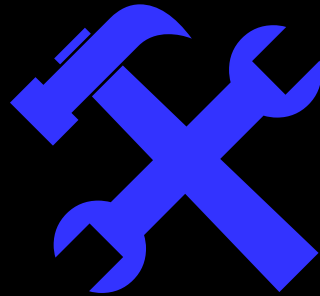


Tehnologii Web

programare Web ► inginerie Web



dezvoltarea aplicațiilor Web

design patterns, servere de aplicații, arhitecturi

“Viitorul este suma pașilor pe care-i faceți,
inclusiv a celor mici, ignorați sau luați în râs.”

Henri Coandă

Aplicații Web \equiv sisteme software complexe,
în evoluție permanentă

Realitate

mijloace multiple de interacțiune Web cu utilizatorul



mobil



laptop



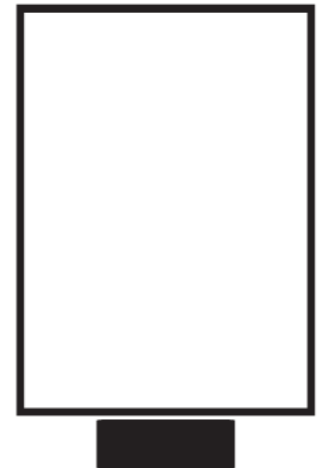
PC



tabletă



(*smart*) TV



ecran urban

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, SDK-uri, biblioteci, *framework*-uri,...)
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

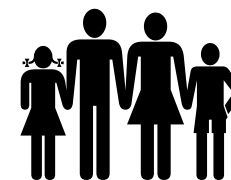
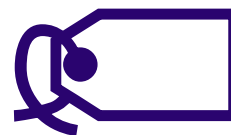
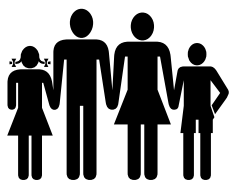
scopuri
psihologie
comportament

interacțiune
controale
limbi naturale

funcționalități
tehnologii
algoritmi

indexare
structurare
meta-date

instrumente
metodologii
stimuli



utilizatori

interfață

software

conținut

creatori

adaptare după Crumlish & Malone, 2009

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

Asigurarea calității aplicațiilor Web

alte aspecte de interes:

integritate

reparabilitate

verificabilitate – inclusiv monitorizare (*logging*)

economie

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**

modelare

Uzual, se recurge la o **metodologie**

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

www.omg.org/mda/

modelare

Metodologii orientate spre modele referitoare la:

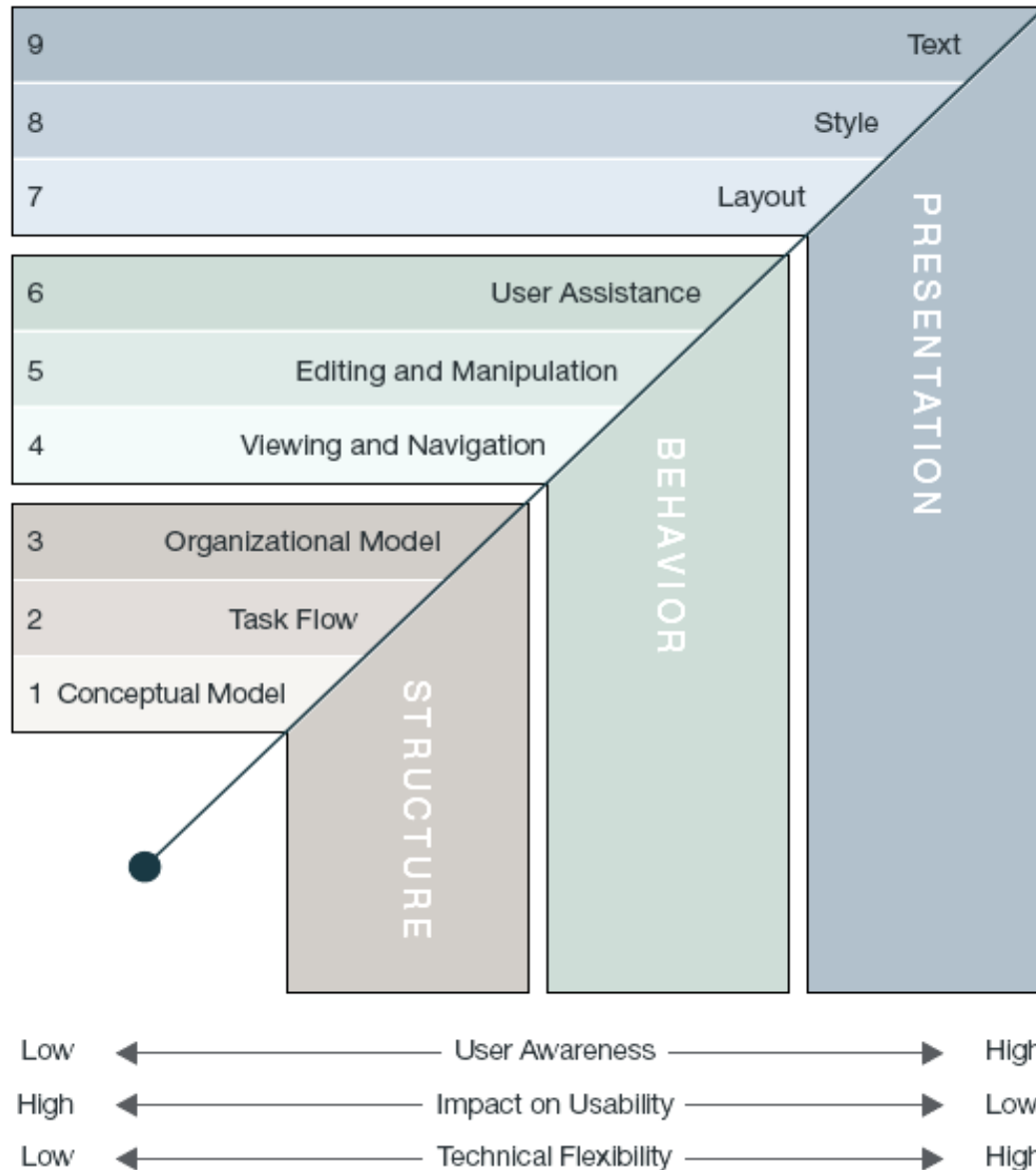
date – RMM (*Relationship Management Methodology*),
Hera, WebML

