

Centralised versus Distributed Computing

Sisteme Distribuite

Mihai Zaharia

Bucătărie

- proful --->
- Regulile de evaluare --> ca și anul trecut
- Examenul --> vezi mai jos

Cum se utilizează cursul ?????

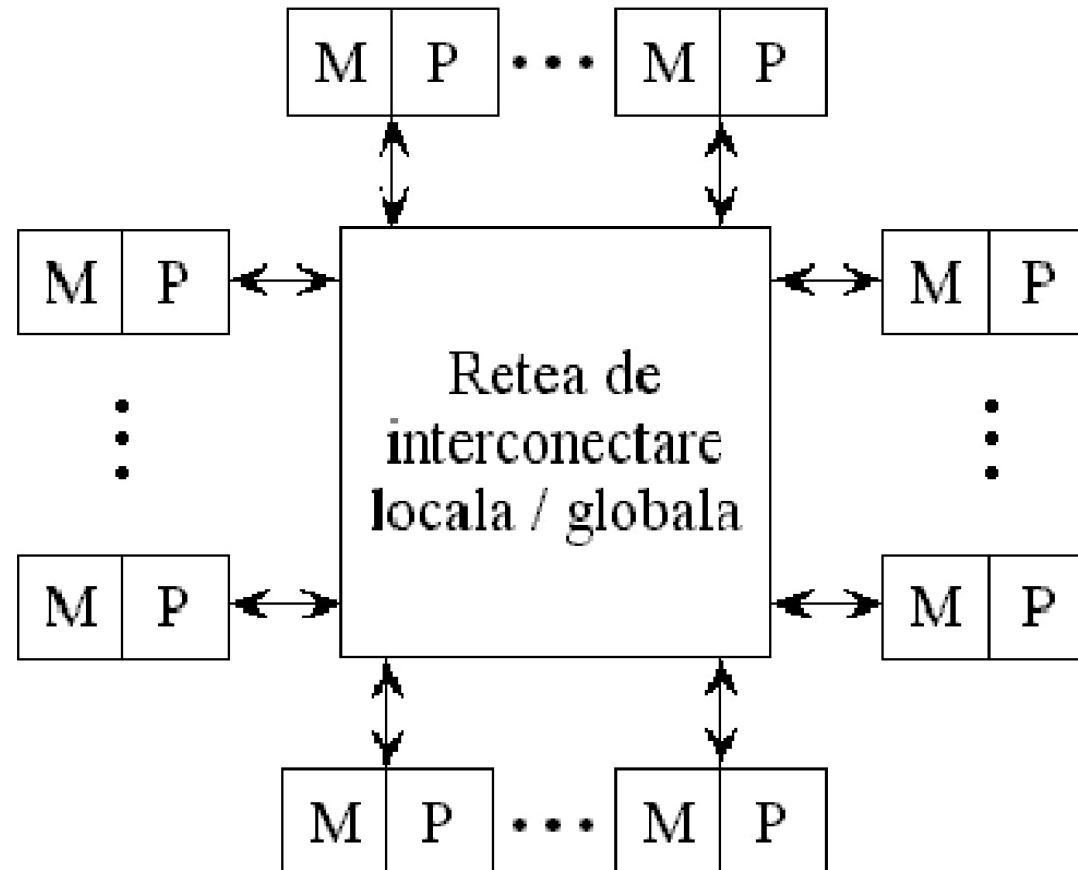
1. se iau notițe! (chiar dacă te doare mâna, neuronul etc)
2. nu este permisă înregistrarea sub nici o formă a titularului - oricare ar fi el - (nu pentru că se supără ci pentru că se încalcă simultan trei legi în vigoare!)
3. te duci acasă și citești despre termenii din notițe și slide-uri până te lămurești 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 si concepte

sisteme strâns cuplate

sisteme slab cuplate

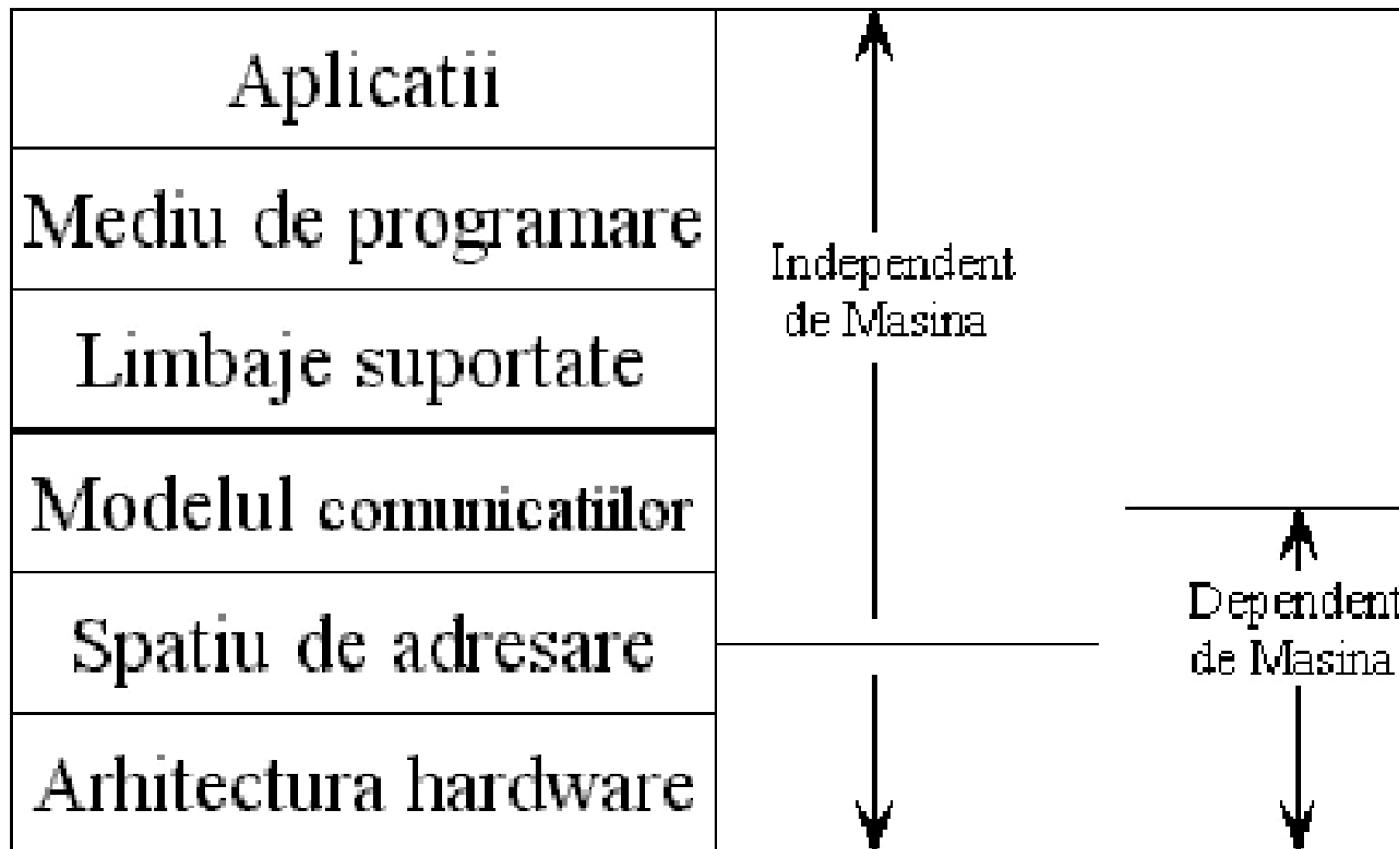
Modelul sistemelor HW slab cuplate



M = Memorie; P = Procesor

Multicalcătoare cu memorie distribuită

Modelul Li al unui sistem distribuit(hw+sw)

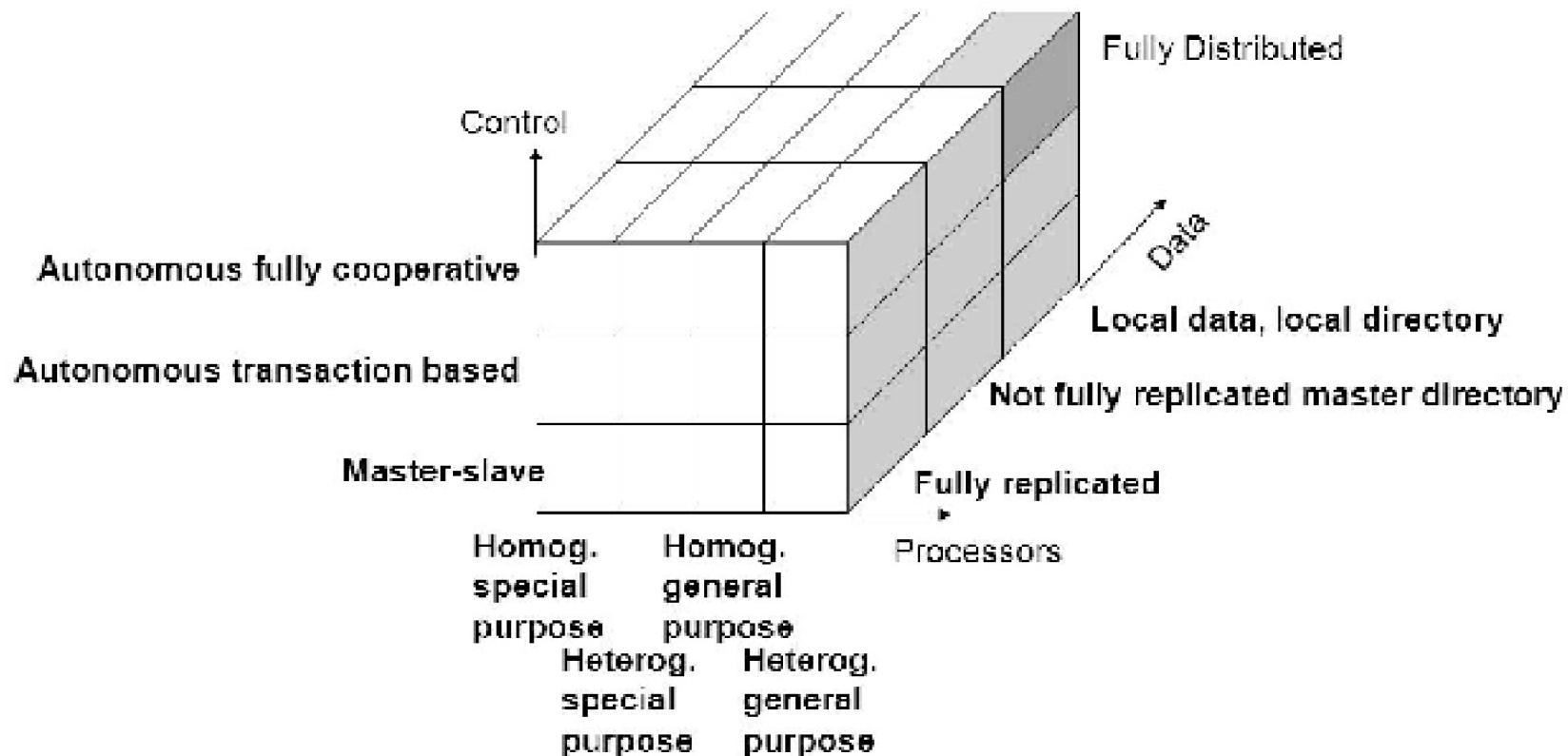


Termeni și concepte

arhitecturi software strâns
cuplate - monolit

arhitecturi software slab
cuplate - servicii/microservicii

Sistem distribuit Enslow 1978

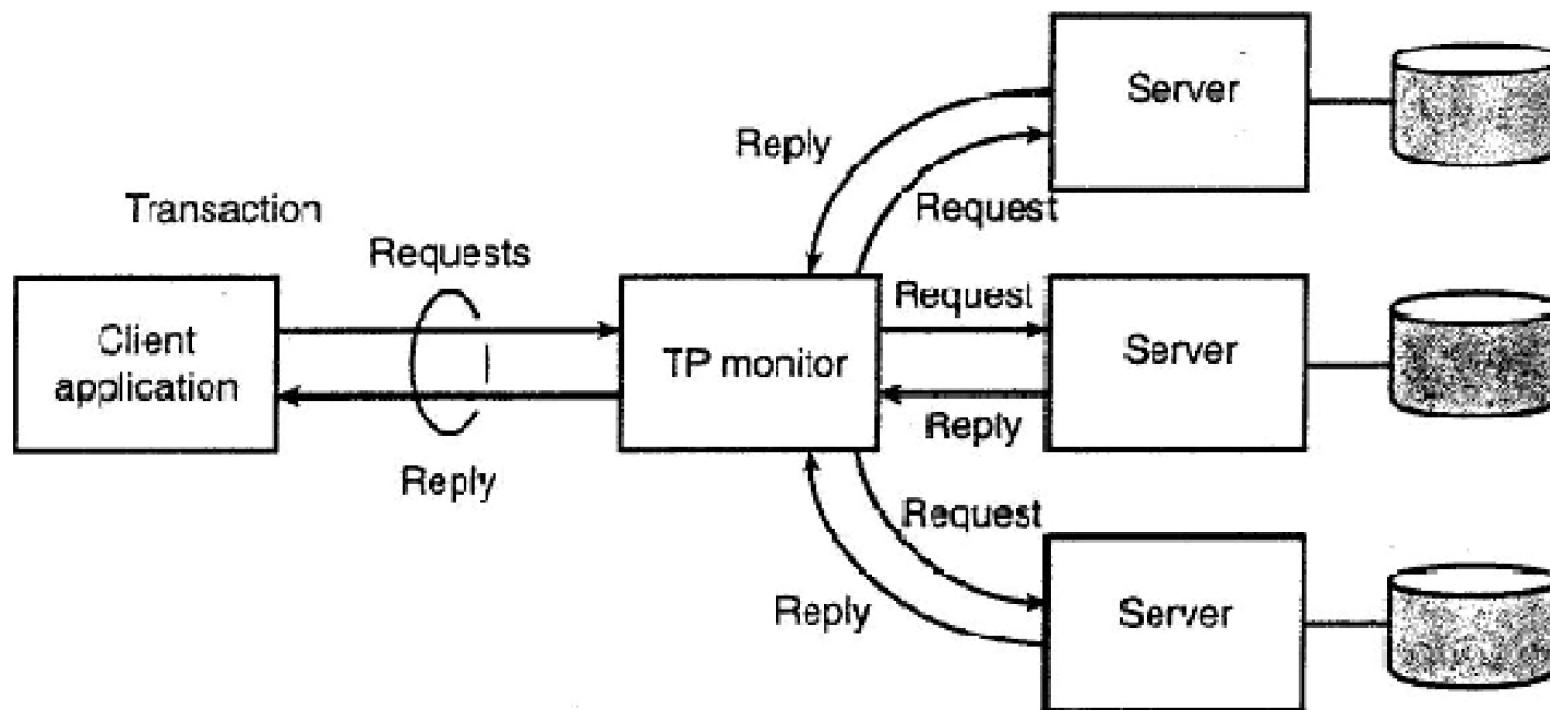


Hardware distribuit fizic/geografic

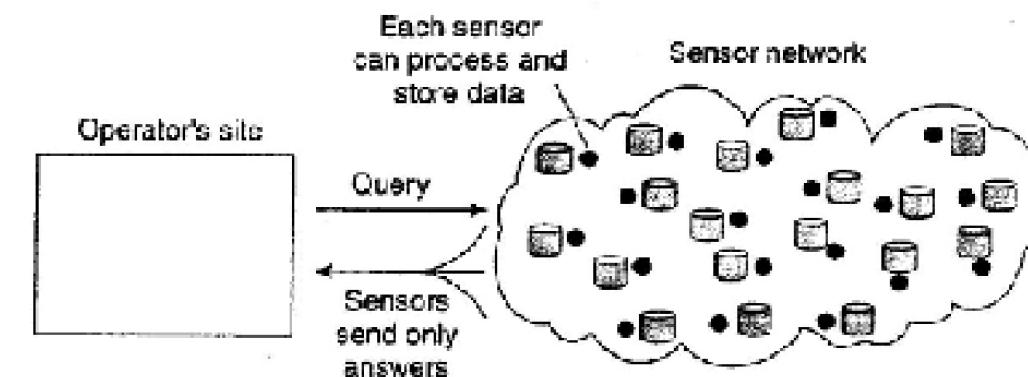
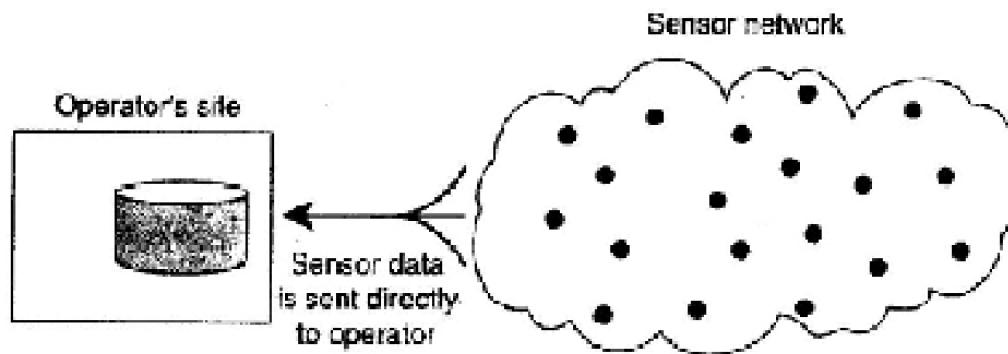
Control Distribuit

Datele distribuite

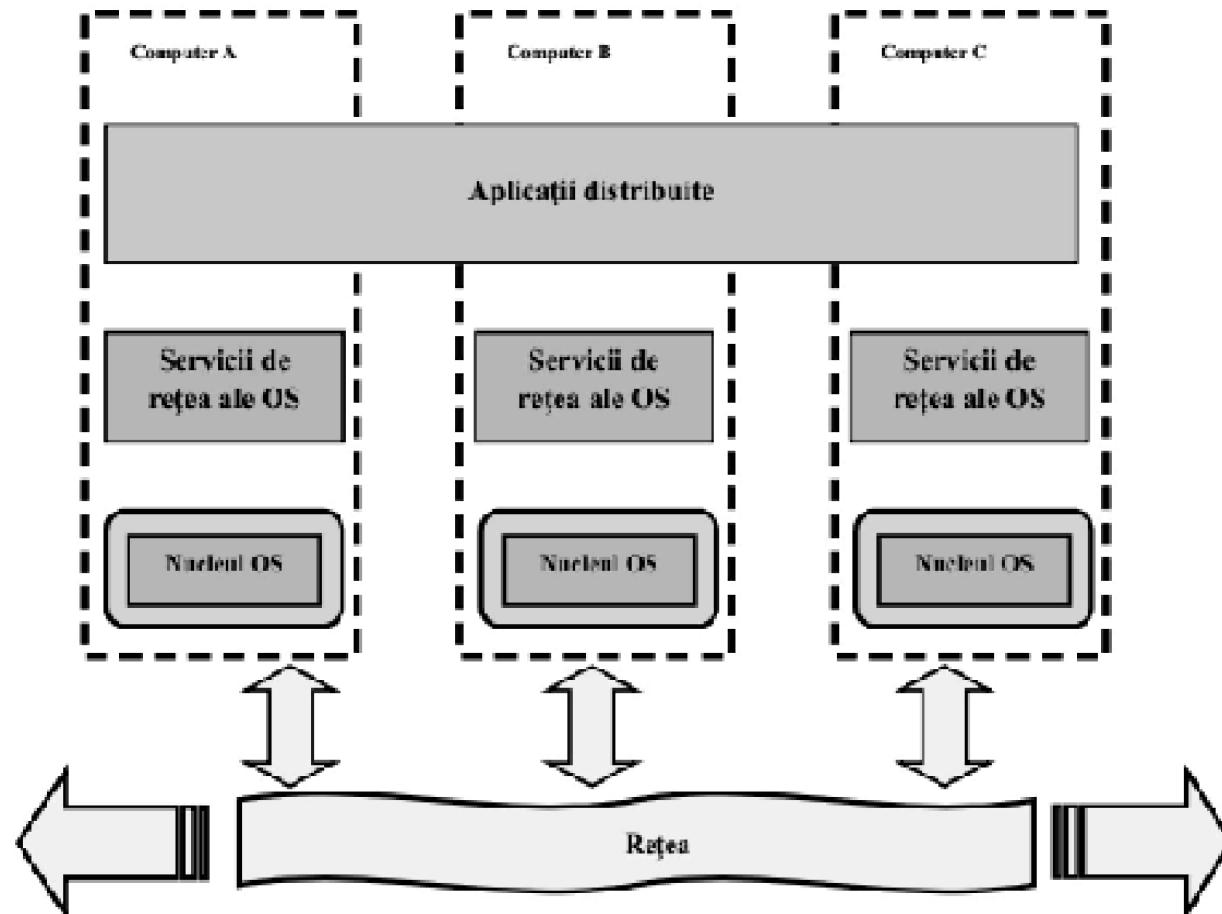
Exemple



Exemple

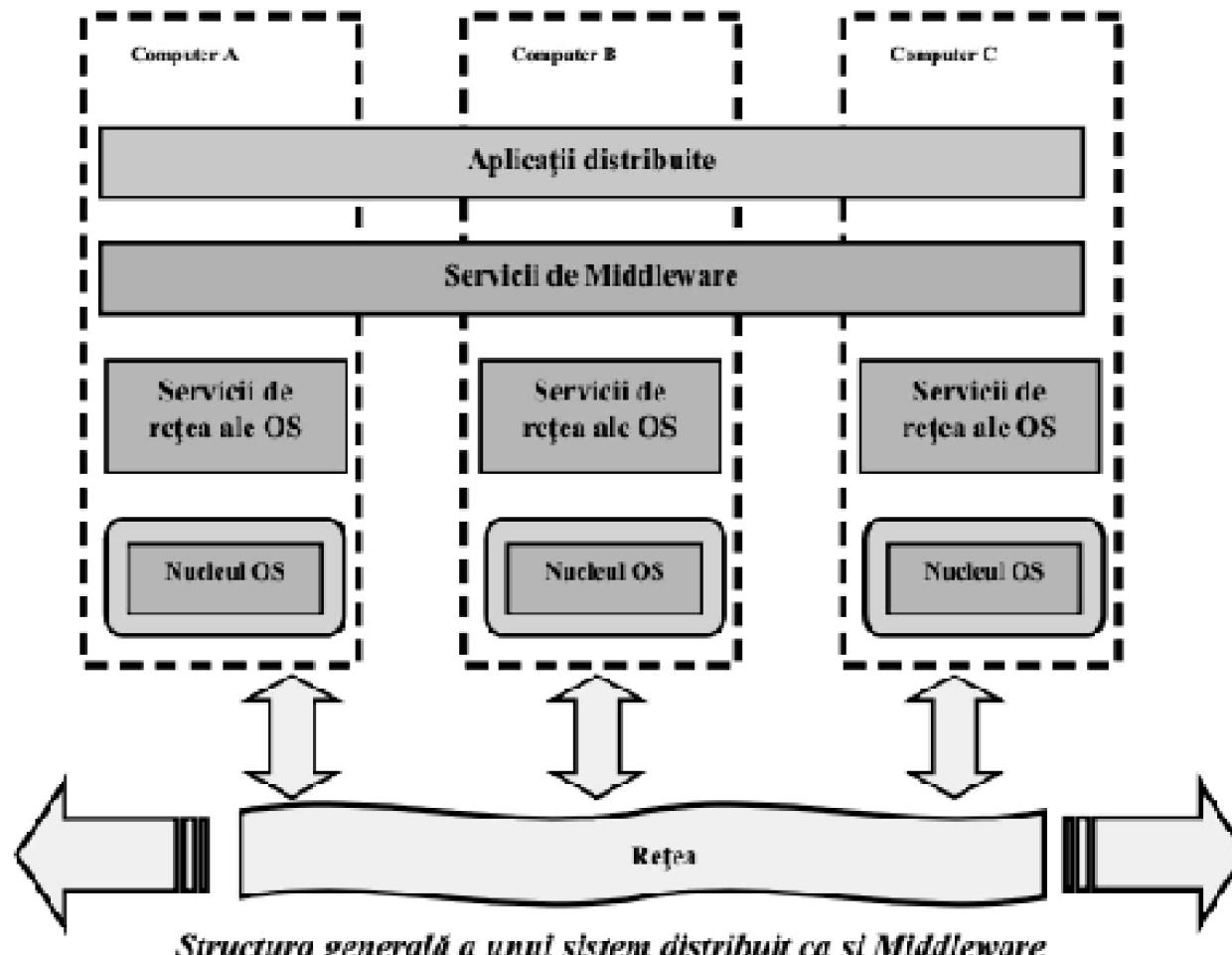


Ce suport ne oferă sistemele de operare?

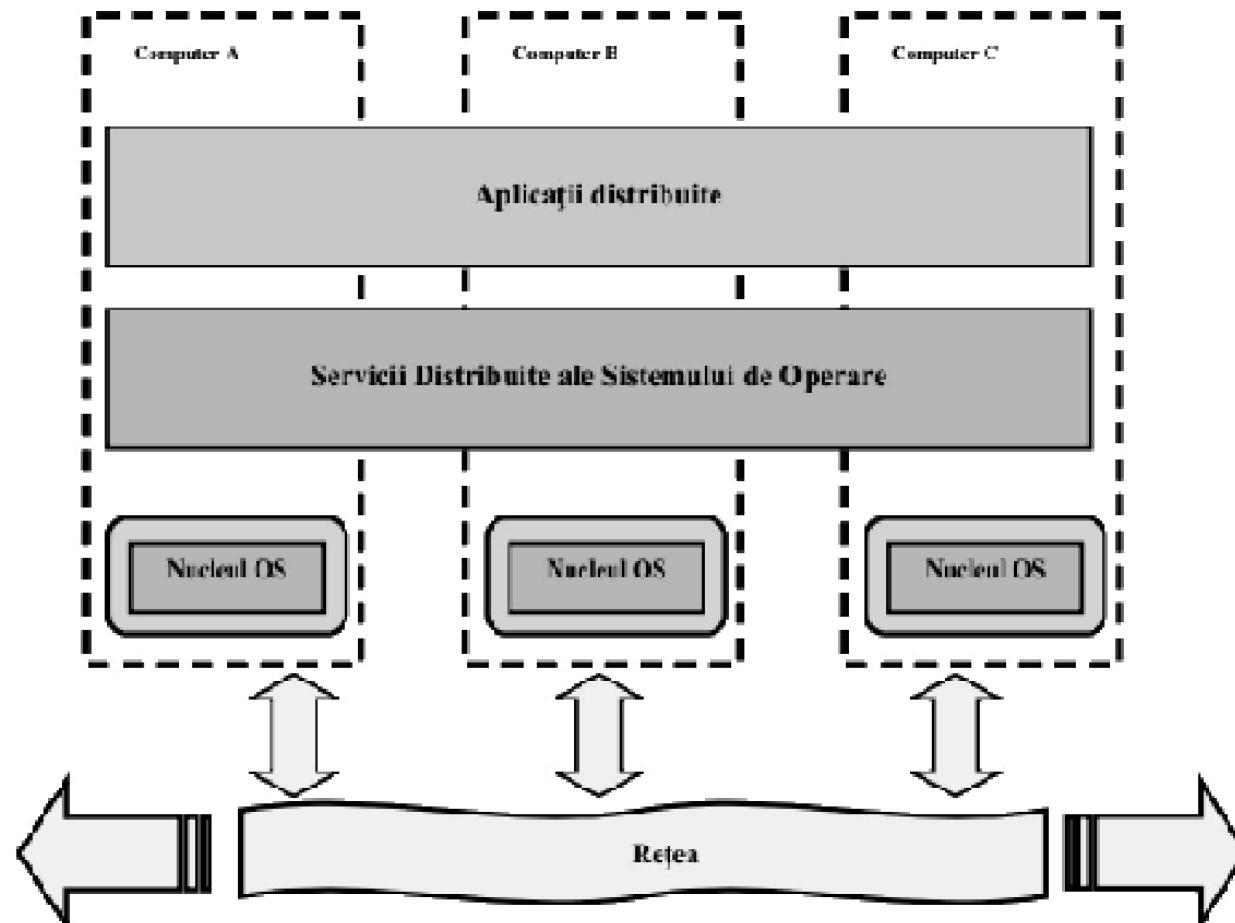


Strucutura generală a unui sistem de operare în rețea

Ce suport ne oferă sistemele de operare?

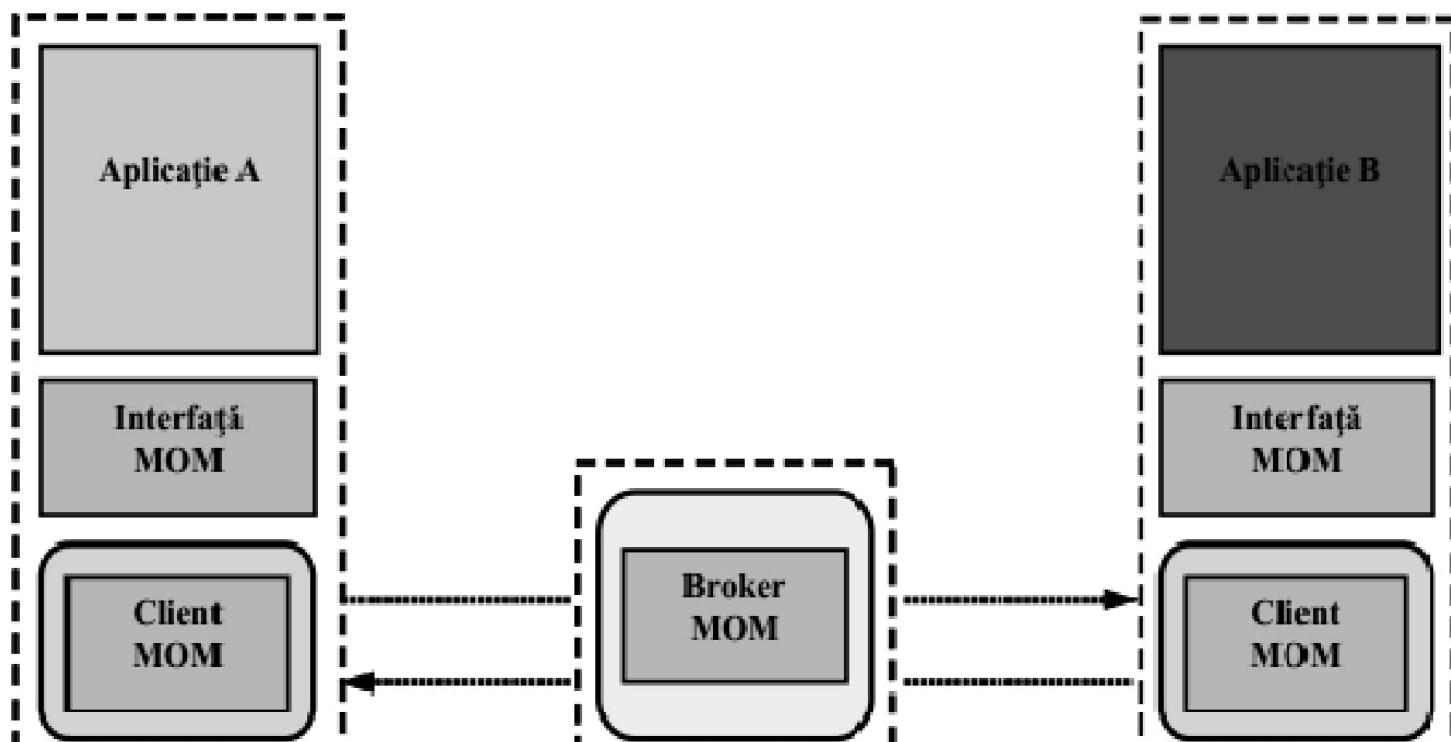


Ce suport ne oferă sistemele de operare?



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

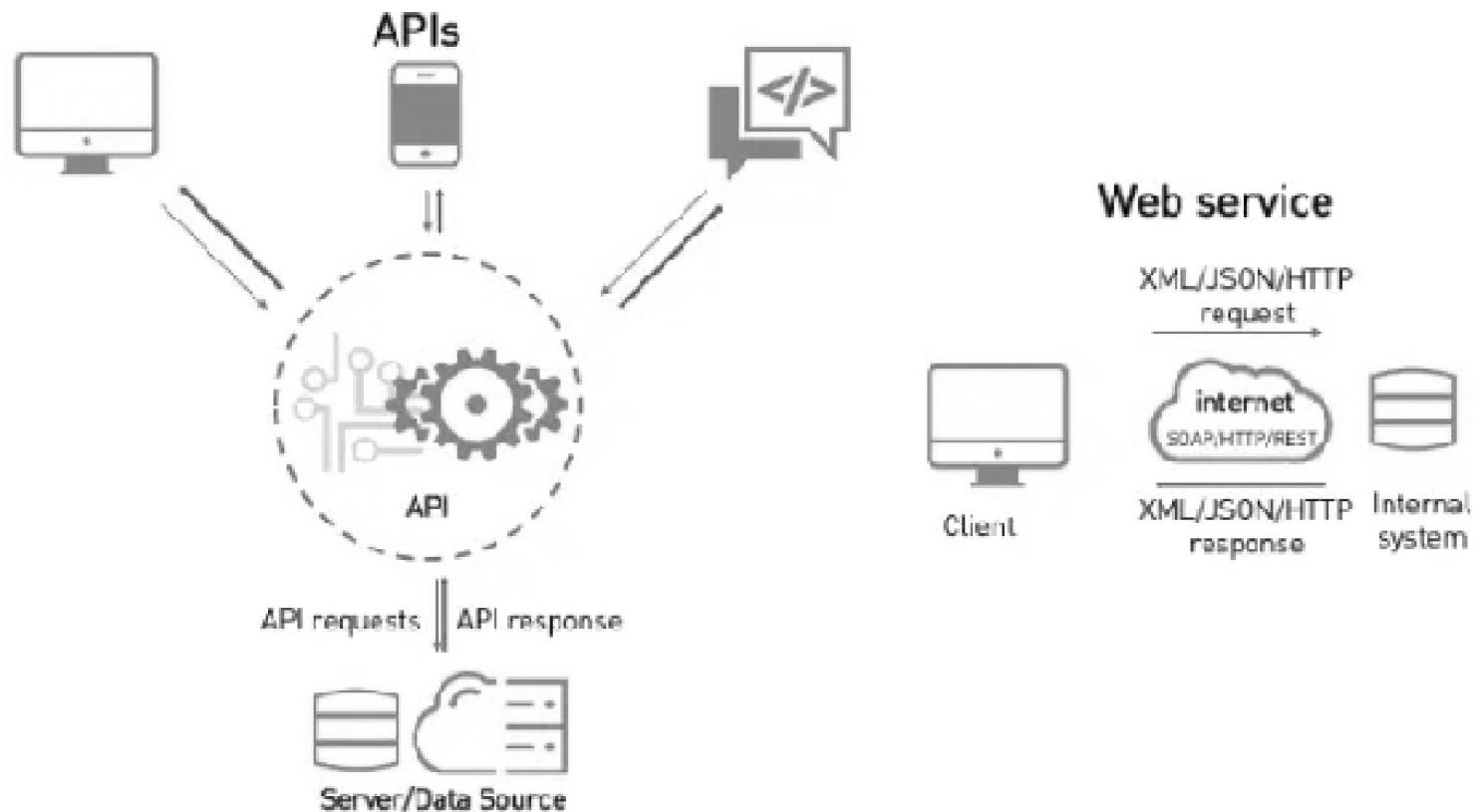
comunicațiile

SWOT - SD - Dezvantaje

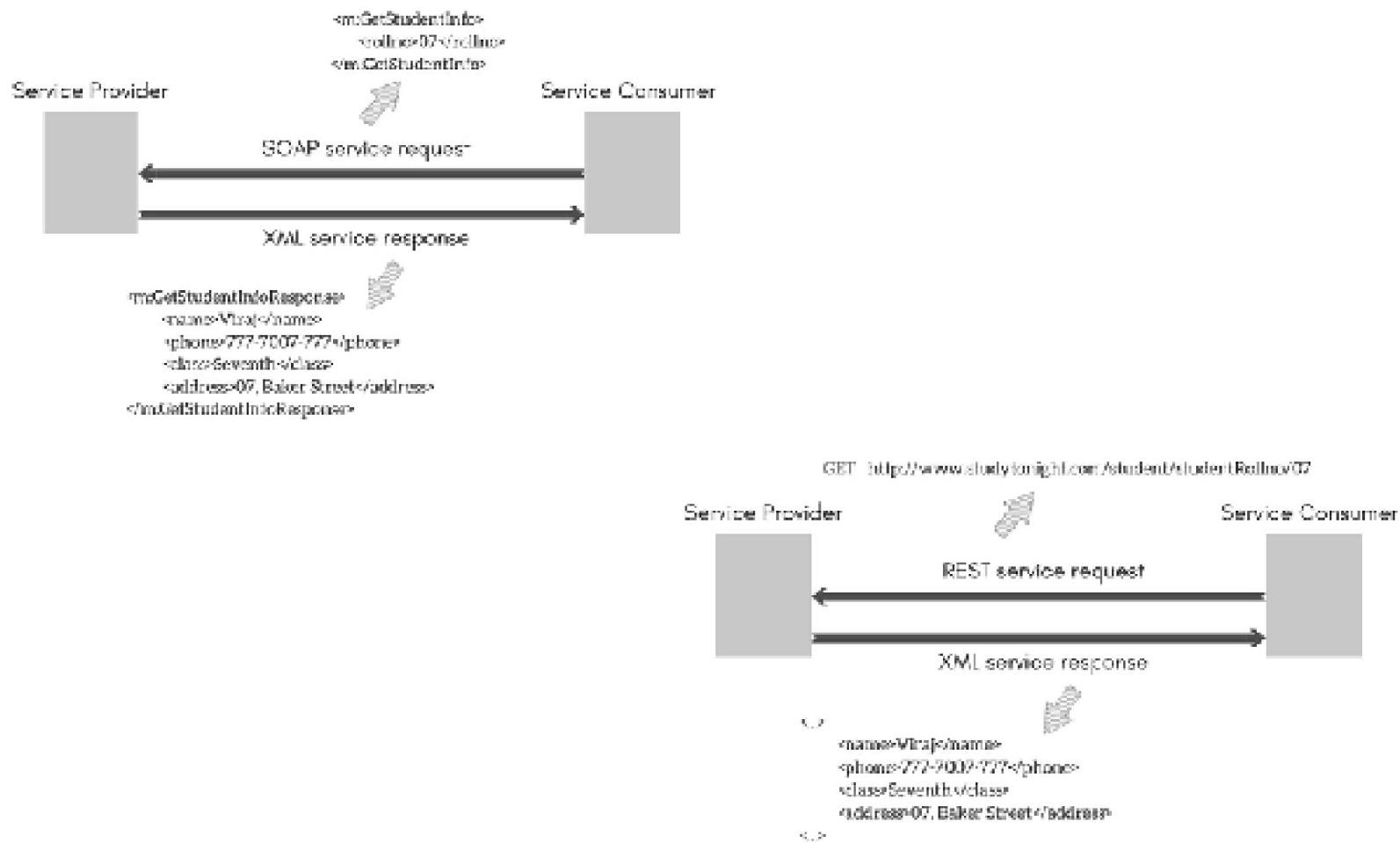
securitatea

Ce sunt Serviciile WEB?

- cum au apărut?



Standarde clasice în serviciile WEB



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.

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

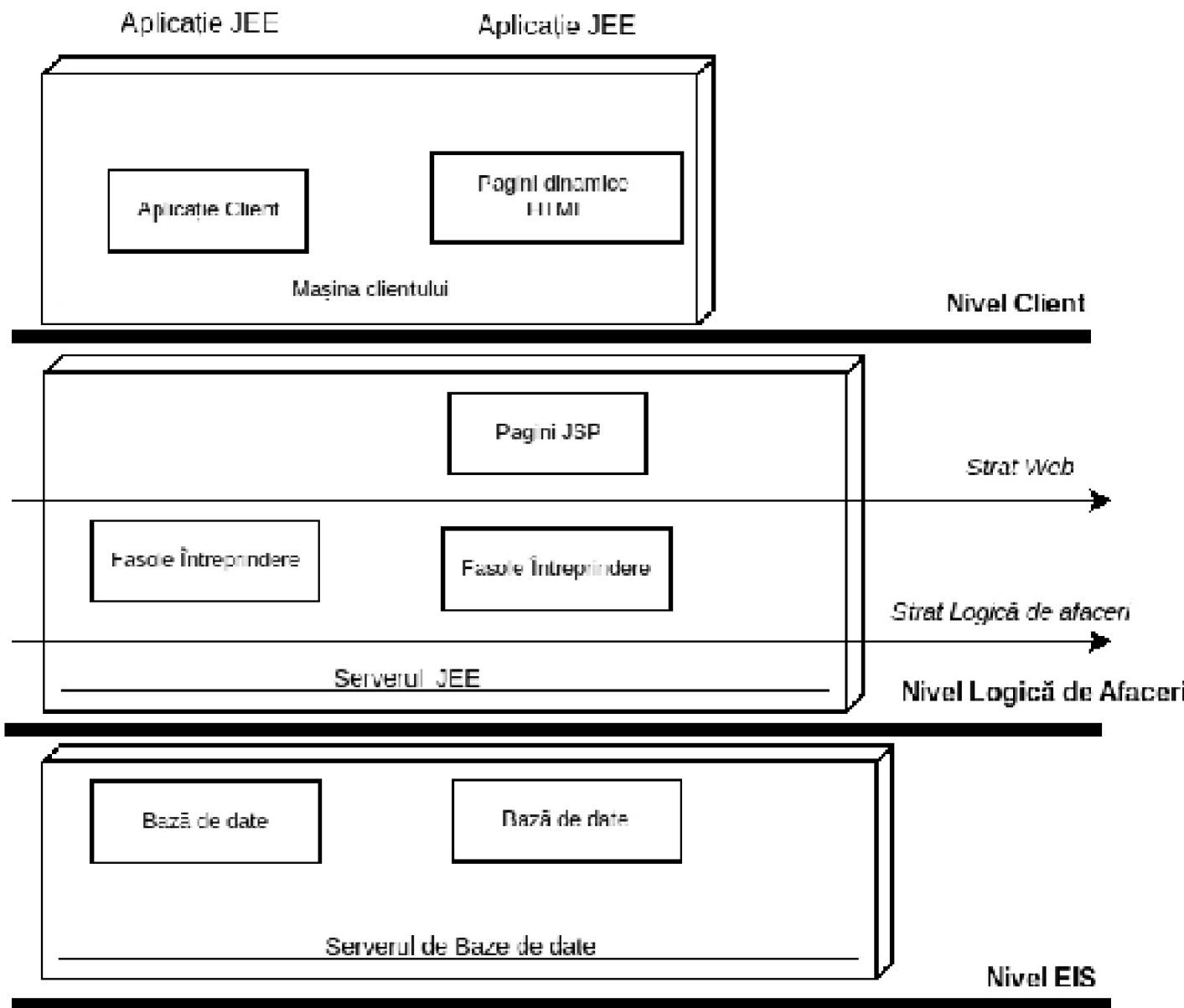
POJO vs Java Beans

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 nu obligatorie utilizarea unui constructor fără argumente (no-arg).	Este obligatorie 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)

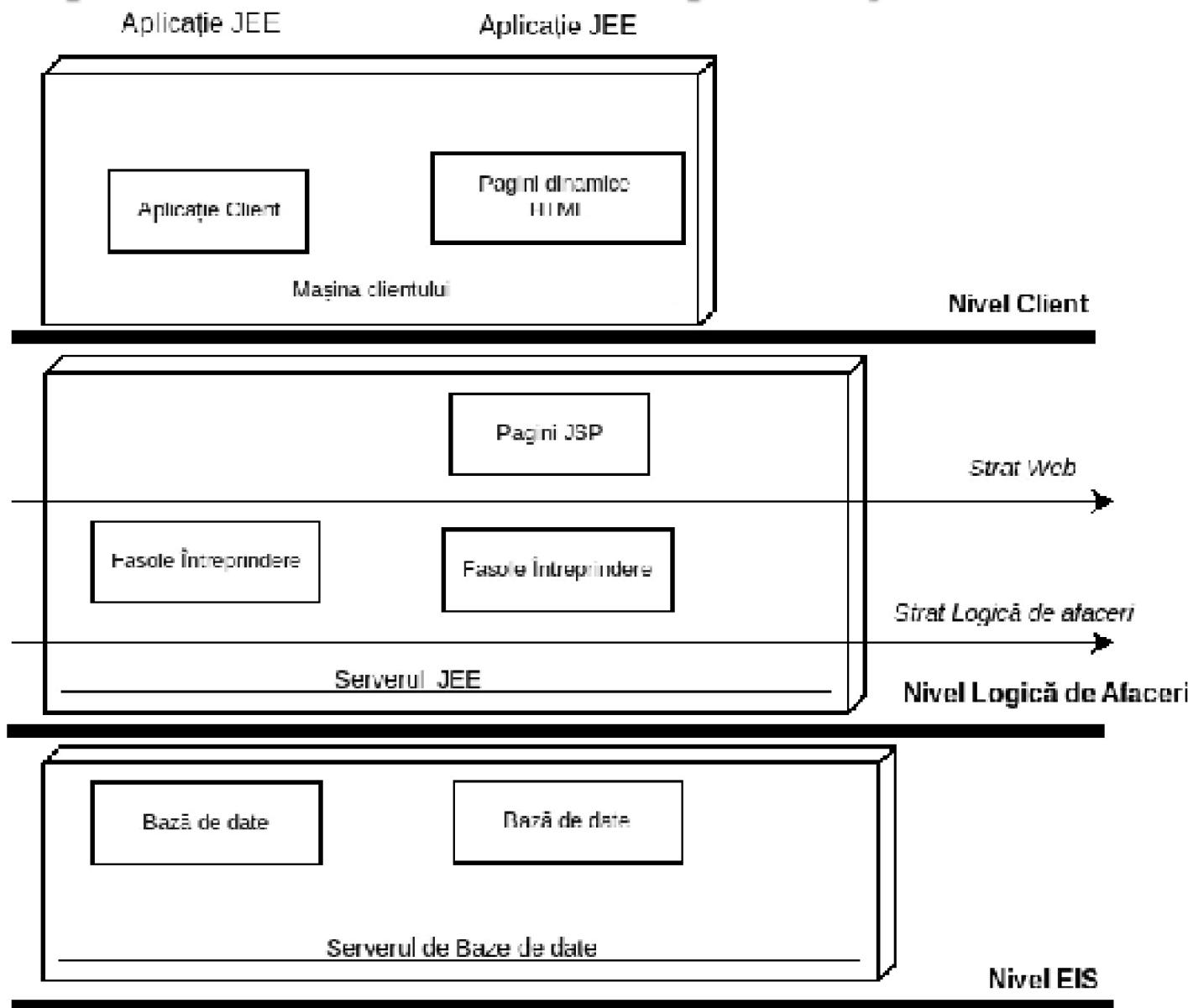
Arhitectura JEE

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

Arhitectura JEE



Componentele unei aplicații Java EE

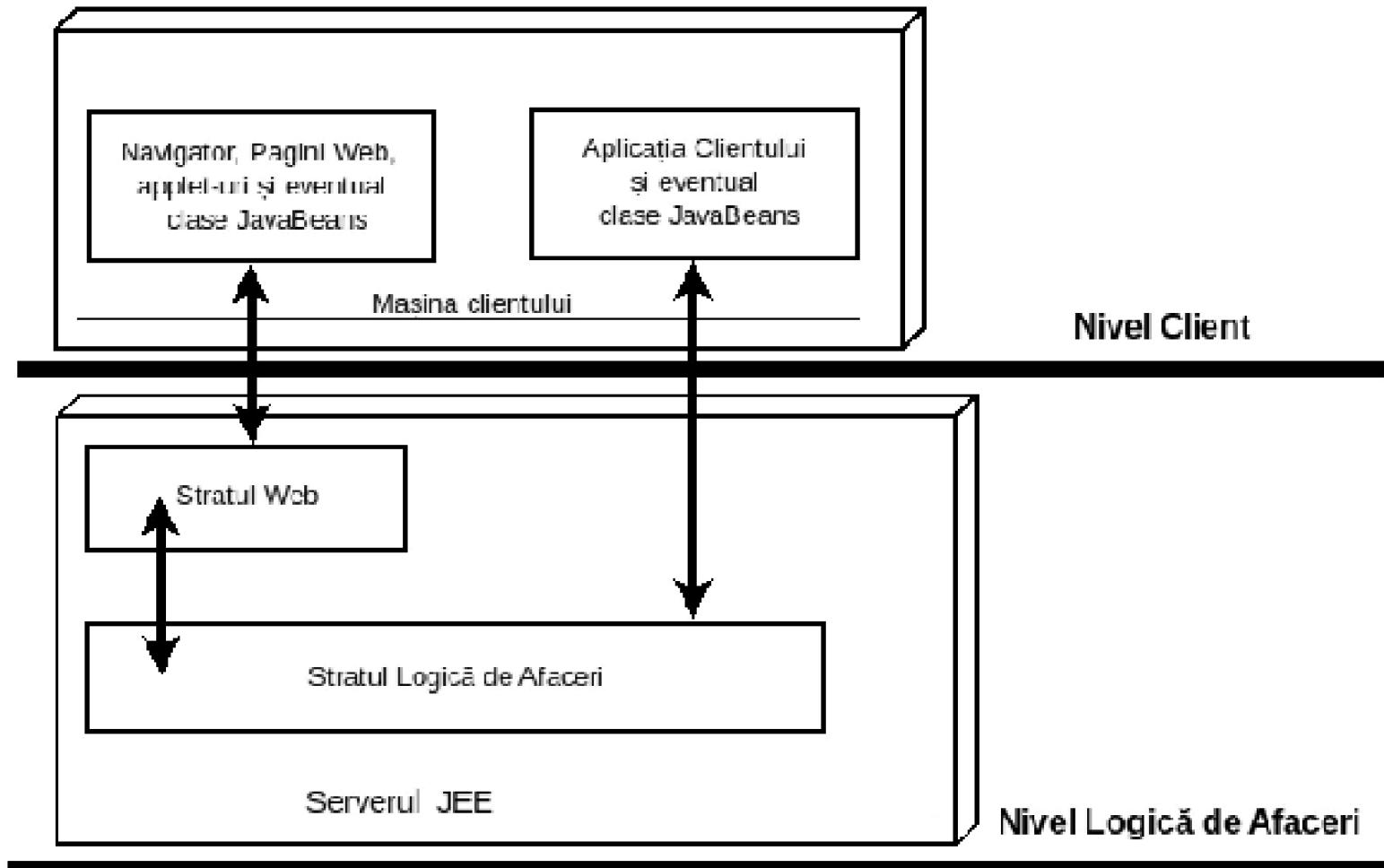




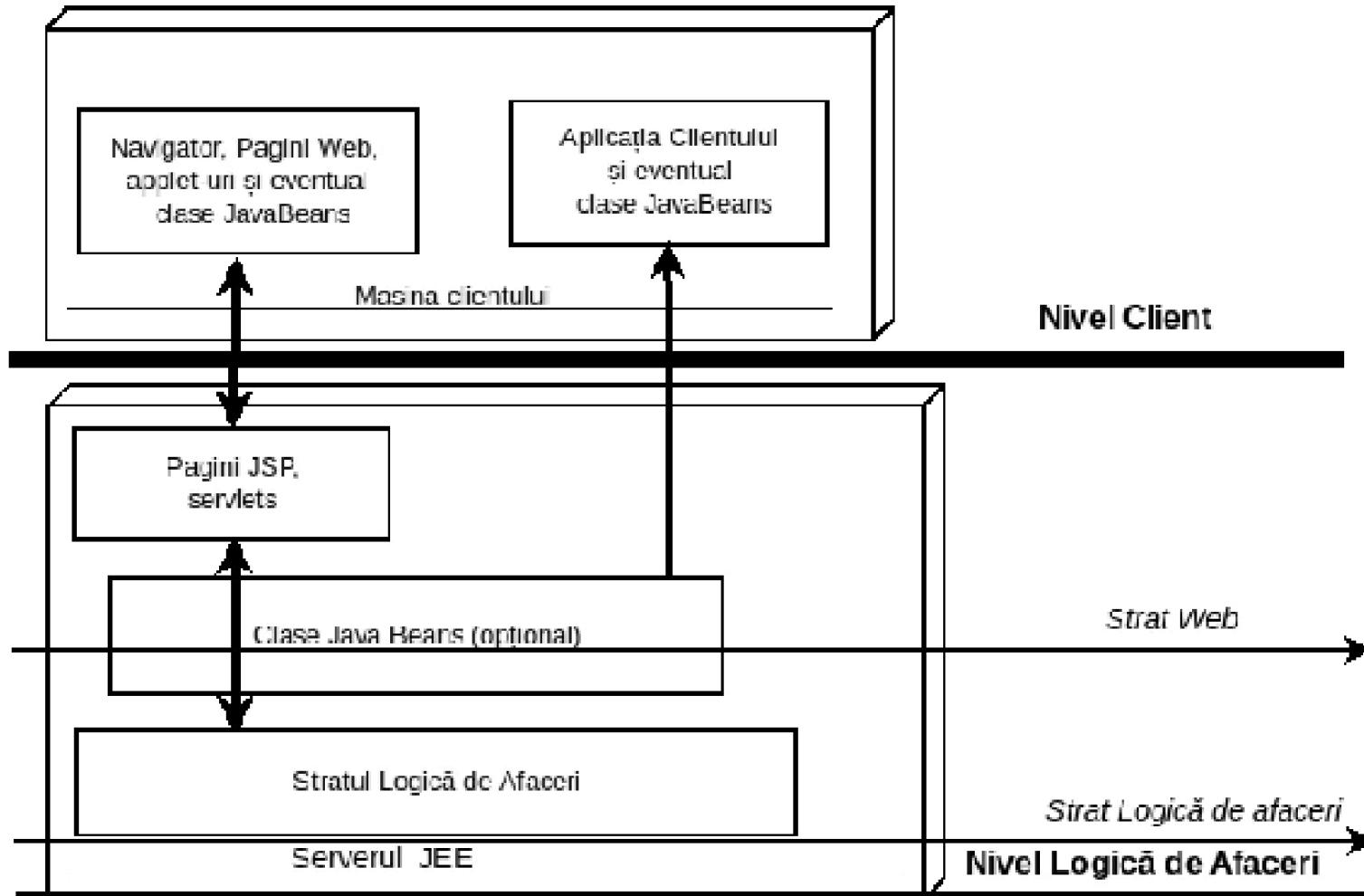
Sisteme Distribuite

Mihai Zaharia
Cursul 2

Comunicarea cu serverul Java EE

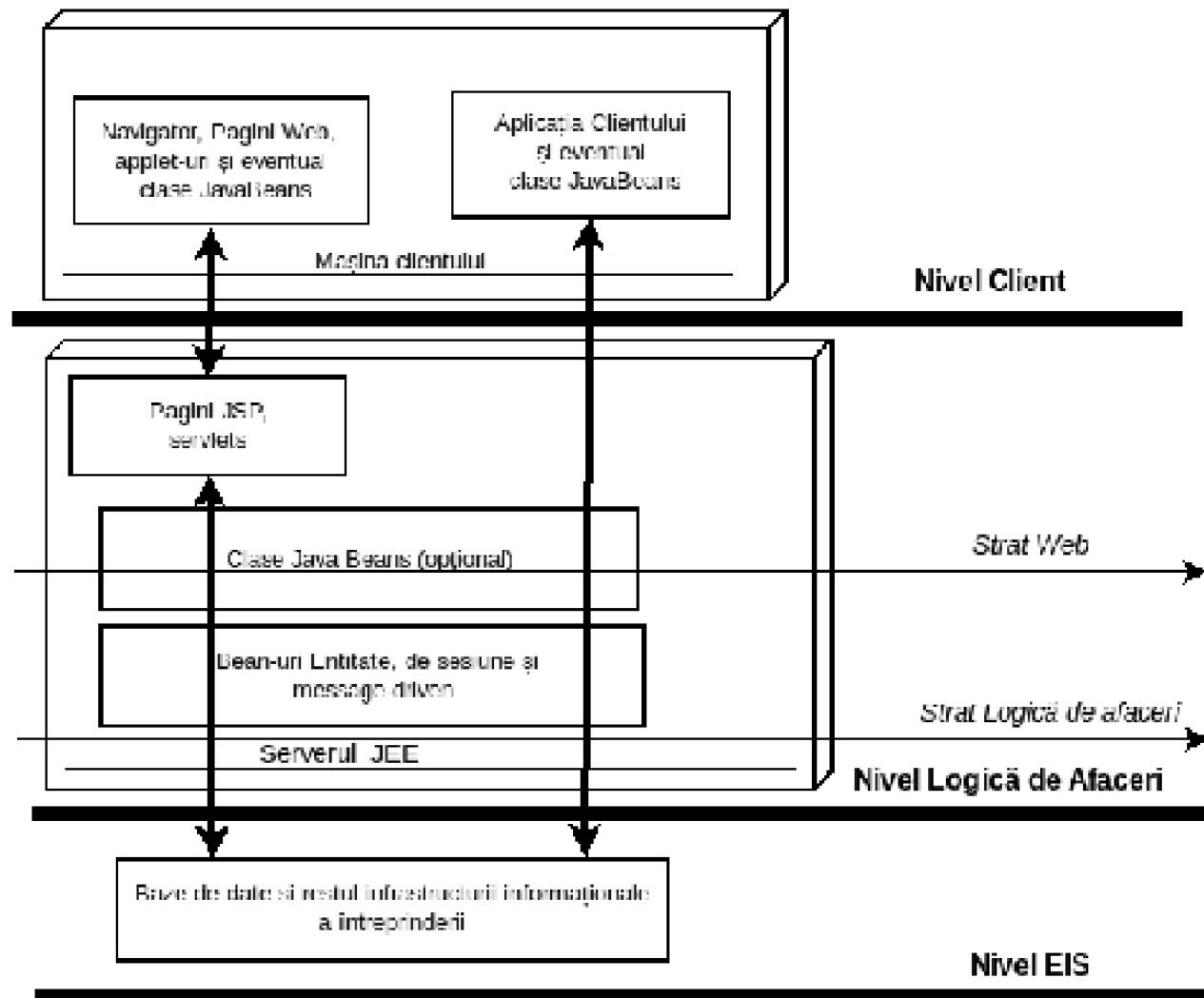


Stratul de Web al aplicației JEE

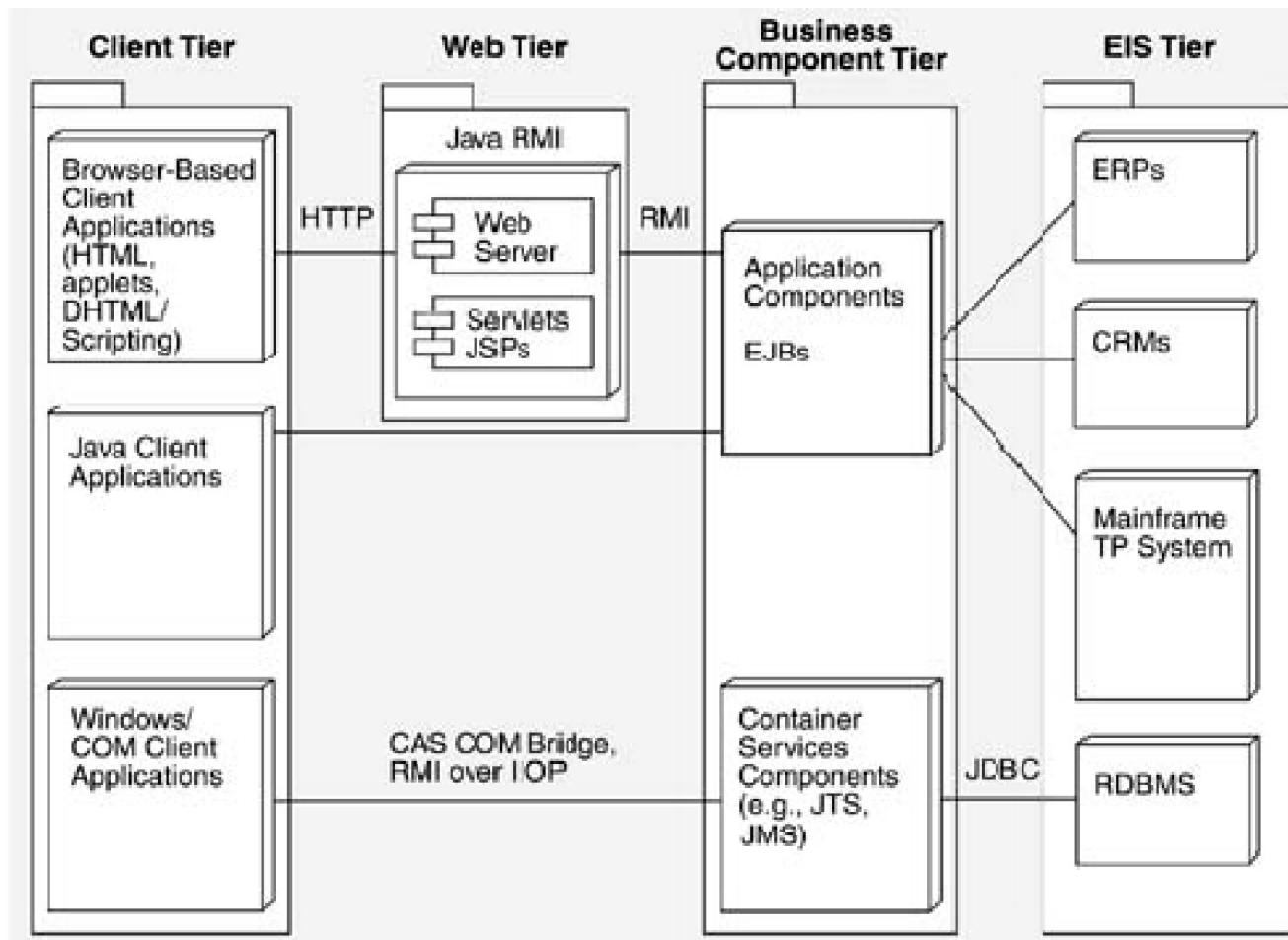


Clienți minimali “Thin” și Componentele Web

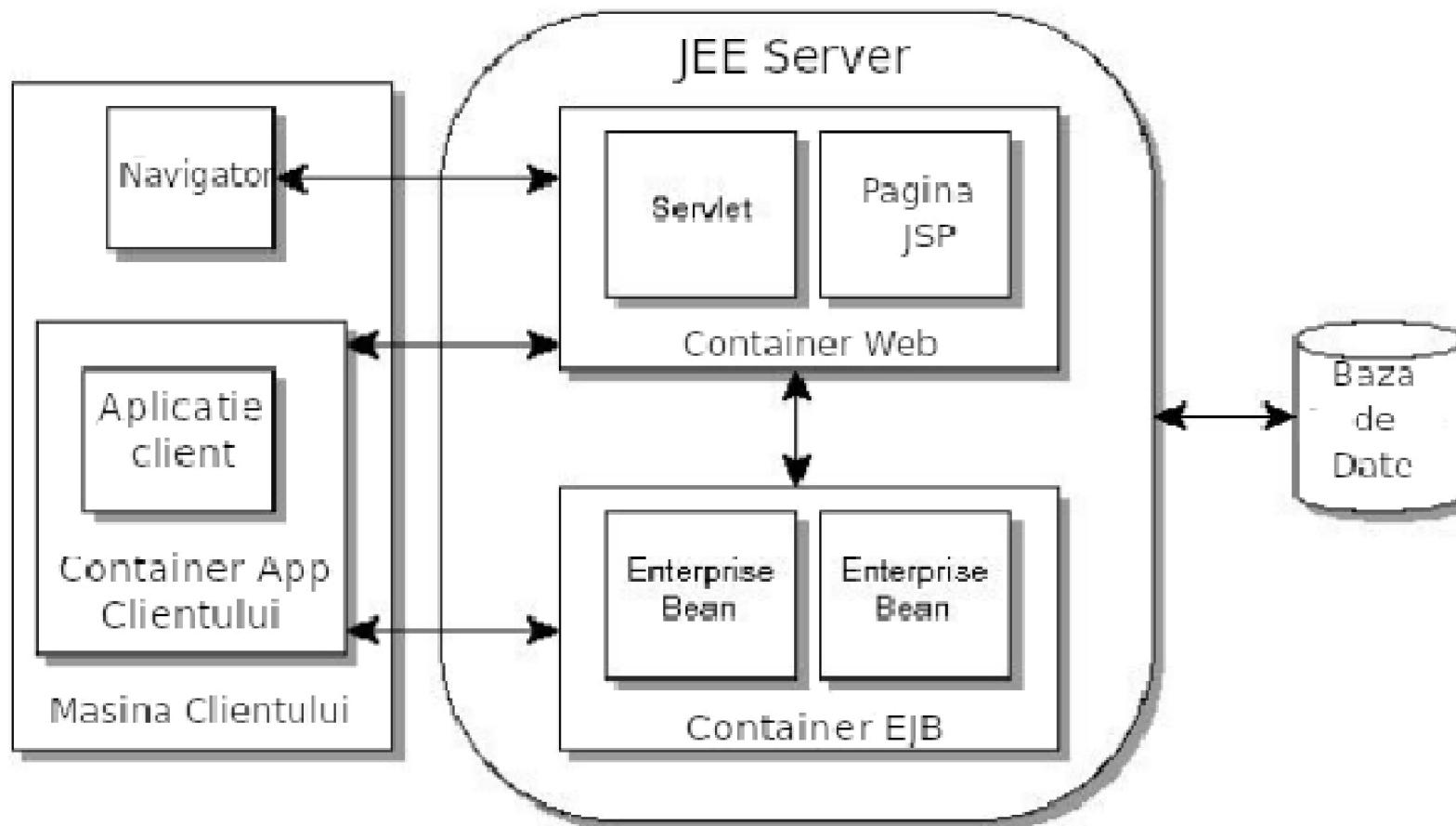
Componentele de afaceri



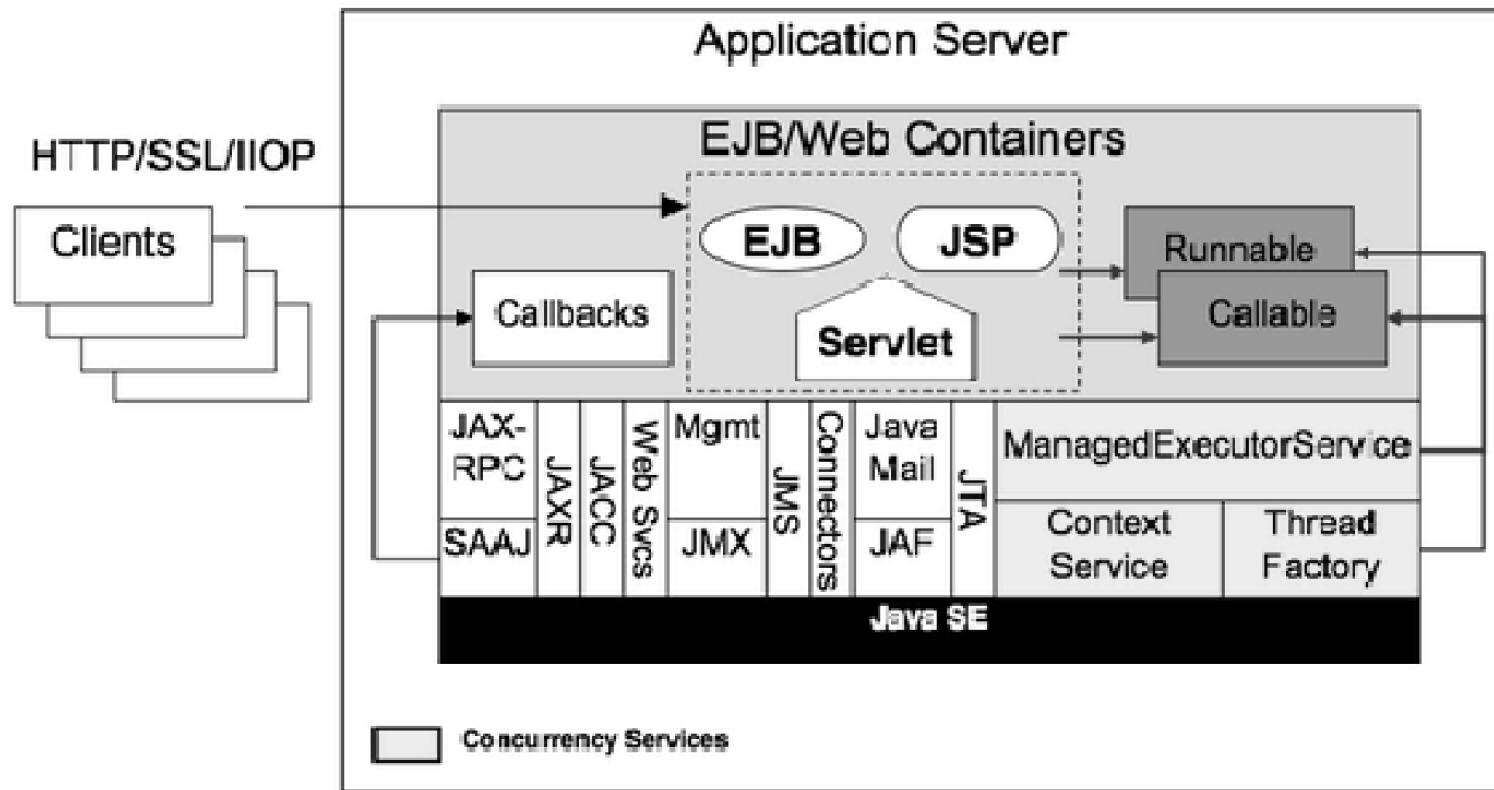
Sistemul informational al întreprinderii (Enterprise Information System)



Containere și Servicii



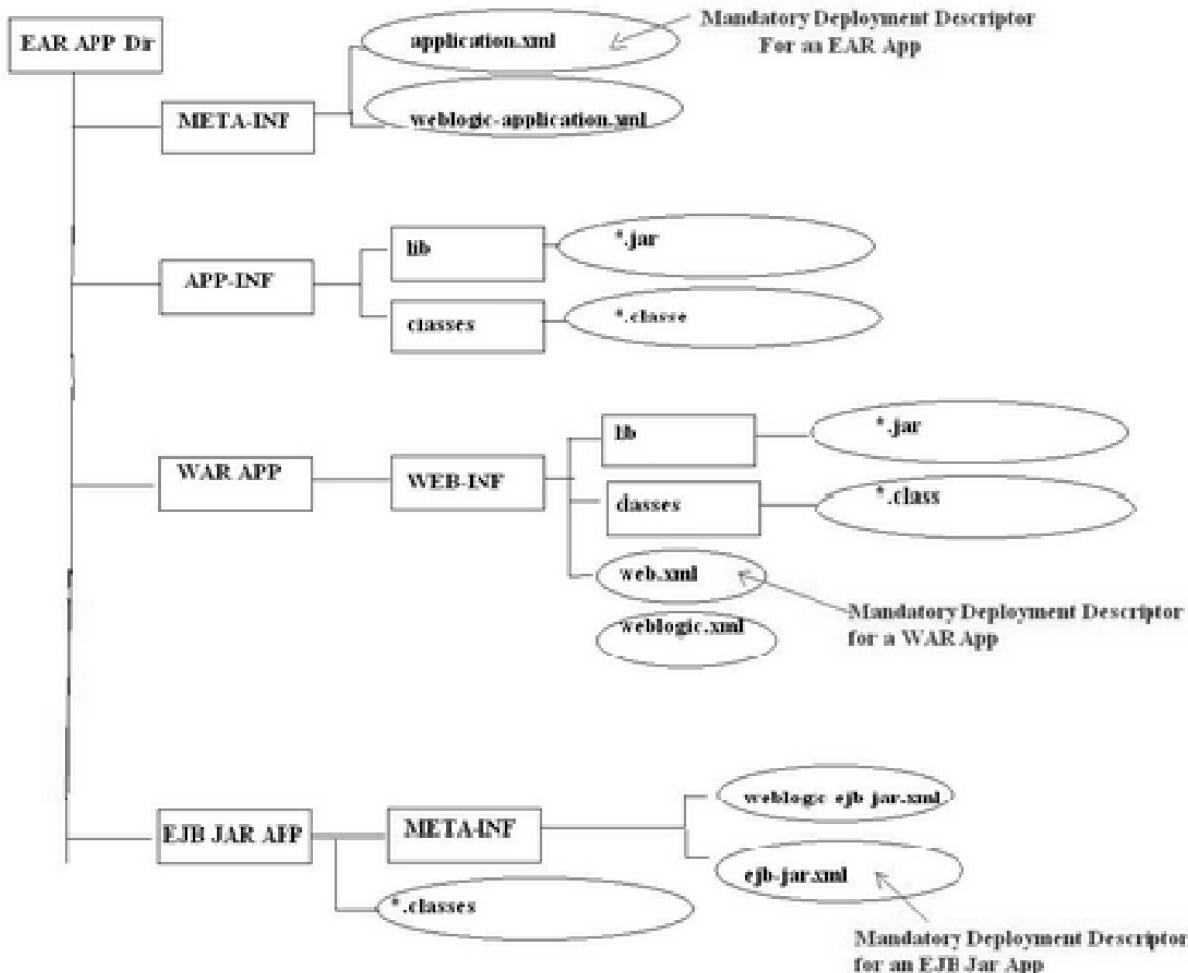
Containere



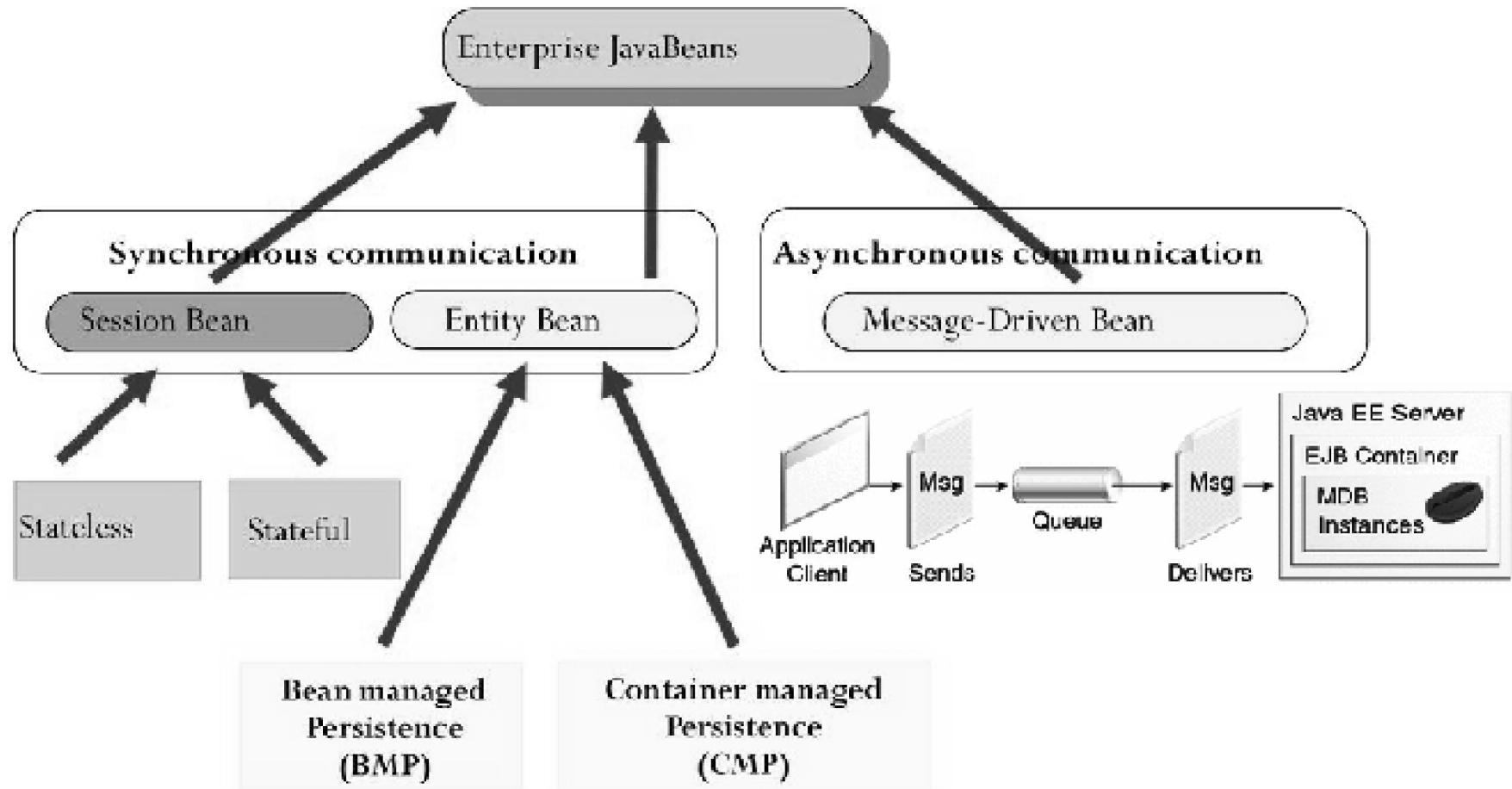
utilizând concurență

Împachetarea componentelor

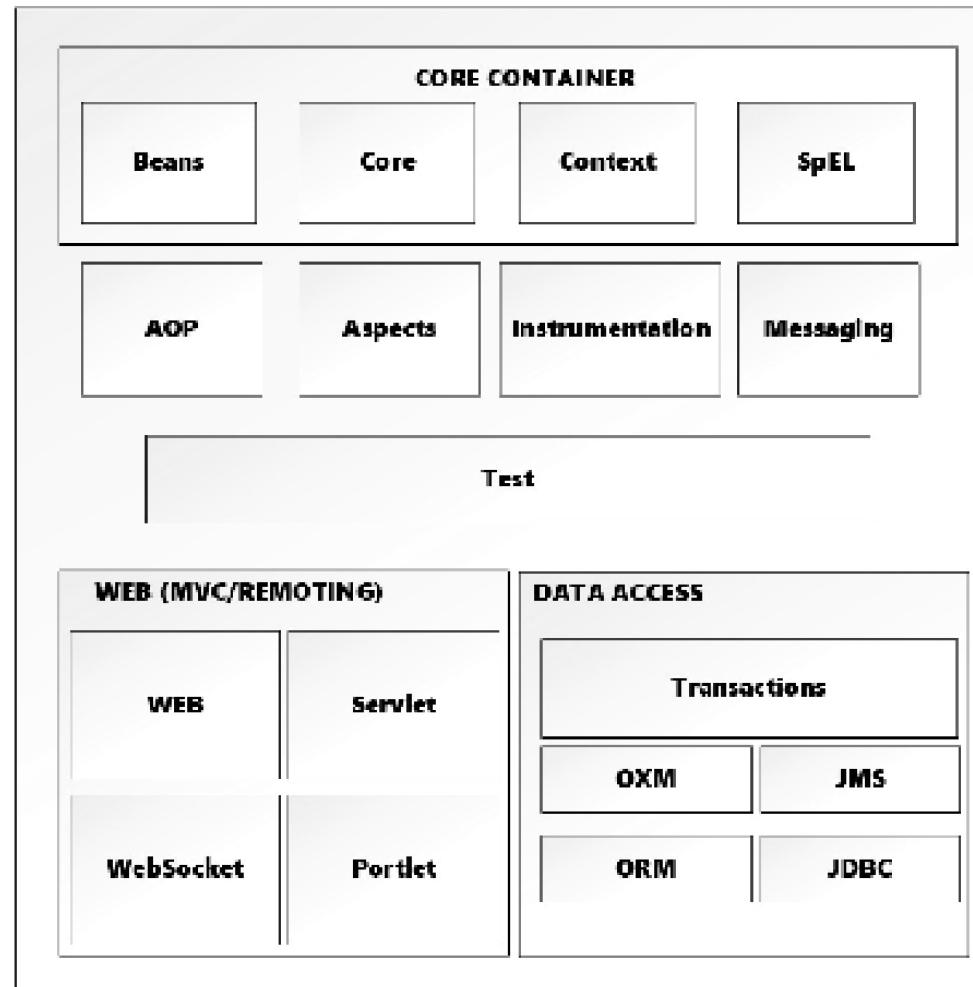
- Jar
- Ear
- War



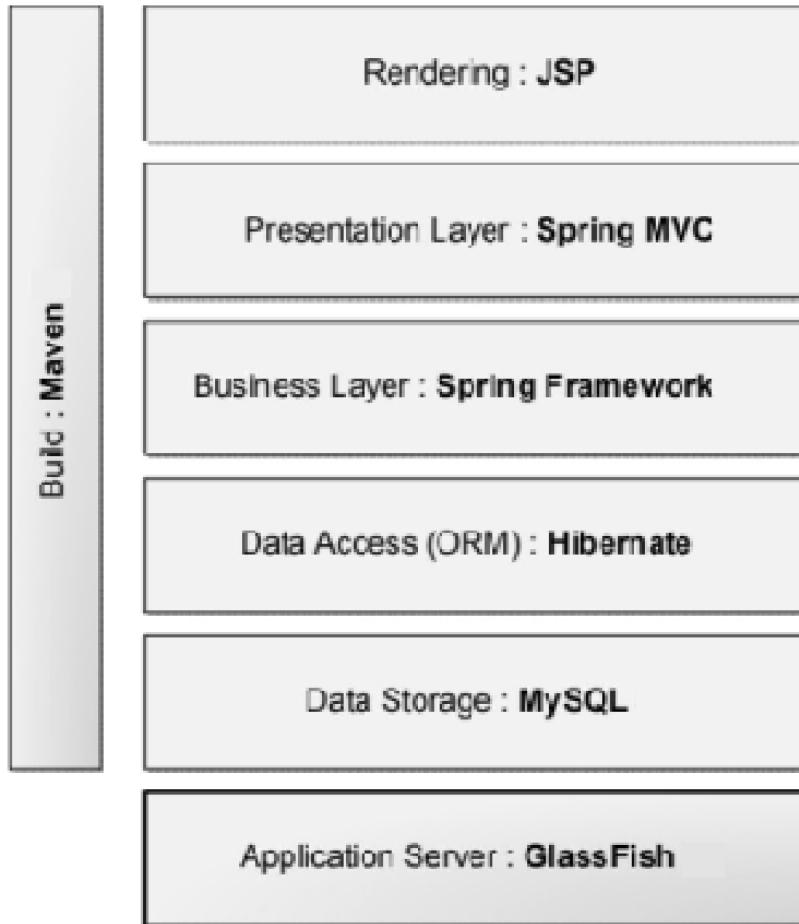
Tipuri de ejb



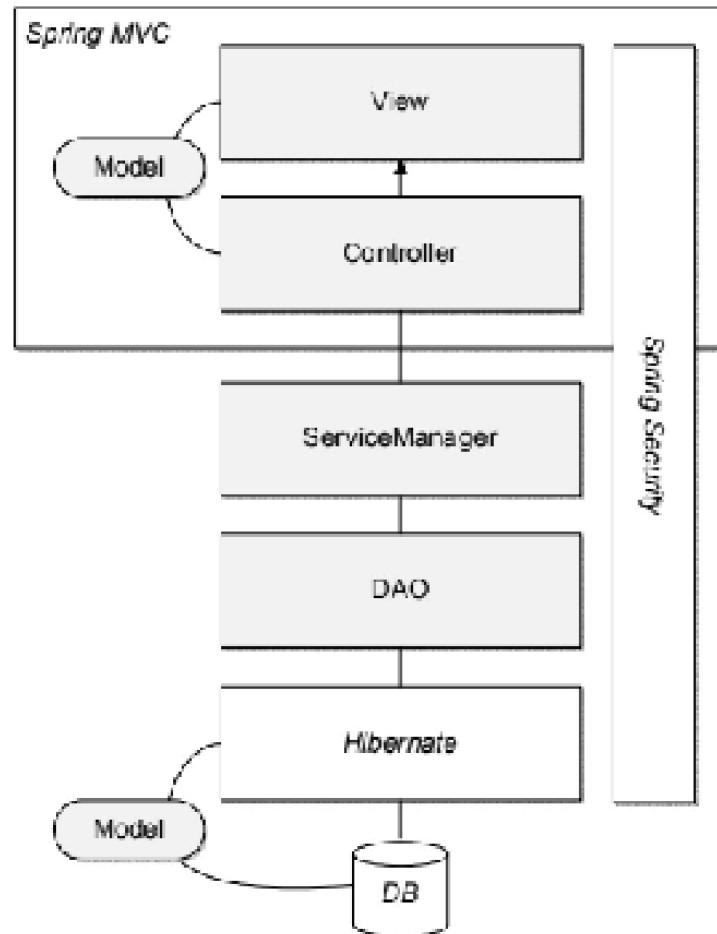
Framework-ul Spring



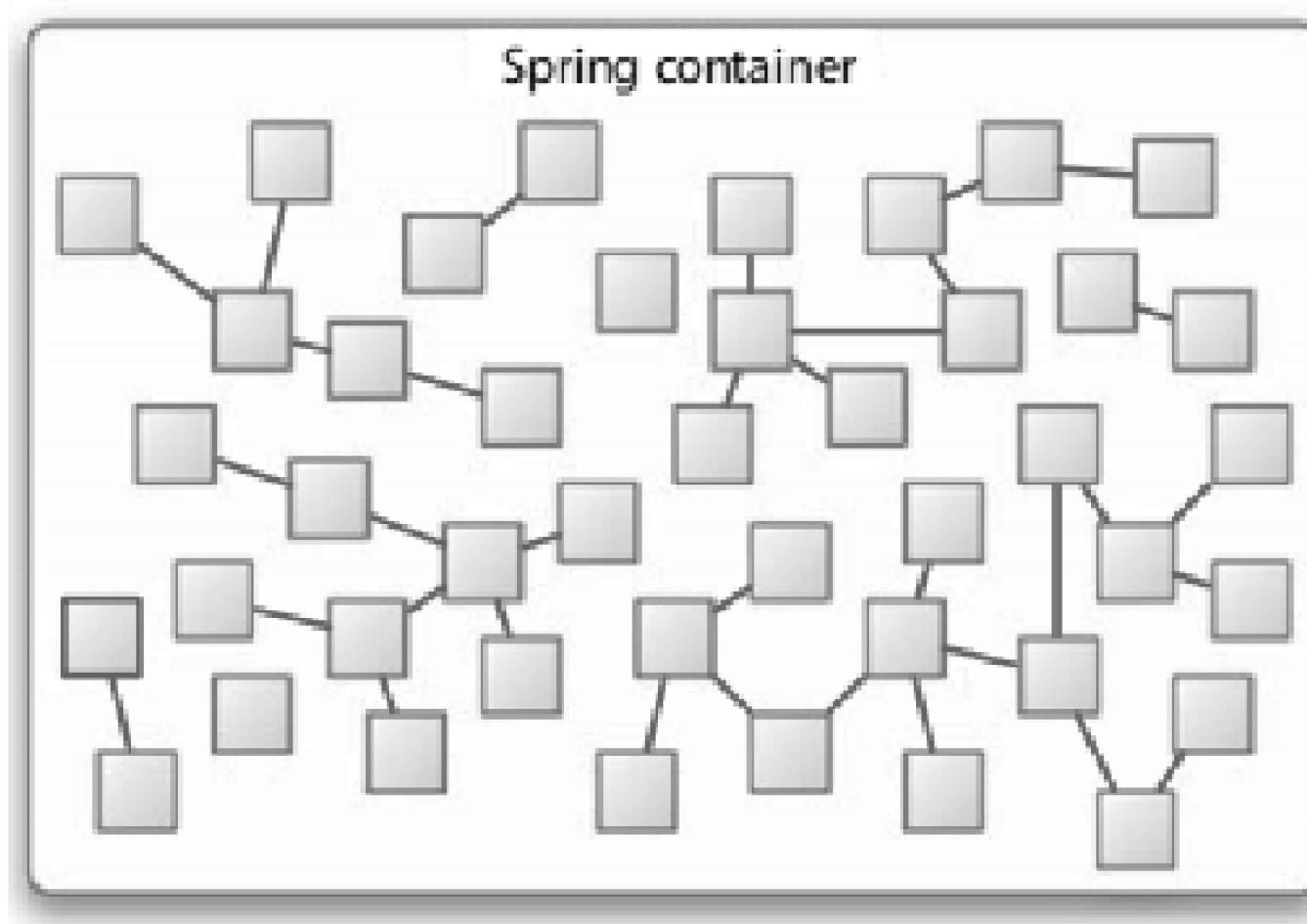
Ce-i cu Stack-urile astea ?



