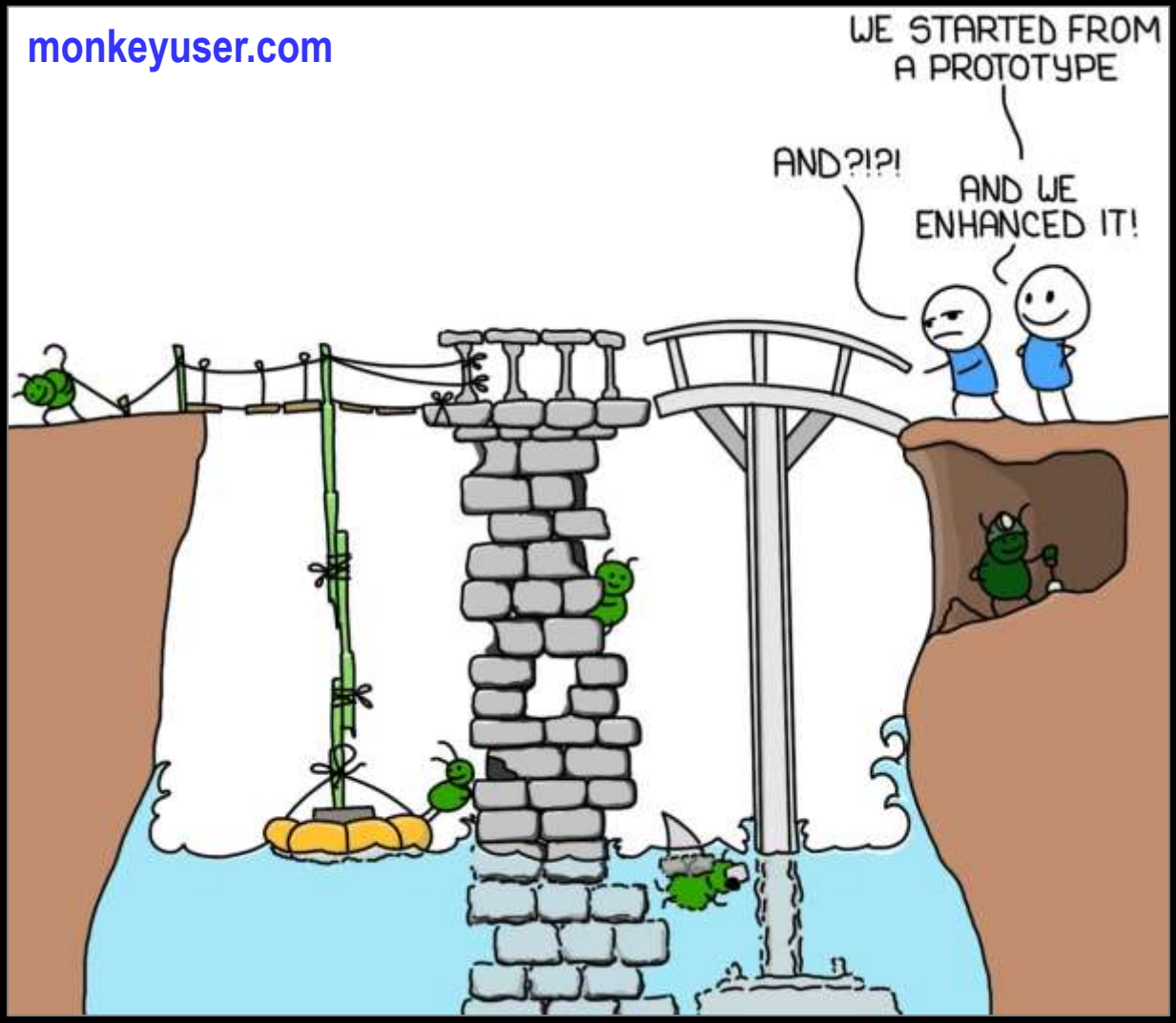
Tehnologii Web



programare Web (II)

de la MVC la arhitecturi Web și studii de caz

„Simplitatea este o complexitate rezolvată.”

Constantin Brâncuși

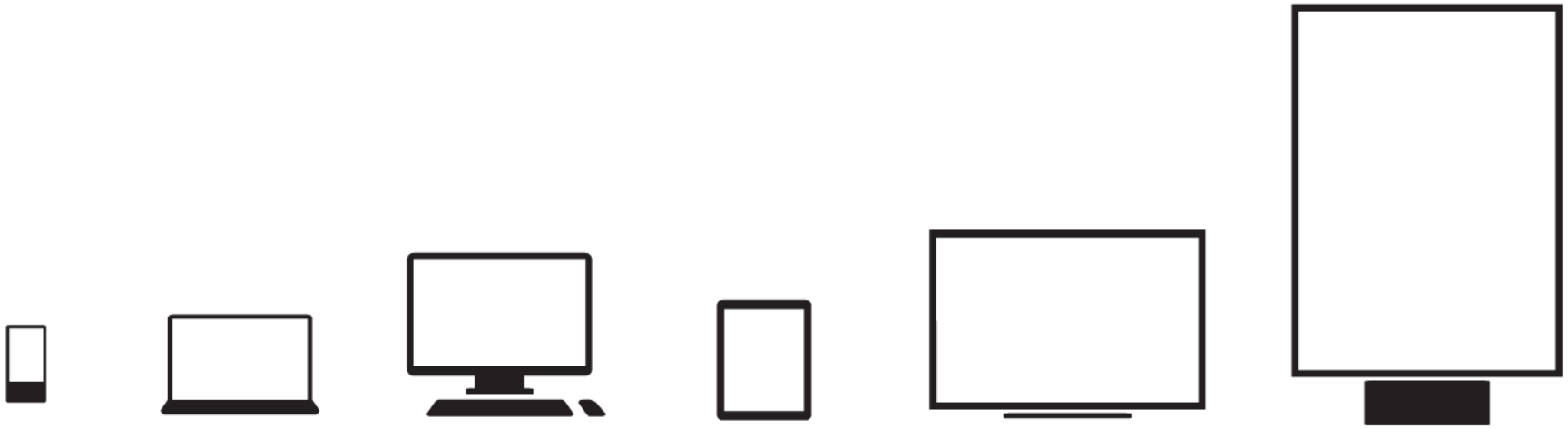
Aplicații Web

sisteme software complexe,
în evoluție permanentă

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

Realitate

mijloace multiple de interacțiune Web cu utilizatorul



mobil laptop PC tabletă (*smart*) TV ecran urban

plus, noii veniți:

game console *smart watch* *smart clothing*

smart appliances *smart home* *smart transportation*

Realitate

creșterea masei de utilizatori,
având așteptări tot mai mari din partea software-ului

de la conținut (hiper)textual 🔗
la aplicații Web sociale 👥 + interacțiune naturală ✋👂

Realitate

suportul privind dezvoltarea de aplicații
(limbaje, API-uri, biblioteci de cod, instrumente,...)
oferit de platforma hardware/software
la nivel de server(e) și/sau de client(i)

Realitate

neadaptare la cerințele economice (de tip *business*)

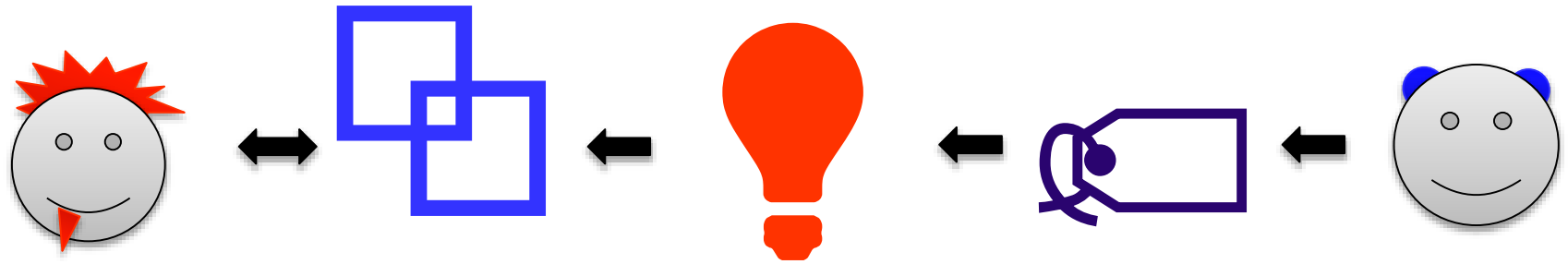
🛠 *development* *vs.* 📝 *marketing* *vs.* 👔 *management*

Realitate

privind proiectele Web de anvergură

întârzieri în lansare
neîncadrare în buget
lipsa funcționalității
calitatea precară a aplicației

scopuri psihologie comportament	interacțiune controale limbi naturale	funcționalități tehnologii algoritmi	indexare structurare meta-date	instrumente metodologii stimuli

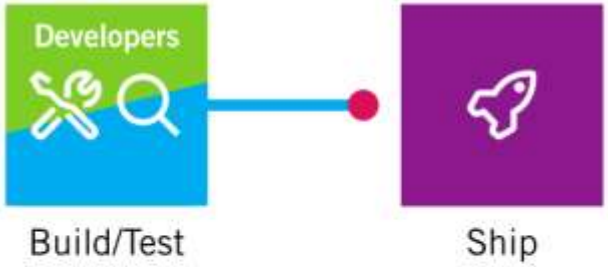


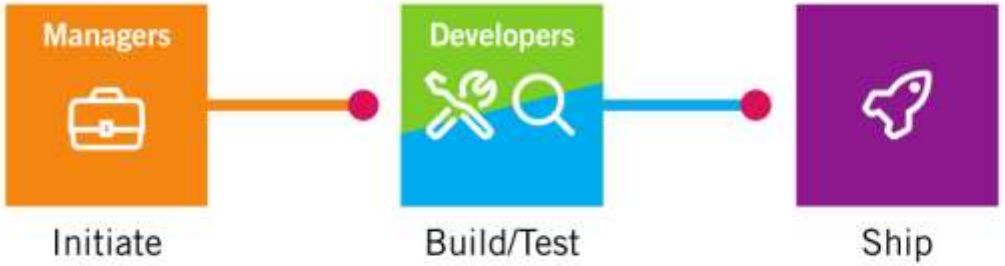
utilizatori	interfață	software	conținut	creatori

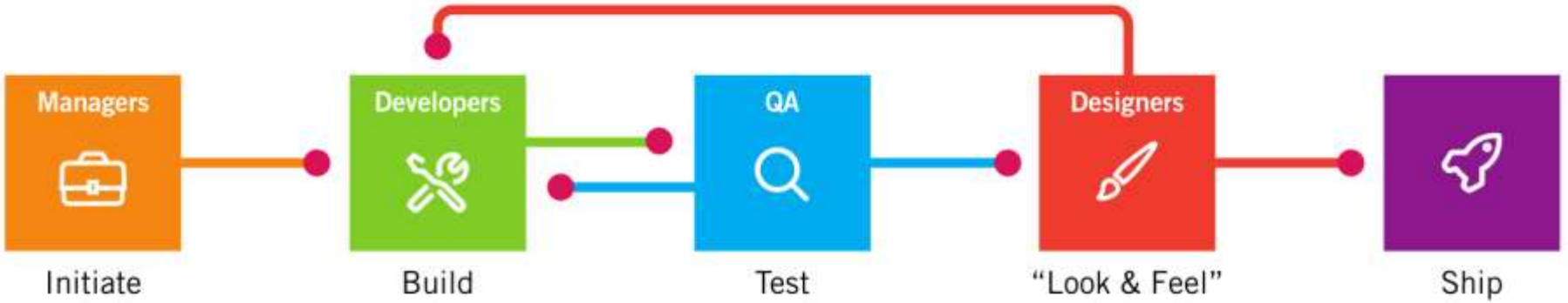
adaptare după Crumlish & Malone, 2009

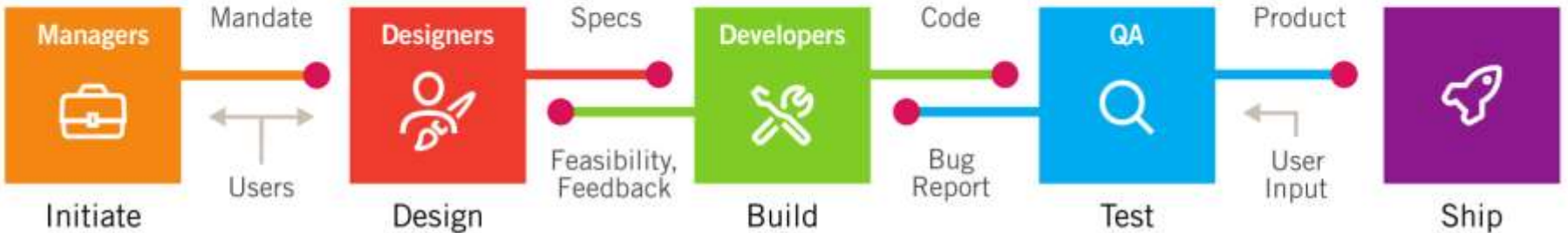
evoluția manierei de dezvoltare a produselor digitale (software)

Alan Cooper *et al.*, 2014









Asigurarea calității aplicațiilor Web

corectitudine și robustețe (*reliability*)
extindere + reutilizare (modularitate)
compatibilitate
eficiență (asigurarea performanței)
portabilitate

Asigurarea calității aplicațiilor Web

facilitarea interacțiunii cu utilizatorul (*usability*)
funcționalitate
relevanța momentului lansării (*timeliness*)
mentenabilitate
securitate

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

Asigurarea calității aplicațiilor Web

alte aspecte de interes:
integritate
reparabilitate
verificabilitate – inclusiv monitorizare (*logging*)
economie

Asigurarea calității aplicațiilor Web

esențialmente, de considerat:

preluarea și dirijarea cererilor – *dispatch*

oferirea funcționalităților de bază – *core services*

asocierea dintre construcții/abstracțiuni software (*e.g.*, obiecte) și modele de date – *mapping*

managementul datelor – *data*

monitorizarea și evaluarea sistemului – *metrics*

adaptare după Matt Ranney, "*What I Wish I Had Known Before Scaling Uber to 1000 Services*", GOTO Chicago 2016

highscalability.com/blog/2016/10/12/lessons-learned-from-scaling-uber-to-2000-engineers-1000-ser.html

Necesități

scopuri + cerințe clar specificate

dezvoltarea sistematică, în faze, a aplicațiilor Web

planificarea judicioasă a etapelor de dezvoltare

controlul permanent al întregului proces de dezvoltare

Necesități

scopuri + cerințe clar specificate

dezvoltarea sistematică, în faze, a aplicațiilor Web

planificarea judicioasă a etapelor de dezvoltare

controlul permanent al întregului proces de dezvoltare

▸ **inginerie Web**

În ce mod dezvoltăm o aplicație Web?