interacțiune – HDM (*Hypertext Design Model*),
WSDM (*Web Site Design Method*), UsiXML

obiecte – OOHDM (*Object-Oriented HDM*),
UWE (*UML-based Web Engineering*),
OOWS (*Object-Oriented Web Solutions*)

software – WAE (*Web Application Extension*)

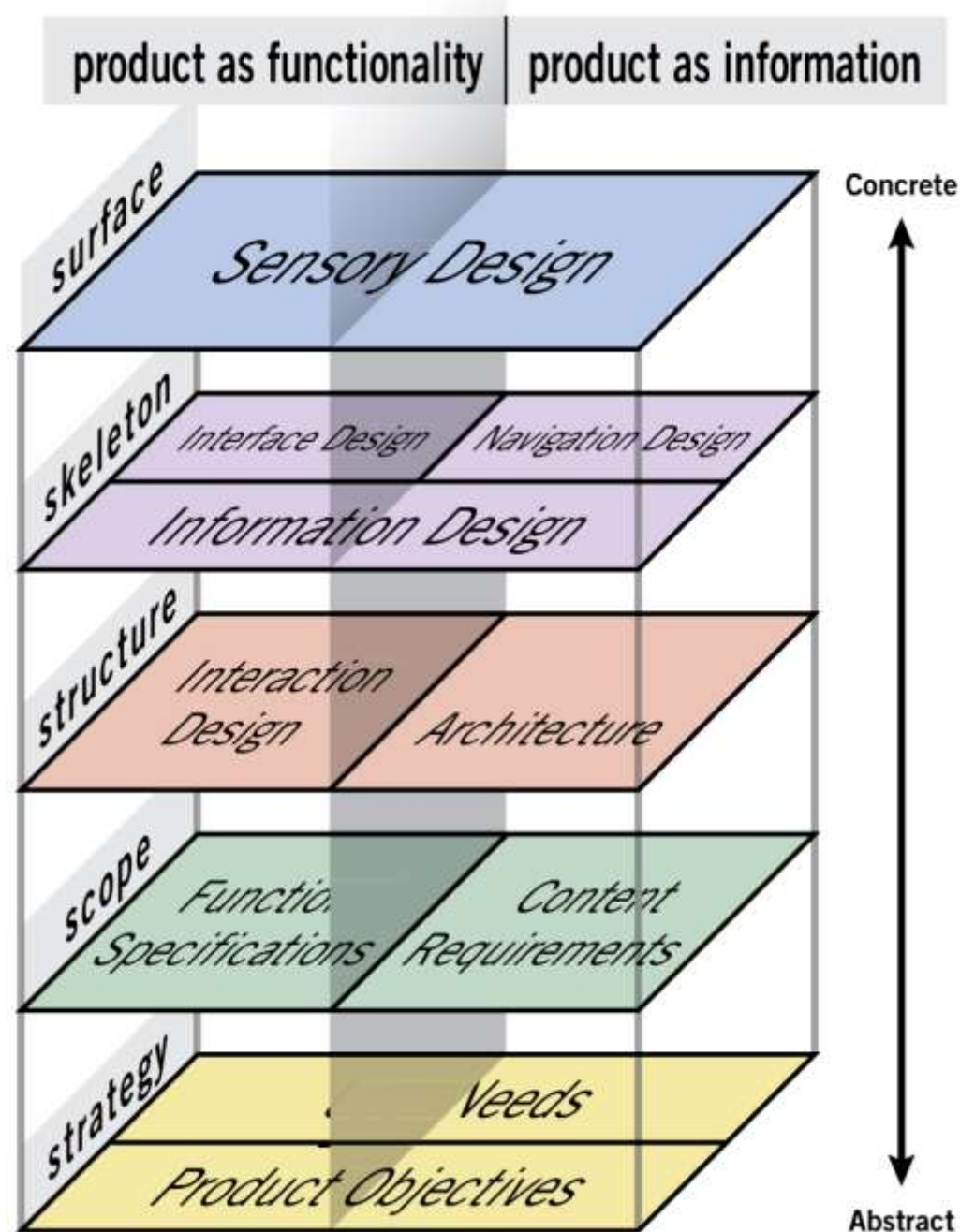
avansat



Robert Baxley

aplicație Web
(produs software)

funcționalitate
+
informații oferite



dezvoltarea aplicațiilor Web

Cerințe (*requirements*)