Arhitectura care a condus la stive tehnologice specifice Spring



Containere Spring



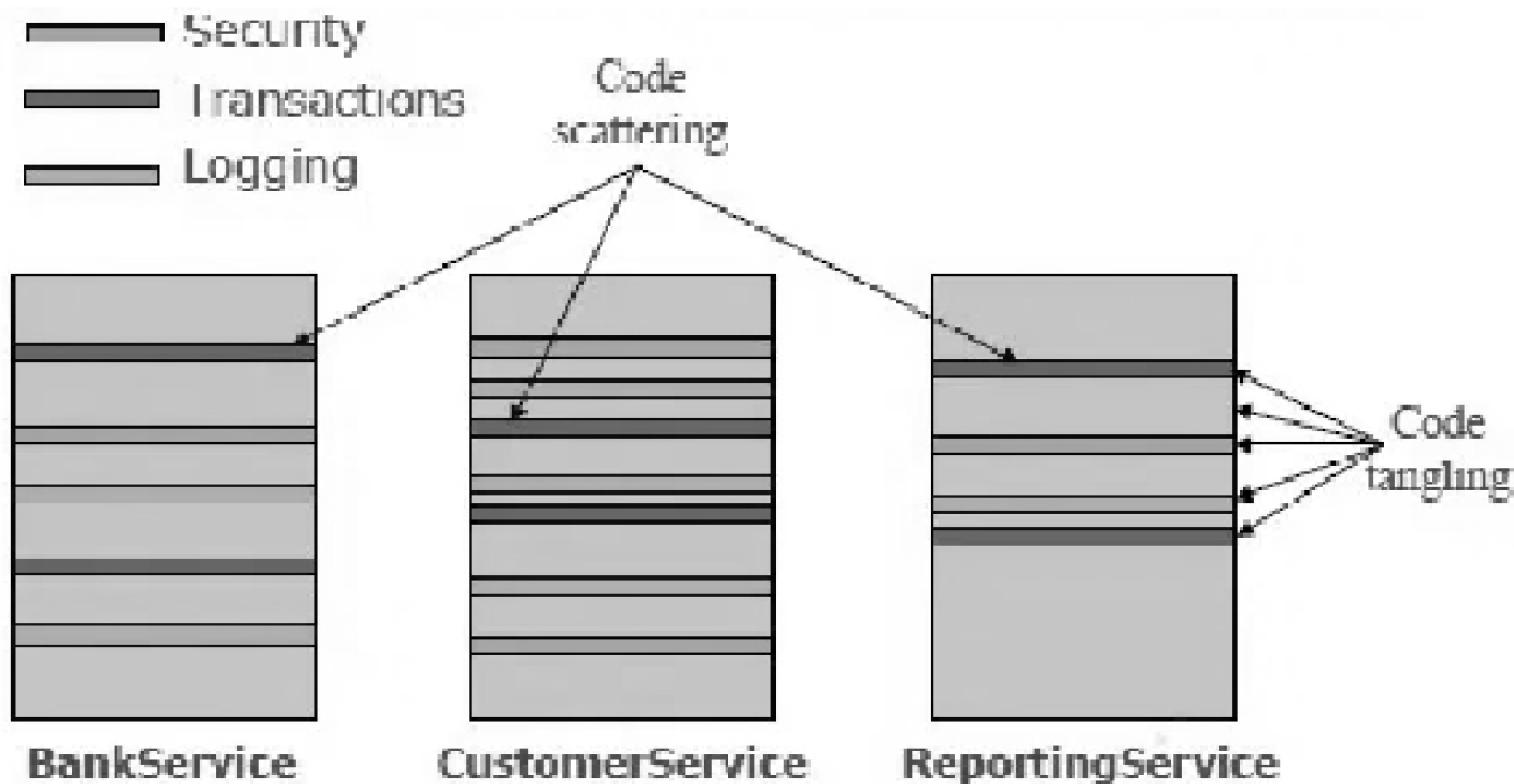
Contexte predefinite pentru aplicații

- ClassPathXmlApplicationContext
- FileSystemXmlApplicationContext
- XmlWebApplicationContext

```
ApplicationContext context = new  
FileSystemXmlApplicationContext("c:/foo.xml");
```

```
ApplicationContext context = new  
ClassPathXmlApplicationContext("foo.xml");
```

AOP



SOA

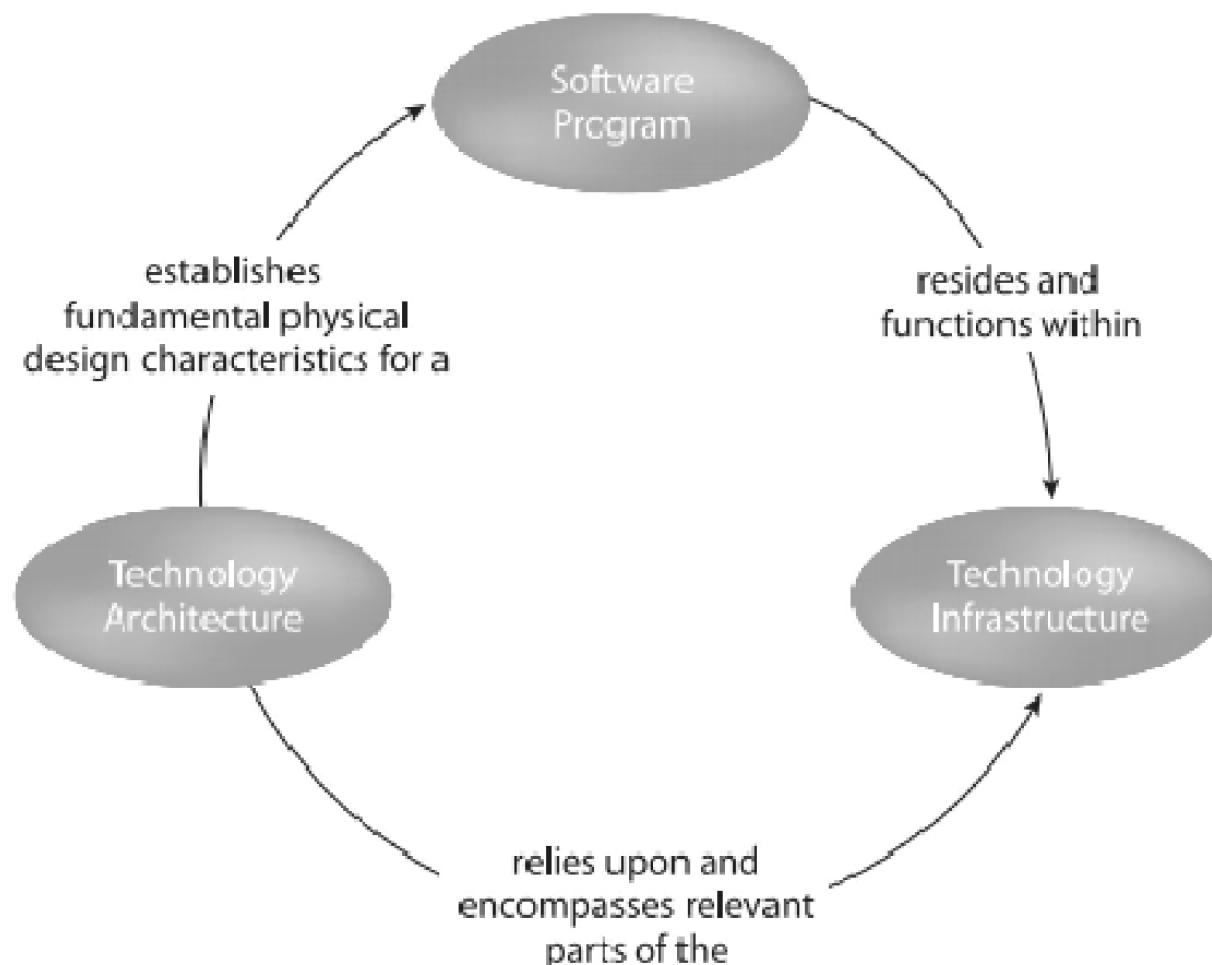


Sisteme Distribuite

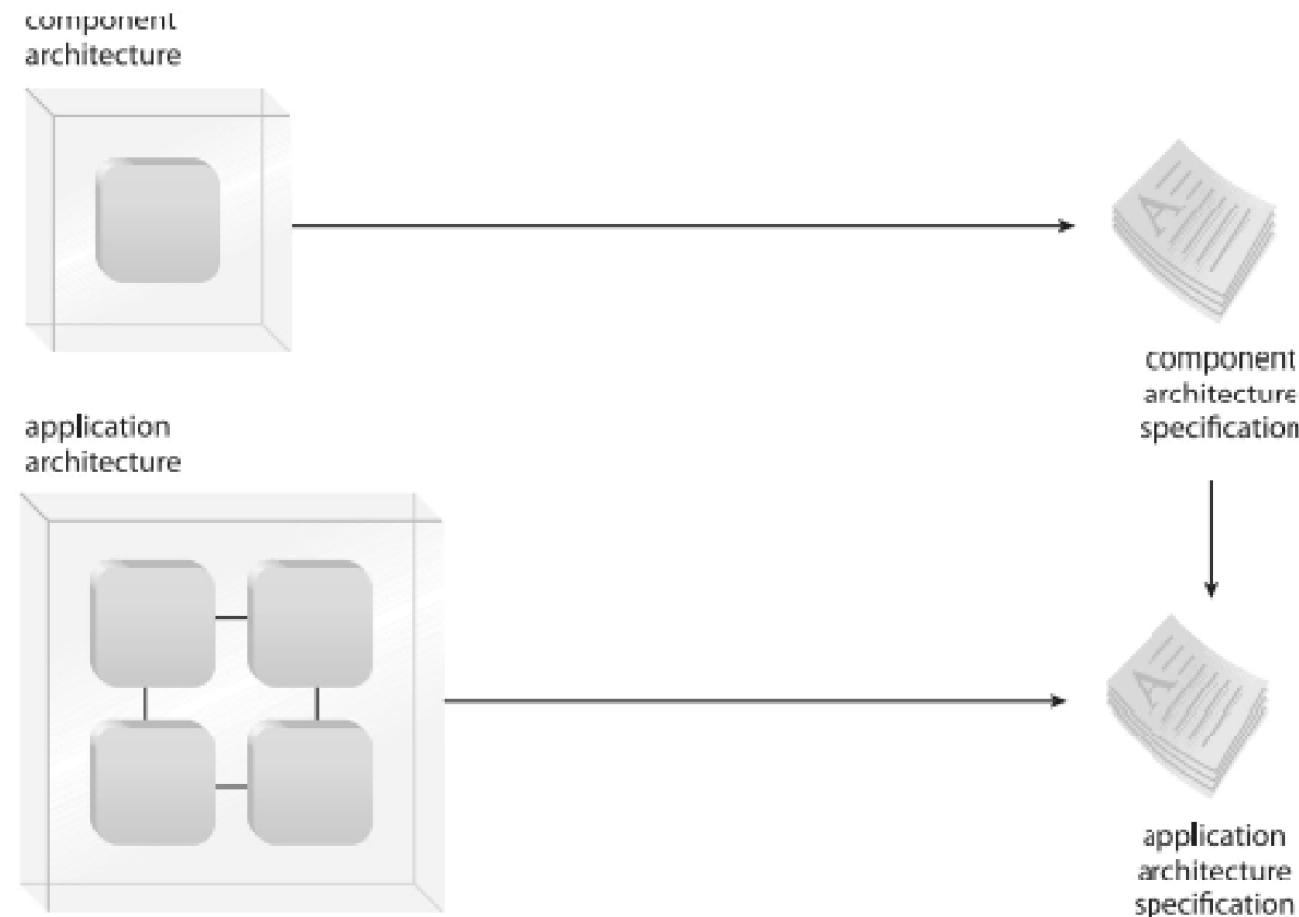
Mihai Zaharia

Cursul 3

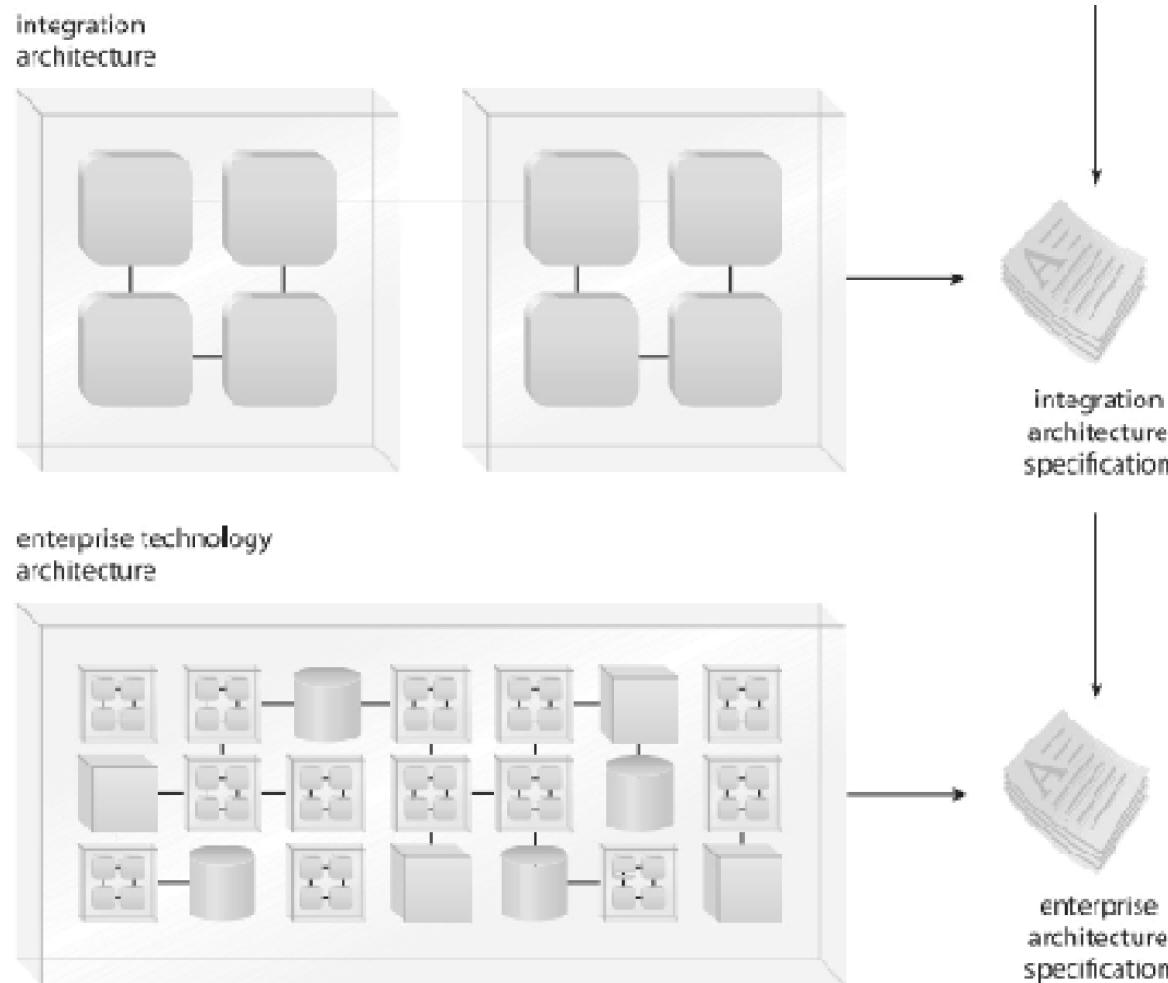
Terminologie - Arhitectura tehnologiei



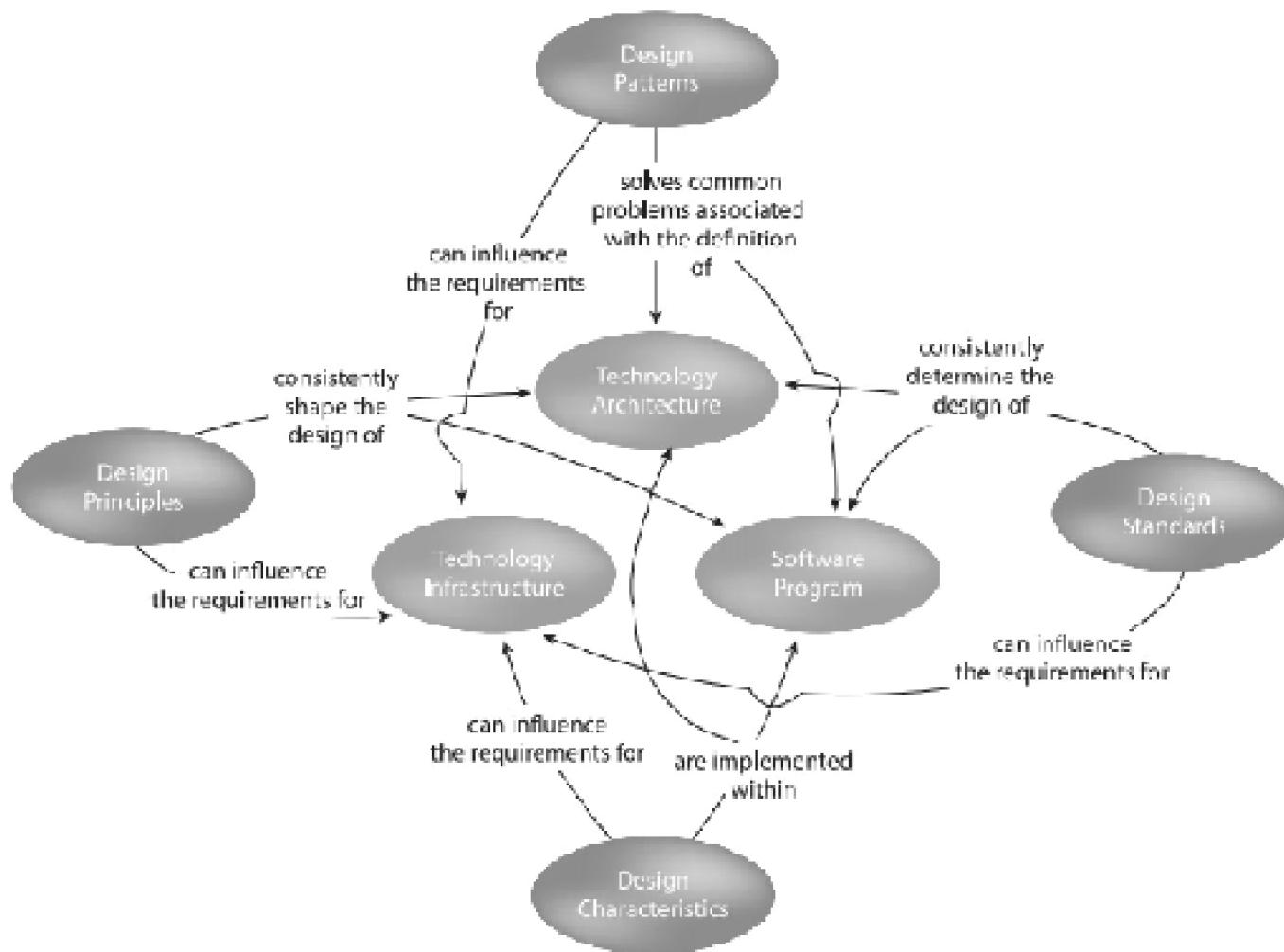
Tipurile arhitecturii tehnologiei



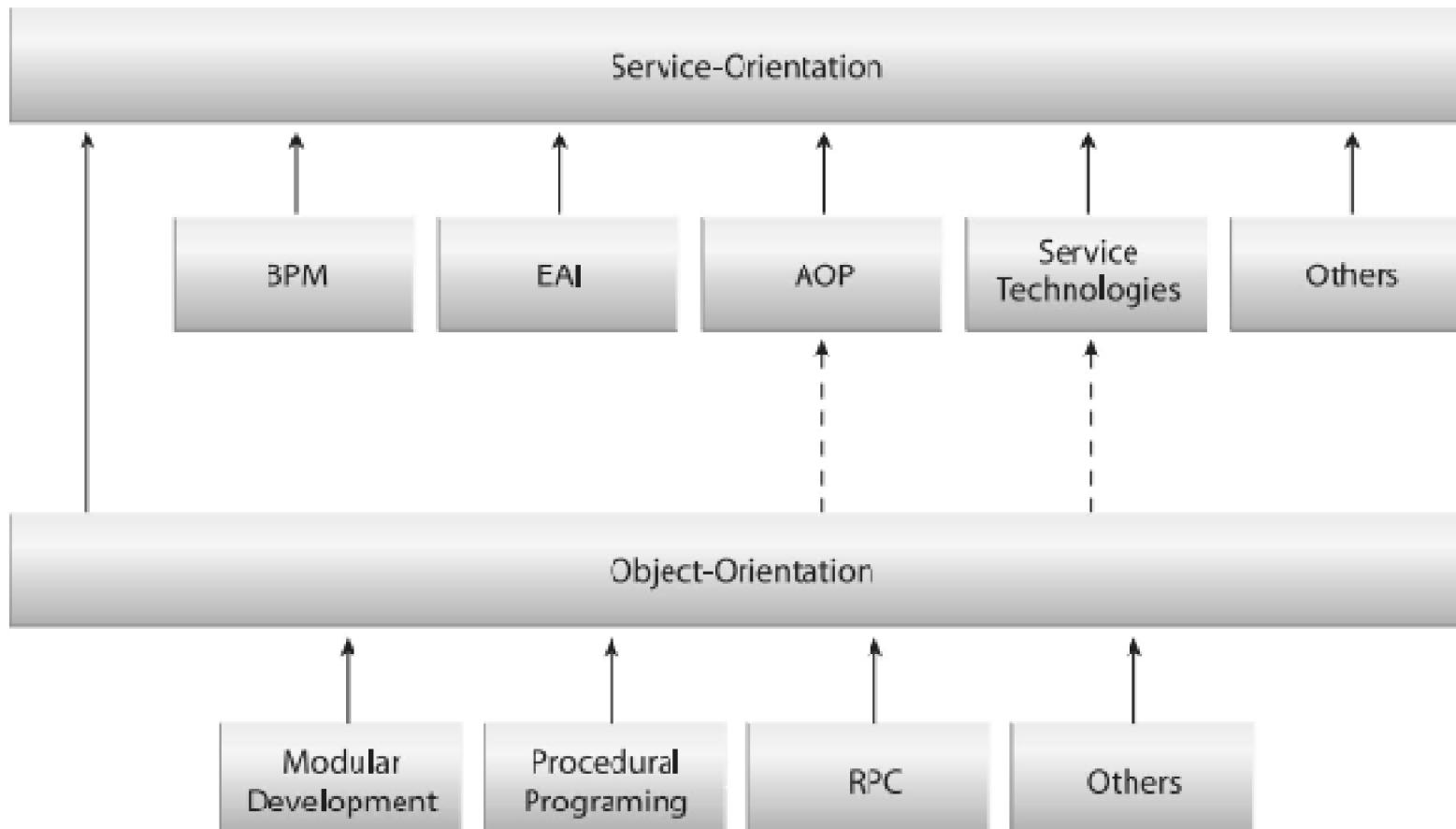
Tipurile arhitecturii tehnologiei



Relații între termenii discuți anterior



Arhitectura orientată pe servicii



Serviciu

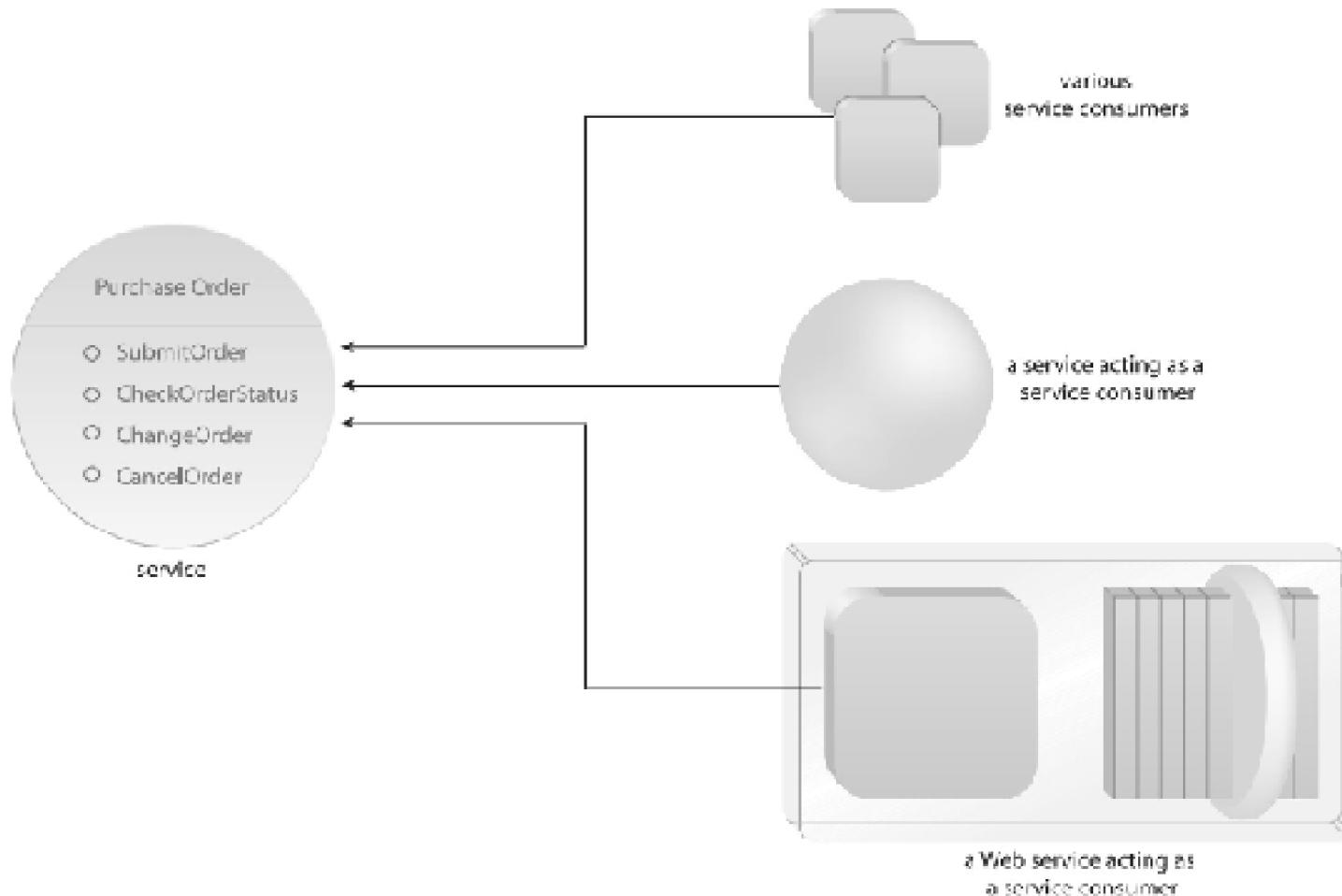
Purchase Order

- SubmitOrder
- CheckOrderStatus
- ChangeOrder
- CancelOrder

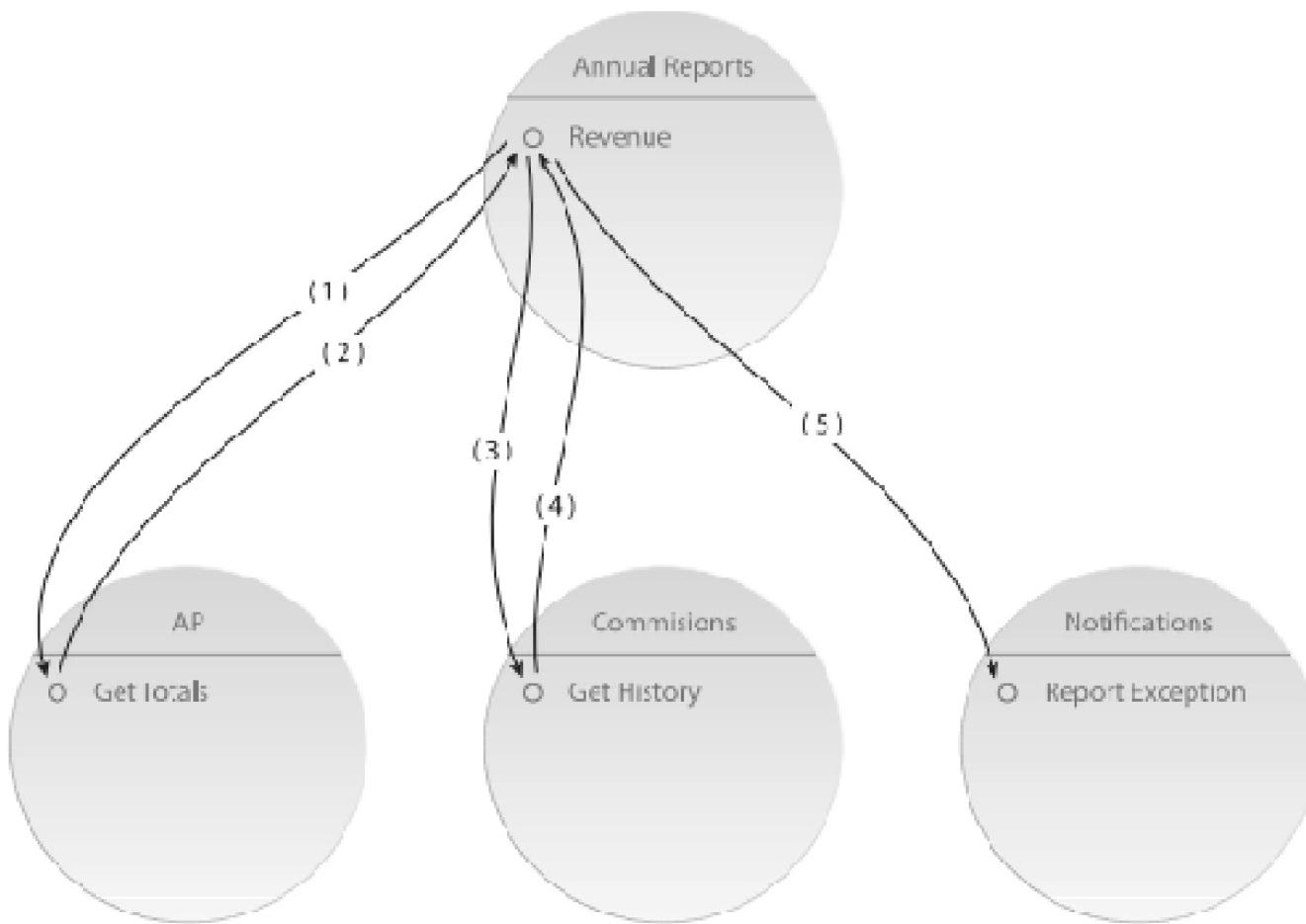
SWOT SOA - Avantaje

- **Reutilizarea serviciilor**
- **Usor de menținut**
- **Independentă platformă**
- **Disponibilitate**
- **Fiabilitate**
- **Scalabilitate**

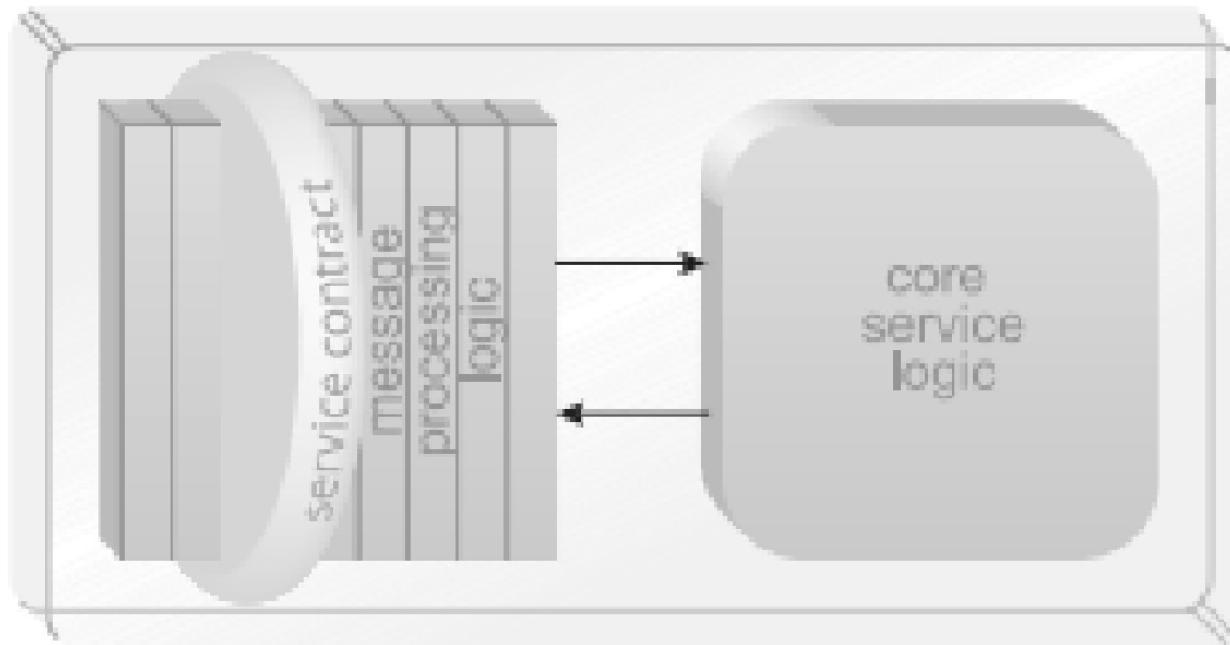
Utilizatorul/Beneficiarul/Consumatorul Serviciului



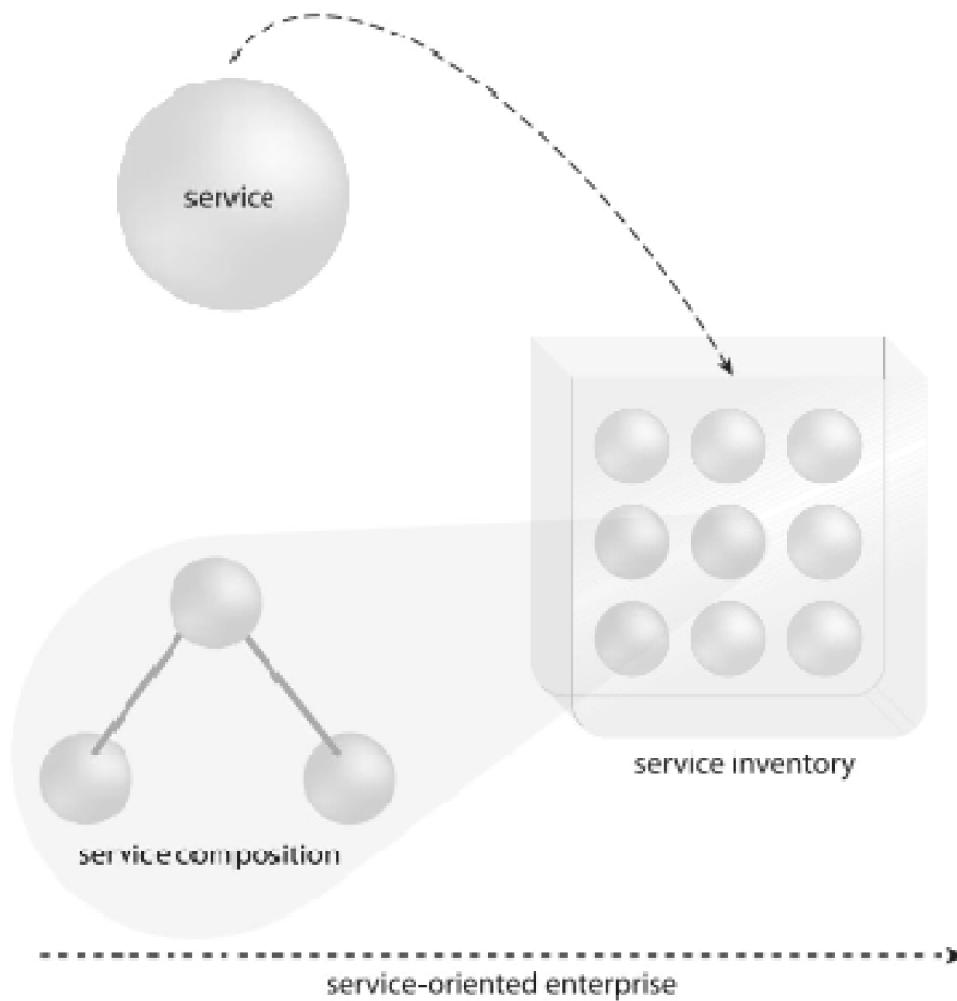
Compunerea Serviciilor



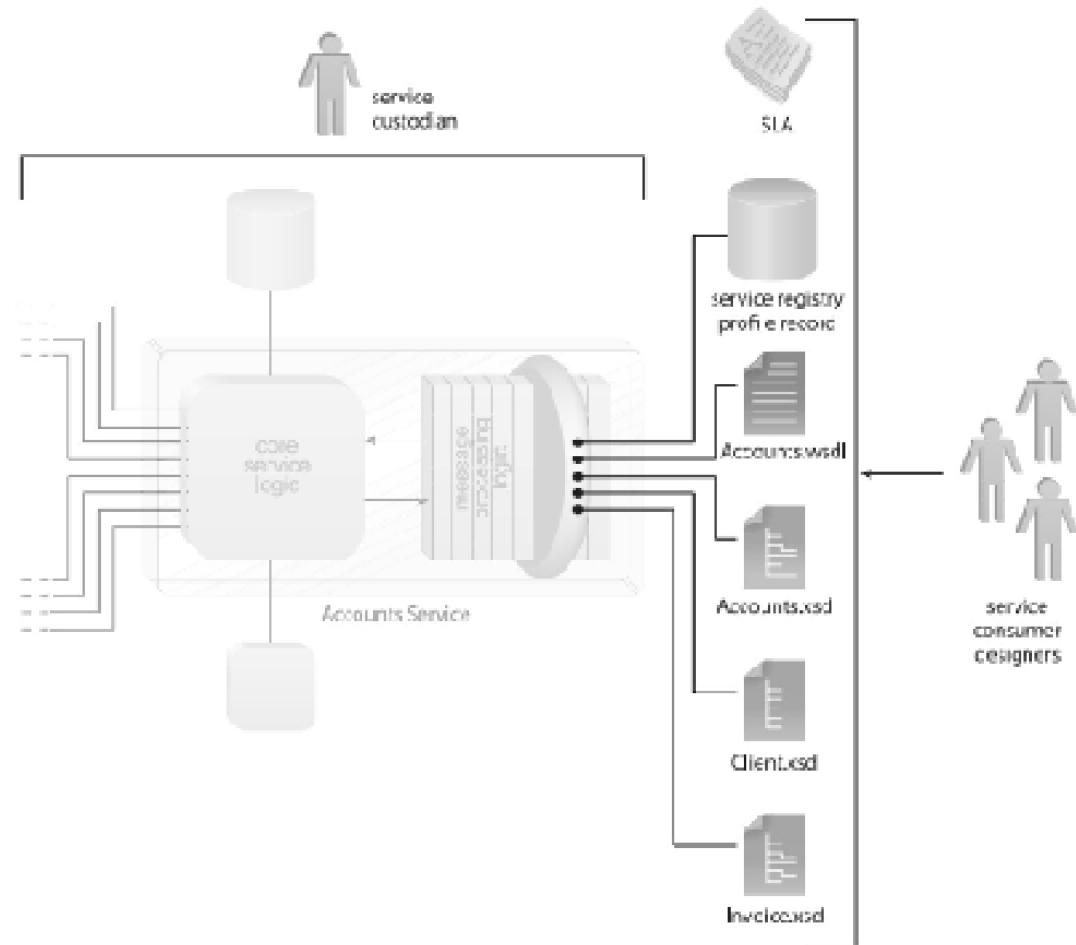
Servicii Web



Cele patru tipuri comune de SOA



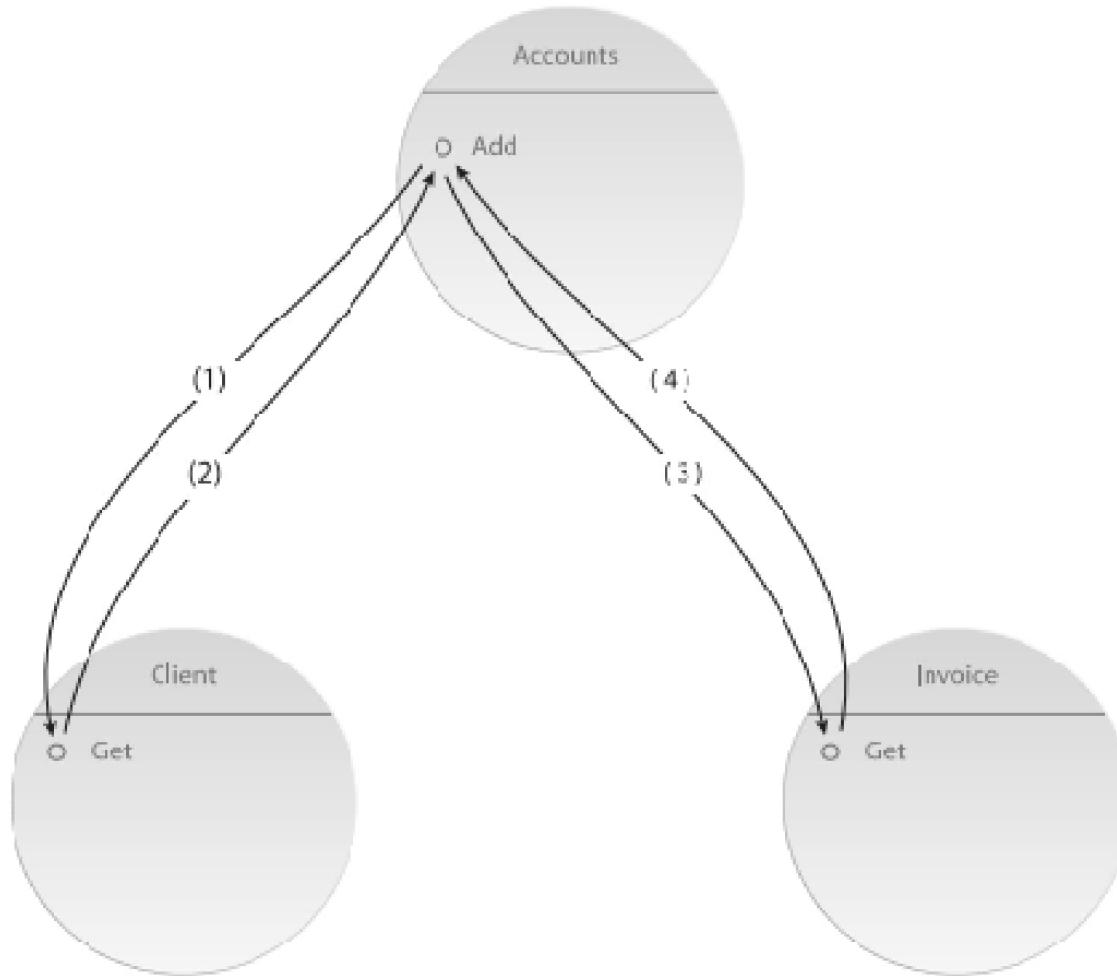
Detalii



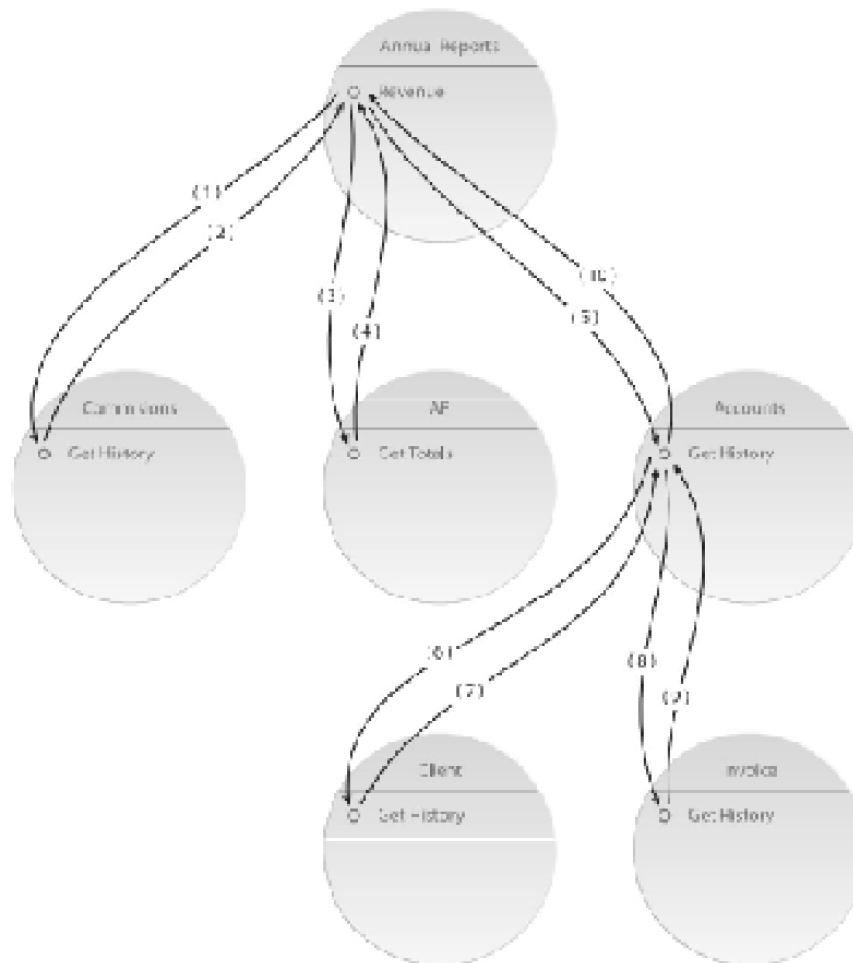
Abilitățile/funcționalitățile unui serviciu



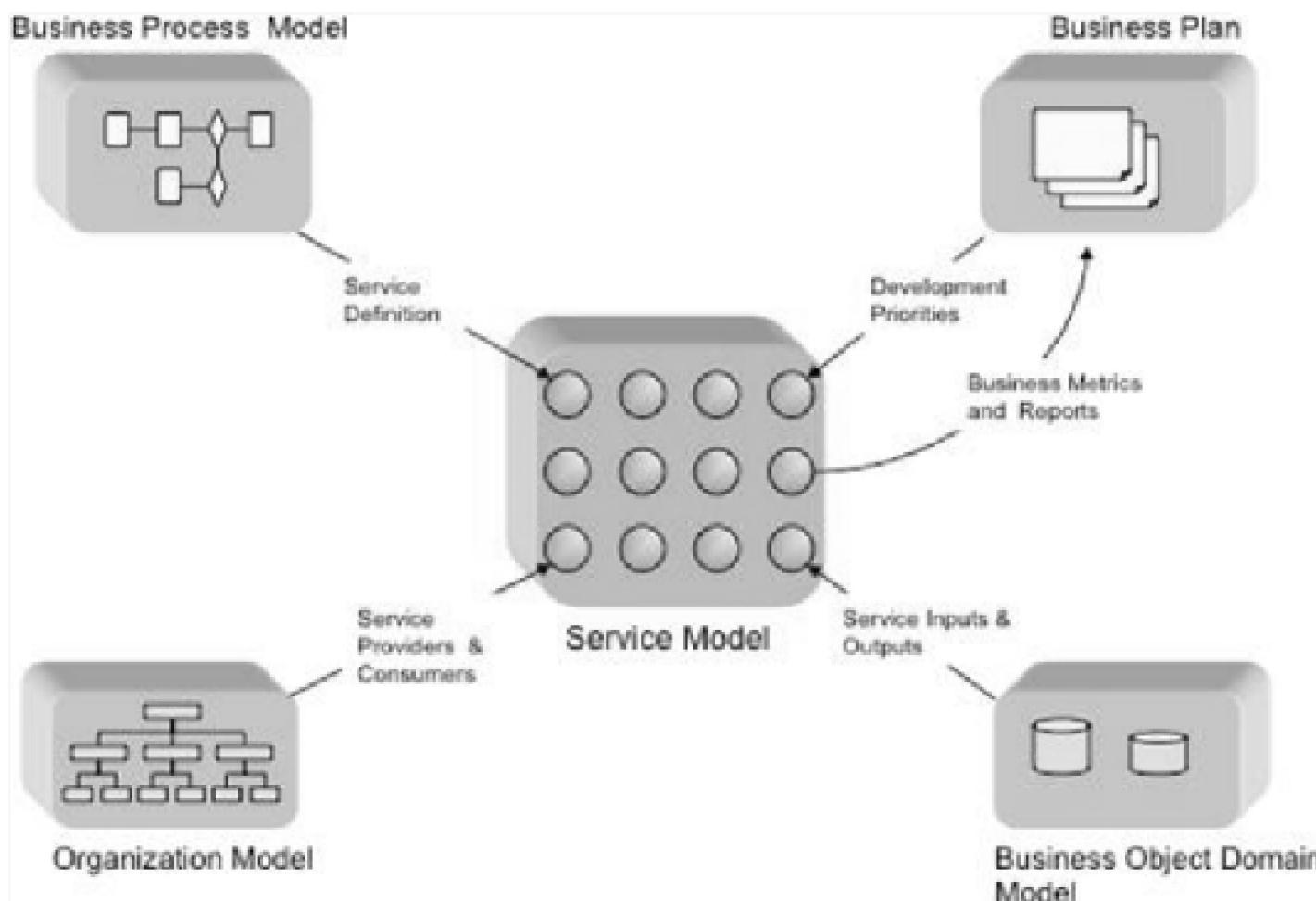
Arhitectura pentru compunerea serviciilor



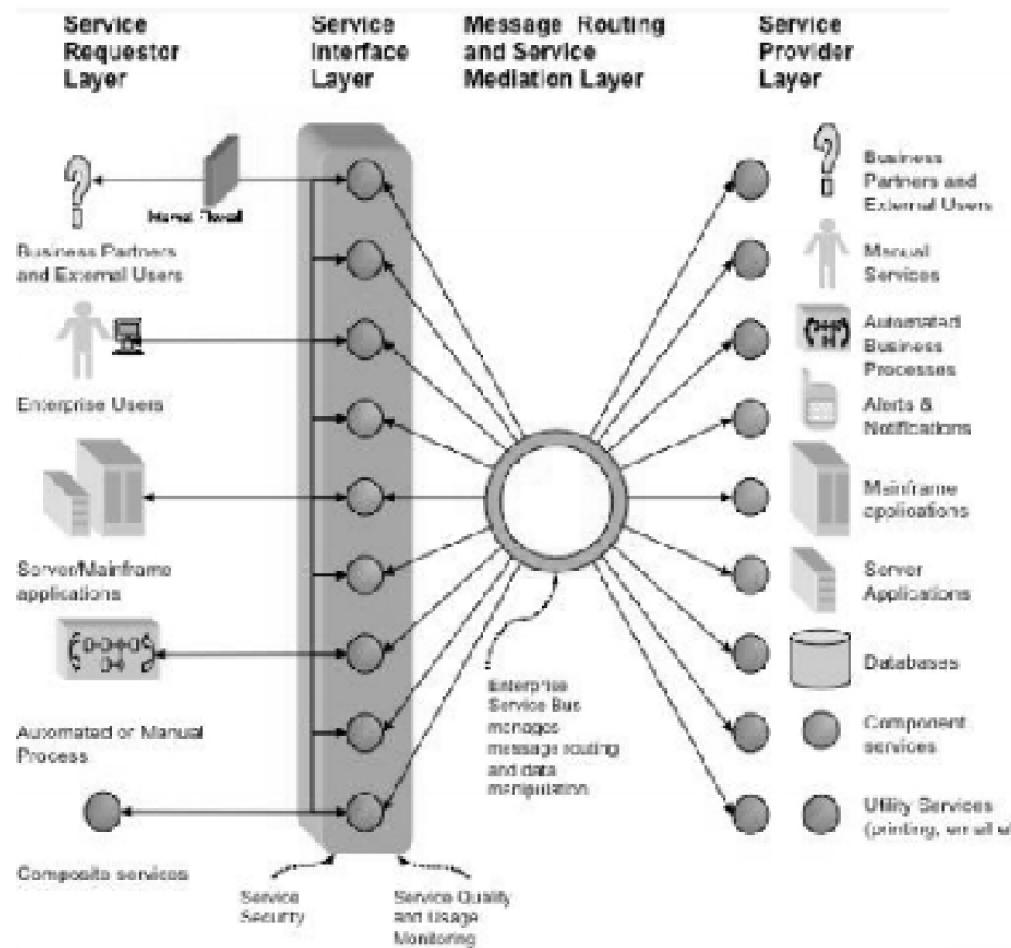
Compunere recursiva (Nested)



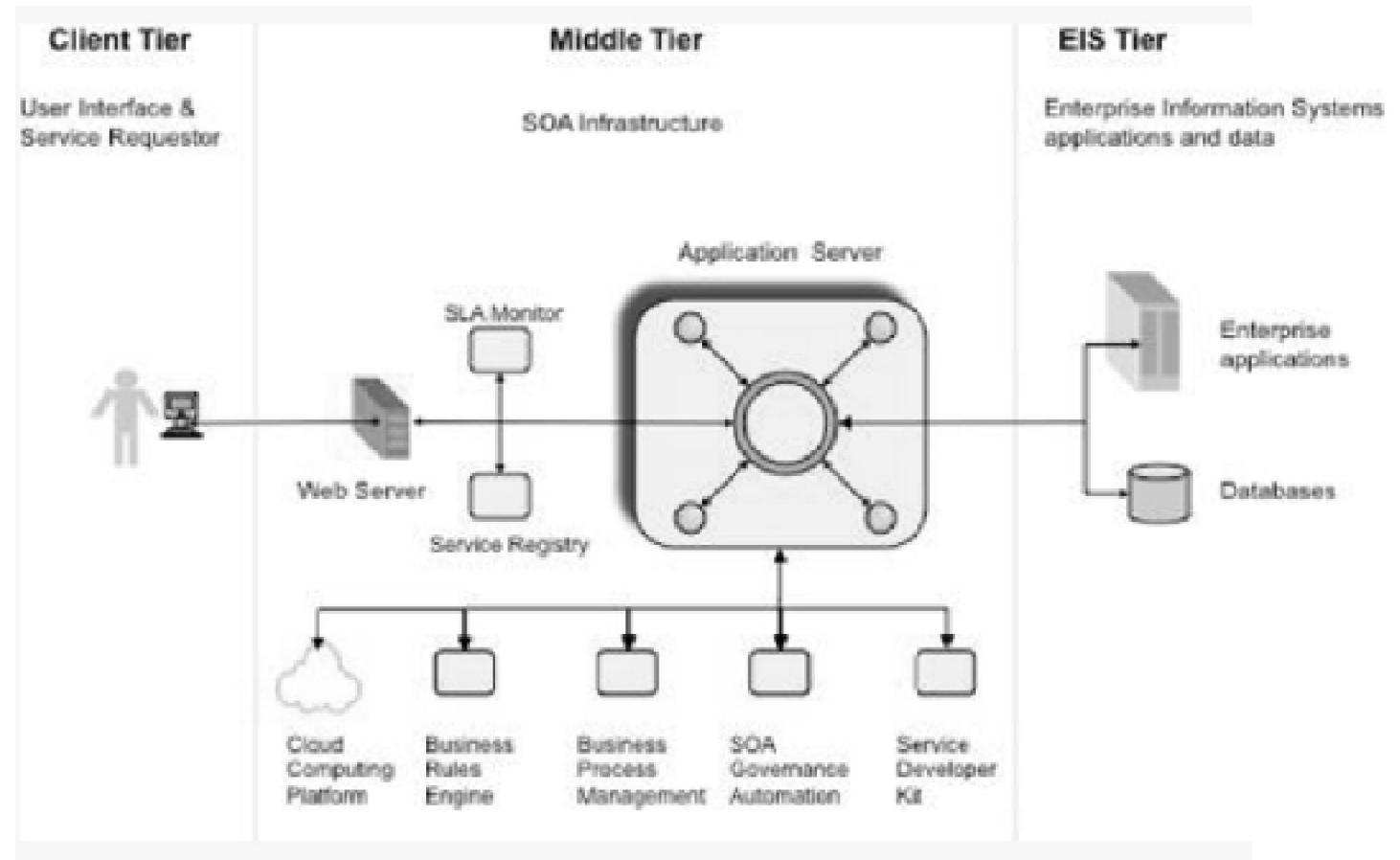
Relațiile între întreprindere și modelul creat și implementat utilizând SOA



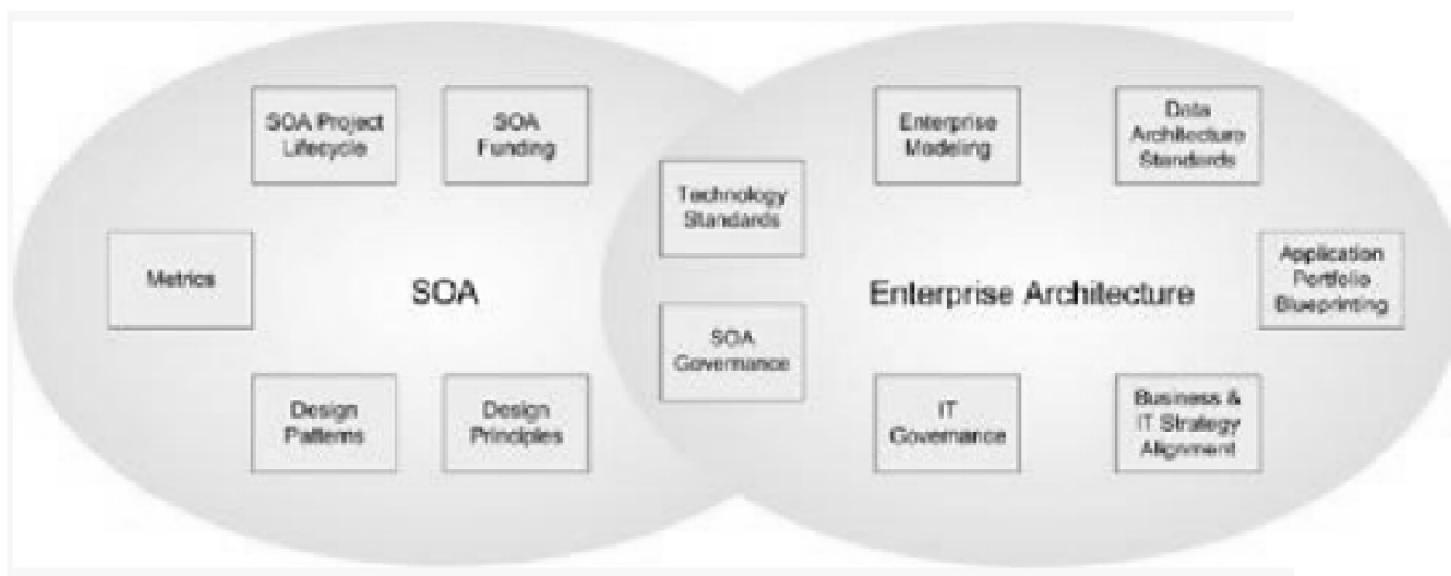
Relațiile dintre clienti și furnizori de servicii implementate pe baza SOA și a arhitecturii multinivel



Dacă se implementează arhitectura clasică pe trei nivele



SOA - subset al întreprinderii



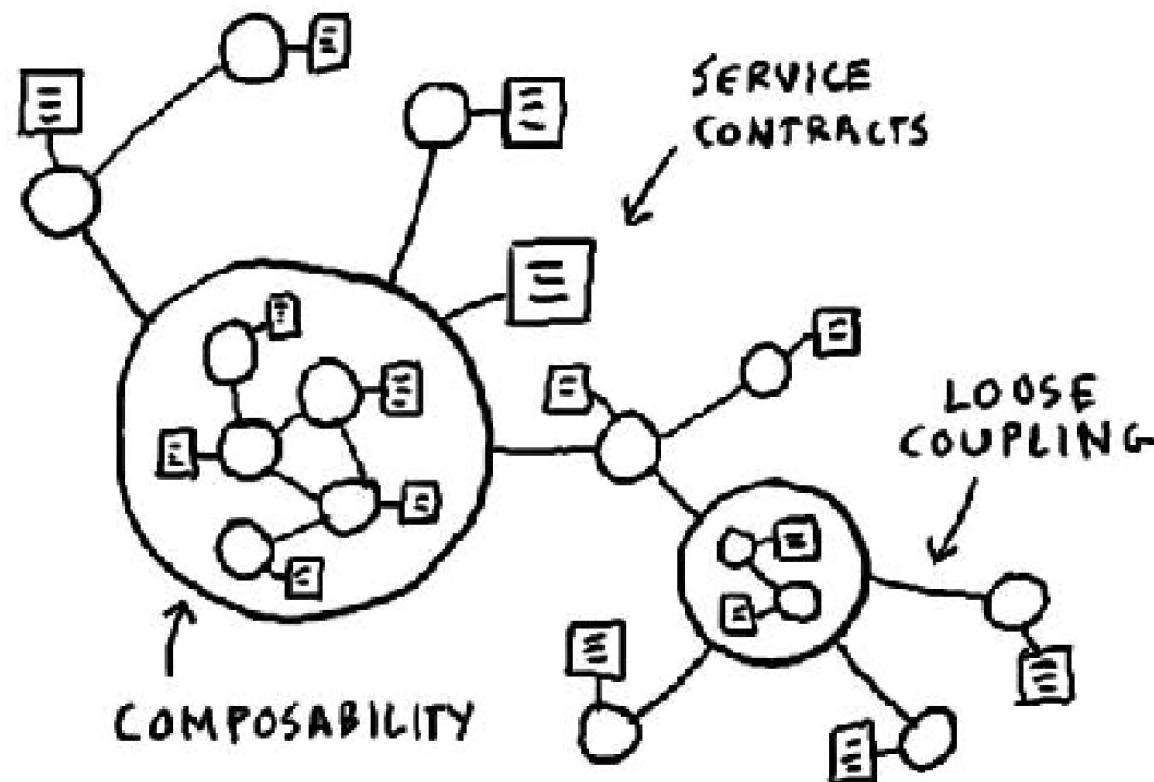
Sisteme Distribuite

Mihai Zaharia

cursul 4

Principiile orientării pe servicii

SERVICE ORIENTATION



Principii legate de utilizarea serviciilor

- **Principiul vizibilitatii serviciului**
- **Contractul Standard al unui serviciu**
- **Tipuri standardizate de conținut al contractelor serviciilor**
- **Principiul contextului desfasurarii/executiei unui serviciu**
- **Separare de functionalitati la nivelul serviciilor**
- **Abstractizarea Serviciilor**

Principiile construirii serviciilor

- **Principiul abstractizării serviciilor**
- **Principiul abstractizării contractului serviciului**
- **Abilitatea de compunere a serviciilor**
- **Principiul compunerii serviciilor**
- **Autonomia serviciilor**

SOA - puternic contestată din 2013 ...

- **Motive de divorț**

- prost înțeleasă de la început
- tehnologie?
- utilizare discreționară a ESB

Business Layer

Business Rationale, Business Processes

Application Layer

Capabilities and Functions

Information Layer

Data Models

Technology Layer

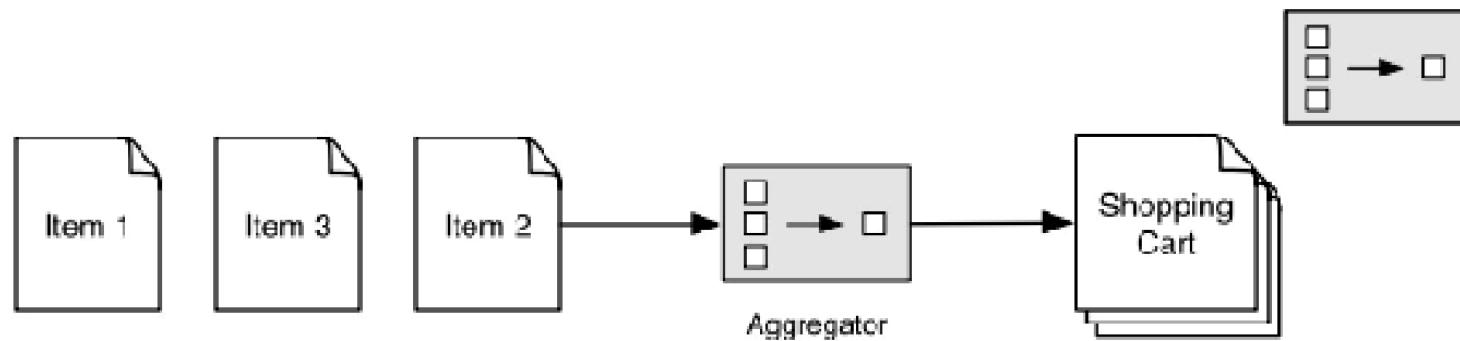
Standards, Bundling and Tooling

**Analysis
&
Design**

Implementation

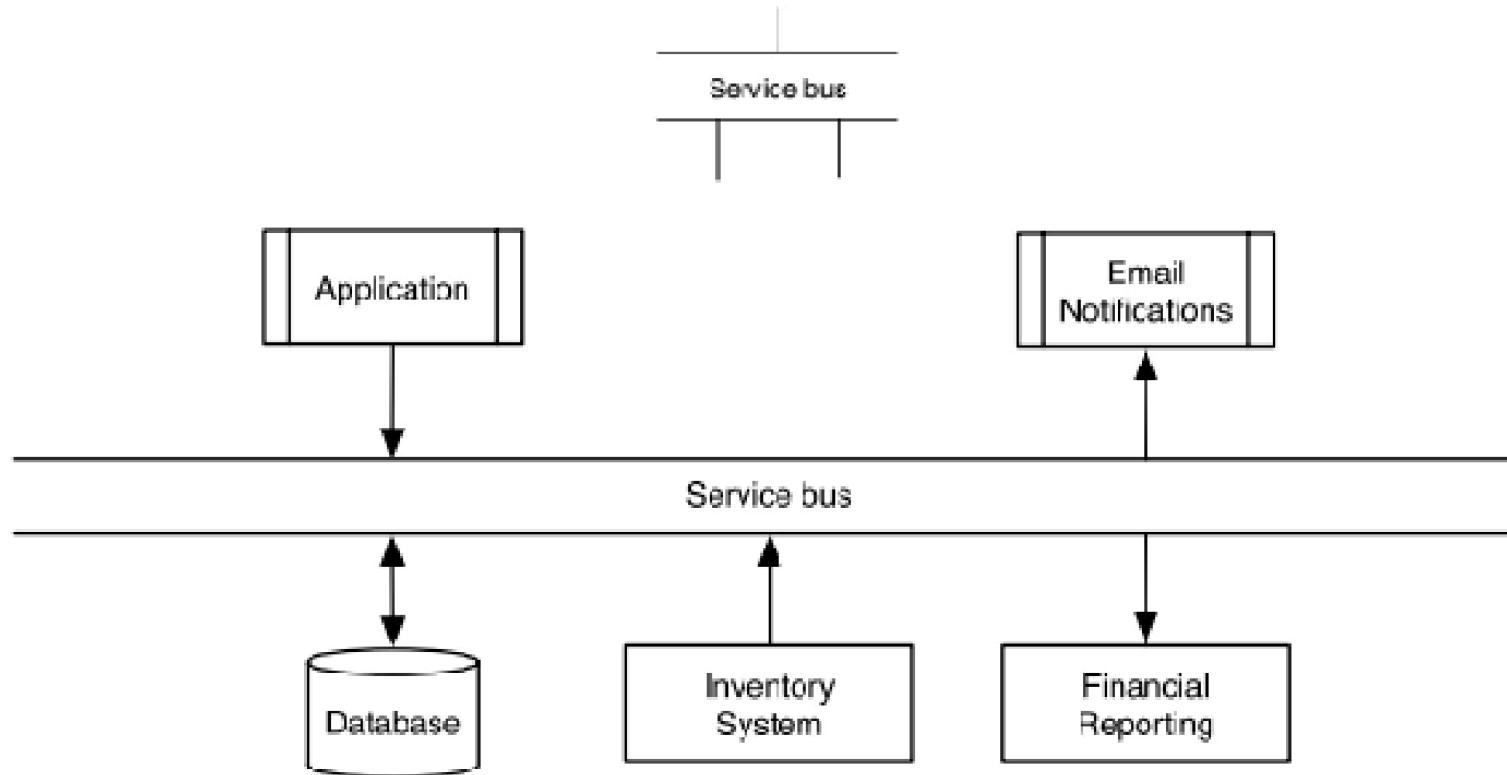
Modele primare de proiectare a serviciilor

- Agregare(Aggregator)



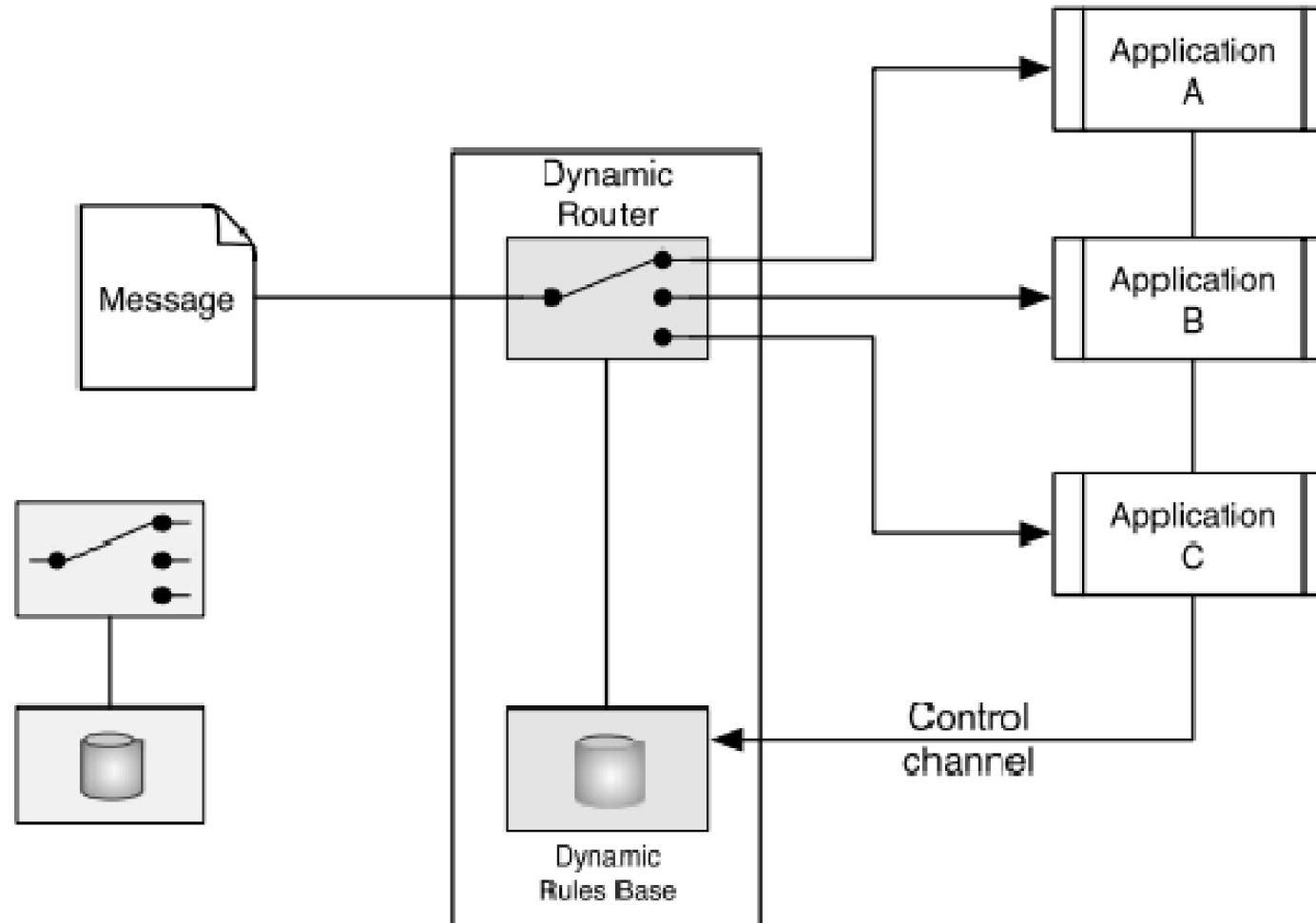
Modele primare de proiectare a serviciilor

- Magistrala de servicii Service Bus



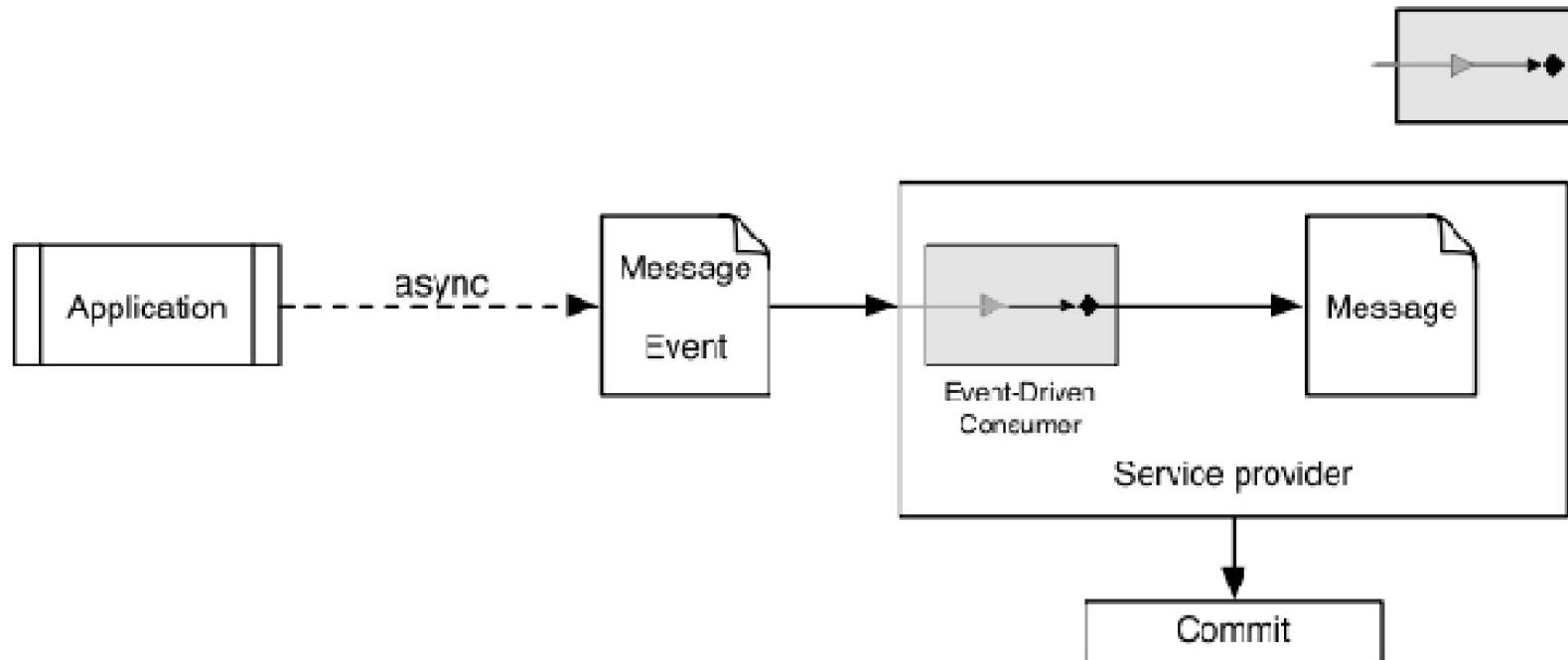
Modele primare de proiectare a serviciilor

- Dirijarea dinamica



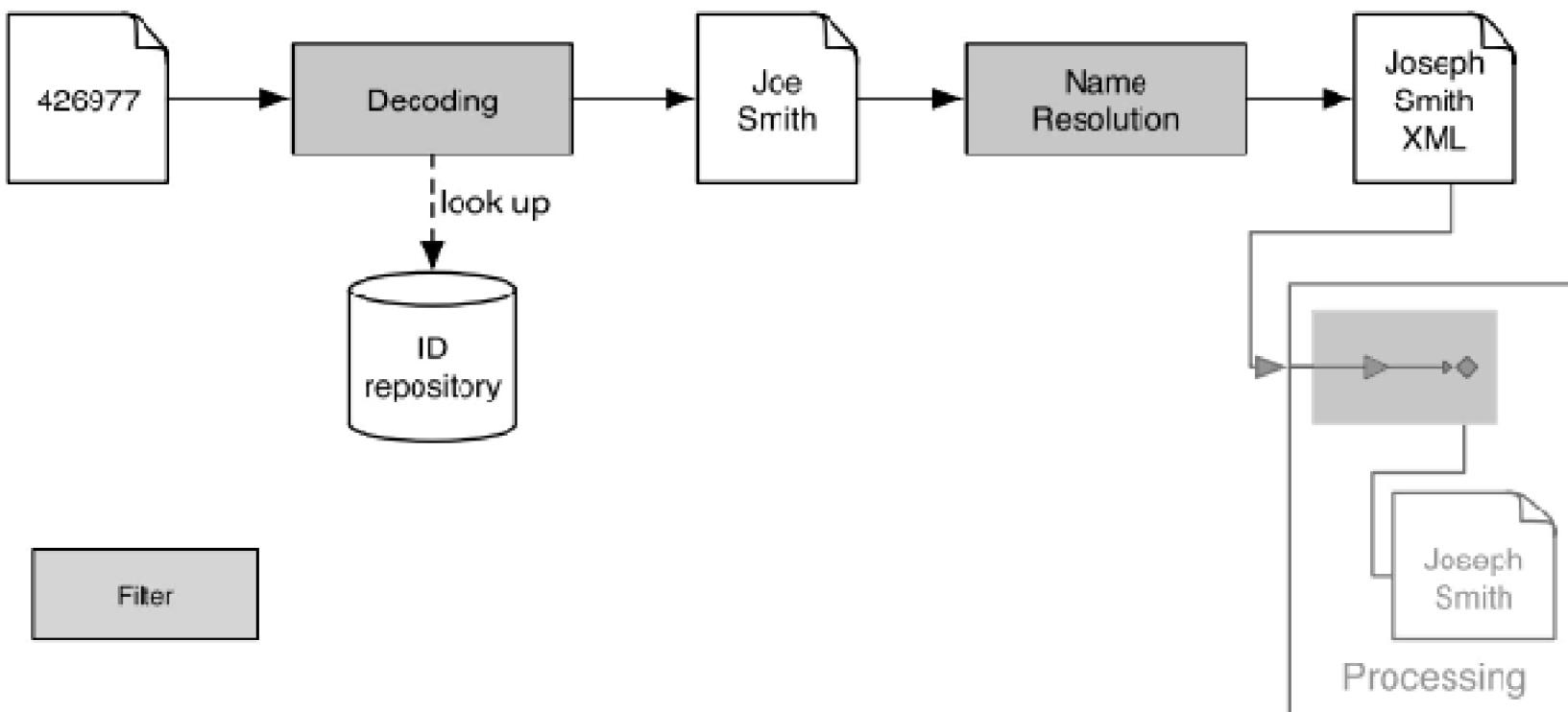
Modele primare de proiectare a serviciilor

- Consumator de evenimente



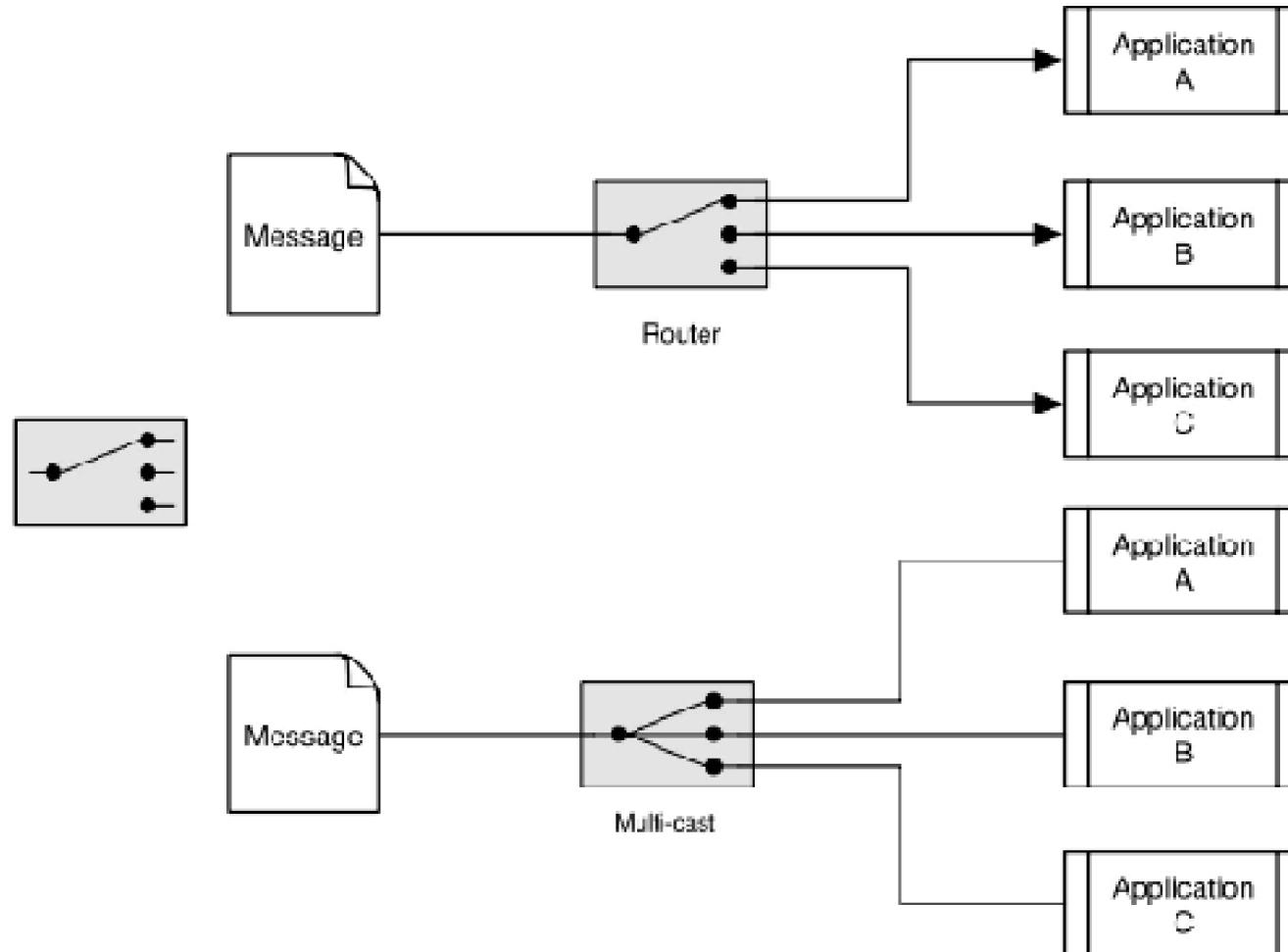
Modele primare de proiectare a serviciilor

- Filtrul



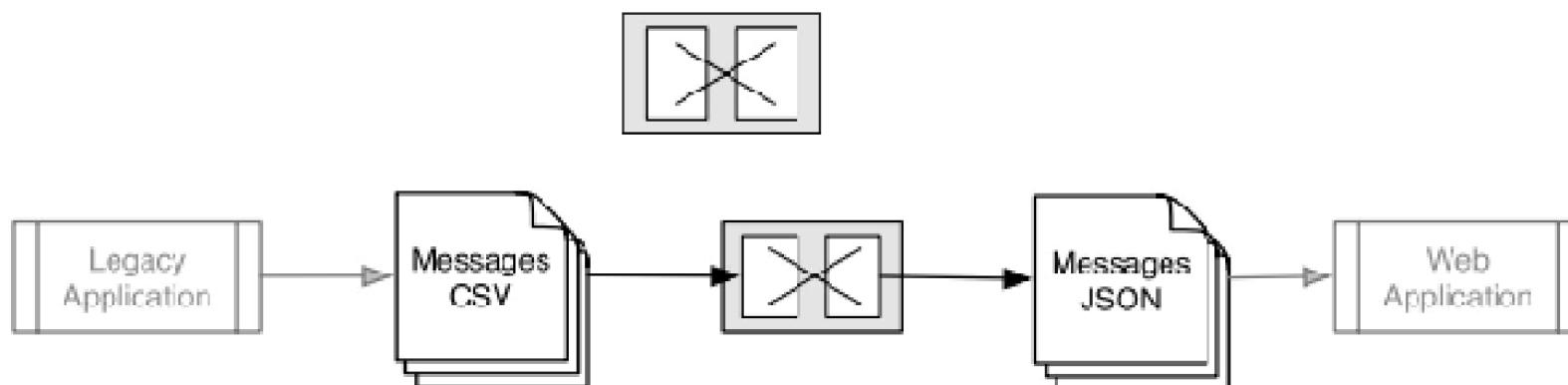
Modele primare de proiectare a serviciilor

- Dirijor -ruter

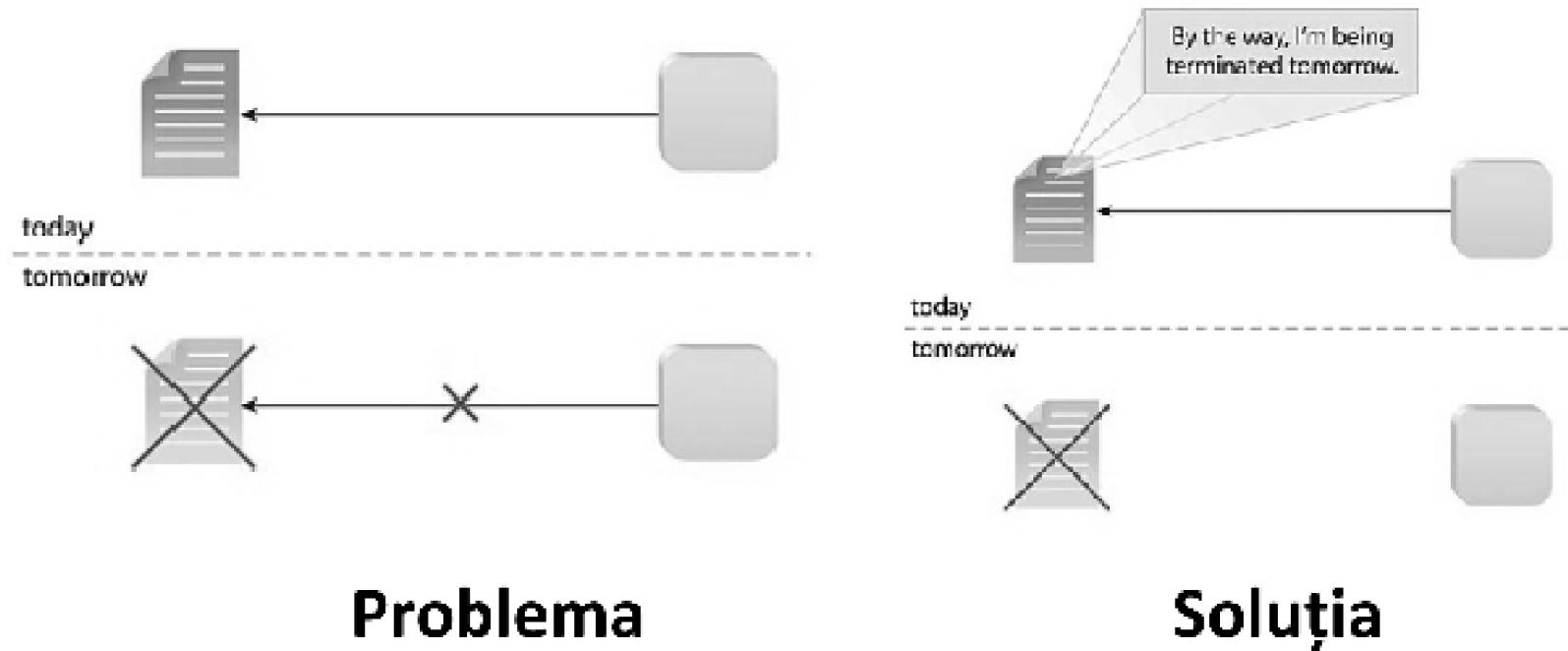


Modele primare de proiectare a serviciilor

- Traducatorul sau transformatorul



Anunțarea incetării definitive a functionării unui serviciu



© Randy Glasbergen
www.glasbergen.com



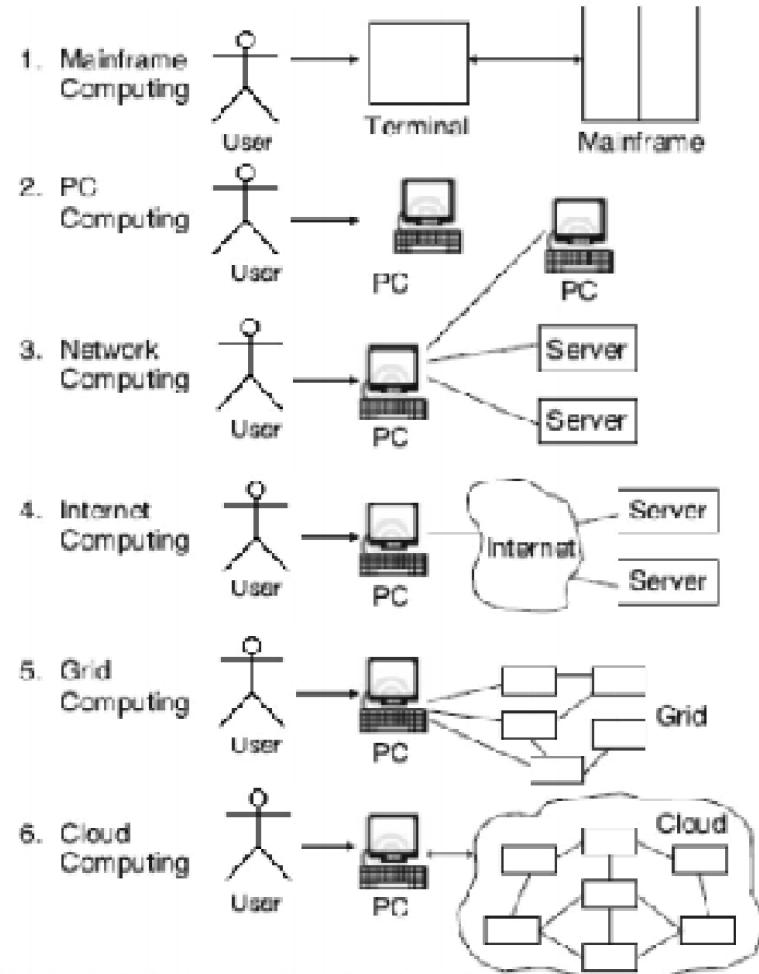
From Here we see all users data

Sisteme Distribuite

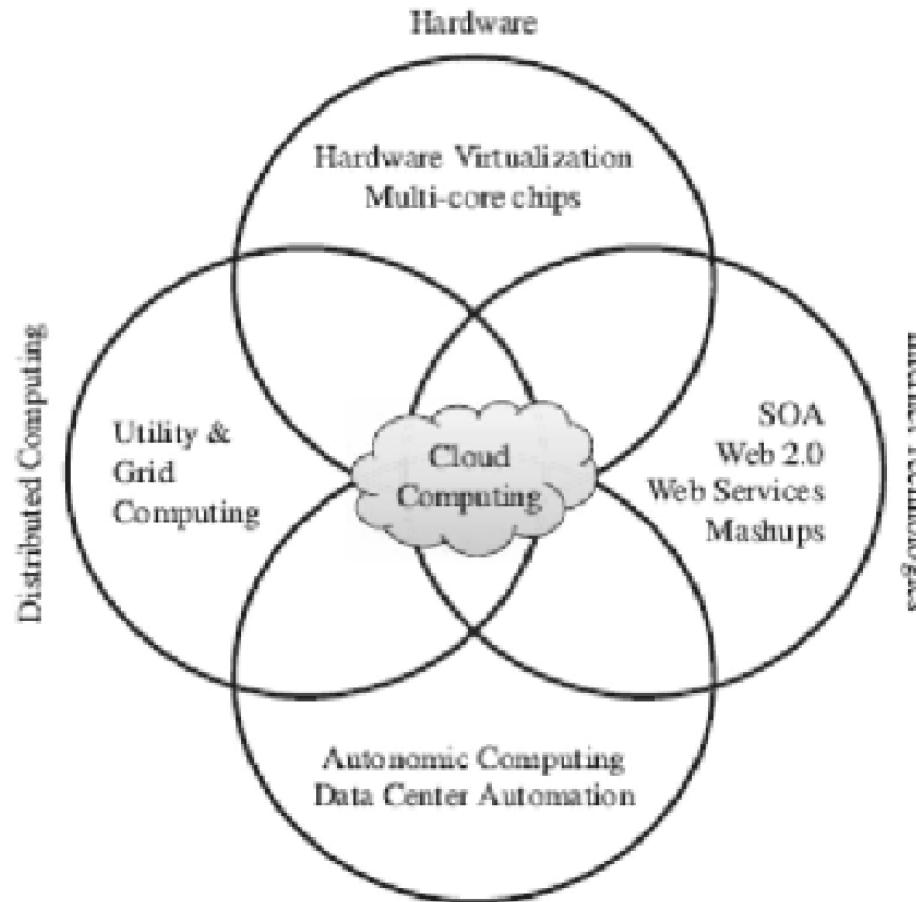
Mihai Zaharia

Curs 5

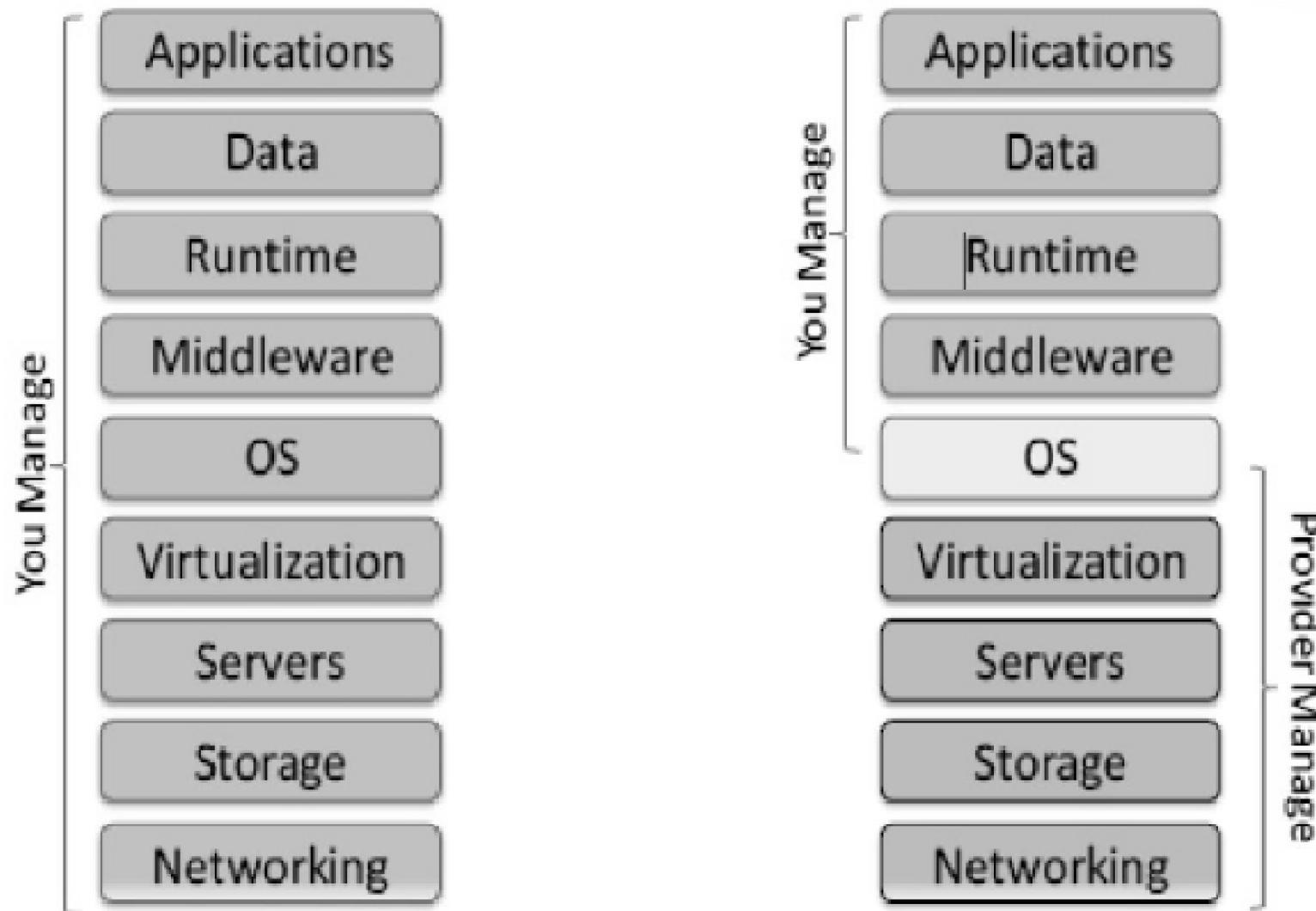
Scurtă analiză a evoluției structurilor de calcul



Unde se află de fapt norul ?