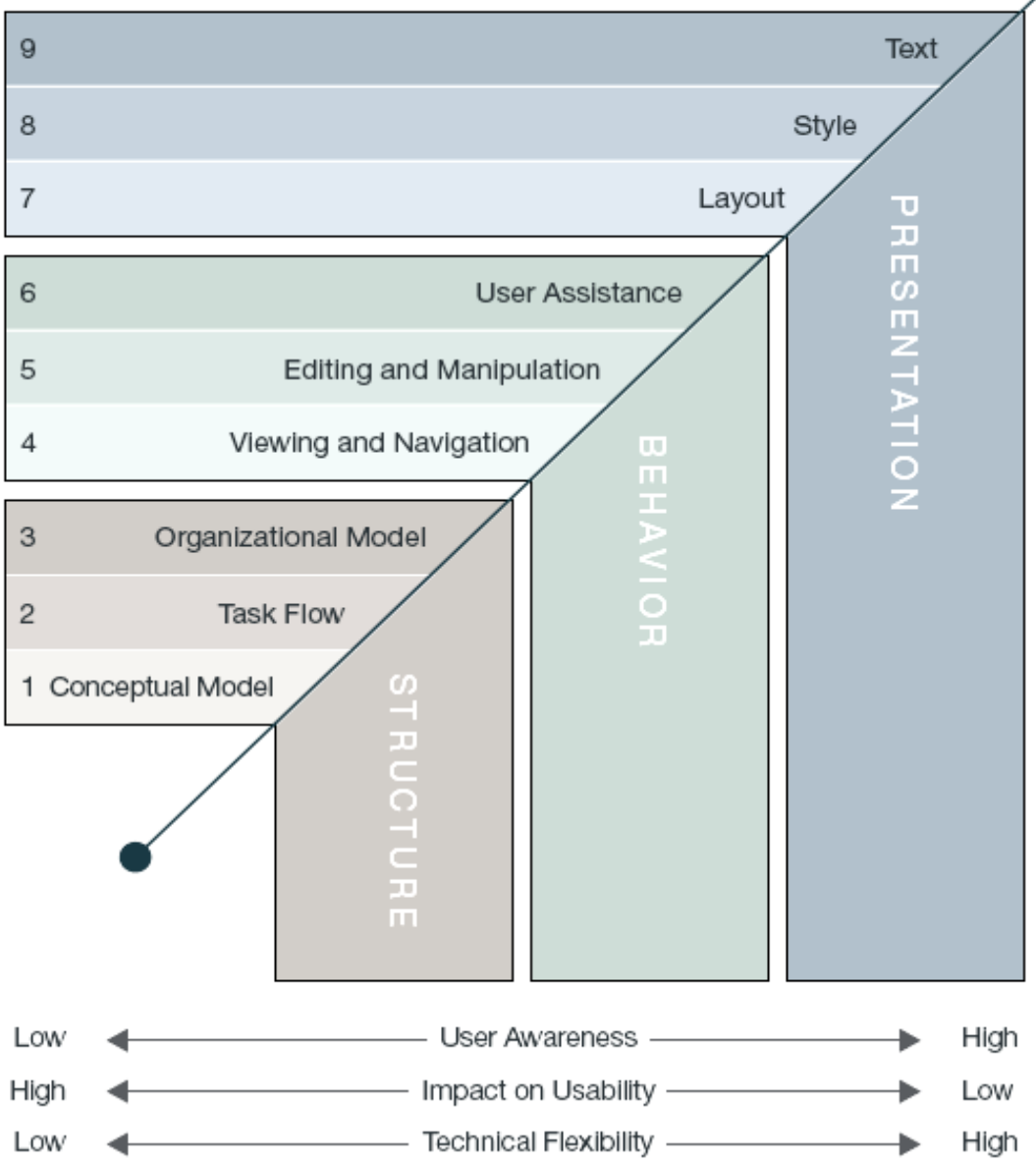
modelare

Uzual, se recurge la o metodologie

se preferă abordările conduse de modele
(MDA – *model-driven architecture*)

www.omg.org/mda/

avansat



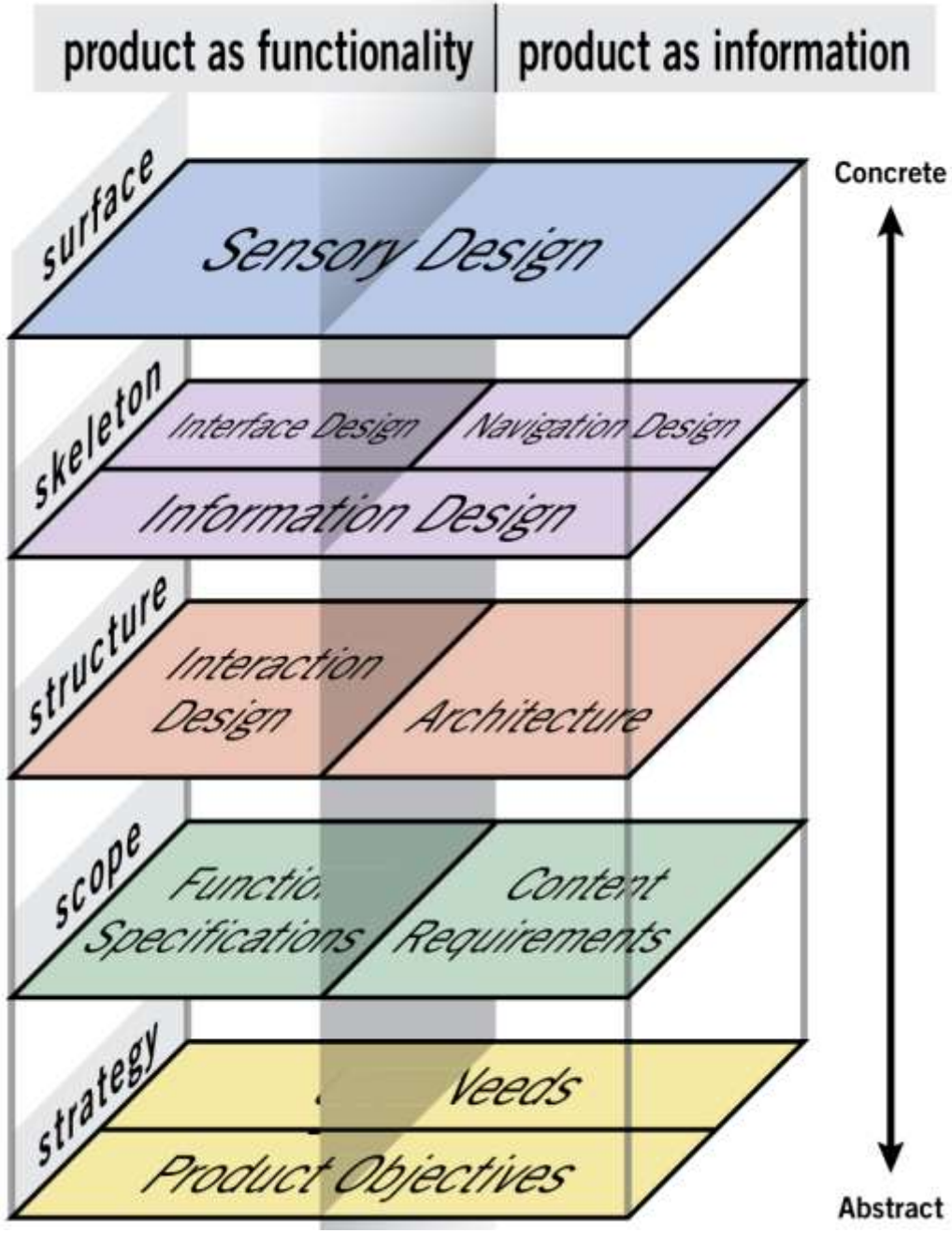
Robert Baxley

dezvoltarea aplicațiilor Web

Cerințe (*requirements*)
Analiză & proiectare (*software design*)
Implementare (*build*)
Testare (*testing*)
Exploatare (*deployment*)
Mentenanță (*maintenance*)
Evoluție (*evolution*)

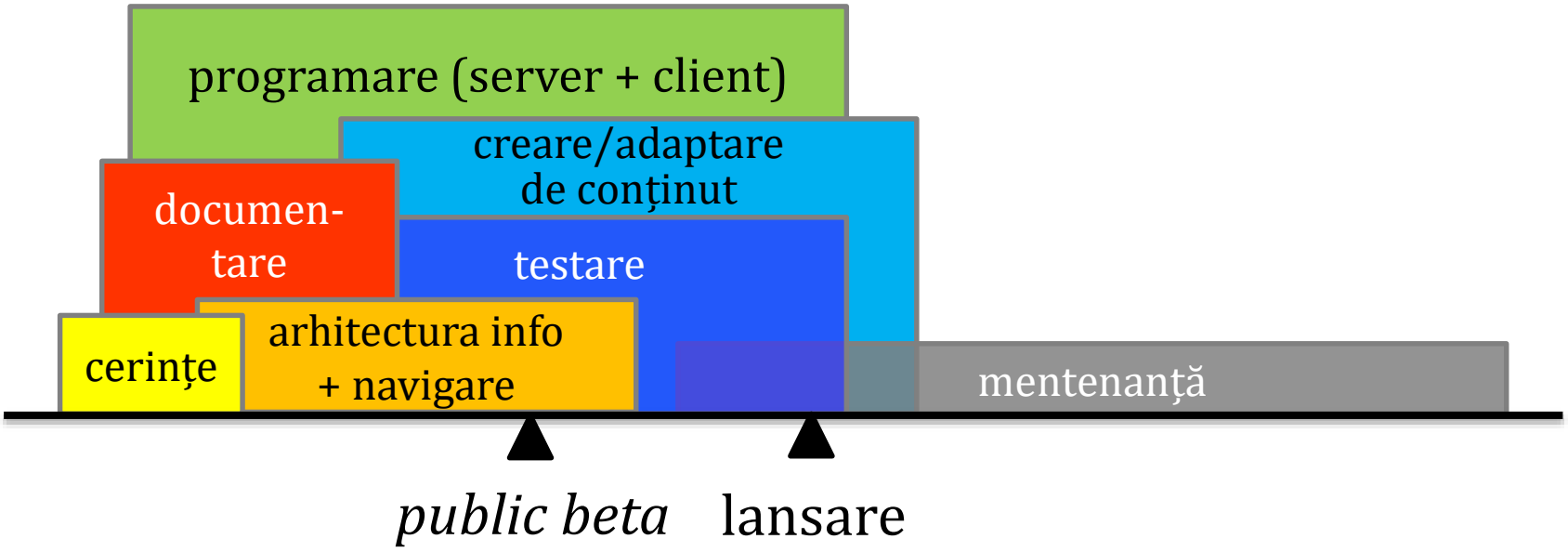
aplicație Web
(produs software)

funcționalitate
+
informații oferite



Jesse Garrett, *The Elements of User Experience* (2nd Edition), New Riders, 2011

dezvoltarea aplicațiilor Web



actualmente, sunt preferate metodologii agile

www.infoq.com/process-practices/

www.webpagefx.com/blog/web-design/agile/

avansat

dezvoltarea aplicațiilor Web

Metodologii moderne – exemple:

aim42 – practici și șabloane privind evoluția, mentenanța, migrarea și îmbunătățirea sistemelor software
aim42.github.io

12 Factor App – vizând aplicațiile aliniate paradigmei SaaS (*Software As A Service*)
12factor.net

dezvoltarea aplicațiilor Web: principii

start with needs

do less

design with data

do the hard work to make it simple

iterate. then iterate again

build for inclusion

understand context

build digital services, not Websites

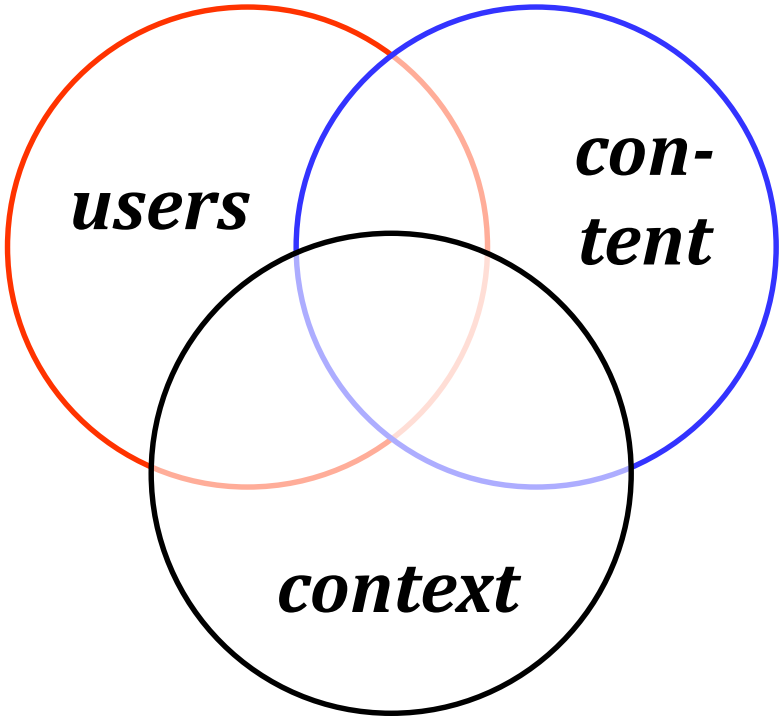
be consistent, not uniform

make things open; it makes things better

exemplu pentru gov.uk – Paul Downey & David Heath (2013)

cerințe

Stabilirea standardelor de calitate



cerințe

Obținere / licitare / negociere
a conținutului (datelor) și/sau codului-sursă

drepturi de autor – *copyright*
versus
cod deschis (*Open Source Licenses*)
www.opensource.org/licenses/category
+
date deschise
Creative Commons – **www.creativecommons.org/licenses/**

cerințe

Documentare privind domeniul aplicației Web

cu atragerea experților
– *subject matter expert* (SME) sau *domain expert* –
în domeniul problemei
ce trebuie soluționată de aplicația Web

cerințe

Aspecte specifice aplicațiilor Web

Lipsa unei structuri reale (tangibile)

Multi-disciplinaritate

Necunoașterea publicului-țintă real

Volatilitatea cerințelor și constrângerilor

Mediul de operare impredictibil

Impactul sistemelor tradiționale (*legacy*)

Aspecte calitative diferite

Inexperiența vizitatorilor

Termenul de lansare

cerințe: exemple

Viziune (*big idea*)

Basecamp: "*solves the critical problems that every growing business deals with*"

Vimeo: "*Watch, upload and share HD and 4K videos with no ads*"

Wikidata: "*a free and open knowledge base that can be read and edited by both humans and machines*"

cerințe: exemple

Punctele de plecare în dezvoltarea Flickr

presupuneri inițiale (*assumptions*):

oamenilor le place să-și împărtășească amintirile

folosirea succesului *blogging*-ului

partajarea nu doar a însemnărilor,
ci și a fotografiilor (personale)

suport pentru realizarea de comentarii + *tagging*

noi tipuri de cerințe

Privitoare la conținut

audiența – *e.g.*, internaționalizare
context de navigare
preferințe
disponibilitate permanentă (7 zile, 24 de ore/zi)
recurgerea la surse eterogene de date
căutare, filtrare, recomandare
etc.

noi tipuri de cerințe

Interacțiunea cu utilizatorul în contextul Web

inclusiv vizând Web-ul social

content mash-up

"*it's yours to take, re-arrange and re-use*"

noi tipuri de cerințe

Privitoare la mediul de execuție

(in)dependența de navigatorul Web

wired vs. wireless
on-line vs. off-line
suport pentru diverse standarde HTML5
interactivitate multi-dispozitiv (*responsive Web design*)

noi tipuri de cerințe

Referitoare la evoluție

utilizatorii sunt capabili să exploateze aplicația Web fără a trebui s-o (re)instaleze pe calculator/dispozitiv

noi tipuri de cerințe: aspecte de interes

inițial:
oferirea funcționalităților esențiale – *less is more*

noi tipuri de cerințe: aspecte de interes

inițial:
oferirea funcționalităților esențiale – *less is more*

versiuni ulterioare:
extinderea aplicației Web
– uzual, via o interfață de programare (API) publică,
încurajând dezvoltarea de soluții propuse de utilizatori

arhitecturi

Calitatea aplicațiilor Web este influențată de arhitectura pe care se bazează

arhitecturi

Dezvoltarea unei arhitecturi software ia în calcul:

cerințe funcționale

impuse de clienți,
vizitatori,
concurență,
factori decizionali (management),
evoluție socială/tehnologică,
...

arhitecturi

Dezvoltarea unei arhitecturi software ia în calcul:

factori calitativi

utilizabilitate
performanță
securitate
refolosire a datelor/codului
etc.

arhitecturi

Dezvoltarea unei arhitecturi software ia în calcul:

aspecte tehn(olog)ice

platforma hardware/software (sistem de operare)
infrastructura *middleware*
servicii disponibile – *e.g.*, via API-uri publice
limbaj(e) de programare
sisteme tradiționale (*legacy*)
...

arhitecturi

Dezvoltarea unei arhitecturi software ia în calcul:

experiența

recurgerea la arhitecturi și platforme existente
șabloane de proiectare (*design patterns*)
soluții „la cheie”: biblioteci, *framework*-uri, instrumente,...
management de proiecte
etc.

client(i)

mandatar (*proxy*)

zid de protecție (*firewall*)

intermediar(i) (*middleware*)

server(e) Web

server(e) de aplicații Web

cadre de lucru, biblioteci, alte componente

server(e) de stocare persistentă – *e.g.*, baze de date

server(e) de conținut multimedia

server(e) de management al conținutului – *e.g.*, CMS, *wiki*

aplicații/sisteme tradiționale (*legacy*)

