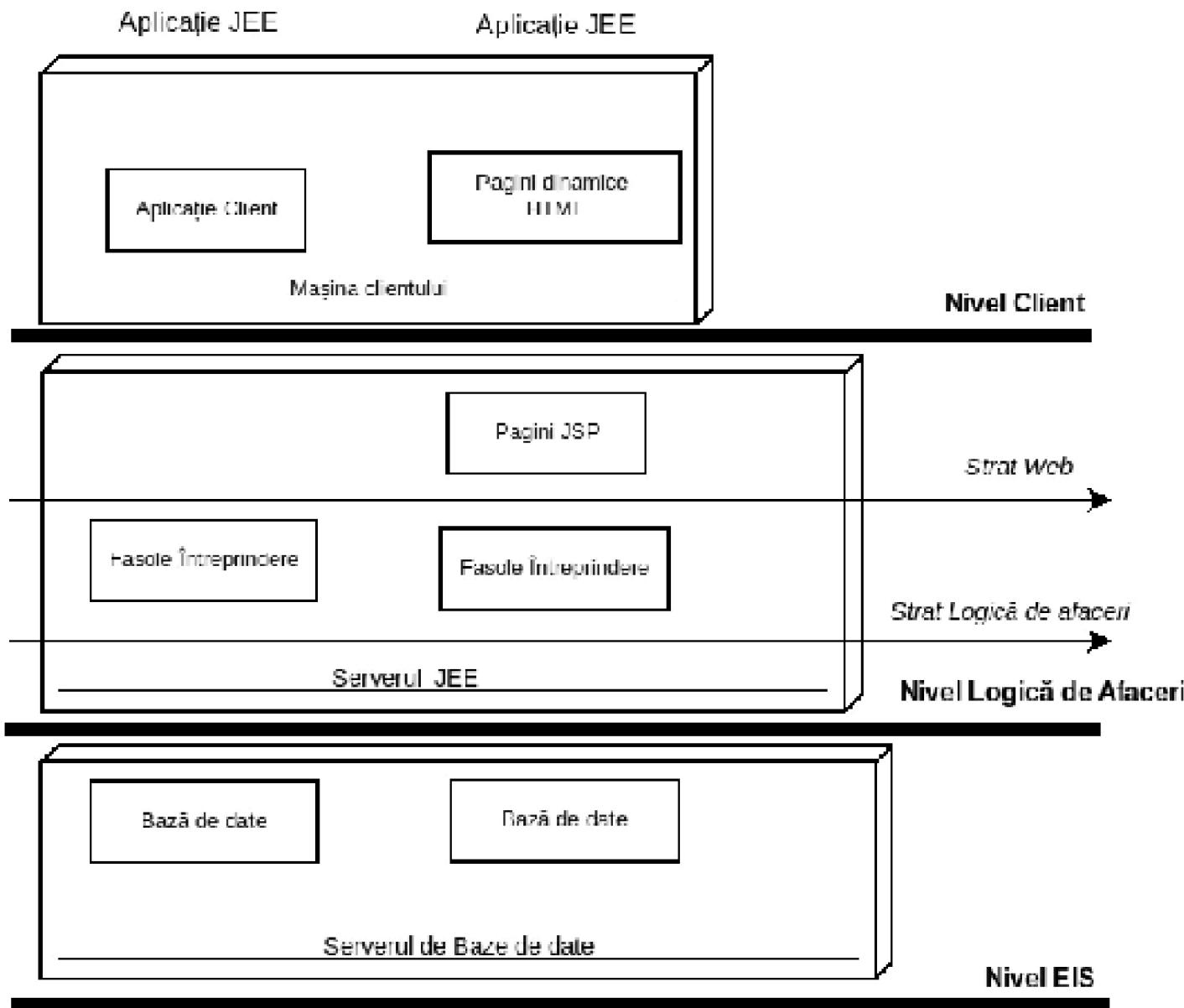
# Diagrama de instalare a EJB



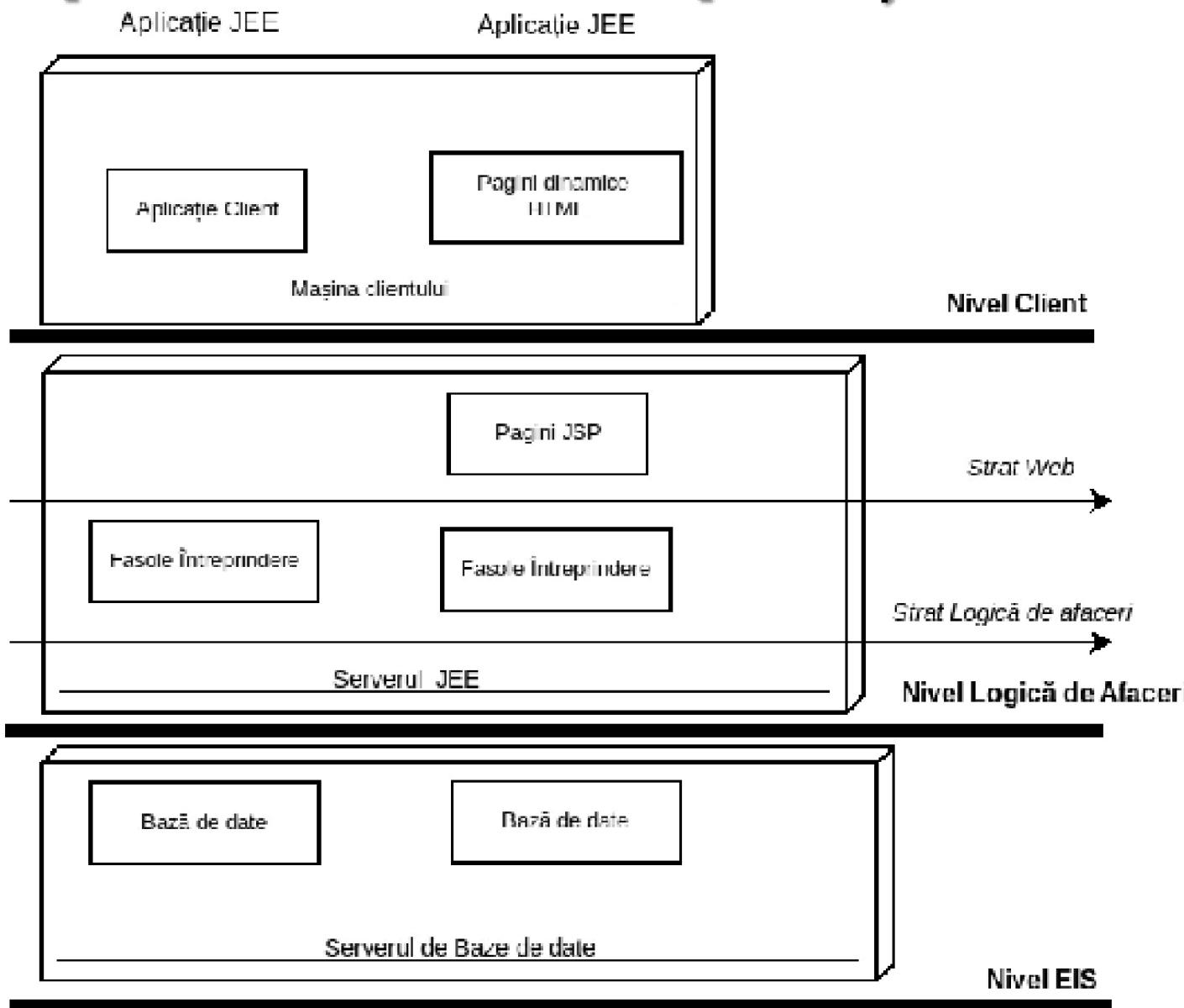
# **Arhitectura JEE**

- **multi-nivel--multistrat**
- **cadru pentru dezvoltare rapidă a aplicațiilor**
- **instalarea implică hardware heterogen**

# Arhitectura JEE



# Componentele unei aplicații Java EE



# **Componentele JEE**

- **Componentele Client**
- **Aplicatiile clientilor**
- Navigatoarele prin Internet
- Applet-urile
- Arhitectura pe componente