IaaS - Infrastructure as a Service



Modelul Traditional (*on-premise*) versus IaaS

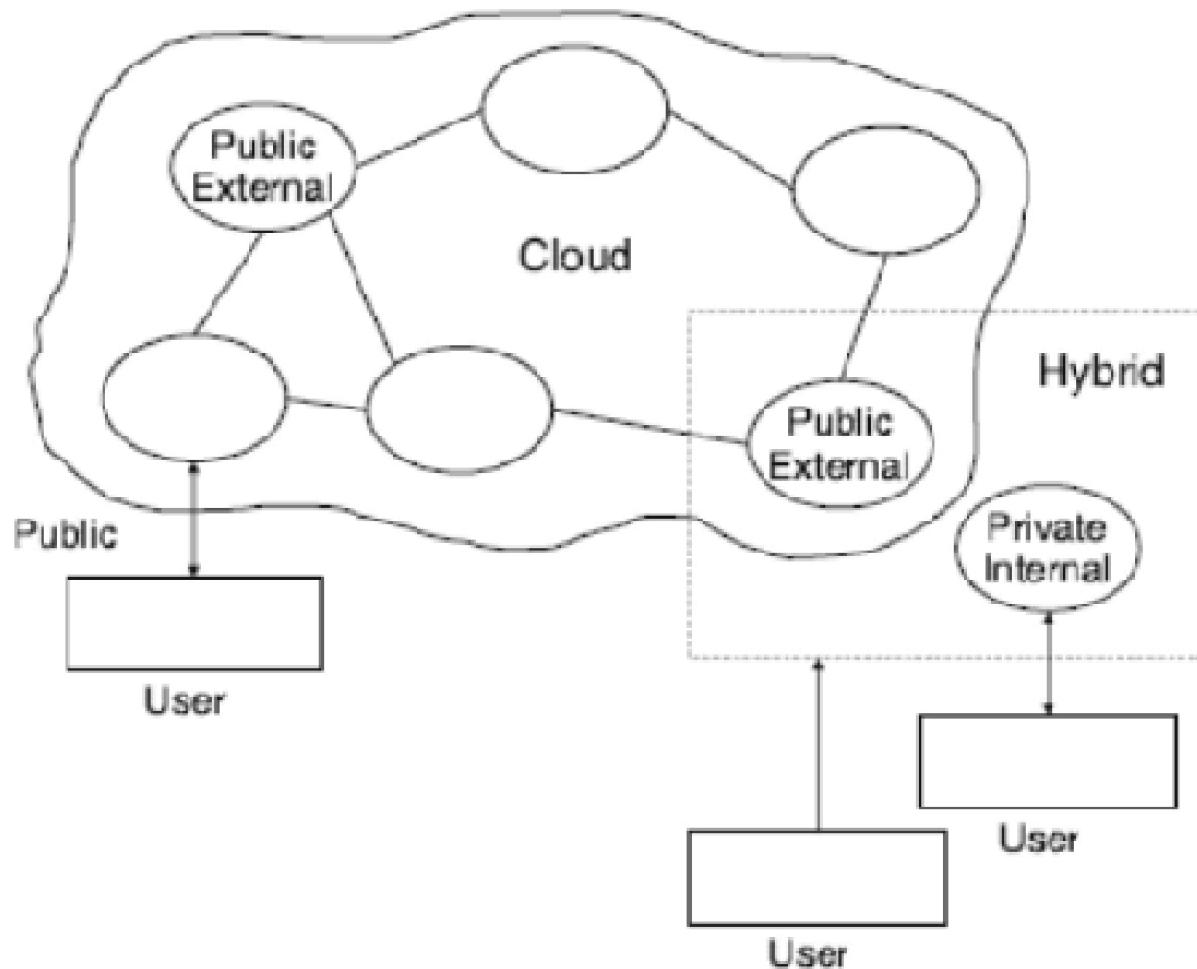
Ce este norul?

- **Definiția NIST**
- Cloud computing este un model de plată funcție de utilizare care permite accesul, pe baza de rețea, la cerere, convenabil, disponibil, la o grupare de resurse de calcul configurabile (ex., rețele, servere, stocare, aplicații).
- Aceste servicii care pot fi oferite rapid și cu un efort de administrare minimal sau cu interacțiune minima cu furnizorul de serviciu.

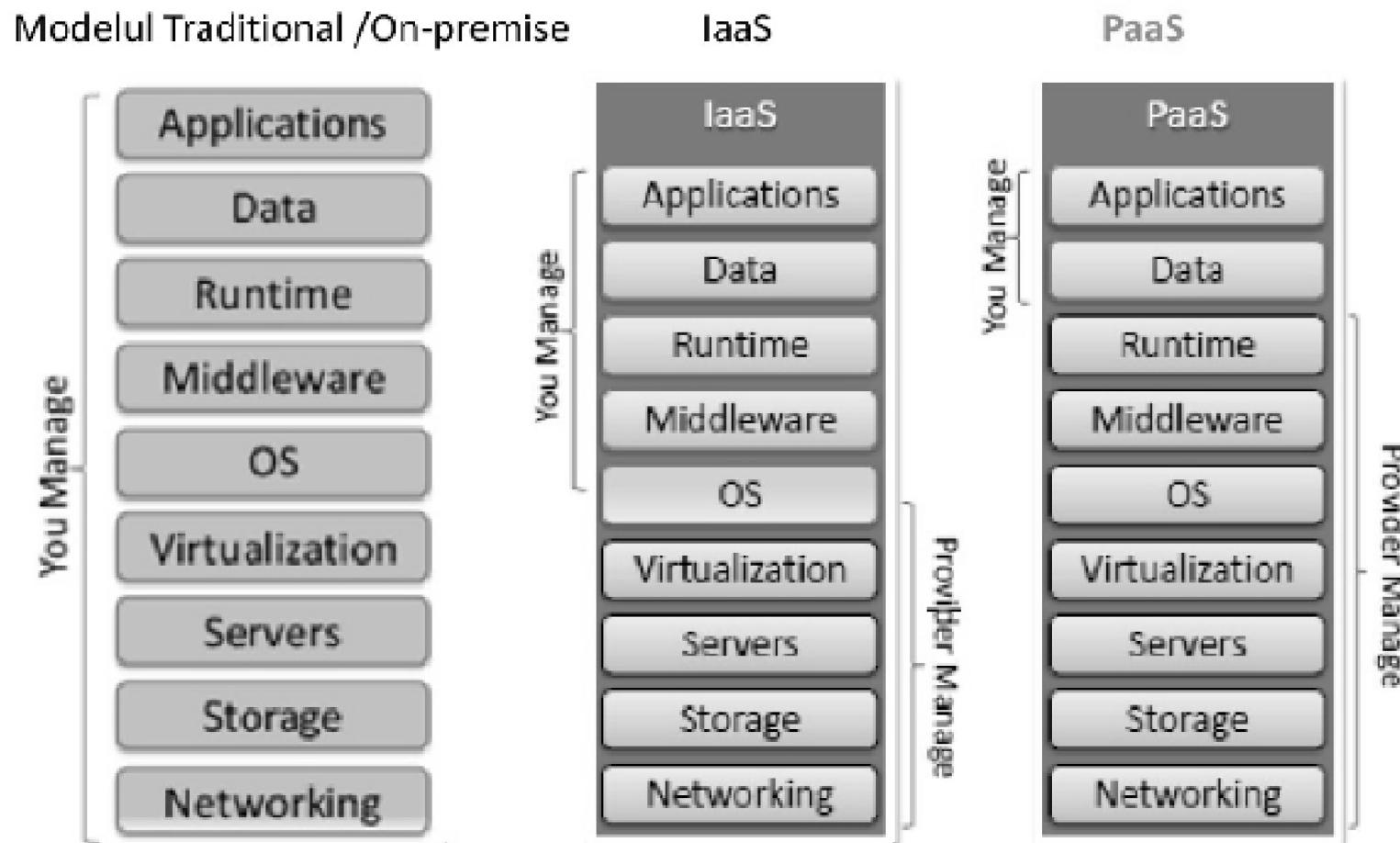
Caracteristicile de baza

- Auto-service la cerere
- Acces de oriunde la retea
- Grupare a resurselor independente de locatie.
- Plata dupa cat consumi

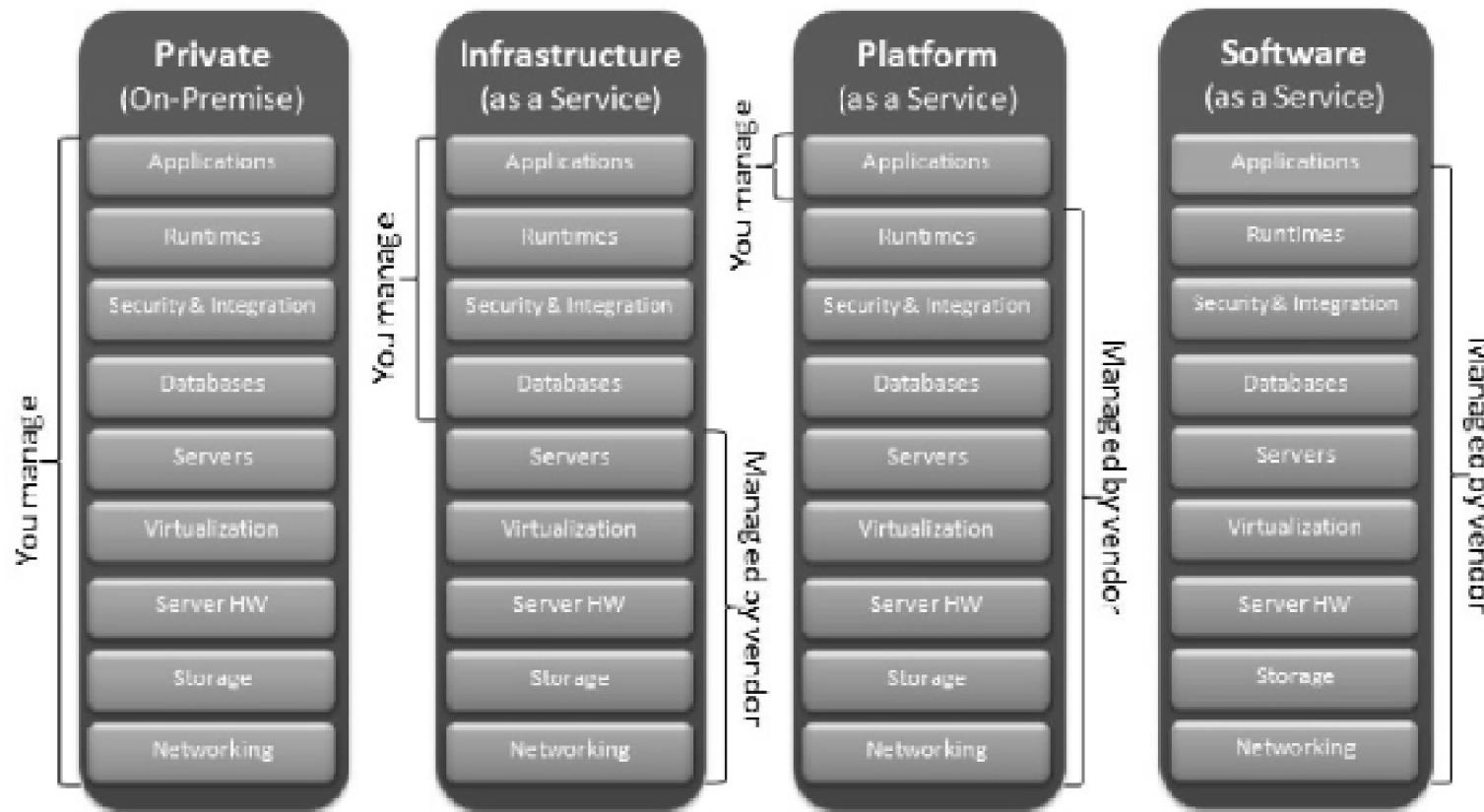
Tipuri de Nori



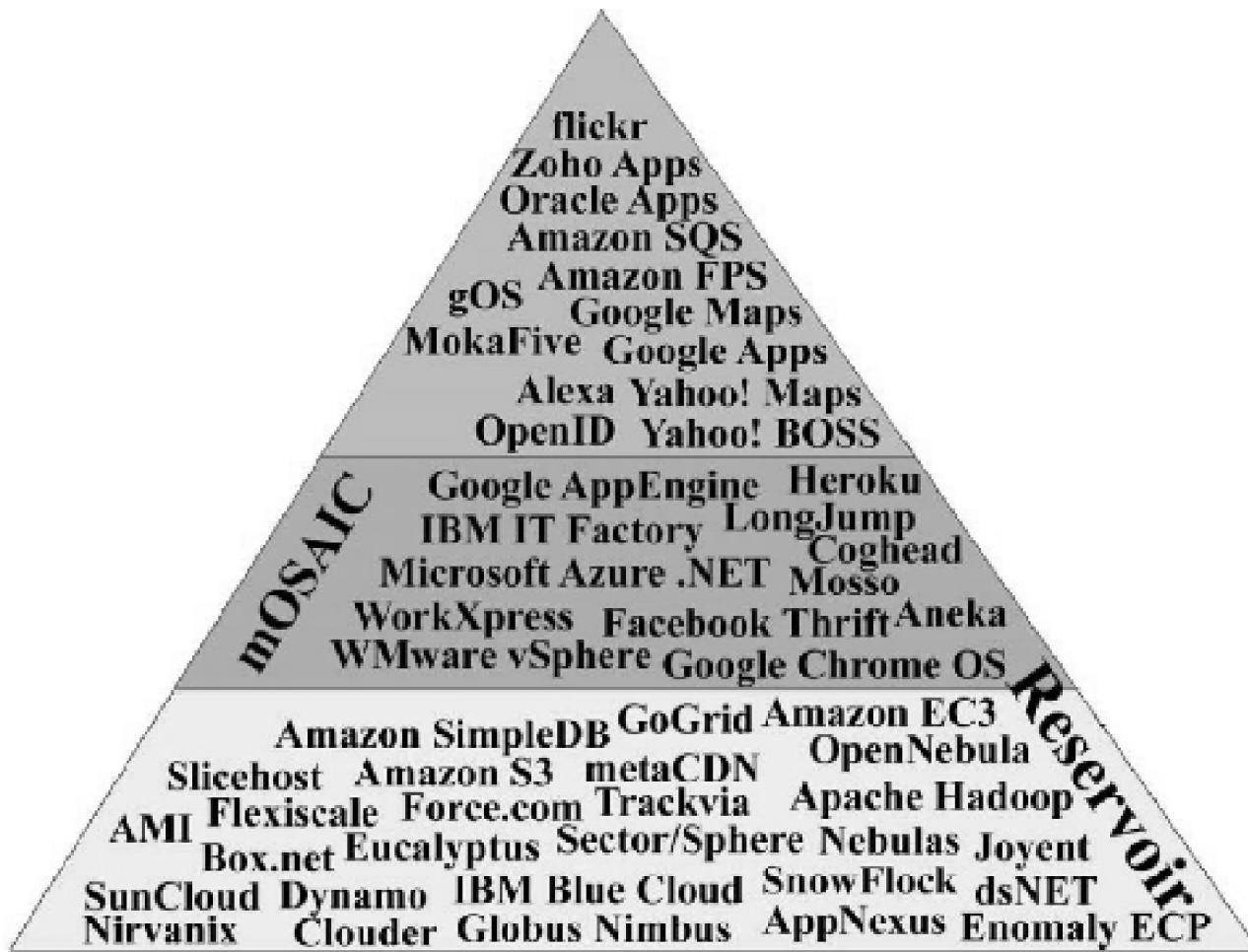
Platform as a Service (PaaS)



Furnizori de servicii Cloud



Piramida Norului



FaaS

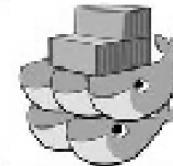
Functions as a Service

API Gateway

Function Watchdog



Prometheus



Swarm



Kubernetes



Orice ca serviciu (XaaS)

- Storage as a Service
- Database as a Service
- Communication as a Service
- Network as a Service
- Monitoring as a Service
- Testing as a Service
- HPC as a Service
- Human as a Service
- Process as a Service
- Information as a Service
- Identity as a Service
- Application as a Service
- Integration as a Service
- Governance as a Service
- Security as a Service
- Backup as a Service

Serviciile Amazon

- *Elastic Compute Cloud*
- *Simple Storage Service*
- *Simple Queue Service*
- *SimpleDB*
- *CloudFront*

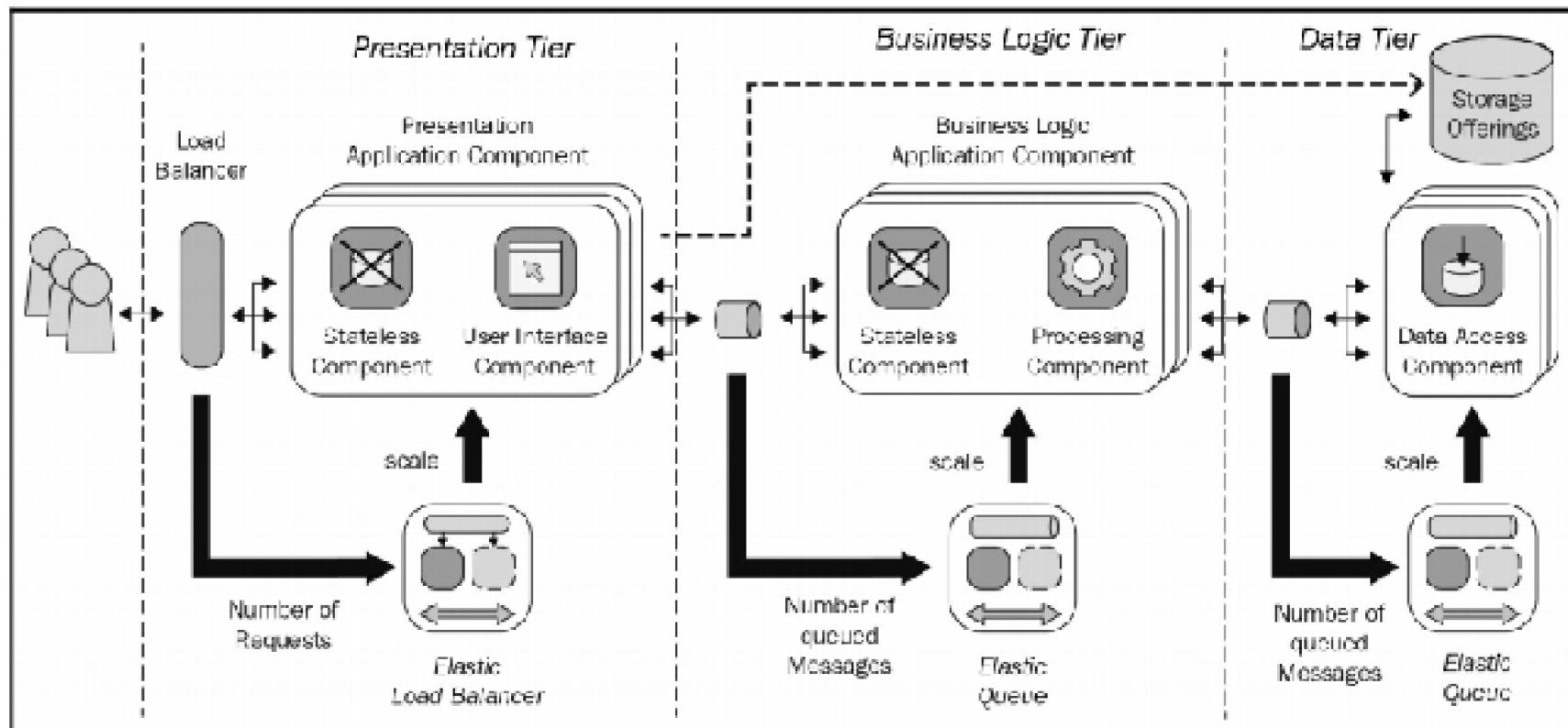
Windows Azure



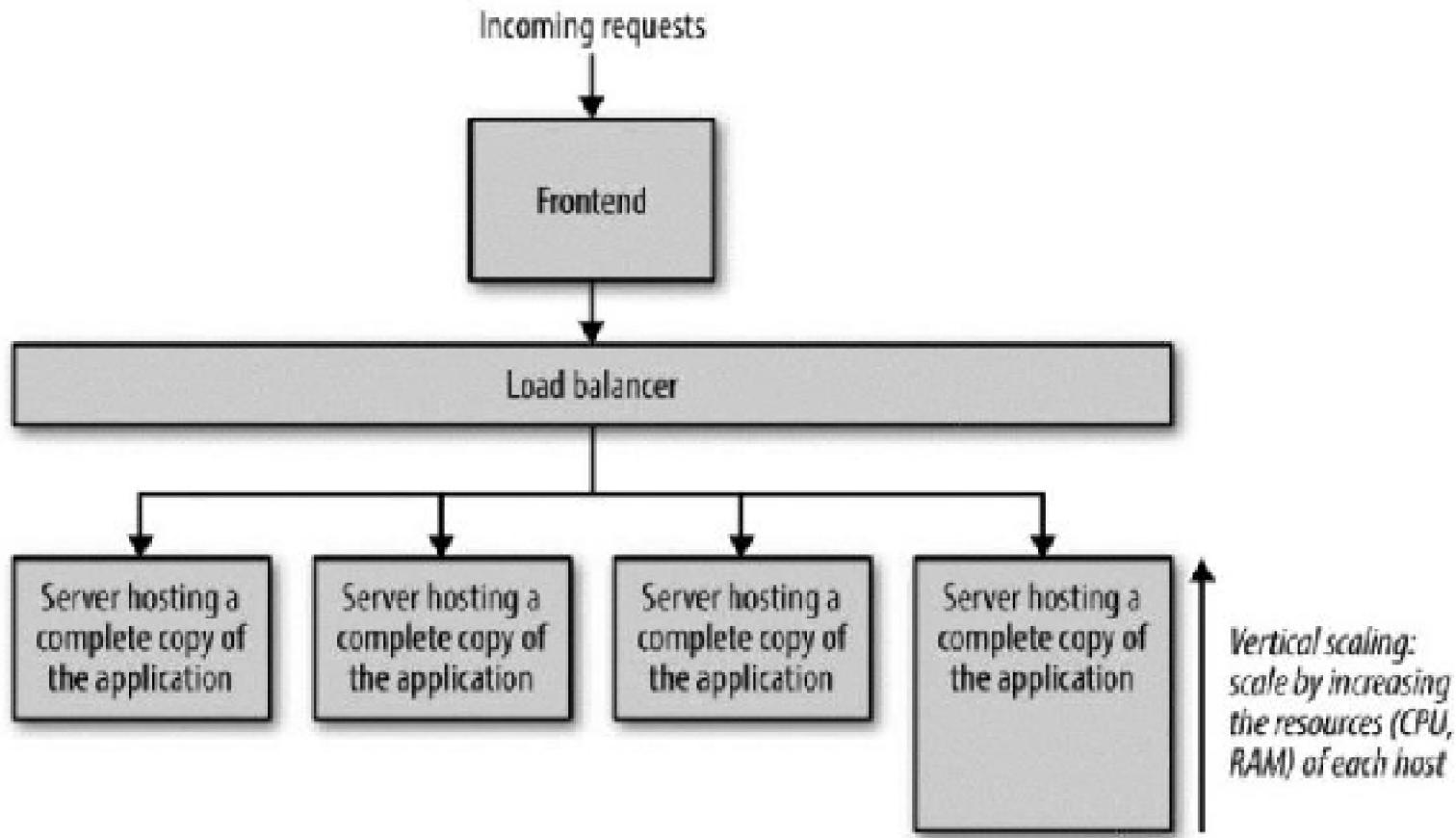
Terminologie și concepte specifice norilor

- **Centru de calcul cu patru straturi (Tier 4 data center)**
- **Hiperconvergentă**
- **Platforma hiperconvergentă**

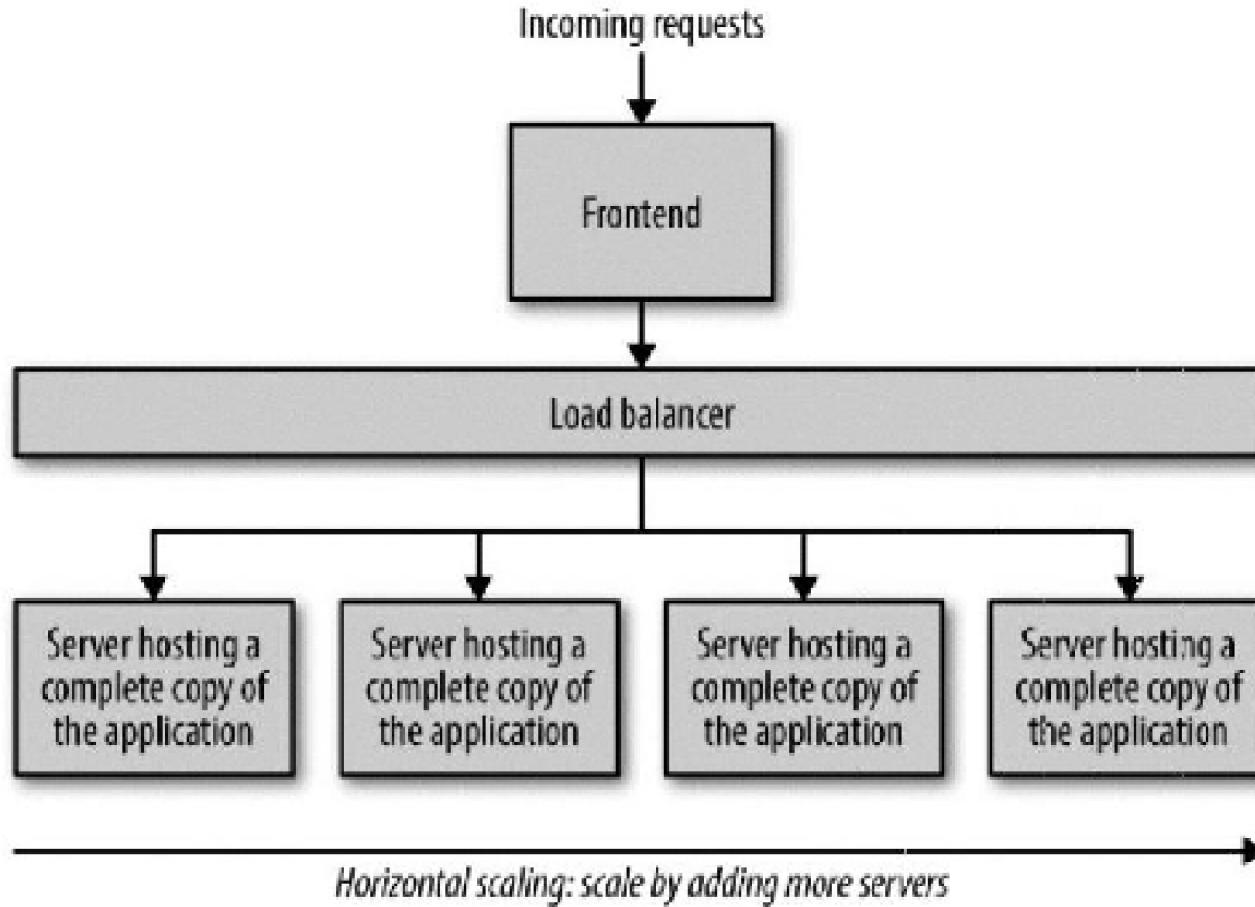
Arhitectura norului



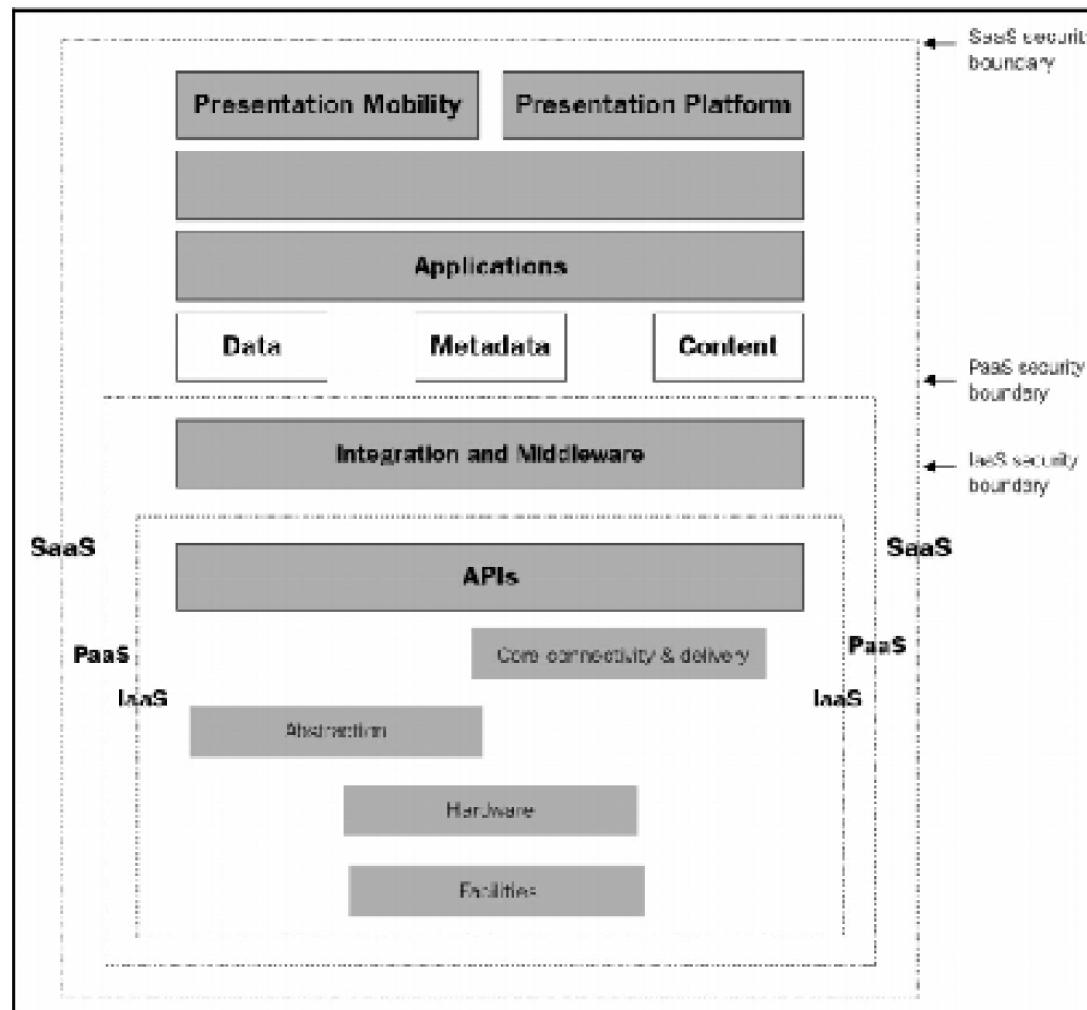
Scale-up sau Vertical Scaling



Scale-out sau Horizontal Scaling



Modelul de securitate al nor



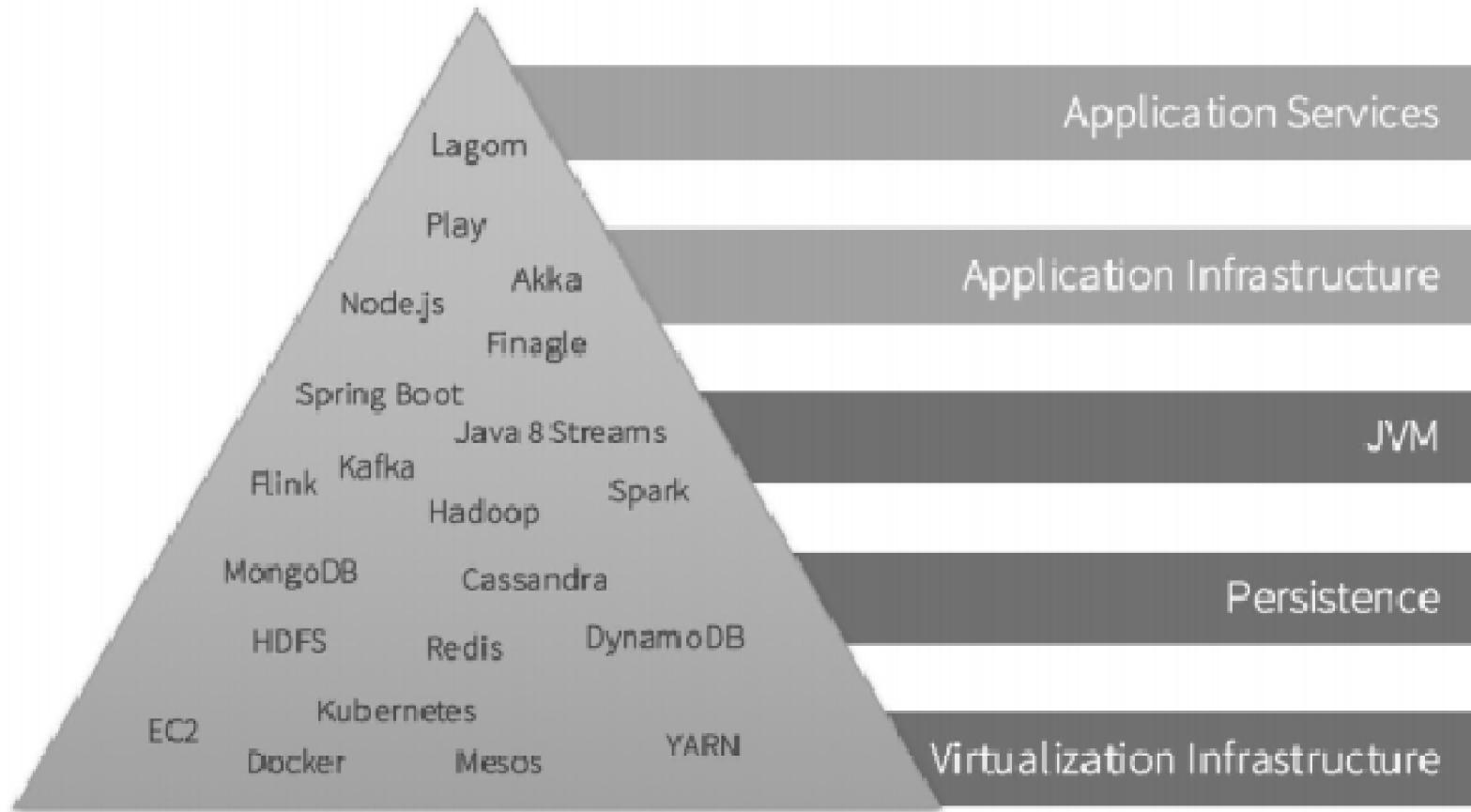
Sisteme Distribuite

Mihai Zaharia
Cursul 6

Arhitecturi bazate pe tratare evenimente

- Avantaje:
 1. Nu este nevoie de o mapare relațională a obiectelor
 2. Cu fiecare schimbare externă capturată ca eveniment starea curentă poate fi repetată
 3. Asigurarea persistenței se face fără modificări
- Rezultă că nu mai am nevoie de un SGBD relațional

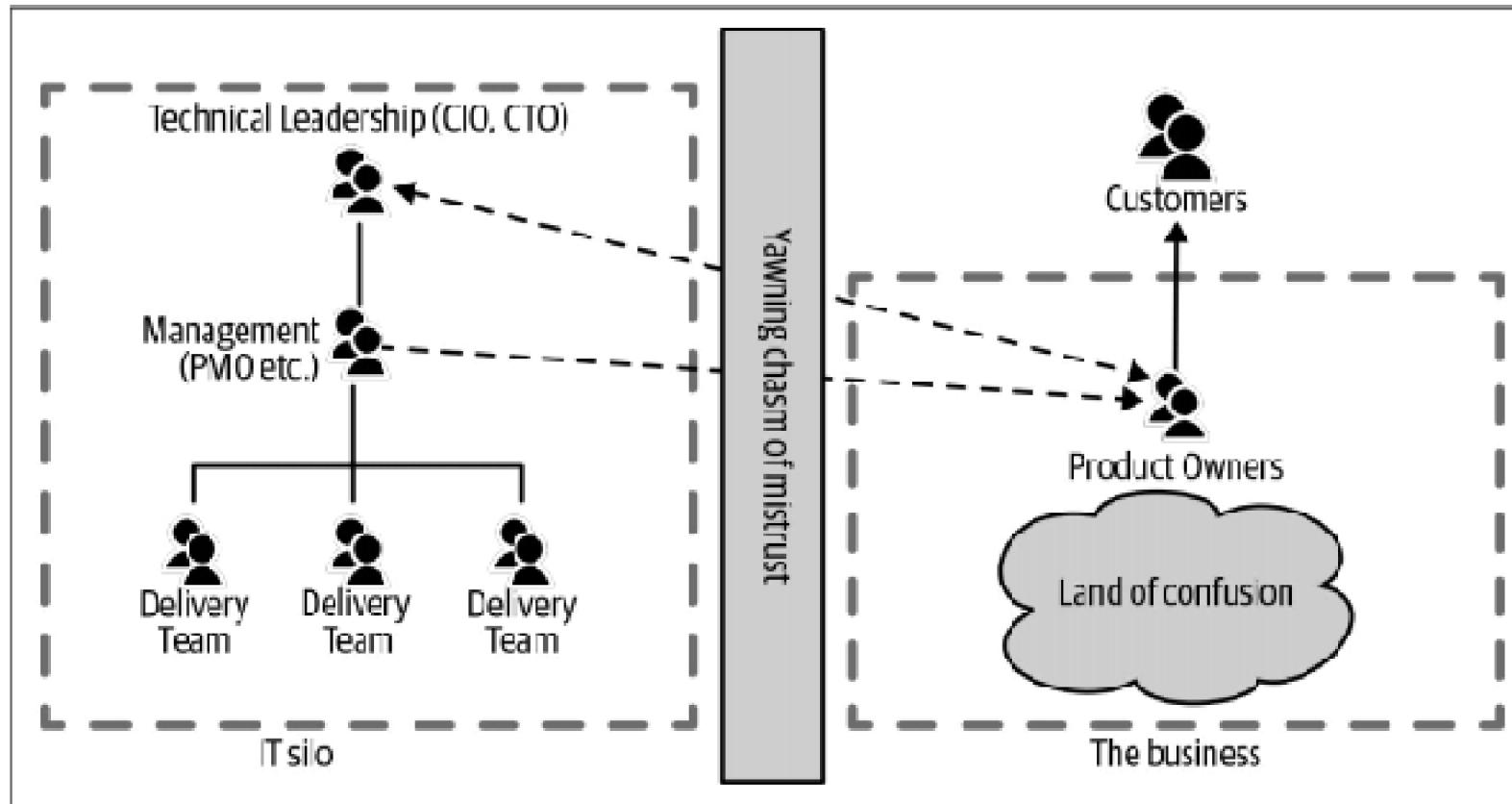
Ce este cu JEE?



Cum s-a carpat JEE containerelor

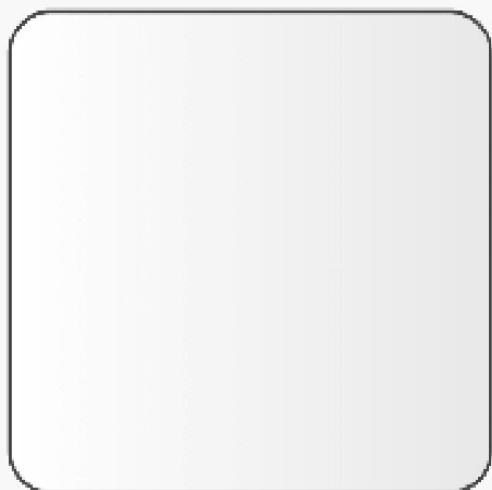
- Asamblările hibride -> probleme
- Frameworkurile clasice - aceleși probleme
- JEE & containerele ei?
- JBoss & Spring | PicoContainer

Proprietarul unui microserviciu



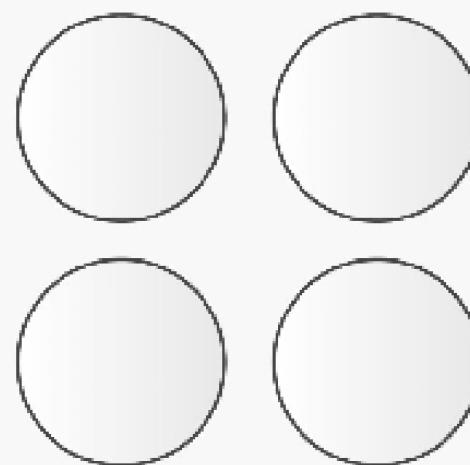
Granularitate software

Monolithic vs. SOA vs. Microservices



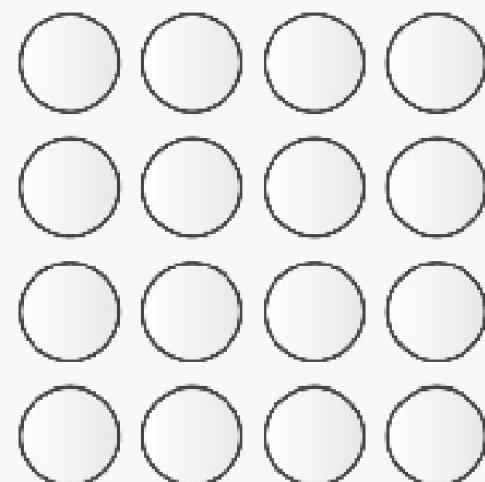
Monolithic

Single Unit



SOA

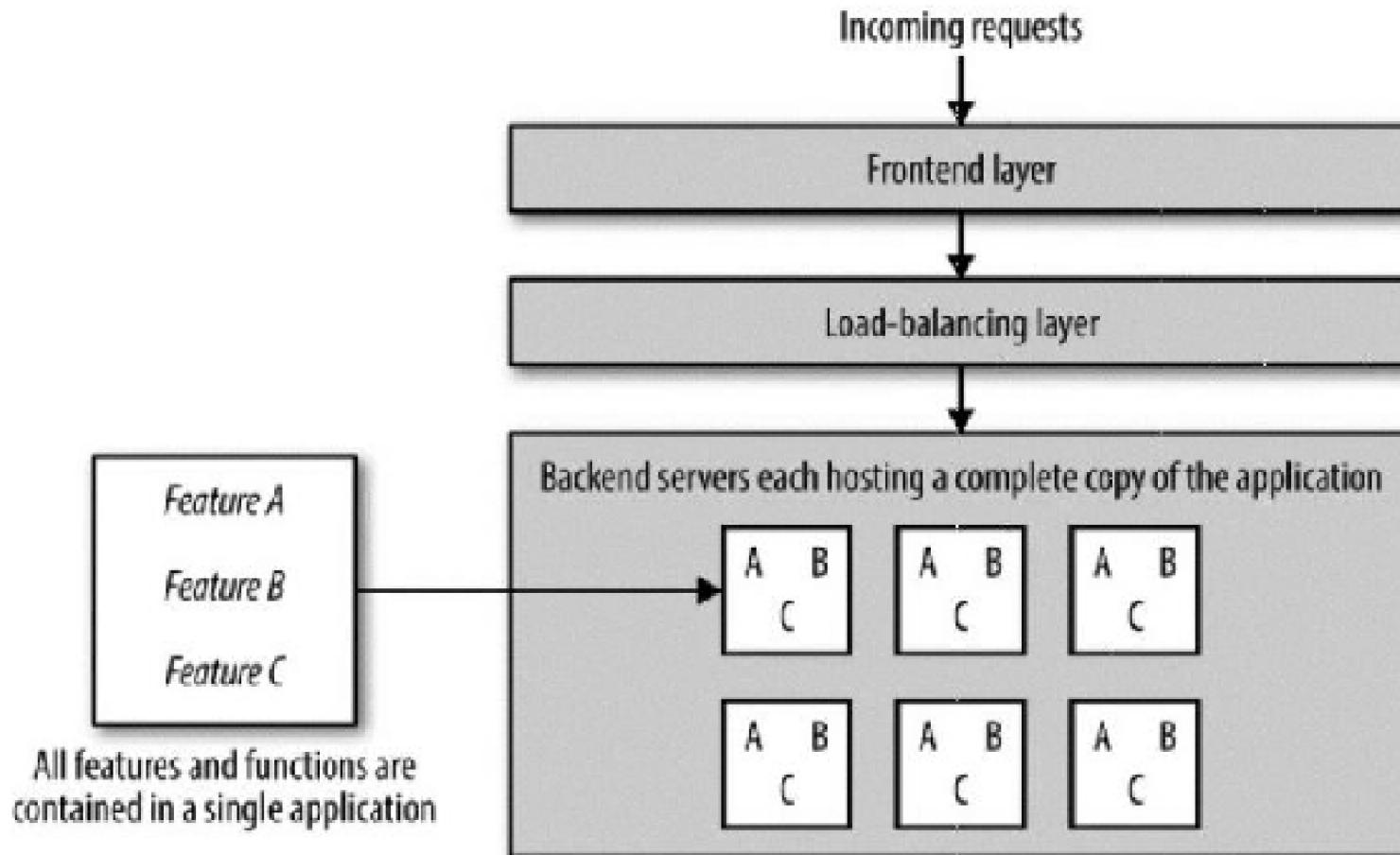
Coarse-grained



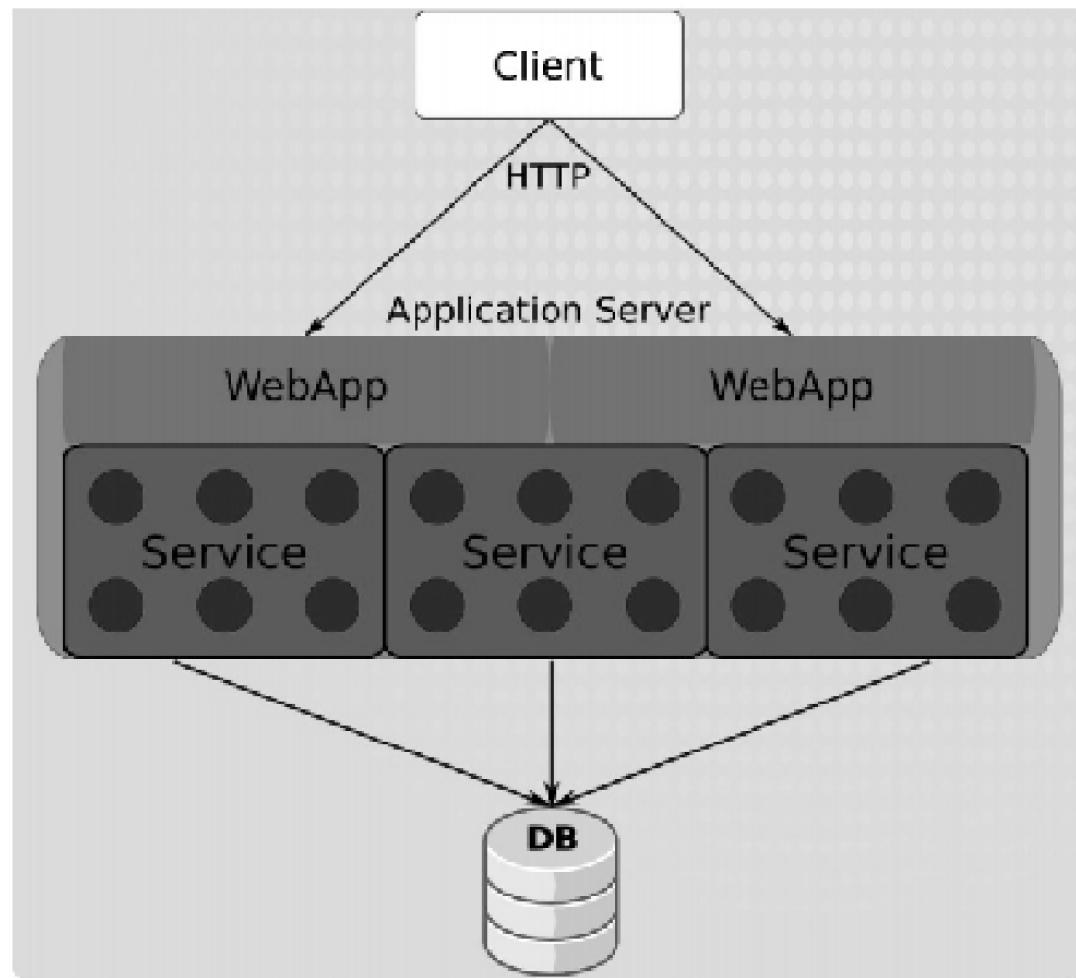
Microservices

Fine-grained

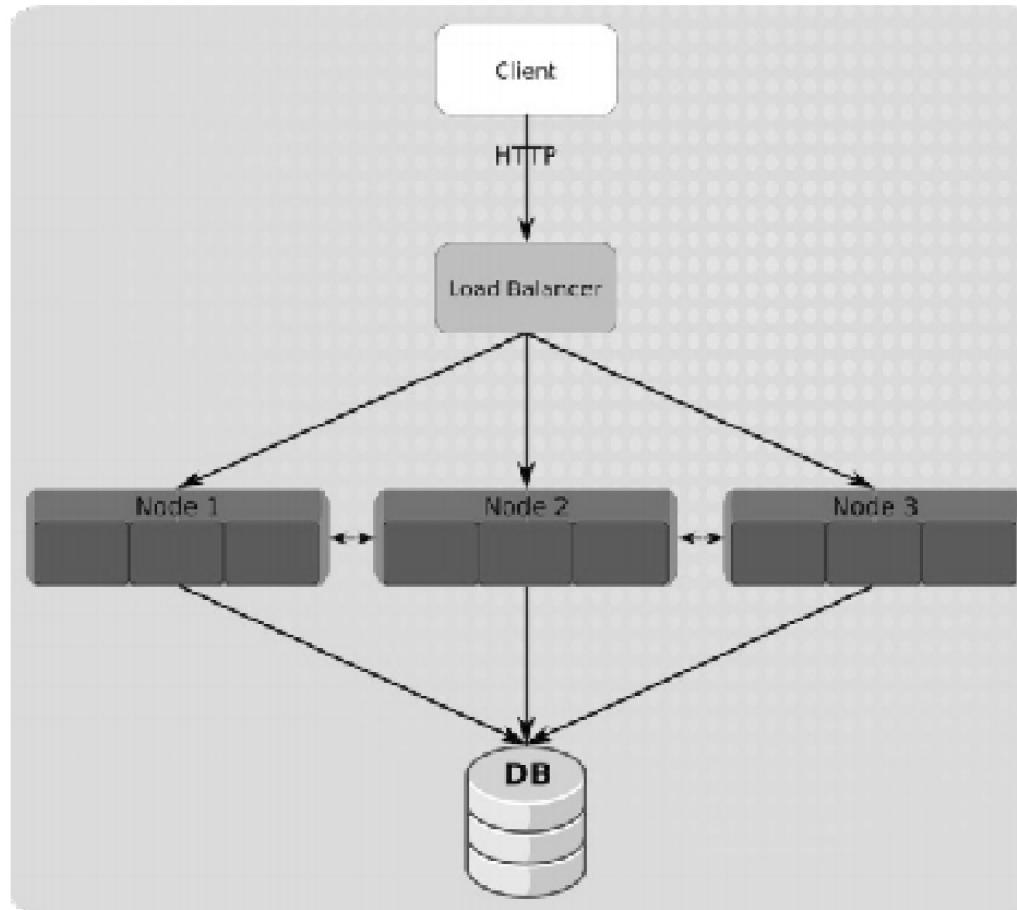
Aplicatiile monolit



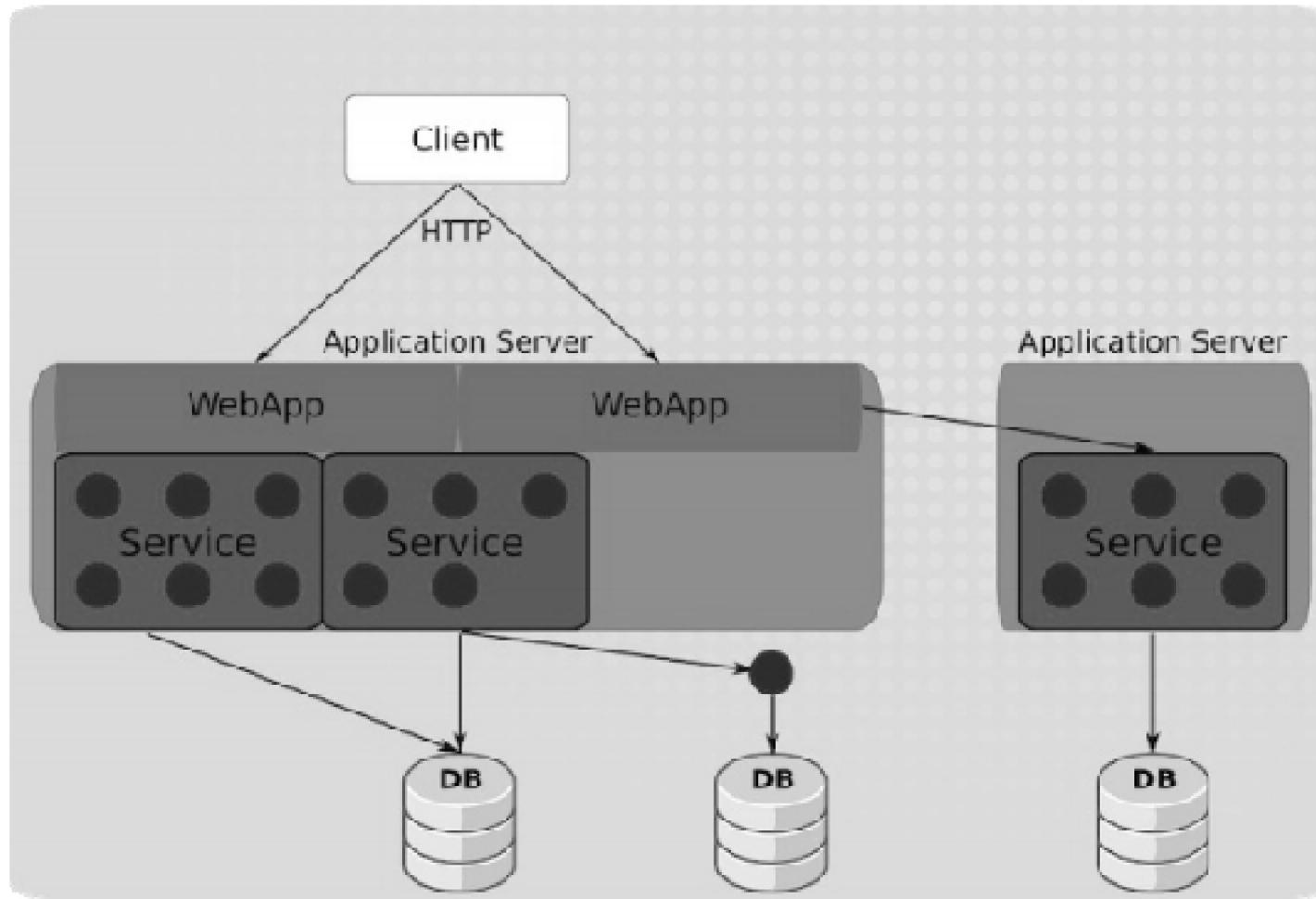
Cârpăcirea monoliștilor



Cârpăcirea monoliștilor

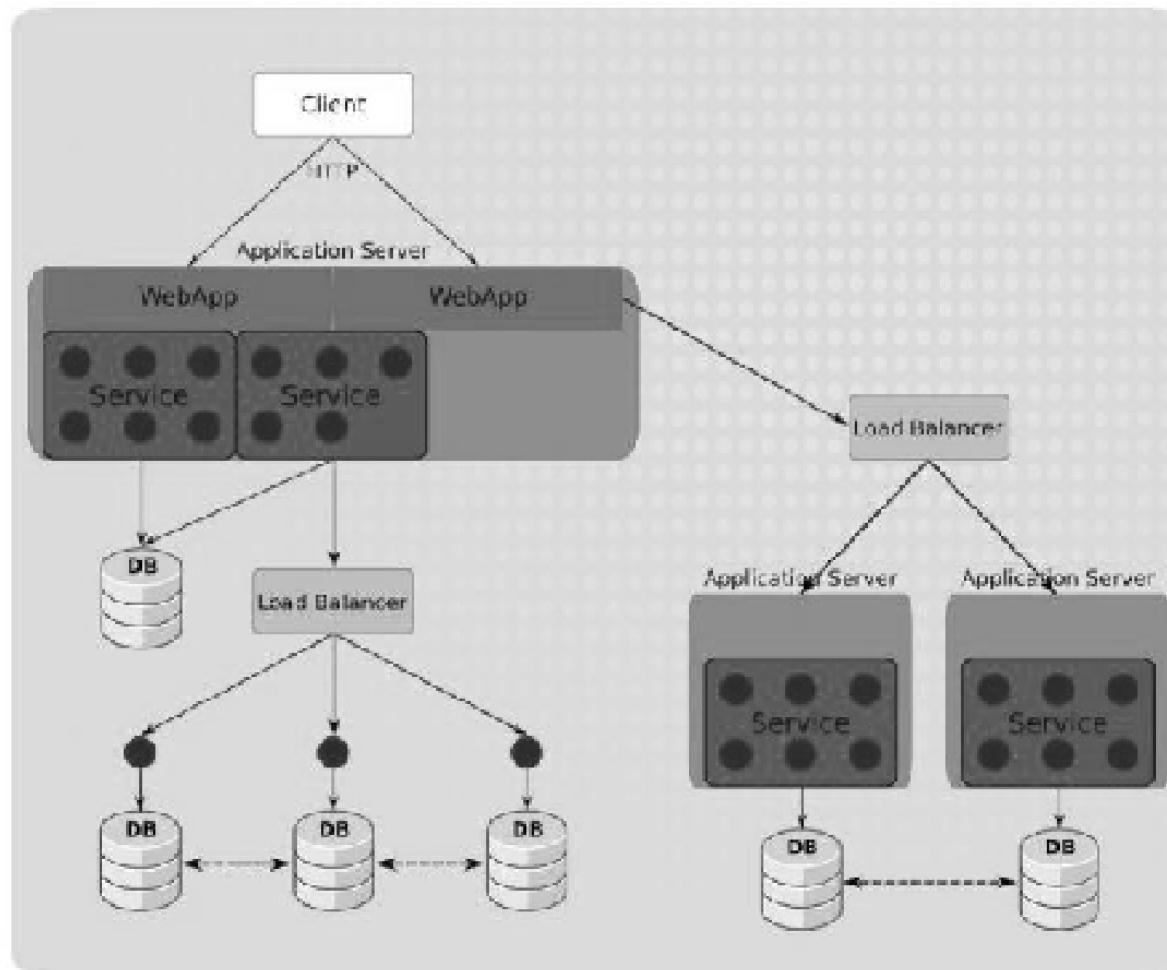


Microservicii tactice

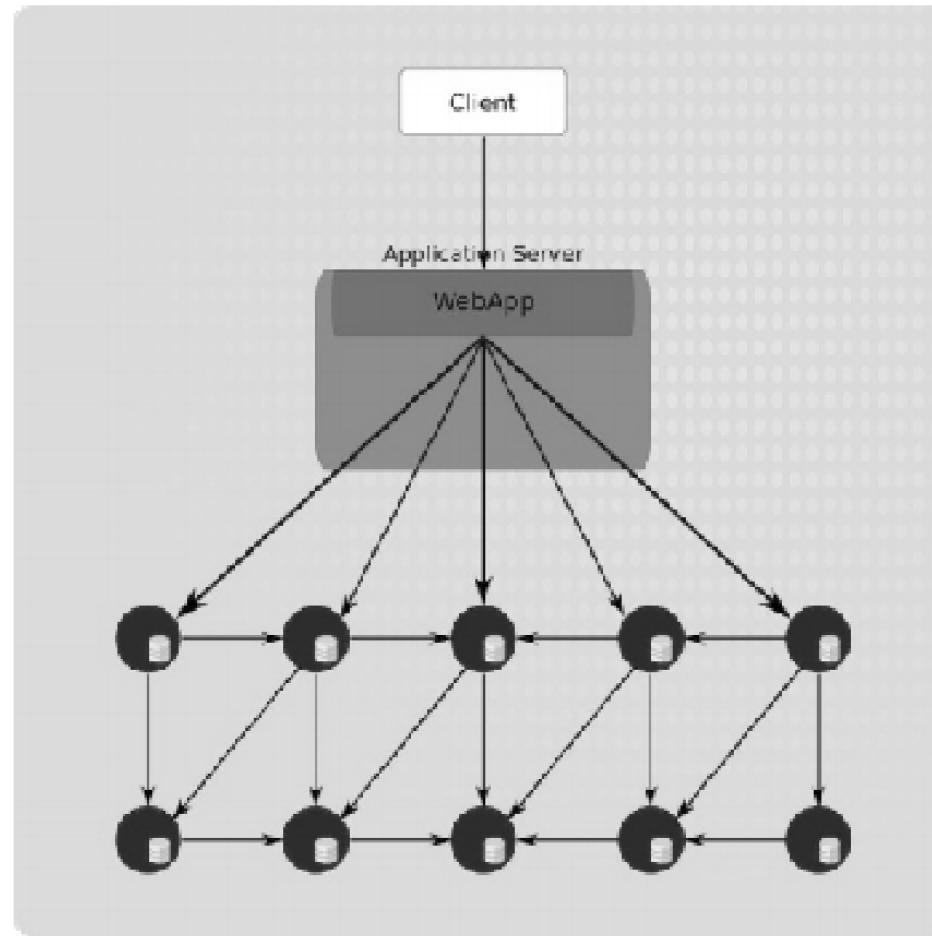


Strategii de trecere la utilizarea microserviciilor

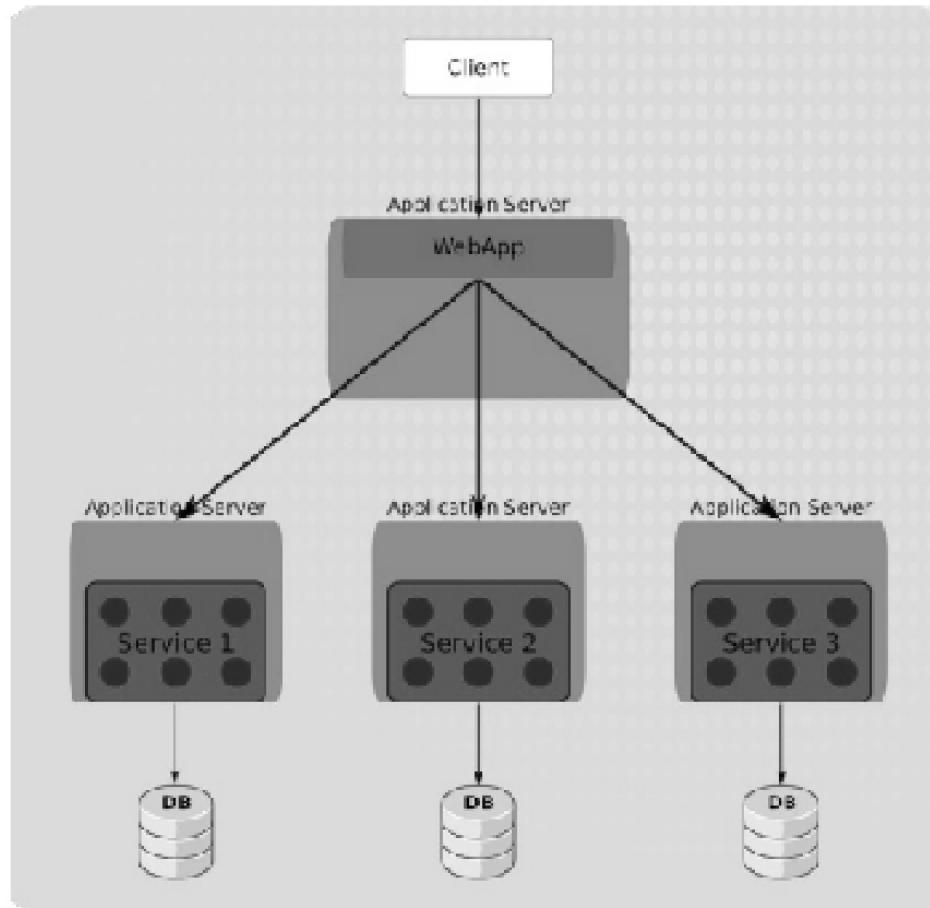
Strategii de trecere la utilizarea microserviciilor



Microservicii strategice

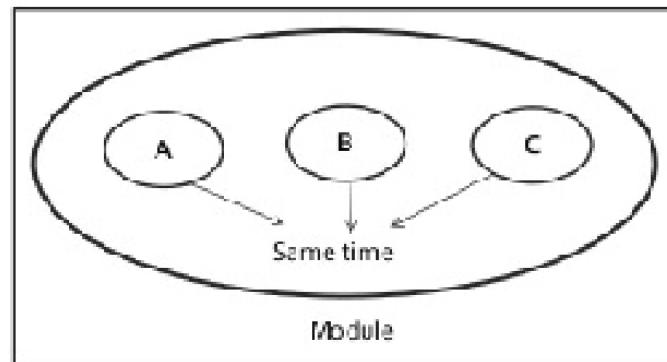


Business-Driven Microservices

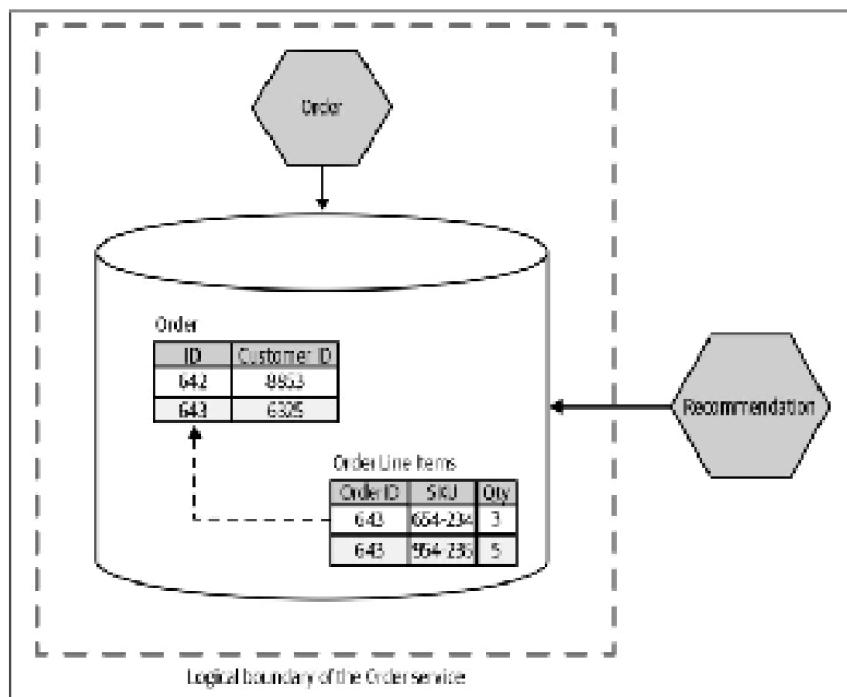


Ascunderea informației

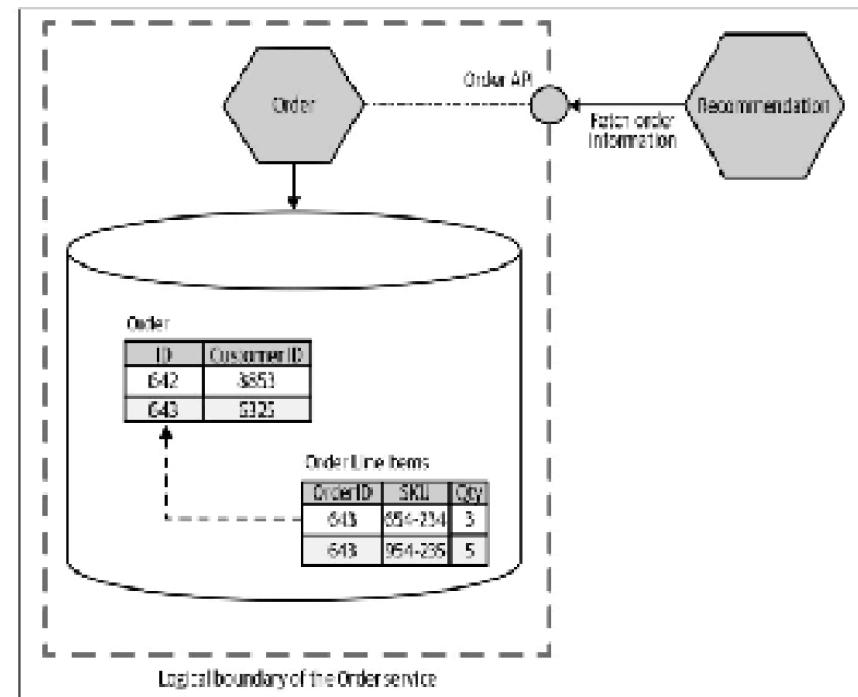
Cohesion: A natural extension of the information hiding concept a module may perform a number of tasks. A cohesive module performs a single task in a procedure with little interactions with others.



Cuplarea în implementare

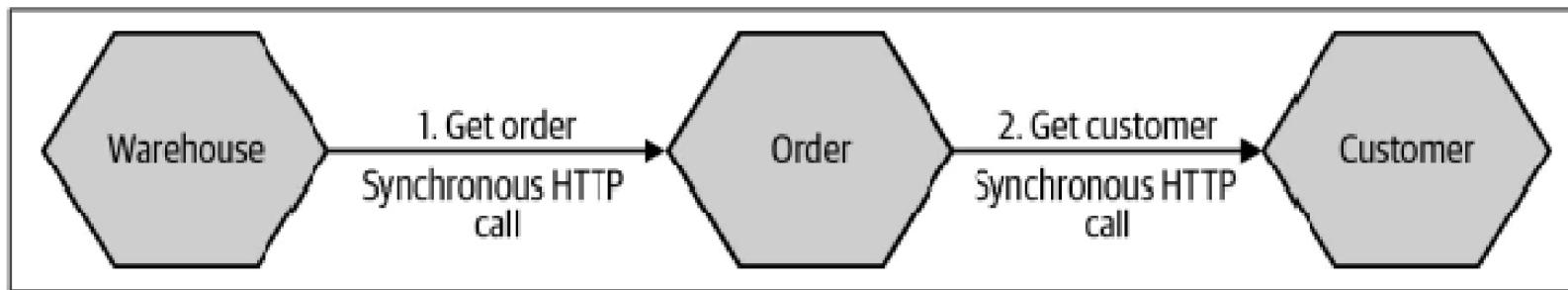


înainte

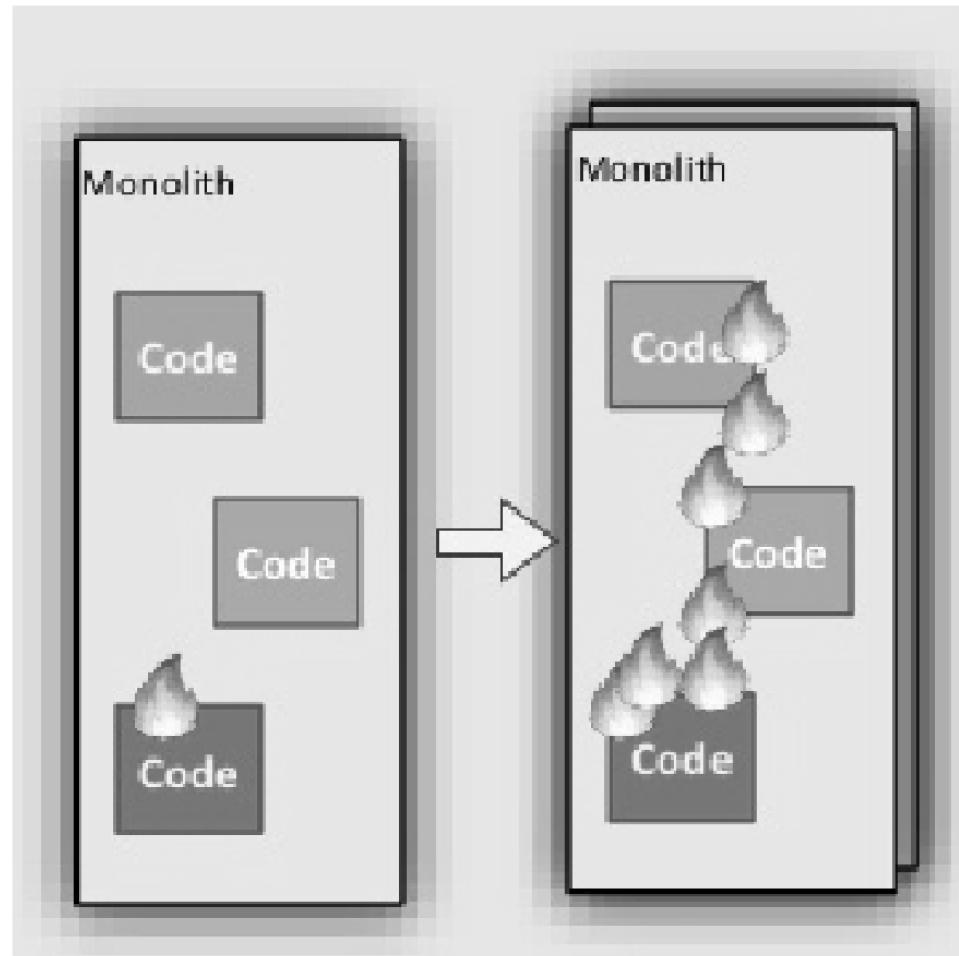


după

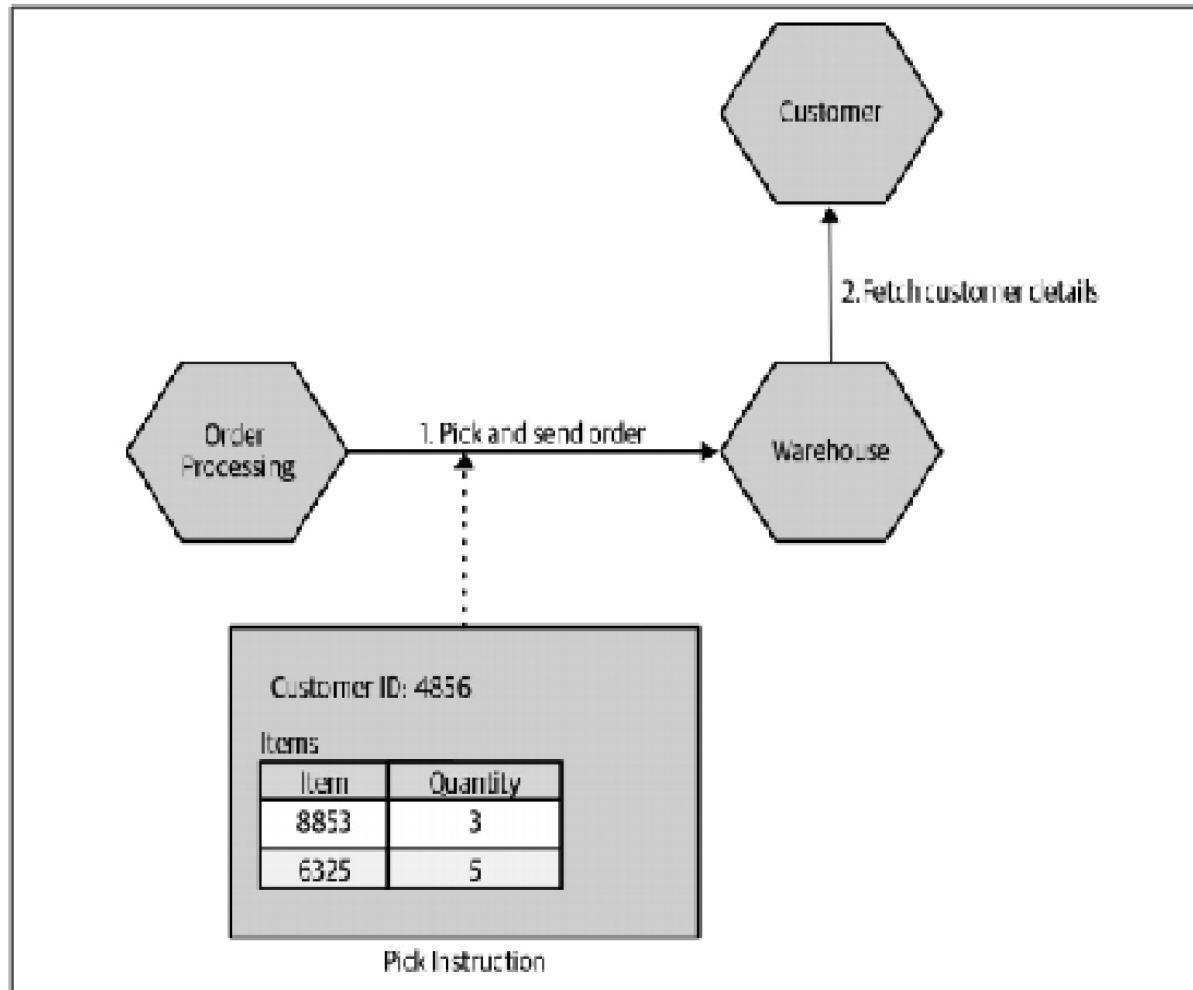
Cuplare temporală



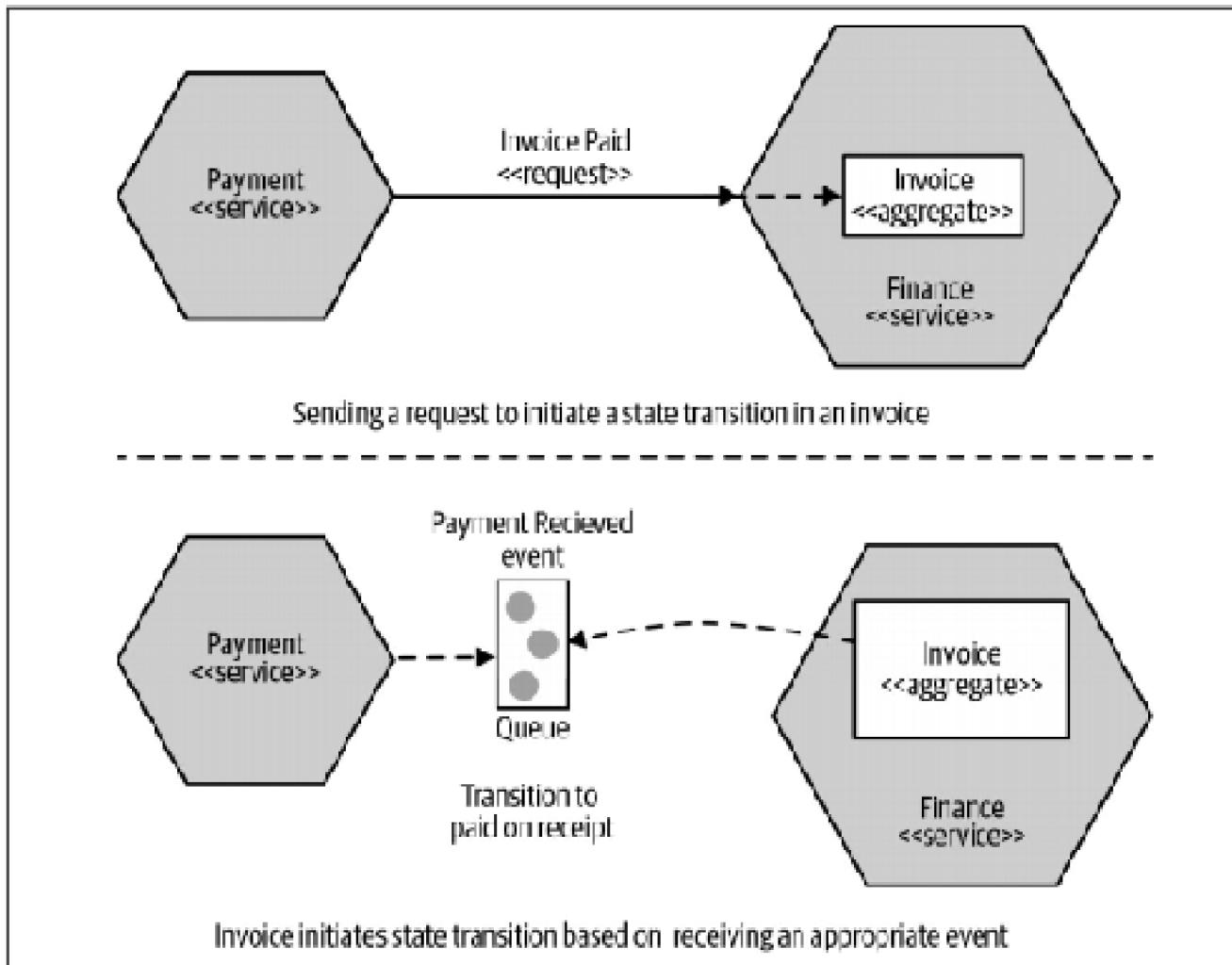
Cuplarea la instalare



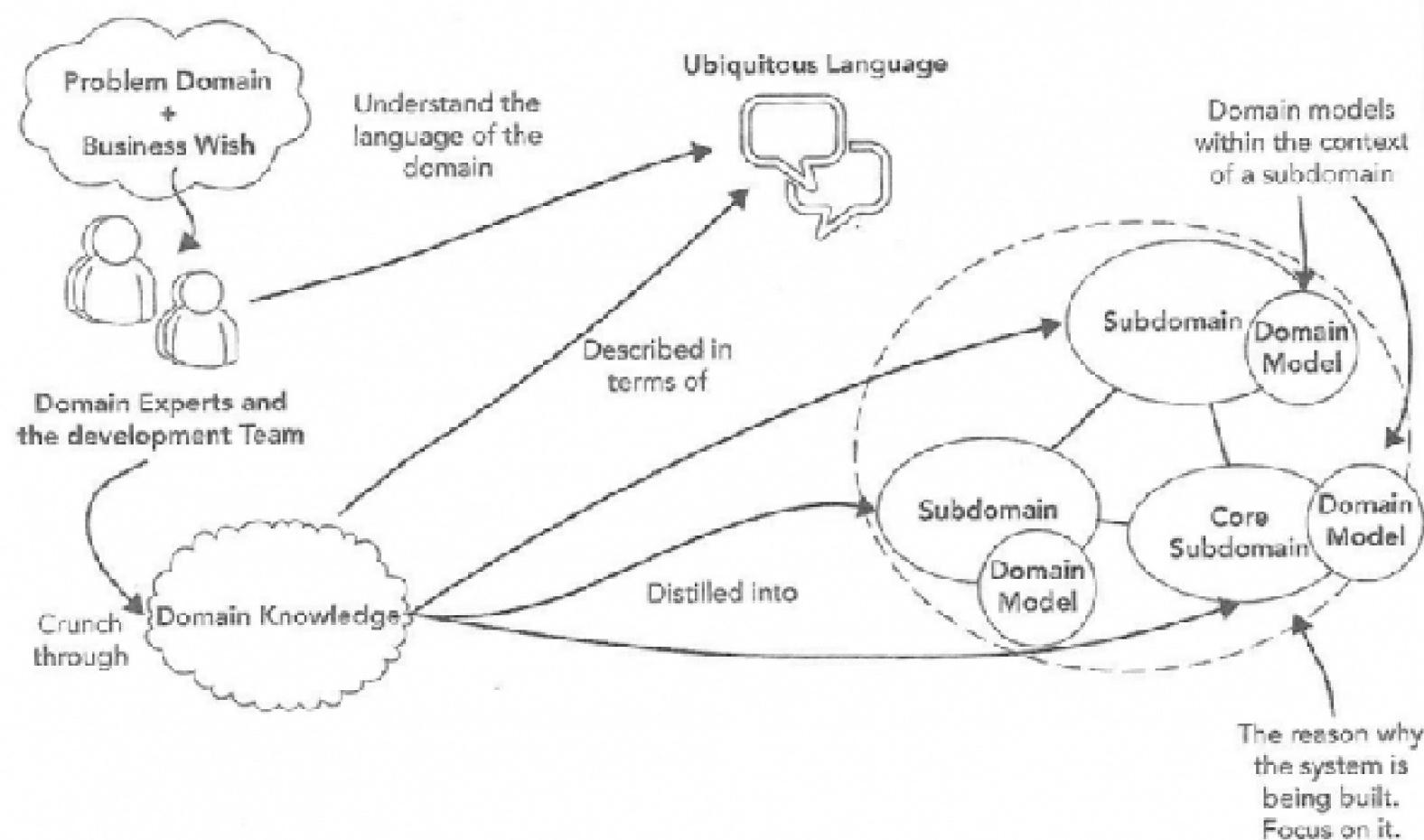
Cuplarea domeniilor - soluția 1



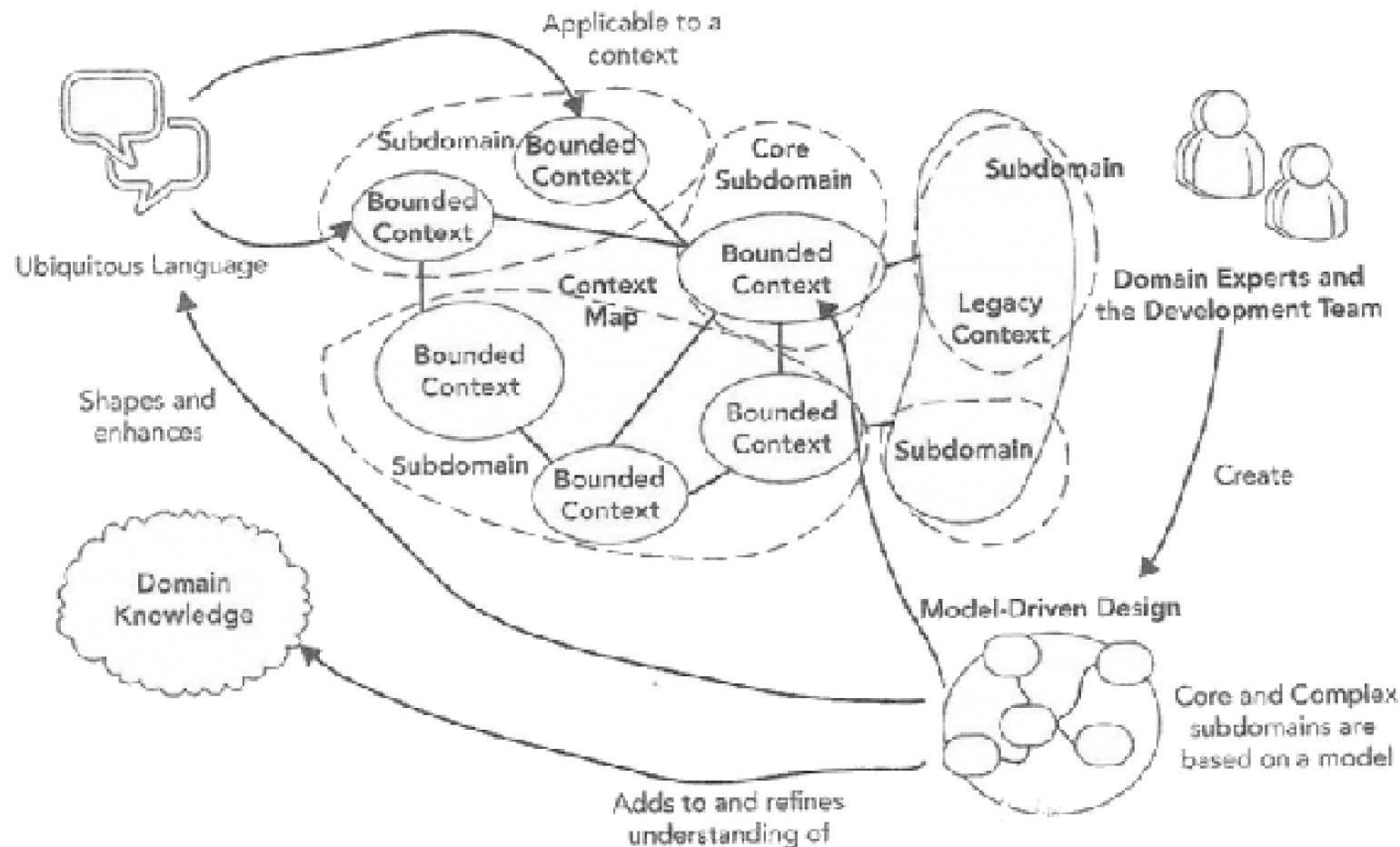
Aggregatul



Context margin



Agregat, context mărginit, microservicii



Categorie: ce-i cu API-urile astăzi ???