„ingrediente” tipice

client(i)

mandatar (*proxy*)

zid de protecție (*firewall*)

intermediar(i) (*middleware*)

server(e) Web

server(e) de aplicații Web

cadre de lucru, biblioteci, alte componente

server(e) de stocare persistentă – *e.g.*, baze de date

server(e) de conținut multimedia

server(e) de management al conținutului – *e.g.*, CMS, *wiki*

aplicații/sisteme tradiționale (*legacy*)

eventual, recurgând la servicii în „nori” – ***cloud computing***

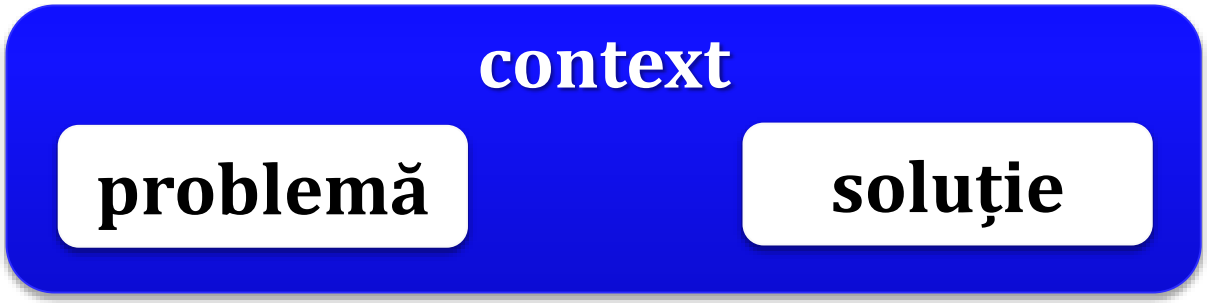
partajarea la cerere a resurselor de calcul și a datelor cu alte calculatoare/dispozitive pe baza tehnologiilor Internet (găzduire, infrastructură scalabilă, procesare paralelă, monitorizare,…)

Există anumite „rețete”
privind dezvoltarea de aplicații Web?

proiectare

Pattern (șablon)

regulă ce exprimă o relație
dintre un **context**, o **problemă** și o **soluție**



proiectare

Pattern (șablon)

regulă ce exprimă o relație
dintre un **context**, o **problemă** și o **soluție**

considerând punctul de vedere al unui expert

proiectare

Specificarea și/sau „recunoașterea” unui *pattern* poate avea loc la diverse niveluri:

prezentare a datelor – UI, *user interaction*, *visualization*,…
procesare – *business logic*, *scripting* etc.
integrare a componentelor – *code library development*
stocare a datelor – *database queries*, *database design*,…

proiectare

Exemple de colecții de șabloane
(*pattern repositories*)

privind proiectarea de software
wiki.c2.com/?DesignPatterns
patterns of enterprise application architecture
martinfowler.com/eaaCatalog/
interacțiunea cu utilizatorul (Adele – *a repository of publicly available design systems and pattern libraries*)
adele.uxpin.com

proiectare

Web Patterns

Model View Controller
Page Controller
Front Controller
Template View
Transform View
Application Controller

M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley (2003)

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

proiectare

Session State Patterns

Client Session State
Server Session State
Database Session State

M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley (2003)

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

proiectare

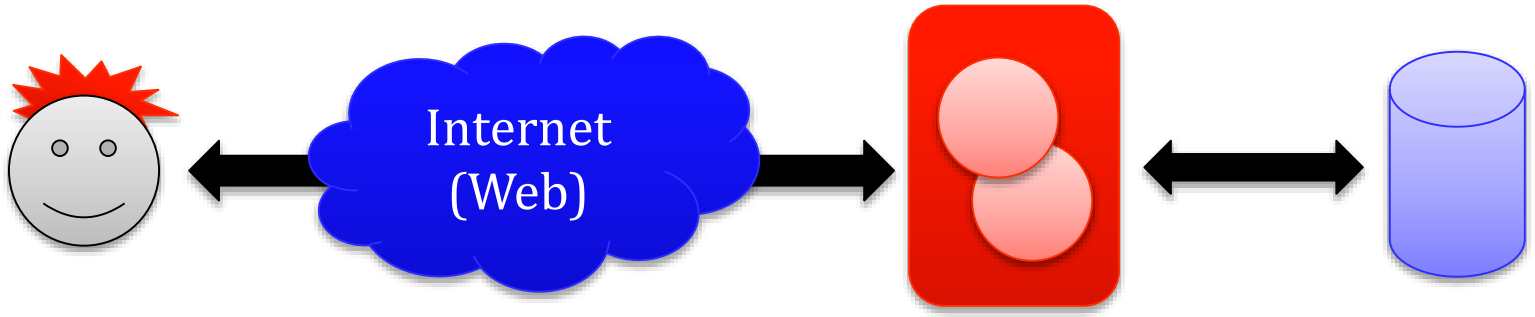
Data Source Architectural Patterns

Table Data Gateway
Row Data Gateway
Active Record
Data Mapper

M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley (2003)

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

aplicație Web = **interfață** + **program** + **conținut** (date)
trei strate (*3-tier application*)



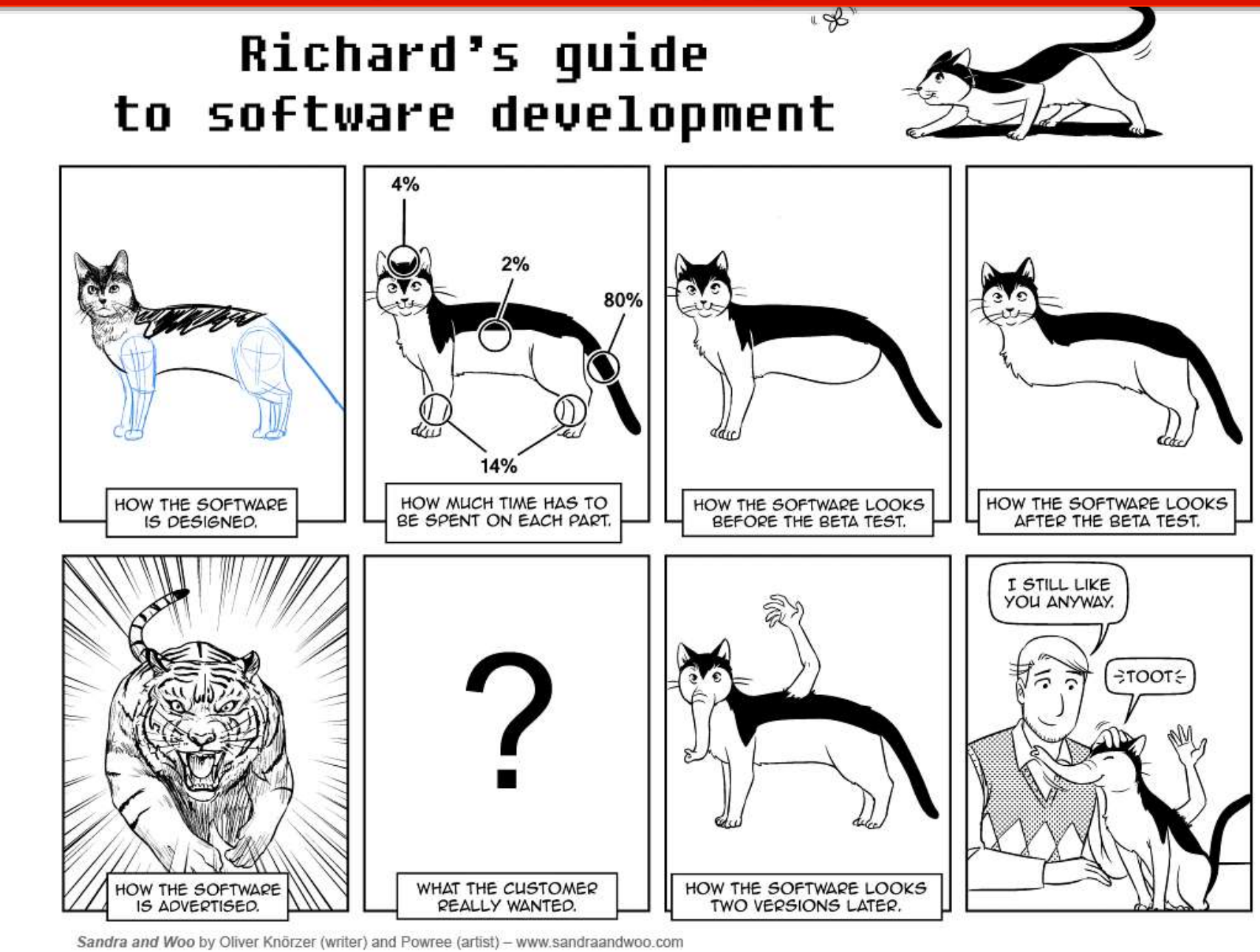
Client
(*interface*)

Server de aplicații
(*application*)

Stocare
(*persistence*)

C. Henderson, *Scalable Web Architectures*, Web 2.0 Expo, 2007: iamcal.com/talks/

(în loc de) pauză



arhitecturi web

Modelul de structurare a datelor este separat
de maniera de procesare
(controlul aplicației, *business logic*) și
de modul de prezentare a acestora (interfața Web)

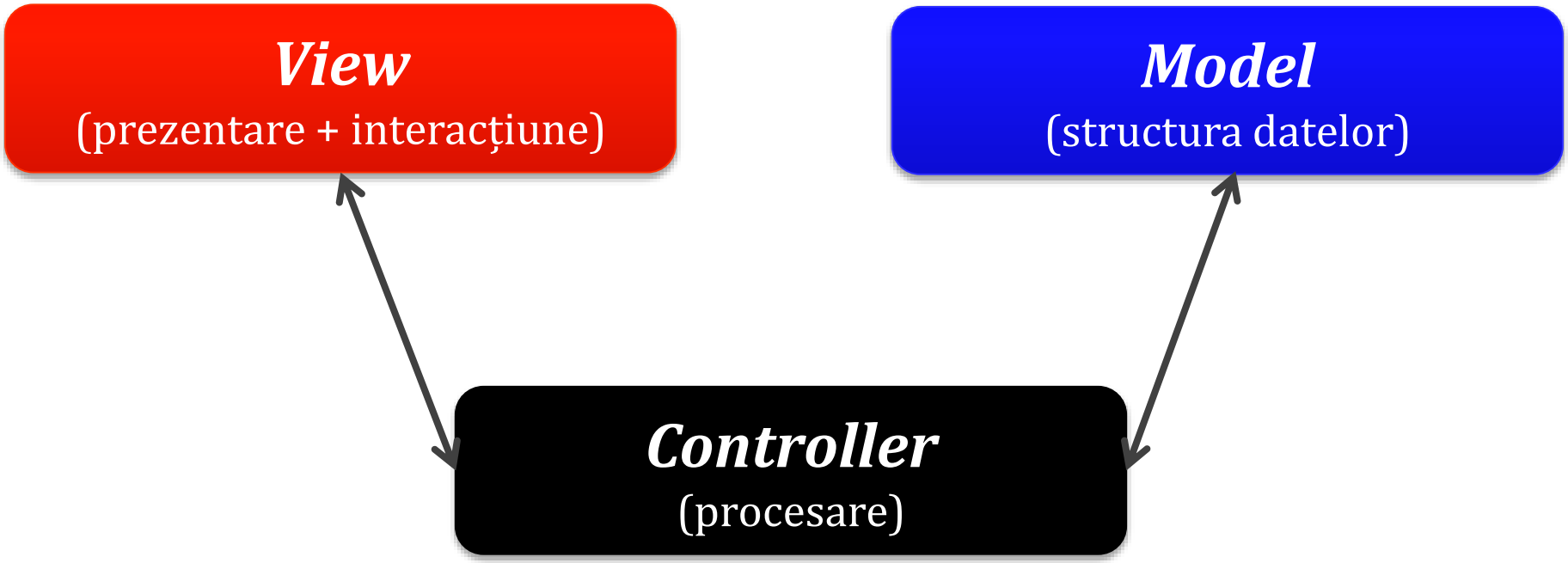
principiu:
demarcarea responsabilităților
(***separation of concerns***)

arhitecturi web: mvc

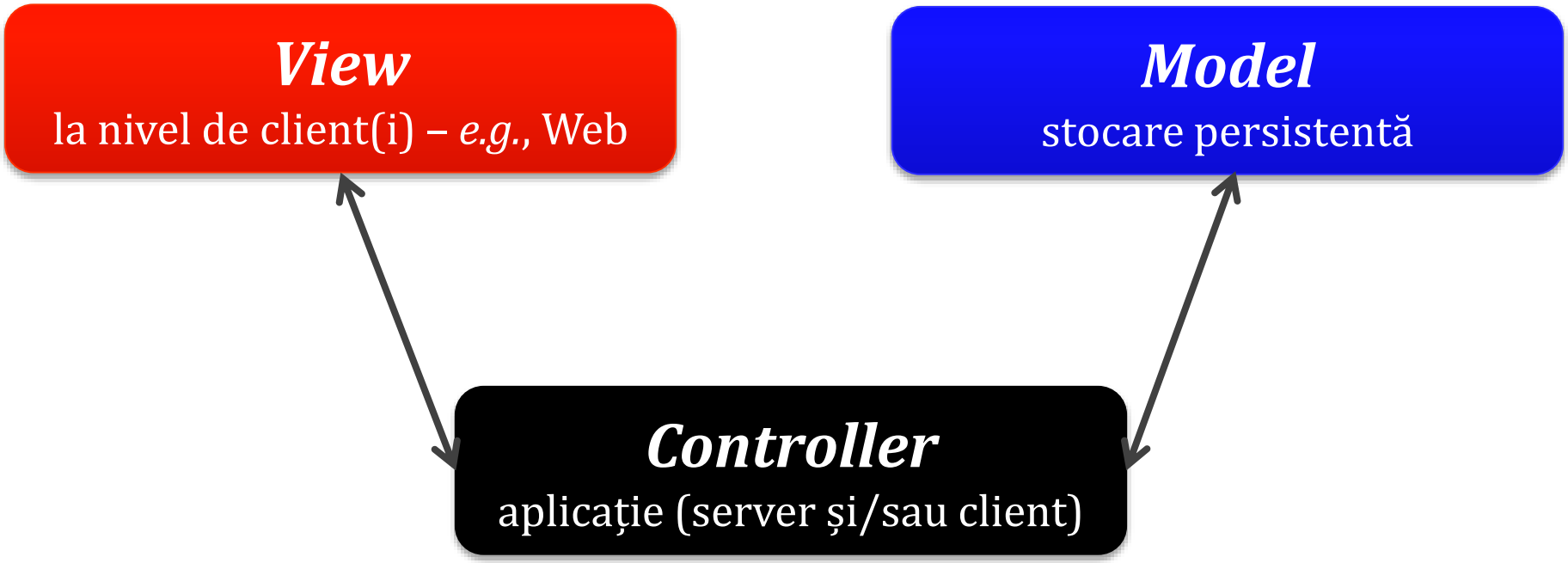
Majoritatea aplicațiilor Web sunt dezvoltate conform **MVC** (***Model-View-Controller***)

Trygve Reenskaug, 1979
heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf

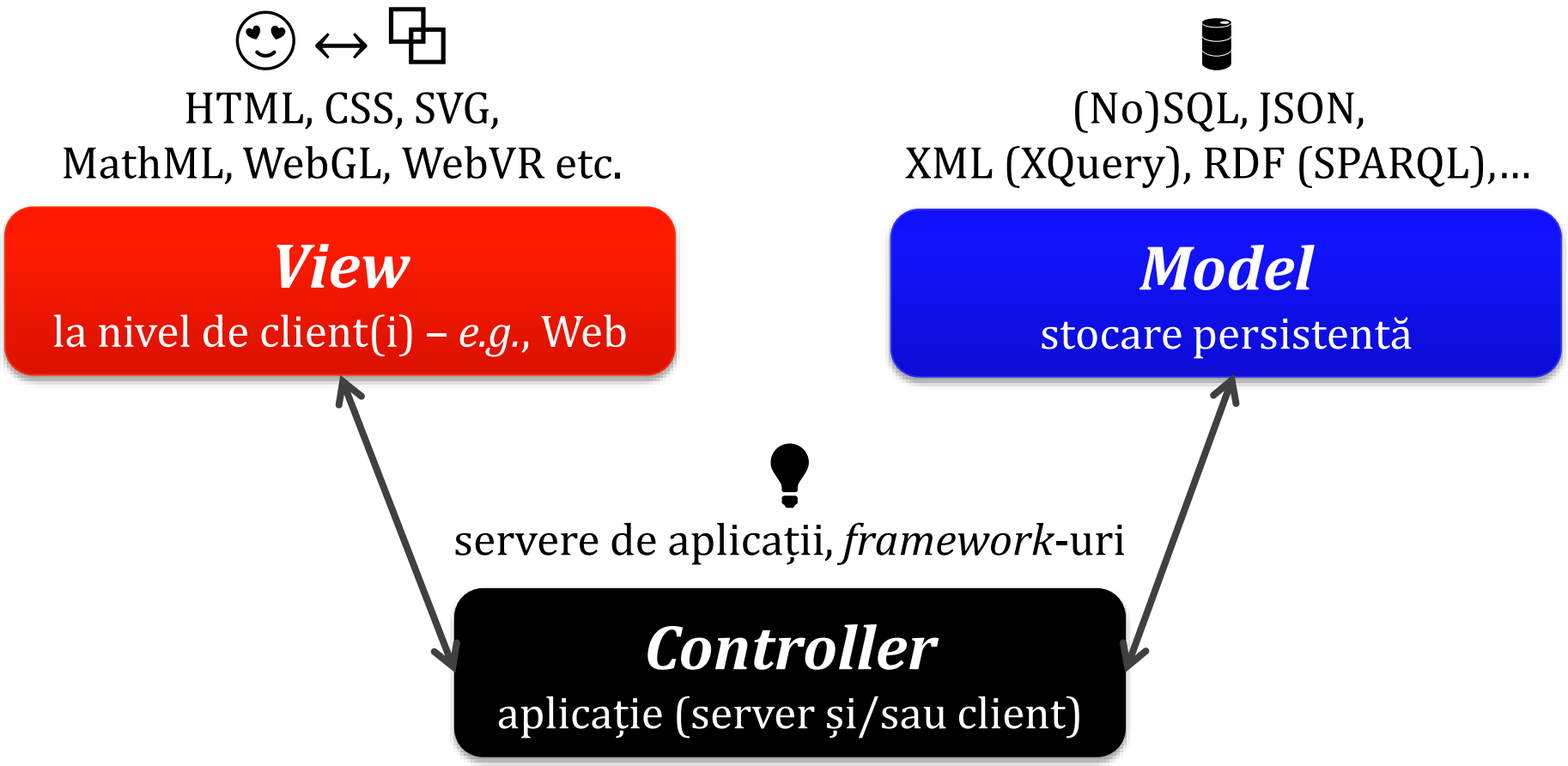
arhitecturi web: mvc



arhitecturi web: mvc



arhitecturi web: mvc



arhitecturi web: mvc

Poate fi implementat și într-un limbaj neorientat-obiect

încurajat/impus de *framework*-uri Web specifice

exemplificări diverse:

ASP.NET MVC (C# *et al.*), Catalyst (Perl), ColdBox (ColdFusion), Django (Python), FuelPHP, Grails (Groovy), Laravel (PHP), Lift (Scala), Rails (Ruby), Sails (Node.js), TurboGears (Python), Yesod (Haskell), Wicket (Java), Wt (C++), Zikula (PHP), ZK (Java)

arhitecturi web: mvc

Controller

responsabil cu preluarea cererilor de la client
(cereri GET/POST emise pe baza acțiunilor utilizatorului)

gestionează resursele necesare satisfacerii cererilor

uzual, va apela un *model* conform acțiunii solicitate
și, apoi, va selecta un *view* corespunzător

arhitecturi web: mvc

Model

resursele gestionate de software – utilizatori, mesaje, produse etc. – au modele specifice

desemnează datele + regulile (*i.e.* restricțiile) vizând datele ▸ **concepte** manipulate de aplicația Web

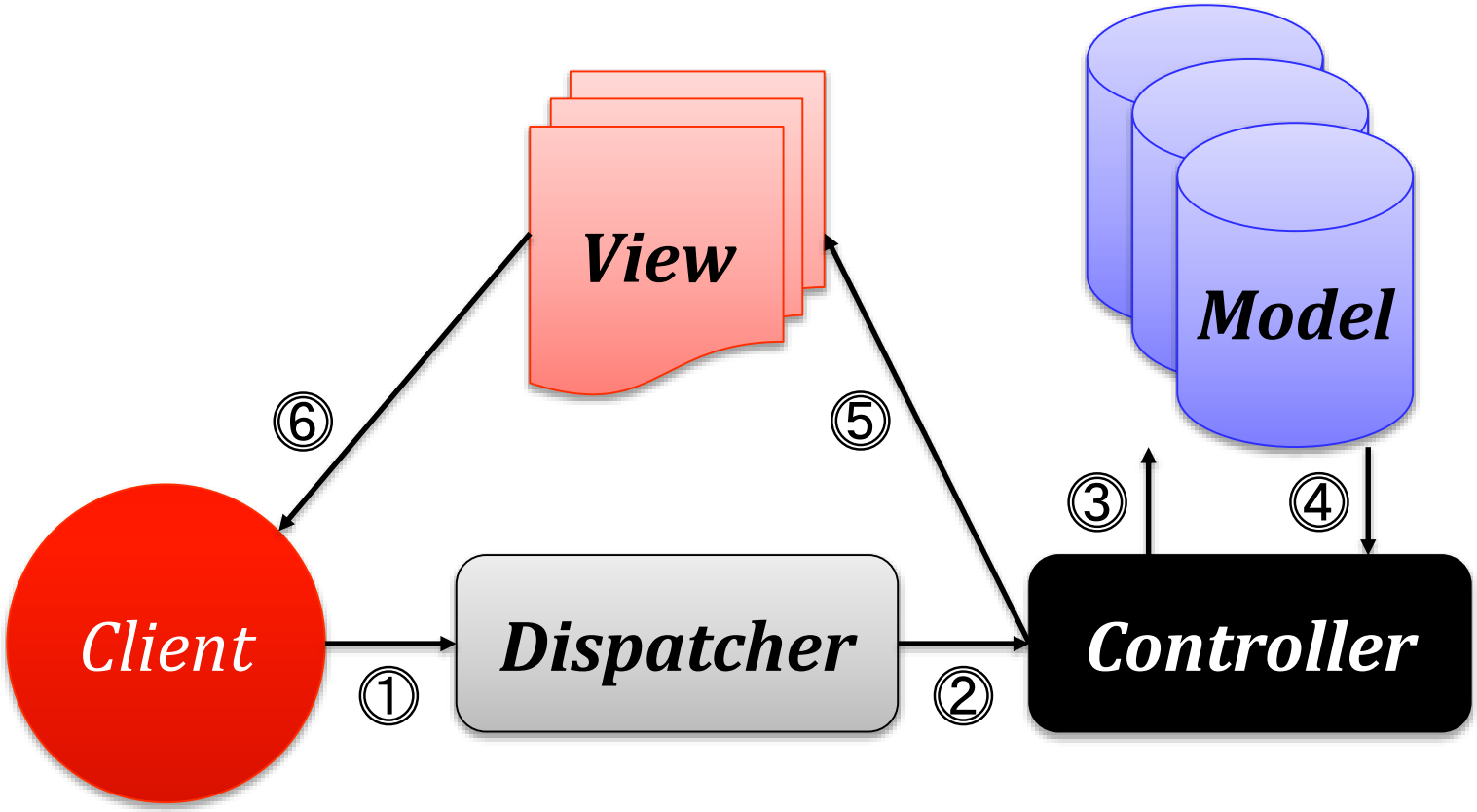
oferă *controller*-ului o reprezentare a datelor solicitate și e responsabil cu validarea datelor menite a fi stocate

arhitecturi web: mvc

View

furnizează diverse maniere de prezentare a datelor
furnizate de *model* via *controller*

pot exista *view*-uri multiple,
alegerea lor fiind realizată de *controller*



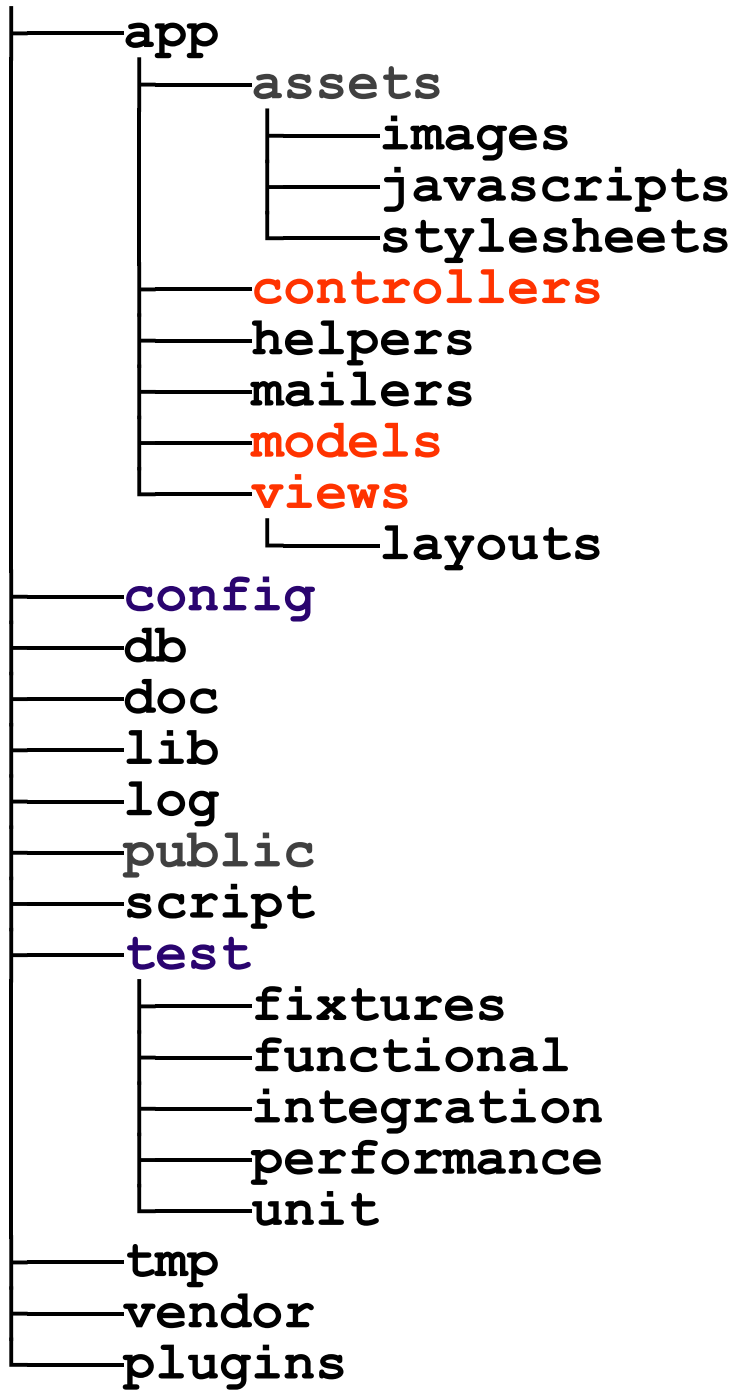
etape tipice:
(1) cerere trimisă de client – *e.g.*, navigator Web,
(2) dirijare (*routing*) a cererii către *controller*,
(3) recurgerea la un *model*, (4) furnizare date dorite,
(5) selectare a unui *view*, (6) transmitere conținut la client

arhitecturi web: mvc

Arhitectura generică a unei aplicații Web va consta dintr-un set de resurse referitoare la *controller*, *model* și *view*

uzual, *framework*-ul Web folosit impune o anumită structură a fișierelor aplicației ce va fi implementată

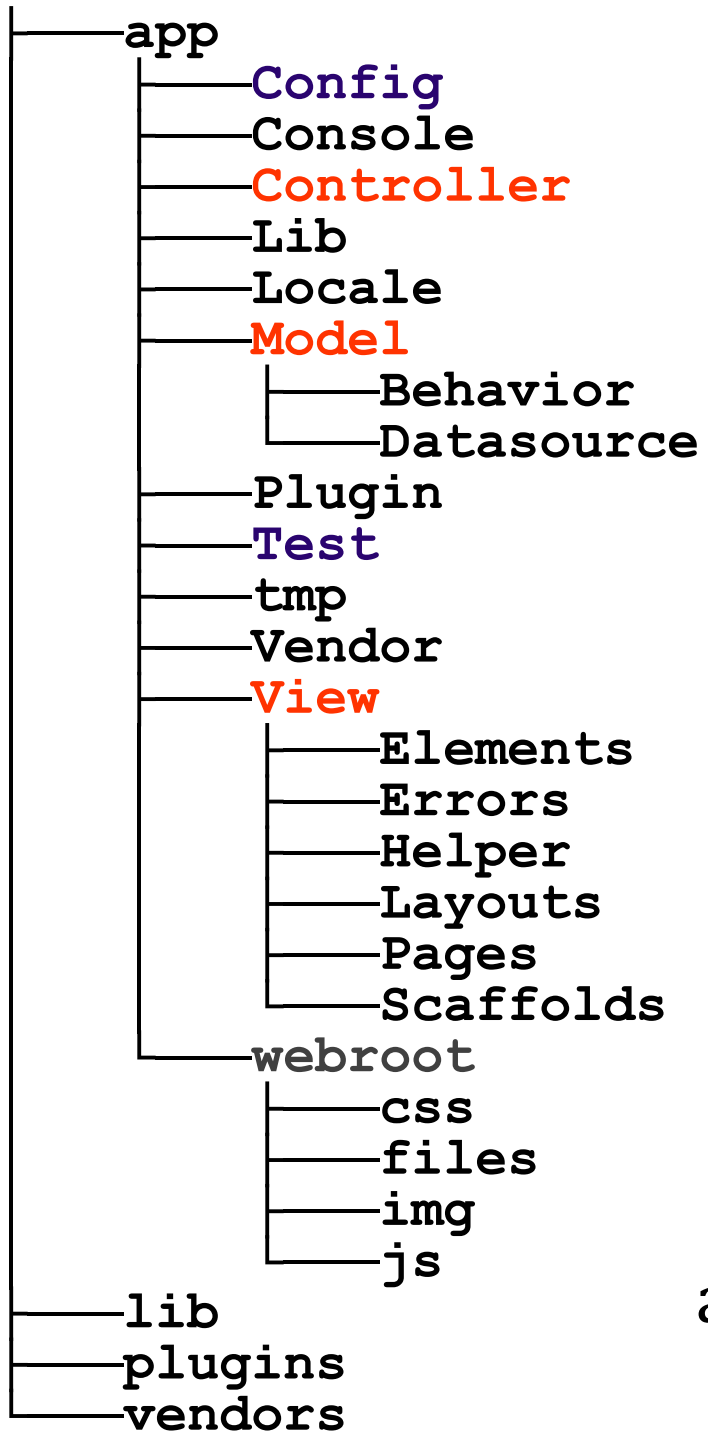
avansat



„scheletul” unei aplicații Web
create în Ruby on Rails
rubyonrails.org

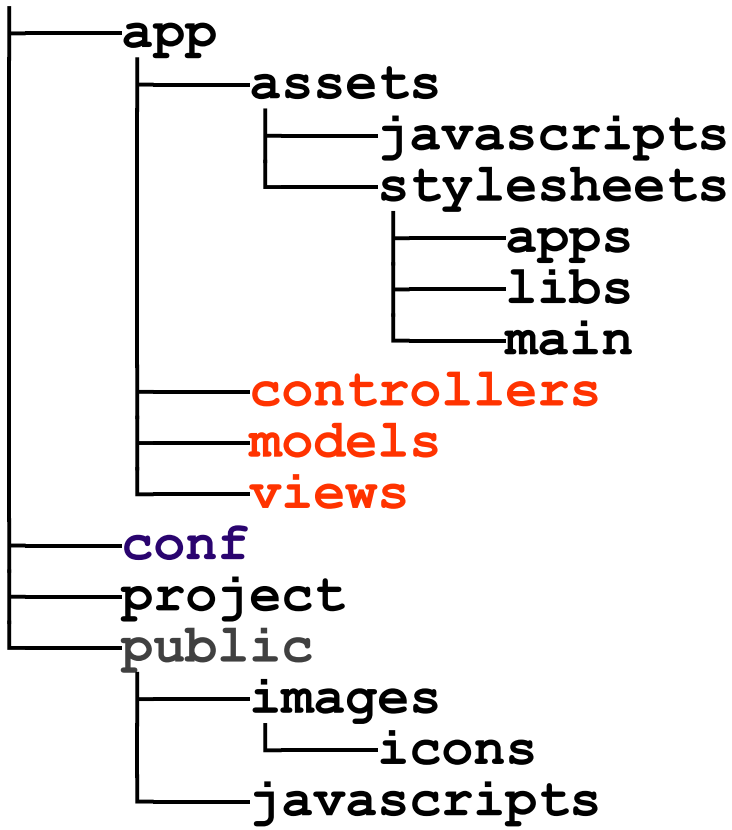
avansat

structura de directoare
în cazul unei aplicații Web
folosind *framework*-ul
CakePHP
cakephp.org