- Apucături vechi
- Interoperabilitate cu soluții vechi
- Utilizare cu justificare
- API spaghetti

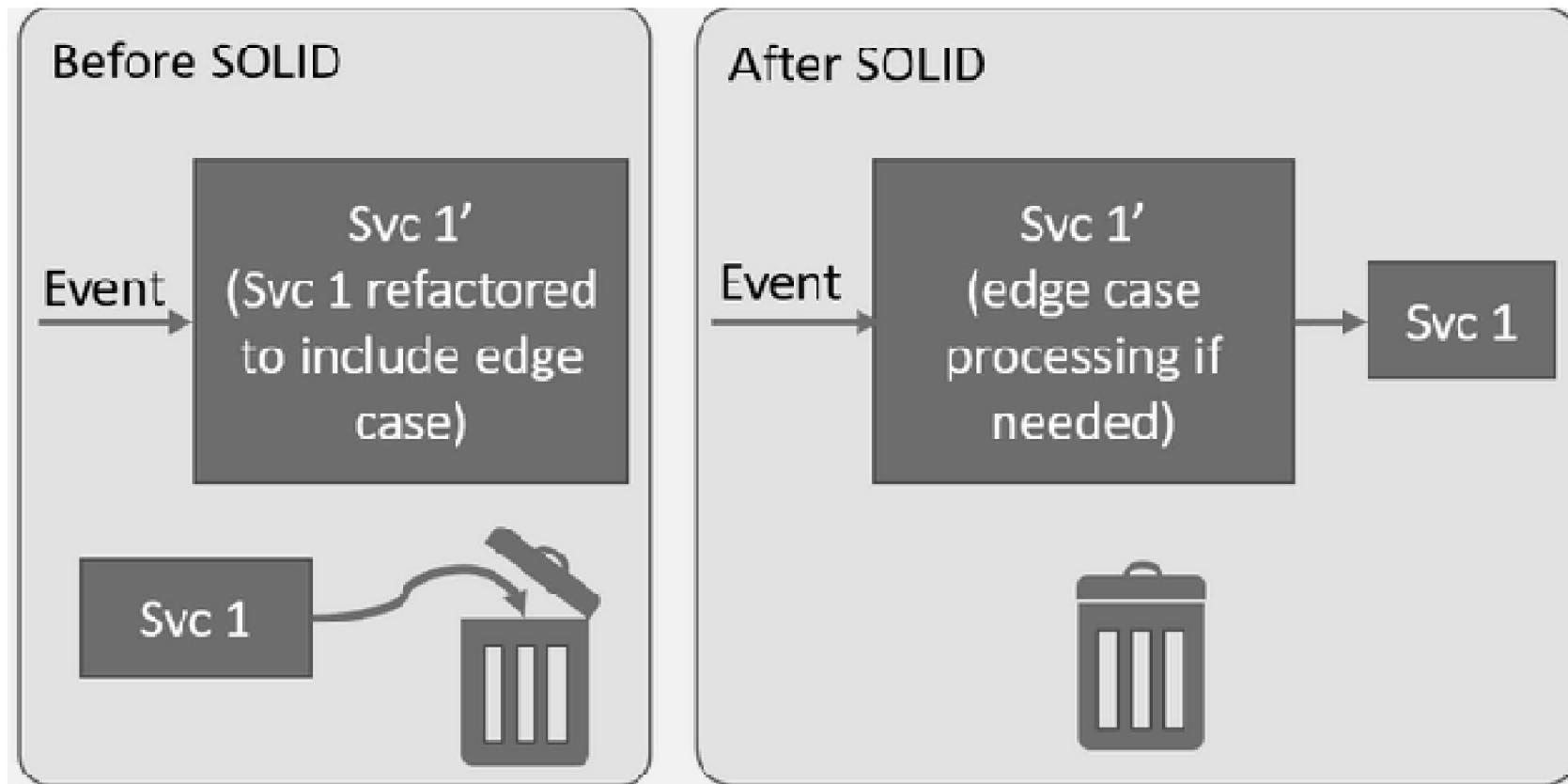
Sisteme Distribuite

Cursul 7

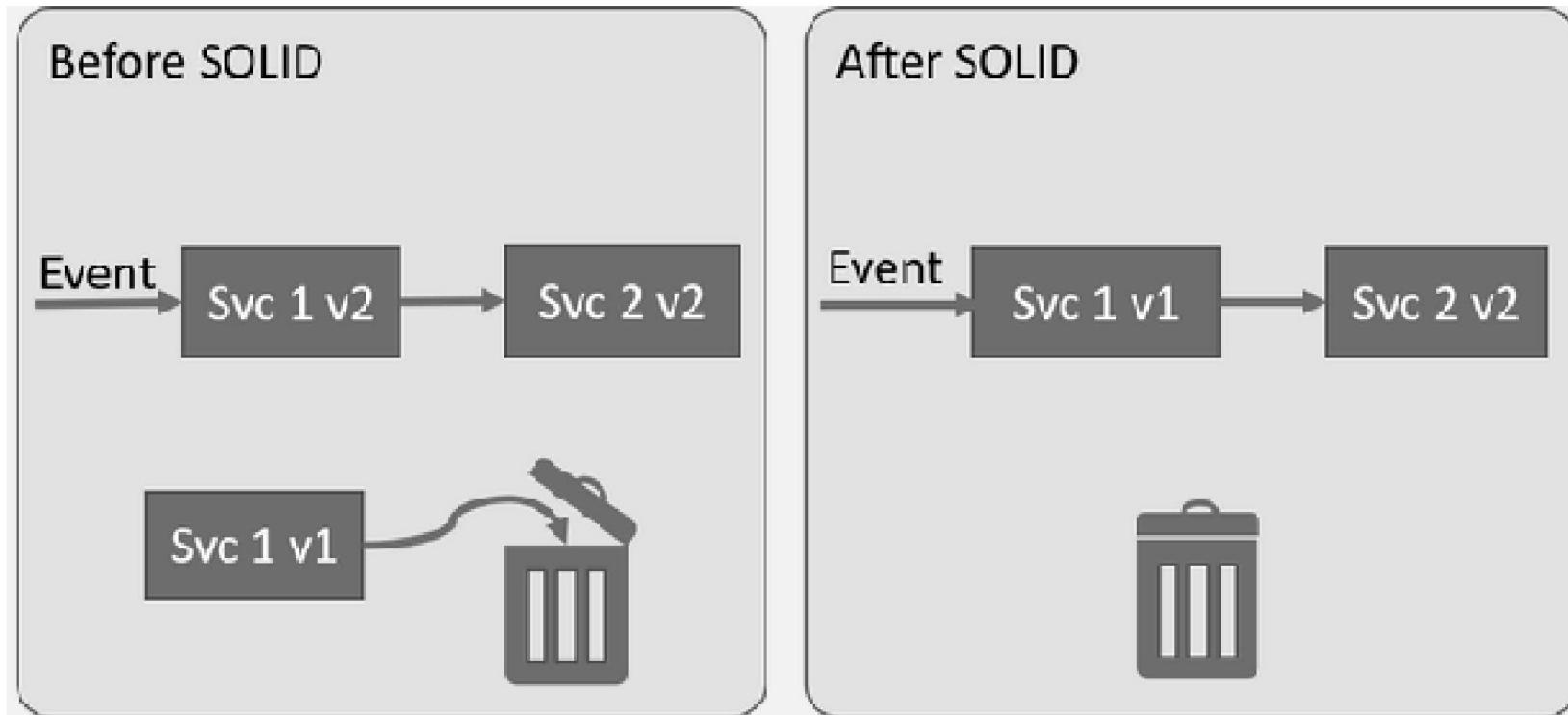
Principul responsabilității unice



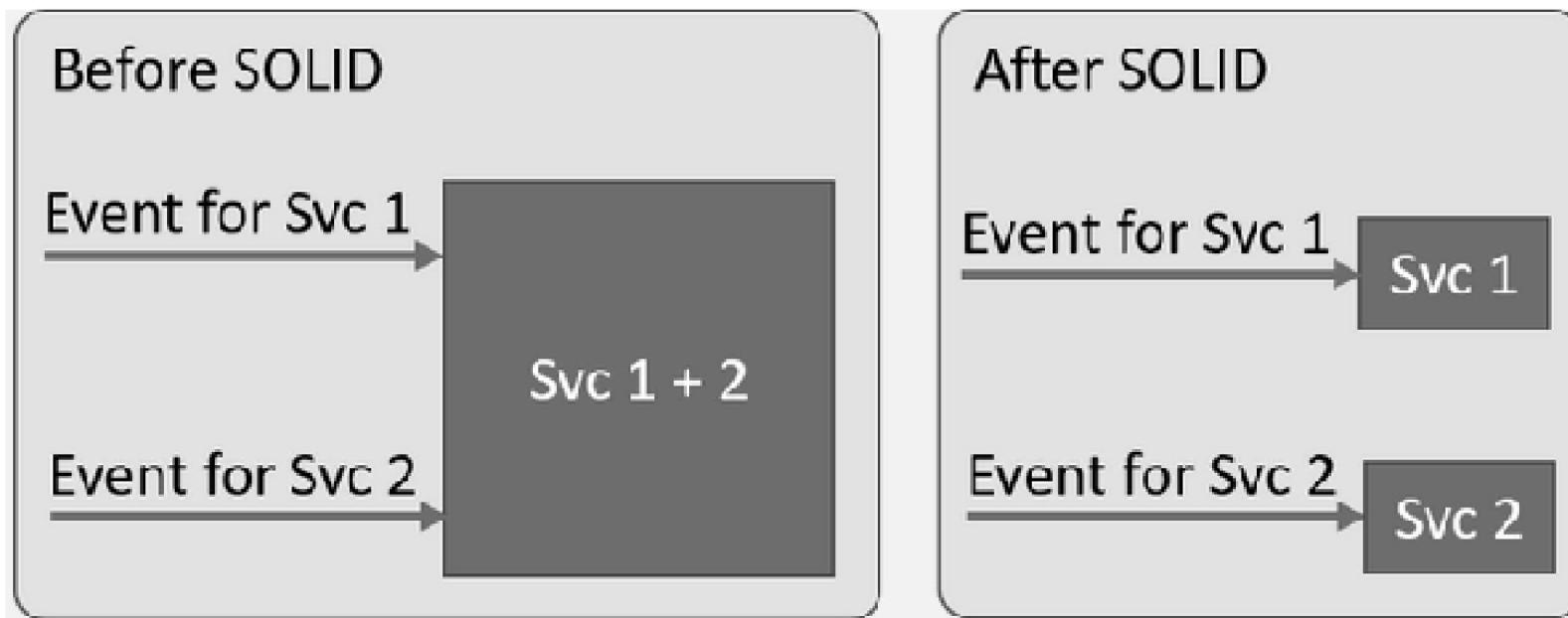
Deschis pentru extindere închis pentru modificare



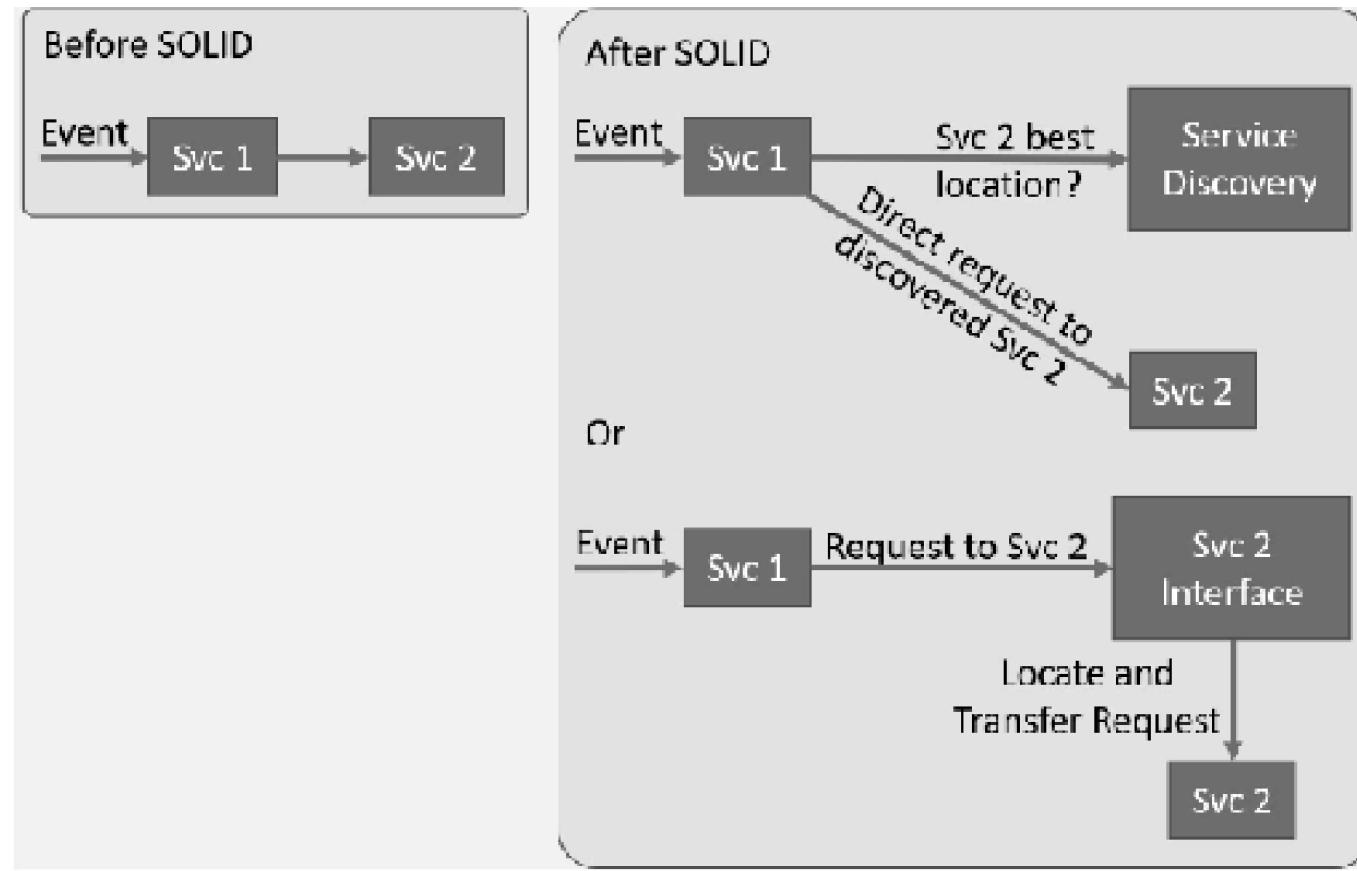
Substituția Liskov



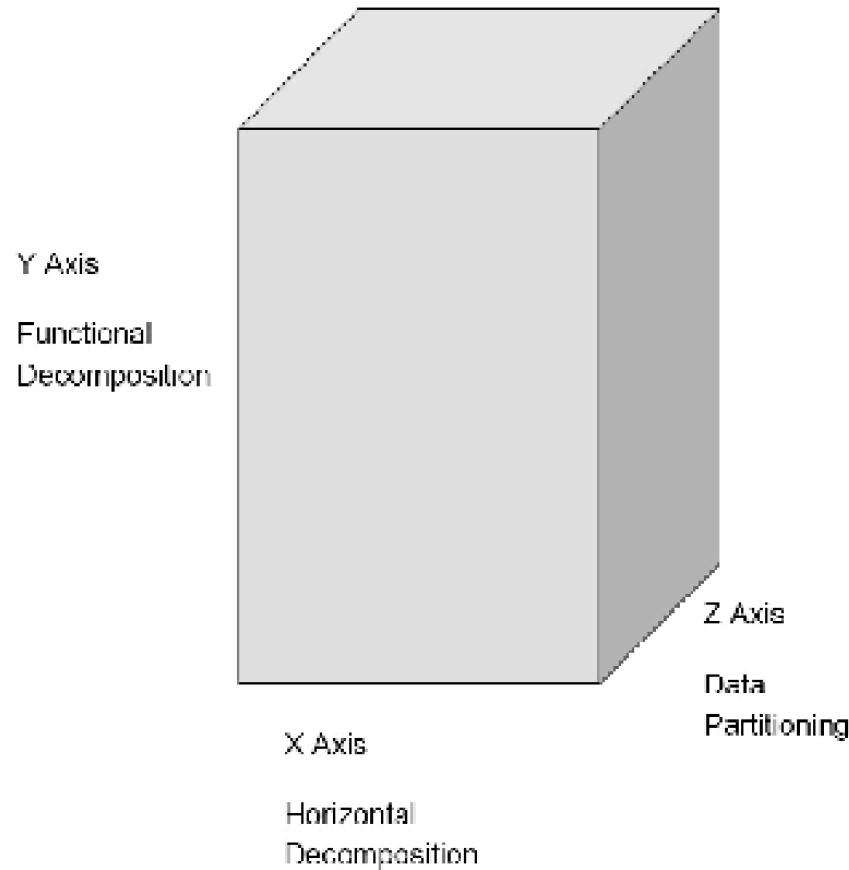
Separarea interfețelor



Controlul invers

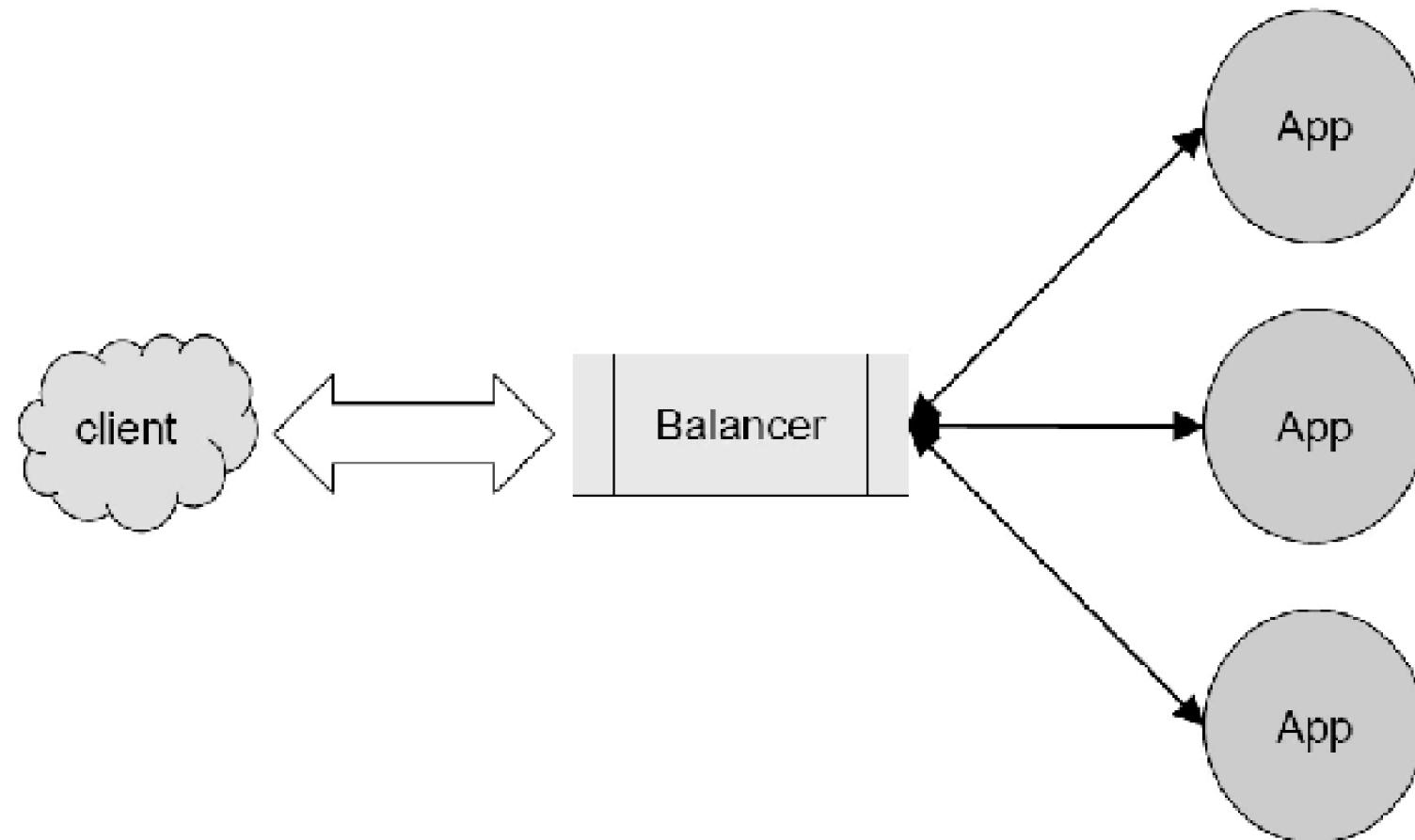


Principii de scalare a microserviciilor

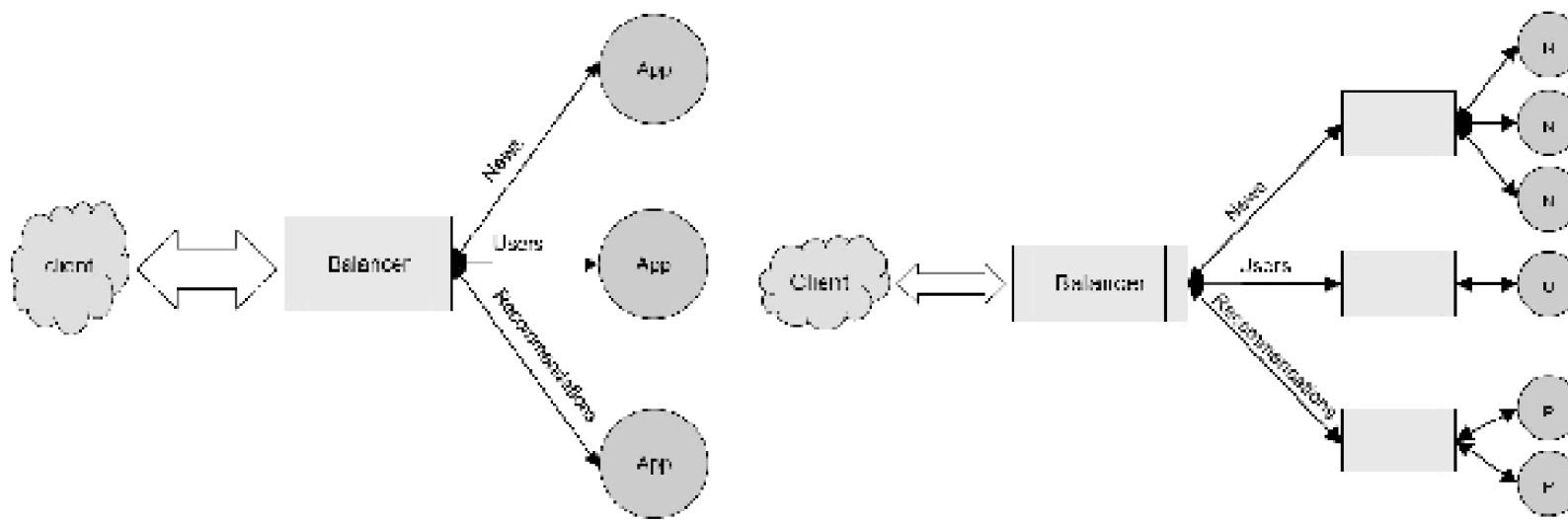


The Scale Cube

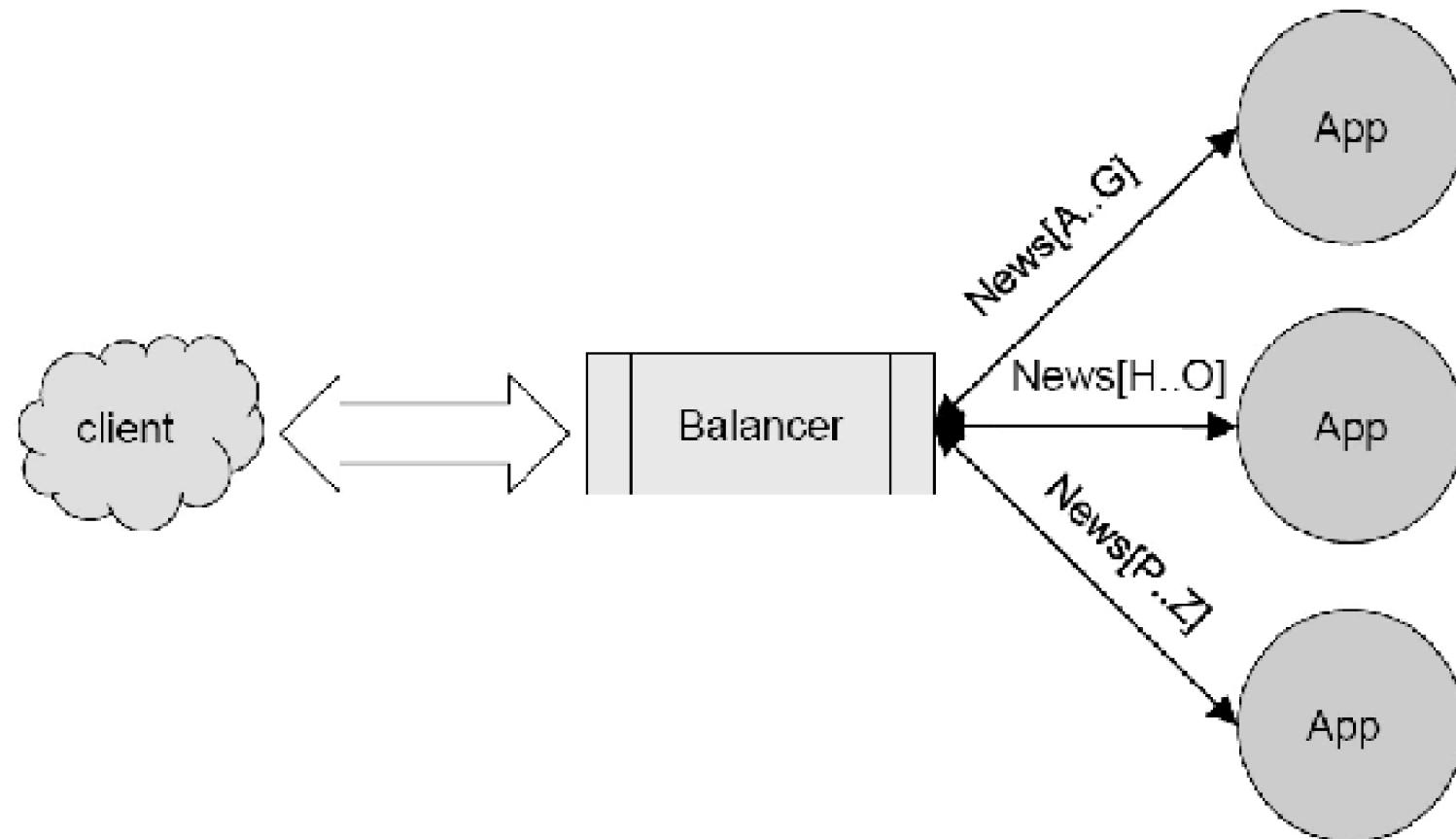
Cubul scalării - axa X



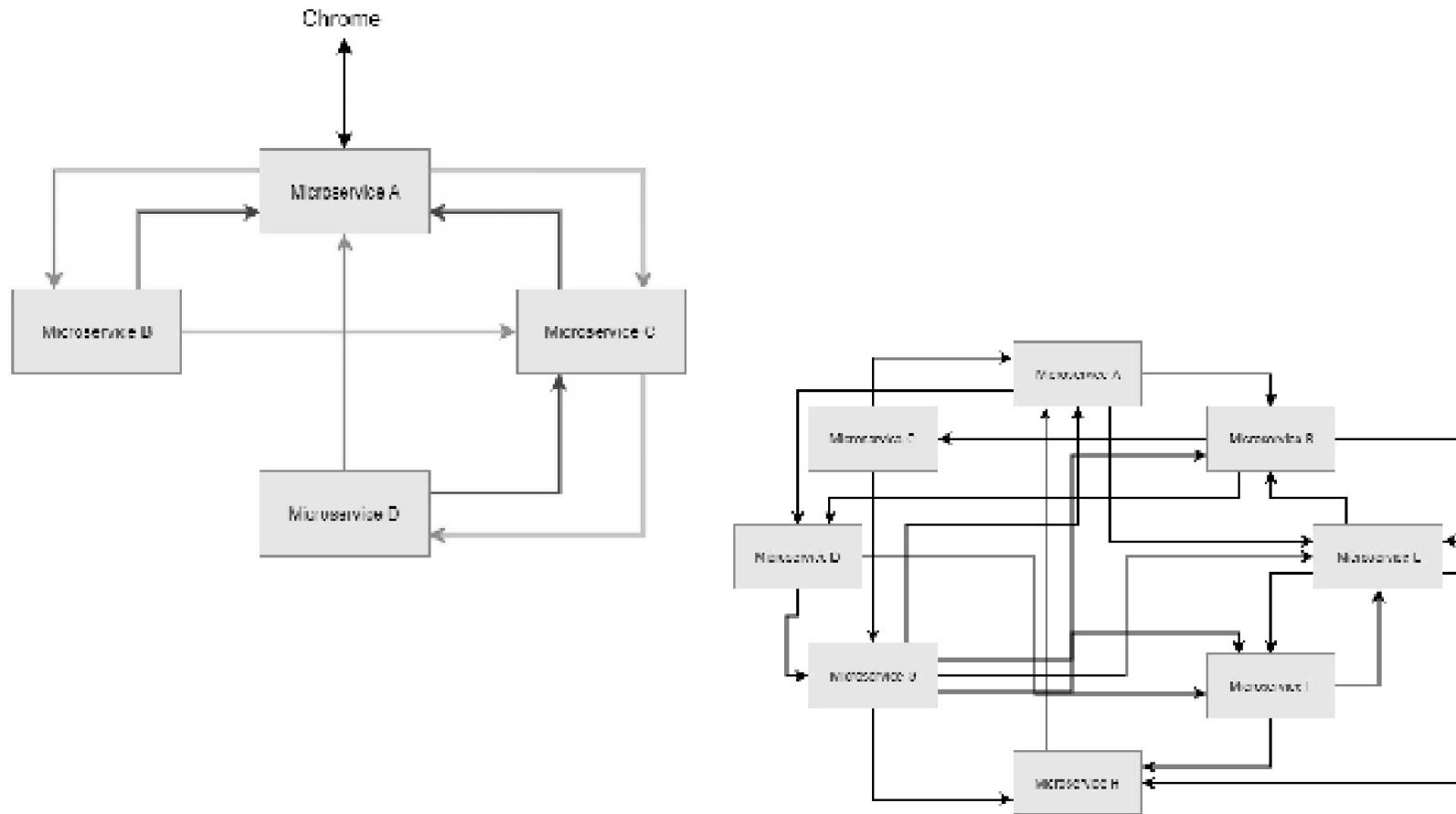
Cubul scalării - axa Y



Cubul scalării - axa Z



Steaua Mortii

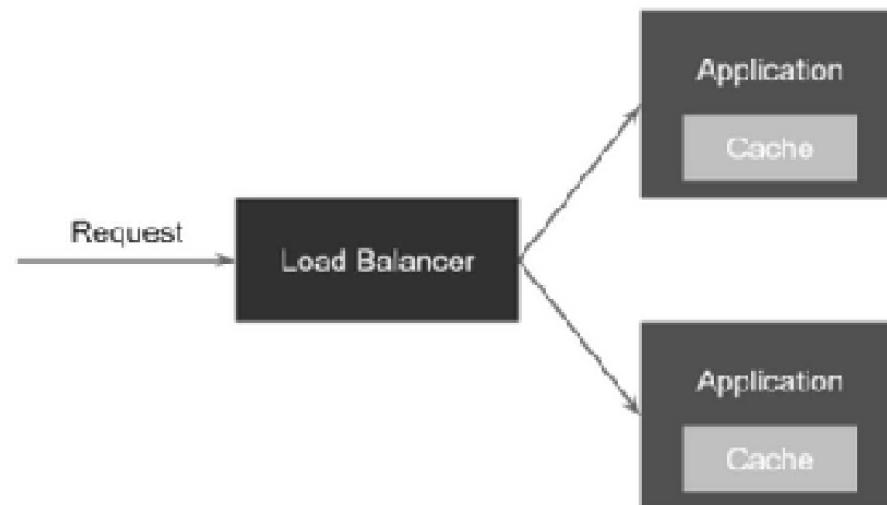


Mecanisme de caching

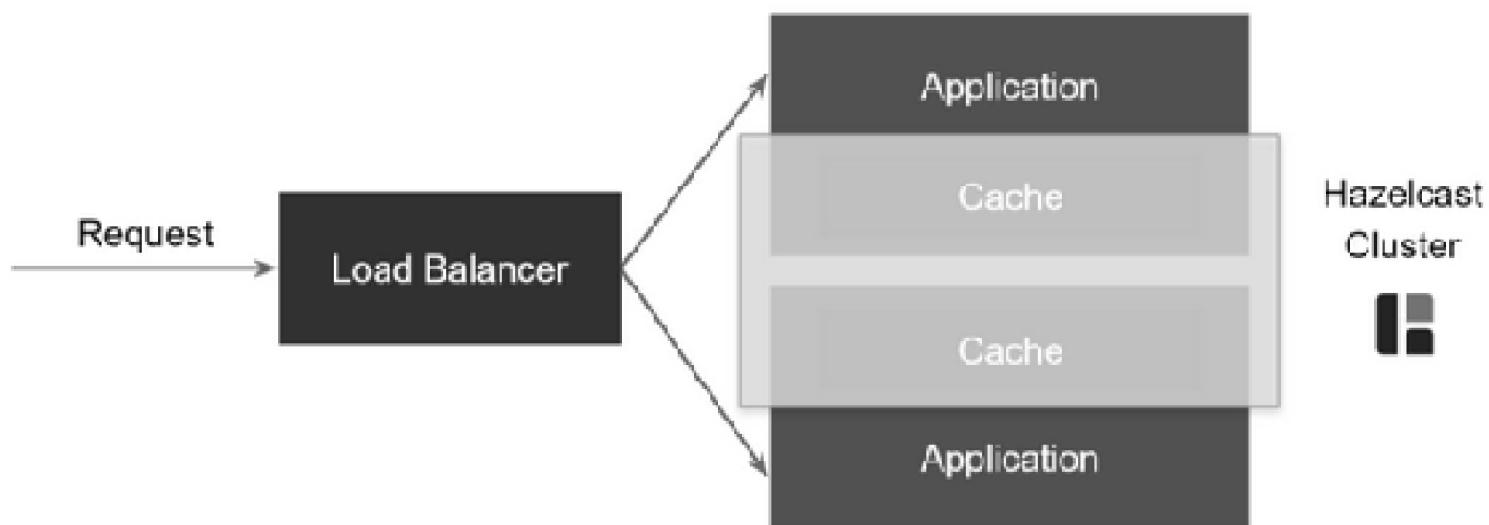
- caching?
- când?
- hand made
- Red Hat JBoss Data Grid

Modelul de proiectare - cache încapsulat (embeded)

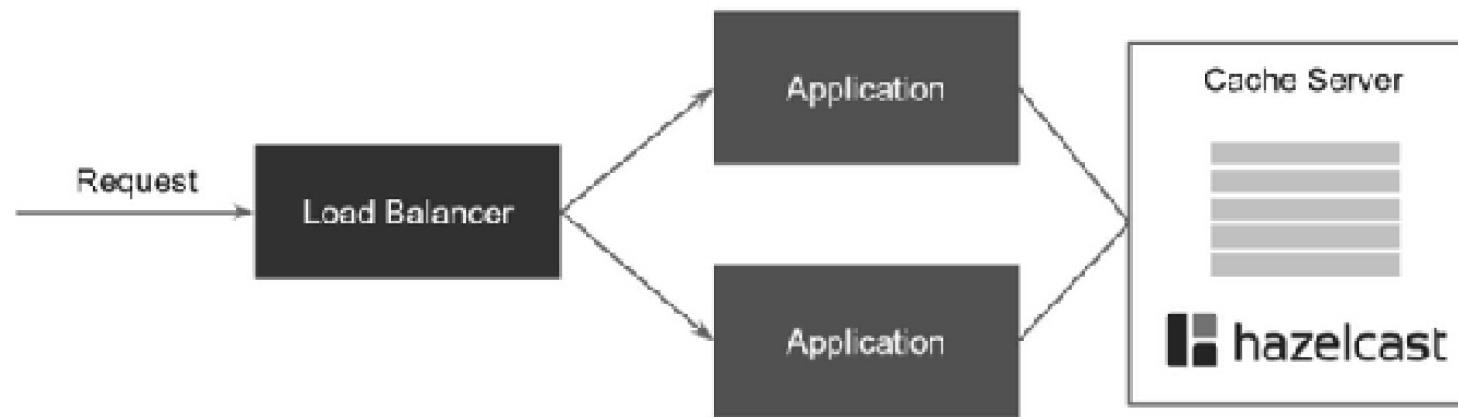
1. Cererile vin
2. modul lb transfera cererea
3. Apoi serviciul verifică duplicate
 1. Daca da valoarea
 2. altfel calcul



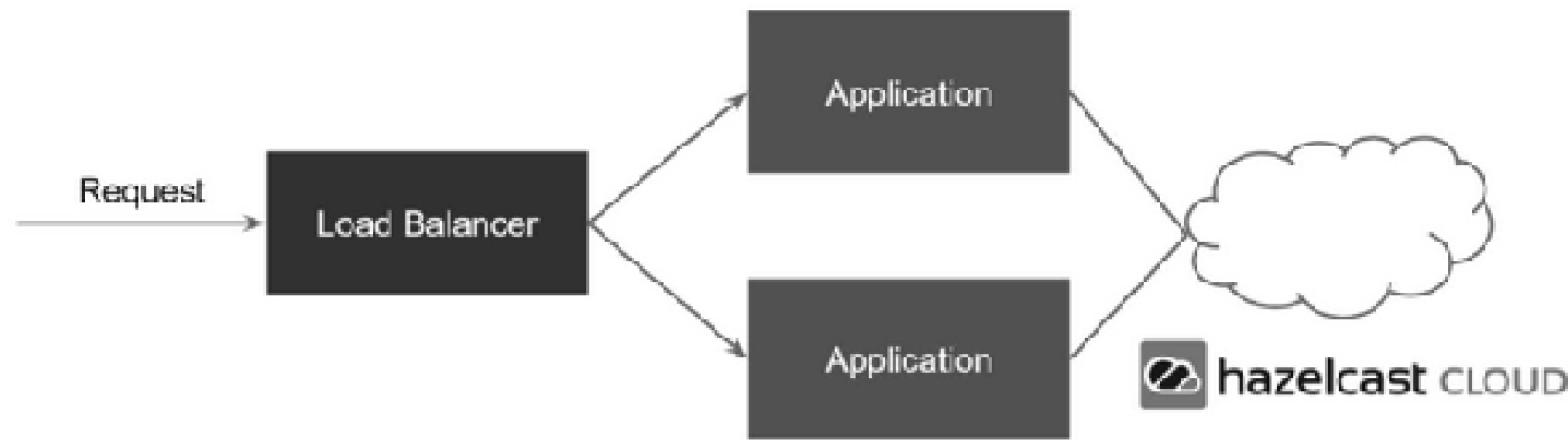
Modelul de proiectare - cache încapsulat distribuit (cache comun)



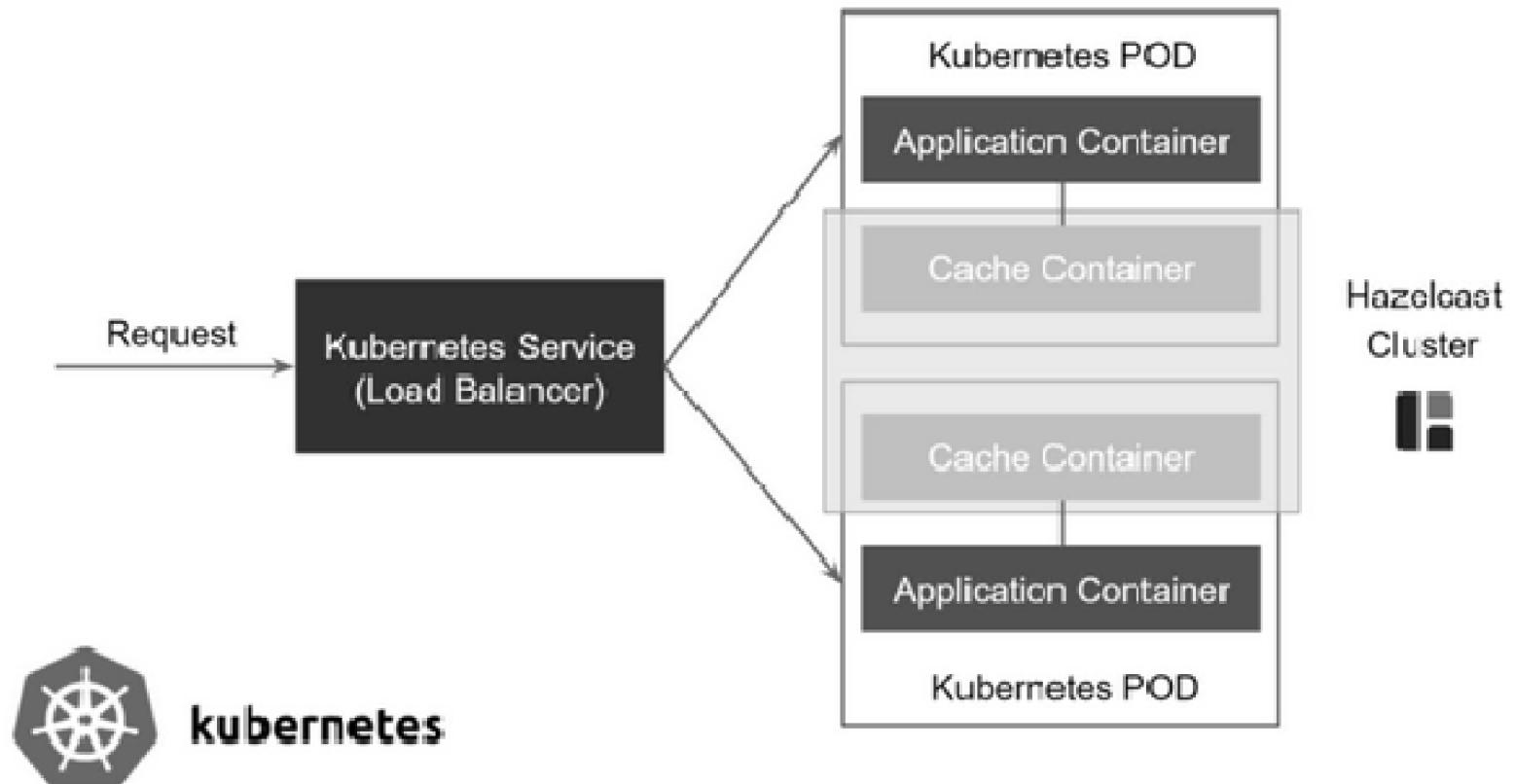
Modelul de proiectare cache client server



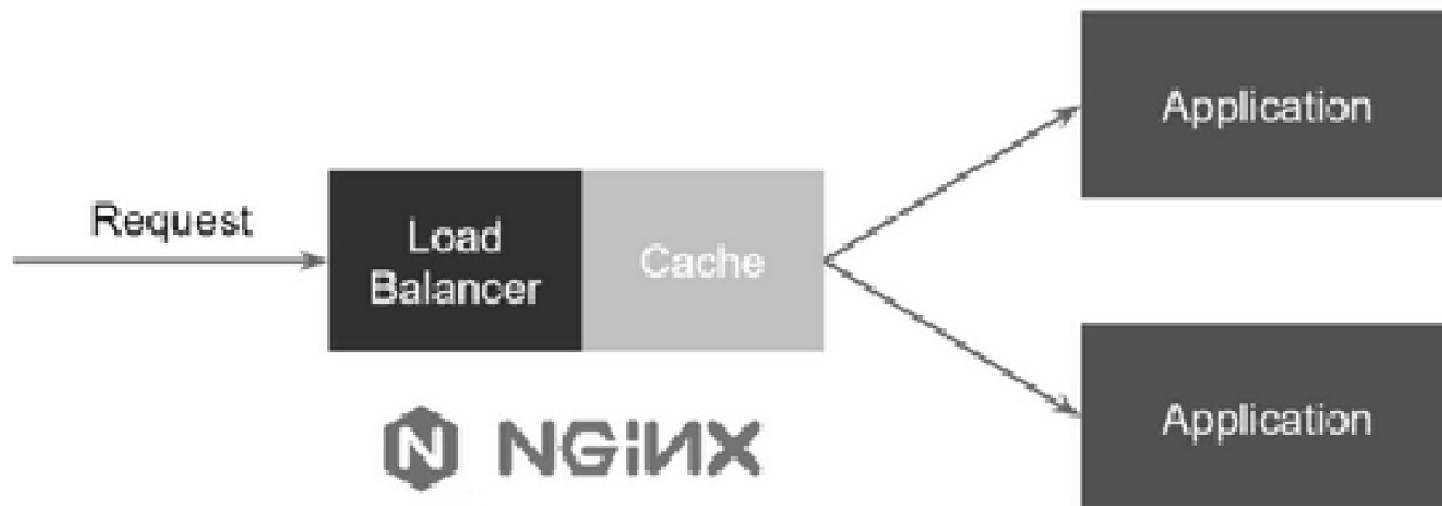
Modelul de proiectare cache în nor



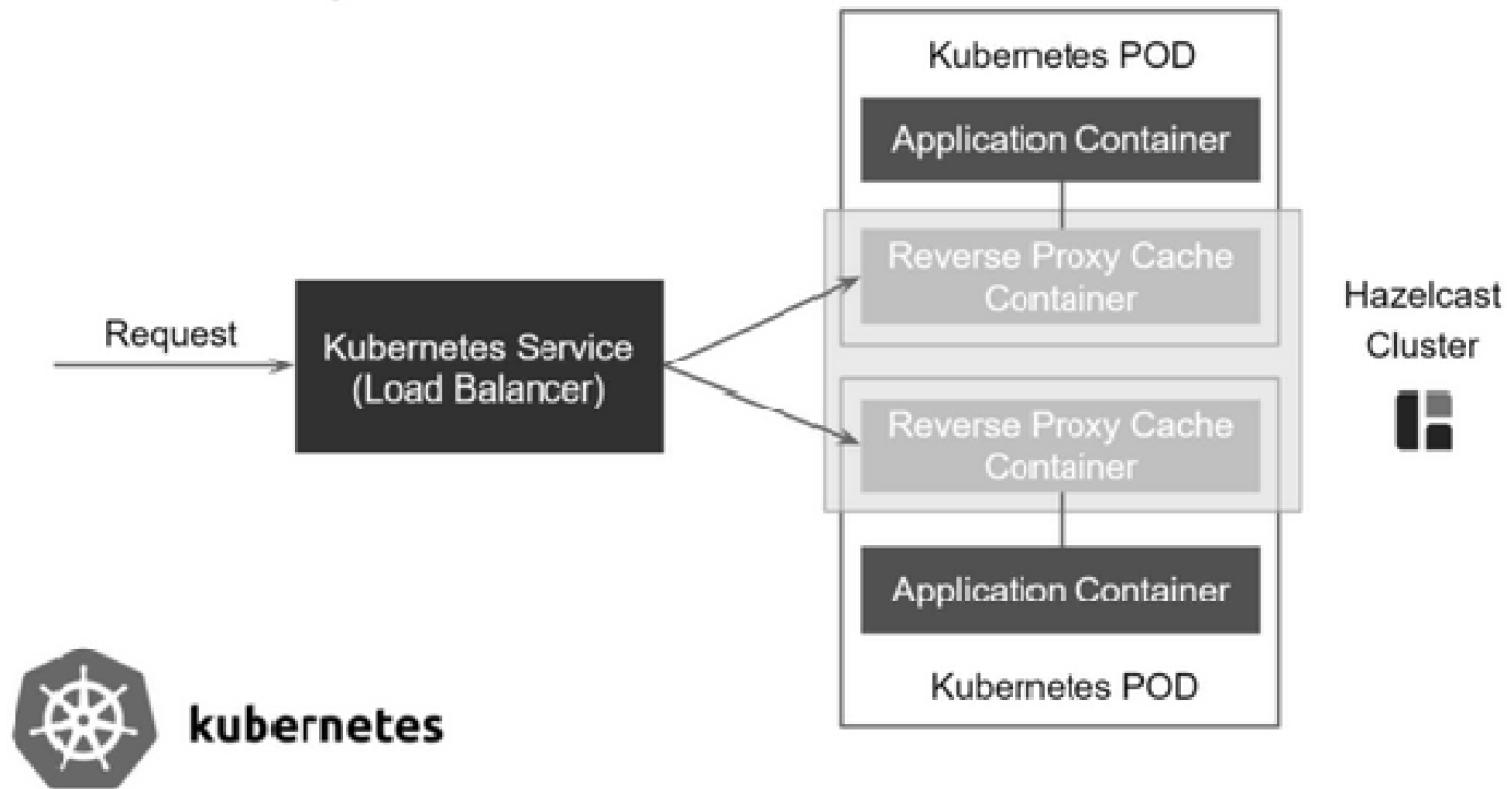
Modelul atașamentului (sidecar)



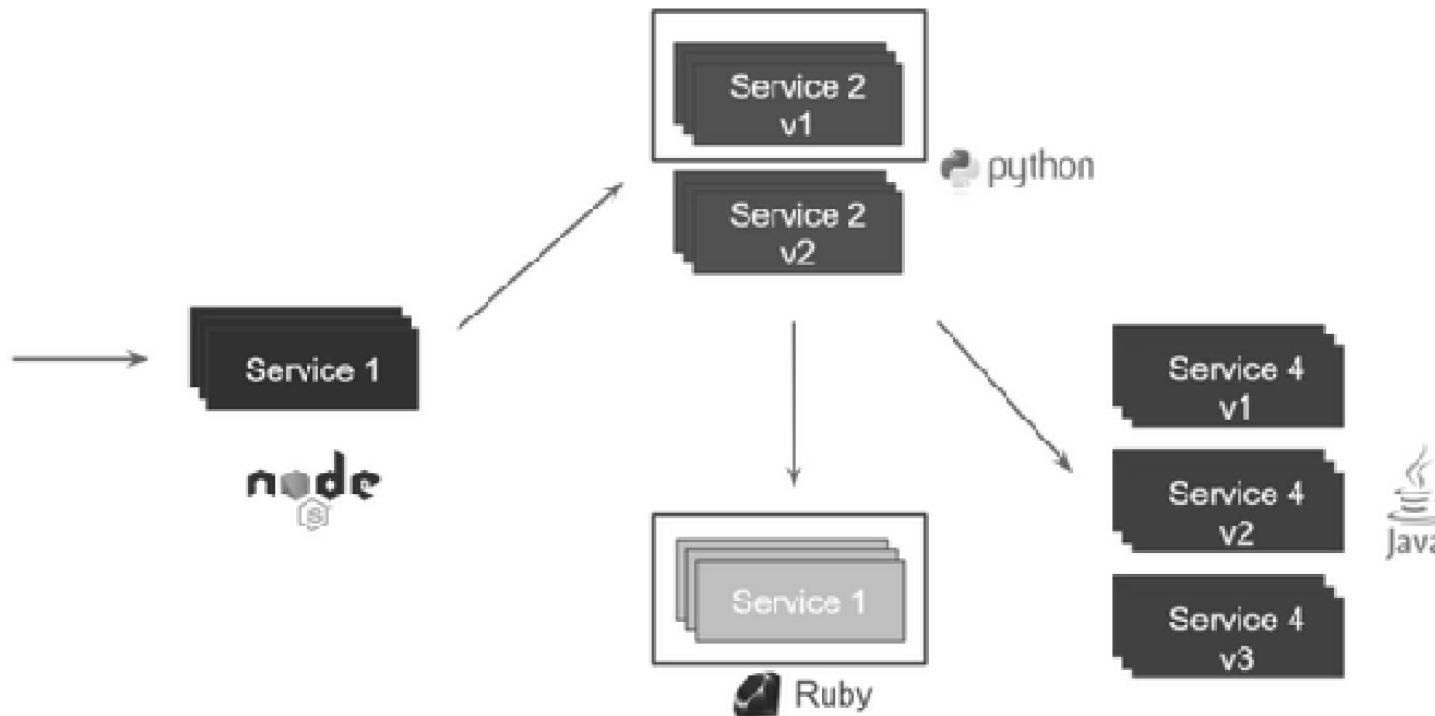
Modelul de proiectare cu intermediar invers (inverse proxy)



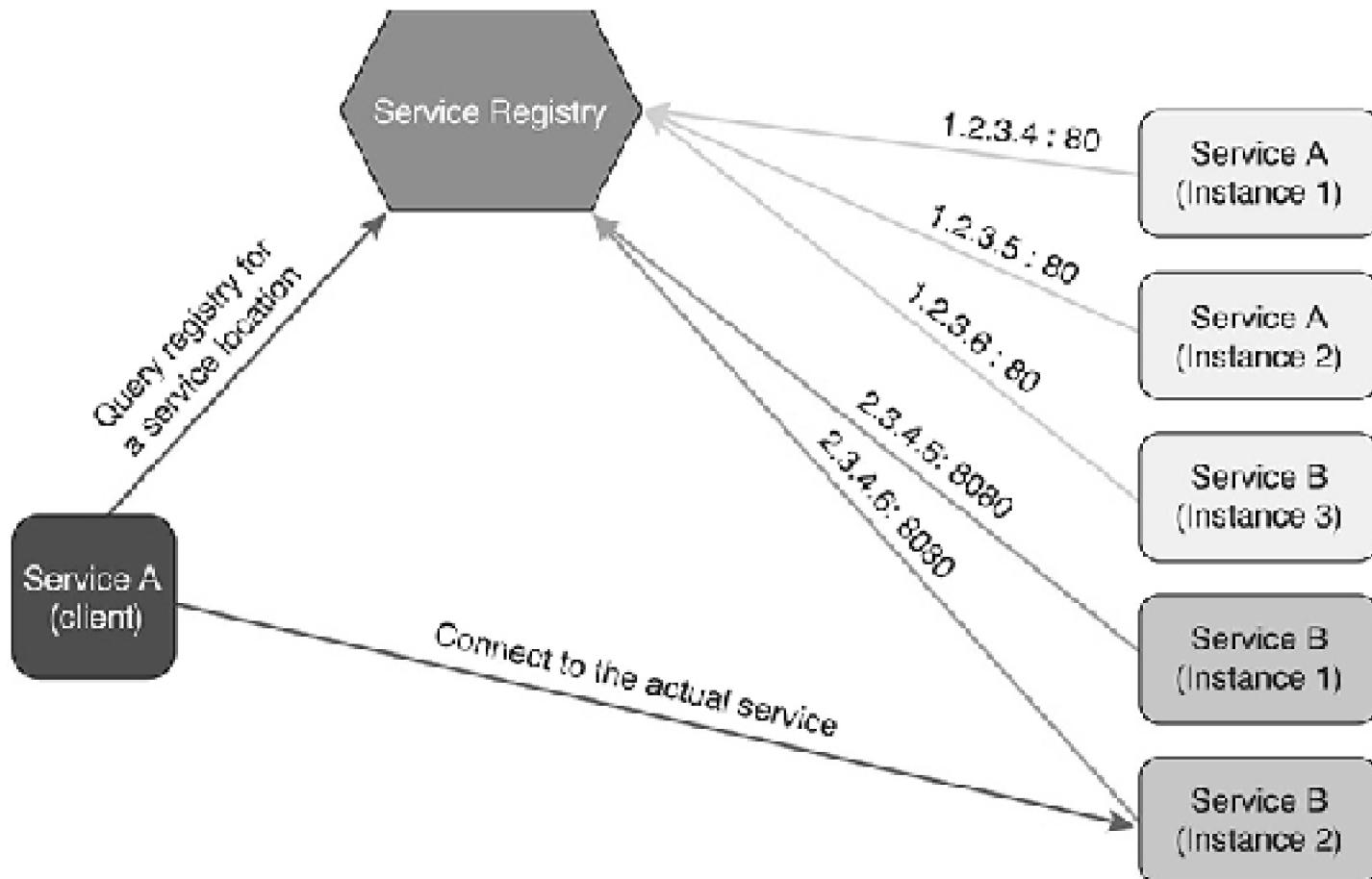
Intermediarul invers & atașamentul



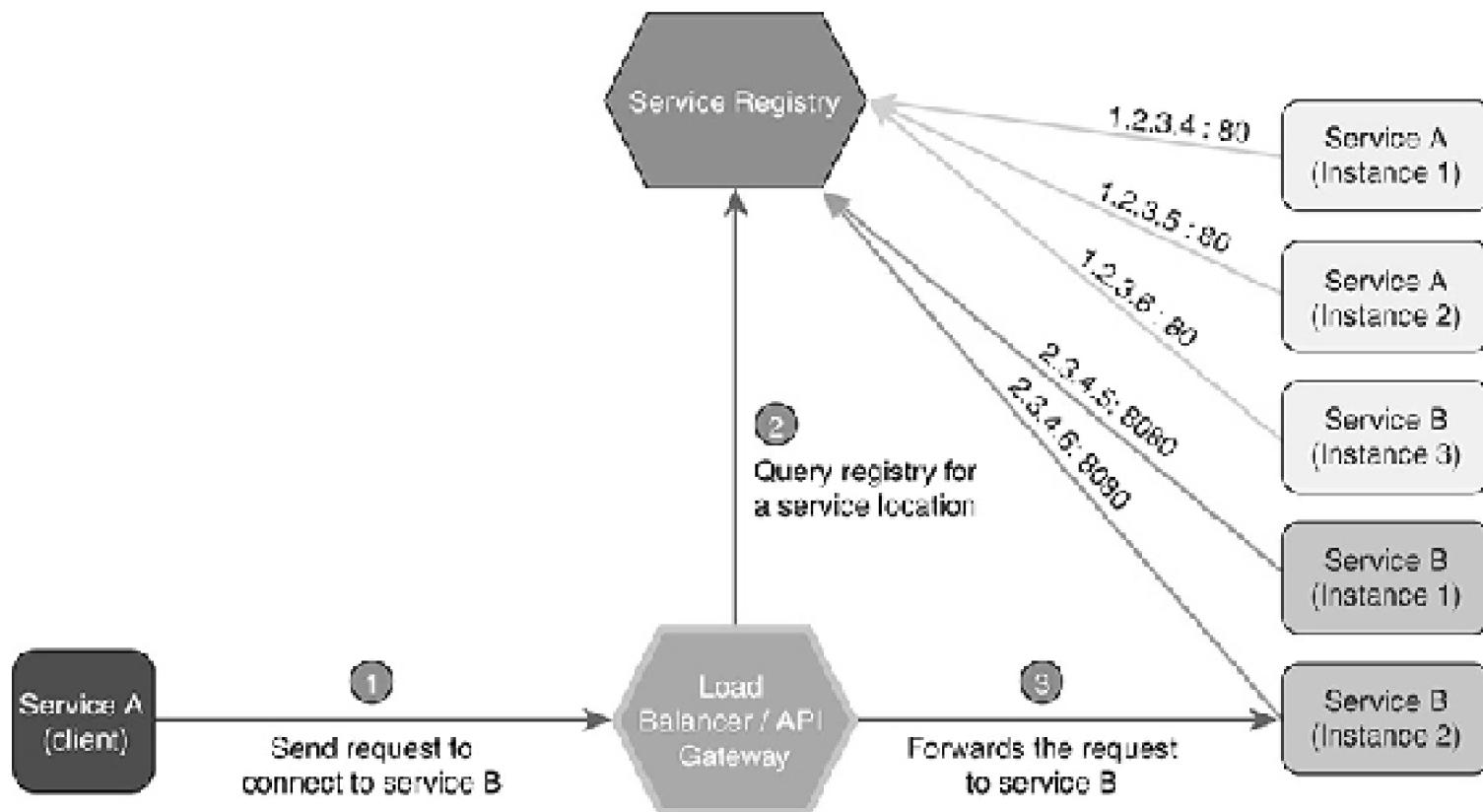
Intermediarul invers & atașamentul



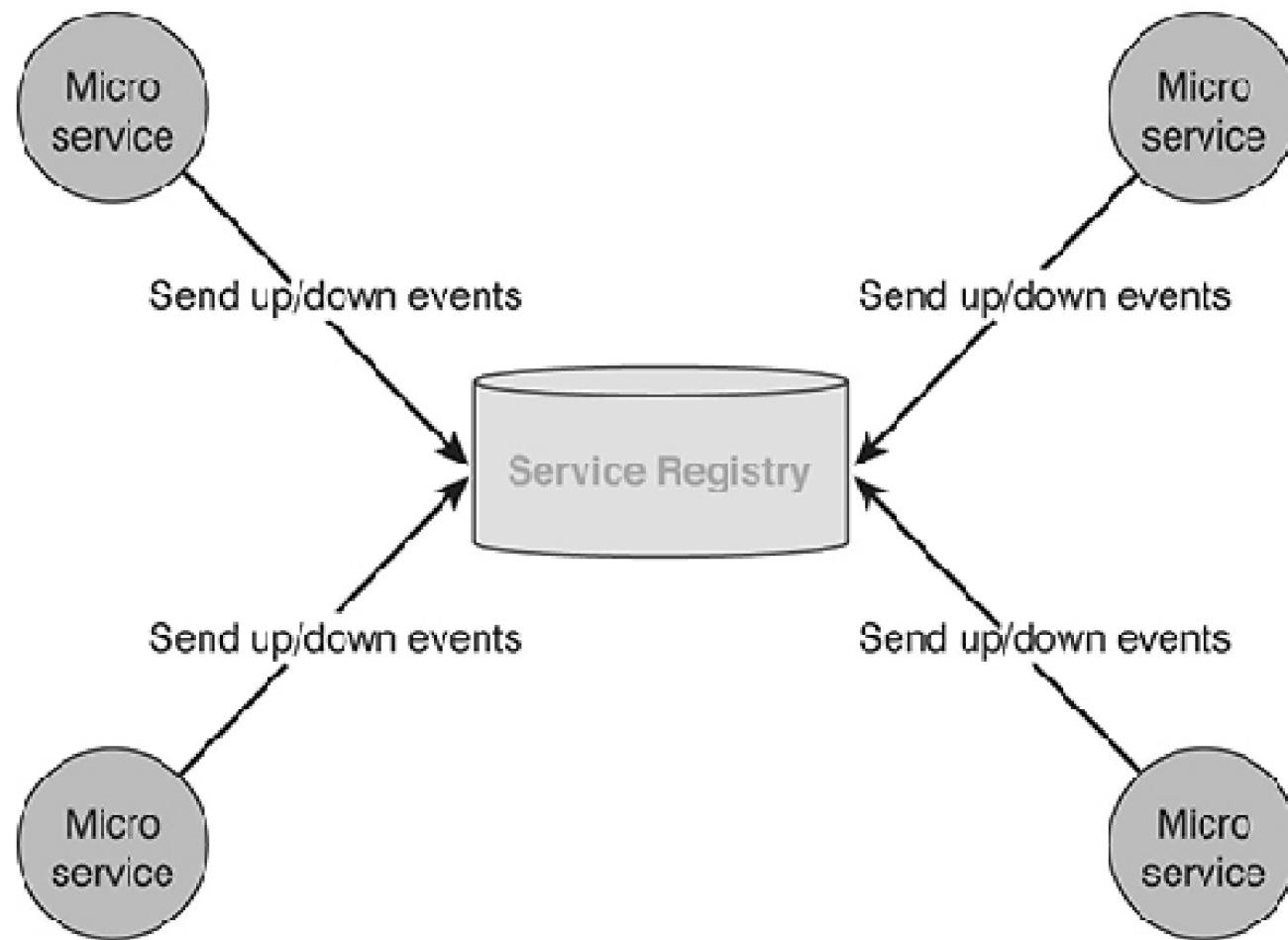
Descoperirea la nivel de client



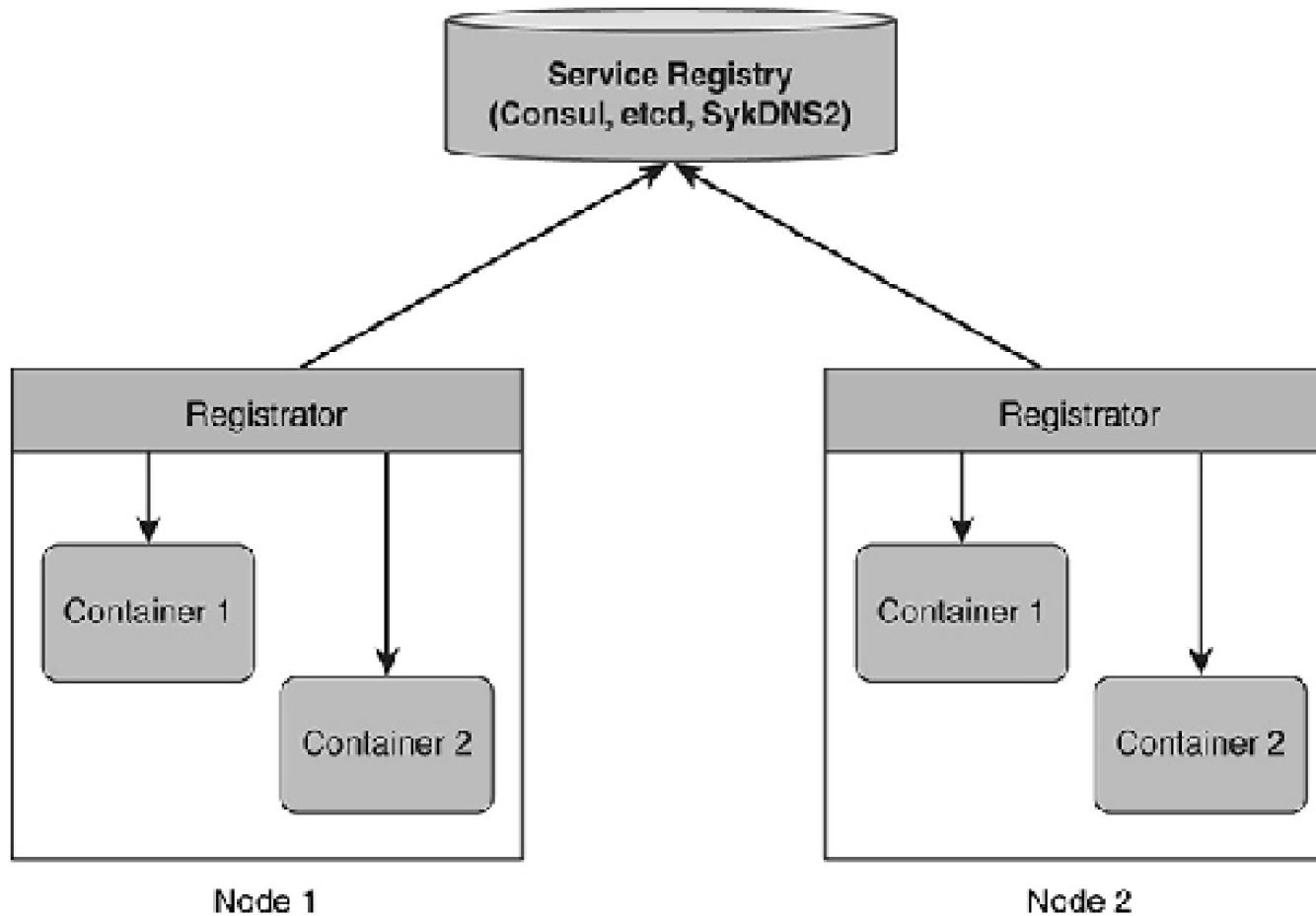
Descoperirea la nivel de server



Registre de servicii - Autoînregistrare



Registre de servicii - Instrumente Externe

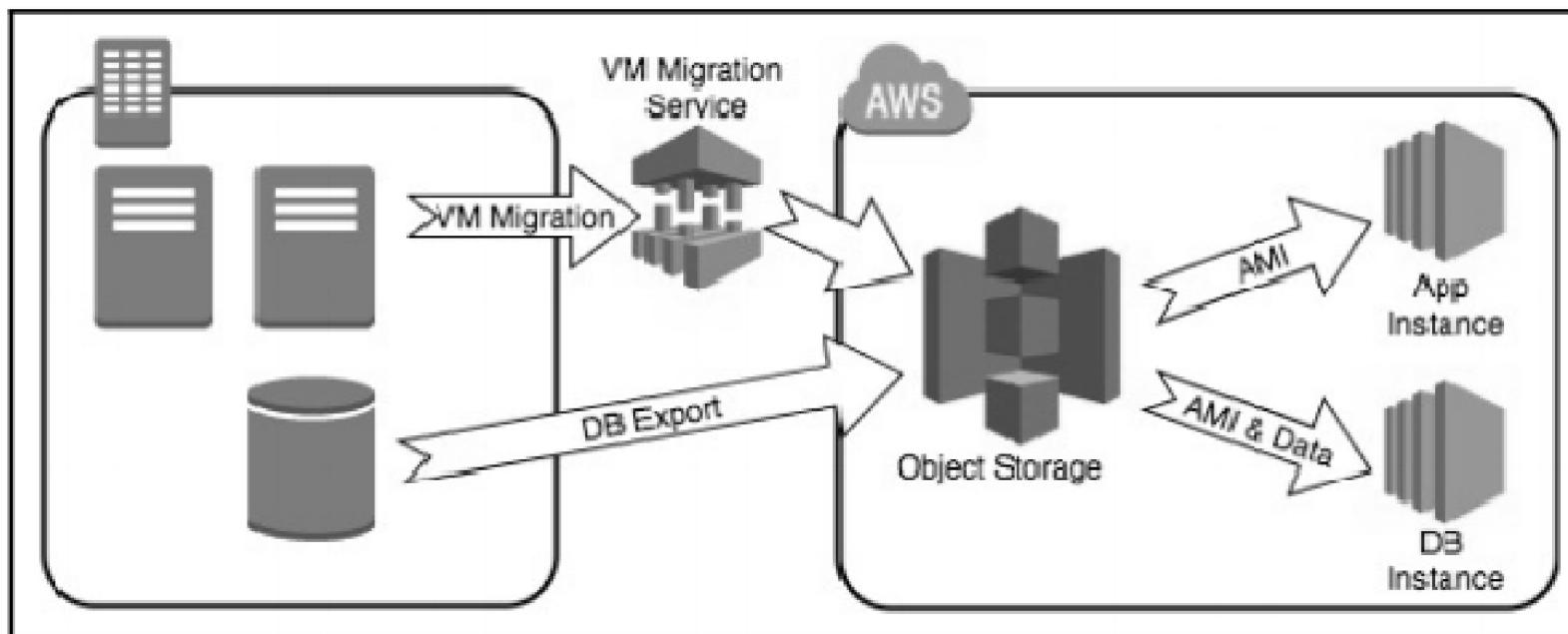


Registrator (<https://github.com/gliderlabs/registrator>)

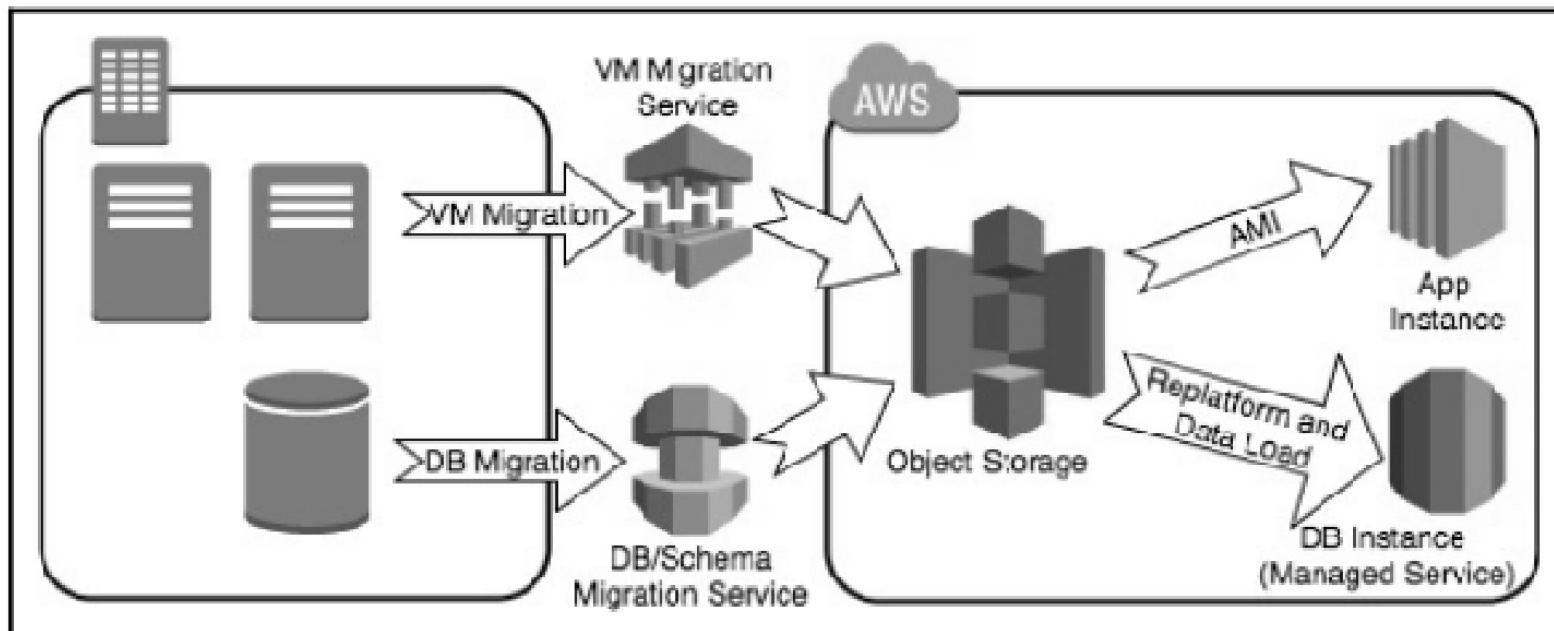
Modele de migrare a aplicațiilor

- Rehost
- Replatform
- Repurchase
- Refactor
- Retire
- Retain

Rehost sau lift-and-shift



Replatform

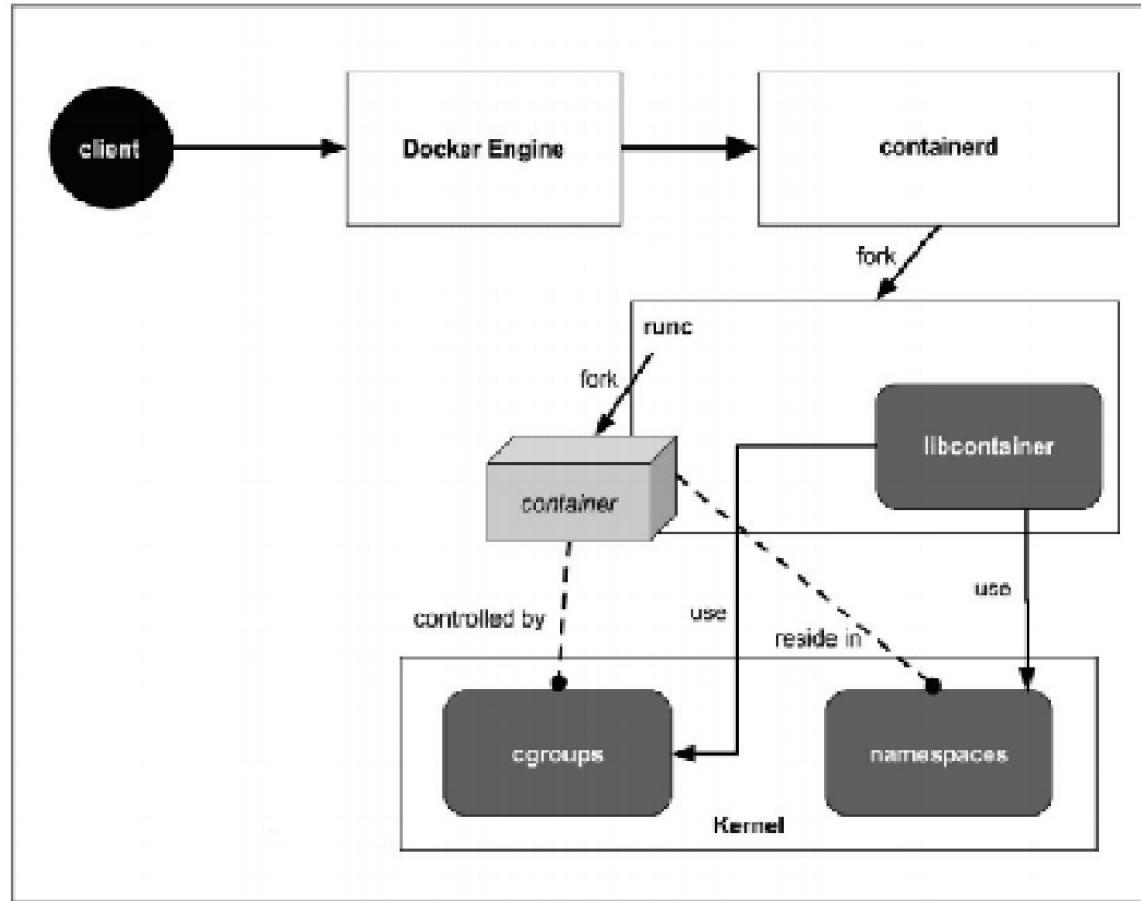


Sisteme distribuite

Mihai Zaharia

Cursul 8

runC

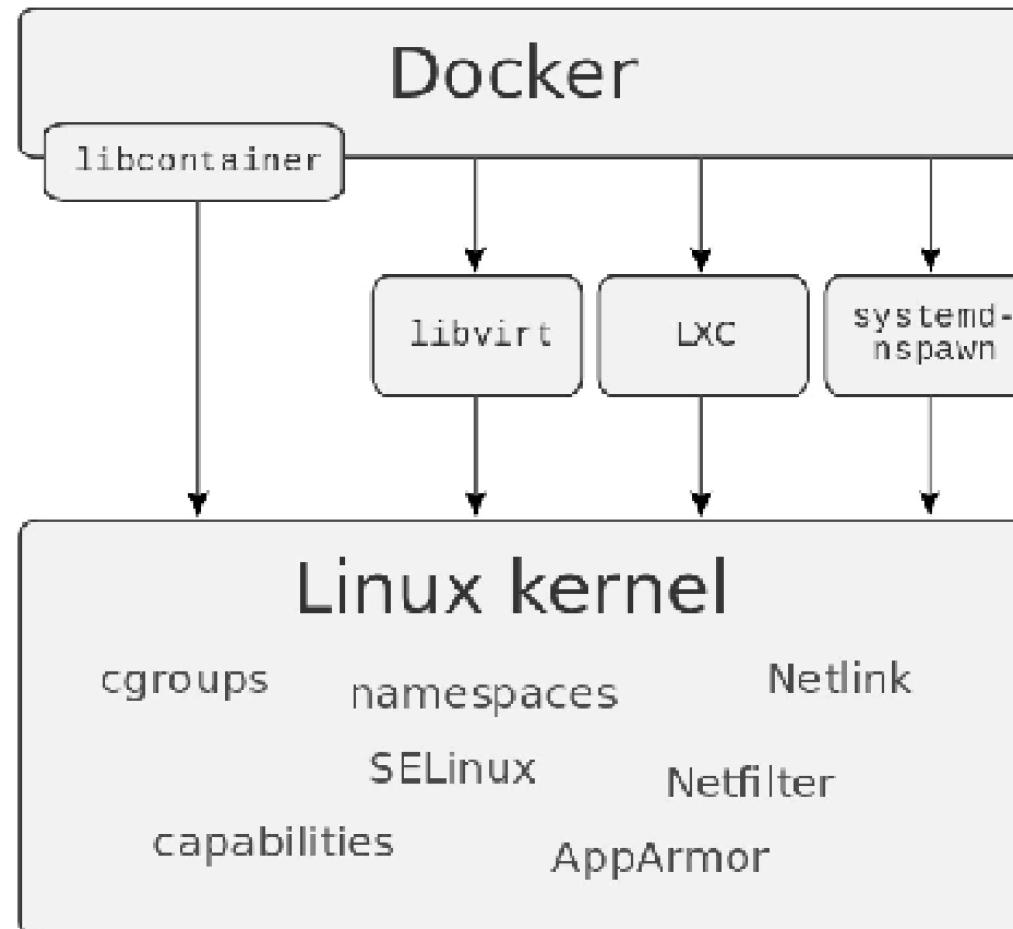


Ce oferă Linux pentru containerele docker

Detalii rețalie mașină Docker - Linux

Spații de lucru utilizate de mașina Docker

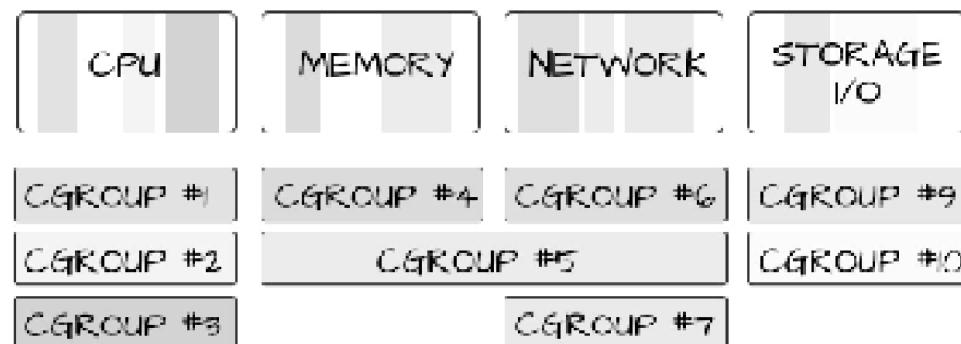
- PID
- NET
- IPC
- MNT
- UTS



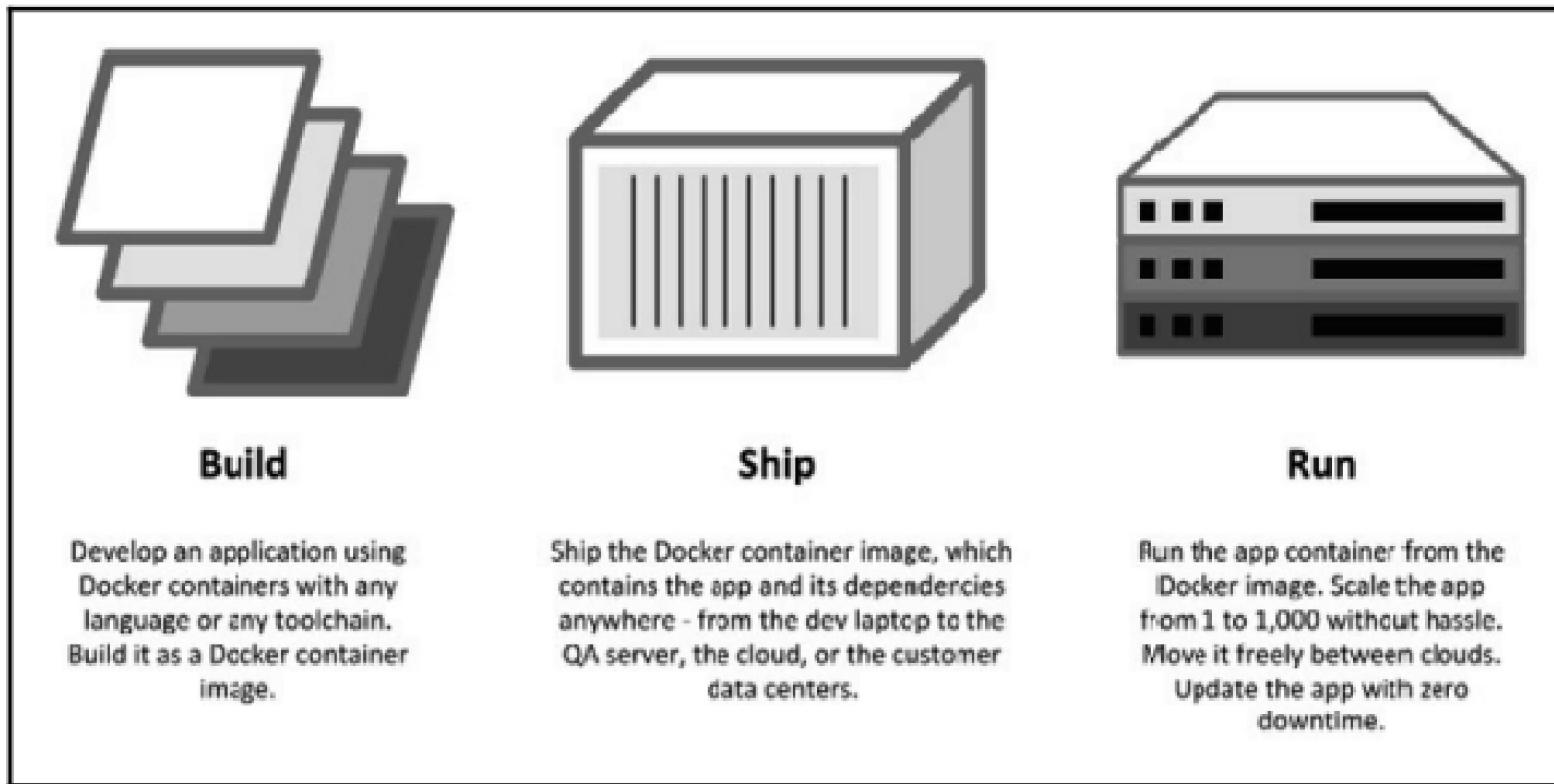
Izolarea resurselor - detalii

Cgroups - Izolare și enumerare resurse sistem

- cpu
- memory
- block i/o
- devices
- network
- numa
- freezer

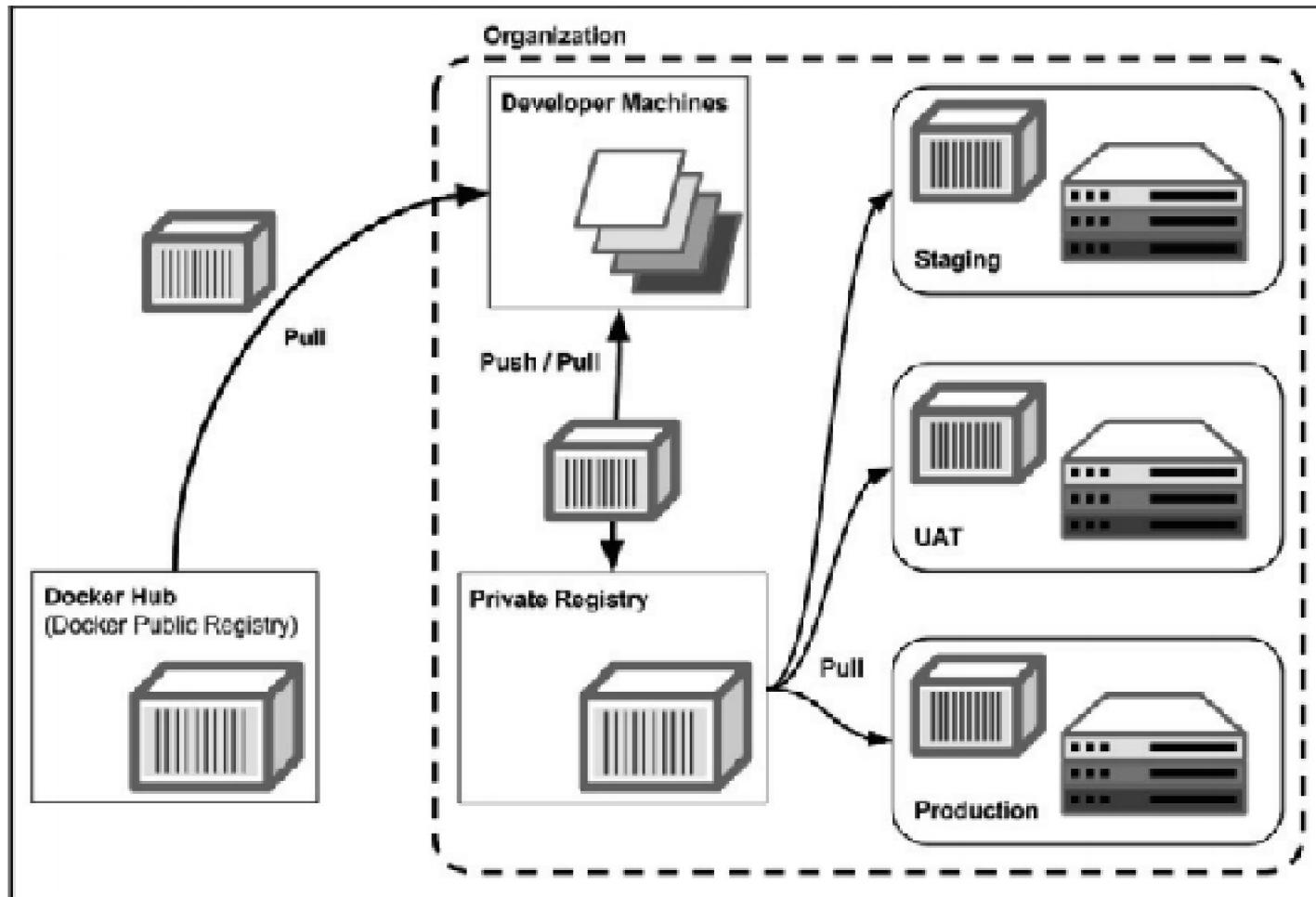


Filozofia Docker



Build-Ship-Run

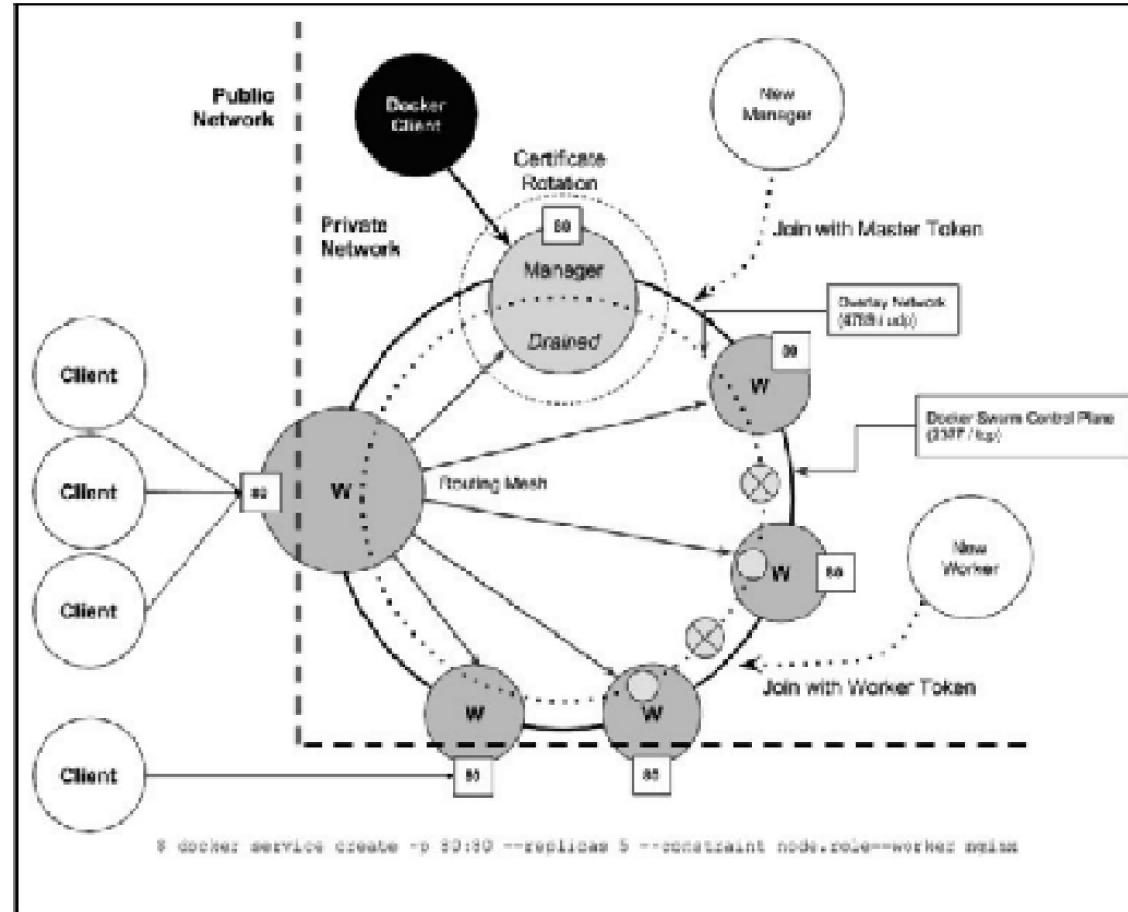
Gestiunea imaginilor Docker



procesul de push/pull

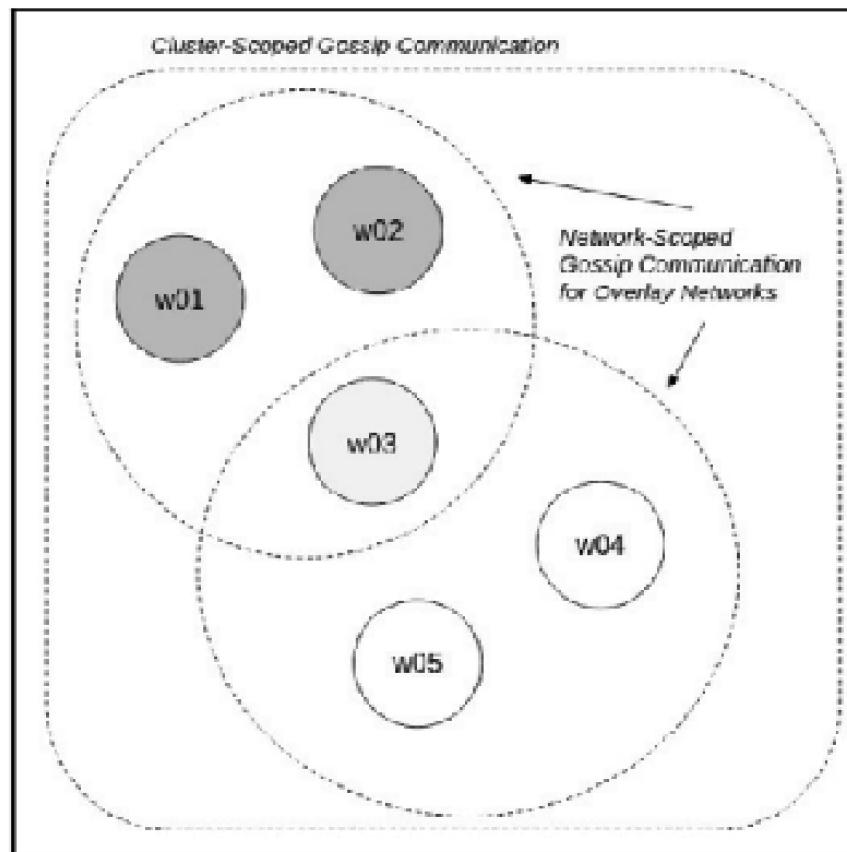
Servicii și sarcini

- --replicas
- --name
- *ingress*
- docker node update --availability drain mg0



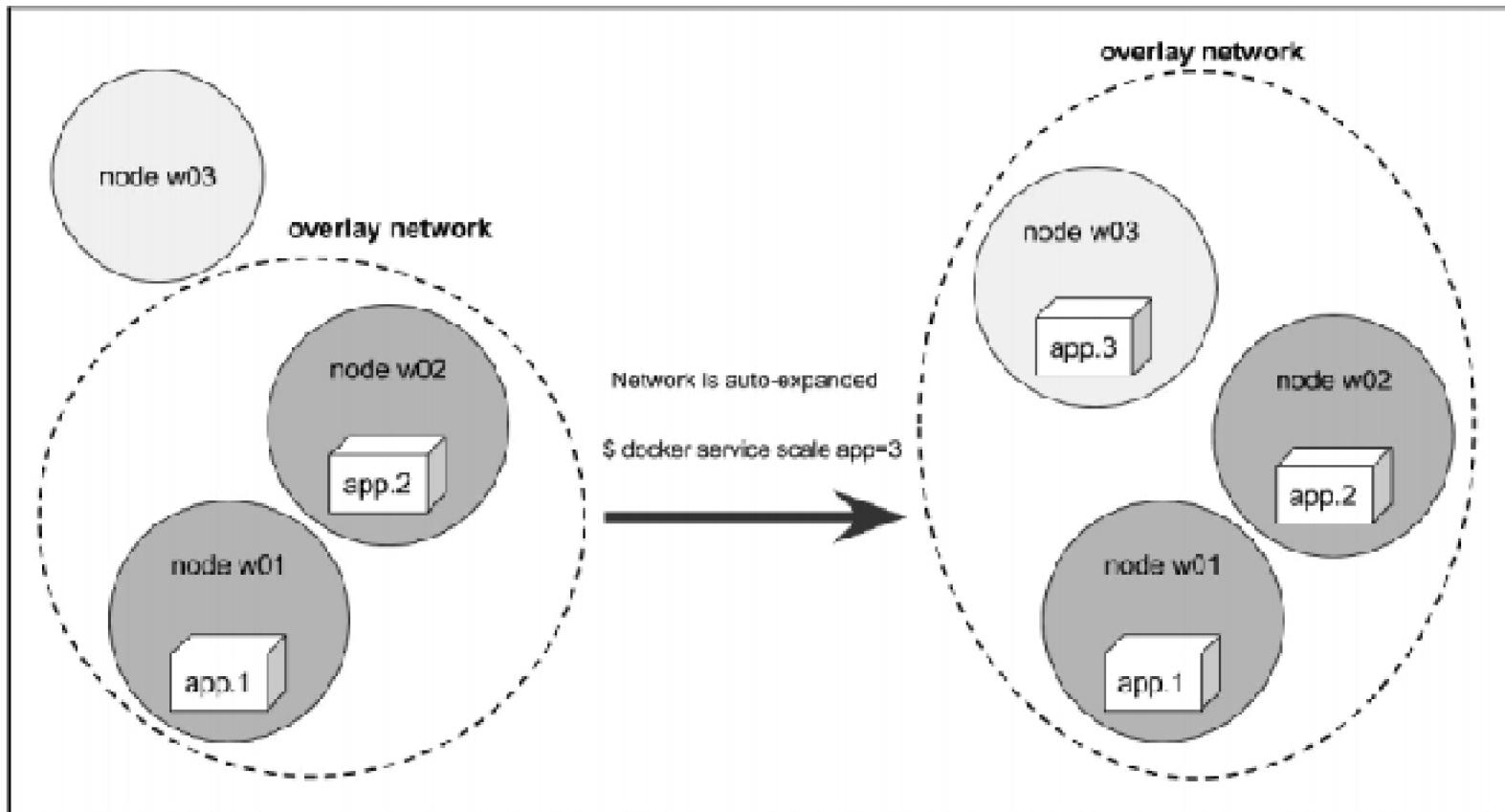
Bârfa în Docker (nu numai la Români)

- docker inspect web



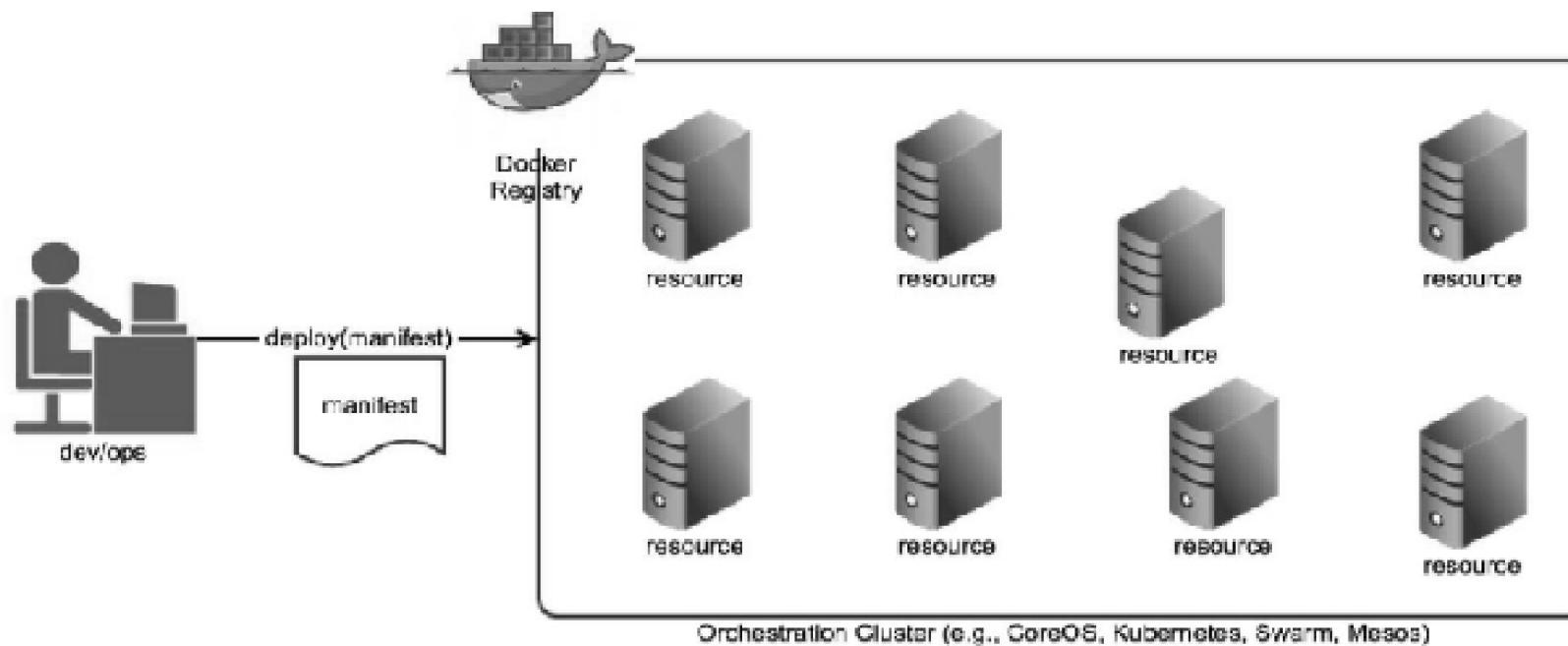
mecanismul de bârfă (gossip) utilizat
pentru comunicarea inter-rețea în roi (swarm)

Multiplicarea/scalarea unui serviciu



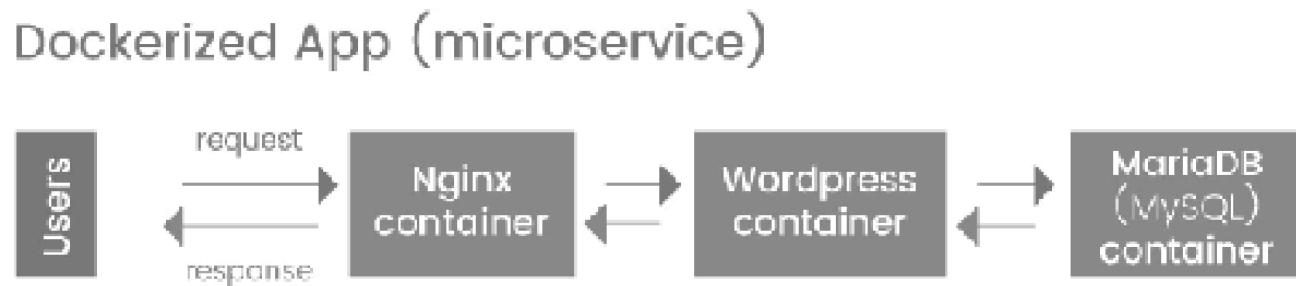
replicare în swarm
docker service scale

Lansarea unui container Docker



pe baza instrucțiunilor din manifest

Cum se pot containeriza microservicii



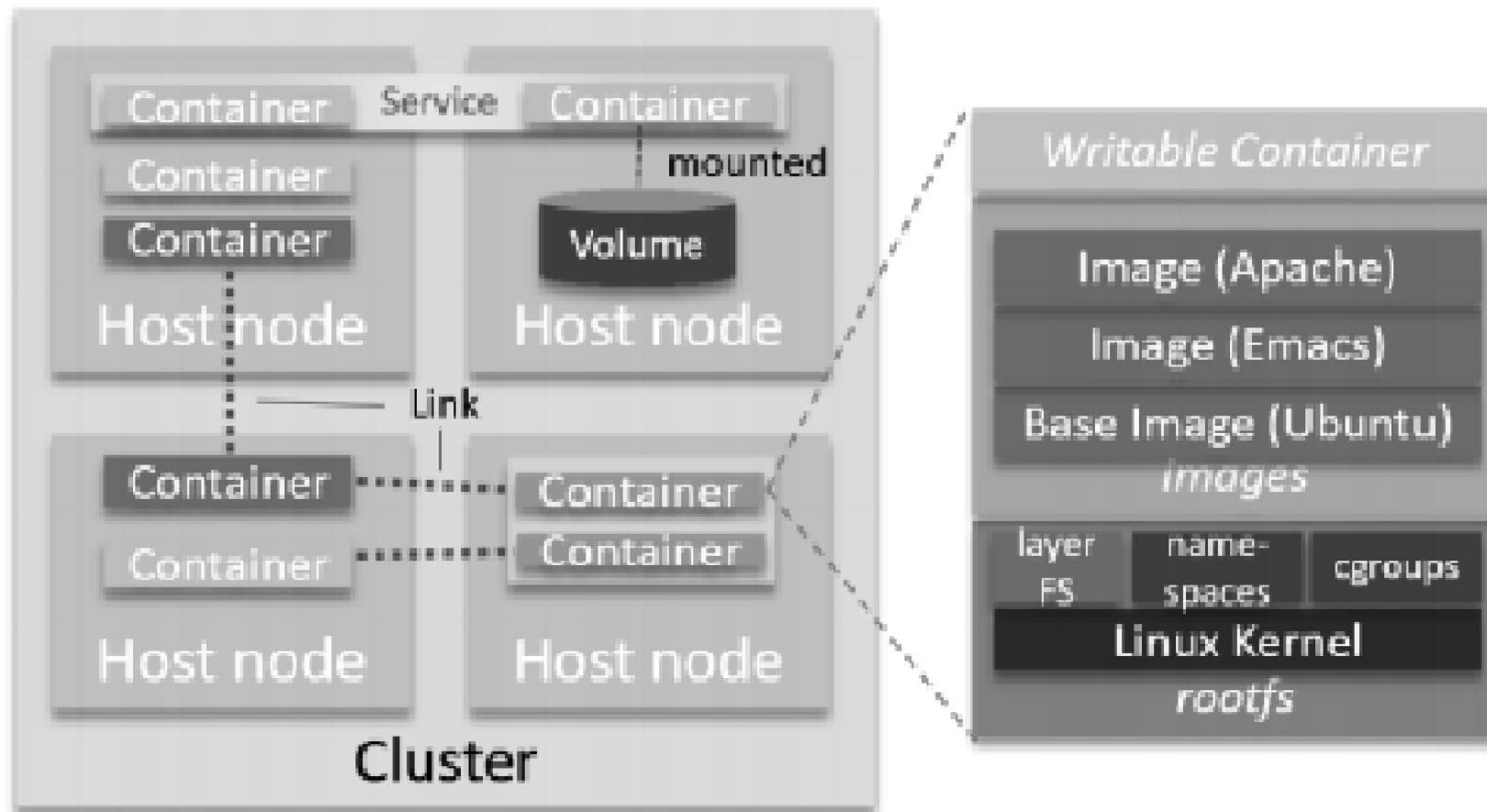
docker-compose.yml

- version: '2'
- services:
- nginx:
 - build: nginx
 - restart: always
 - ports:
 - - 8080:80
 - volumes_from:
 - - wordpress
- wordpress:
 - image: wordpress:php7.1-fpm-alpine
 - environment:
 - WORDPRESS_DB_HOST: mysql
 - WORDPRESS_DB_PASSWORD: example
 - mysql:
 - image: mariadb
 - environment:
 - MYSQL_ROOT_PASSWORD: example
 - volumes:
 - - ./demo-db:/var/lib/mysql

deci în esență cum fac o aplicație

1. creez un microserviciu intr-un limbaj/tehnologie
2. il testez
3. creez o imagine de container
4. creez o componzie de servicii conform proiectarii anterioare (model arhitectural + combinatii de modele de proiectare specifice)

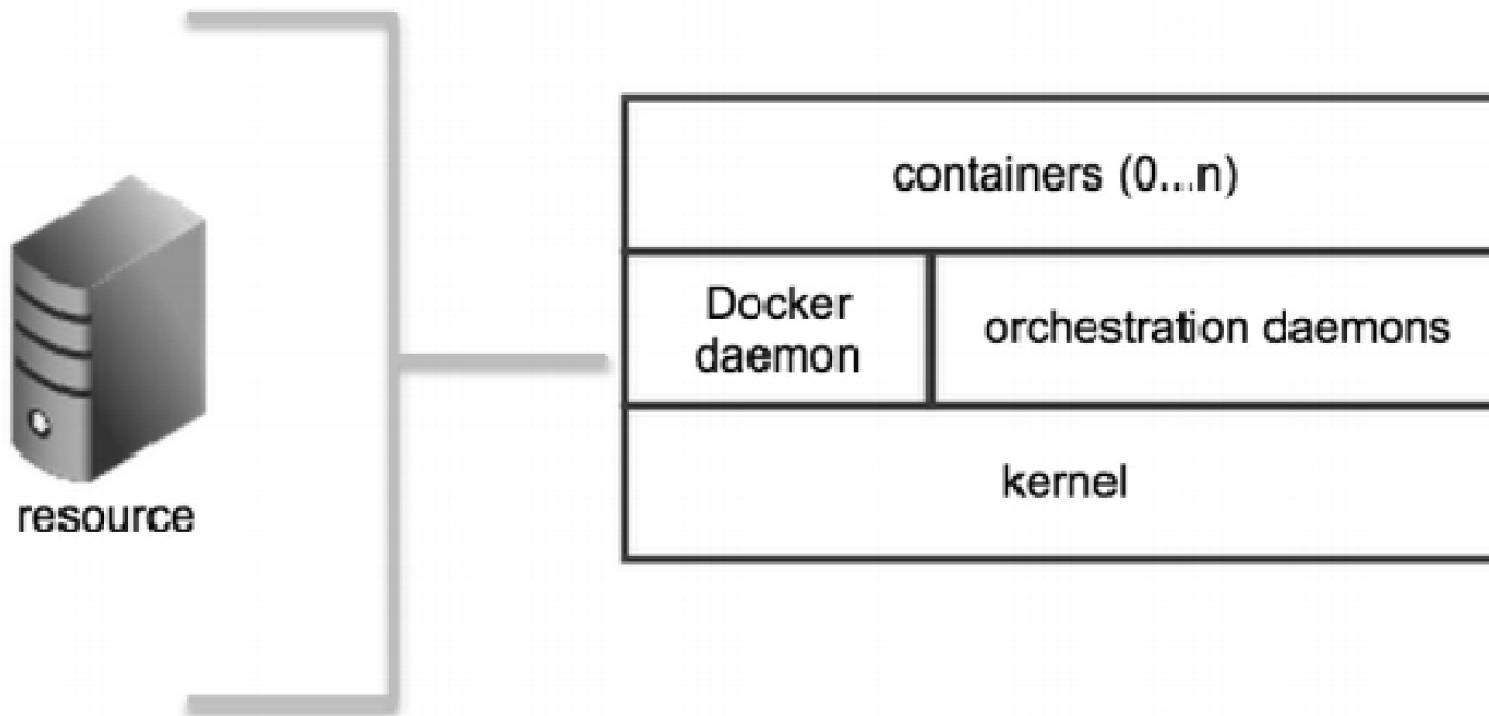
Cluster de containere



Containere în nor

- gestiune dependențe
- agenți pe ciclul de afaceri
- PaaS & containere

Cum intervine orchestrarea?

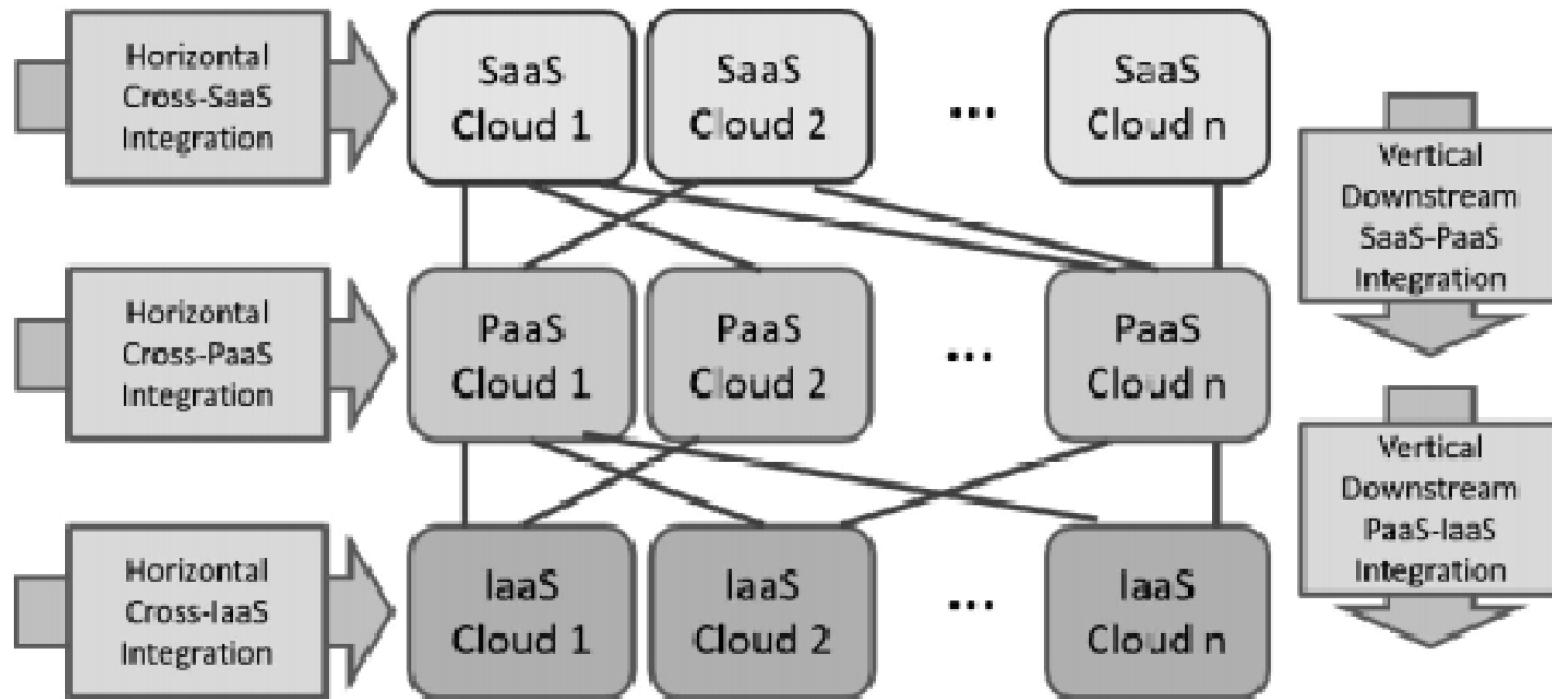


- X

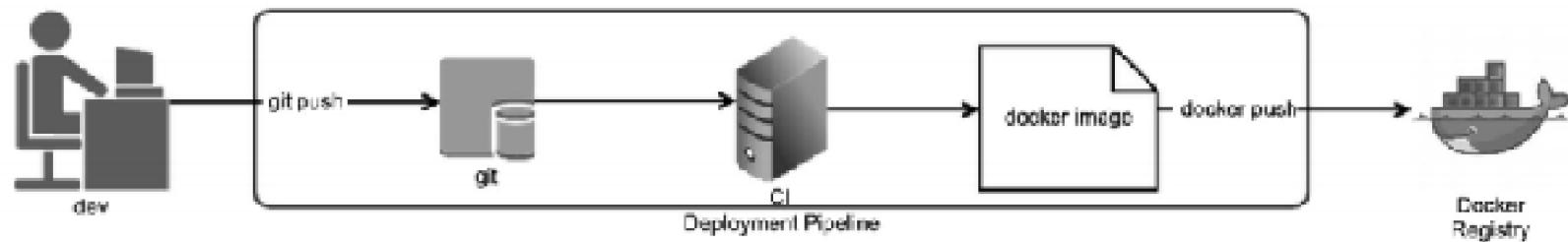
Orchestrarea

- asigurarea resurse
- instanțele
- replanificări
- relația cu interfața
- expunere servicii
- Mesos, Kubernetes, CorCos Tectonic și Docker Swarm

Arhitectura containerelor în nor



Dezvoltare continuă în Docker



Linie de productie (pipeline deployment) bazată pe containere

Utilizarea Docker Compose

- definesc mediul
- definesc serviciile
- reguli suplimentare
- se lanseaza în execuție

```
docker-compose.yml
1. version: '3'
2. services:
3.   web:
4.     build: .
5.     ports:
6.       - "5000:5000"
7.     volumes:
8.       - ./code
9.       - logvolume01:/var/log
10.    links:
11.      - redis
12.    redis:
13.      image: redis
14.    volumes:
```

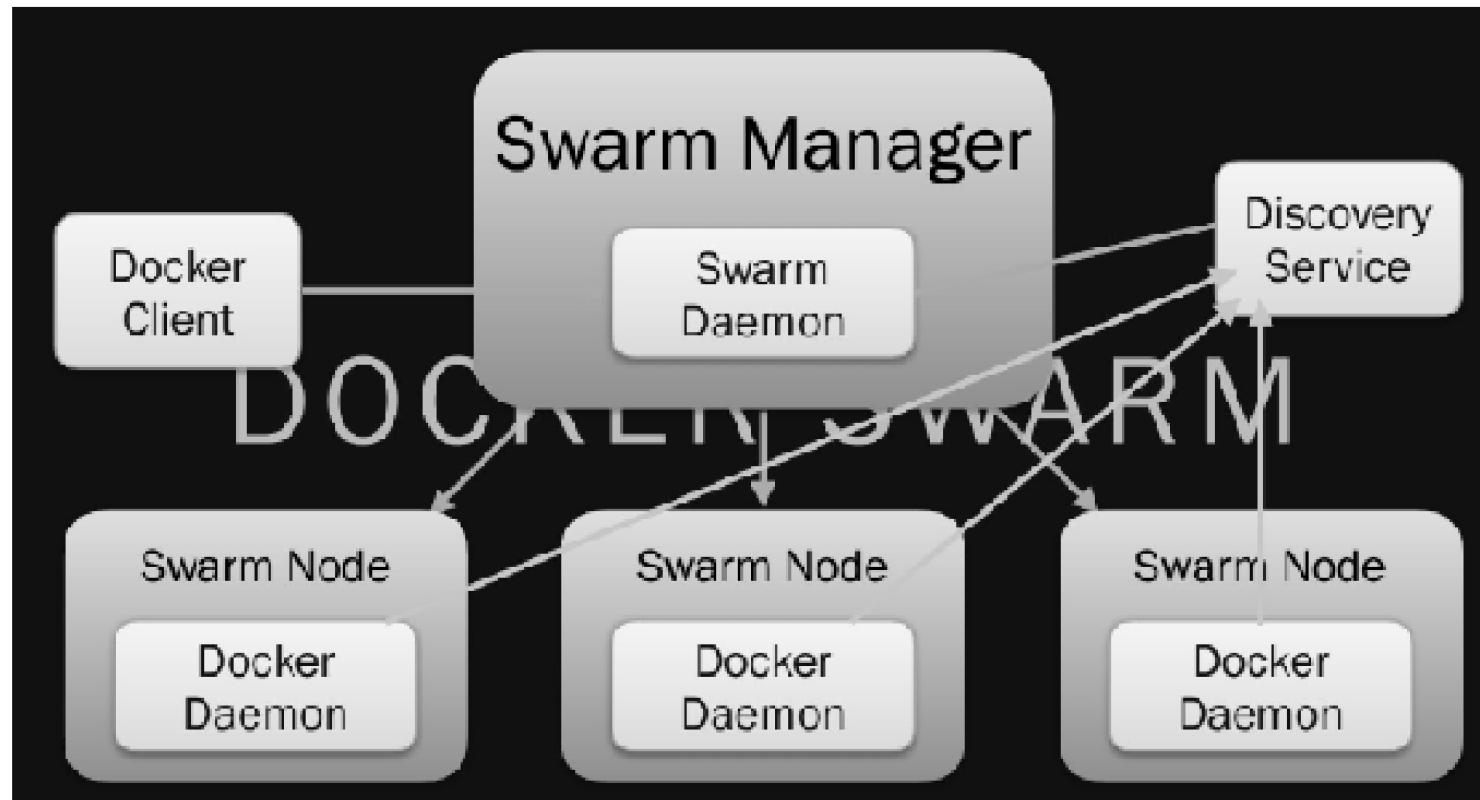
Docker Compose

- Permite multiple medii de lucru izolate pe o singura masina gazda
- Pastreaza volumele de date in momentul crearii containerelor
- Recreaza numai containerele care s-au modificat
- Variabile si posibilitatea schimbarii componitiilor dintr-un mediu intr-altul

Cand se foloseste Compose?

- dezvoltare
- testare automată
 - \$ docker-compose up -d
 - \$./run_tests
 - \$ docker-compose down
- gazdă unică

Orchestrarea containerelor



Docker Swarm

Facilități oferite de Docker Swarm

- Management
- Scalarea
- Rețea între gazde
- Descoperirea serviciilor
- Echilibrarea încărcării - Load balancing
- Nodul din roi

Servicii și sarcini in Docker swarm

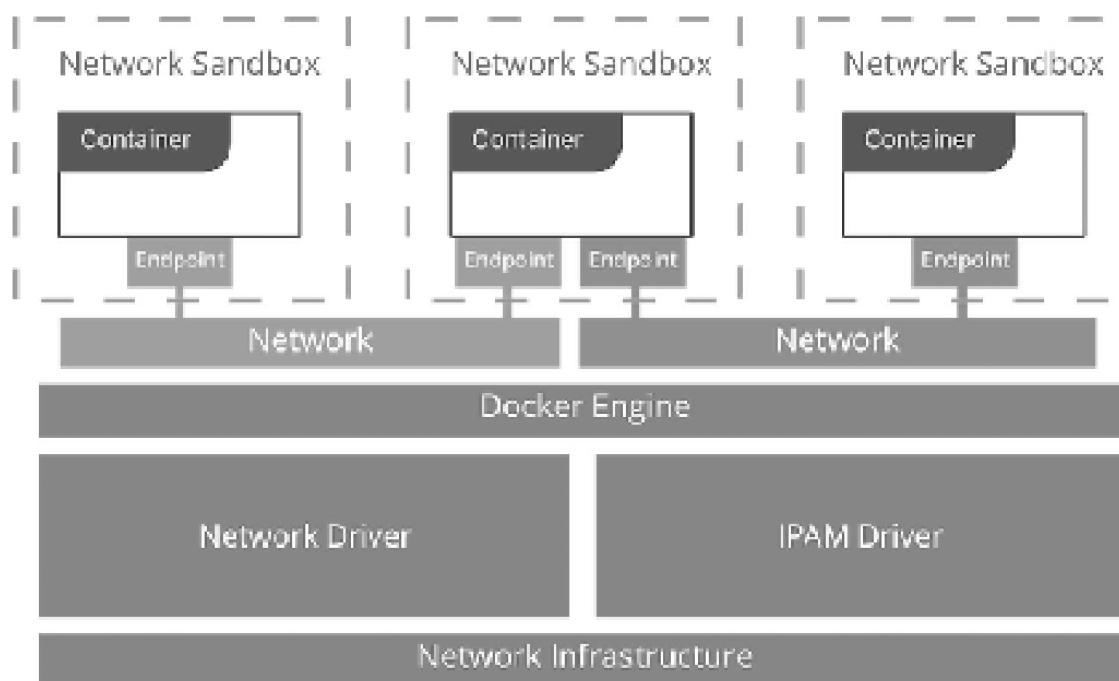
- serviciu
- sarcină/task
- planificare statică

Echilibrarea încărcării în Docker swarm

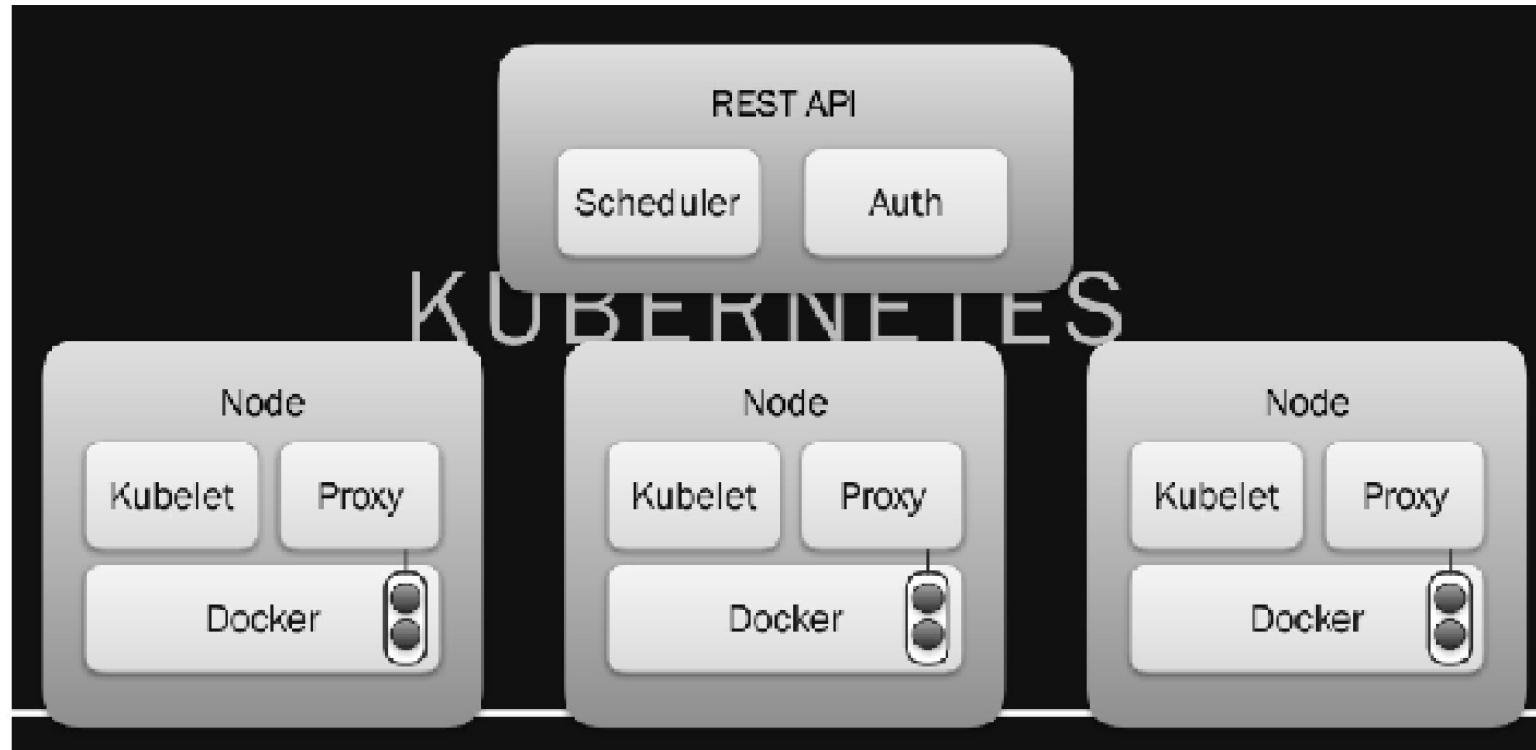
- port
 - manual
 - automat 30000-32767
- DNS local

Filtrele Docker swarm

- Contrangere sau marcaj nod
- Afinitate
- Portul
- Dependență
- Sănătatea

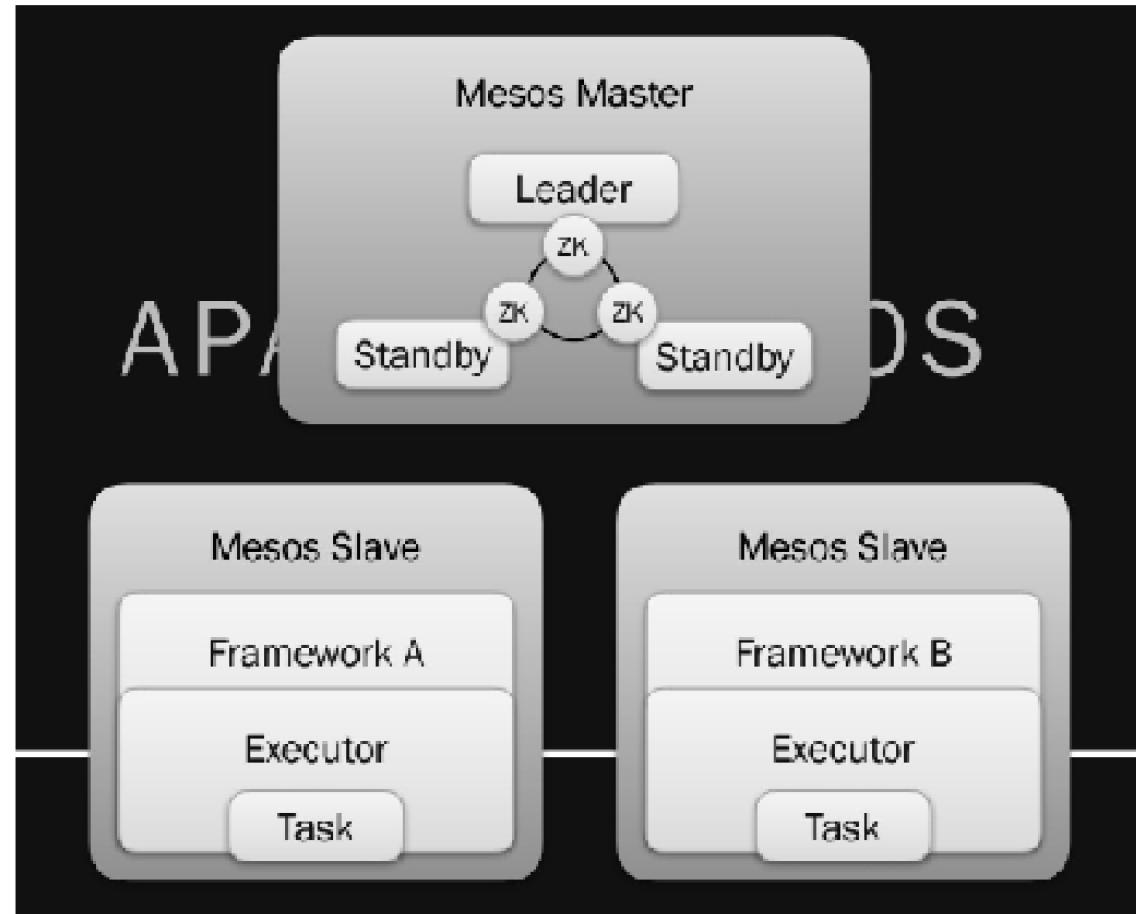


Alte soluții pentru orchestrarea containerelor



Kubernetes

Alte soluții pentru orchestrarea containerelor



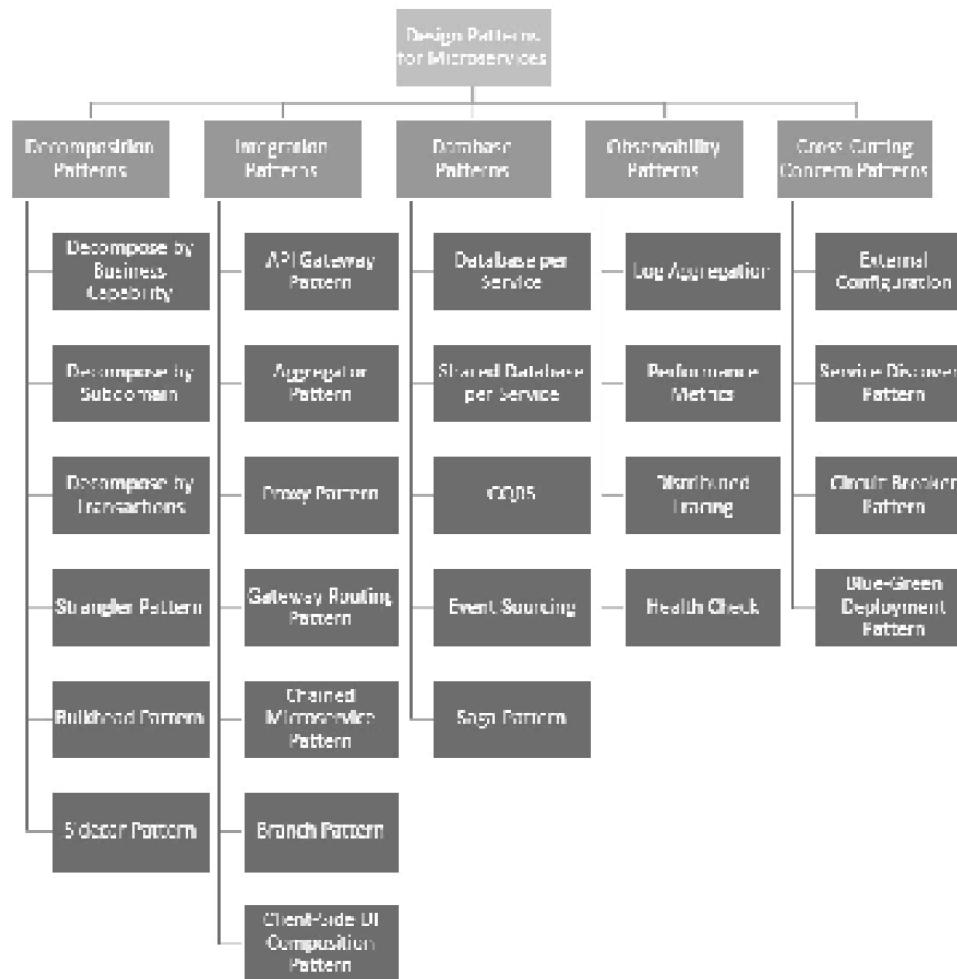
Apache Mesos

Sisteme Distribuite

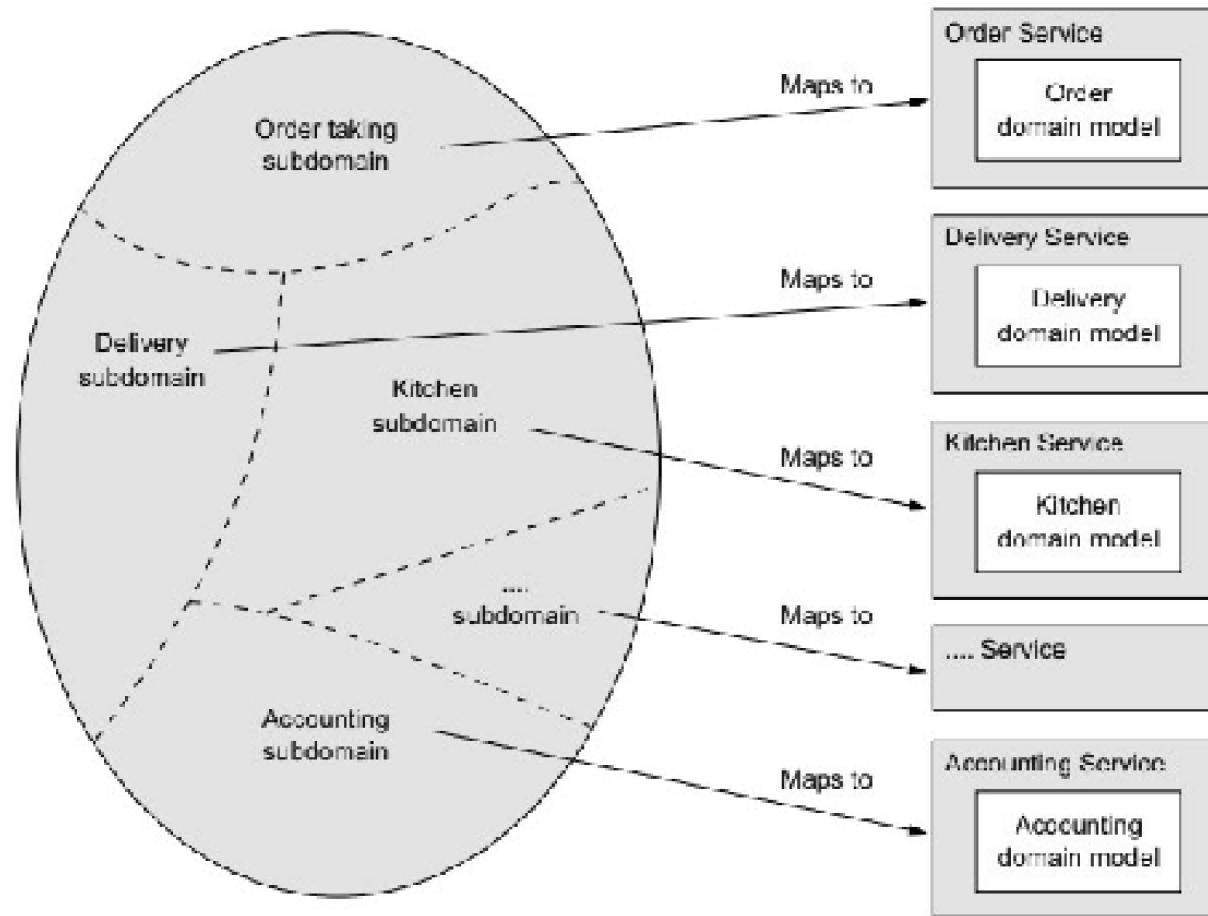
Cursul 9

Mihai Zaharia

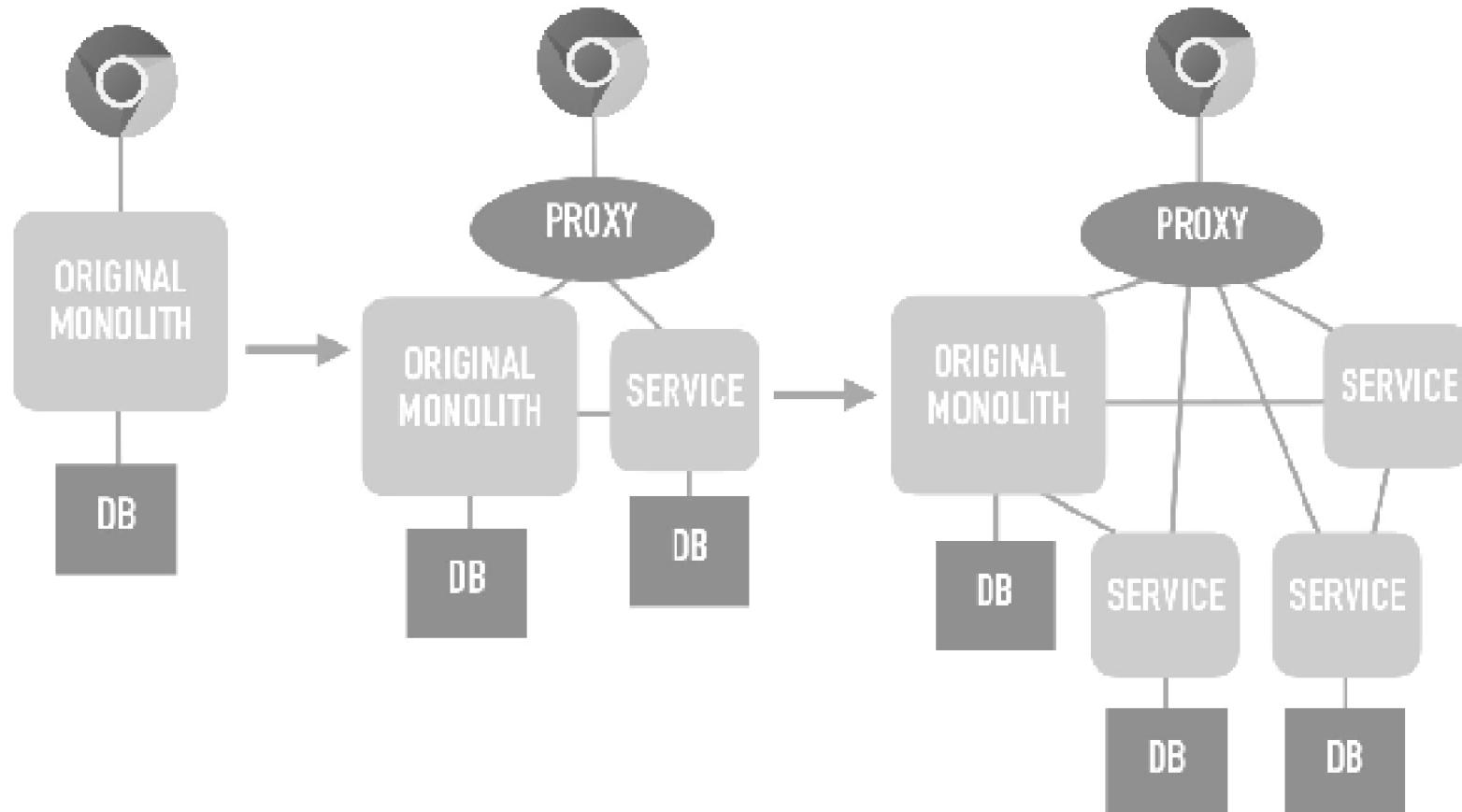
Modele de proiectare pentru MSA



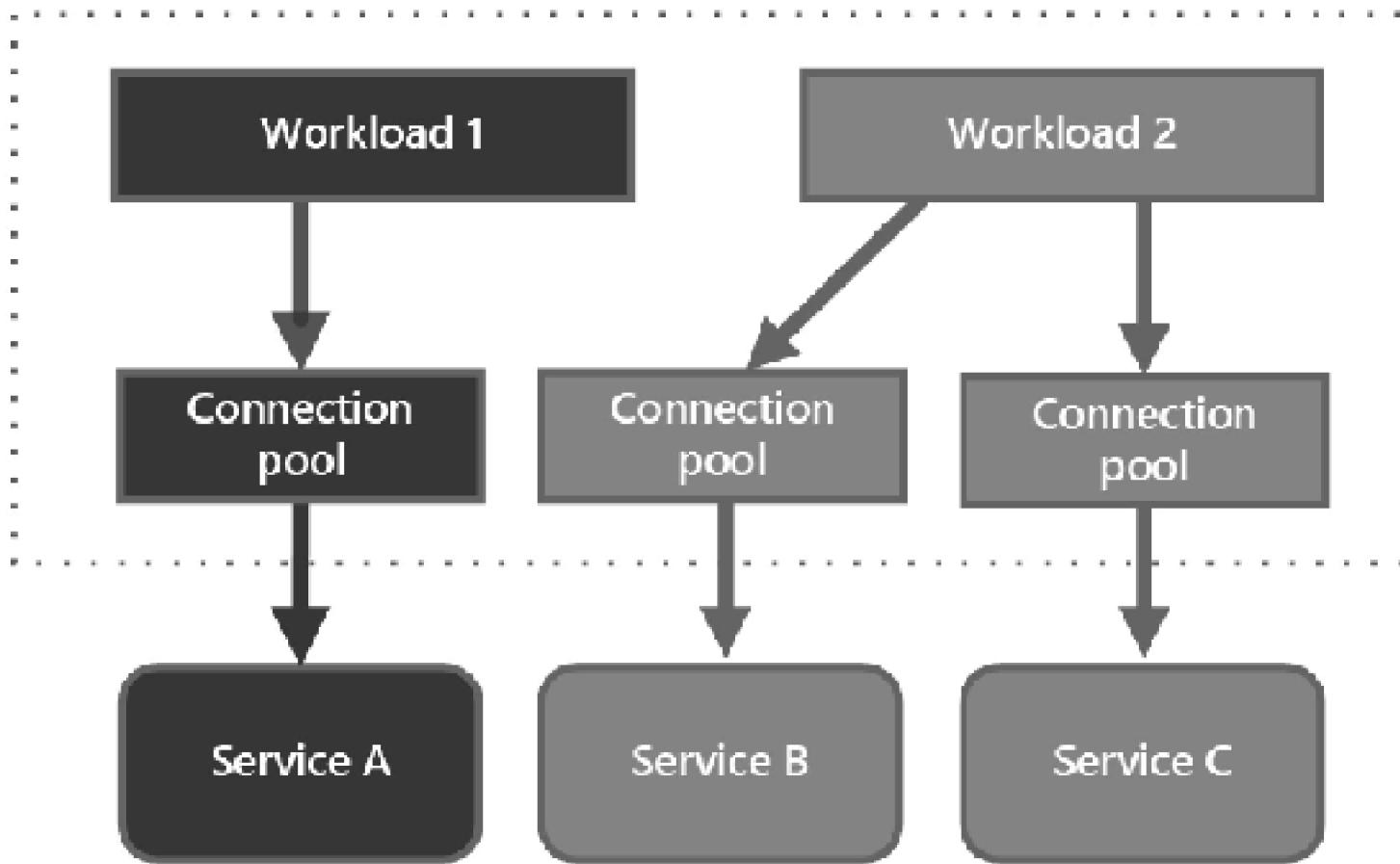
Descompunerea pe subdomenii



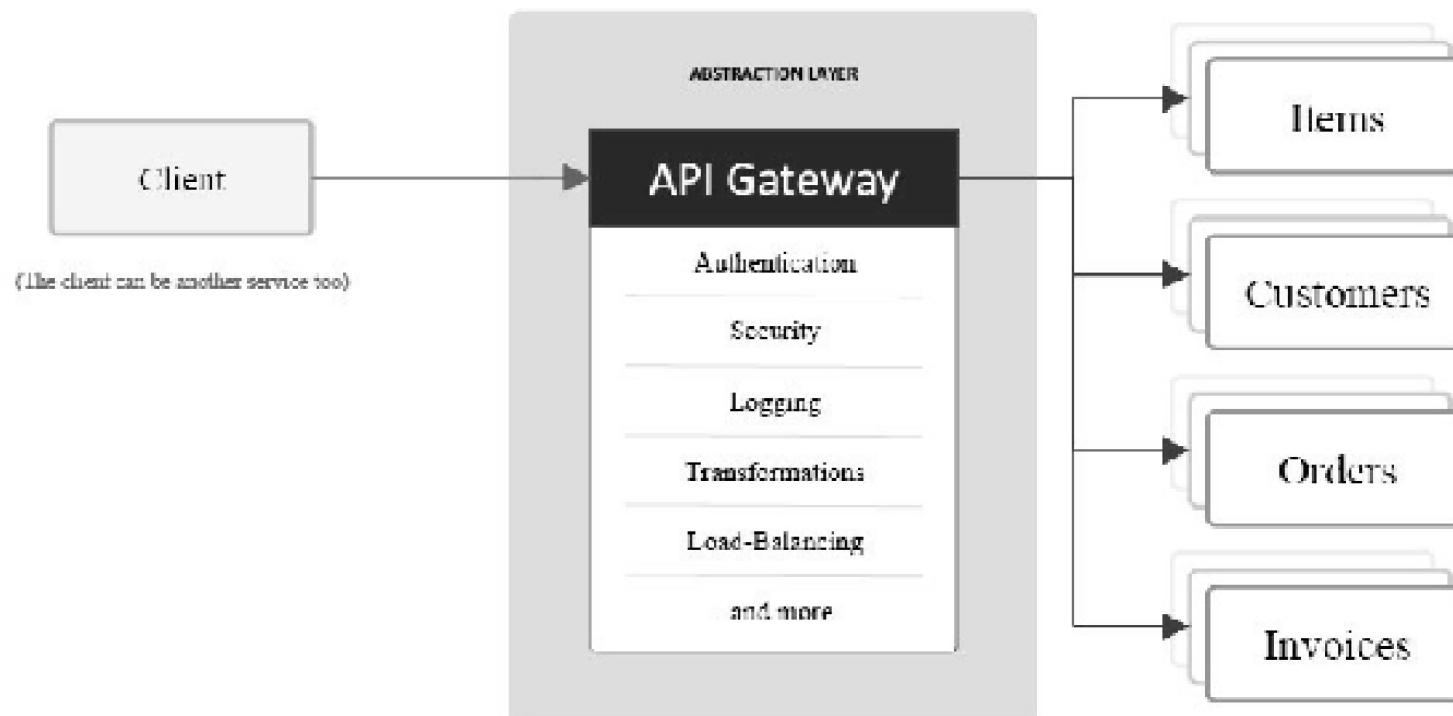
Modelul sugrumatătorului - Strangler Pattern



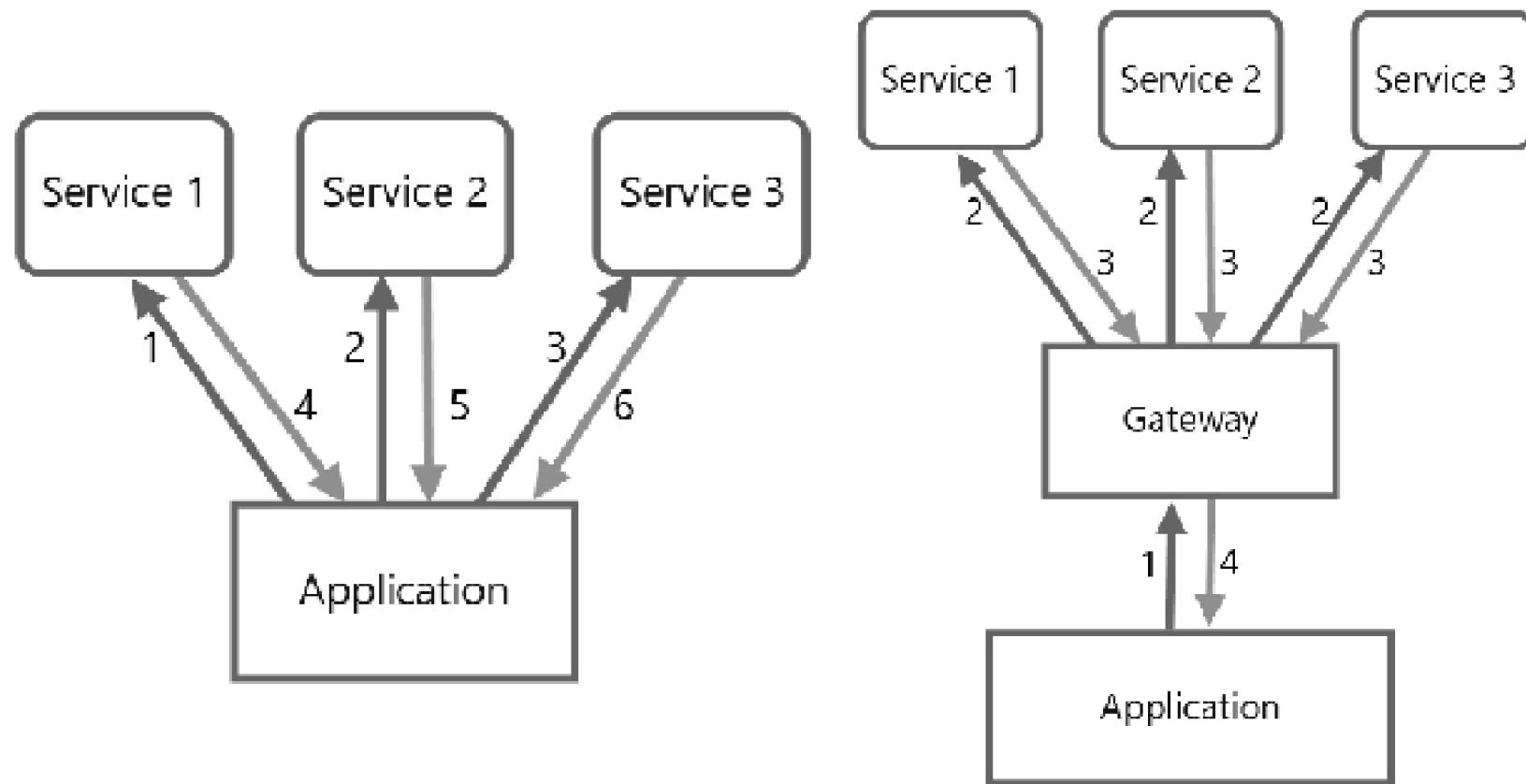
modelul compartimentat - Bulkhead



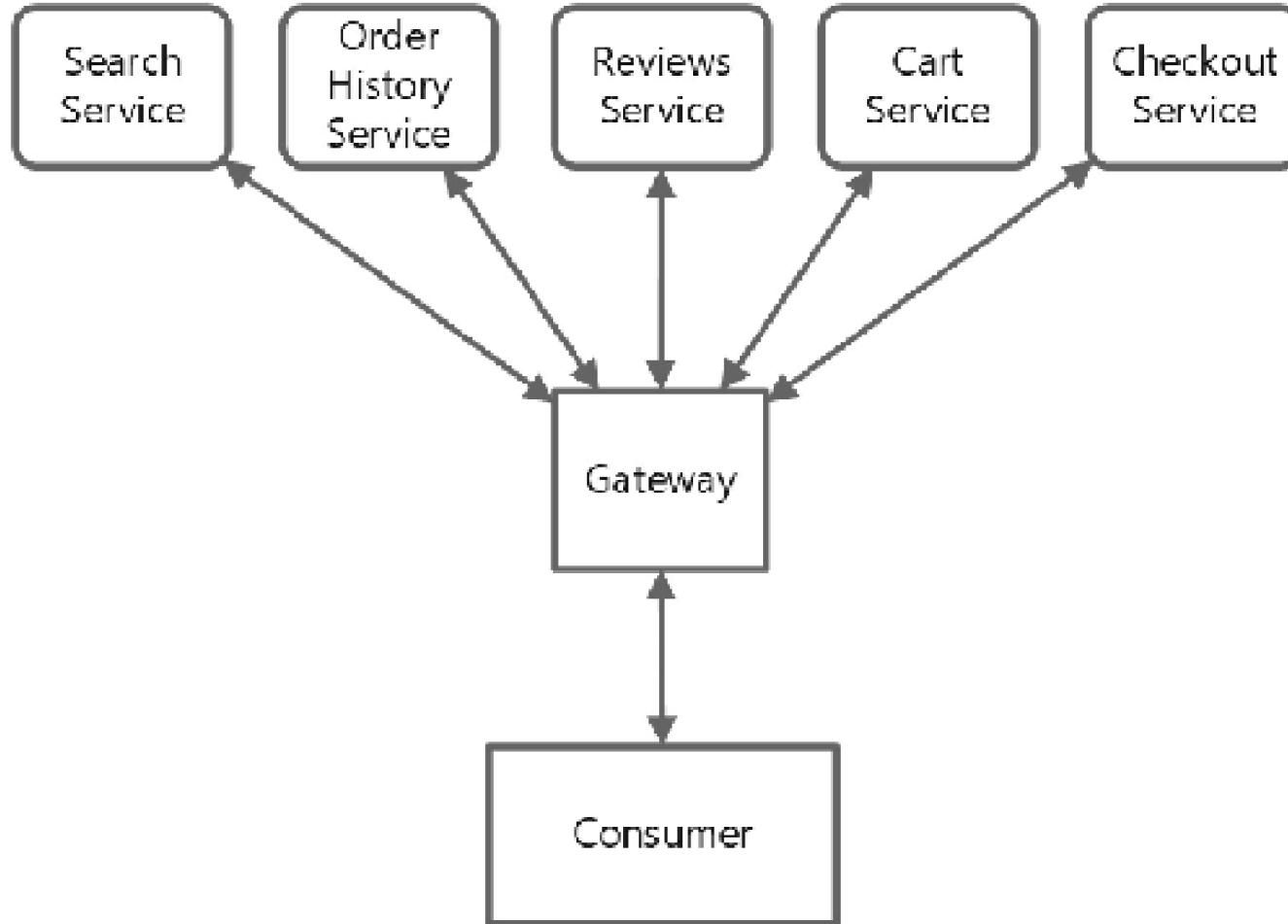
modelul portii API



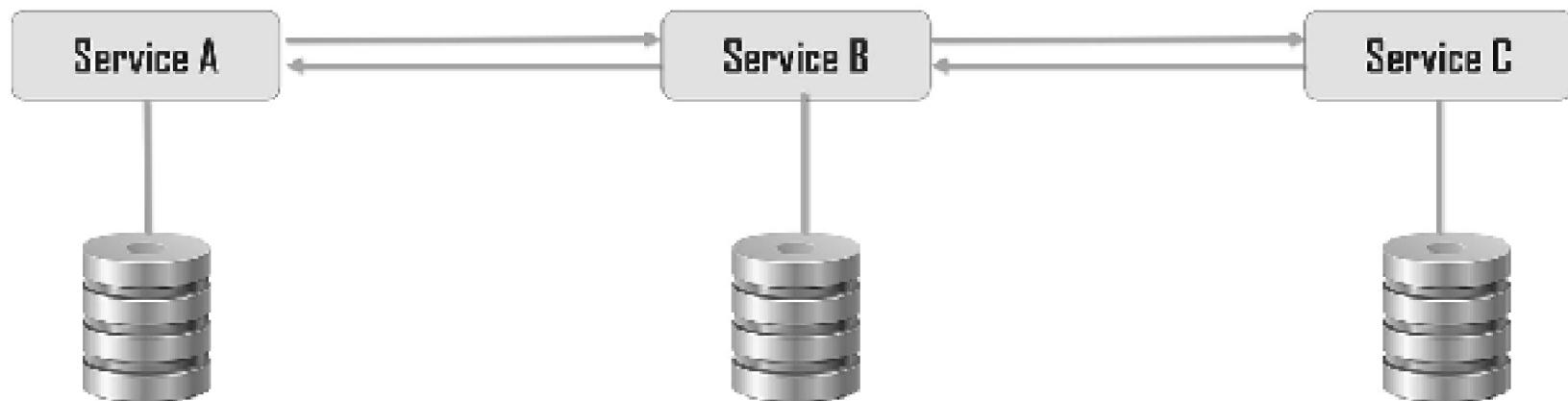
Modelul compunere (Aggregator)



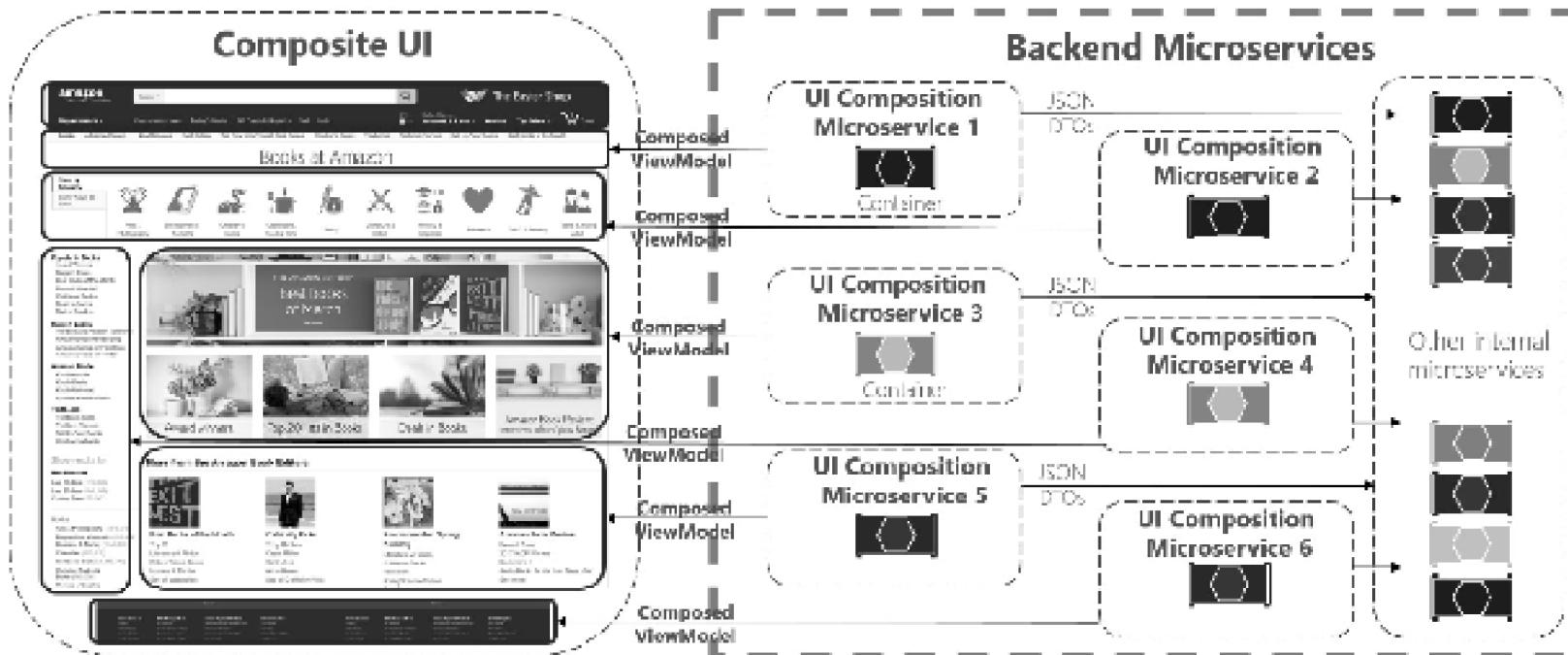
modelul cu poartă pentru dirijare



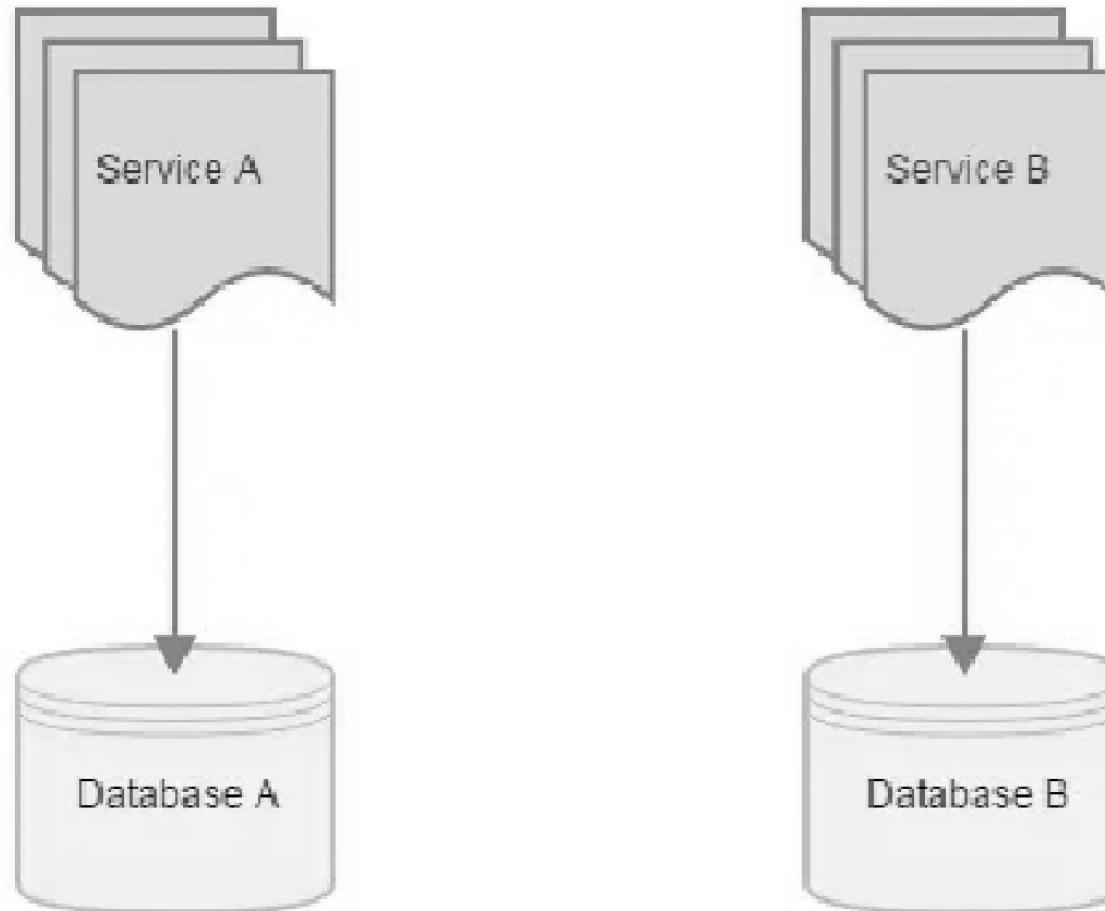
microservicii înlăntuite



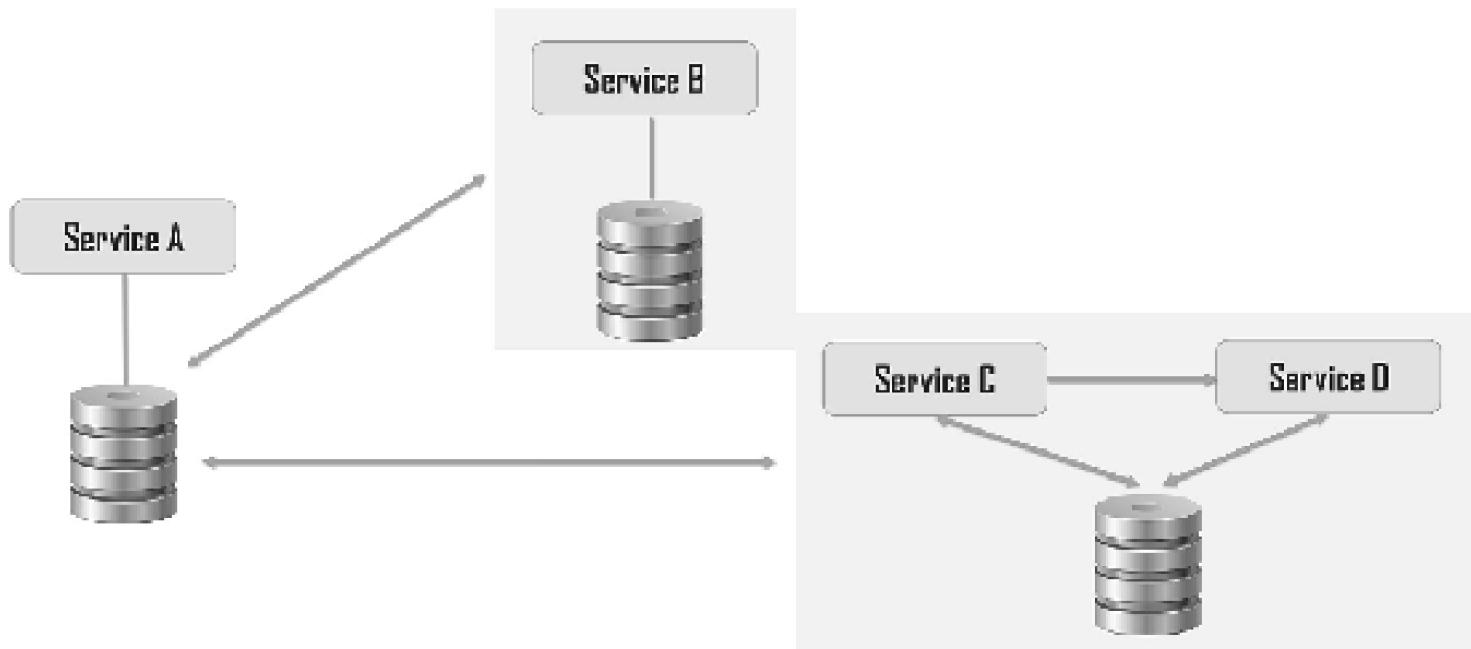
Modelul de proiectare pentru compunerea interfaței utilizator pentru client



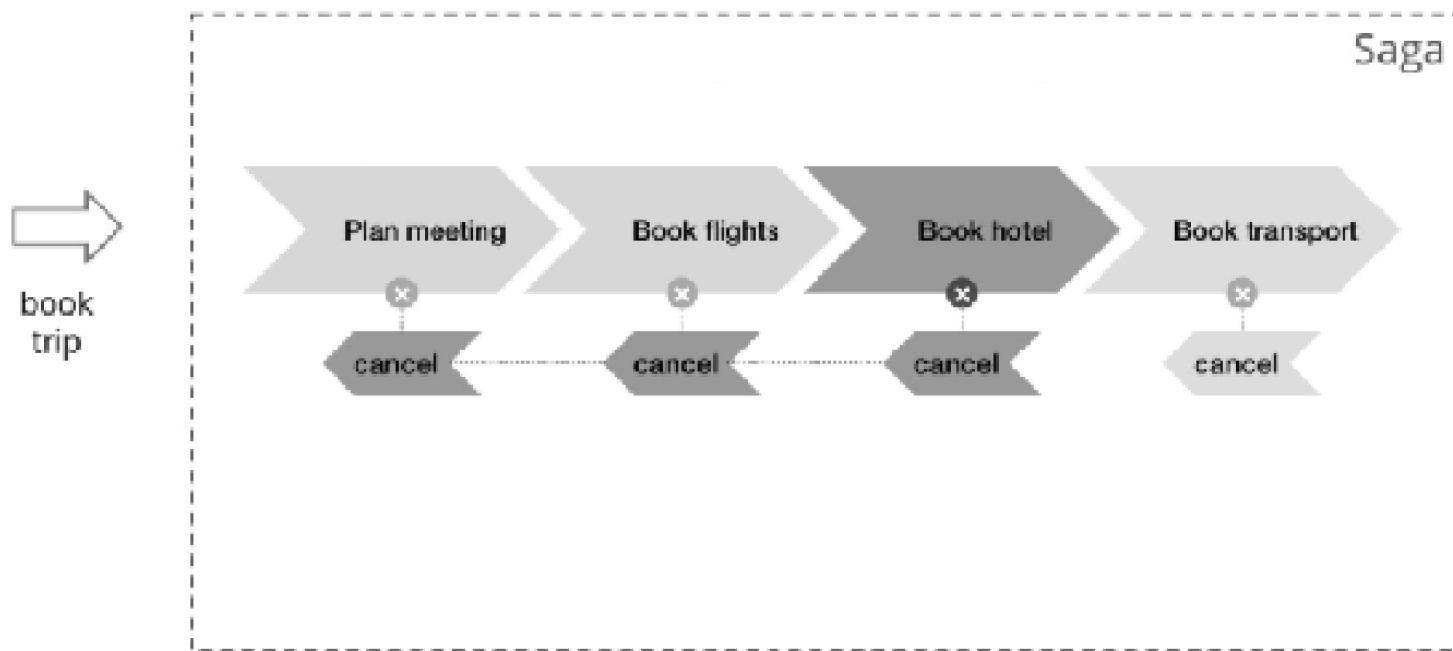
Baza de date pentru un serviciu



Bază de date comună pentru un microserviciu



Model de proiectare SAGA





Sisteme Distribuite

Cursul 10

Mihai Zaharia

Evoluția arhitecturilor software distribuite

- Obiectele distribuite (RPC sync),
- ESB-ul SOA,
- arhitecturi orientate pe eveniment - EDA,
- programarea reactivă
- microserviciile
- arhitecturi evoluționare
- serverless computing
- Faas