altele: CodeIgniter, FuelPHP, Laravel,
Symfony, Yii, Zend Framework

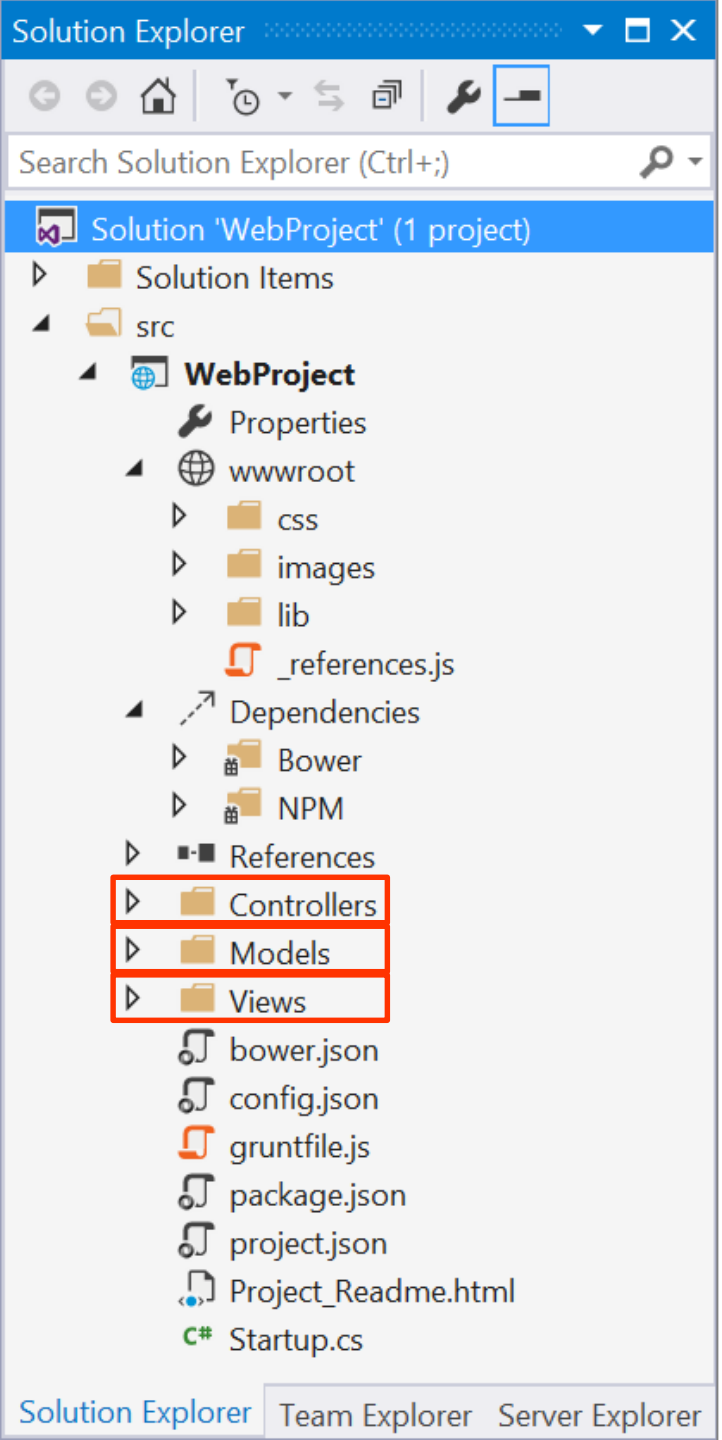
avansat



structura de directoare în cazul unei aplicații Web ce recurge la *framework*-ul Play pentru Java și Scala

www.playframework.com

avansat



structura proiectului software
pentru o aplicație ASP.NET MVC
www.asp.net/mvc

arhitecturi web: mvc

Variante derivate:

HMVC (*Hierarchical Model-View-Controller*)
MVP (*Model View Presenter*)
MVVM (*Model View ViewModel*)

pentru detalii, a se studia
Herberto Graca, *MVC and its alternatives* (2017)
herbertograca.com/2017/08/17/mvc-and-its-variants/

Prin ce mijloace poate fi implementată o aplicație Web?

implementare

Server de aplicații Web

scop:
eficientizarea proceselor de dezvoltare
a aplicațiilor Web complexe

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

implementare

Server de aplicații Web

simplifică invocarea de programe (*script*-uri)

- generarea de conținut la nivel de server Web

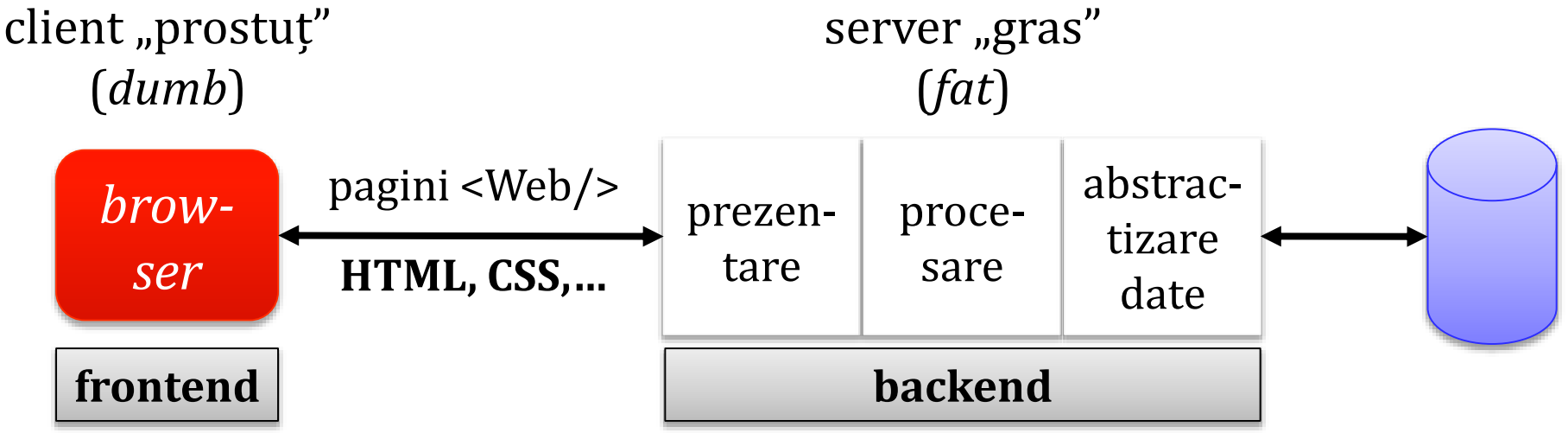
implementare

Server de aplicații Web

poate încuraja sau impune o viziune arhitecturală privind dezvoltarea de aplicații Web

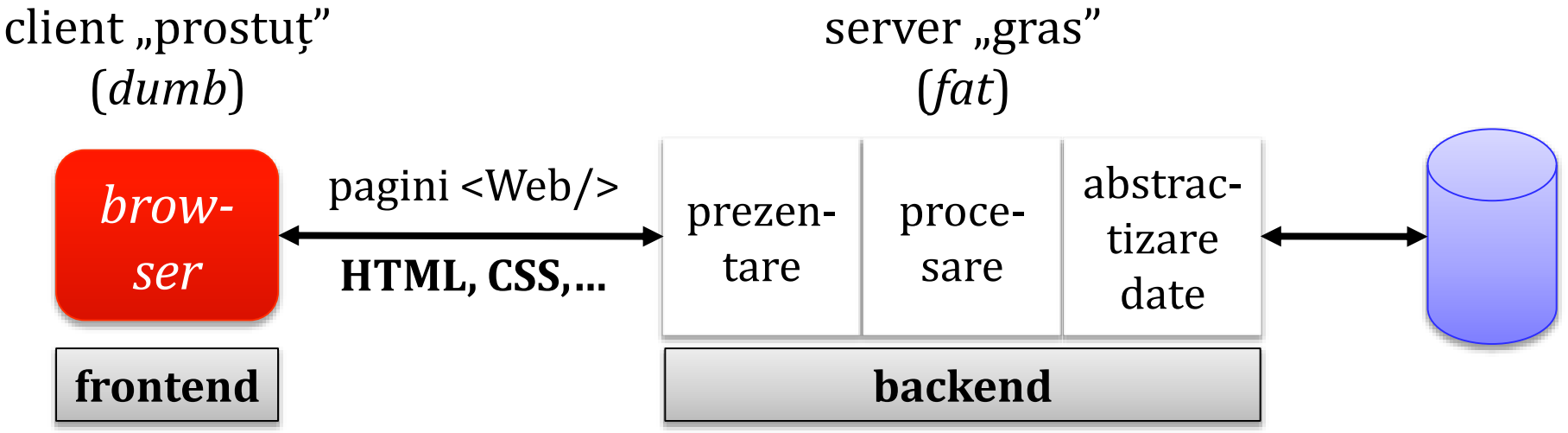
situație tipică:
MVC ori variații

arhitectura aplicațiilor Web: **abordarea MV* tradițională**



www.leaseweblabs.com/2013/10/api-first-architecture-fat-vs-thin-server-debate/

arhitectura aplicațiilor Web:
abordarea MV* tradițională



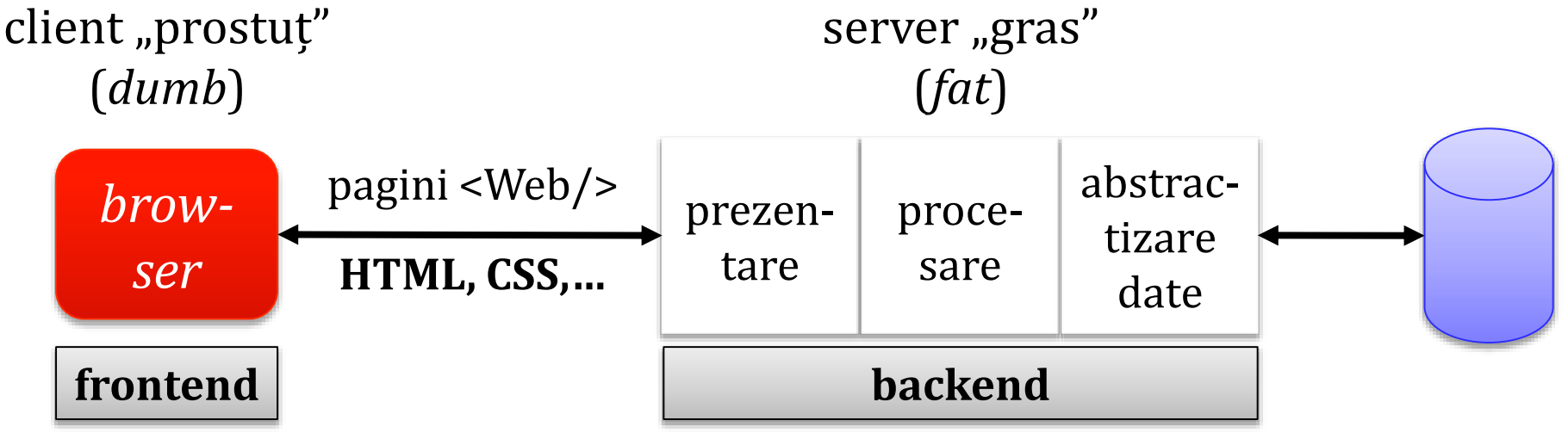
principiu de proiectare:

layers of isolation

modificările operate la un anumit strat nu au impact sau nu afectează componentele din alt strat

avansat

arhitectura aplicațiilor Web: **abordarea MV* tradițională**



frecvent, **aplicație monolitică**
(*e.g.*, un WAR: 2.2 M linii de cod, 418 .jar-uri,
startare în 12 min. – conform plainoldobjects.com)

implementare

Framework (cadru de lucru)

facilitează dezvoltarea de aplicații Web complexe, simplificând unele operații uzuale
(*e.g.*, acces la baze de date, *caching*, generare de cod, management de sesiuni, control al accesului) și/sau încurajând reutilizarea codului-sursă

avansat

implementare

Diverse *framework*-uri care facilitează dezvoltarea de aplicații Web la nivel de server:

ASP.NET: ASP.NET Core MVC, Vici MVC
Java: Play, Spring, Struts, Tapestry, WebObjects, Wicket
JavaScript (Node.js): Express, Geddy, Locomotive, Tower
Perl: Catalyst, CGI::Application, Jifty, WebGUI
PHP: CakePHP, CodeIgniter, Symfony, Yii, Zend Framework
Python: Django, Grok, web2py, Zope
Ruby: Camping, Nitro, Rails, Sinatra

implementare

Bibliotecă Web (*library*)

colecție de resurse computaționale reutilizabile
– *i.e.*, structuri de date + cod –
oferind funcționalități (comportamente) specifice
implementate într-un limbaj de programare

implementare

Bibliotecă Web (*library*)

colecție de resurse computaționale reutilizabile
– *i.e.*, structuri de date + cod –
oferind funcționalități (comportamente) specifice
implementate într-un limbaj de programare

poate fi referită de alt cod-sursă (software):
server de aplicații, *framework*, bibliotecă,
serviciu, API ori componentă Web

Biblioteci cu acces liber la codul-sursă – exemple:

Apache PDFBox – **pdfbox.apache.org**
Beautiful Soup – **www.crummy.com/software/BeautifulSoup**
D3.js – **d3js.org**
Expat – **libexpat.github.io**
ImageMagick – **www.imagemagick.org**
libcurl – **curl.haxx.se**
Libxml2 – **www.xmlsoft.org**
Lodash – **lodash.com**
OpenCV – **opencv.org**
Requests-HTML – **github.com/kennethreitz/requests-html**
zlib – **www.zlib.net**

implementare

Serviciu Web

software – utilizat la distanță de alte aplicații/servicii – oferind o funcționalitate specifică

implementarea sa nu trebuie cunoscută
de programatorul ce invocă serviciul

avansat

implementare

API (***Application Programming Interface***)

"any well-defined interface that defines the service that one component, module, or application provides to other software elements"
(de Souza *et al.*, 2004)

detalii în cursurile viitoare

implementare

SDK (***Software Development Kit***)

încapsulează funcționalitățile API-ului într-o bibliotecă (implementată într-un anumit limbaj de programare, pentru o platformă software/hardware specifică)

API façade pattern

exemplu: Octokit (pentru .NET, Objective-C, Ruby) oferit de Github – **developer.github.com/libraries/**

implementare

Web component

parte a unei aplicații Web
ce include o suită de funcții înrudite

e.g., calendar, cititor de fluxuri de știri,
buton de partajare a URL-ului în altă aplicație

Dr. Sabin Buraga ▲ profs.info.uaic.ro/~busaco/

implementare

Web component

dezvoltare bazată pe o bibliotecă/*framework* JavaScript

soluții – uzual, la nivel de client: Polymer, React, X-Tag,...

în lucru la Consorțiul Web (martie 2019)
github.com/w3c/webcomponents/

resurse + exemplificări: www.webcomponents.org

implementare

Widget

aplicație – de sine-stătătoare sau
inclusă într-un container (*e.g.*, un document HTML) –
ce oferă o funcționalitate specifică

rulează la nivel de client (platformă pusă la dispoziție
de sistemul de operare și/sau de navigatorul Web)

implementare

(*Web*) *app*

o aplicație (Web) instalabilă
care folosește API-urile oferite de o platformă:
browser, server de aplicații, sistem de operare,...