Eveniment

- **TOPICĂ**



"Event Cloud"

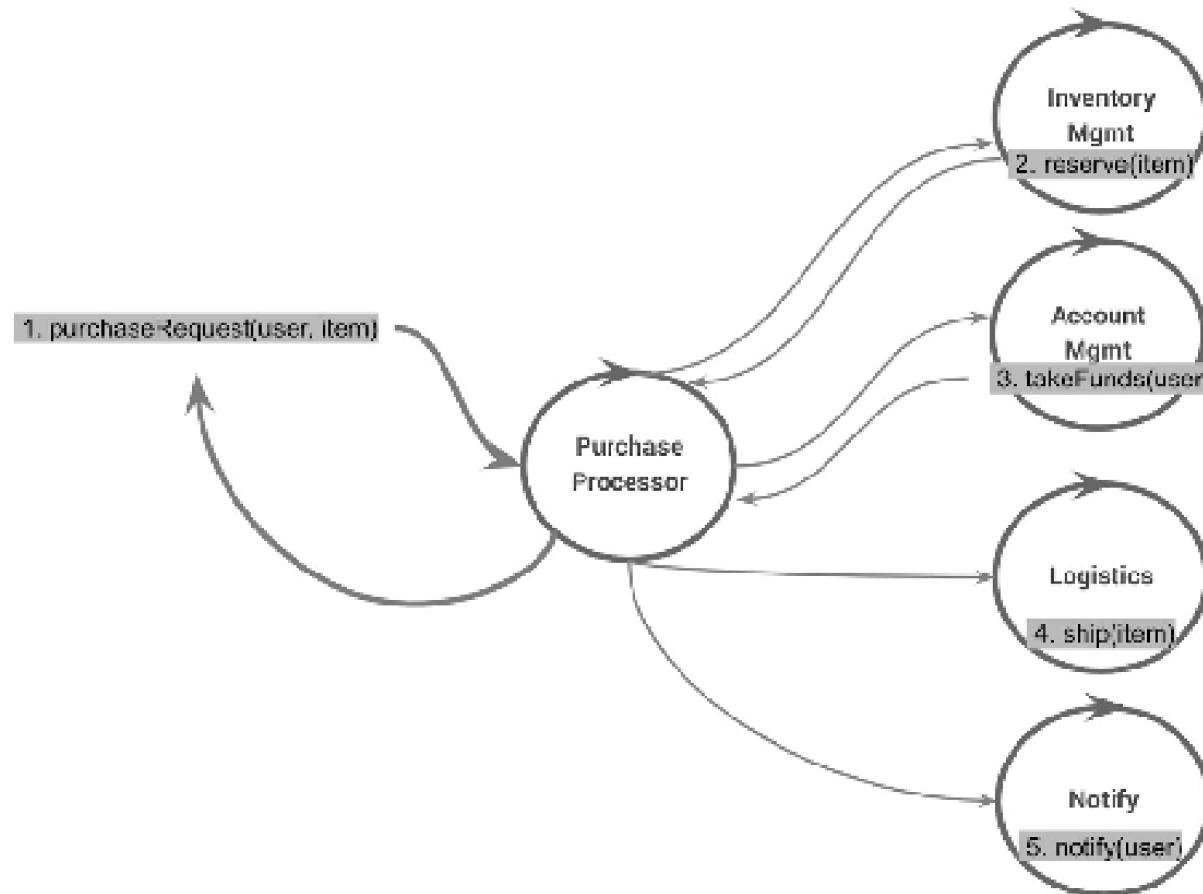
Eveniment și comportament

- Eveniment:
 - este atomic
 - înrudire
 - comportament
- Partea de comportament este cea mai importantă
 - În proiectare
 - În utilizare

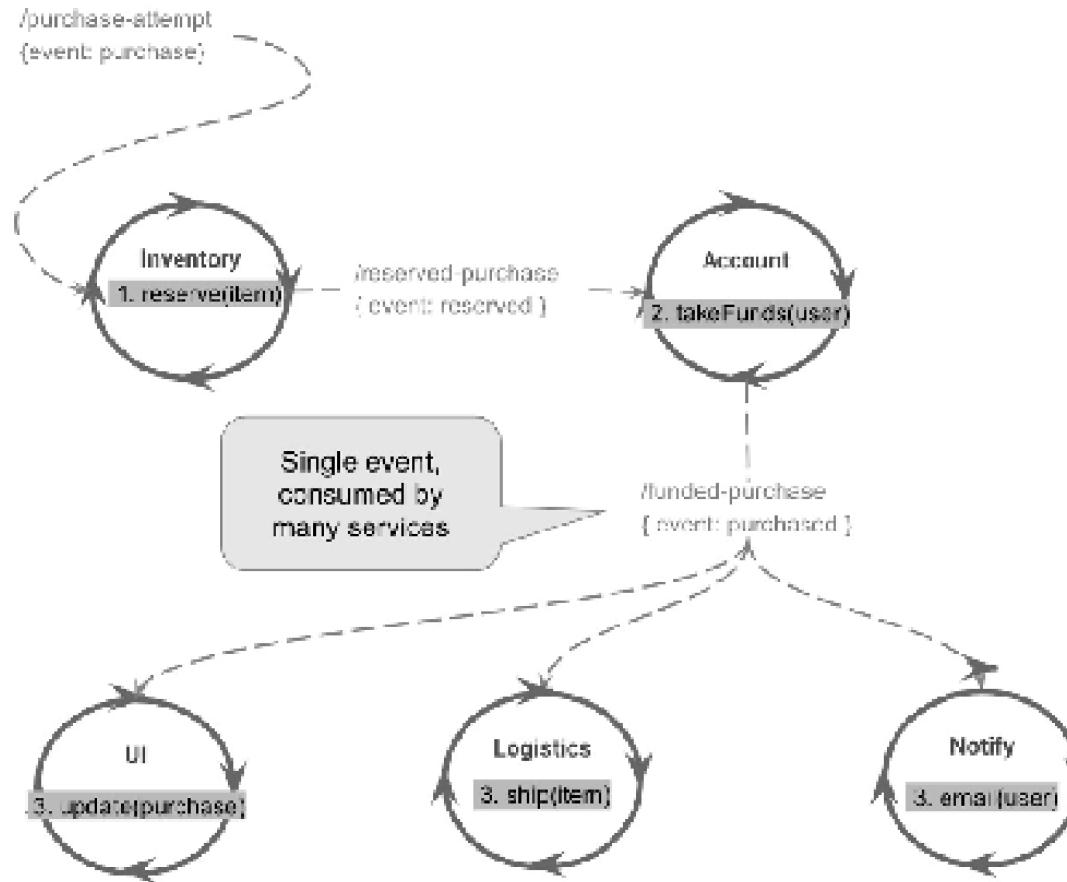
Abordări arhitecturale comune

- Concret
 - Evenimentul primul - analog
 - Eveniment comandă
- REST - clasic
- REST - modern - cârpăceală
- Pentru “evenimentul primul” trebuie schimbată complet gândirea de la REST

Caz de utilizare evenimentul comandă



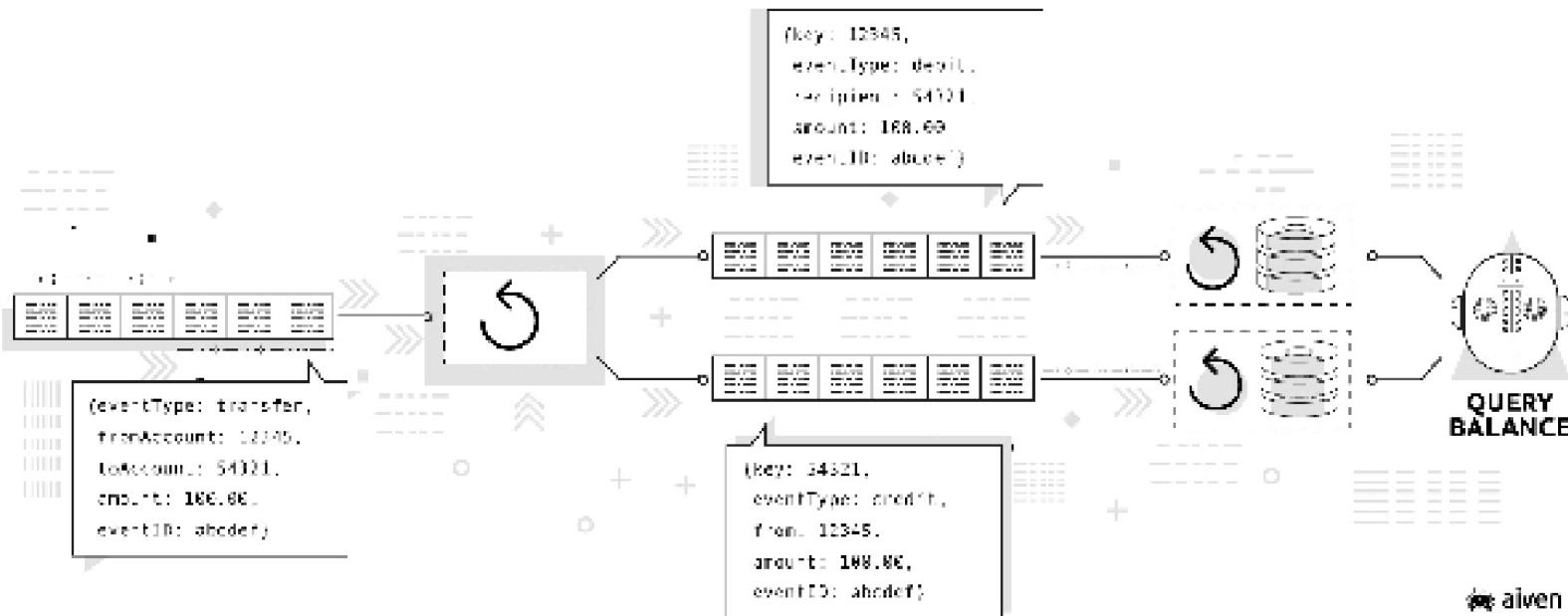
Caz de utilizare evenimentul primul



Kafka

- Istoric
- Ce este Kafka?
- Ce se poate face cu el?
 - un message broker
 - un sistem distribuit de jurnalizare
 - o componentă dintr-o arhitectură bazată pe stream-uri

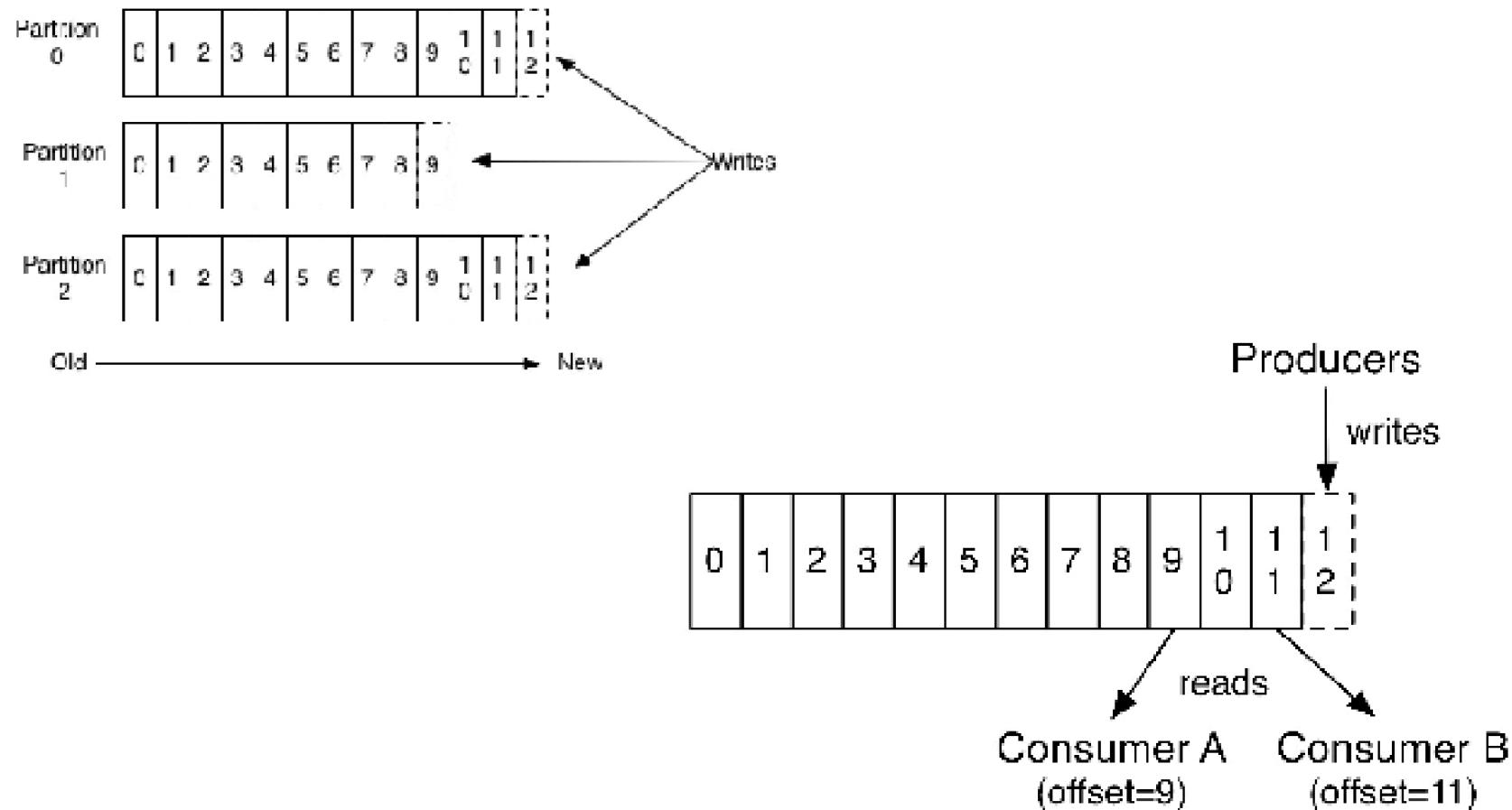
Kafka - Cum Ticăie?



- X

Kafka - Cum Ticăie?

Anatomy of a Topic



- X

Arhitectură orientată eveniment



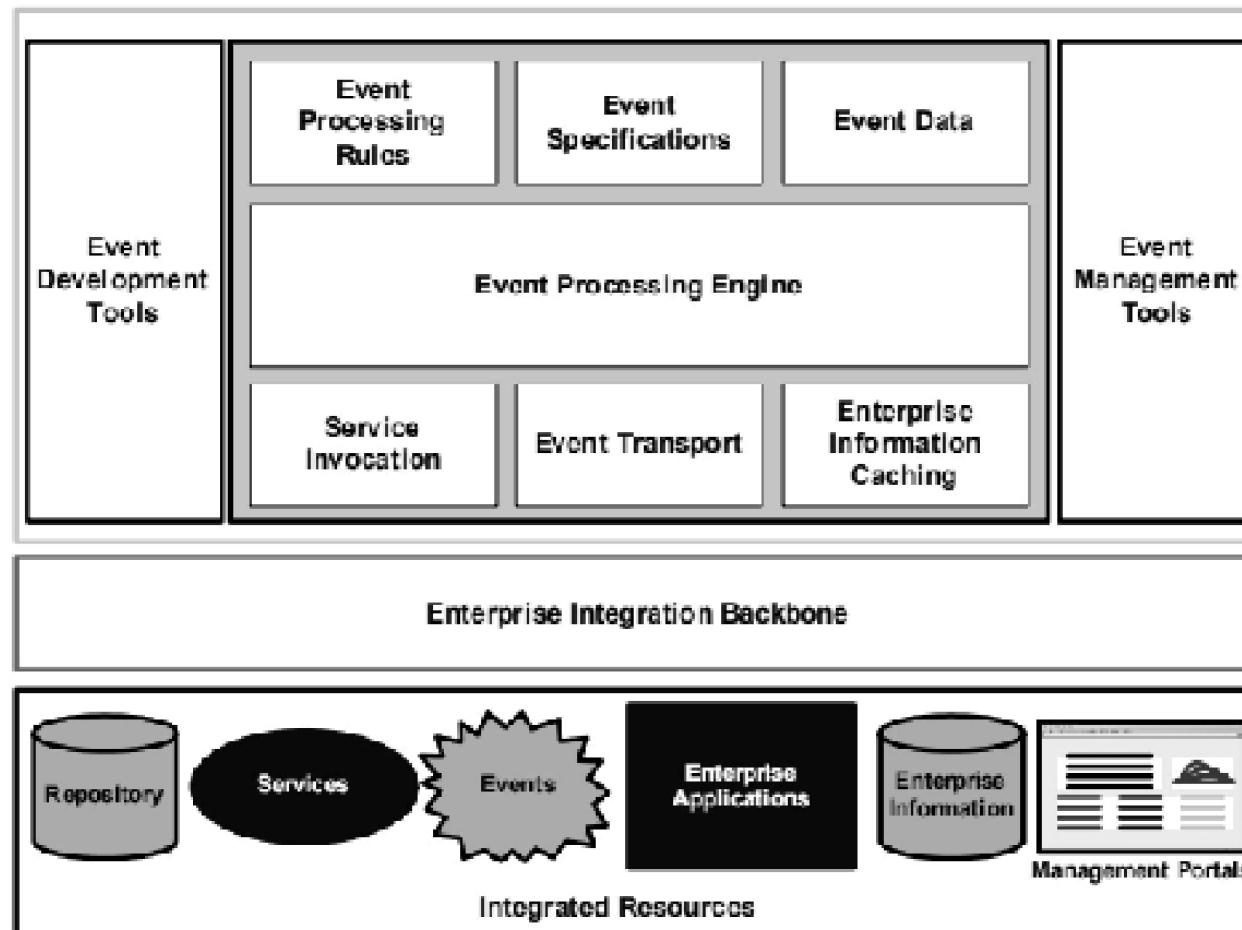
EDA reactiv

- **Reactive Manifesto 2014**
- *"Sistemele construite ca fiind de tip reactiv sunt mai flexibile, scalabile si au un grad mai scazut de cuplare. Au o toleranta mult mai buna la erori decat predecesoarele fapt care le face mult mai usor de dezvoltat si ulterior de modificat. Cand o eroare apare aceasta este gestionata si nu se mai produc caderi catastrofale. Mai mult ele sunt flexibile cu viteza mare de reactie/raspuns si furnizeaza utilizatorului un feedback interactiv"*

Caracteristicile EDA

- Comunicare de masa (Broadcast)
- Reactie eficienta
- Evenimente cu granularitate mică
- Ontologie
- Procesarea evenimentelor complexe - CEP

componentele unei implementari EDA



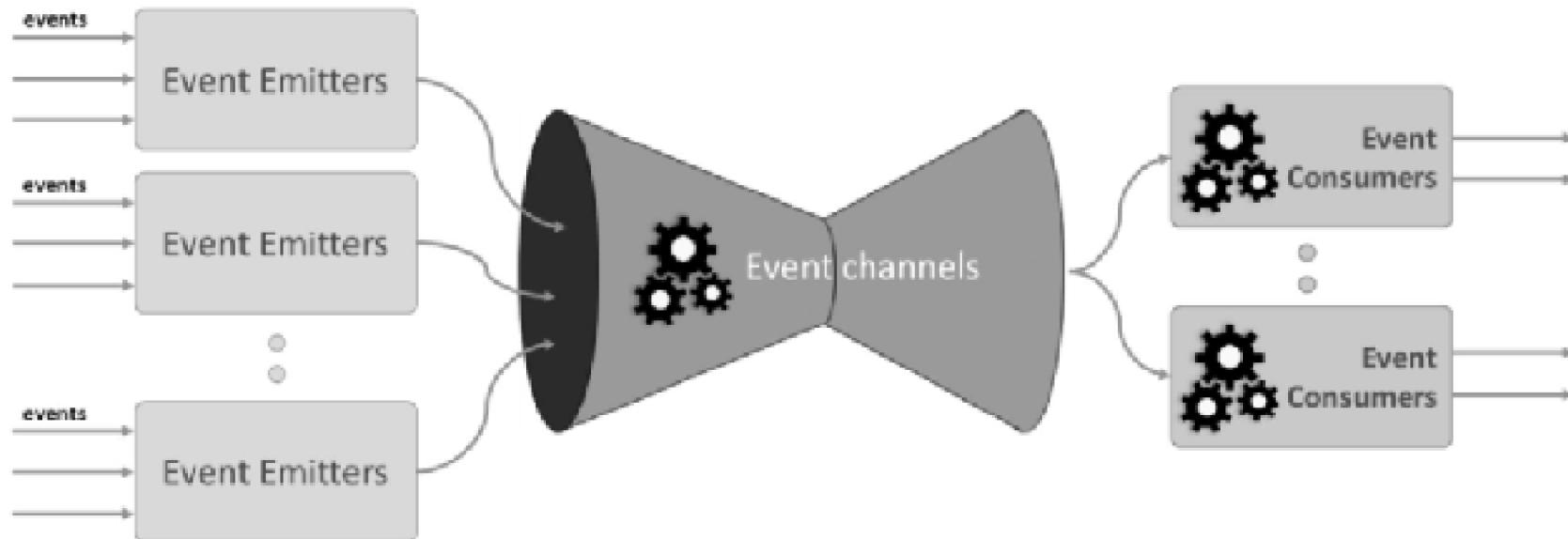
nivelul eveniment

- primește evenimente de la surse
- convertește într-un format comun respectivele evenimente

nivel comunicare - Canale de comunicație

Canal de evenimente

Entitate+protocol folosită pentru transportul evenimentelor



Modele de comunicare

- Publicare/Subscriere (Publish-subscribe (PubSub))
- Publicare/Subscriere bazată pe topică
- Punct la Punct
- Cerere/Raspuns (Request-reply)
- Stocheaza și retrimit (Store and forward)
- Pipeline

Produse și tehnologii

- Produse bazate pe protocolul XMPP
 - Ejabberd
- Produse bazate pe protocolul AMQP
 - RabbitMQ
 - OpenAMQP
 - Apache QPID
- Produse bazate pe protocolul STOMP
 - ActiveMQ
- JMS
- Altele (MS)
 - MSMQ
 - Azure Service Bus
 - ZeroMQ

Extensible Messaging and Presence Protocol (XMPP)

- în middleware pe mesaje cu XML
- pentru mesaje instant, sisteme tip publish-subscribe, VoIP, video, transfer de fisiere, jocuri, aplicații tip Internet of Things , smart grid si servicii de social network.

Advanced Message Queuing Protocol (AMQP)

- în middleware pe mesaje
- în layer de aplicatii open standard folosit în middleware orientate mesaj
- message orientation, queuing, routing (inclusiv point-to-point și publish-and-subscribe), reliability și securitate.
- este wire-level

RabbitMQ & STOMP

- **RabbitMQ** este un broker de mesaje open source sau middleware orientat pe mesaje care implementează protocolul (AMQP).
- **Simple (or Streaming) Text Oriented Message Protocol (STOMP)**, anterior TTMP, este un protocol simplu bazat pe text specific middleware orientat mesaj

Apache ActiveMQ & JMS

- **Apache ActiveMQ** este un broker de mesaje open source scris în Java împreună cu un client complet de Java Message Service (JMS).
- API-ul **Java Message Service (JMS)** este un middleware orientat pe servicii bazat pe Java (MOM) API

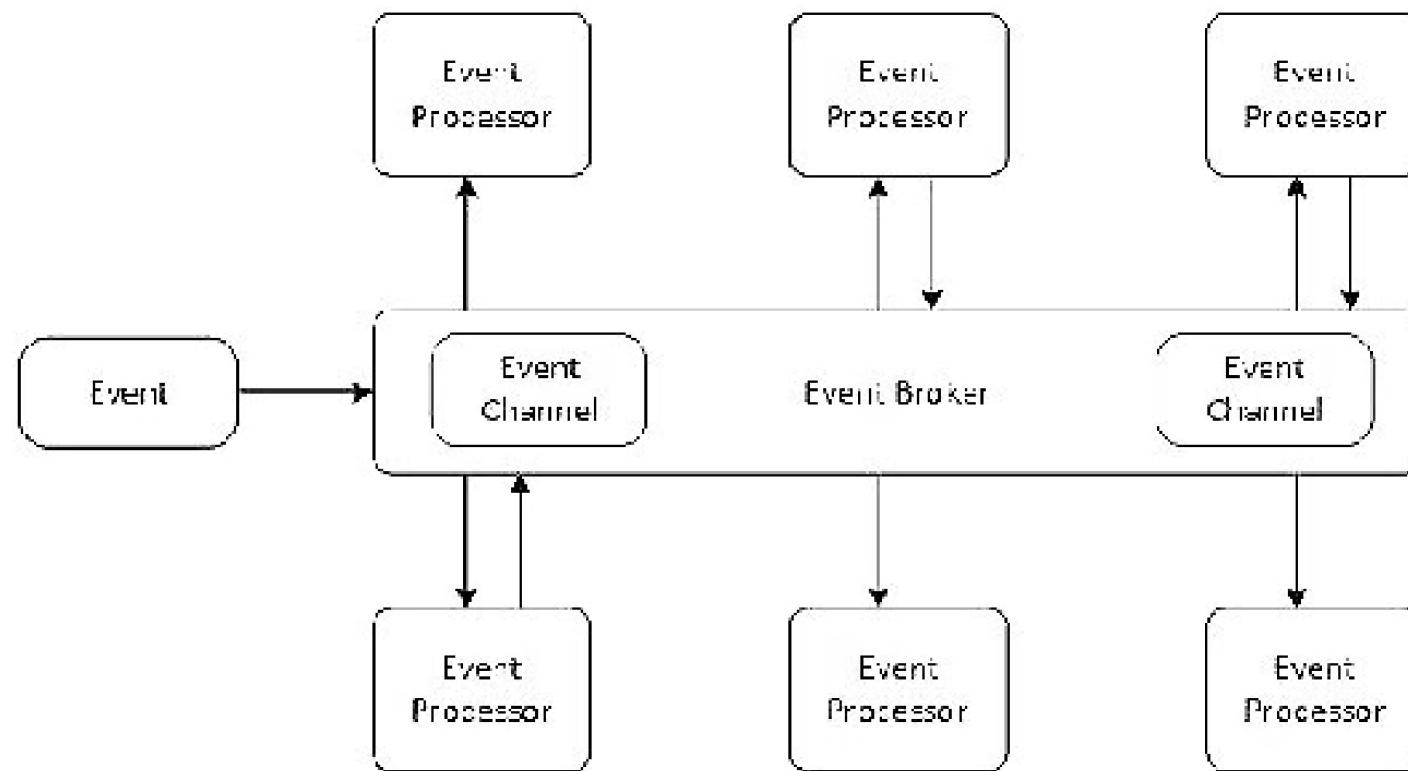
MSMQ & Windows Azure Service Bus

- **Microsoft Message Queuing** sau **MSMQ** este o coadă de mesaje dezvoltată de MS
- **Windows Azure Service Bus**
 - în cloud
 - notificări către dispozitive mobile
 - Cozi

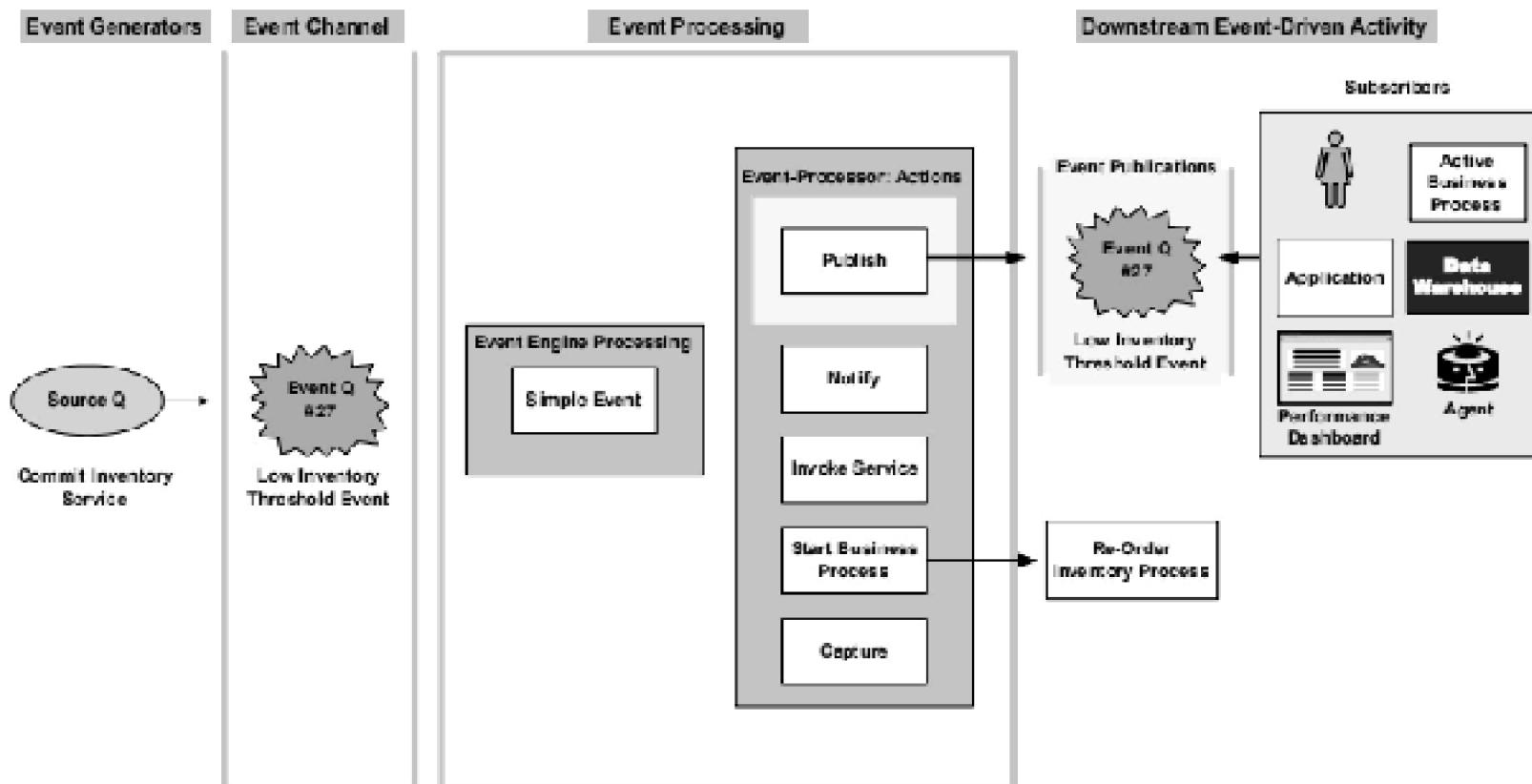
ZeroMQ

- **ØMQ** un alt sistem de mesaje orientat pe middleware

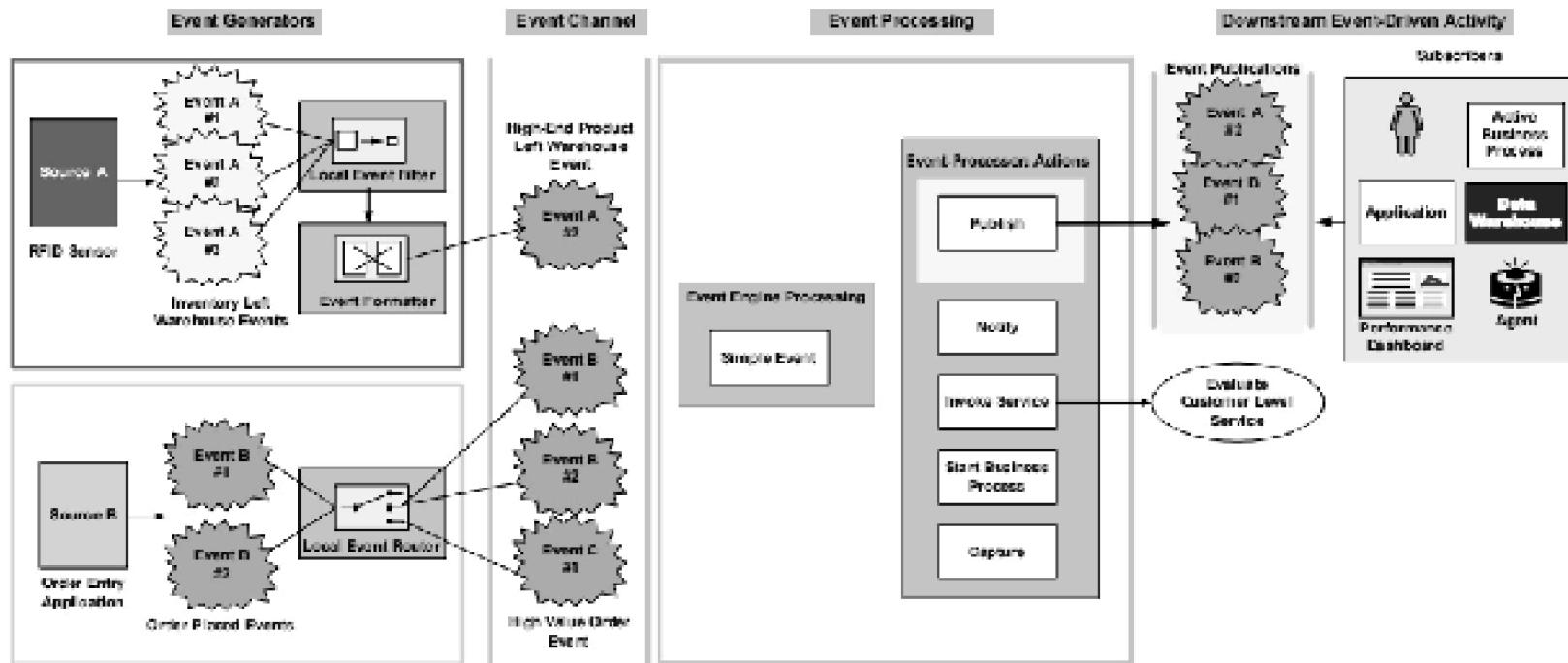
Modele de comunicare - broker



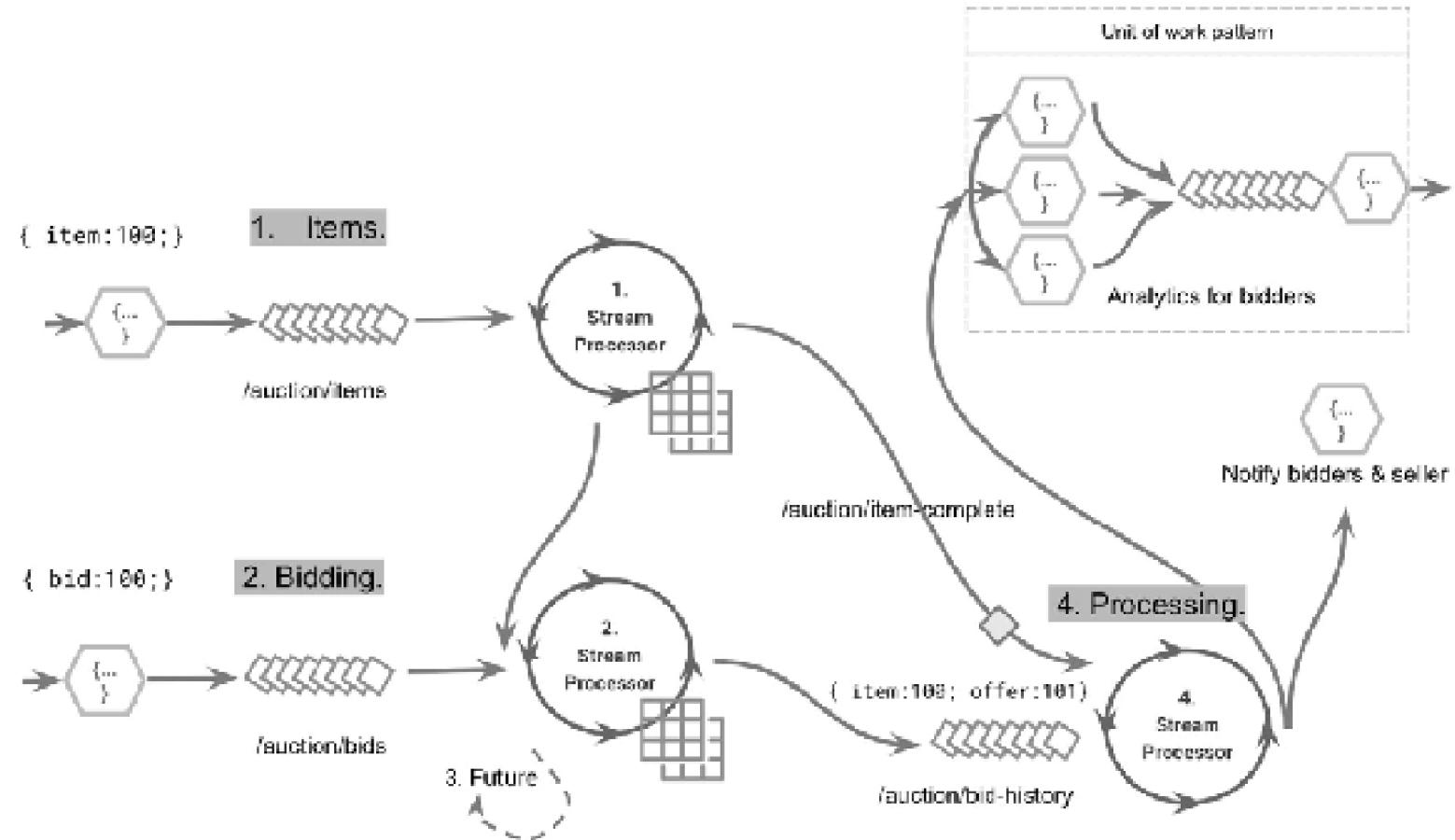
procesare eveniment simplu



procesare flux de evenimente

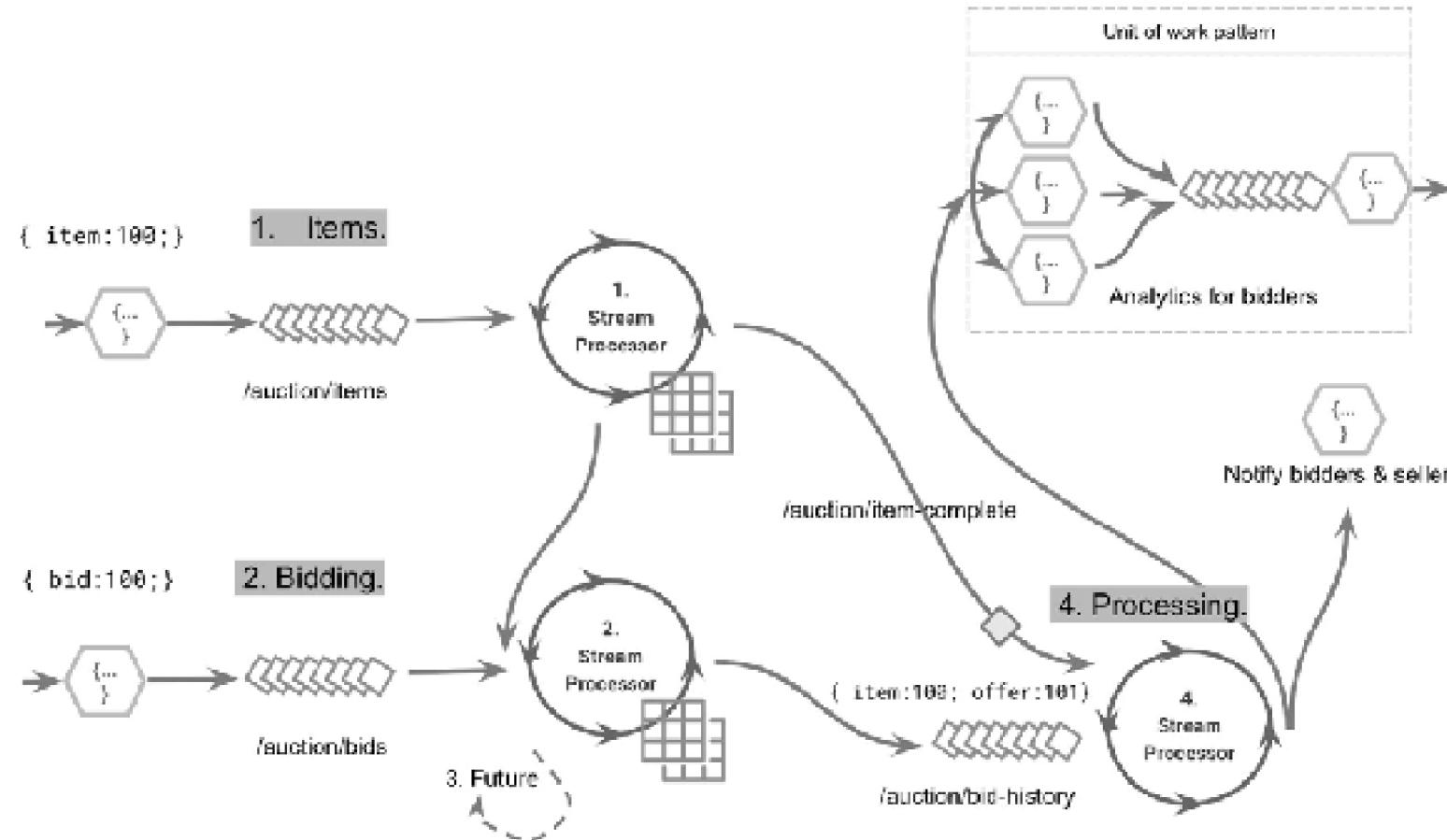


EDA - exemplu - licitație



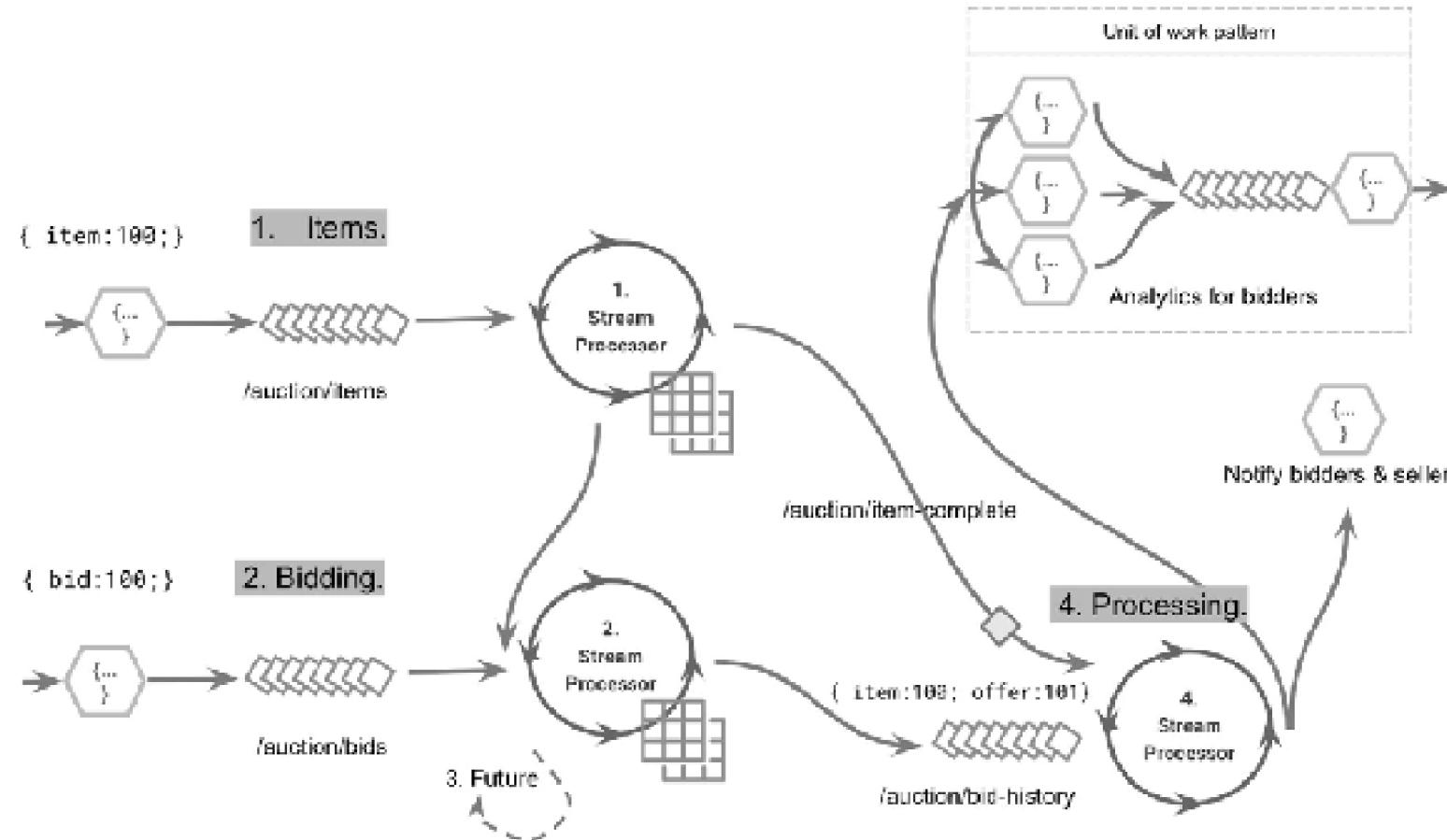
graful pentru fluxul de afaceri pentru licitație

EDA - exemplu - licitație - funcțiile de bază



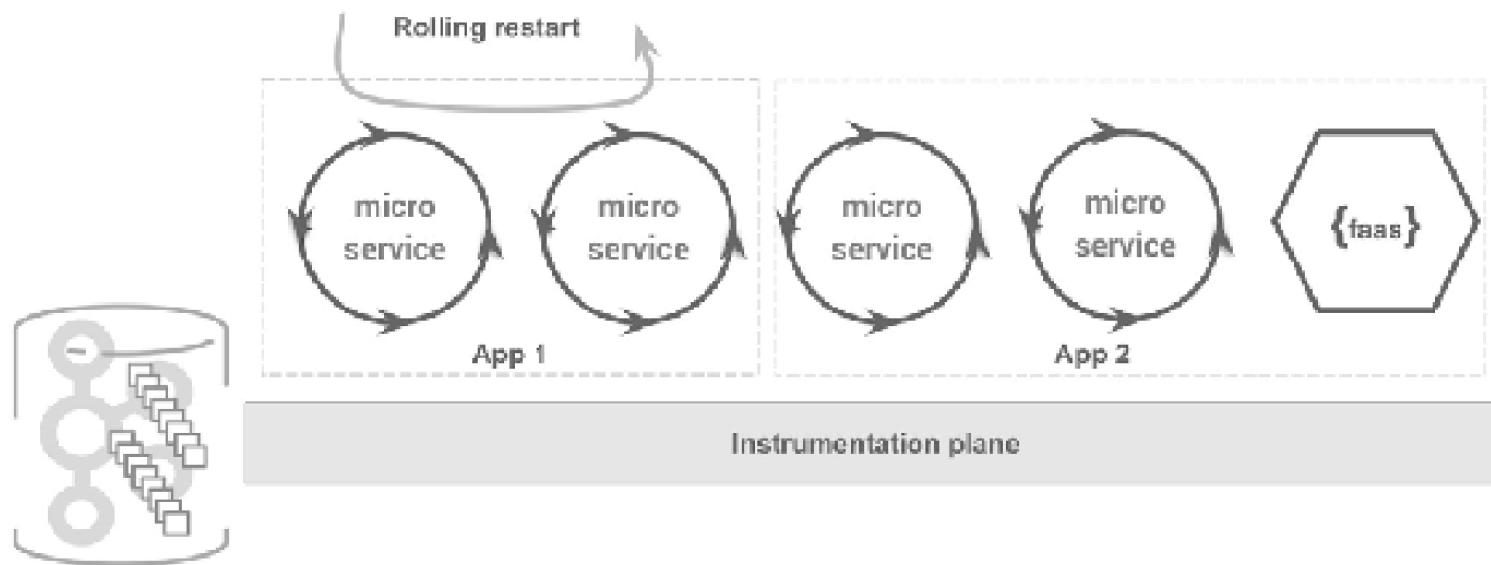
graful pentru fluxul de afaceri pentru o licitație

EDA - exemplu - licitație - funcțiile de bază



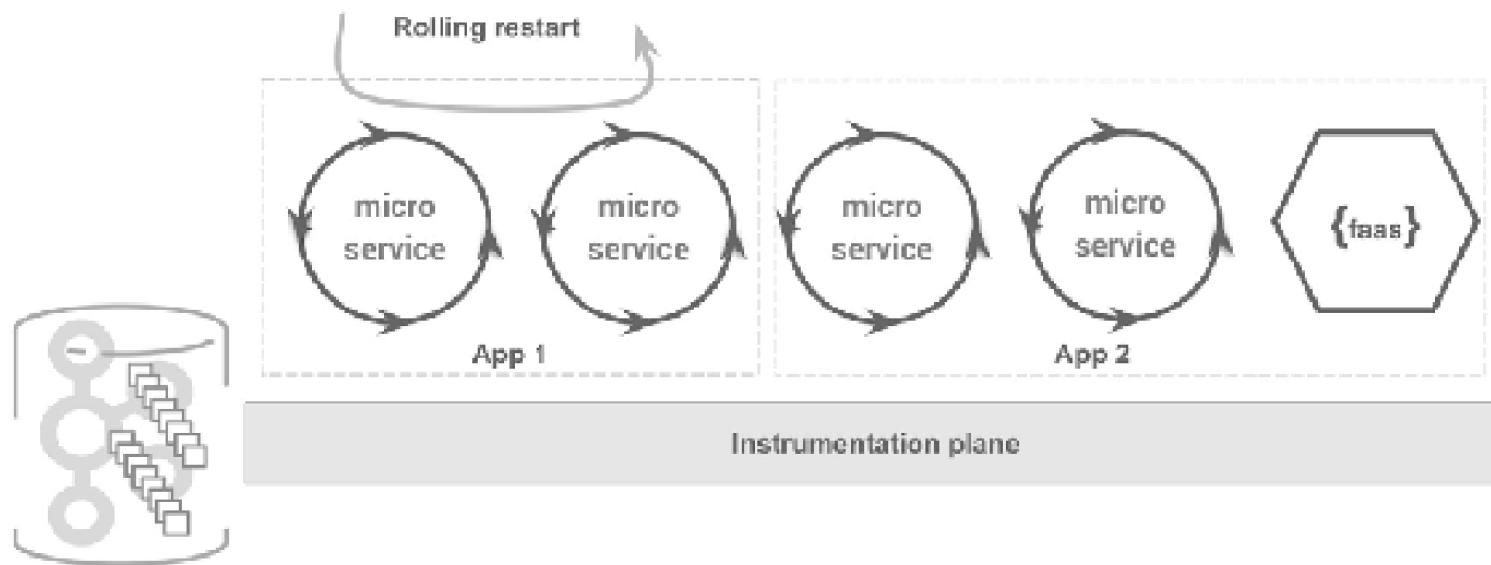
graful pentru fluxul de afaceri pentru o licitație

EDA - exemplu - licitație - Control - Oprire



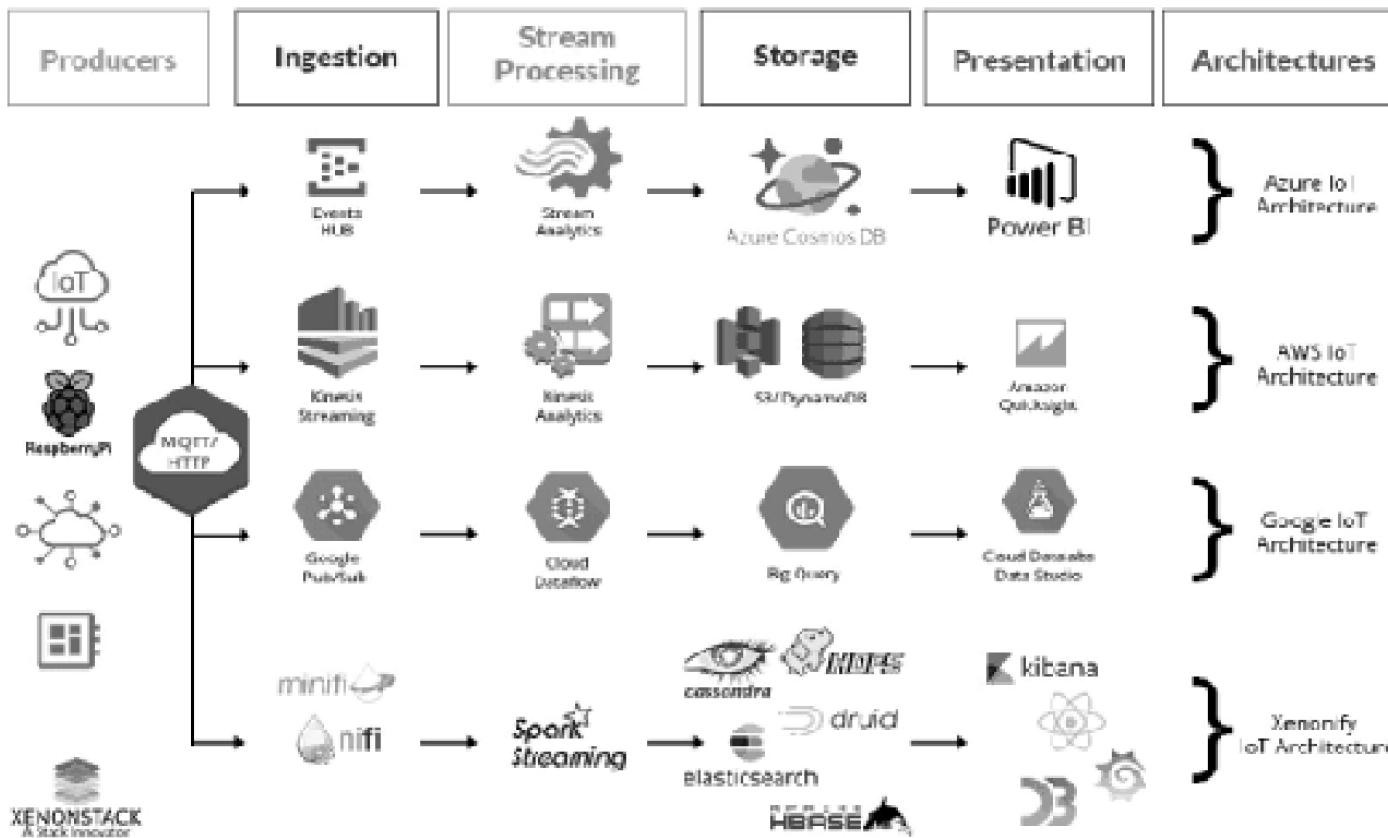
oprirea/reporarea microserviciilor din flux

EDA - exemplu - licitație - Control - Oprire



oprirea/reporarea microserviciilor din flux

EDA pe fluxuri - orice inclusiv IoT

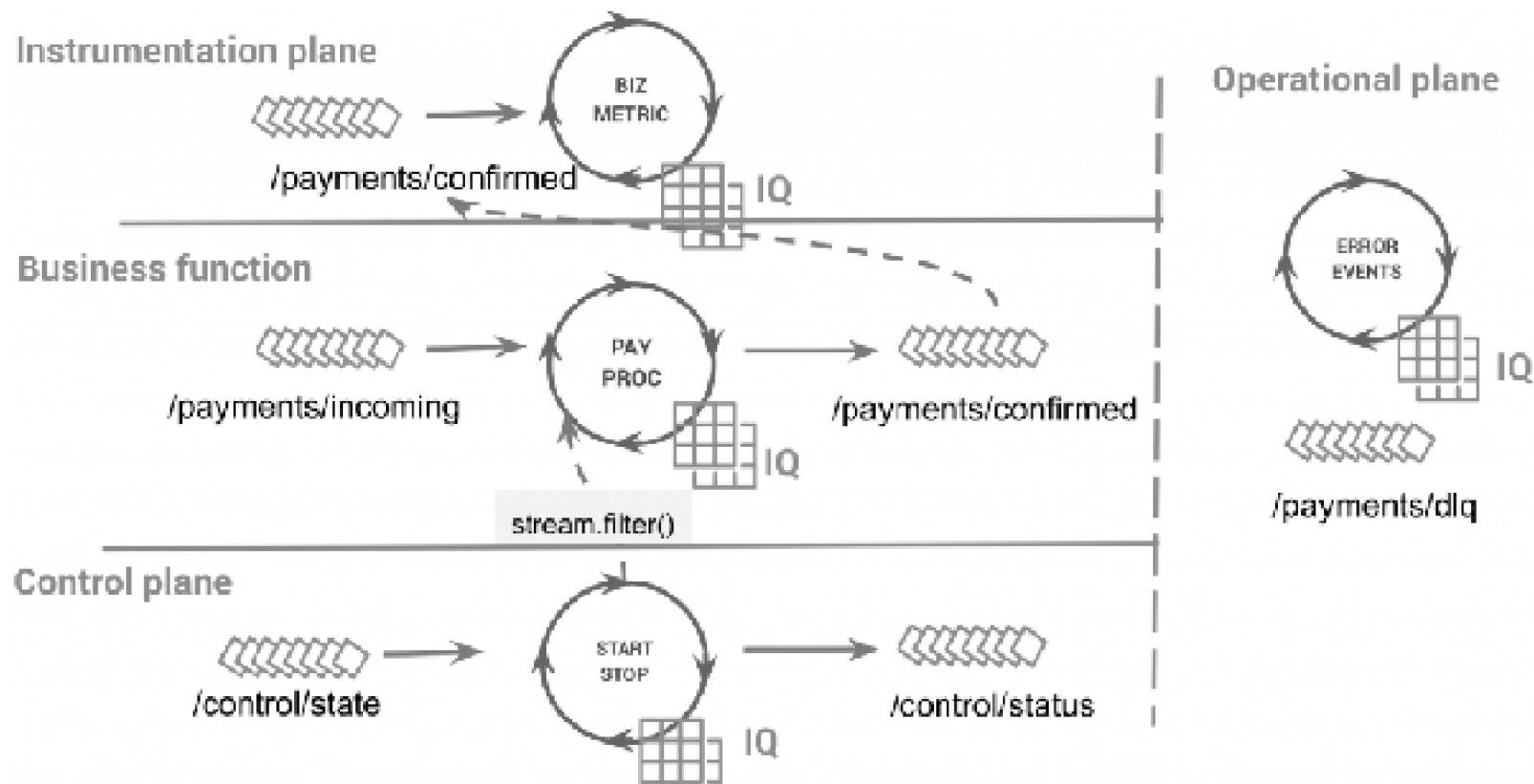


Sisteme Distribuite

Cursul 11

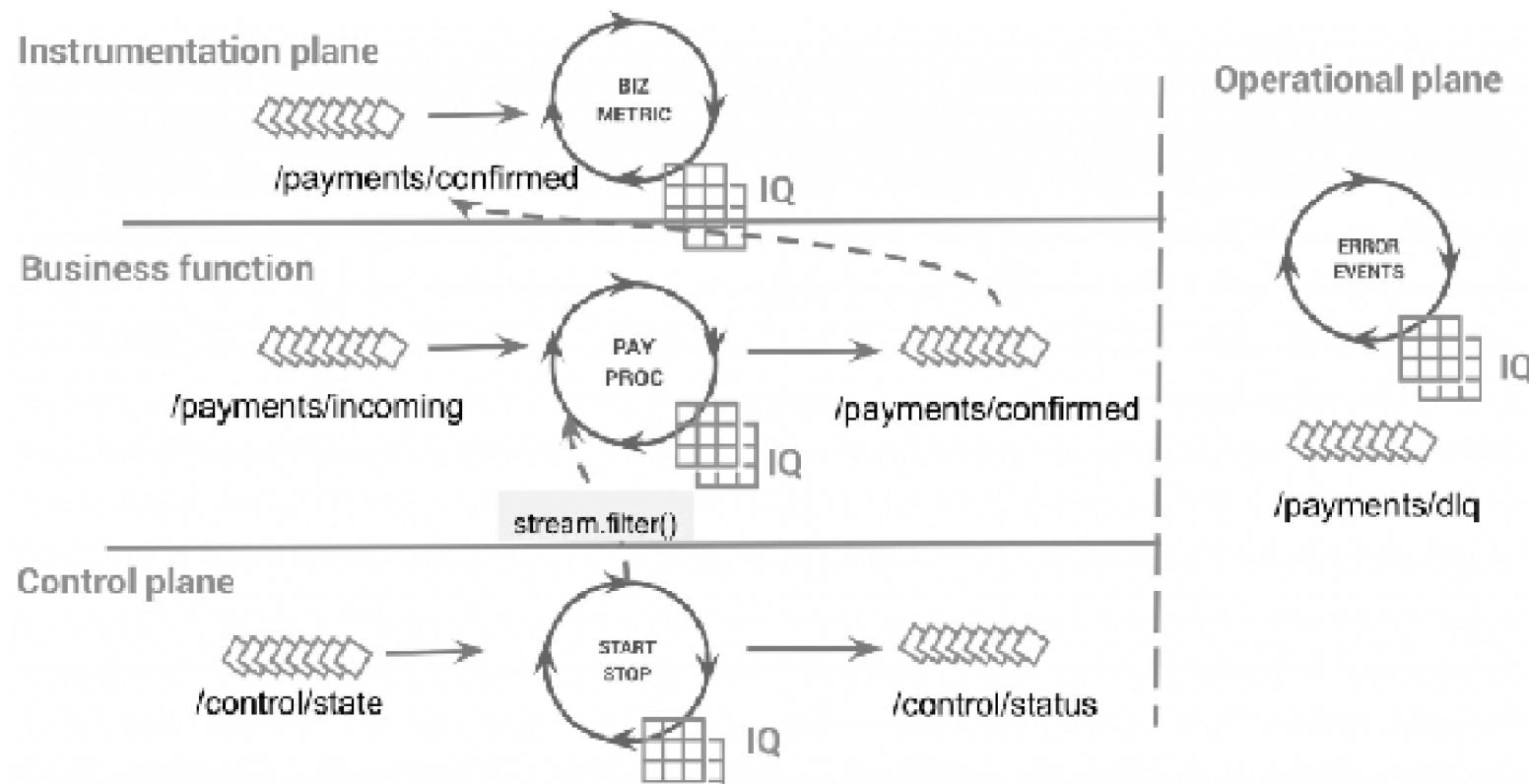
Mihai Zaharia

Afaceri și iar afaceri!

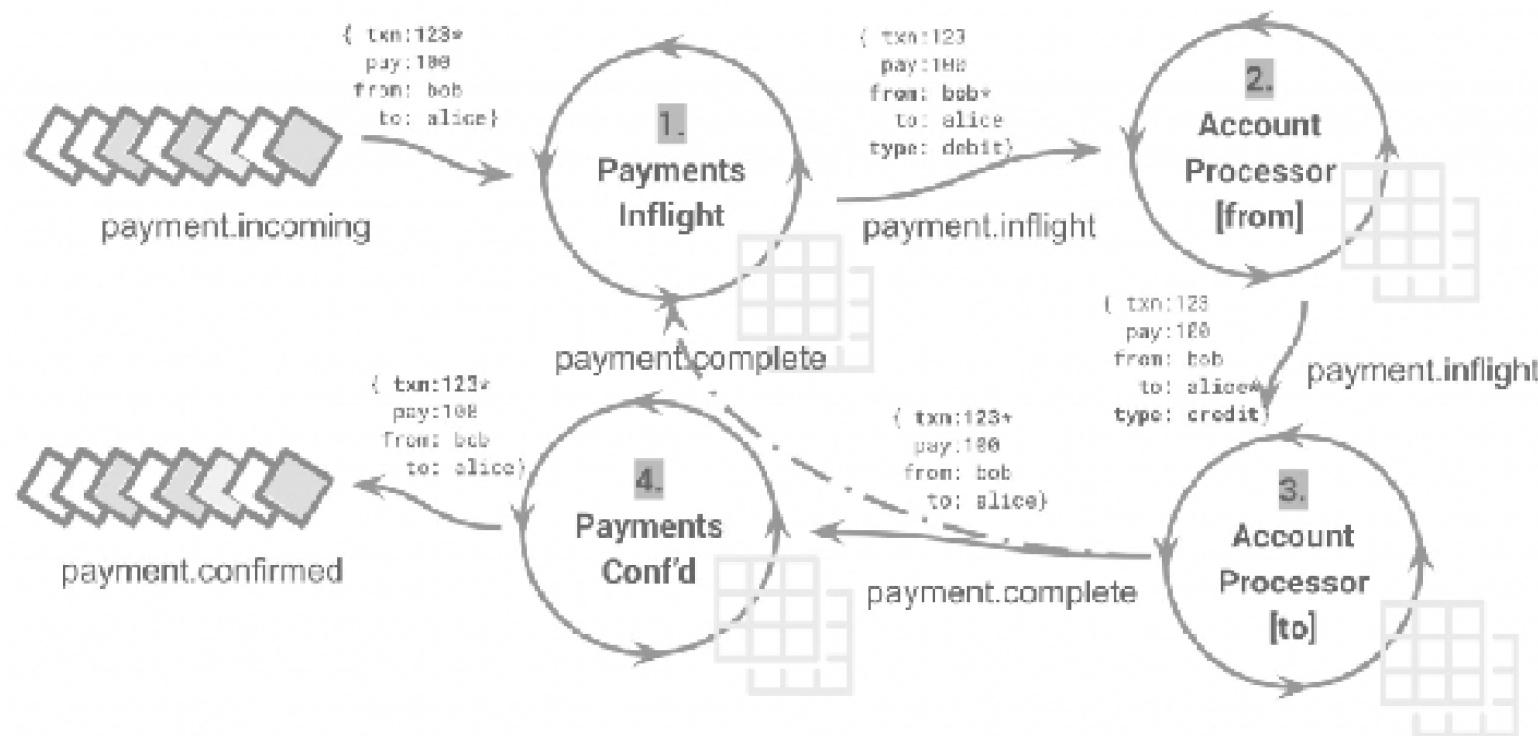


Cele patru planuri ale unei arhitecturi bazate pe fluxuri de evenimente

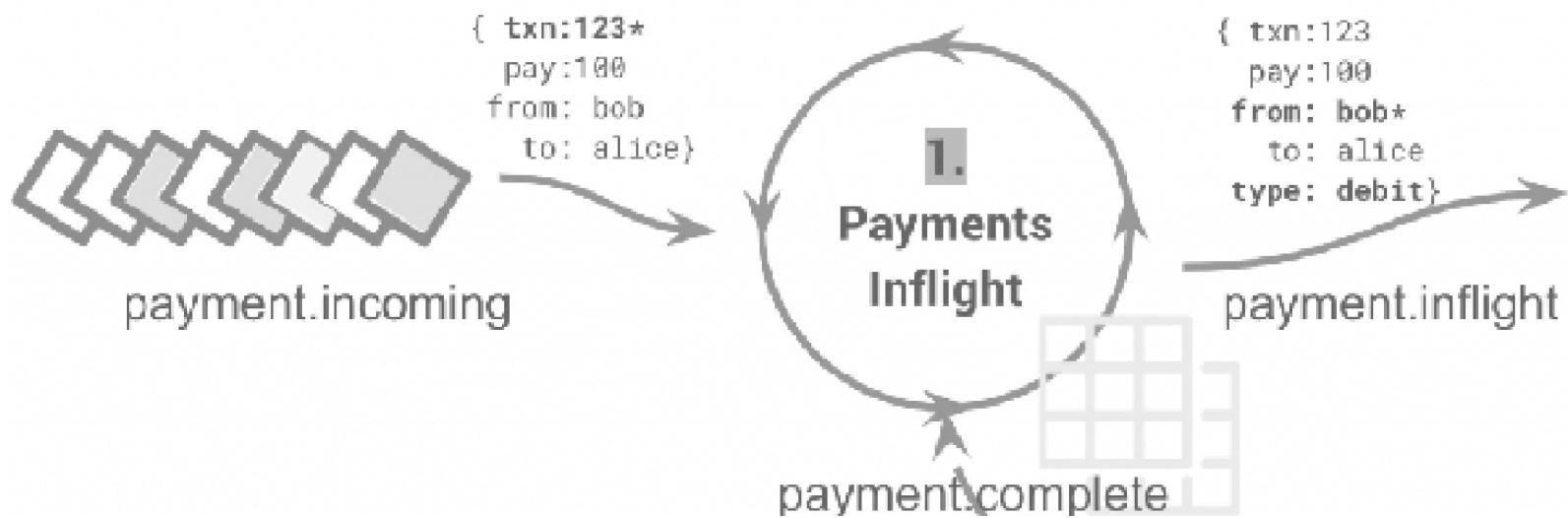
Detalii



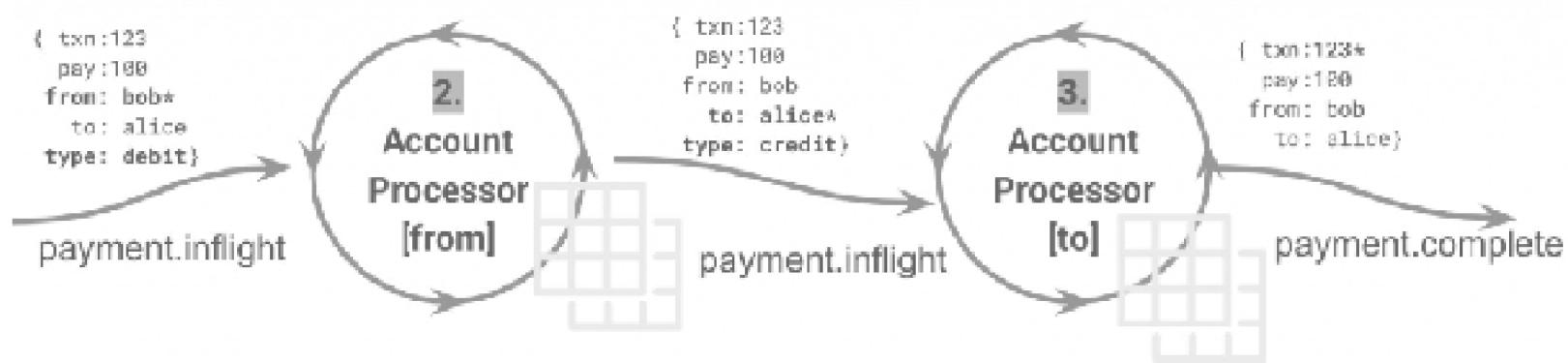
Fluxul de rezolvare al plășilor



Analiza fluxului de evenimente

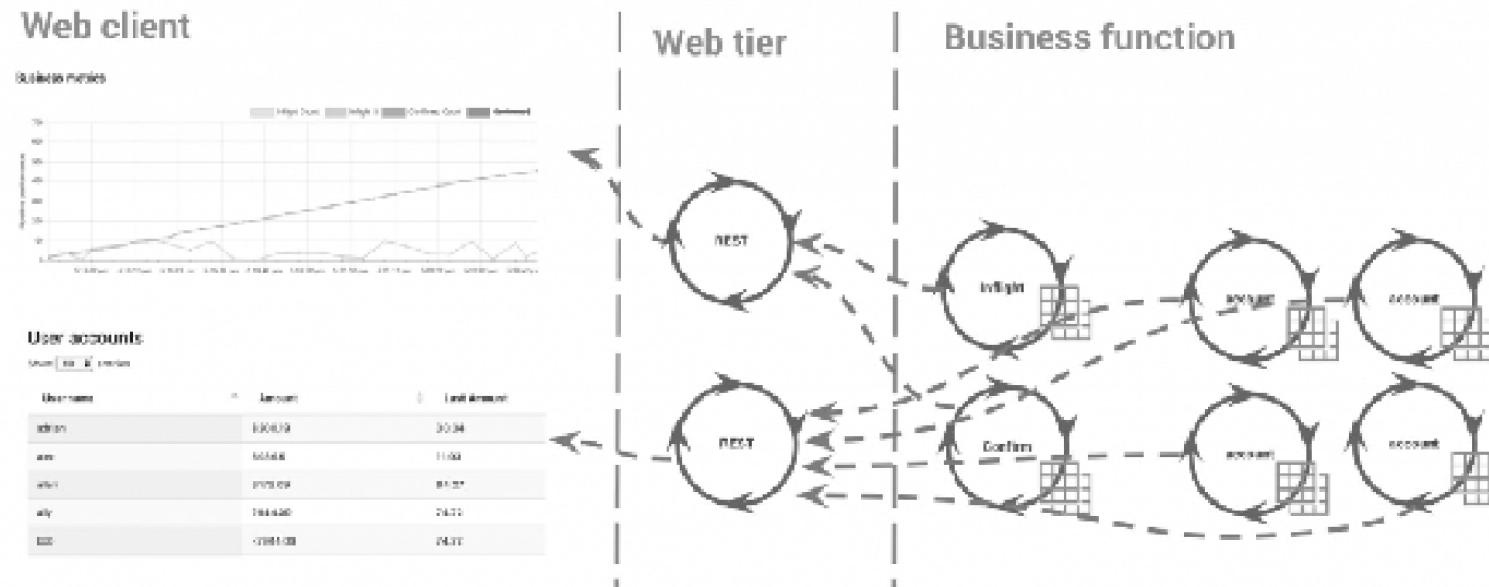


Analiza procesorului de cont

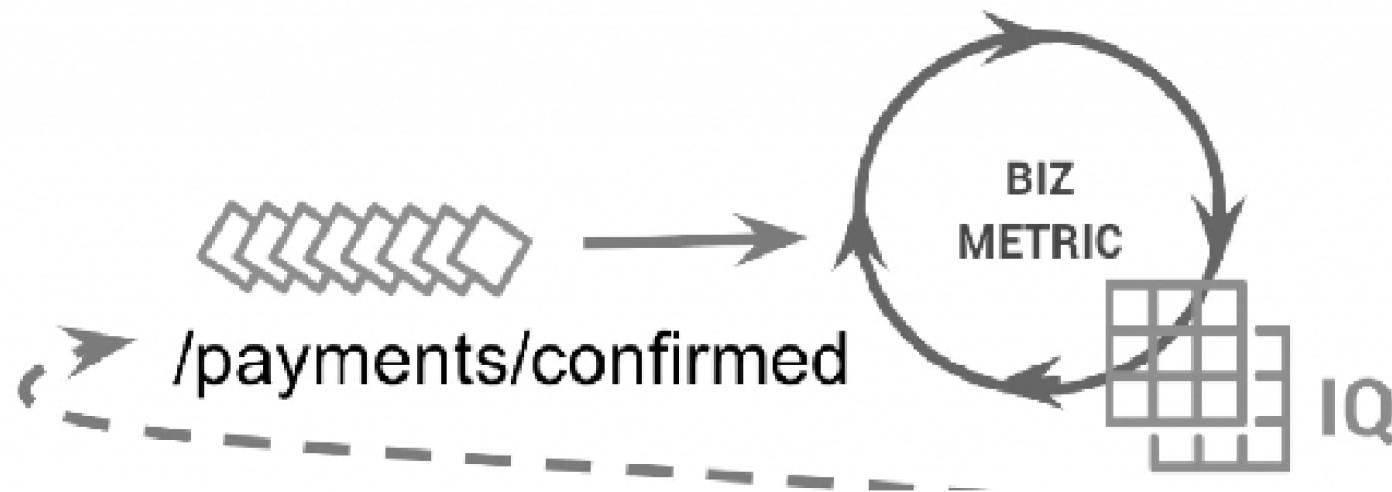


- X

Detalii pentru confirmarea balanței

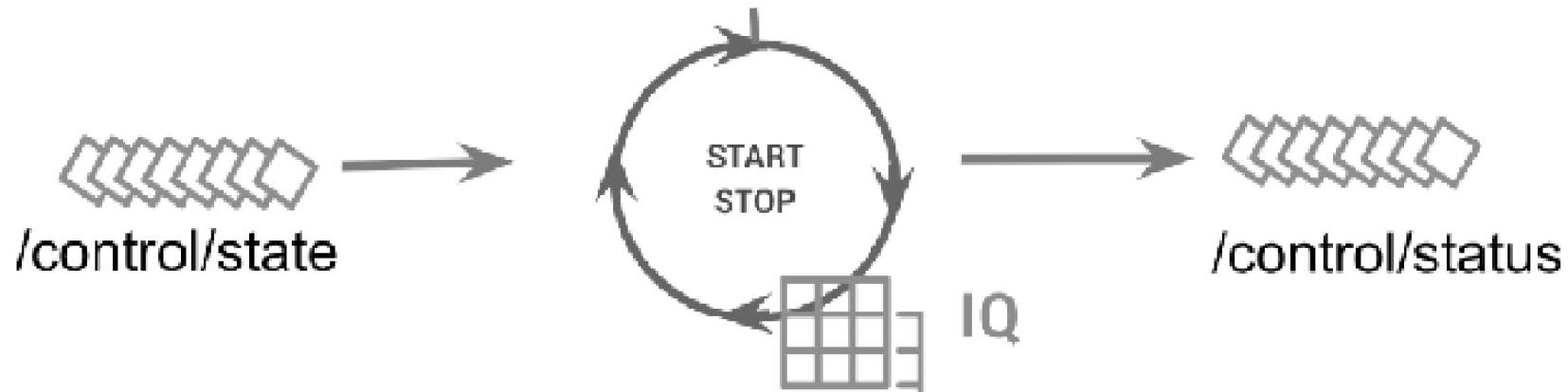


Detalii la planul de instrumentație



- X

Controlul Execuției

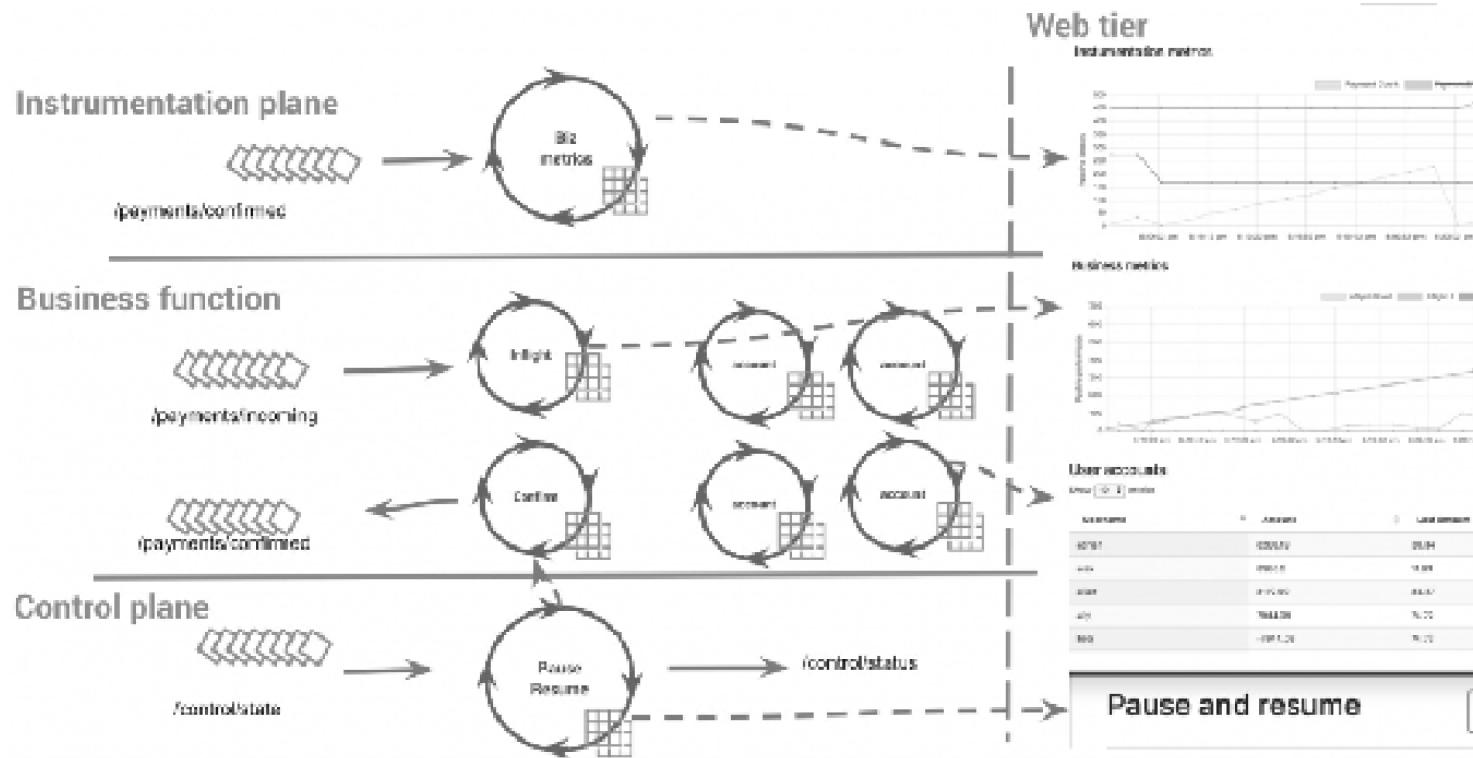


- X

Planul operațional

- Logica Aplicației
- Jurnale de erori
- Jurnale de securitate
- Informații pentru restaurare
- Cozile de mesaje pierdute

Modelul de instalare



Fără servere?

Origine

Unde se află în contextul tehnologic?

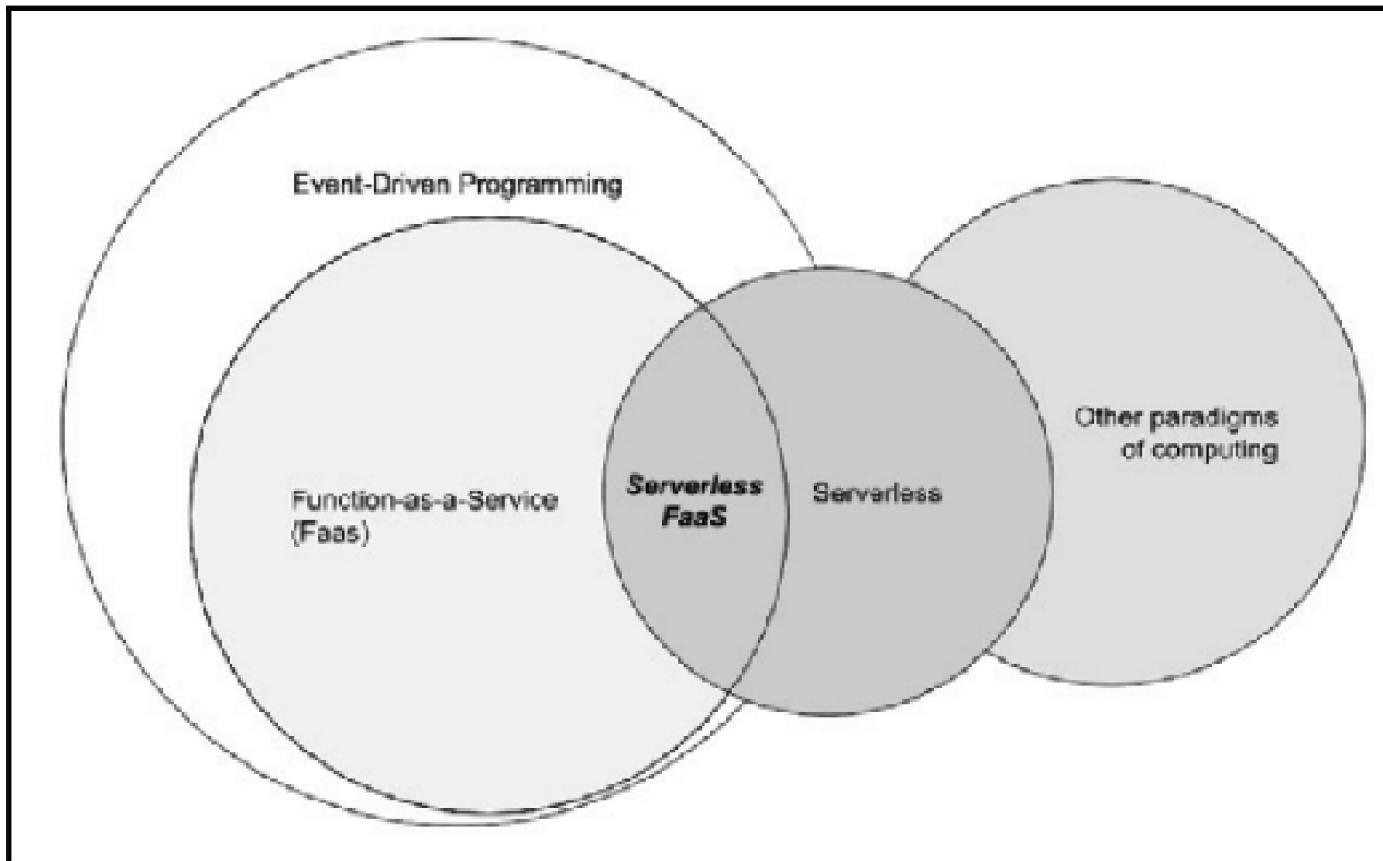
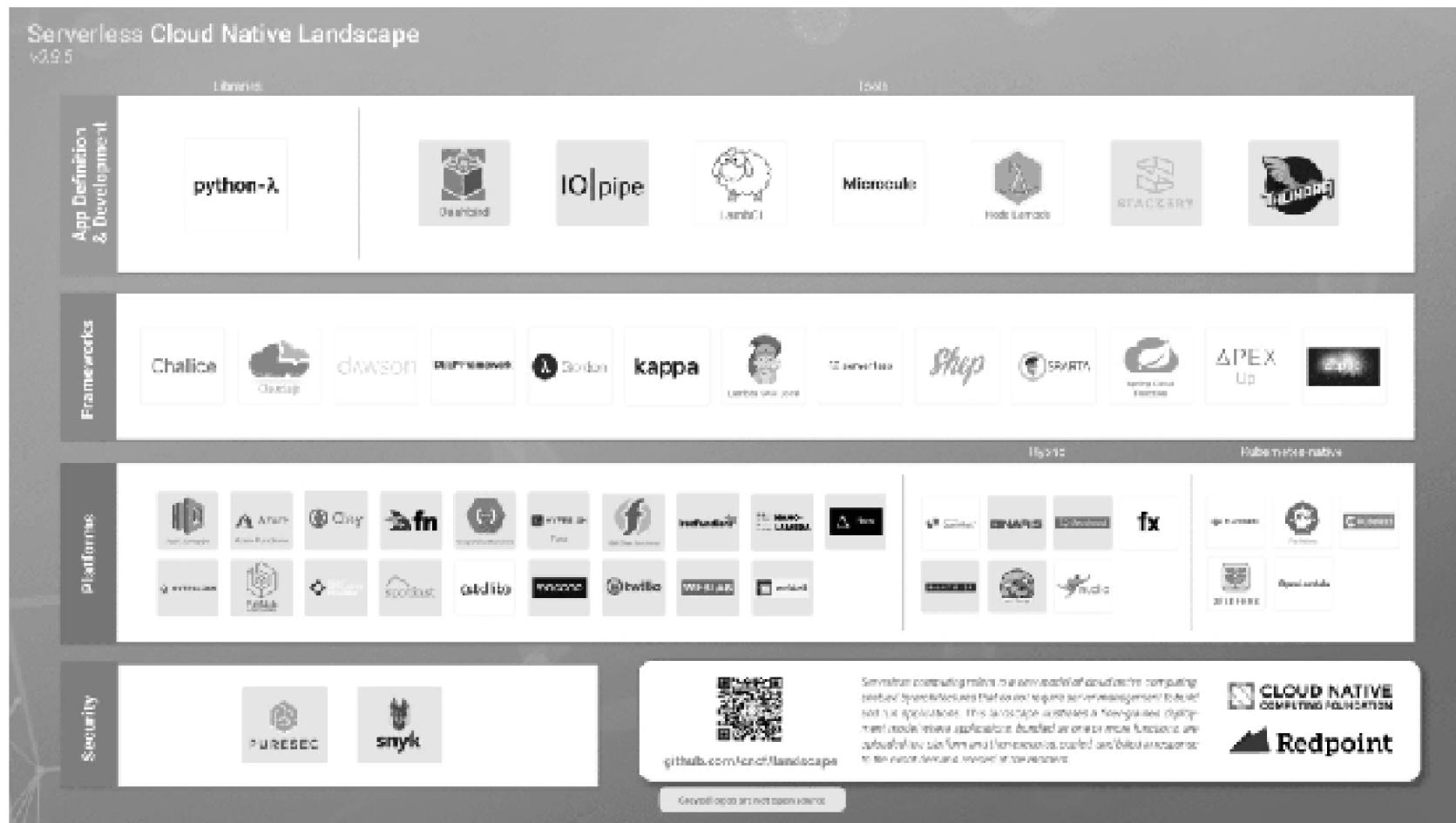


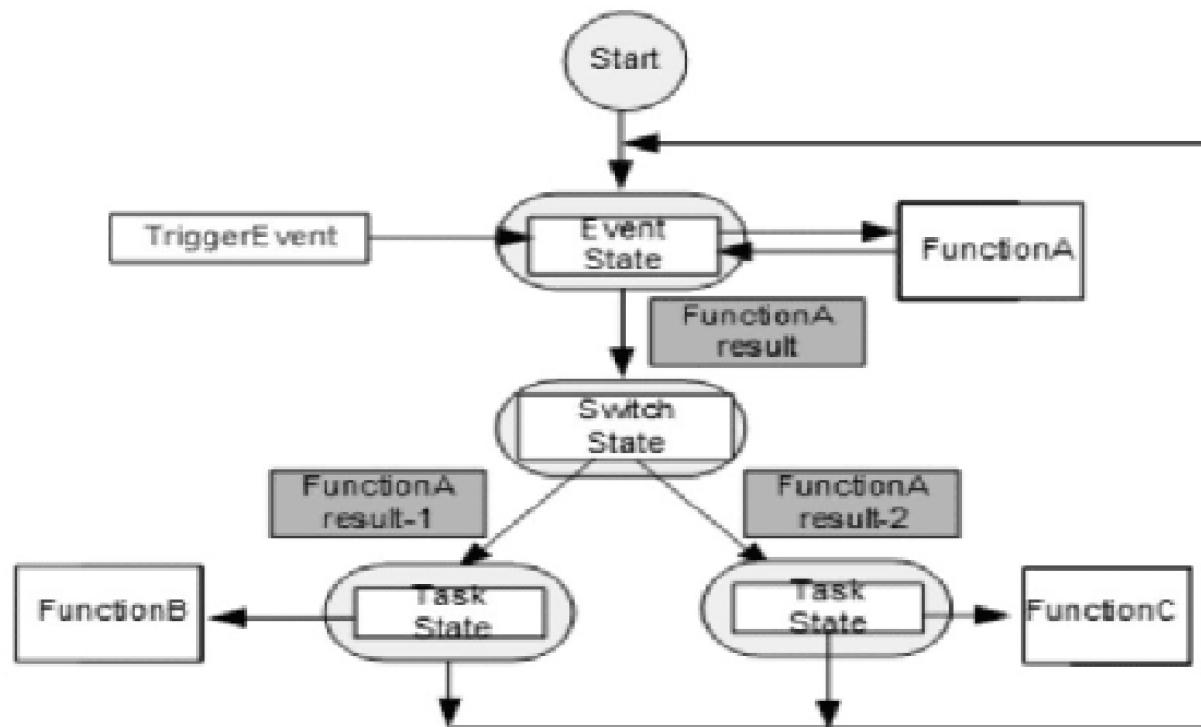
Diagrama Venn de plasare a conceptului

Serverless Working Group - CNCF



- Harta aplicațiilor serverless în nor

Modelul de functionare al FaaS



- X

Care este totuși diferența între PaaS și FaaS?