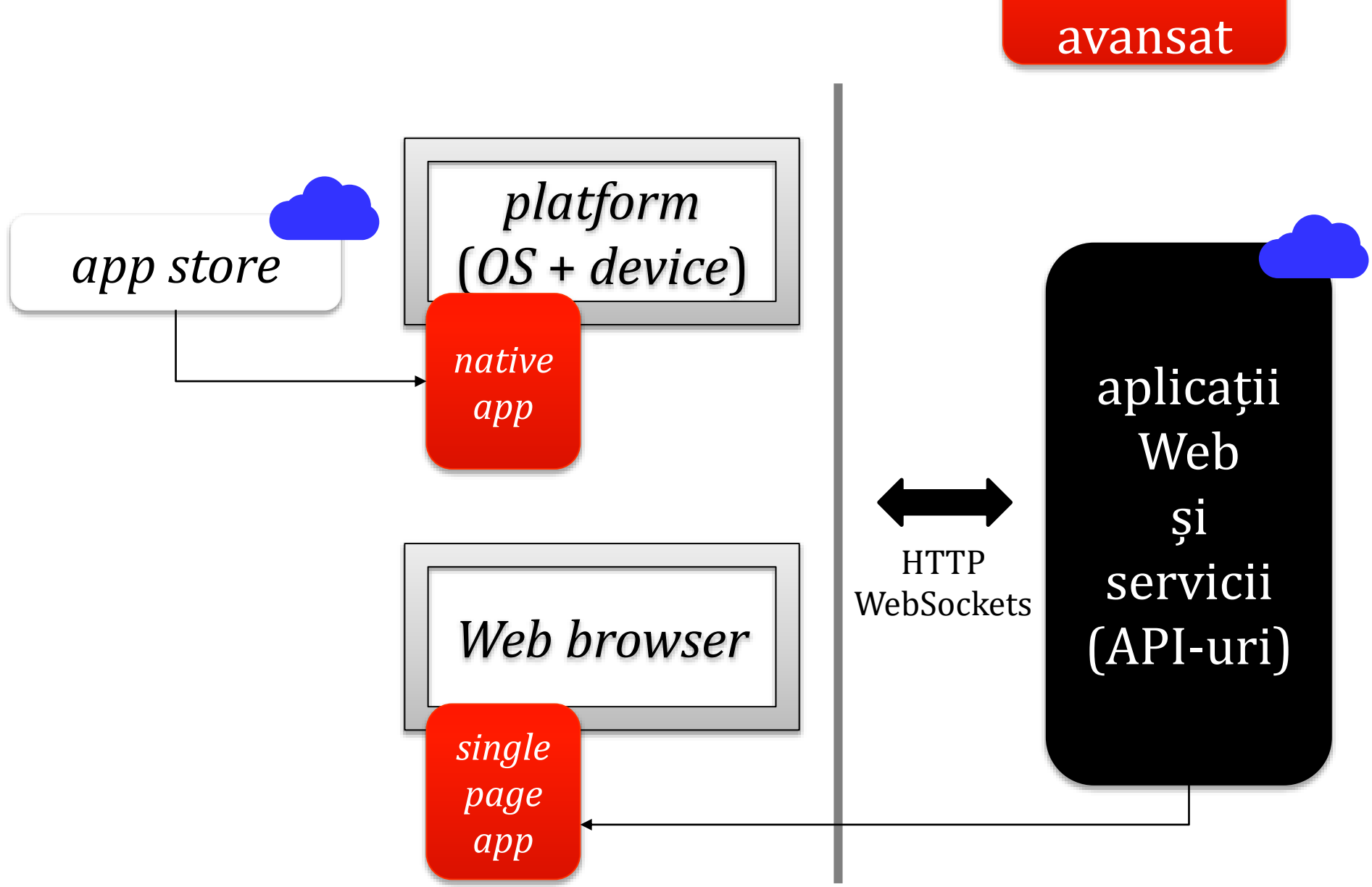
a distributed computer software application designed for
optimal use on specific screen sizes *and*
with particular interface technologies
Robert Shilston, 2013

implementare

(*Web*) *app*

uzual, se poate obține via un *app store*
(centralizat sau descentralizat)

exemple notabile:
Chrome Apps
aplicații Windows dezvoltate în JavaScript
aplicații Web mobile pentru Firefox, Kindle Fire,...



adaptare după Adrian Colyer (2012)

implementare

Add-on

denumire generică a aplicațiilor asociate unui *browser*
(extensii, teme vizuale, dicționare,
maniere de căutare pe Web, *plug-in*-uri etc.)

exemplificare: **addons.mozilla.org**

dezvoltare

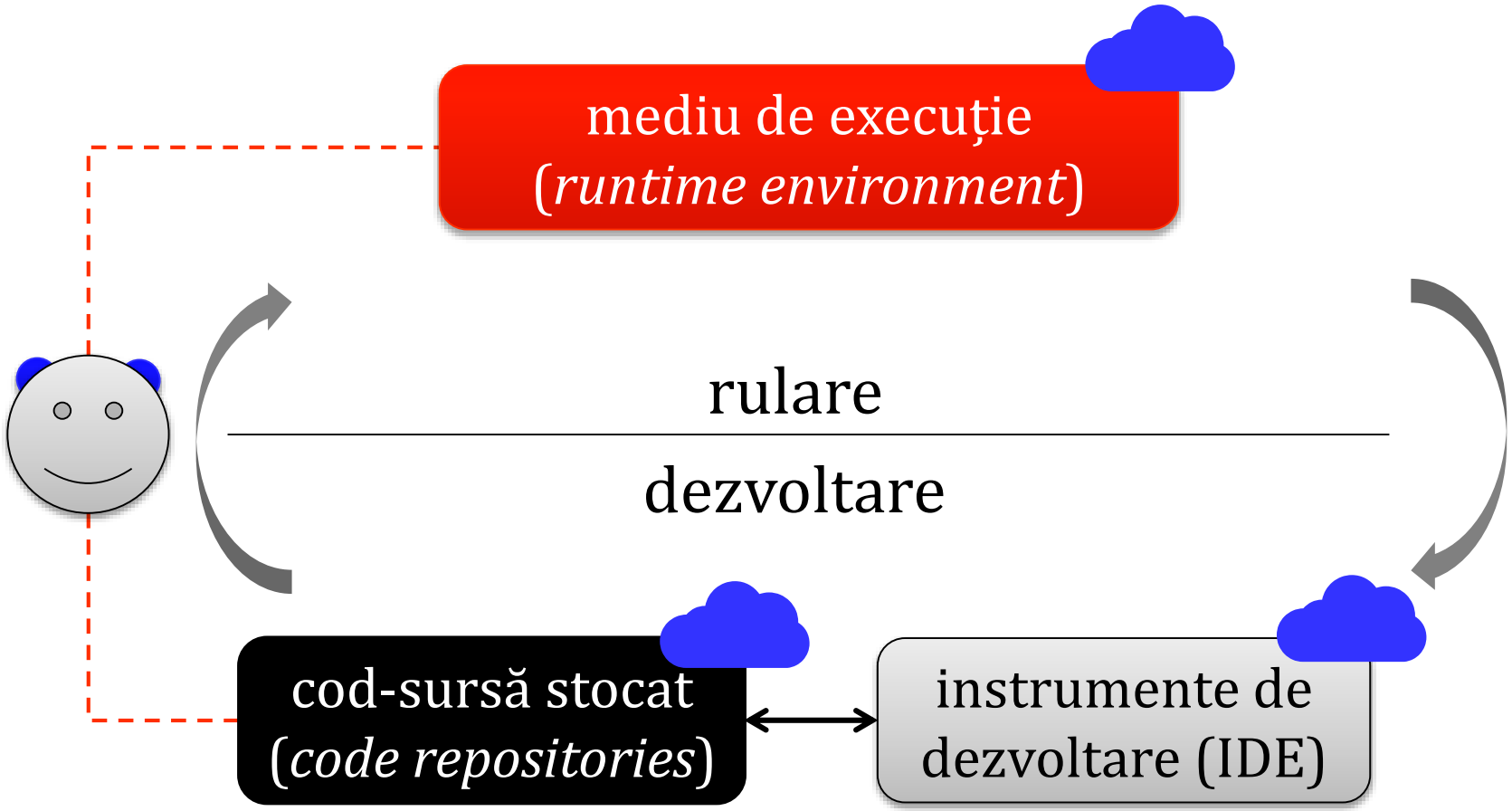
Recurgerea la medii de dezvoltare

exemplificări – aplicații native (pentru *desktop*):
Anjuta, Aptana Studio, Eclipse, Emacs, IntelliJ IDEA, KomodoIDE, Padre, PHPStorm, PyCharm, RubyMine, Visual Studio, Zend Studio

soluții bazate pe *cloud computing*:
AWS Cloud9, Codenvy, Koding etc.

S. Buraga, „Cu codul în nori”: **www.slideshare.net/busaco/cu-codul-n-nori**

avansat

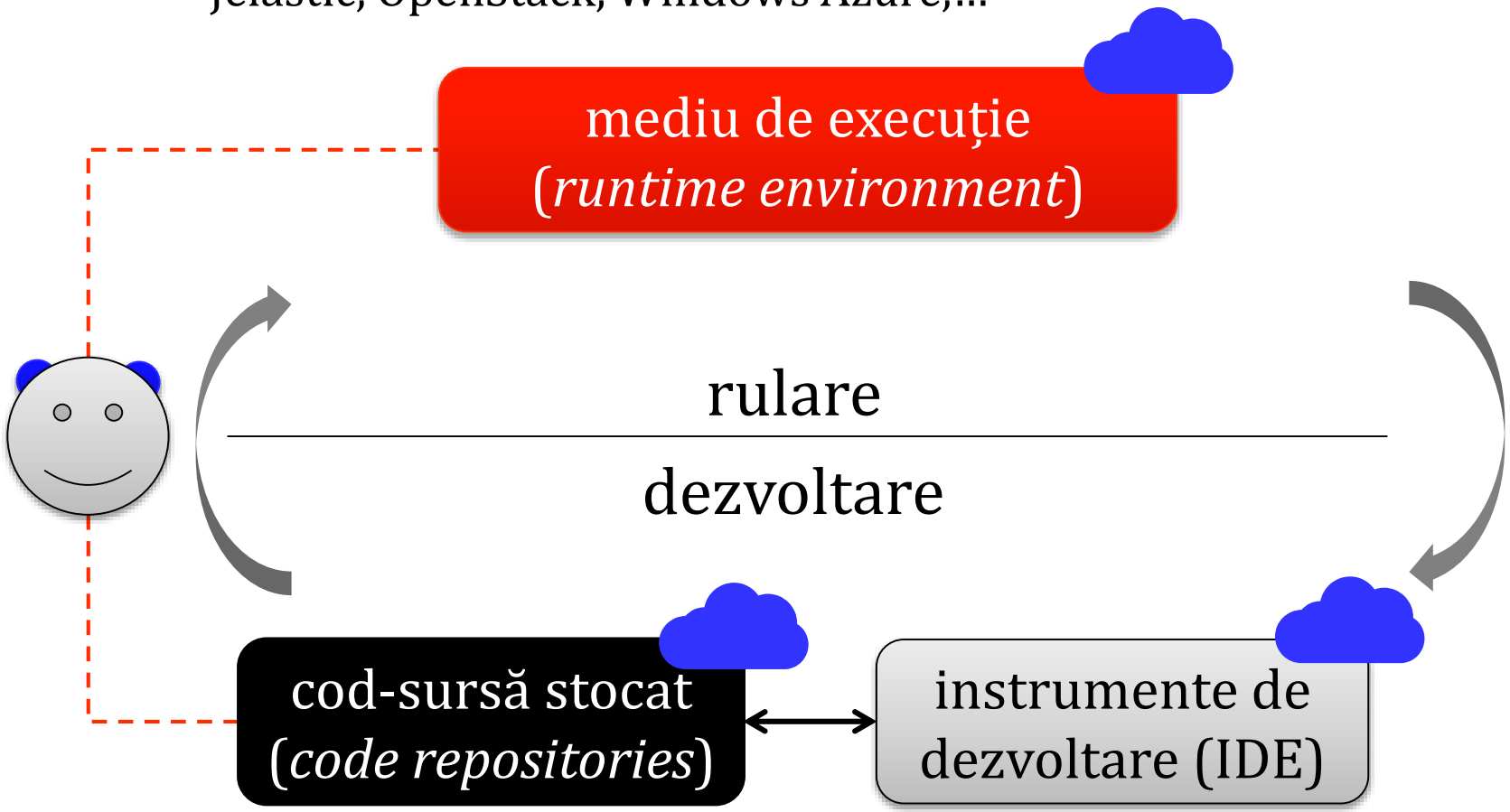


A. Iqbal, M. Haunsenblas, S. Decker (2012)

Development as a Service

avansat

DigitalOcean, Google Cloud Platform, Heroku,
Jelastic, OpenStack, Windows Azure,...



BitBucket,
GitHub

Web: AWS Cloud9, Koding, Ideone etc.
desktop: Eclipse, Visual Studio (Code),...

instrumente utile la **github.com/ripienaar/free-for-dev**

dezvoltare

Generarea automată de documentații,
în diverse formate

instrumente specifice (*documentation generators*)

exemplificări:
Doc, Document! X, Doxygen,
JavaDoc, JSDoc, phpDocumentor

dezvoltare

Controlul versiunilor surselor de programe
(VCS – *Version Control System*)

code review, *revision control*, *versioning*

monitorizarea modificărilor asupra codului-sursă
realizate de o echipă de programatori
asupra aceleași suite de programe (*codebase*)

Instrumente client/server:

Apache Subversion – SVN

Microsoft Team Foundation Server – TFS

Soluții distribuite:

Git (implementat în bash, C și Perl)

git-scm.com

Mercurial (dezvoltat în Python)

mercurial.selenic.com

Rational Team Concert (oferit de IBM)

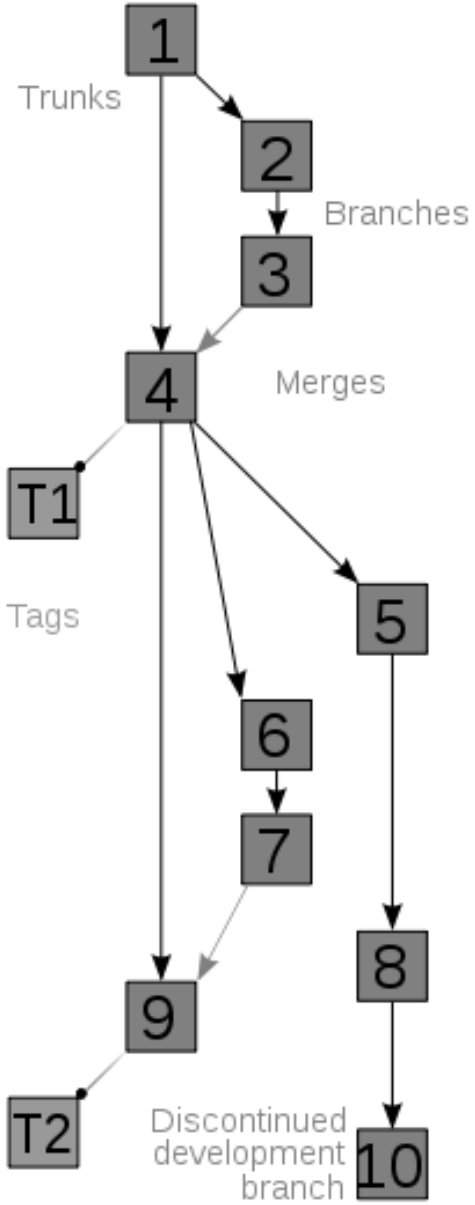
jazz.net/products/rational-team-concert/

Sisteme Web de găzduire de software (SCM – *source code management*):

BitBucket – **developer.atlassian.com/cloud/bitbucket**

GitHub – **developer.github.com**

GitLab – **about.gitlab.com/handbook/**



Încurajarea/impunerea unui stil de redactare a codului-sursă

la nivel de client:

HTML + CSS

www.oreilly.com/web-platform/free/files/little-book-html-css-coding-guidelines.pdf

JavaScript – profs.info.uaic.ro/~busaco/teach/courses/cliw/web-film.html#week9

la nivel de server:

C# – github.com/dennisdoomen/csharpguidelines

Perl – perldoc.perl.org/perlstyle.html

PHP – www.php-fig.org/psr/psr-2/

Python – www.python.org/dev/peps/

Ruby – github.com/styleguide/ruby

Scala – docs.scala-lang.org/style/

pentru altele, de considerat google.github.io/styleguide/

dezvoltare

Management de pachete software

căutare, instalare, compilare, verificare a dependențelor

exemplificări:
Bower, Composer, npm, NuGet, RubyGems, Yarn

de experimentat **github.com/showcases/package-managers**

dezvoltare

Suport pentru fluxuri de activități (*workflow*-uri)
eventual, realizate automat

„construirea” unei aplicații Web pornind
de la codul-sursă + componentele adiționale (*build tool*)

exemplificări:
Ant, Grunt, Gulp, make, Mimoza, Rake, tup, Yeoman

testare

Teste referitoare la codul-sursă

unități de testare automată – cadrul general dat de xUnit
HttpUnit, JUnit (Java), PHPUnit, xUnit.net (C#, F#),
Test::Class (Perl), unittest (Python), Unit.js
+
JSUnit, FireUnit, Mocha, Selenium etc.
la nivel de client

pentru amănunte, de parcurs **xunitpatterns.com**

testare

Teste specifice în contextul aplicațiilor Web

privind **conținutul** – structură, validare HTML, CSS,...

probleme la nivel de **hipertext** (*e.g.*, *broken links*)

utilizabilitate – inclusiv accesibilitate, multi-lingvism

estetica interfeței Web – dificil de evaluat/testat

testare

Teste specifice în contextul aplicațiilor Web

integrare a componentelor

gradul de **disponibilitate** permanentă și de flexibilitate
(evoluție continuă)

gradul de **independență** de dispozitiv – *multi-screen*
(număr mare de dispozitive + caracteristici potențiale)

testare

Alte tipuri de testări:

privind **performanța**

încărcare (*load*), *stressing*, testare continuă, scalabilitate

studii de caz reale:

High Scalability – **highscalability.com**

Performance Failures – **perf.fail**

Performance Planet – **calendar.perfplanet.com**

Web Performance Stats – **wpostats.com**

testare

Alte tipuri de testări:

referitoare la **securitate**

testare: exemplificare

Documente HTML – serviciul **validator.w3.org**
Foi de stiluri CSS – CSS Lint: **csslint.net**
Date JSON – validare via JSONLint
Documente XML – bine-formatate / valide
Script-uri pe partea client (JavaScript) via JS/ES Hint
Programe rulate la nivel de server – *e.g.*, xUnit
Integritatea și accesul la sisteme de fișiere
Integritatea și accesul la sisteme de baze de date
Suport oferit de navigatorul Web – **caniuse.com**
Probleme de securitate – **www.owasp.org**
Aspecte vizând performanța aplicațiilor Web

exploatare

Publicarea sitului

server dedicat

versus

furnizor de găzduire Web (*hosting*)

soluție gratuită *vs.* comercială

timp de răspuns, scalabilitate, securitate, suport tehnic,...

exploatare

Mentenanța (administrarea) conținutului

obținerea, crearea, pregătirea, managementul, prezentarea, procesarea, publicarea și reutilizarea conținuturilor în manieră sistematică și structurată

exploatare: management

La nivel organizațional:

managementul cunoștințelor (*knowledge management*)

managementul relațiilor cu clienții
(CRM – *Client Relationship Management)*

planificarea resurselor
(ERP – *Enterprise Resource Planning*)

managementul *workflow*-urilor + *business rules*

integrarea aplicațiilor (EAI – *Enterprise App Integration*)

exploatare: management

La nivel tehnic:

managementul conținutului de către personal non-tehnic
pe baza principiului *separation of concerns*

sisteme de management al conținutului
(CMS – *Content Management Systems*)

instrumente colaborative
(*e.g., enterprise wiki*)

avansat

exploatare: management

Privind utilizatorul:

interacțiune Web – *e.g.*, utilizabilitate
profs.info.uaic.ro/~busaco/teach/courses/hci/

șabloane de proiectare a aplicațiilor Web sociale
profs.info.uaic.ro/~busaco/teach/courses/hci/hci-film.html#week7

performanța Web la nivel de *browser*
profs.info.uaic.ro/~busaco/teach/courses/cliw/web-film.html#week13

exploatare: analiza utilizării

Usage analysis

metode explicite

bazate pe date oferite de utilizator

e.g., chestionare și monitorizare (*user testing*),
analiza mesajelor de *e-mail*, reacții pe rețele sociale etc.

metode implicite

colectare automată a datelor de interes (*user analytics*)

uzual, folosind *cookie*-uri

exploatare: analiza utilizării

Usage analysis

construirea profilului utilizatorilor: *Web usage mining*

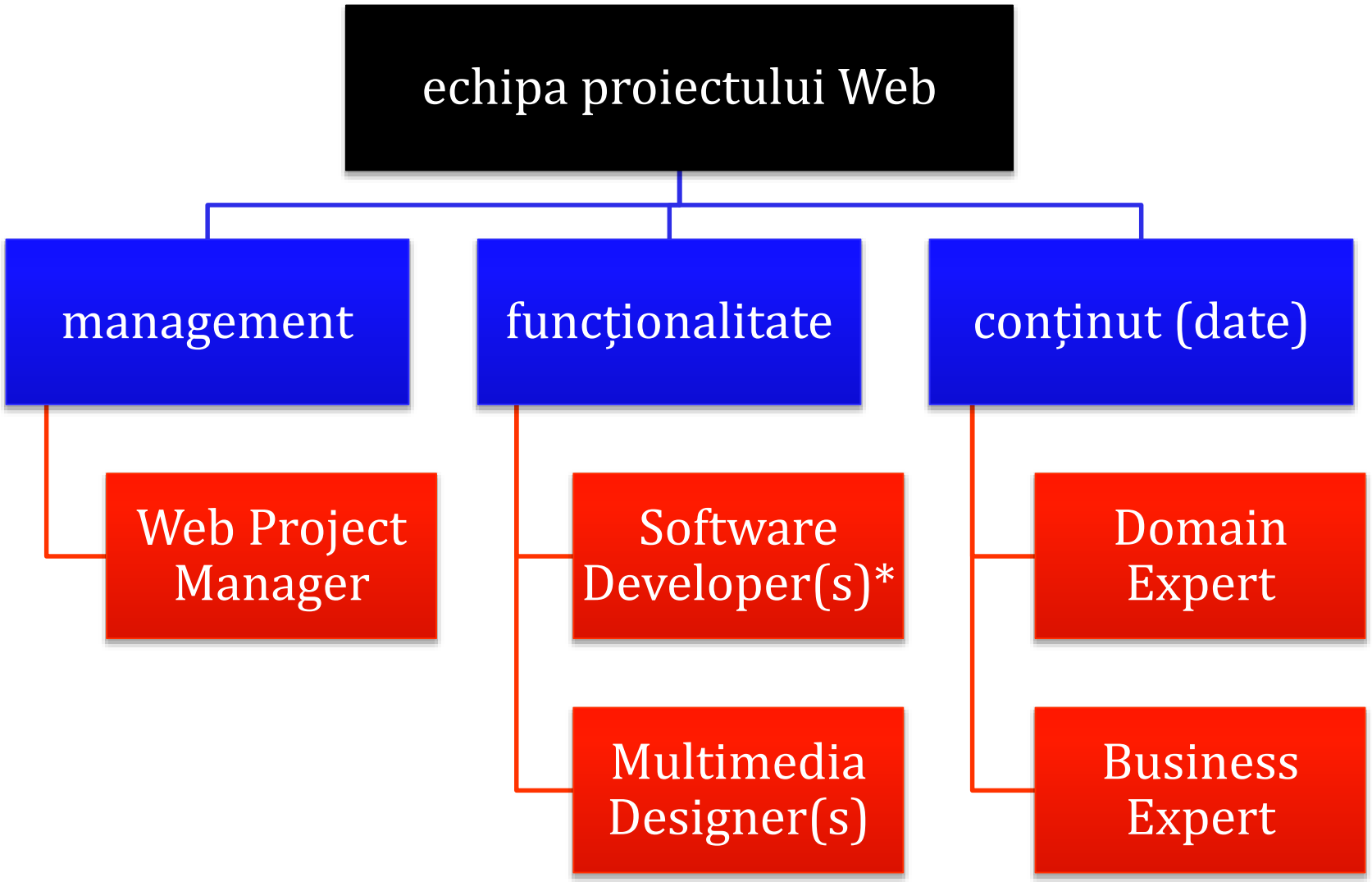
analiza fișierelor de jurnalizare a accesului
(*e.g.*, access.log la Apache, AWStats,...)

măsurarea „popularității” sitului: viteză de încărcare,
numărul de accesări, timpul + durata de vizitare etc.

servicii de monitorizare/raportare
exemple: Google Analytics, WordPress Statistics

parametrii unui proiect web

obiectiv principal
durată
cost
abordare
tehnologii
procese
rezultat
resurse umane
profilul echipei



**frontend* sau *backend* sau *full-stack* (*frontend* + *backend*)

www.slideshare.net/busaco/sabin-buraga-dezvoltator-web-n-2017

Câteva exemplificări privind arhitectura unor aplicații Web?

studiu de caz: flickr

Scop:
partajare *on-line* a conținutului grafic (fotografii)

aplicație reprezentativă a Web-ului social

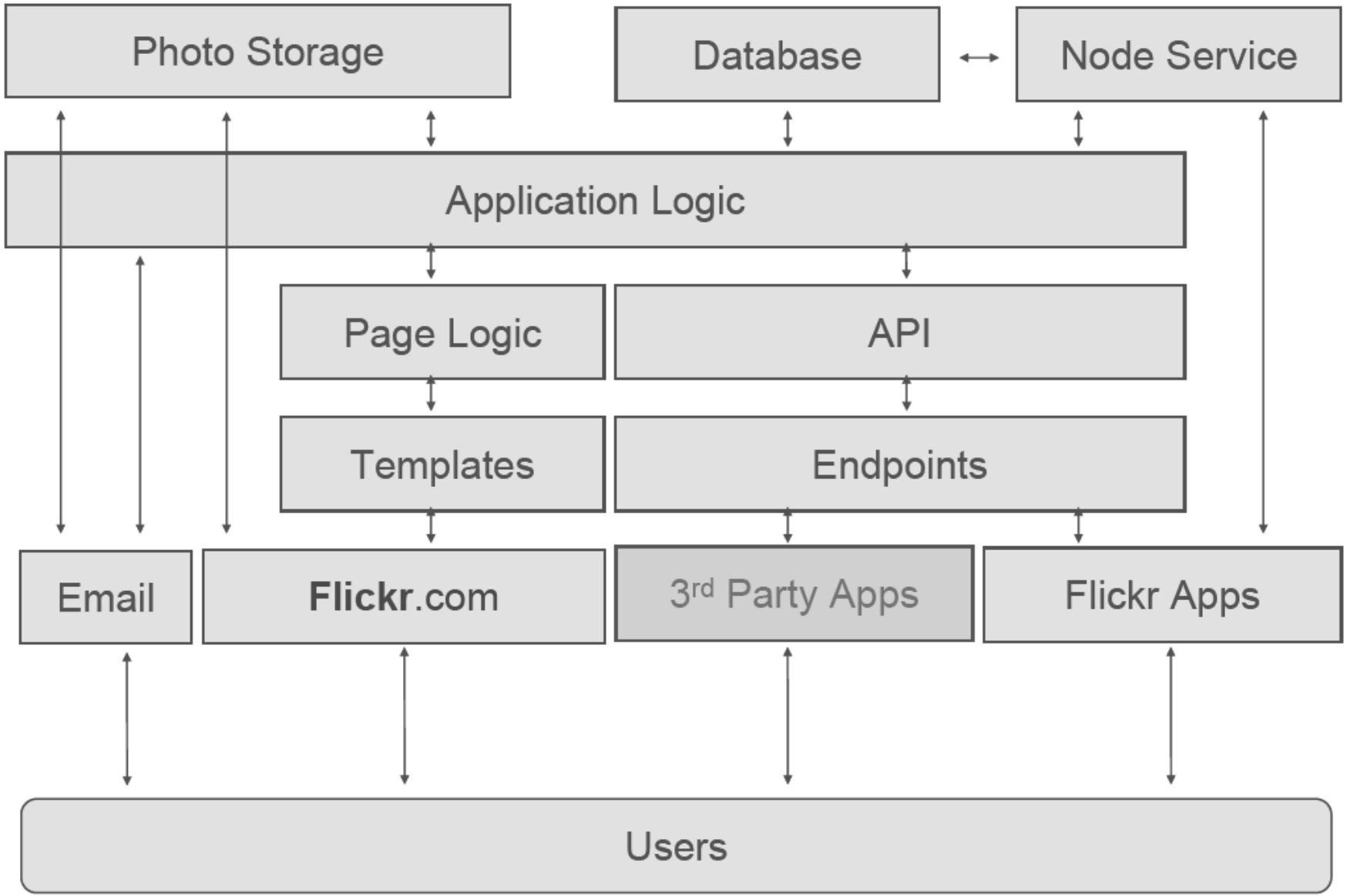
agregare de comunități – imaginea ca obiect social

suport pentru adnotări via termeni de conținut (*tagging*)
+ comentarii

studiu de caz: flickr – tehnologii

PHP (procesare – *application logic*, acces la API, prezentare de conținut via Smarty, modul de *e-mail*)
Perl (validarea datelor)
Java (managementul nodurilor de stocare)
MySQL (stocare în format InnoDB)
ImageMagick (bibliotecă C de prelucrare de imagini)
Ajax (interacțiune asincronă)
Linux (platformă de rulare)

alte detalii la **highscalability.com/flickr-architecture**



arhitectura inițială – conform (Cal Henderson, 2007)

avansat

API Kits

C
- Flickcurl

Cold Fusion
- CFlickr

Common Lisp
- Clickr

cUrl
- Curlr

Delphi
- dFlickr

Go
- go-flickr

Java
- Flickr4Java
- flickr-jandroid

.NET
- Flickr.NET

Node.js
- node-flickrapi

Objective-C
- ObjectiveFlickr
- FlickrKit

Perl

interfețe de programare (API-uri)
oferite de Flickr

facilitează accesul la serviciile Web
în cadrul aplicațiilor rulând
pe platforme variate

cereri via REST, XML-RPC, SOAP
răspunsuri REST, XML-RPC, SOAP, JSON

www.flickr.com/services/api/

aspecte generice vizând proiectarea sistemului:

categorii de resurse: *user* + *picture*

relații între instanțe de tip *user* – *e.g.*, *follow*

relații între instanțe de tip *user* și *picture*
(*make*, *depicts*, *comment*, *like*,…)

asigurarea performanței:
timp de răspuns, arhitectură software scalabilă,
stocare persistentă scalabilă, optimizarea imaginilor

recomandarea resurselor (*user*/*picture*) de interes

detalii în articolul *Create a Photo Sharing App* (2016)
blog.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/

studiu de caz: lanyrd

Scop: descoperire și management *online* de evenimente (*e.g.*, conferințe cu caracter tehnologic)

agregare de comunități – evenimentul ca obiect social

suport pentru vorbitori și audiență, *slide*-uri,…
+ calendare și localități de desfășurare

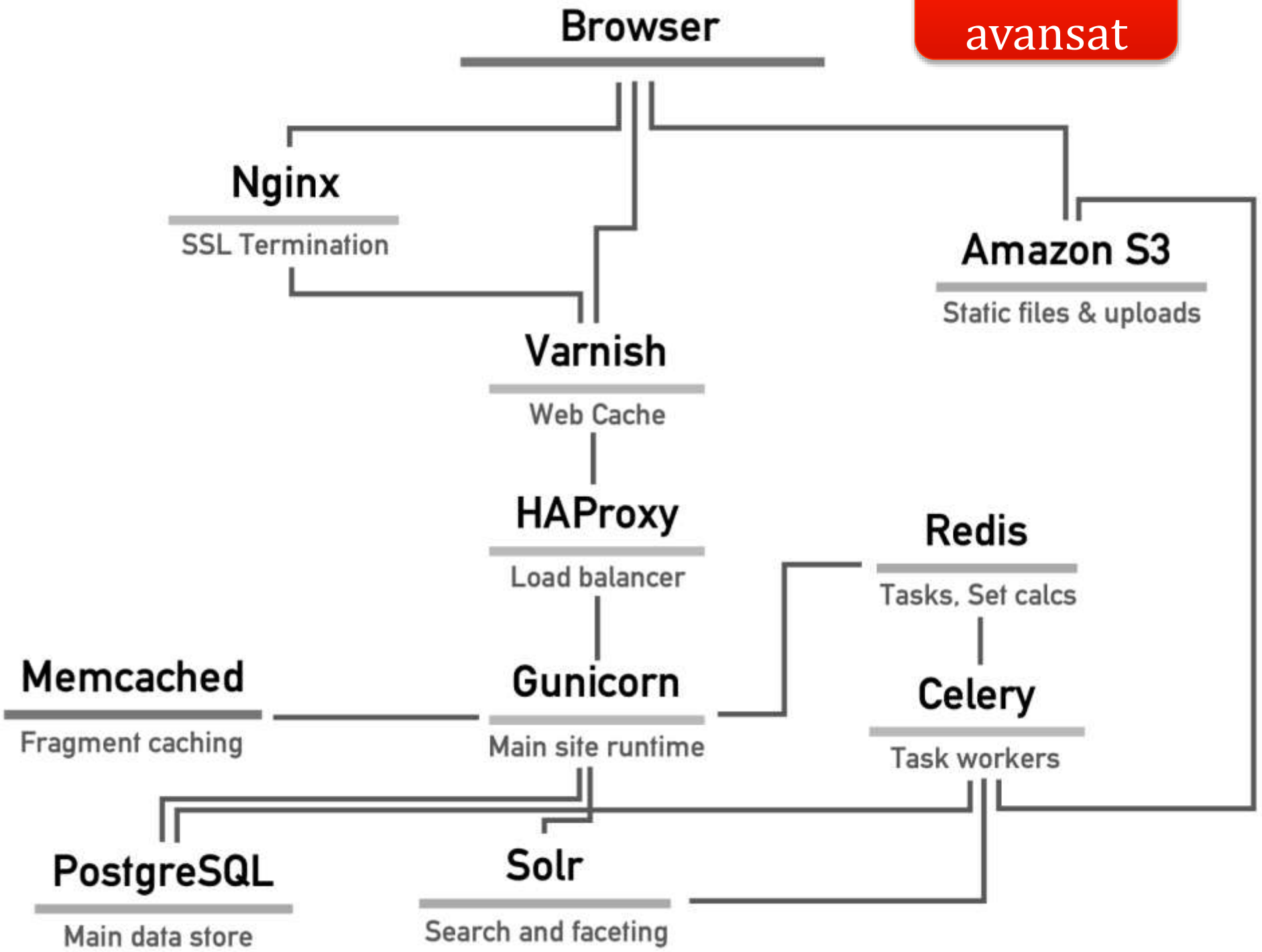
concepte importante: *conferences*, *user profiles*, *e-mails*, *dashboard*, *coverage*, *topics*, *guides*

studiu de caz: lanyrd

Creat aproape complet în Python (folosind Django)
și întreținut de 6 persoane

2½	*backend developers*
1¾	*frontend developers*
½	*mobile developers*
1½	*designers*
¾	*system administrators*
¾	*business operations*