“If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless”

Adrian Cockcroft

VP Cloud Architecture Strategy at AWS

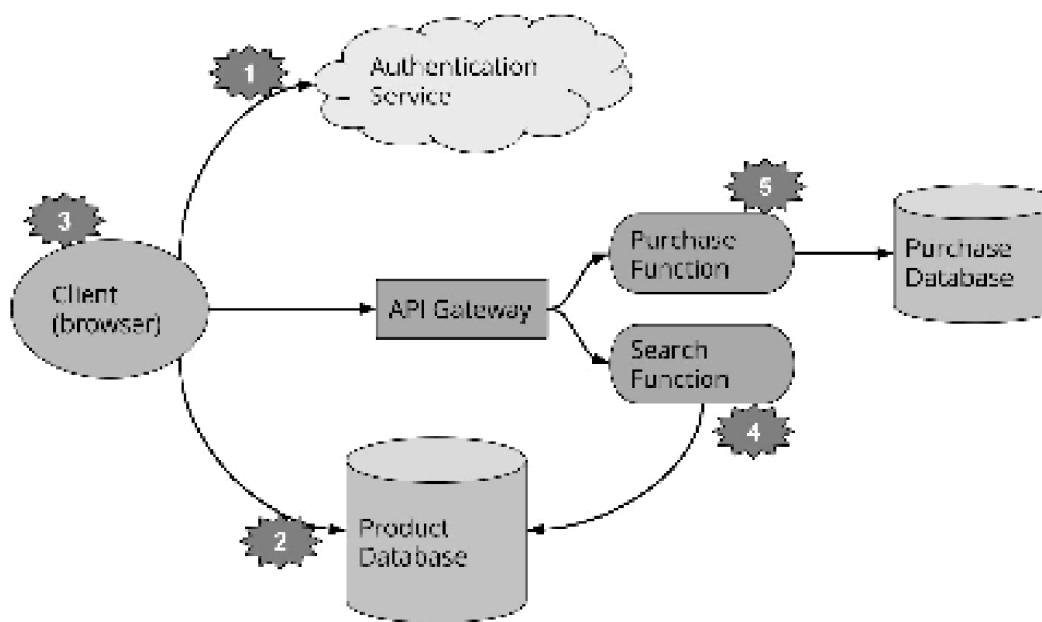
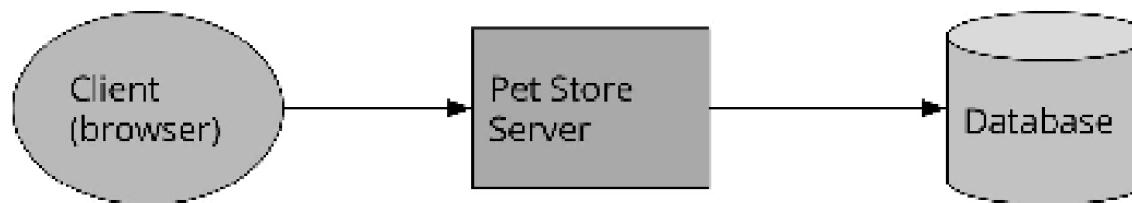
Este aşa de important?

- și da și nu
- depinde la ce am nevoie de el
- limitări
 - timp
 - memorie
- tehnologie proprietară
- soluții publice
- confidențialitate ?

Avantaje:

- induc încărcări mici
- decuplare maximă
- scalabilitate
- eficiente economic

Exemplu simplu - multilevel clasic



"favoritul" meu - magazinul on-line

Exemple simple - bazat pe mesaje



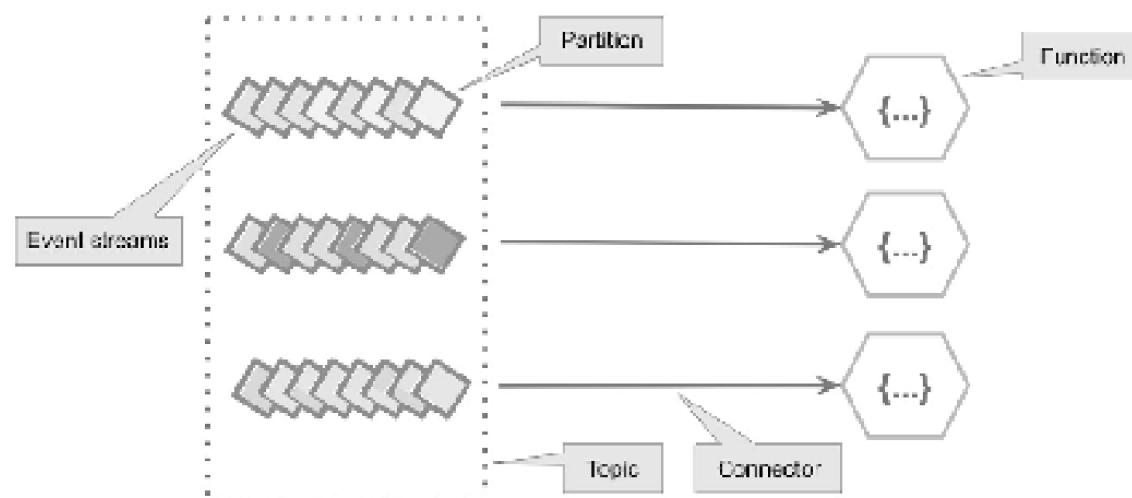
- arhitectură orientată pe mesaje - clasică



- arhitectură orientată pe mesaje - cu FaaS

Evenimentul primul și FaaS

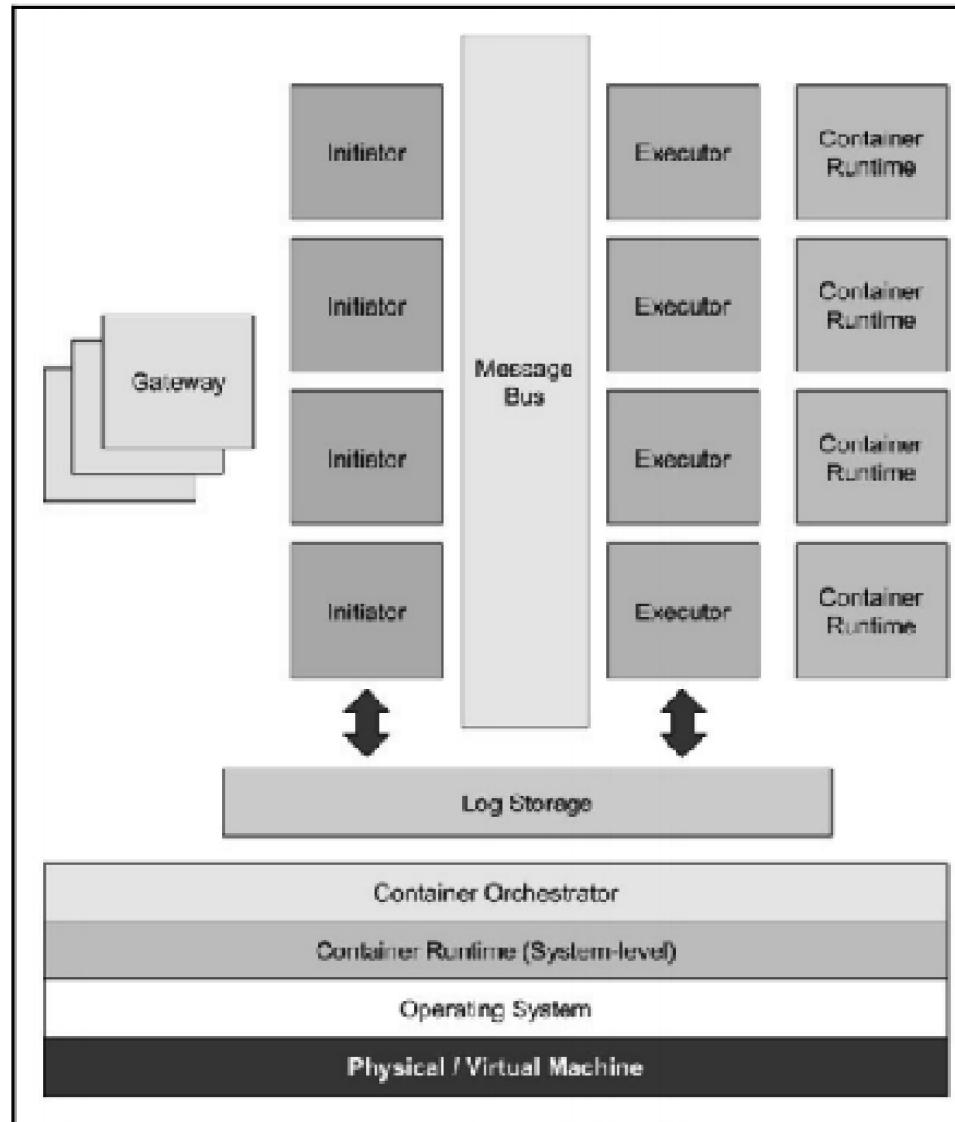
- furnizor de fluxuri
- kafka
- coregrafia și încâlceala
- nivelurile app
- conector



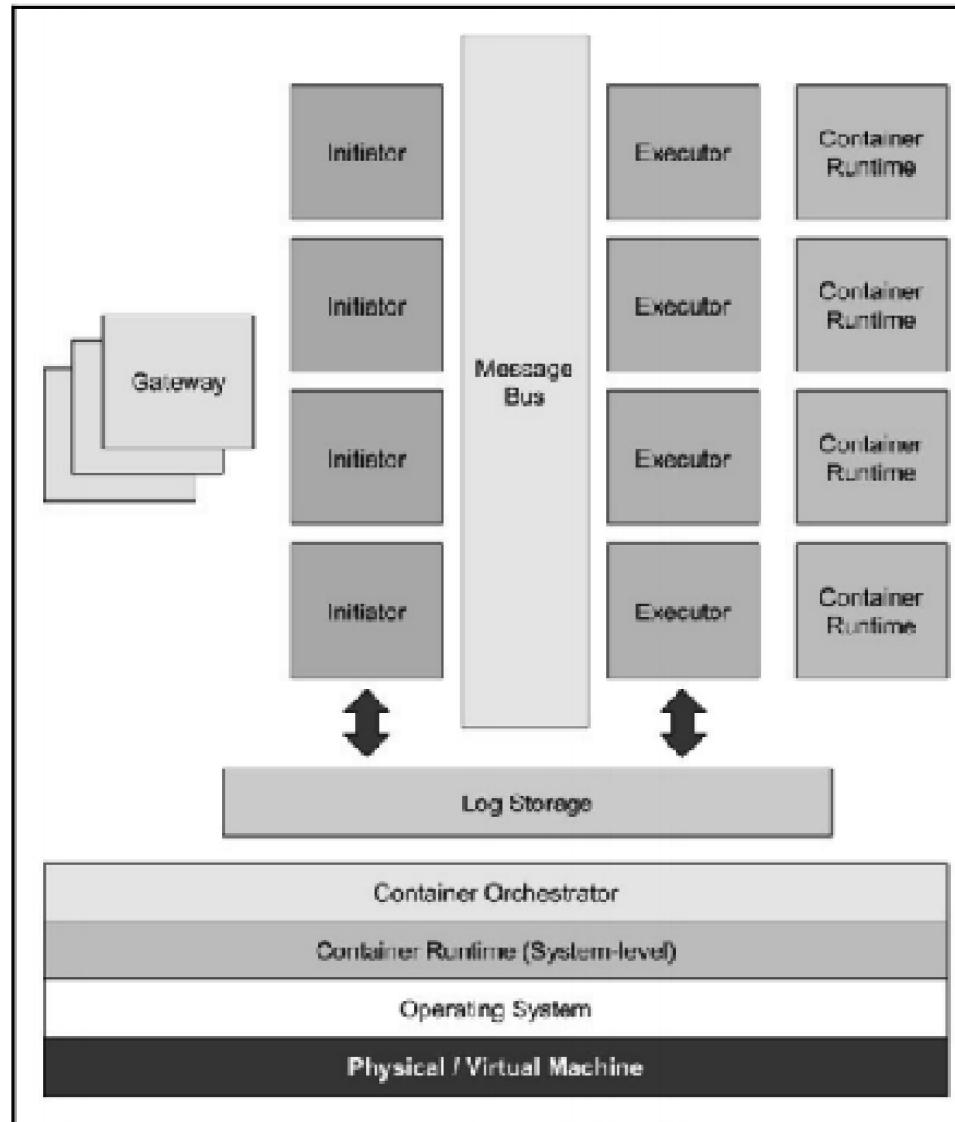
FaaS și procesarea pe flux

- *"AWS Lambda and serverless architectures are well-suited for stream processing workloads which are often event-driven and have spiky or variable compute requirements"*
 - Amazon, 2018

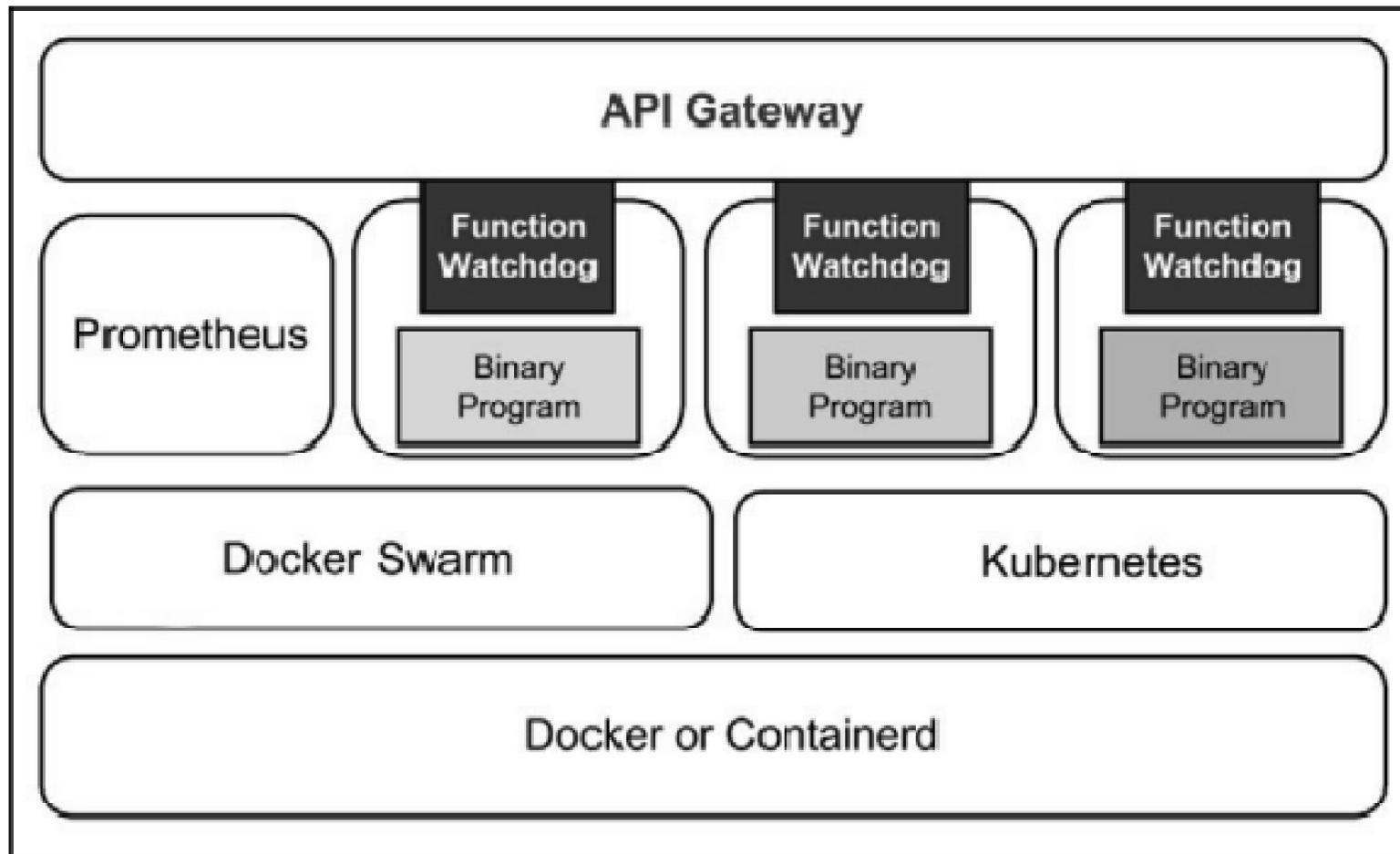
Arhitectura FaaS - nivelurile Sistem



Arhitectura FaaS - nivelurile FaaS

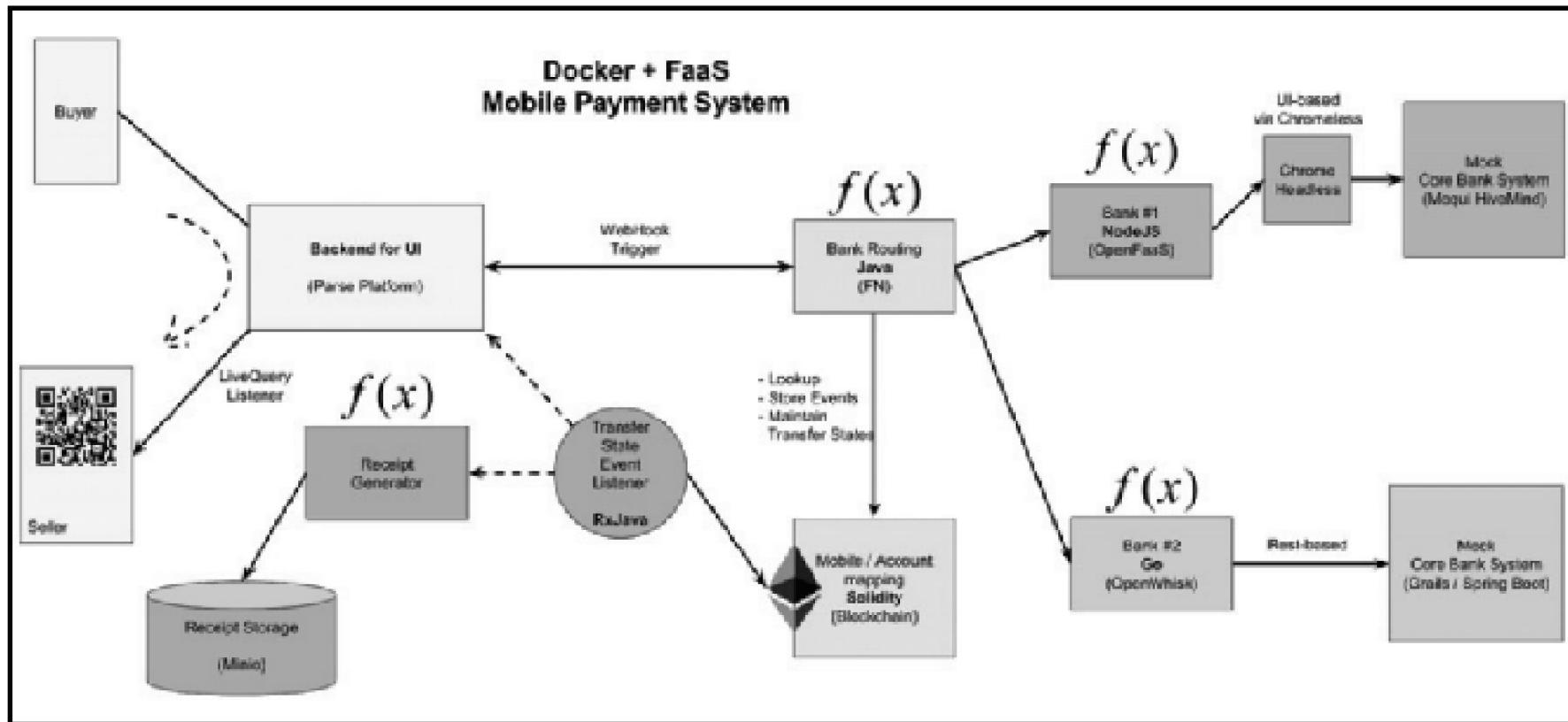


Componentele OpenFaaS

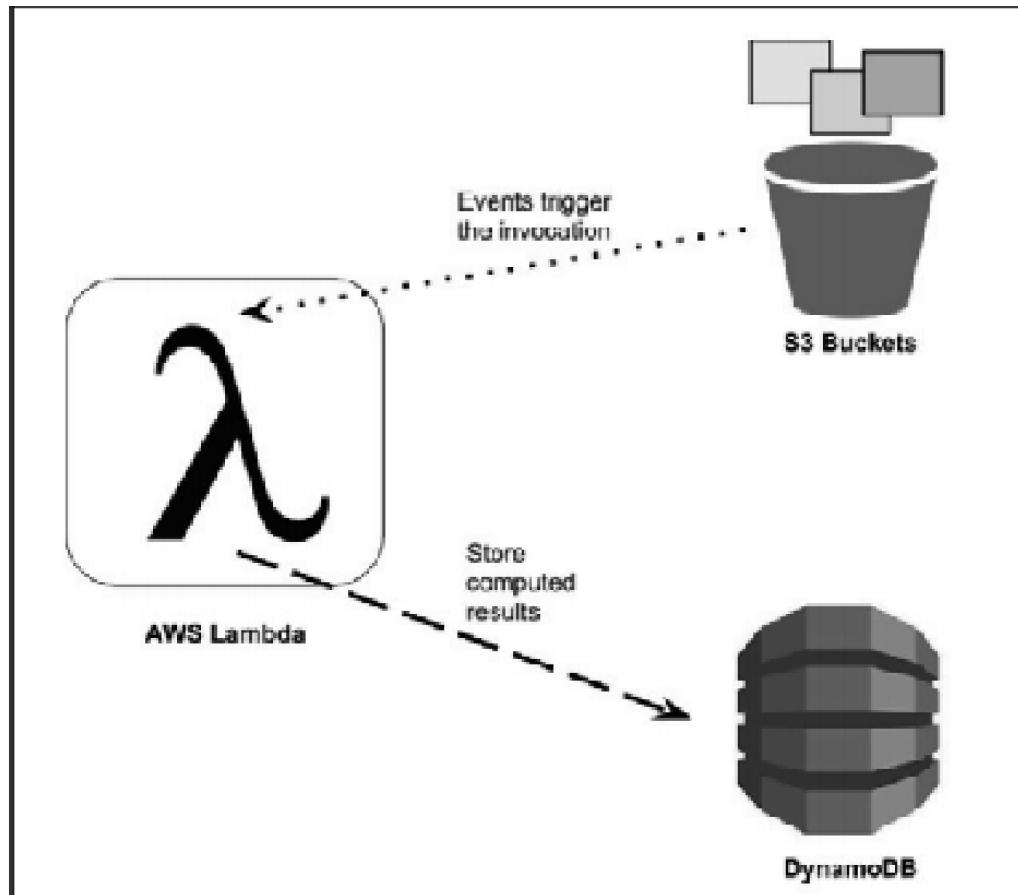


Arhitectura OpenFaaS

Exemplu de caz de utilizare pentru FaaS

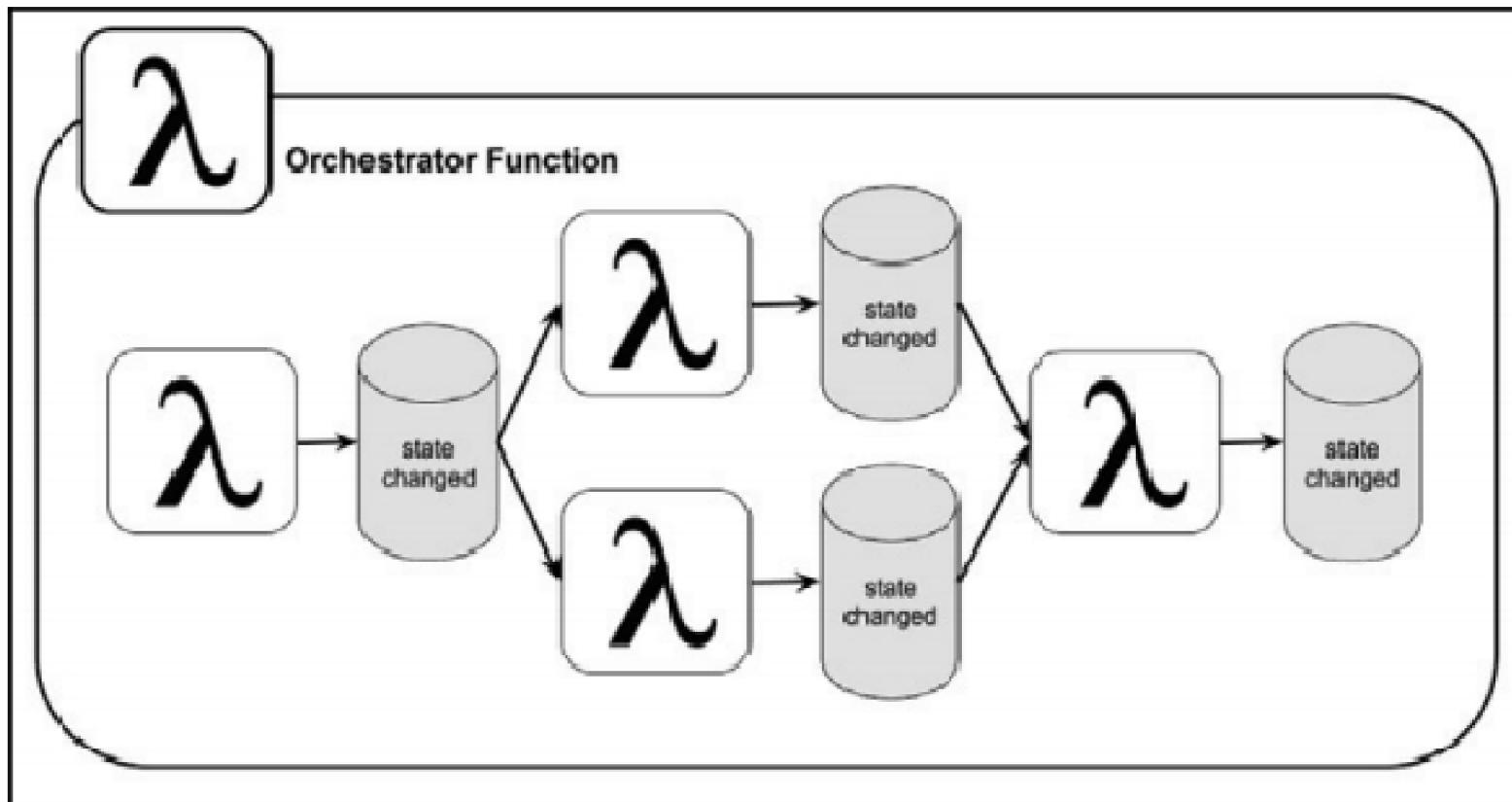


AWS Lambda



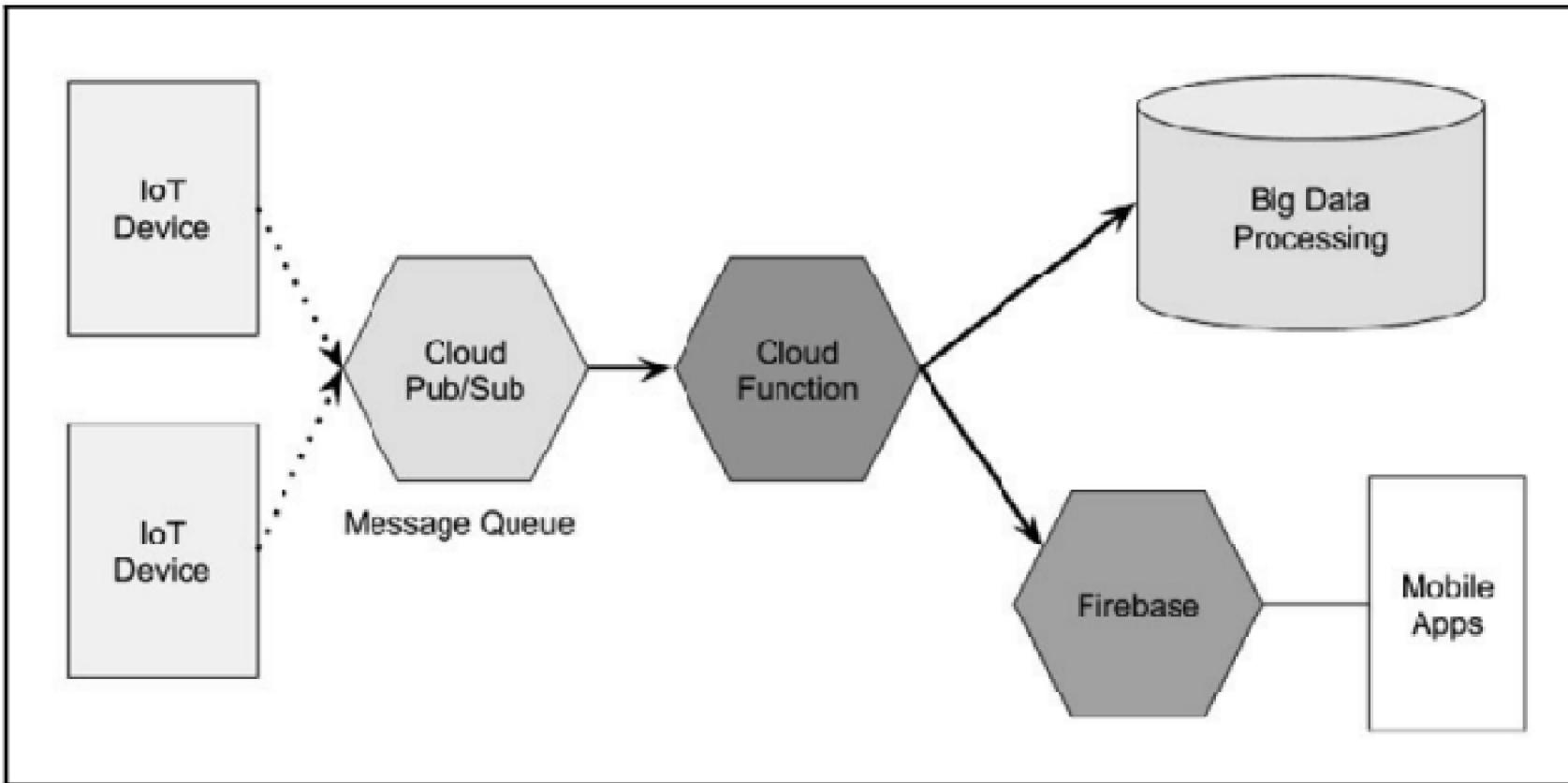
Exemplu simplu de utilizare

Funcții durable Azure



orchestrator bazat pe funcții durable în Azure

FaaS în norul Google



exemplu simplu de IoT implementat utilizând Google FaaS

Sisteme Distribuite

Cursul 12

Mihai Zaharia

Big data governance

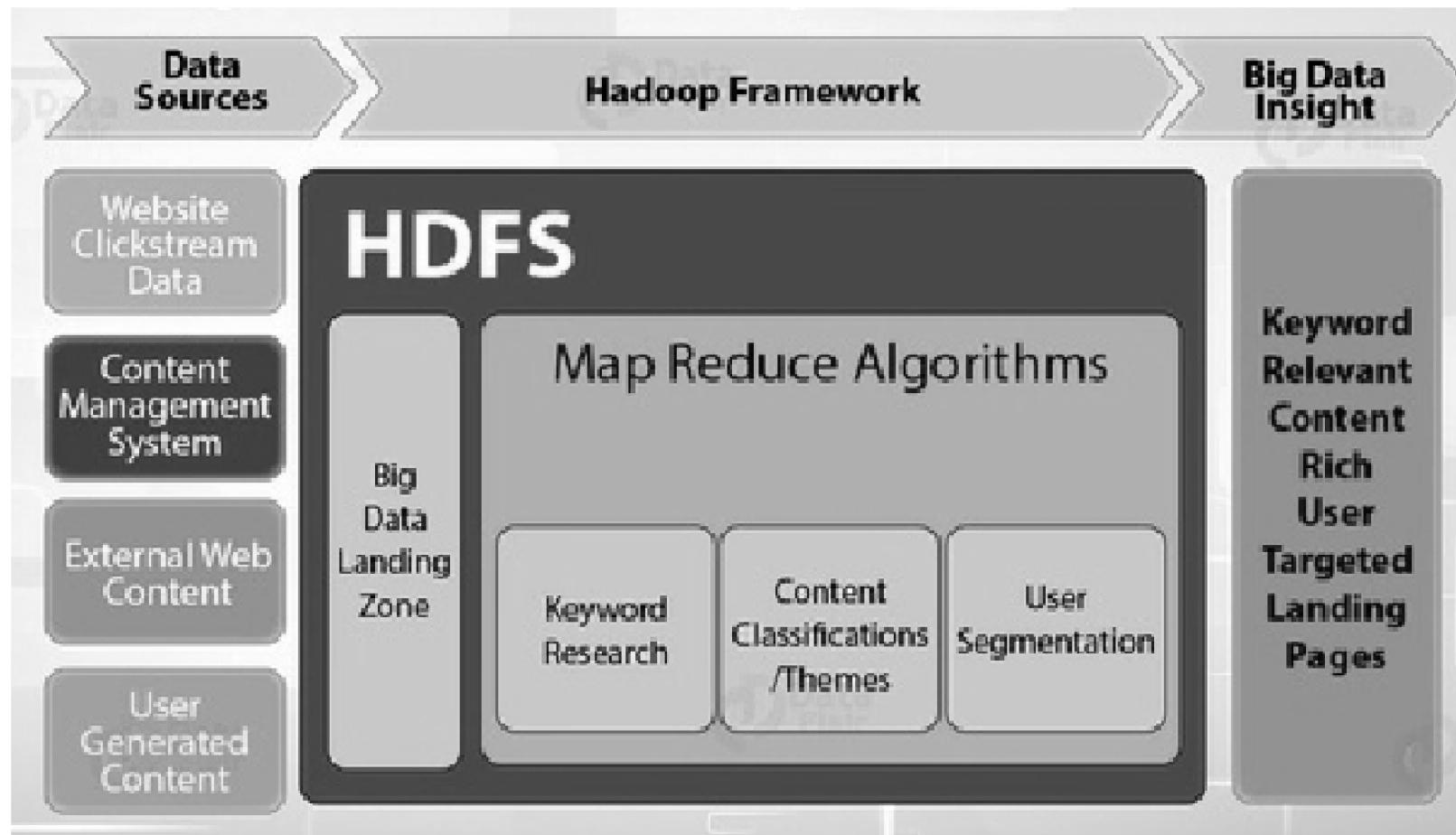


bazată pe Hadoop

Hadoop?

studenți sau profesori?

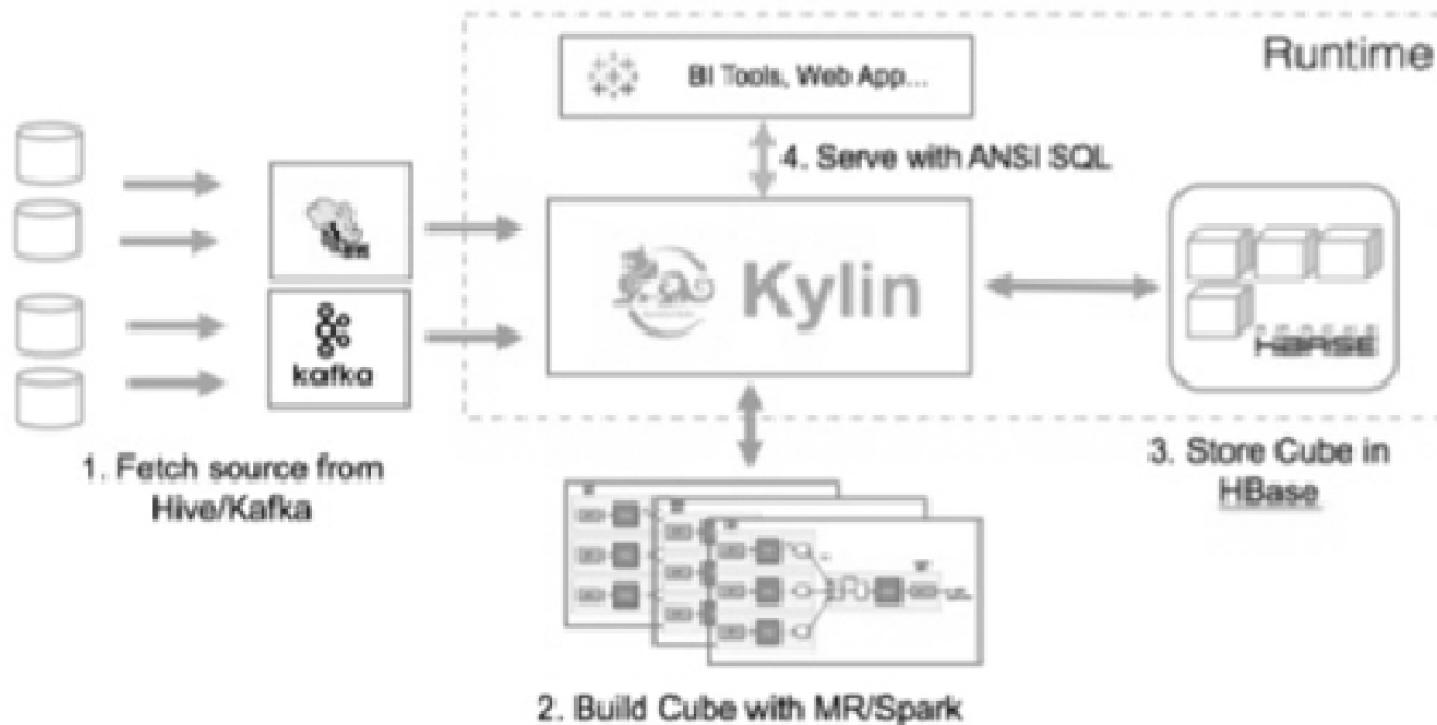
Hadoop



- X

Guvernarea datelor: Tactică sau Strategică?

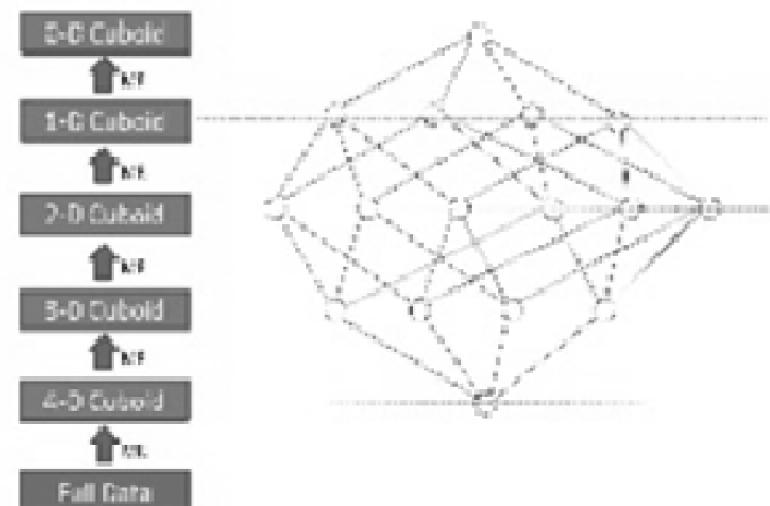
OLAP & Hadoop



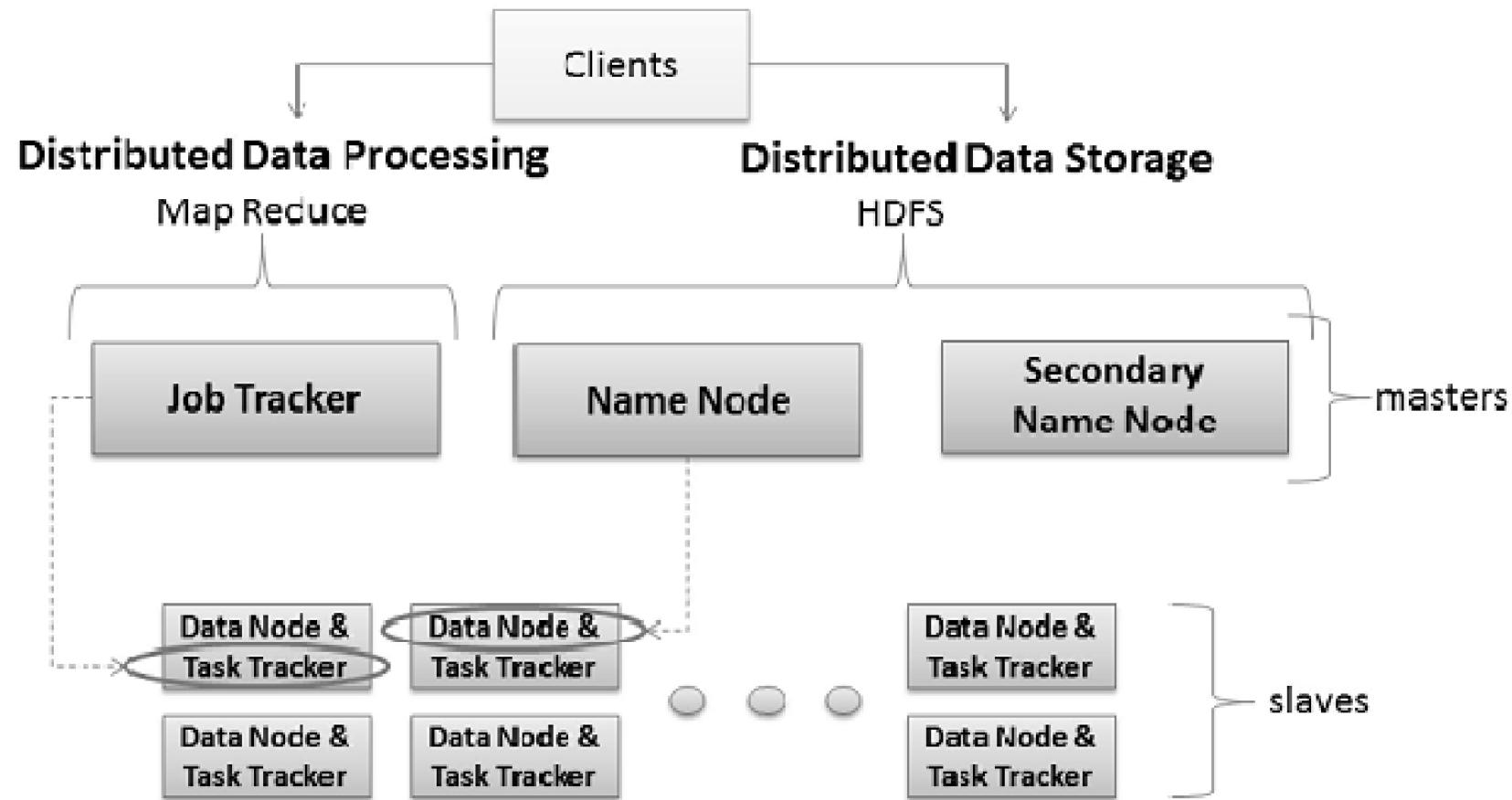
- un exemplu bazat pe kylin

OLAP & Hadoop

- Calculate Cuboids by layer : N dim (Base cuboid), N-1 dim, N-2..., 1, 0
- Reuse previous layer's result
- HDFS used for data sharing
- Totally need N round MR;
- generarea cubului cu mapreduce

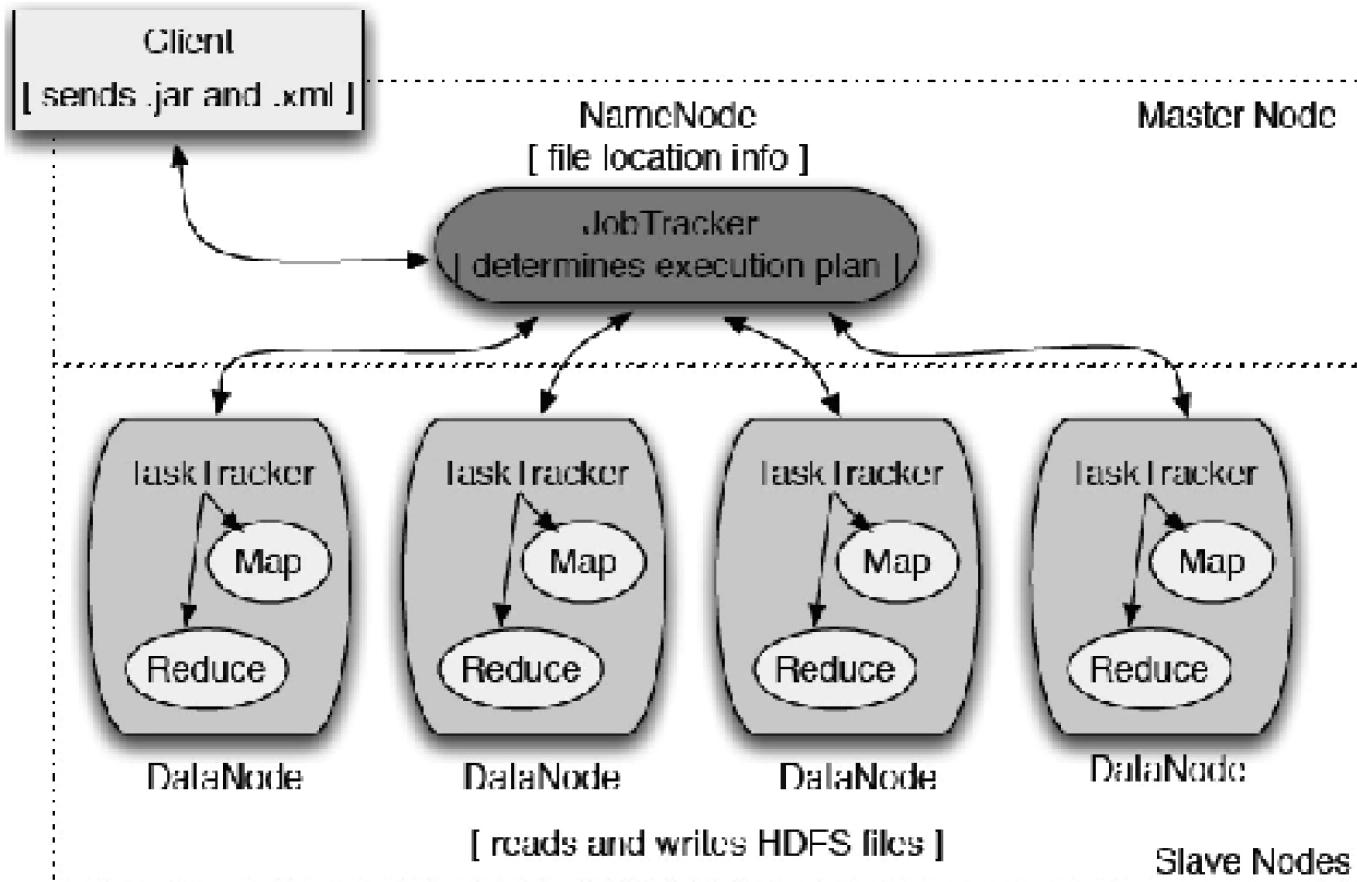


HDFS



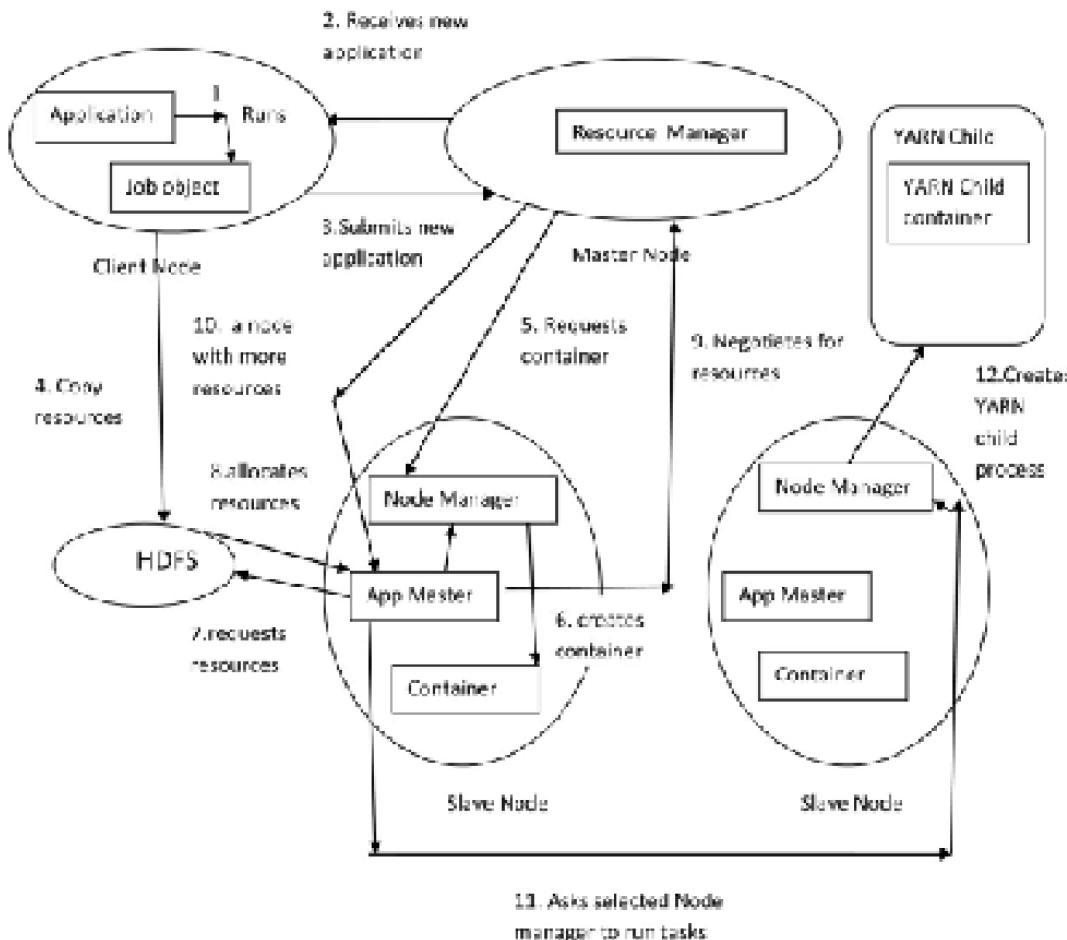
- roluri server in hadoop

MapReduce



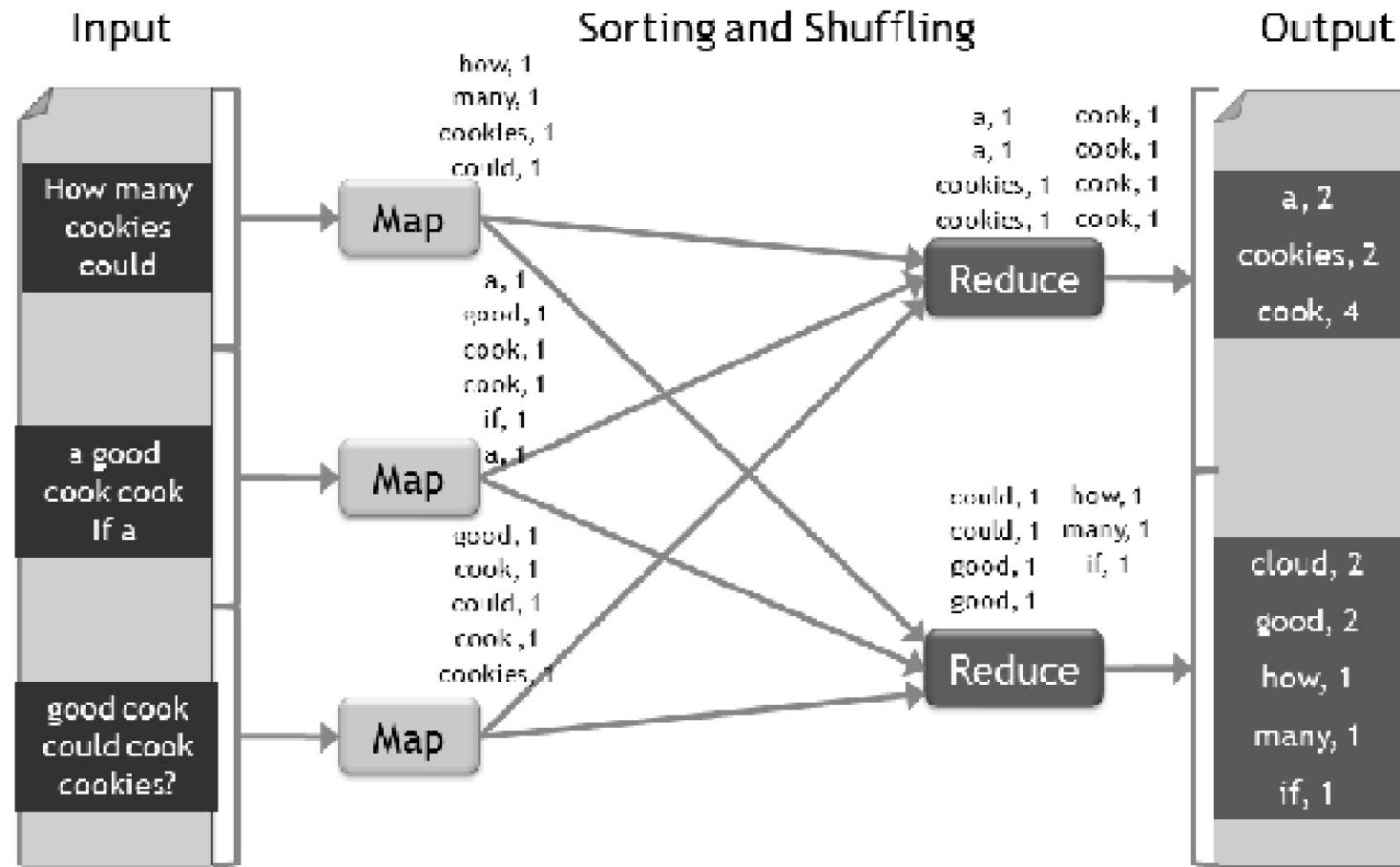
un instrument!

MapReduce



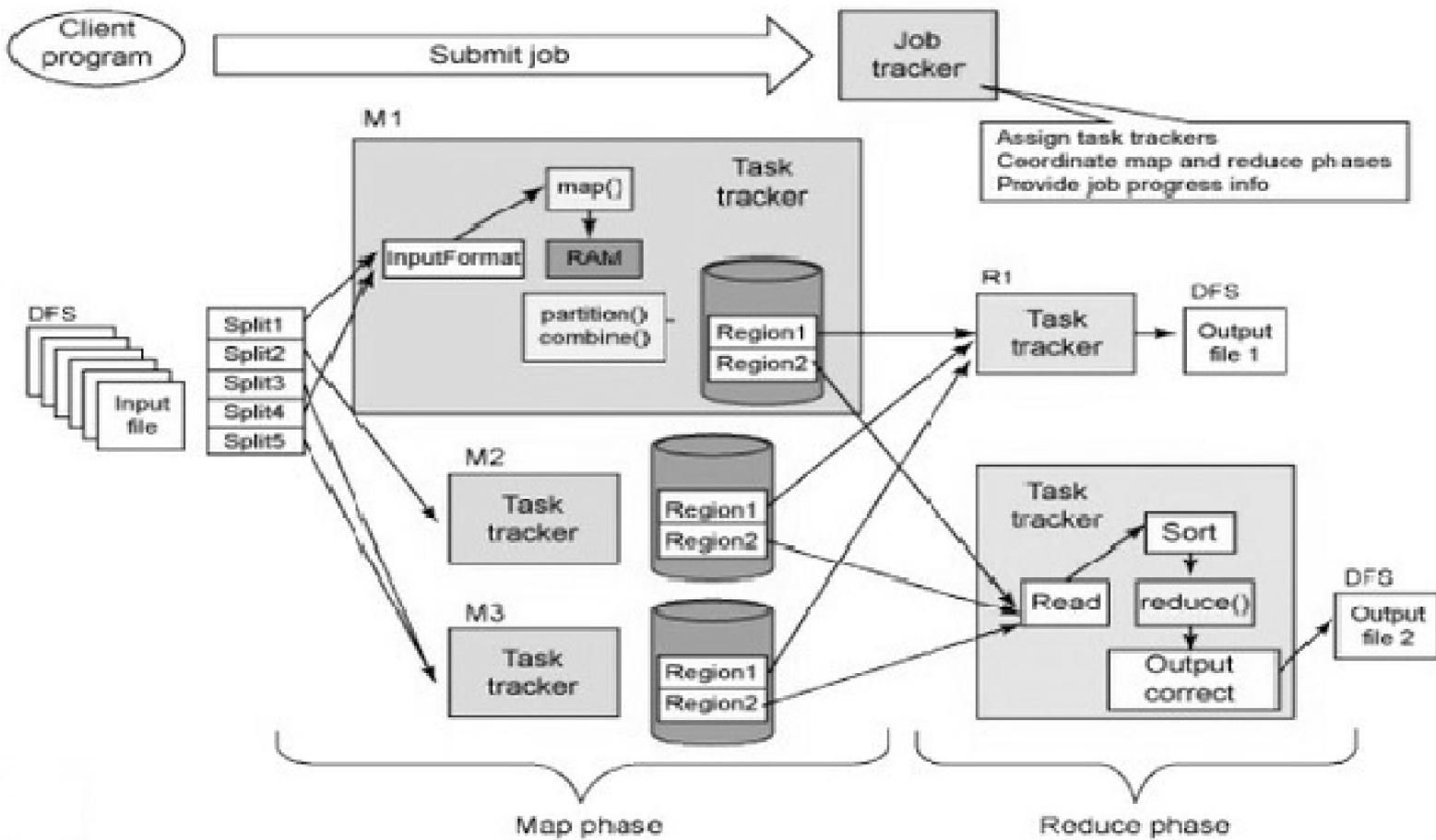
execuția unui job în map reduce

Map Reduce - exemplu



$\langle k1, v1 \rangle \rightarrow$ transformare $\rightarrow \langle k2, v2 \rangle \rightarrow$ reducere $\rightarrow \langle k3, v3 \rangle$

Map Reduce



Paşii

Hadoop sau ElasticSearch?

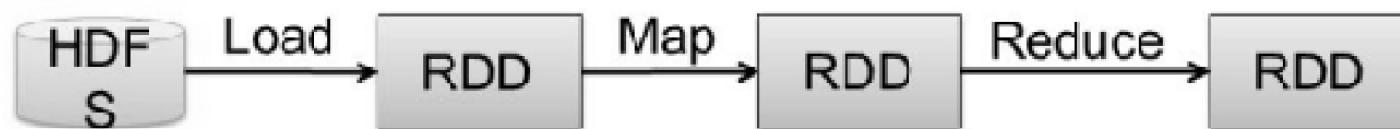
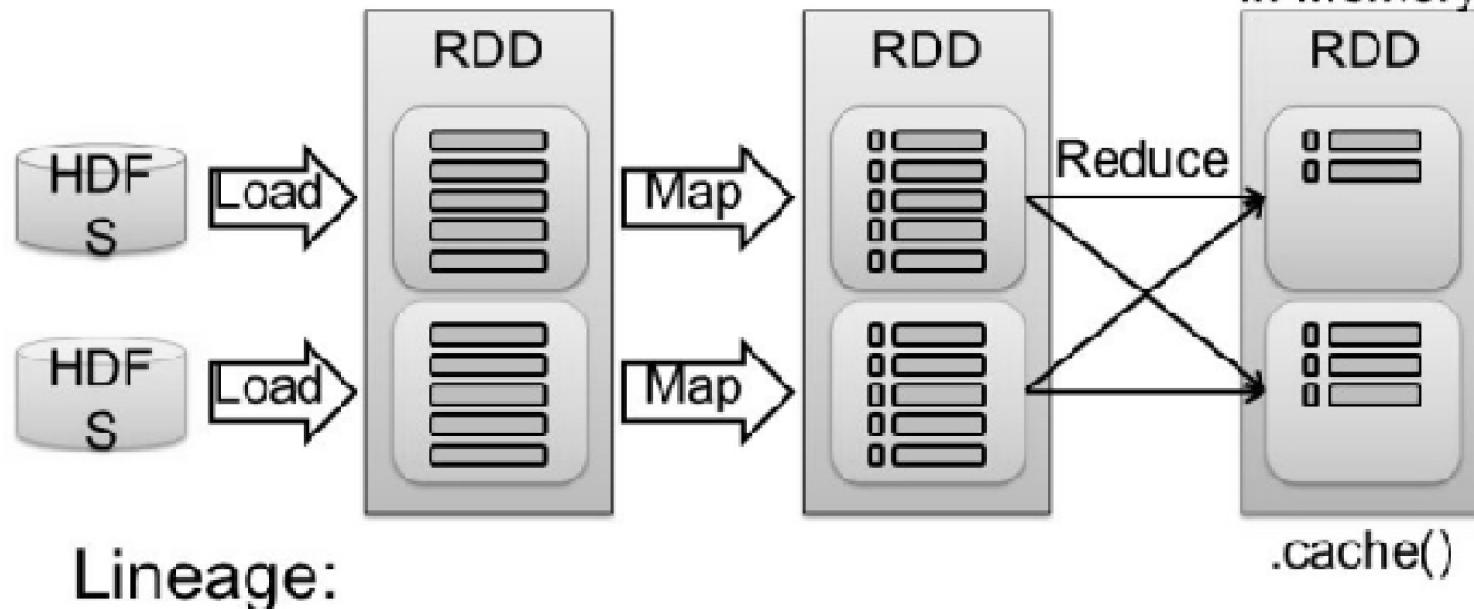
Hadoop

Spark

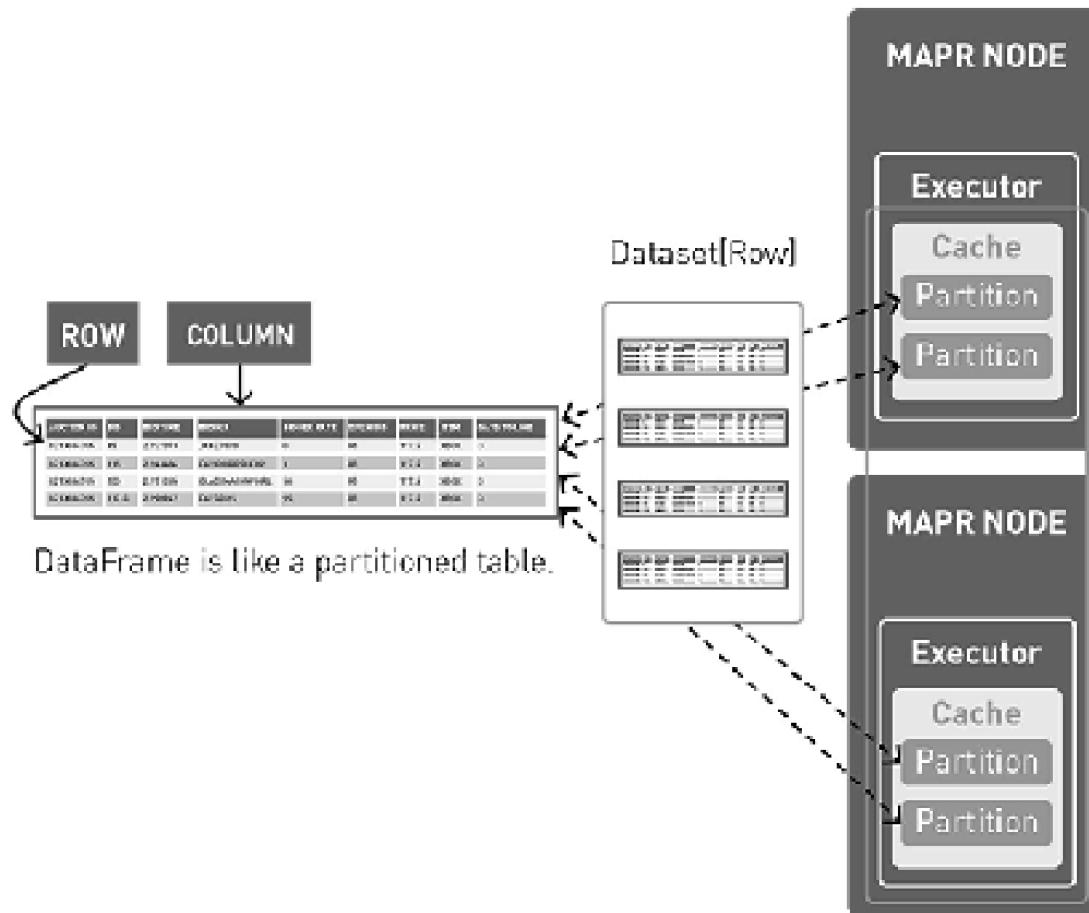
Zanaria et al., NSDI 12

Resilient Distributed Datasets (RDD):

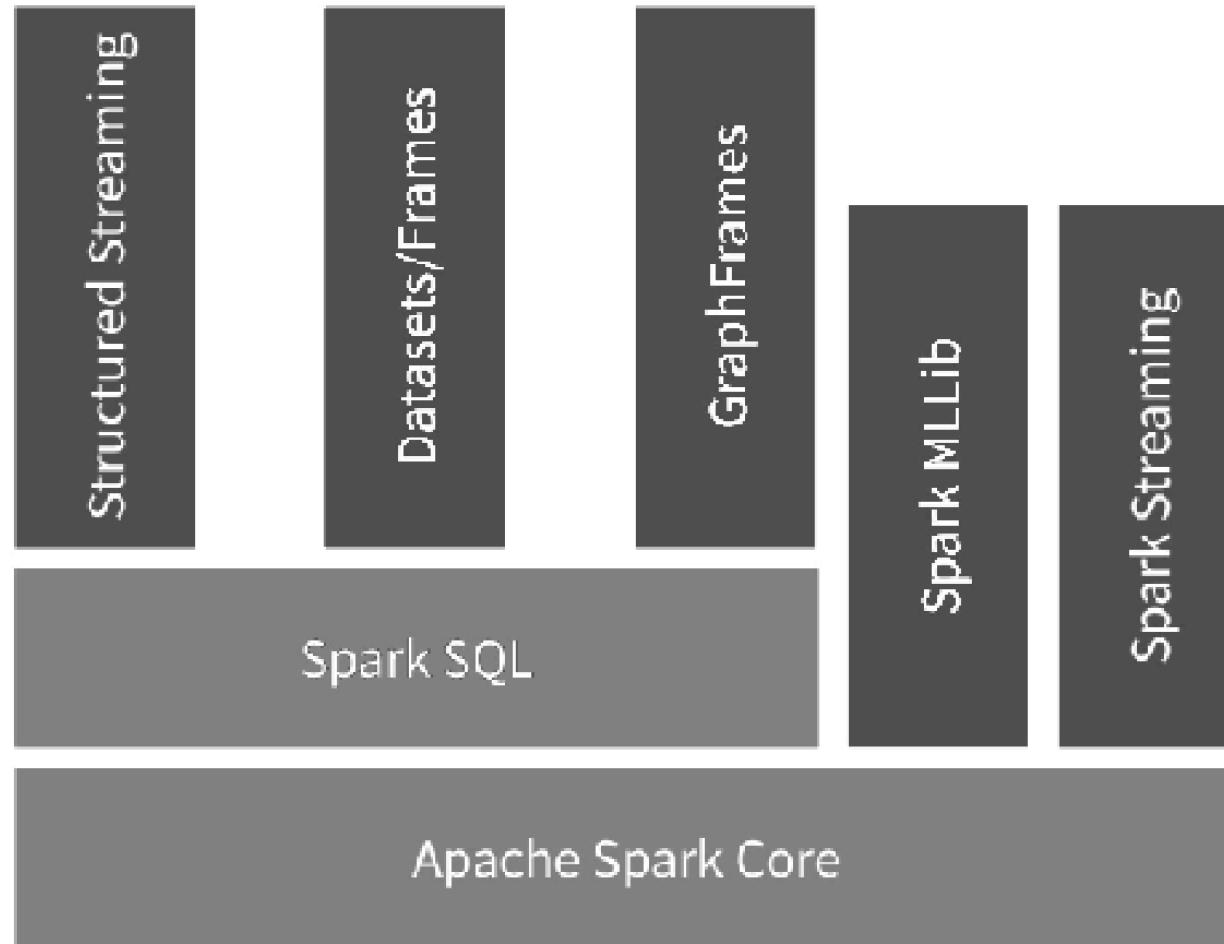
Persist
in Memory



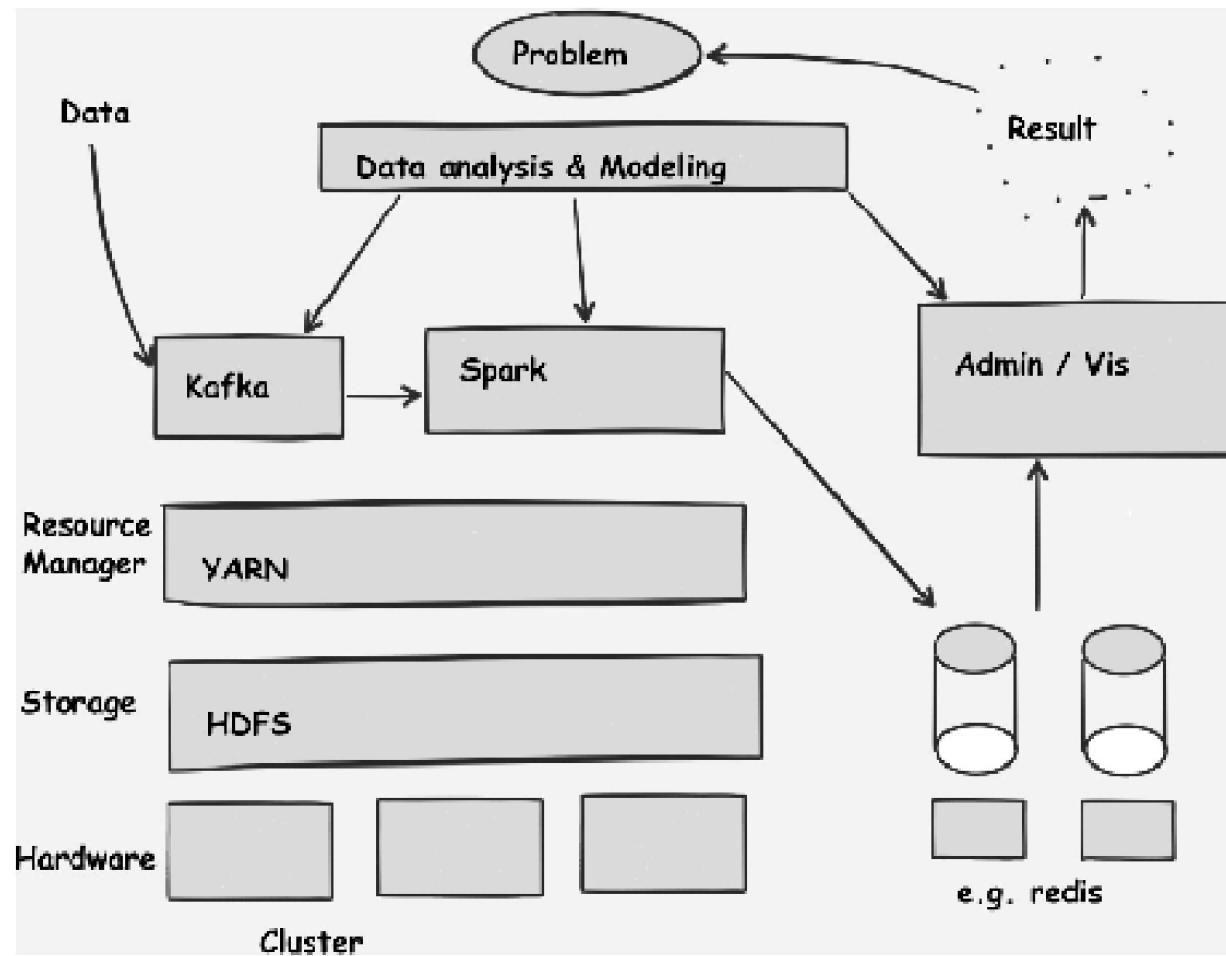
Spark SQL



Structura Spark

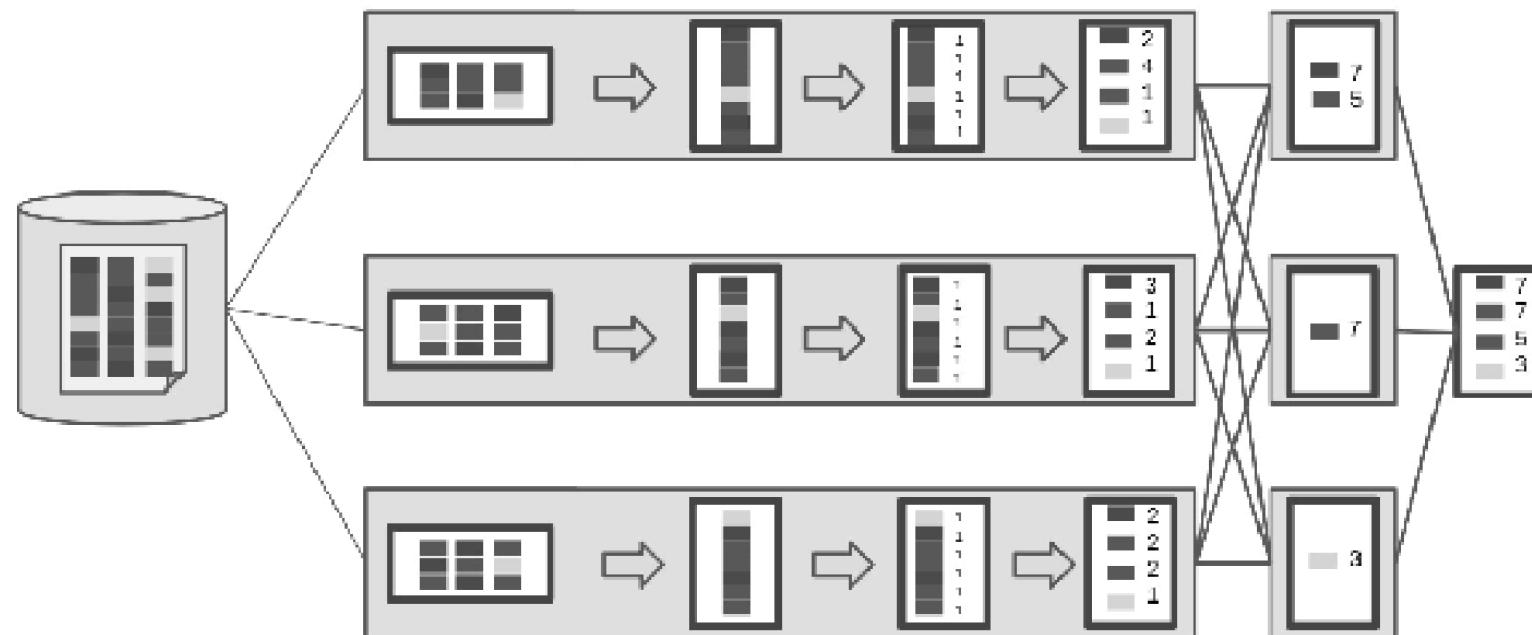


O posibilă platformă de date bazată pe Spark



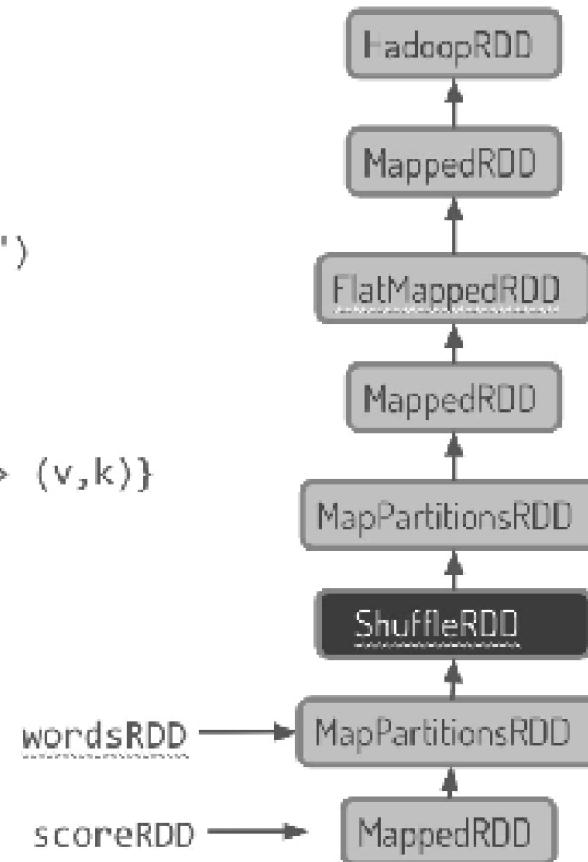
RDD

```
.textFile("...") .flatMap(l => l.split(" ")) .map(w => (w,1)) .reduceByKey(_ + _)
```

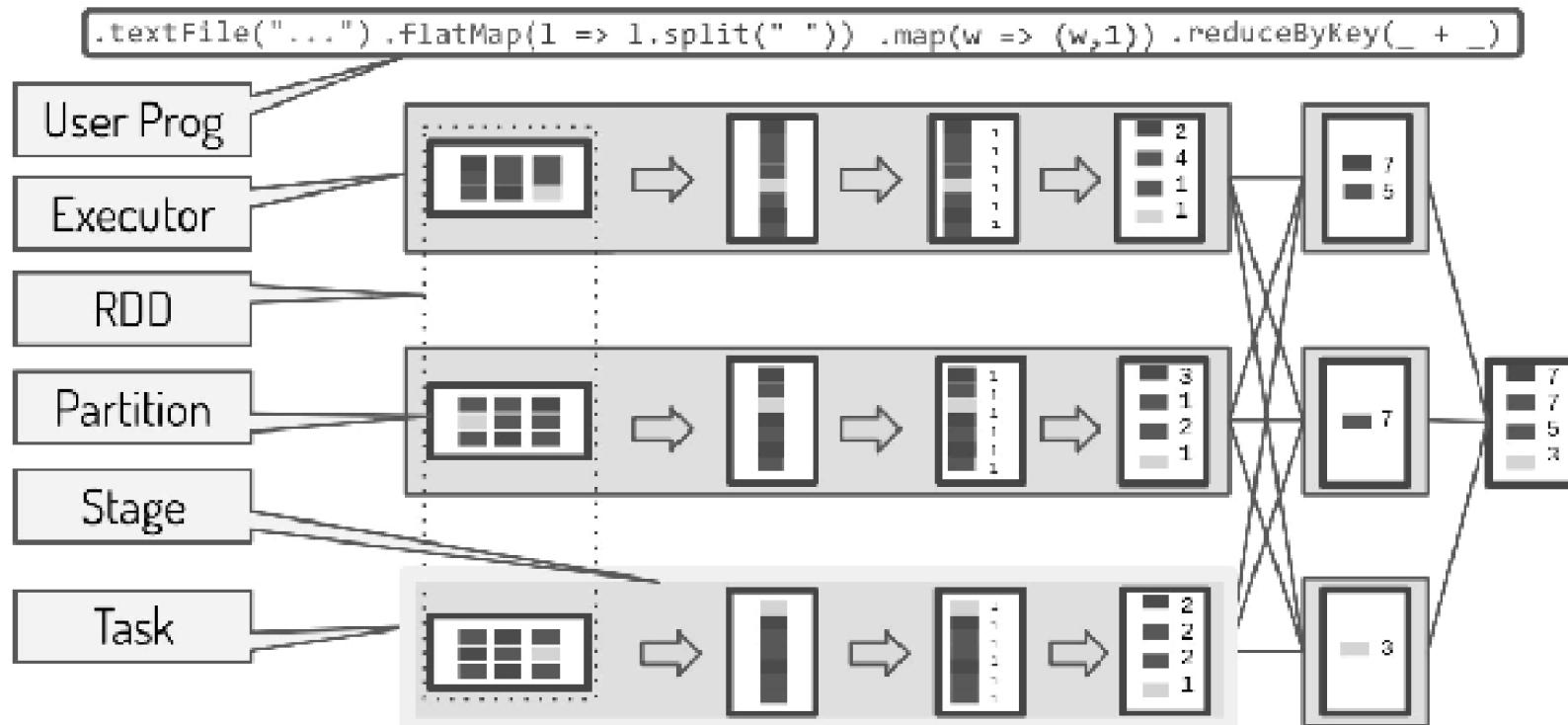


RDD & DAG

```
val file = spark.textFile("hdfs://...")  
val wordsRDD = file.flatMap(line =>  
    line.split(" ")).  
    .map(word => (word, 1))  
.reduceByKey(_ + _)  
val scoreRDD = words.map{case (k,v) => (v,k)}
```

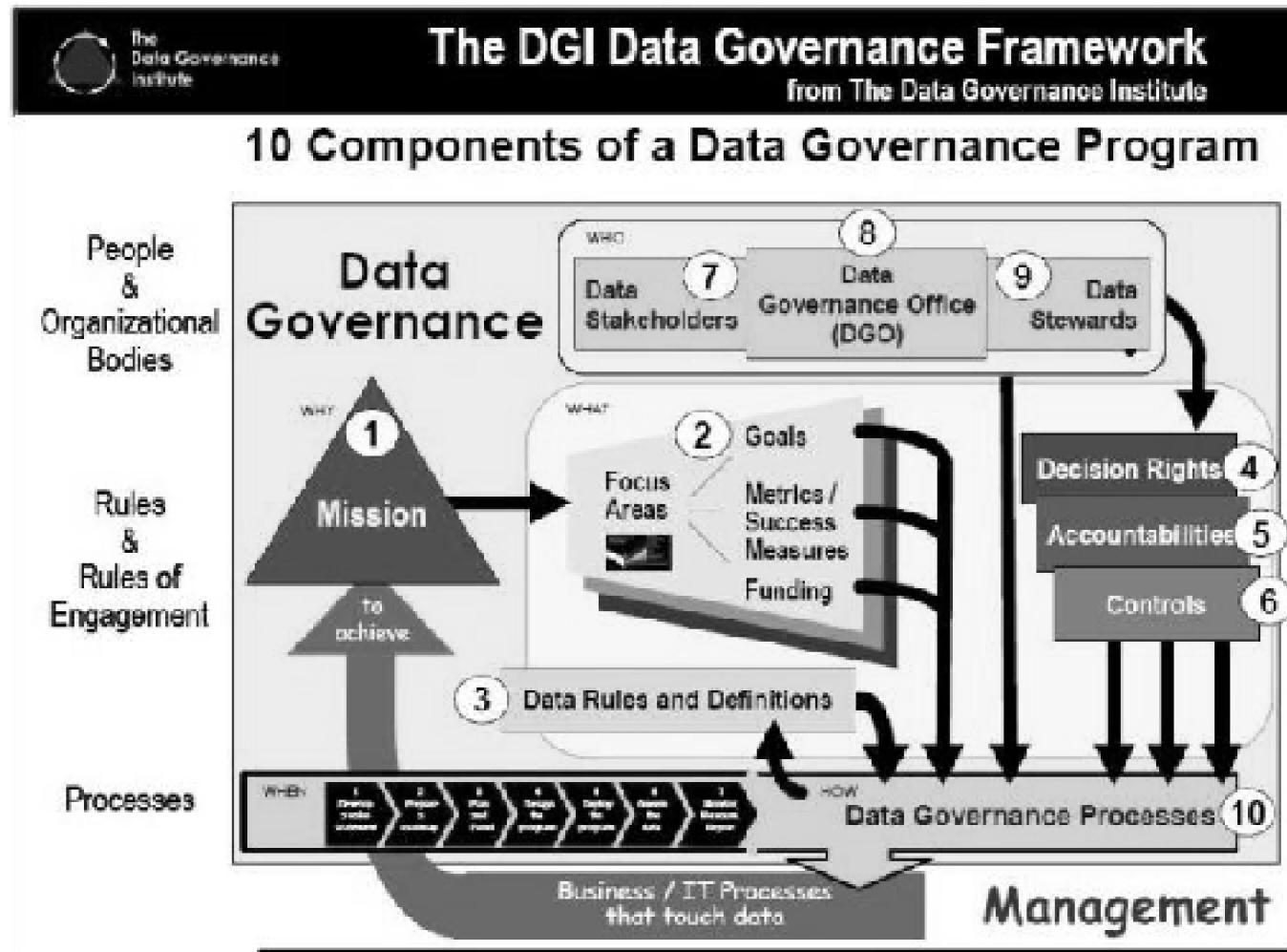


Spark Components



- X

Guvernarea datelor



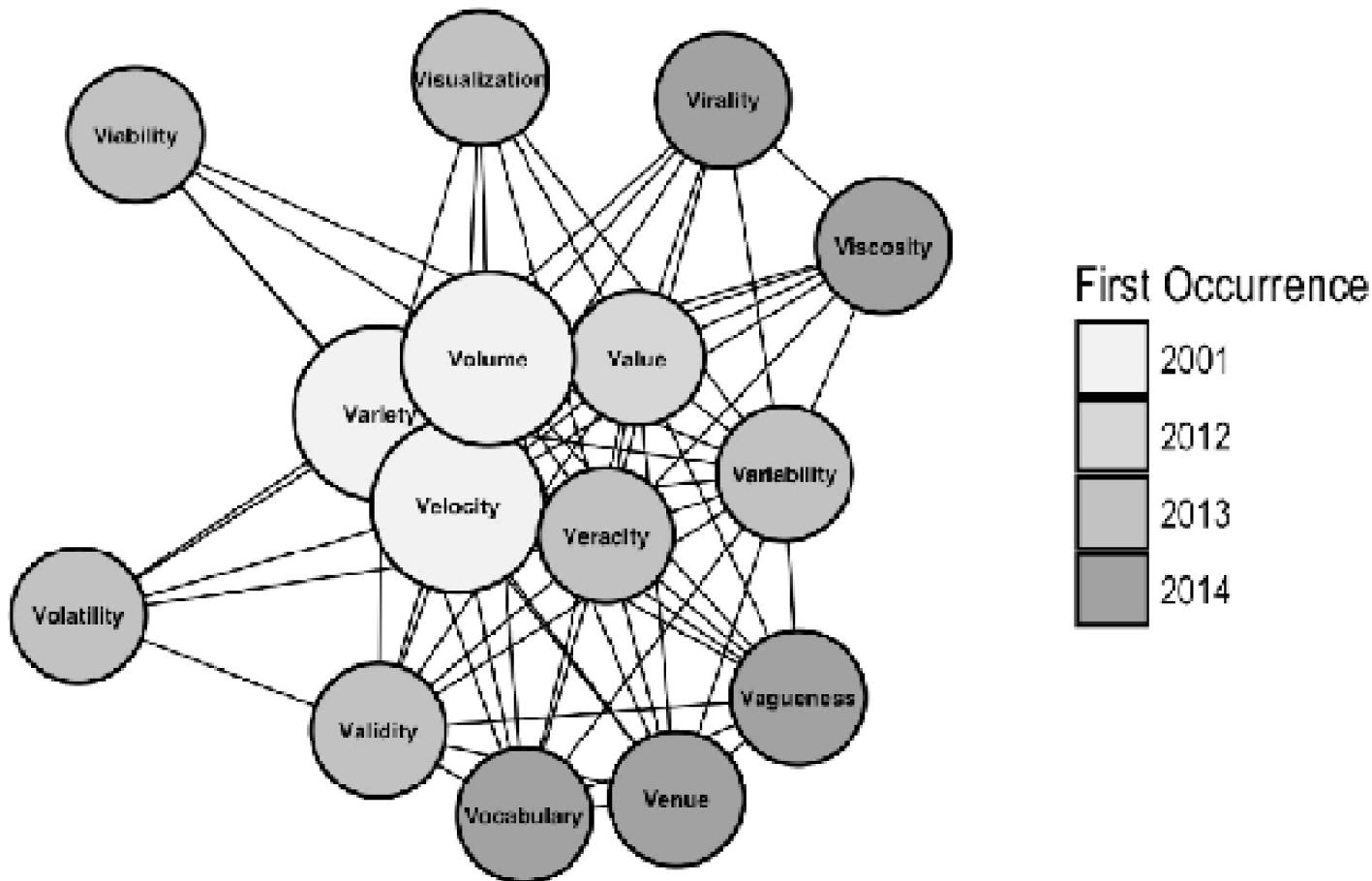
- X

Big Data

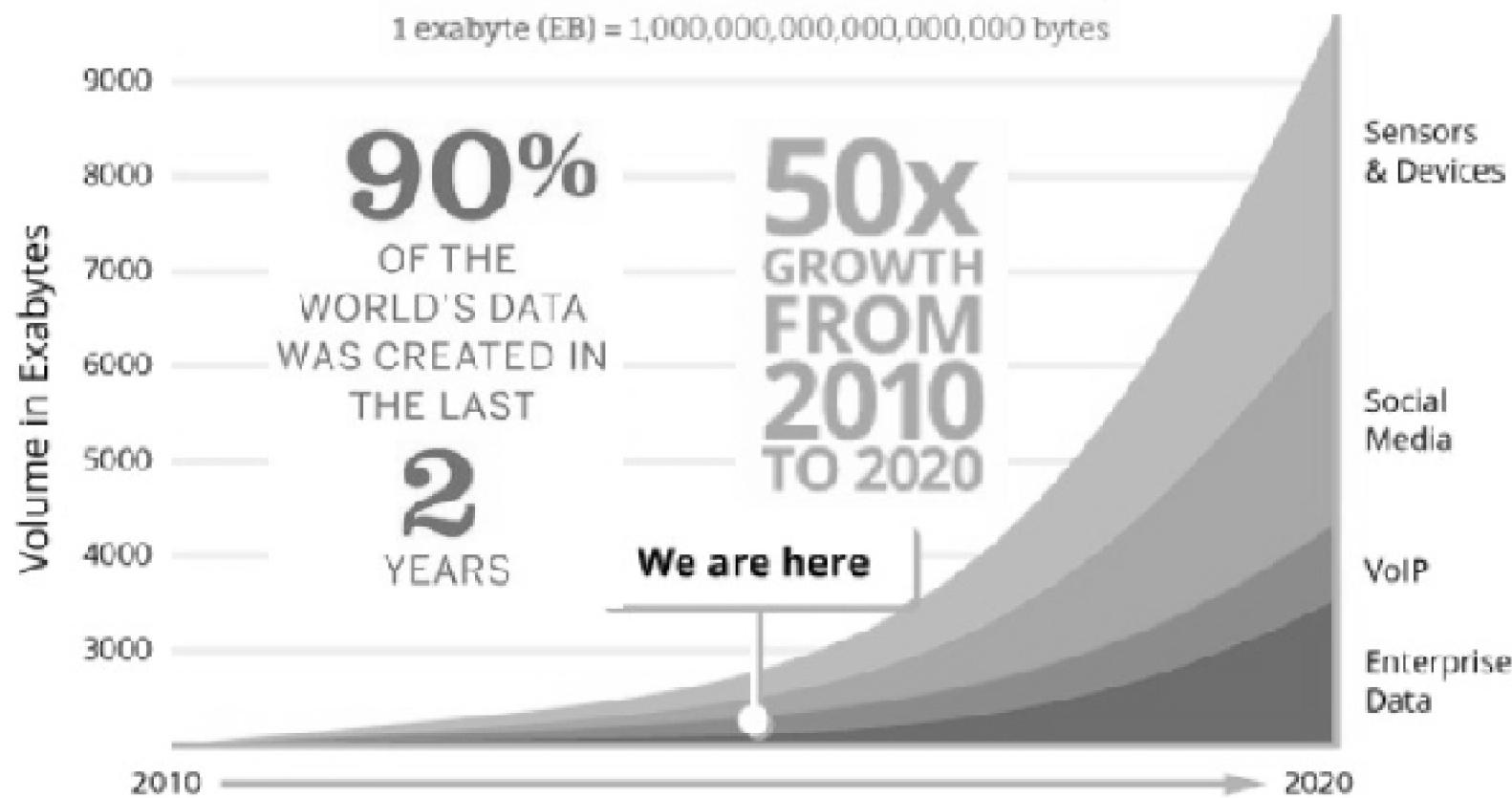
- *"Visualization provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of big data. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources."*