A. Godwin, *Inside Lanyrd's Architecture*, QCon London, 2013
www.infoq.com/presentations/lanyrd-architecture


Browser
avansat
Nginx
SSL Termination
Amazon S3
Static files & uploads
Varnish
Web Cache
HAProxy
Load balancer
Redis
Tasks, Set calcs
Memcached
Fragment caching
Gunicorn
Main site runtime
Celery
Task workers
PostgreSQL
Main data store
Solr
Search and faceting

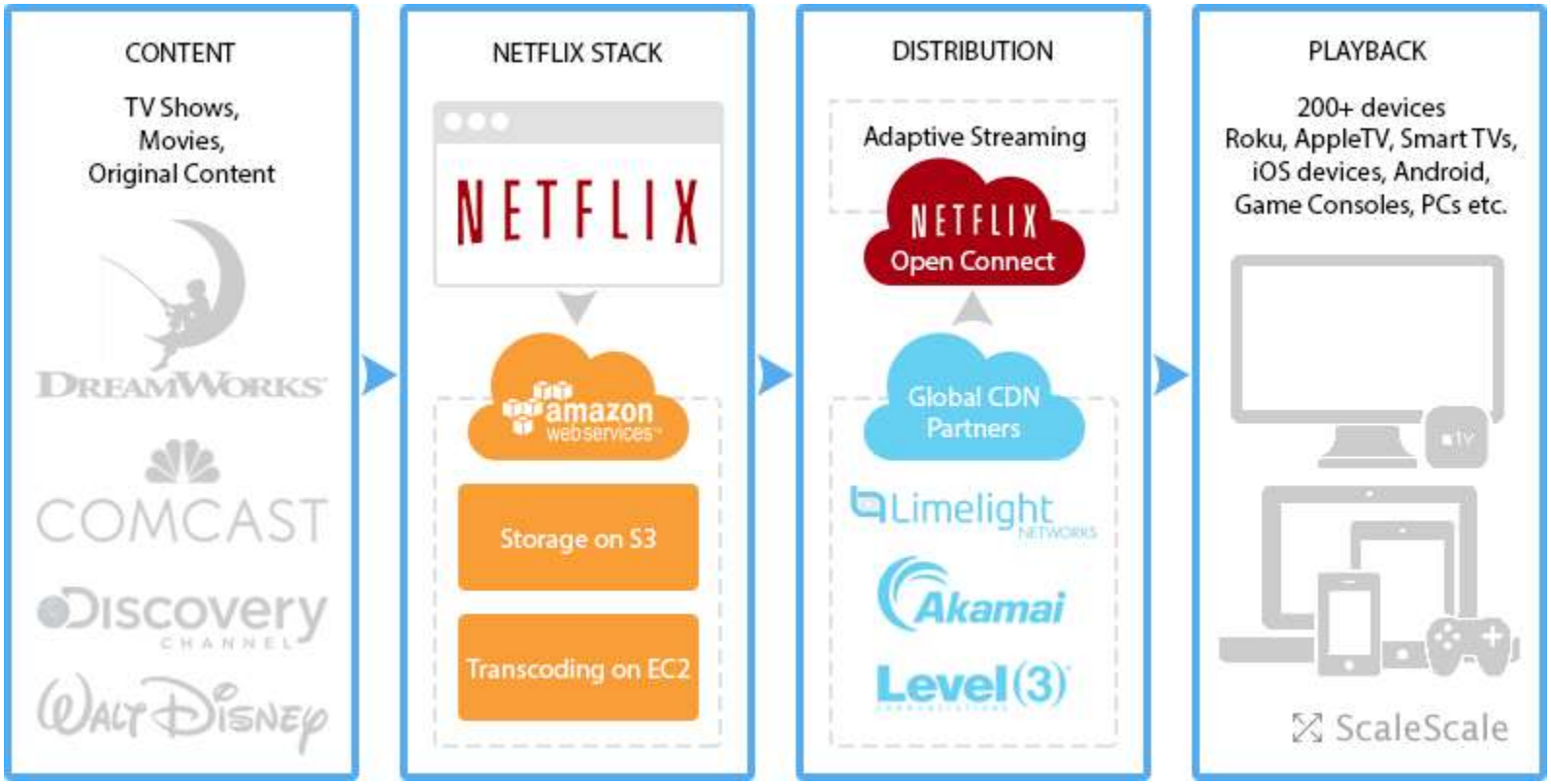
studiu de caz: netflix

Scop: oferire de conținut video la cerere (*streaming*) + televiziune Web (Web TV)

servicii disponibile pe dispozitive/platforme multiple

exploatare „în nori”

recurge și la tehnologii deschise



aspecte de interes:

conținut – filme, emisiuni TV,…

stocare + (de)codificare (*transcoding*)

difuzare adaptivă (*adaptive streaming*)

redare (*playback*) – suport pentru dispozitive eterogene

procesare *backend*	Java, Python, Node.js (JavaScript)
procesare *frontend*	React, winjs (JavaScript)
sisteme de stocare	MySQL, Apache Cassandra, Apache Hadoop, Apache Hive, Oracle DB
servicii în „nori”	Amazon EC2 (procesare video) Amazon S3 (stocare)
servicii SQL	Amazon RDS (*Relational DB Service*)
servicii NoSQL	Amazon DynamoDB
management de cod	GitHub (implementat în Ruby + C)
integrare continuă	Jenkins (implementare Java)
gestionare servere	Apache Mesos (implementare C++)
distribuire de conținut (*content distribution network*)	Open Connect CDN (FreeBSD, Nginx), Akamai, Level 3, Limelight
monitorizare	Boundary, LogicMonitor, Vector,…

highscalability.com/blog/2015/11/9/a-360-degree-view-of-the-entire-netflix-stack.html
stackshare.io/netflix/netflix

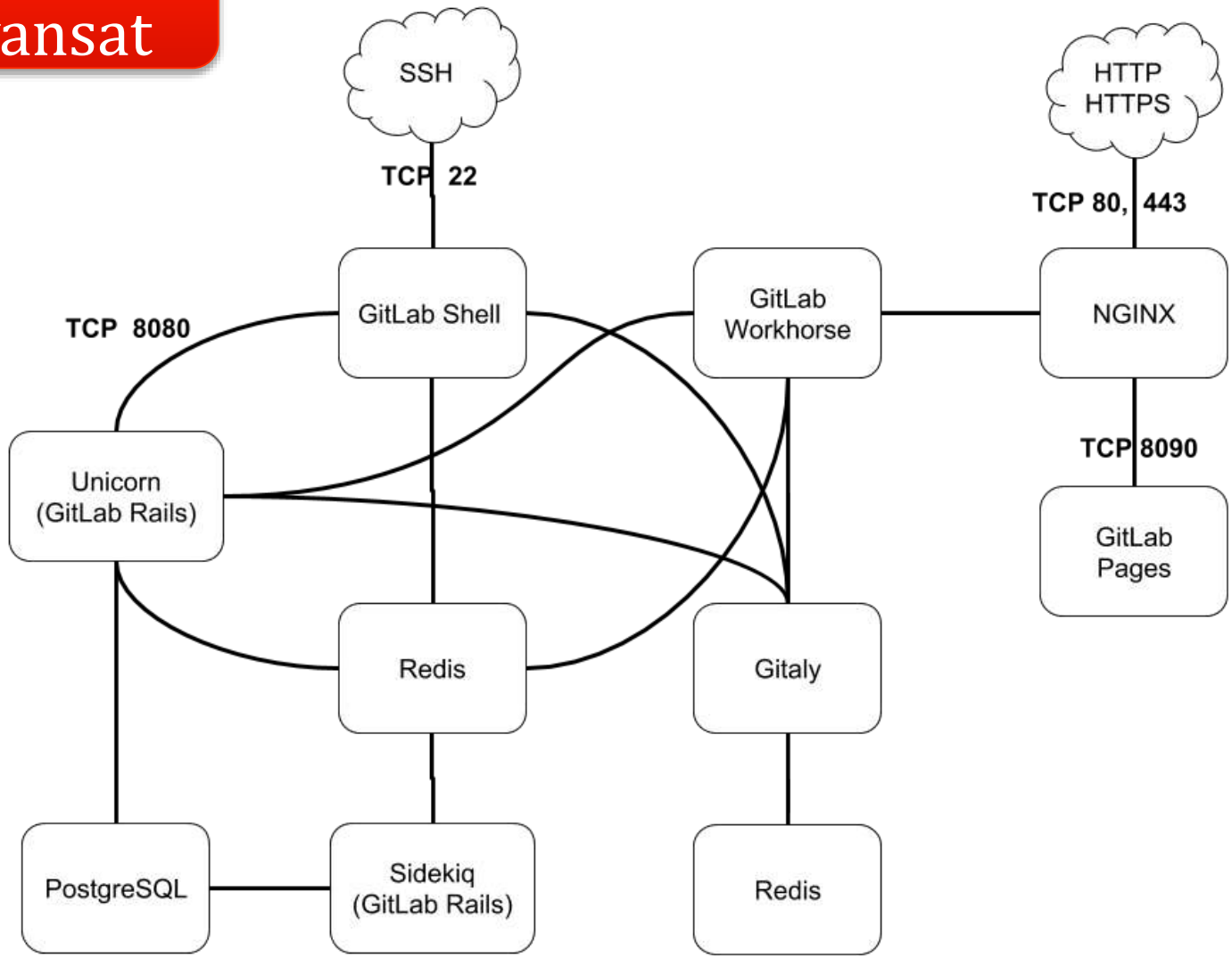
studiu de caz: gitlab

Scop: aplicație Web pentru managementul ciclului de dezvoltare software via Git
de la planificarea proiectului și gestiunea codului-sursă
până la integrare continuă și monitorizare

distribuit liber (*community edition*) – instalabil pe Linux –
ori prin subscripții (*enterprise edition*)

docs.gitlab.com/ee/development/architecture.html
about.gitlab.com/handbook/engineering/infrastructure/production-architecture/

avansat



reverse proxy: NGINX • server Web: Unicorn • stocare persistentă: PostgreSQL (utilizatori, meta-date), Redis (sesiuni Web, *cache*, cozi de mesaje) • acces la Git: Gitaly • procesare cozi de mesaje: Sidekiq • monitorizare (metrici): Prometeus • infrastructură: Terraform

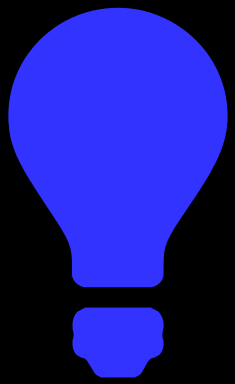
Aplicație Web	Procesare la nivel de server (*backend*)	Stocare persistentă
Amazon	Perl, Java	MySQL (MariaDB), Amazon DynamoDB, Amazon SimpleDB, Amazon ElastiCache
Coursera	Django (Python), Node.js (JavaScript), Play (Scala)	MySQL, Apache Cassandra
DuckDuck Go	Node.js, Perl	PostgreSQL
Facebook	Hack, PHP (HHVM), Tornado (Python), Java, JavaScript	RocksDB, Presto, Cassandra, Beringei
Google	C++, Dart, Go, Java, Python	BigTable, MariaDB
Linkedin	Grails (Java), JavaScript, Scala	MySQL, Oracle DB, RocksDB, Hadoop

stackshare.io/stacks

Aplicație Web	Procesare la nivel de server (*backend*)	Stocare persistentă
Lyft	PHP, Flask (Python), Java, Go, C++	MongoDB, DynamoDB, Redis
Medium	Node.js, Go	Neo4j, DynamoDB, Redis
Pinterest	Django (Python), Java, Go	MySQL, Hadoop, Apache HBase, Redis, Memcached
Stack Overflow	.NET Framework (C#)	MS SQL Server, Redis
Sound Cloud	Clojure, Scala, JRuby	MySQL
Wikipedia	PHP (HHVM), Node.js	MySQL

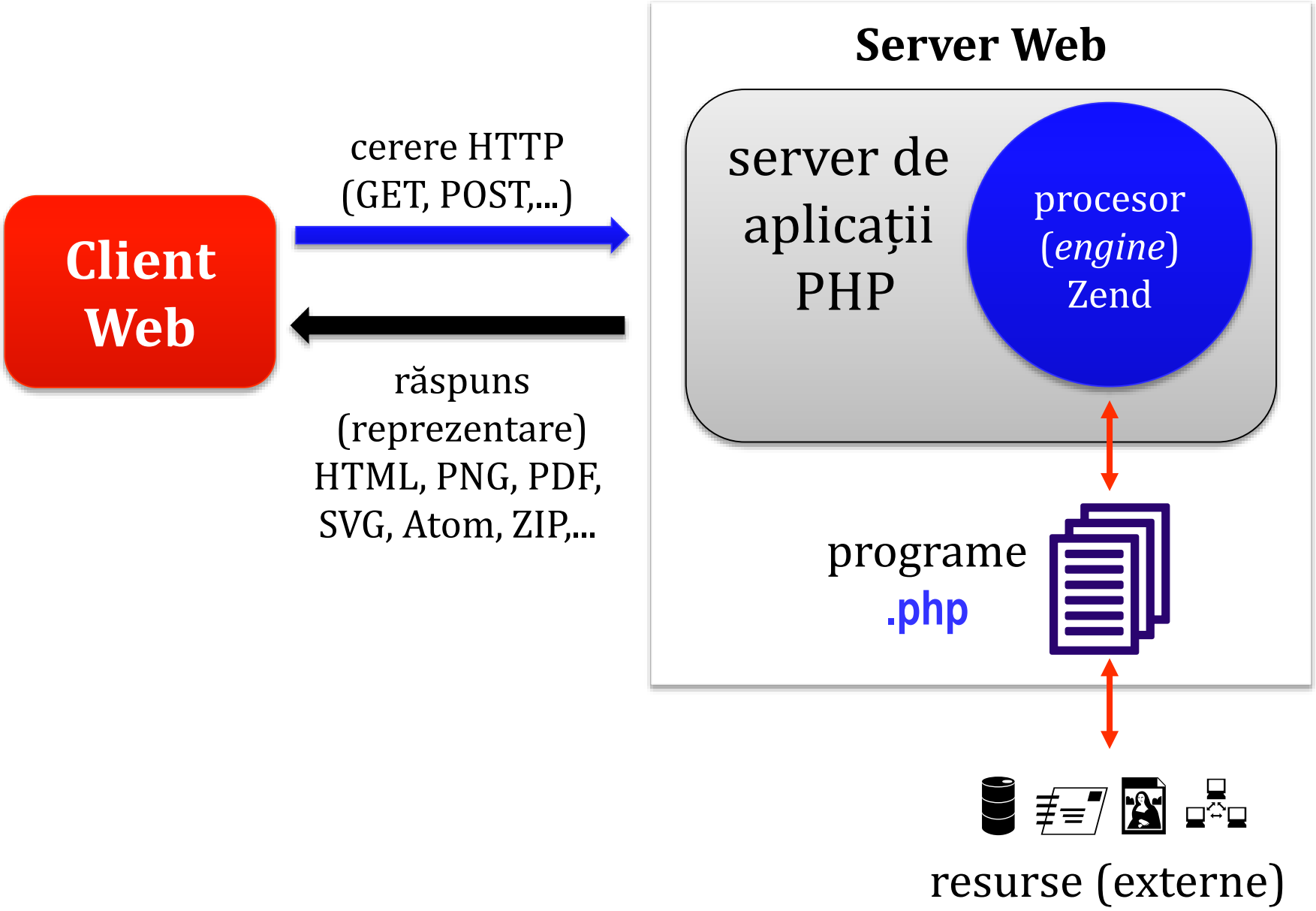
stackshare.io/stacks

rezumat



programare Web ▸ inginerie Web

dezvoltarea aplicațiilor Web – aspecte esențiale



episodul viitor:

dezvoltarea de aplicații Web în PHP