Cox si Ellsworth, IEEE, 1997

Dimensiunile big data

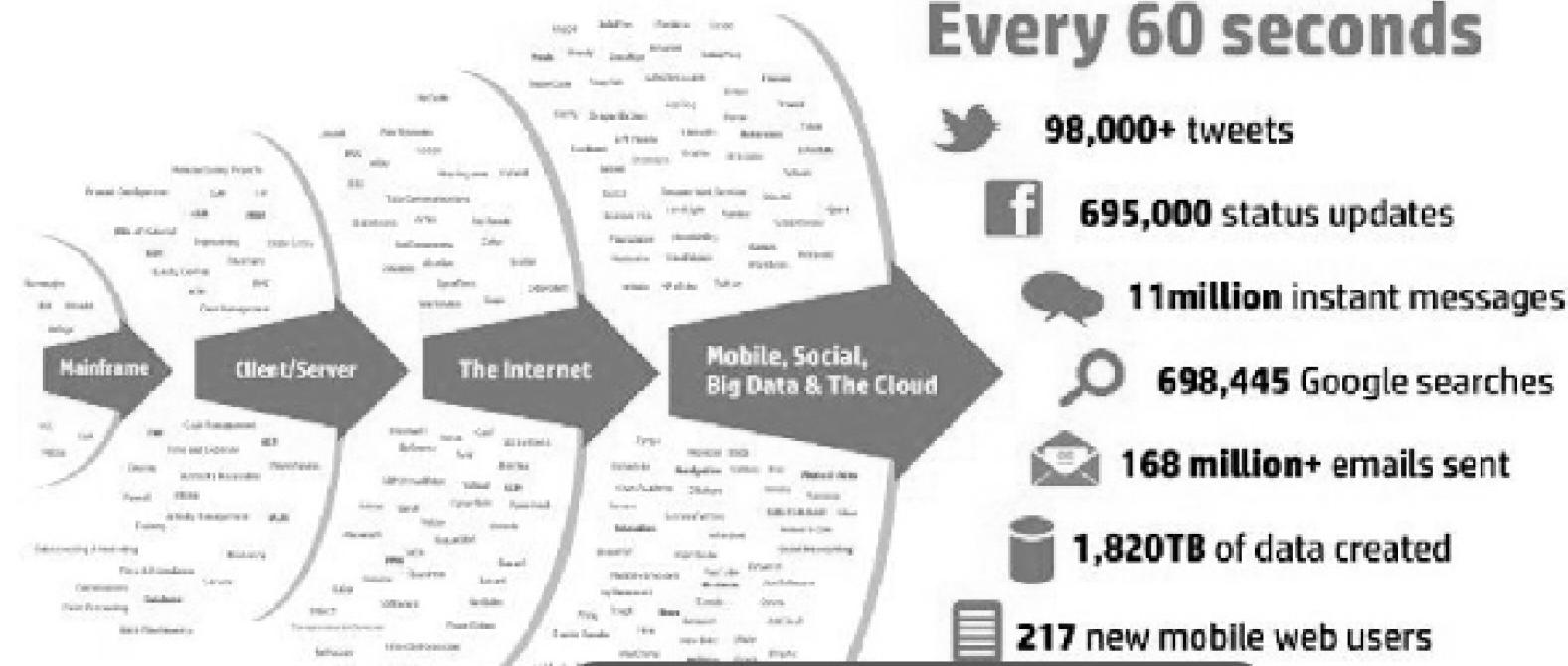


Dimensiunile Big Data 1. Volumul



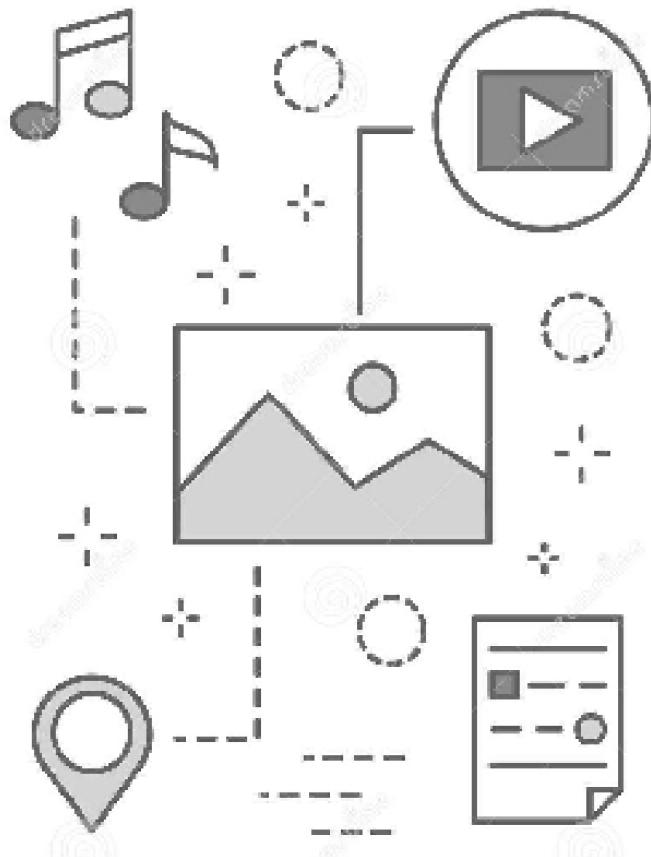
creștere exponențială a datelor generate

Dimensiunile Big Data 2. Velocitate



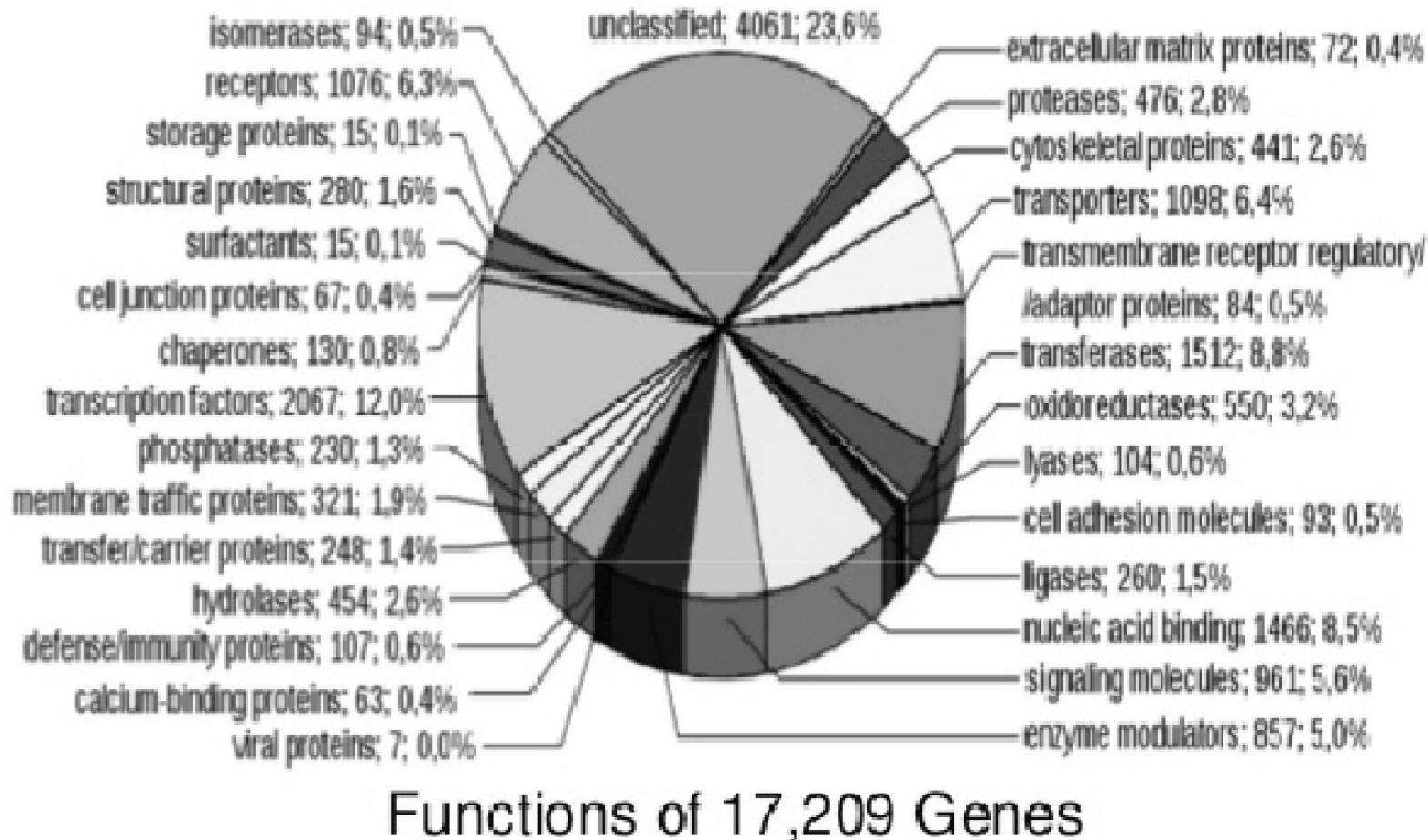
viteza de producție a noilor date

Dimensiunile Big Data 3. Varietate



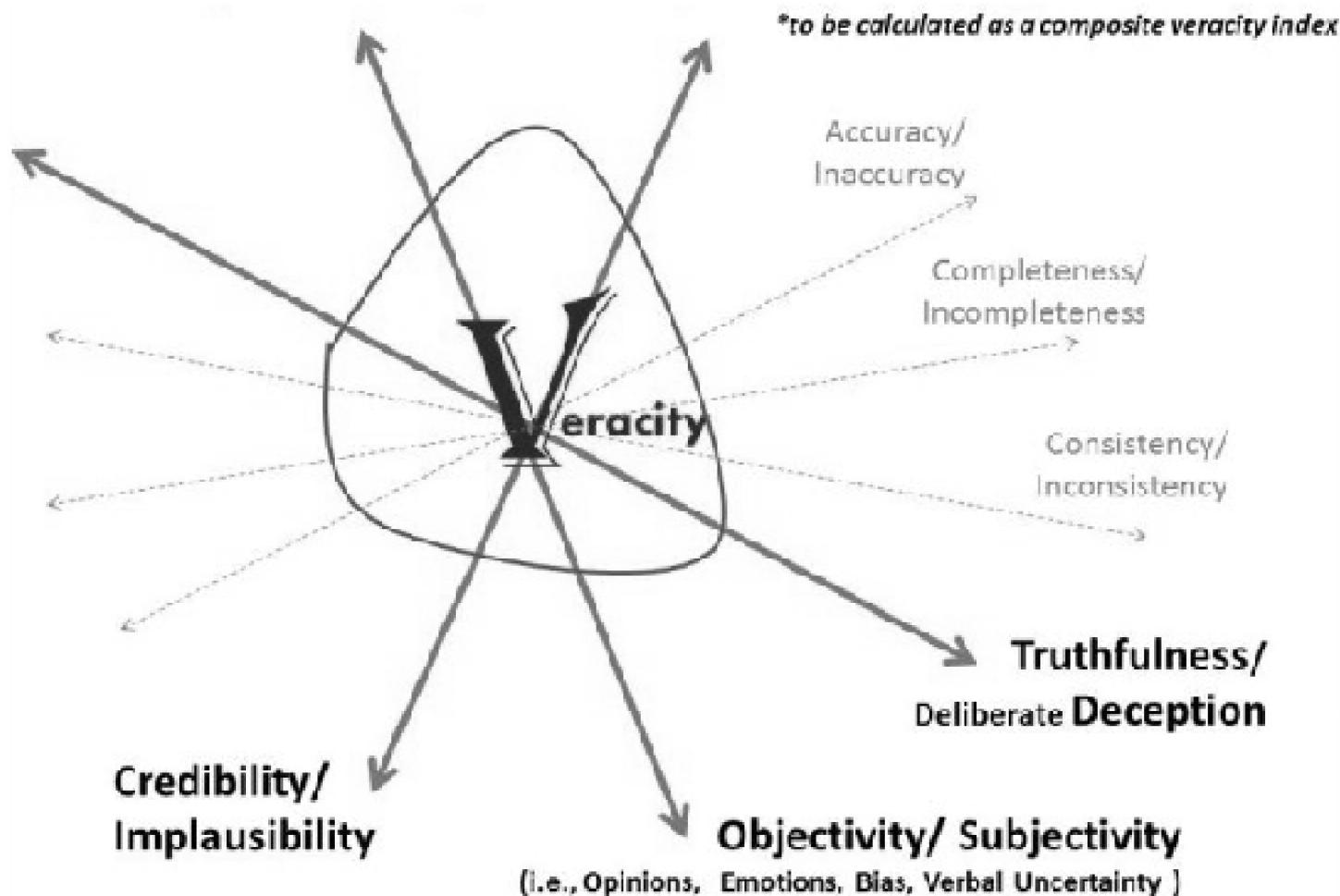
- Datele structurate
- Datele semistructurate
- Date nestructurate

Dimensiunile Big Data 4. Variabilitate



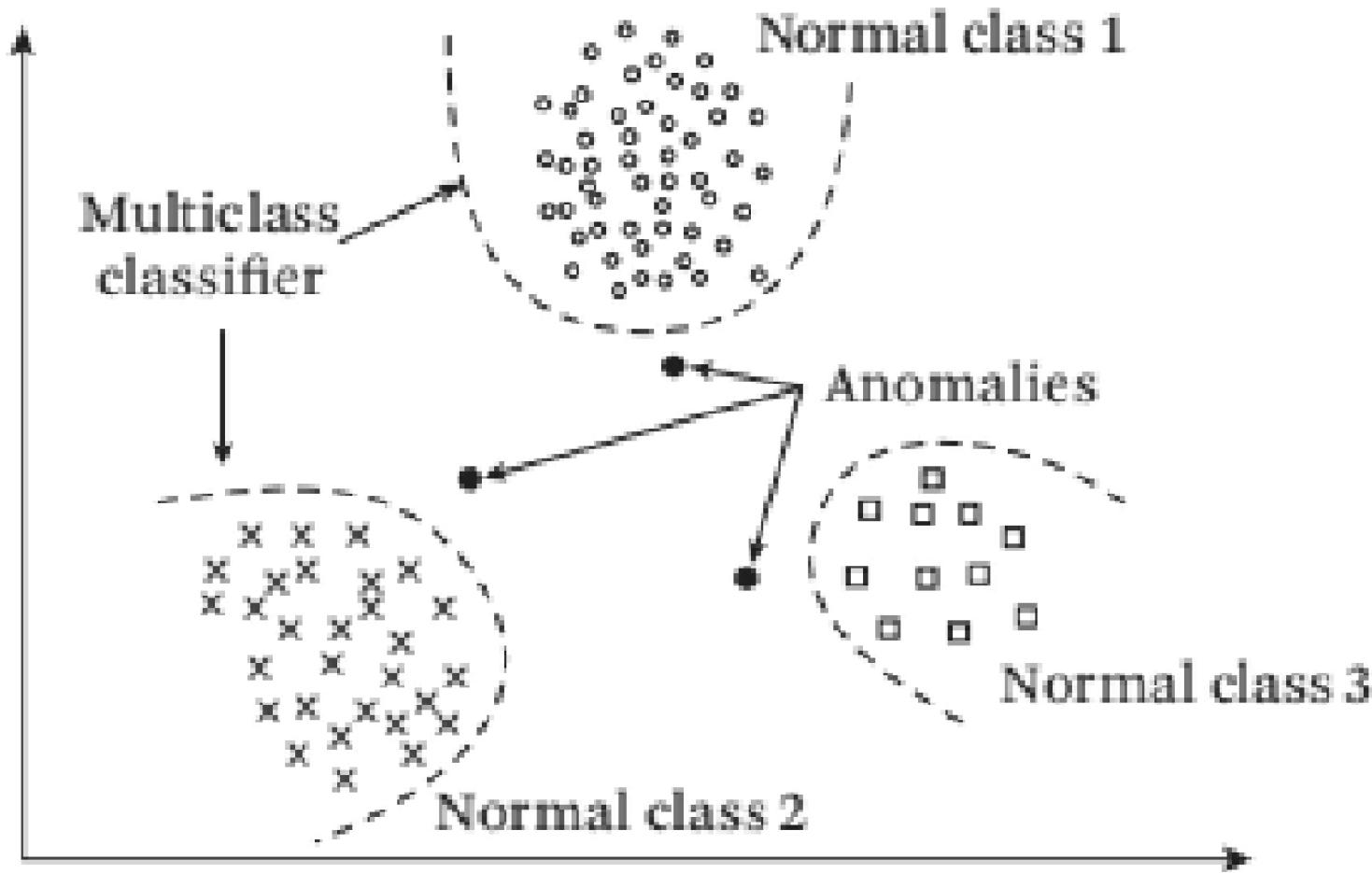
Neomogenitatea sau
Variatia dimensională și compozițională

Dimensiunile Big Data 5. Veracitate



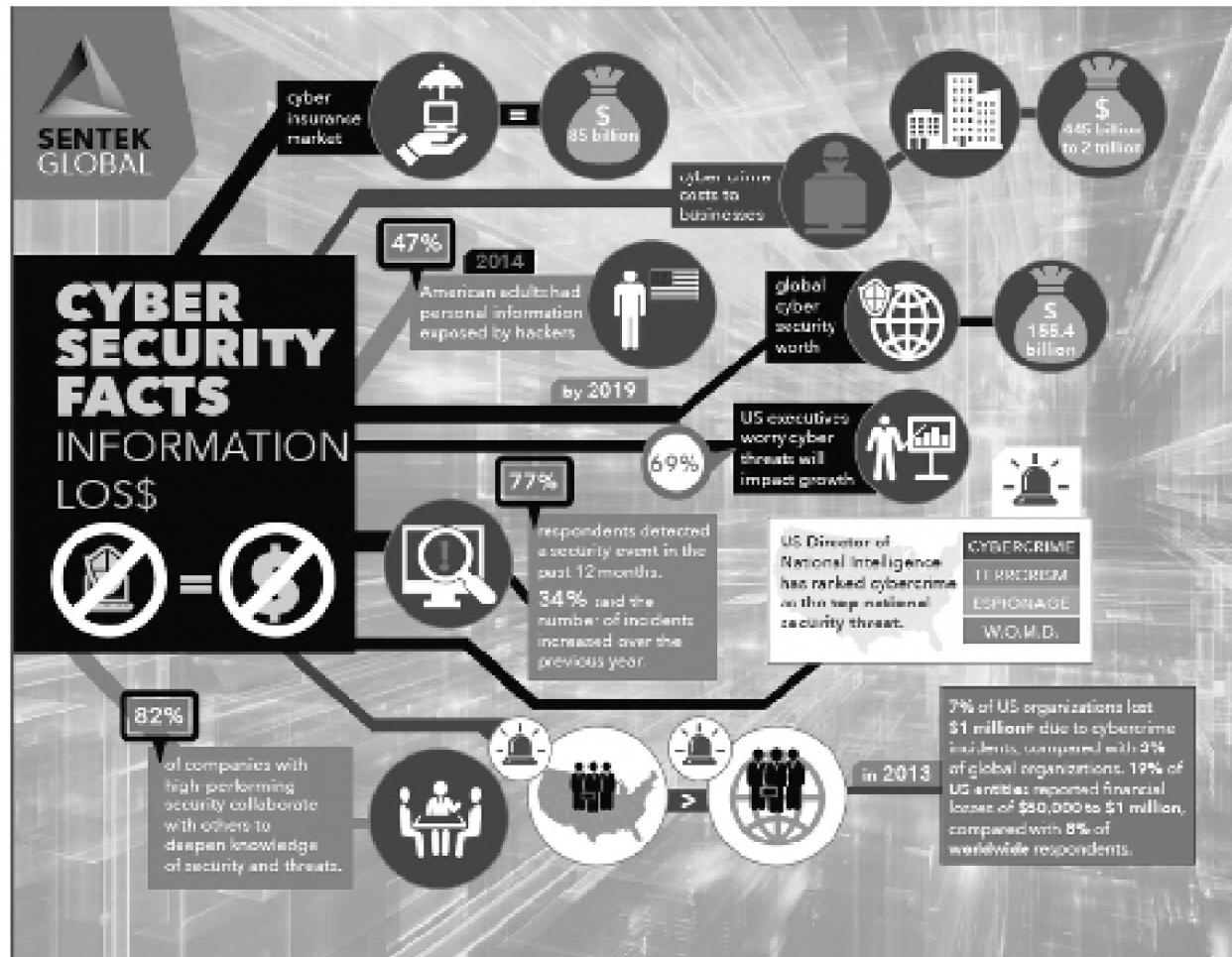
veriditate sau corectitudine

Dimensiunile Big Data 6. Validitatea



Detectarea anomaliilor după gruparea pe categorii

Dimensiunile Big Data 7. Vulnerabilitate



după Forbes

Sisteme distribuite

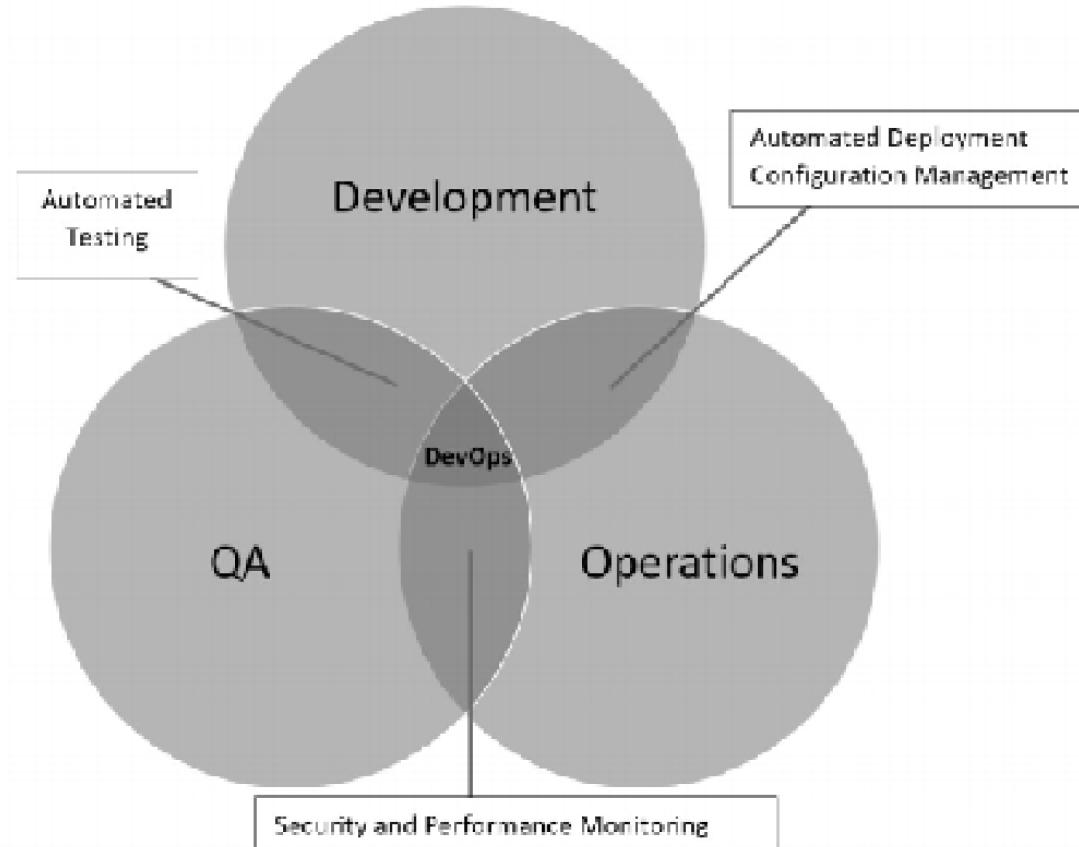
Mihai Zaharia

Cursul 13

Reziliența unei aplicații distribuite

- Aceasta descrie măsura în care un sistem/infrastructură digitală sau arhitectură a unei aplicații este capabilă să își mențină o corectă funcționare a serviciilor oferite
- refacerea automată
- Percepție greșită

DevOps

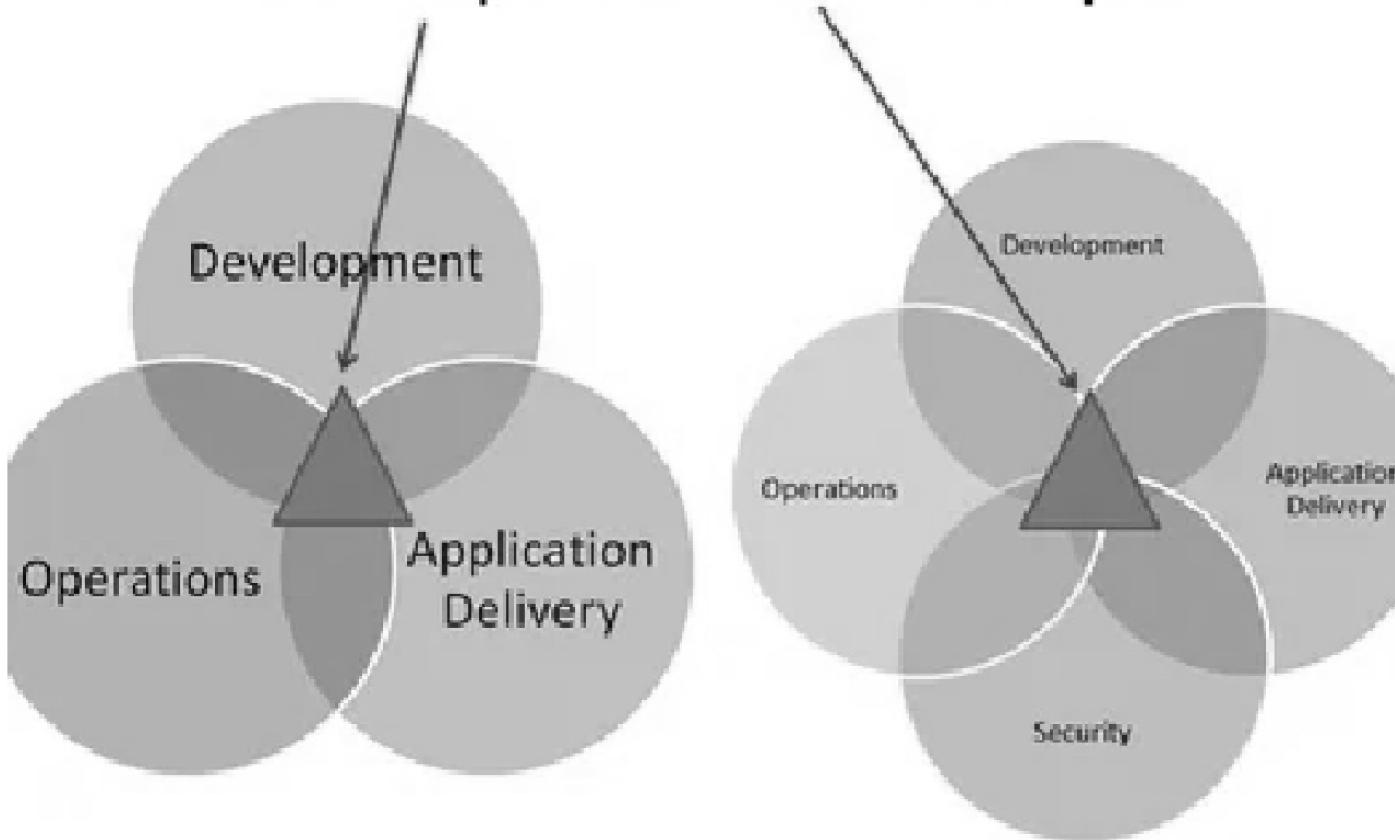


SecOps



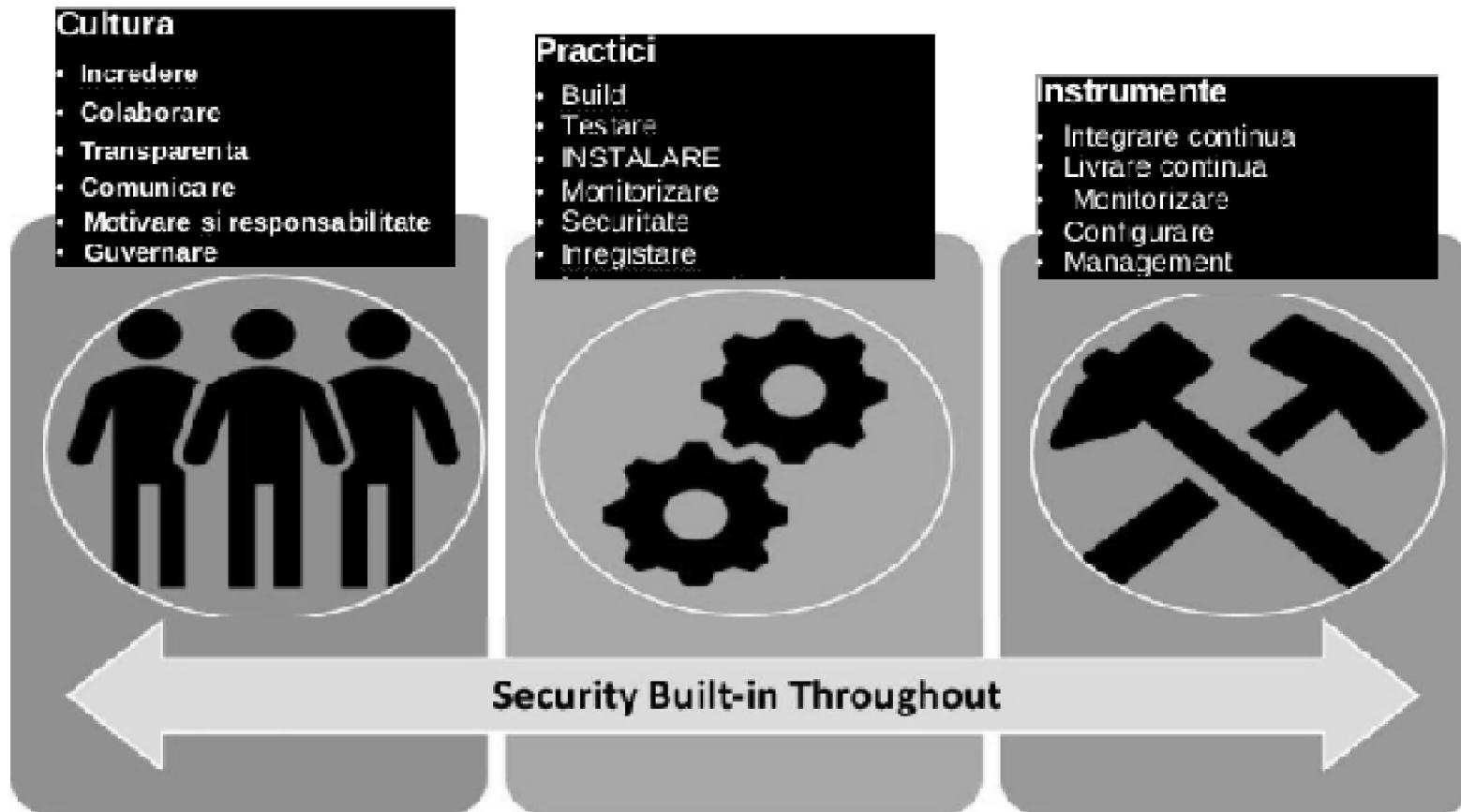
DevSecOps?

DevOps vs. DevSecOps



- X

Dimensiunile DevSecOps

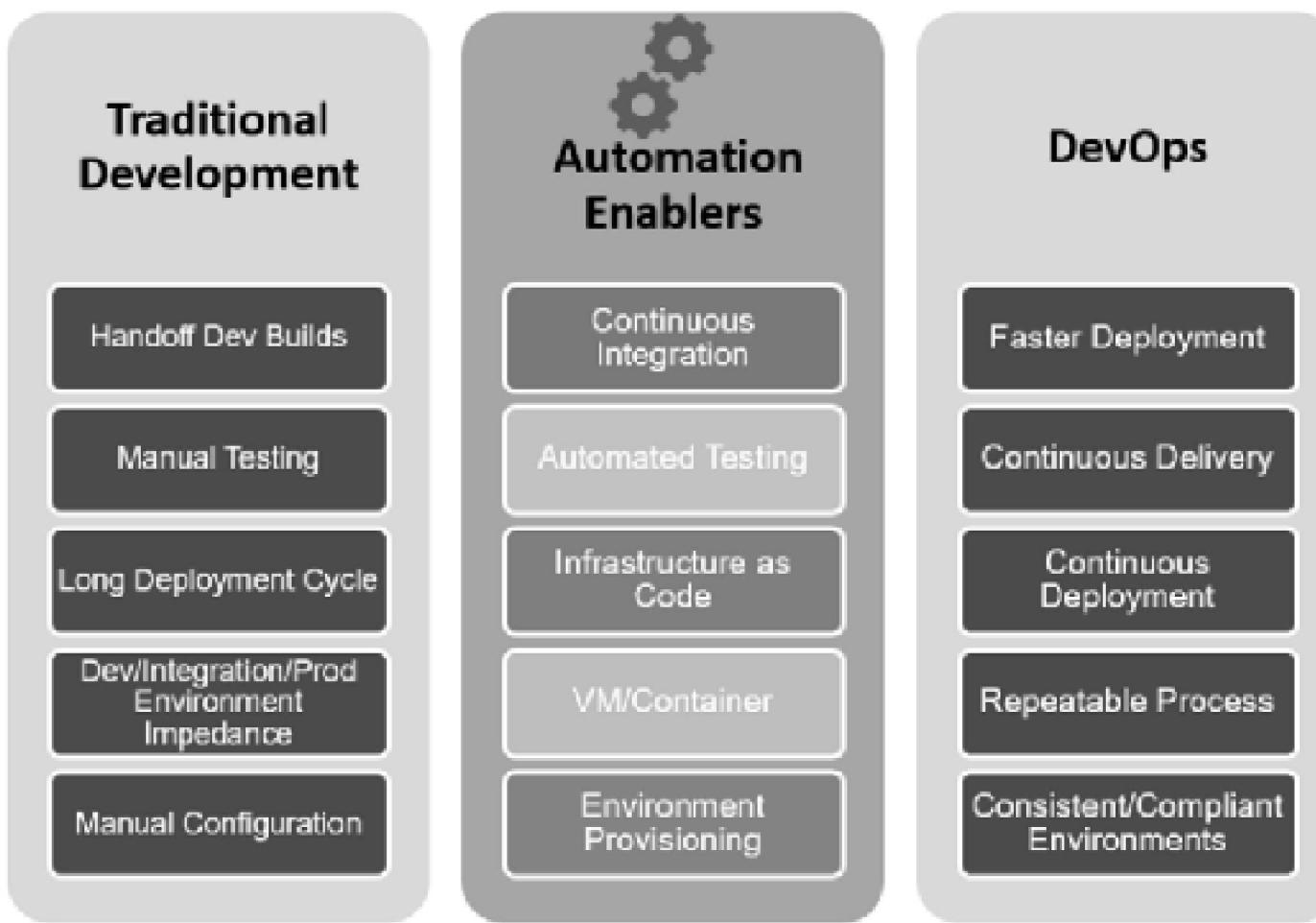


- X

De ce ar fi mai bun DevSecOps?

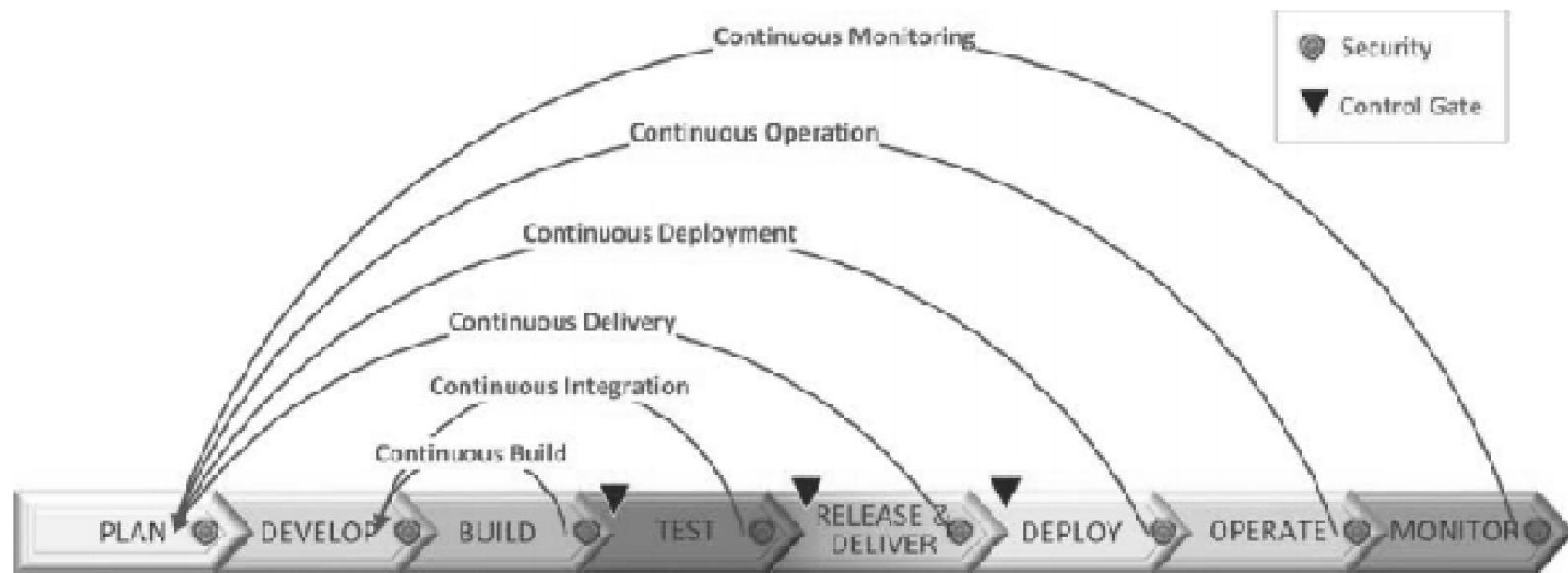
Problemele specifice ciclului clasic de dezvoltare	Beneficiile oferite de catre DevSecOps
un proces manual repetitiv	configurarea și instalarea automată a aplicației
dezvoltarea durează zile sau săptămâni	instalarea durează câteva minute
nu este repetabil și este supus erorilor	este un proces continuu și repetabil
intervenția umană conduce la inconsistențe	rezultatul este consistent
timpi de cădere/pierduți frecvenți	timpi de cădere minimi
este mai ușoară - trebuie oameni mai puțin antrenați	mai complicat trebuie experți
echipile lucrează în silozuri	colaborare continuă între echipe/dept etc
testarea incipientă/primară de securitate nu este efectuată asupra codului	testare de securitate automată încă din faza de scriere a codului

De ce ar fi mai bun DevSecOps?



- X

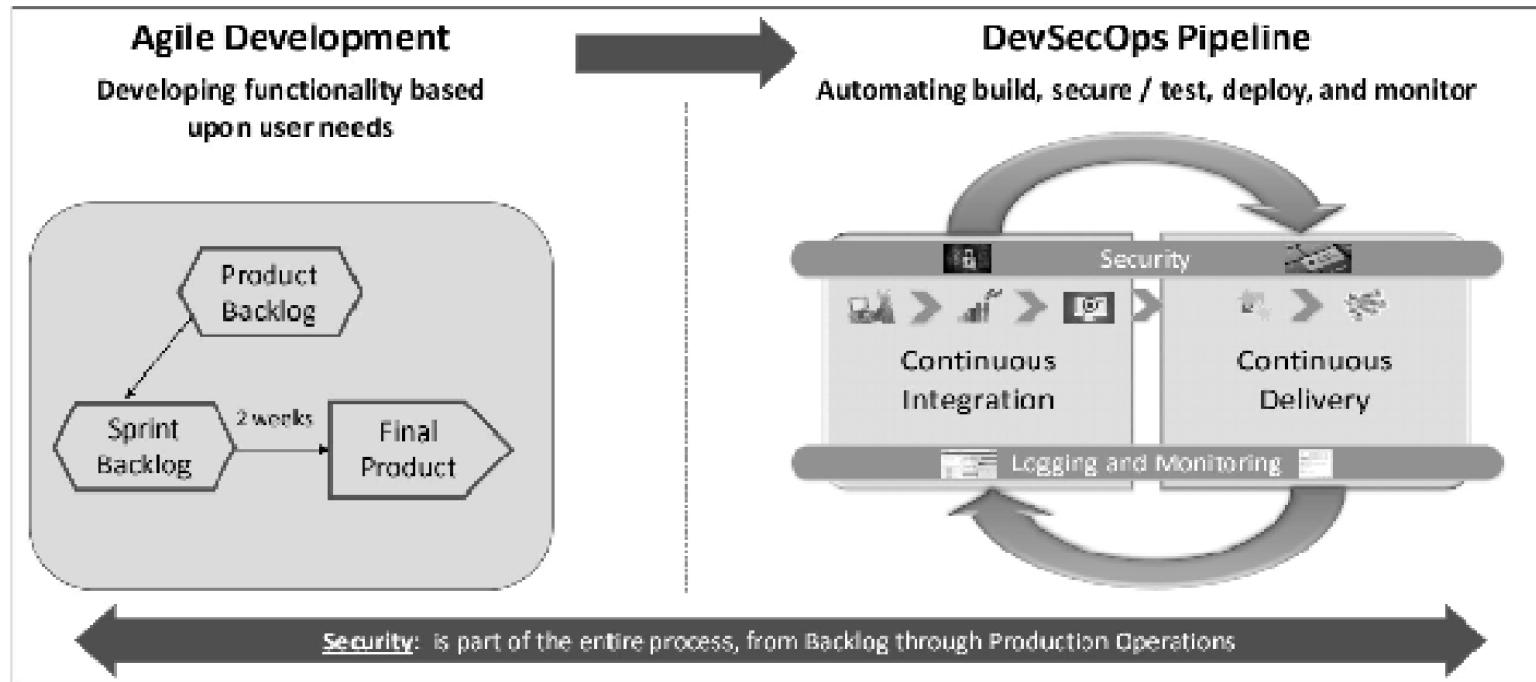
Cum se aplică ?



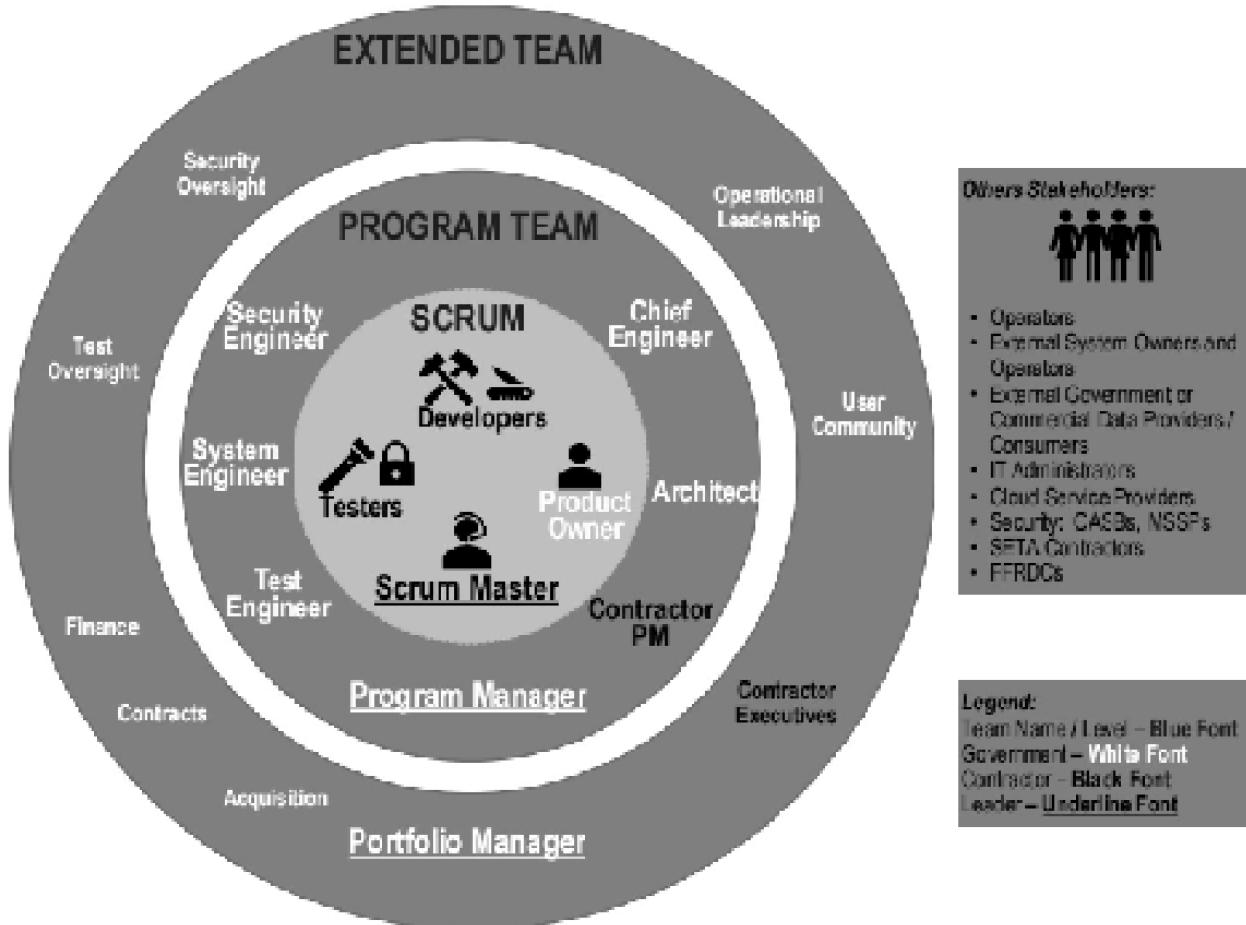
Securitate vs. Testare

Securitate	Testare
<ul style="list-style-type: none">• procesele de securitate	<ul style="list-style-type: none">• evenimentele din teste• mediile de testare
<ul style="list-style-type: none">• instrumentele de securitate	<ul style="list-style-type: none">• instrumentele de testare
<ul style="list-style-type: none">• securitatea accesului (de ex, la mediile DevSecOps)	<ul style="list-style-type: none">• testarea accesului (de ex la mediul DevSecOps)
<ul style="list-style-type: none">• Vizibilitatea instrumentelor de/pentru securitate (de ex, de-a lungul lantului de dezvoltare (pipeline))	<ul style="list-style-type: none">• datele de test
<ul style="list-style-type: none">• rapoartele de securitate	<ul style="list-style-type: none">• raportarea testarii

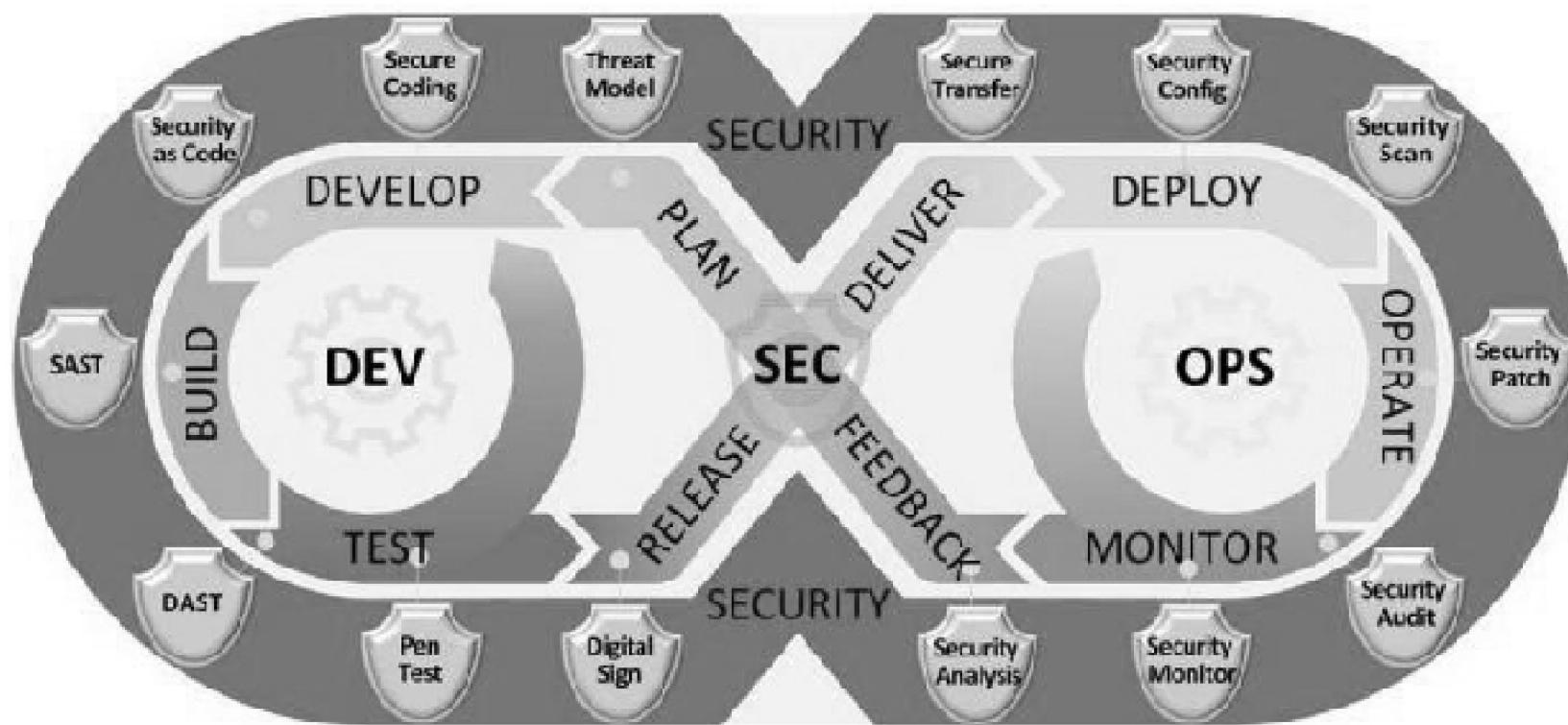
Agile + DevSecOps Pipeline



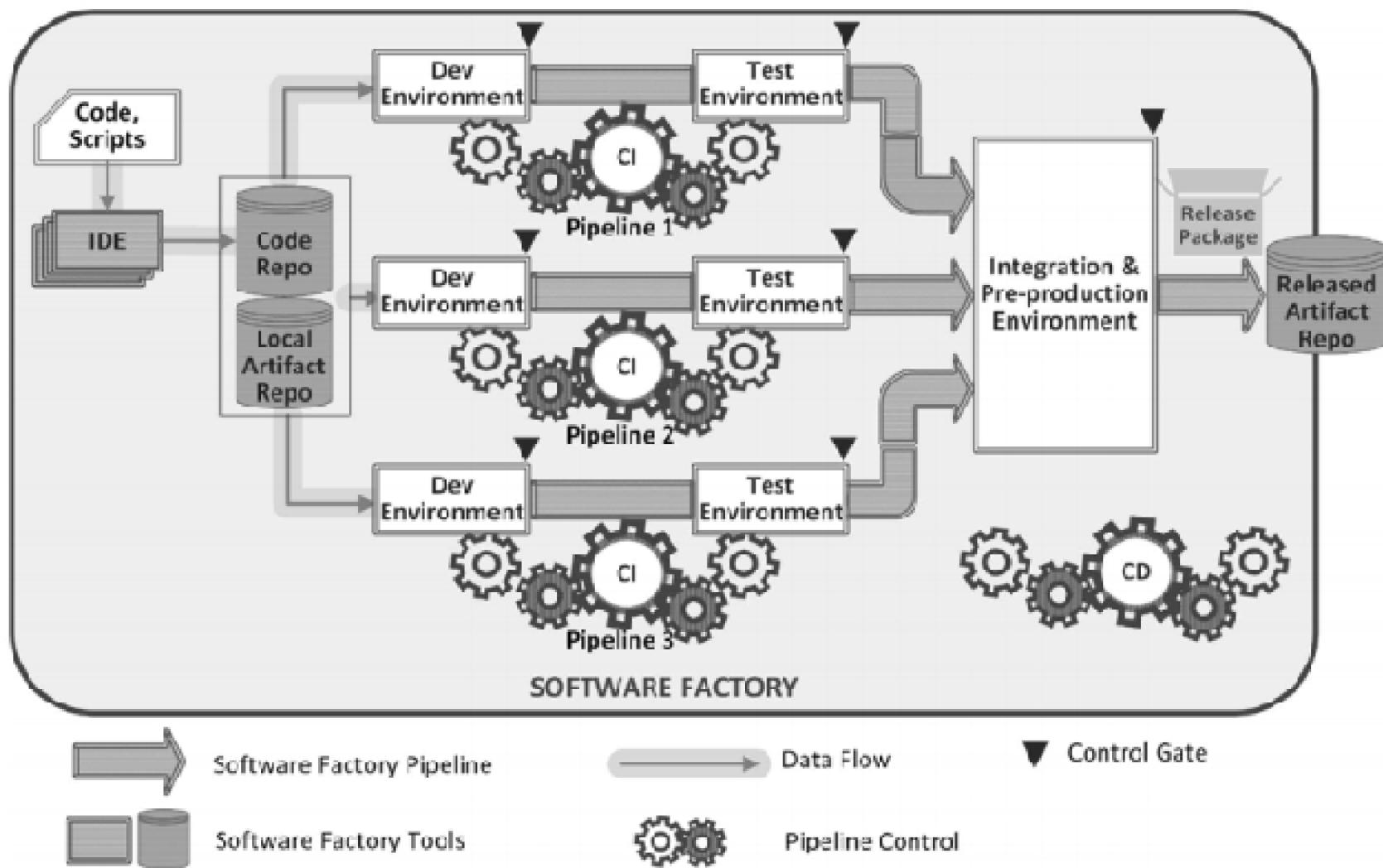
DevSecOps & Agile Scrum



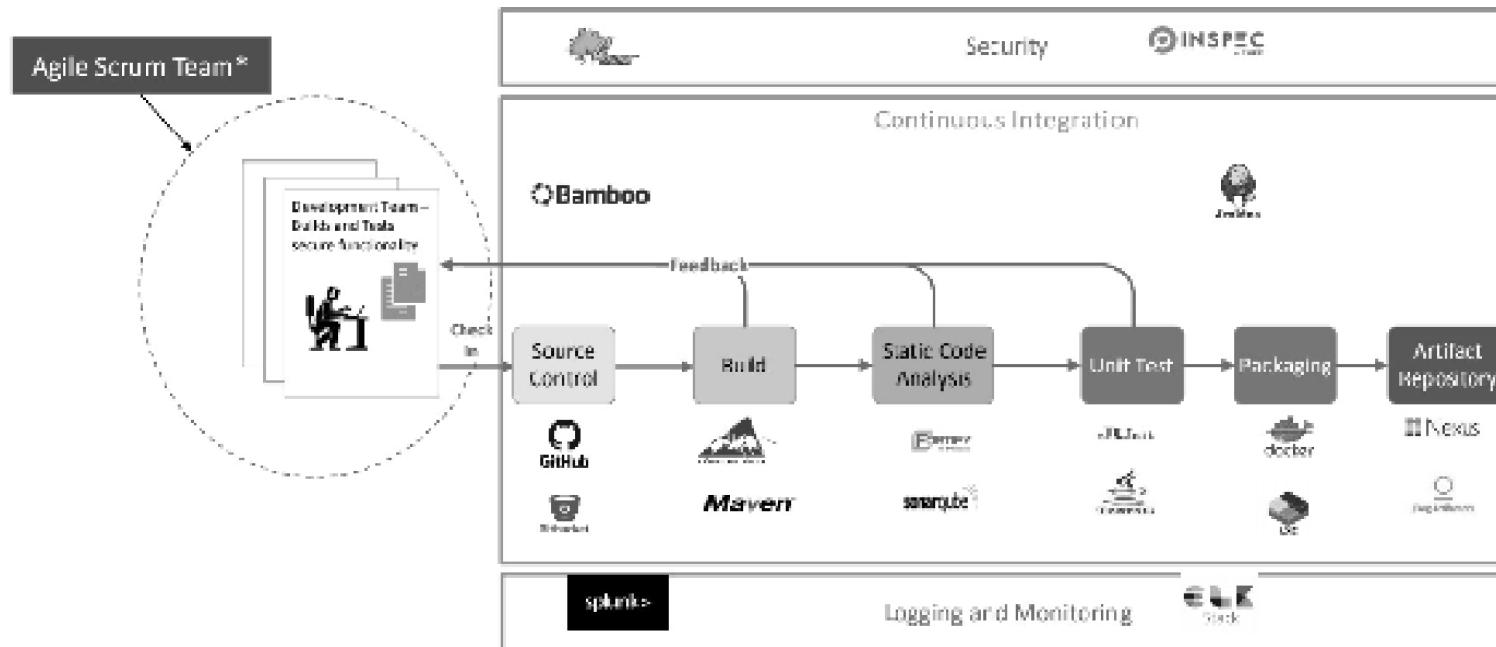
Ciclul de viață al DevSecOps



Fabrica software

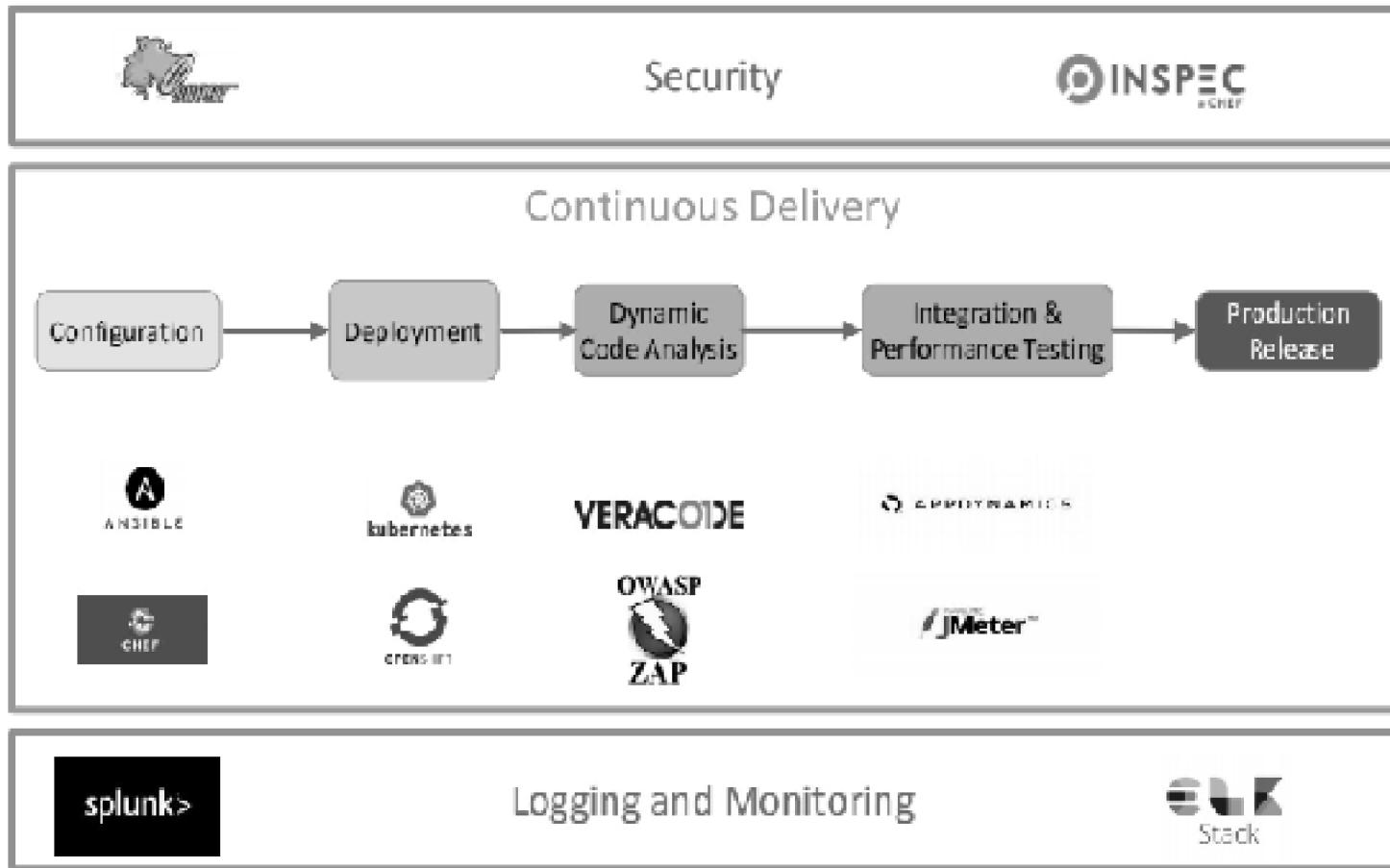


Integrarea continua (CI)



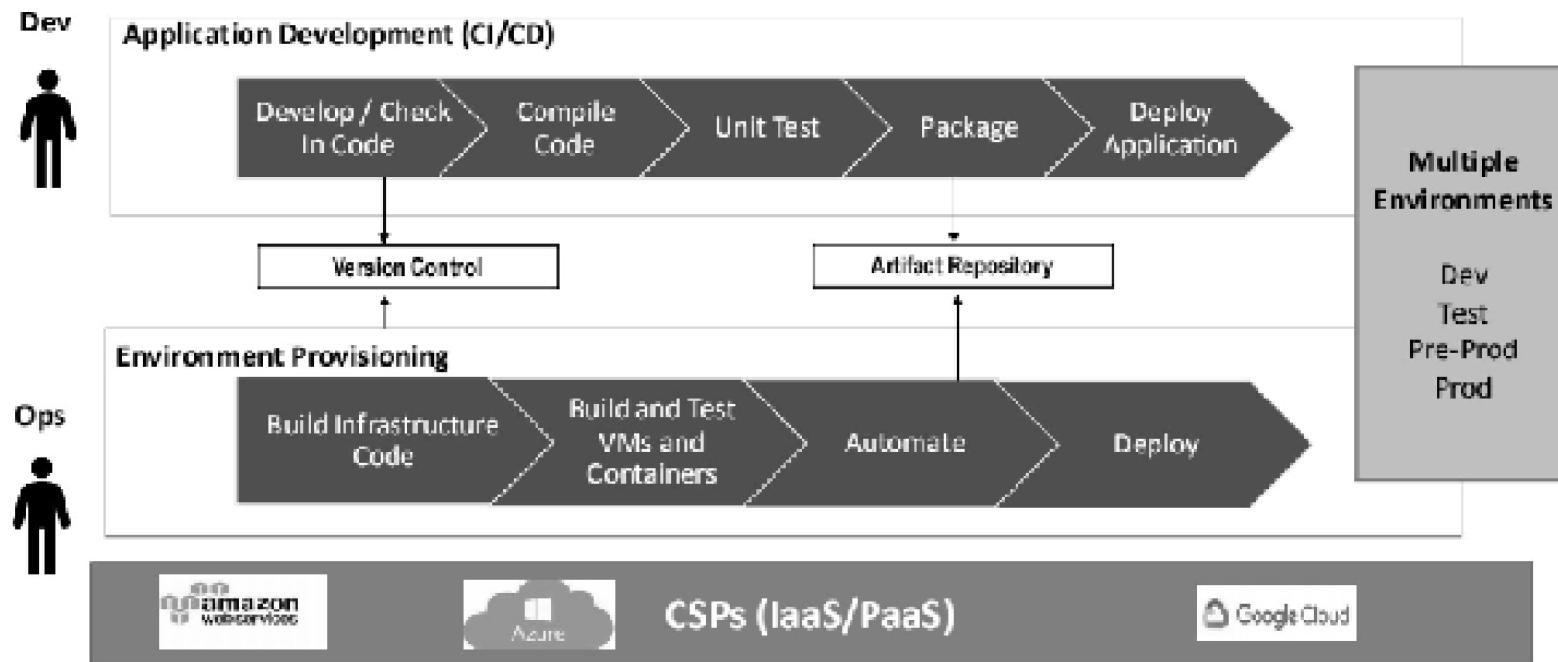
- X

Livrarea continua (CD)



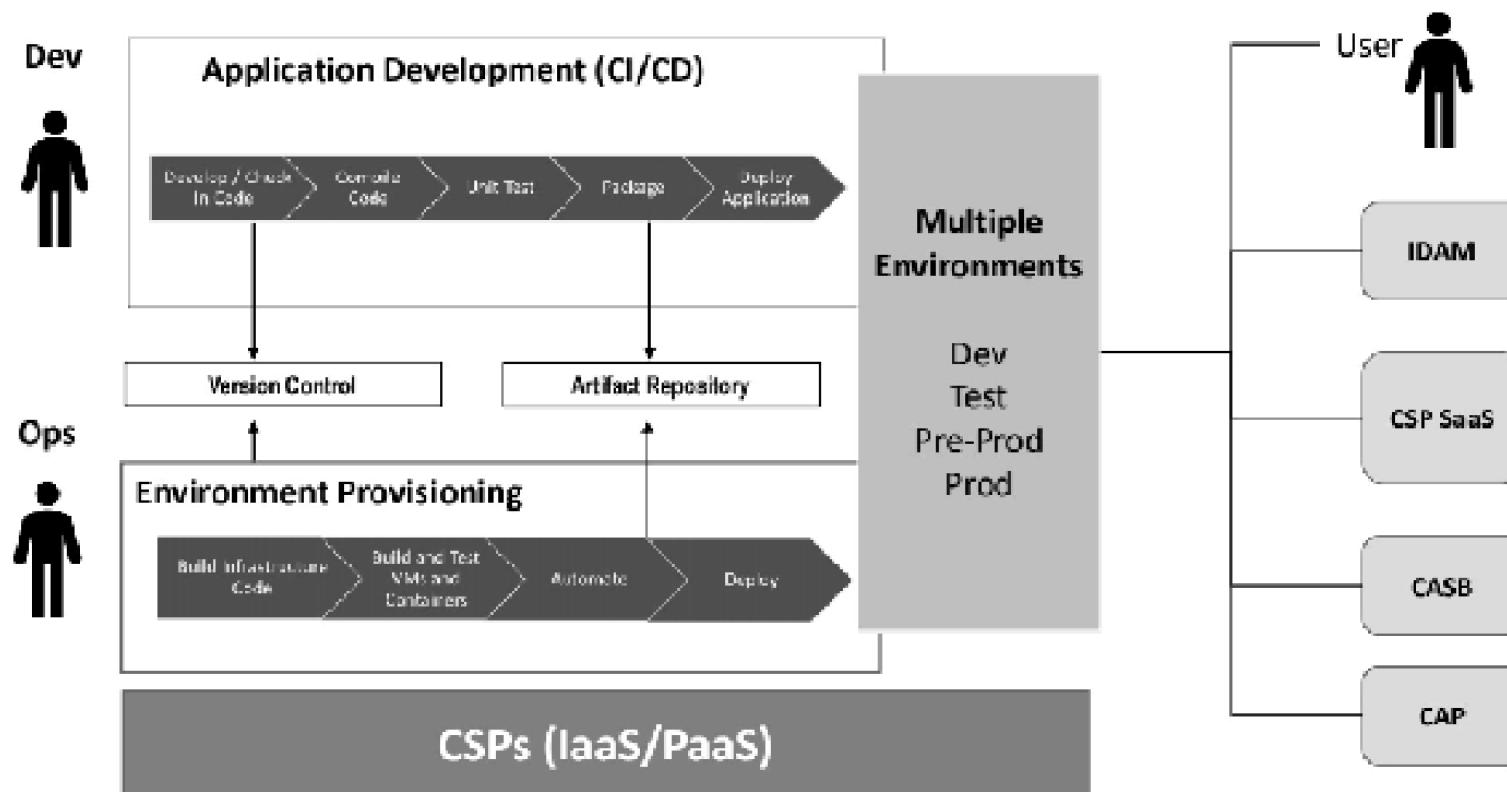
- X

CI - CD în nor



Cum se aplică în operații sistemul de integrare și livrare continuă (CI-CD) în nor

CI - CD în operații



- Cum se aplică în operații sistemul de integrare și livrare contiunuă (CI-CD) în operații

Instrumente specifice asigurării securității în DevSecOps

Instrument/securitate	Descriere	UNDE	Gratuit
Snort		OT& E	DA
Fortify SCA		DT& E	Nu
Gauntlet		DT& E	DA
HashiCorp Vault		DT& E	DA
Sonar Qube		DT& E	DA
OWASP Zap		DT& E și OT& E	DA

Sisteme Distribuite

Mihai Zaharia

Cursul 14

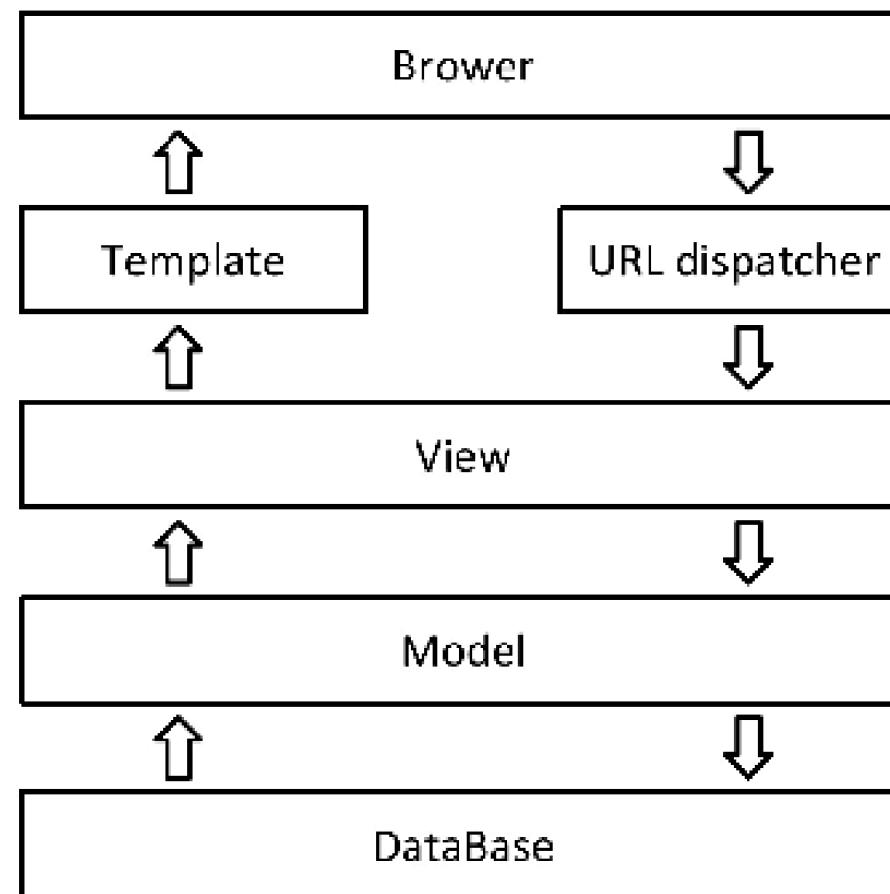
Caracteristici

- Respectă principiile paradigmăi modulare
- utilizează modelul MCV(MVT)
- Furnizează interfe' e de administare
- Furnizează o multitudine de API-uri
- Furnizează un sistem de gestiune a scheletelor
- Respectă principiul DRY (Don't repeat yourself)
- Respectă principiul cuplării scăzute
- Respectă principiul separării activitătilor transversale (concerns)
- consistență
- flexibilitate
- securitate

Cine il foloseste?

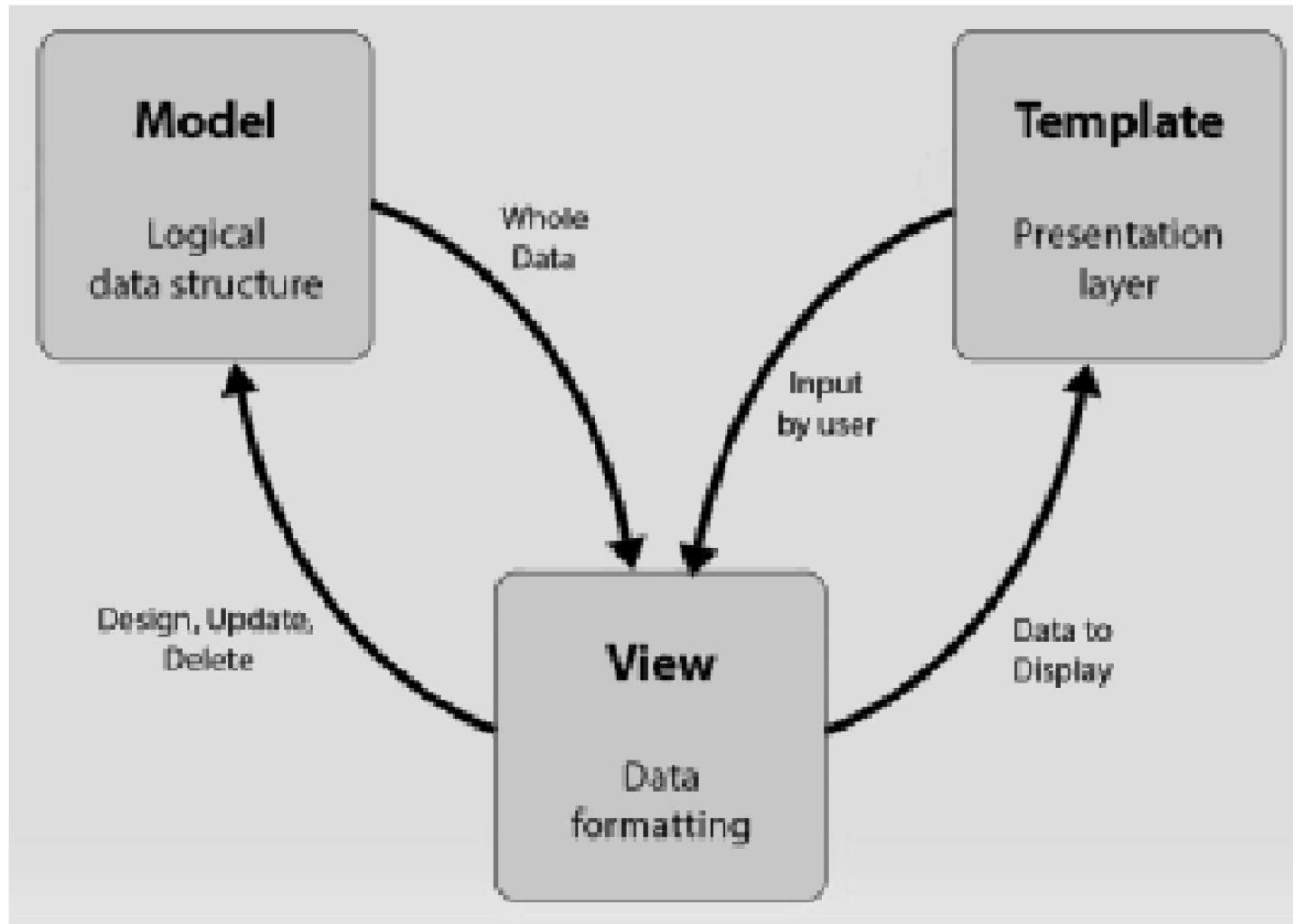
- Instagram
- Disqus
- Spotify
- YouTube
- NASA
- etc
- pt detalii vezi la Django Sites database

Model - Vizualizare - Schelet (MVT)



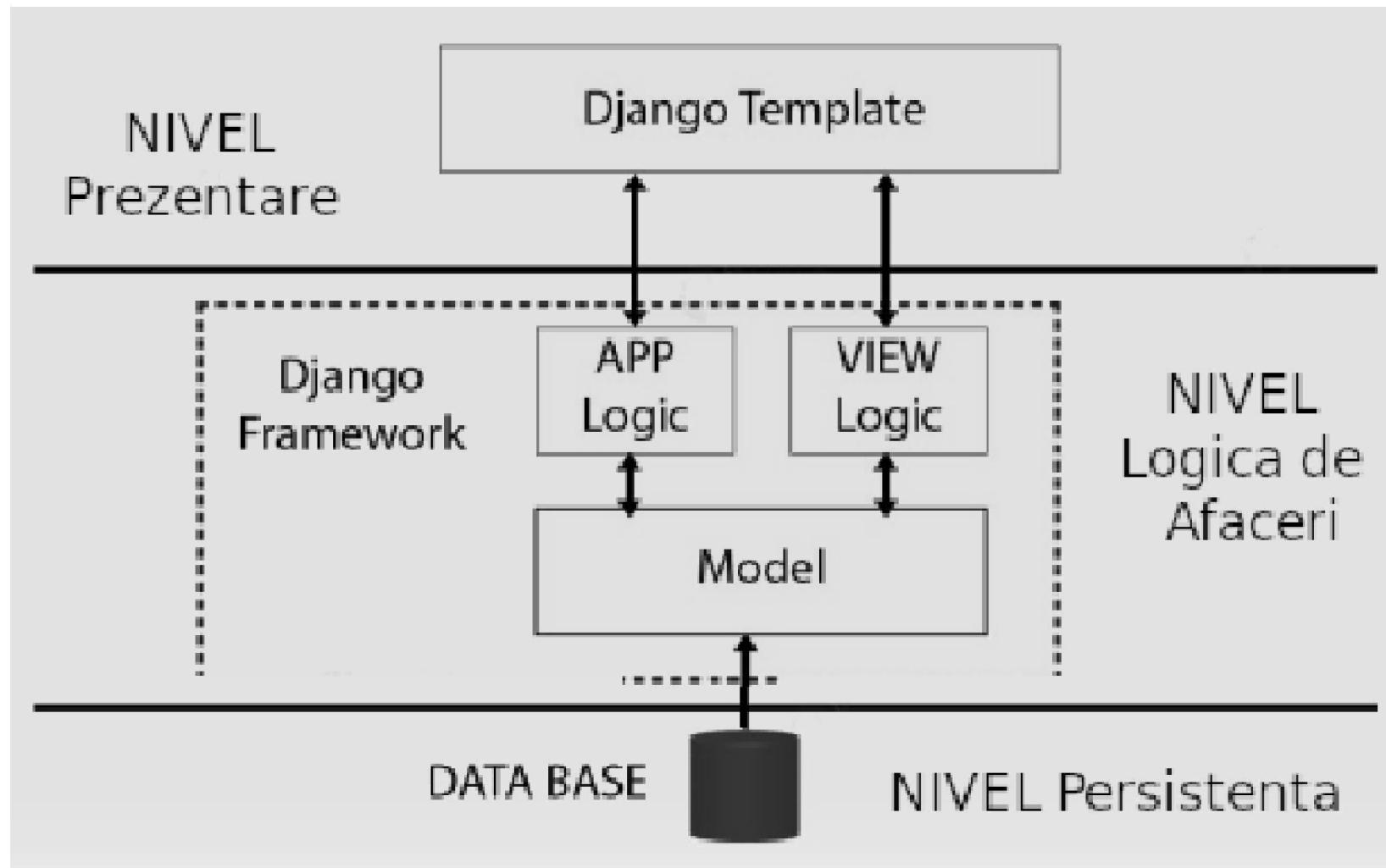
- X

MVT - detalii



- X

MVT - detalii



- X

Flask vs Django?

			
Type of Framework	Full Stack Web Framework		WSGI framework
Flexibility	Feature-packed		Full flexibility
ORM Usage	Built-in ORM		SQLAlchemy is used
Design	Batteries-included		Minimalistic design
Working Style	Monolithic		Diversified

Modificări settings.py

- TEMPLATES = [
 {
 'BACKEND':
 'django.template.backends.django.DjangoTemplates',
 'DIRS': [os.path.join(BASE_DIR, '_templates')],
 'APP_DIRS': True,
 'OPTIONS': {
 'context_processors': [
 'django.template.context_processors.debug',
 'django.template.context_processors.request',
 'django.contrib.auth.context_processors.auth',
 'django.contrib.messages.context_processors.messages',
],
 },
 },
]

Modificări în settings.py

- Dacă doresc un director static:

```
# necesar când va fi mutat în producție  
# STATIC_ROOT = os.path.join(BASE_DIR, '_static')
```

```
STATICFILES_DIRS = (  
    os.path.join(BASE_DIR, '_static'),  
)
```

- Modificați urmatoarele linii:

- DEBUG = False
 - ALLOWED_HOSTS = ['localhost', '127.0.0.1']

- Adăugați noile aplicații în INSTALLED_APPS:

- 'signup',

Modificări în views.py

- from django.shortcuts import render,
 render_to_response, RequestContext
from django.contrib import messages

```
def home(request):  
    # Render page  
    return render_to_response("index.html", locals(),  
  
context_instance=RequestContext(request))  
    return render(request, "index.html", context)
```

models.py

```
import uuid

class signupModel(models.Model):
    first_name = models.CharField(max_length=120, null=False, blank=False)
    last_name = models.CharField(max_length=120, null=False, blank=False)
    email_address = models.EmailField(max_length=120, null=False, blank=False)
    tech256_username = models.CharField(max_length=120, null=True, blank=True)
    website_url = models.URLField(max_length=200, null=True, blank=True)
    # Defineste variabilele pentru optiuni
    newsletter_nu = 'NU'
    newsletter_da = 'DA'
    newsletter_options = (
        (newsletter_nu, 'Nu - nu-mi plac gunoaiele!'),
        (newsletter_da, 'Da - trimite-o catre contul de gunoaie'),
    )
    # Coloane implice ale bazei de date si utilizarea variabilelor definite anterior
    newsletter_preference = models.CharField(max_length=4, choices=newletter_options,
                                               default=newletter_da)
    talk_description = models.TextField(null=False, blank=False)
    active = models.BooleanField(default=True)
    sid = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    def __str__():
        return self.sid
```

forms.py

- from django import forms
from .models import signupModel

```
class SignupForm(forms.ModelForm):  
    class Meta:  
        model = signupModel  
        fields =  
['first_name','last_name','email_address','tech256_use  
rname','website_url','newsletter_preference','talk_desc  
ription','active']
```

Modicare views.py

- # Creati aici vizualizarile (views).
from django.shortcuts import render, render_to_response, RequestContext
from django.contrib import messages

from .forms import SignupForm

def home(request):
 form = SignupForm(request.POST or None)
 if request.method == "POST":
 if form.is_valid():
 human = True
 save_it = form.save(commit=False)
 save_it.save()
 messages.success(request, "Bine că v-ati înregistrat!")
 else:
 form = SignupForm(request.POST or None)
 messages.error(request, "OOPS o eroare regretabila.")
 else:
 form = SignupForm(request.POST or None)

 # Renderul paginii
 return render_to_response("signupform.html", locals(),
 context_instance=RequestContext(request))
 return render(request, "signupform.html", context)

Crearea scheletelor

- index.html

- ```
{% if messages %}
 {% for message in messages %}
 {% if message.tags %} class="{{ message.tags }}"{% endif %}>
 {{ message }}

 {% endfor %}
 {% endif %}
 {% block content %}{% endblock %}
```

- signuform.html

- ```
{% extends 'index.html' %}  
    {% block content %}  
        <form method="POST" action="#"> {% csrf_token %}  
            {{ form.as_p }}  
            <input type='submit' value='Sign Up!'>  
        </form>  
    {% endblock %}
```

admin.py

```
from .models import signupModel

class signupAdmin(admin.ModelAdmin):
    list_display =
    ["sid","first_name","last_name","email_address","tech2
56_username"]
    class Meta:
        model = signupModel

admin.site.register(signupModel, signupAdmin)
```

Applicarea unui schelet

- Copie peste elementele statice din directorul _static.
- Copie codul HTML peste index.html și plasează o formă.
- Modică calea către URL-ul directorului _static
- Modifică CSS-ul pentru a corecta elementele.
- Vizualizează în formatele 0Desktop și Mobile.