Analiză & proiectare (*software design*)

Implementare (*build*)

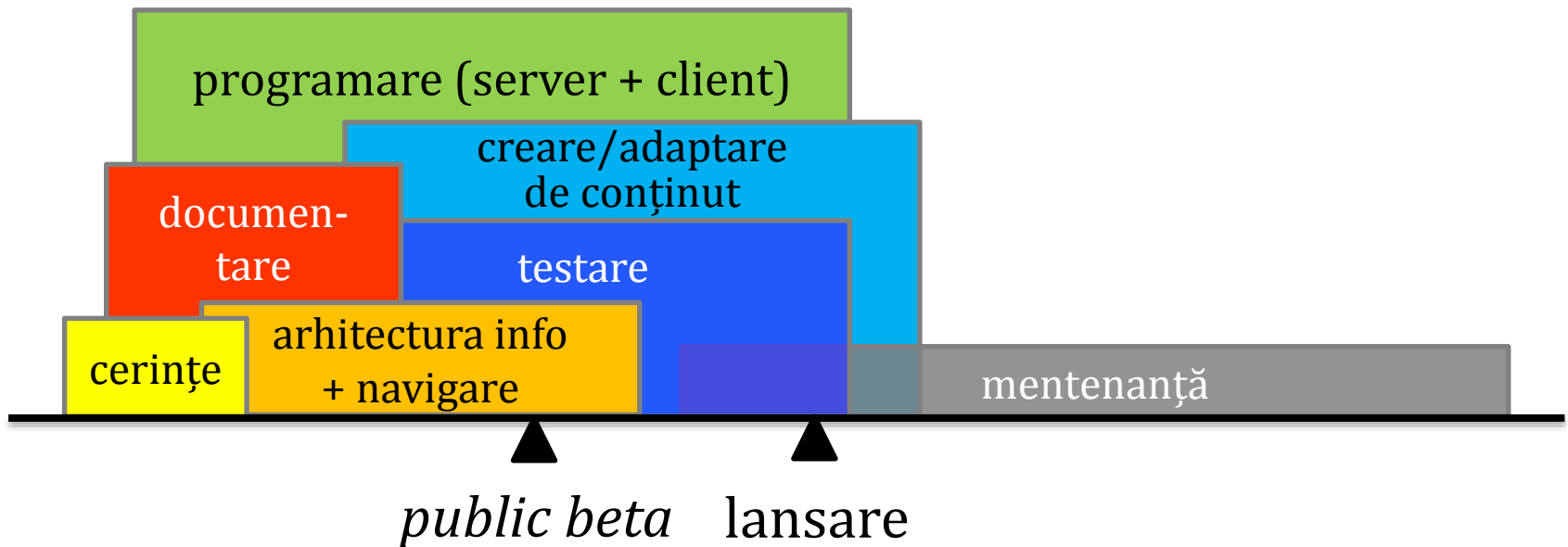
Testare (*testing*)

Exploatare (*deployment*)

Mentenanță (*maintenance*)

Evoluție (*evolution*)

dezvoltarea aplicațiilor Web



actualmente, sunt preferate **metodologii agile**

<http://www.infoq.com/process-practices/>

<http://sixrevisions.com/web-development/agile/>

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

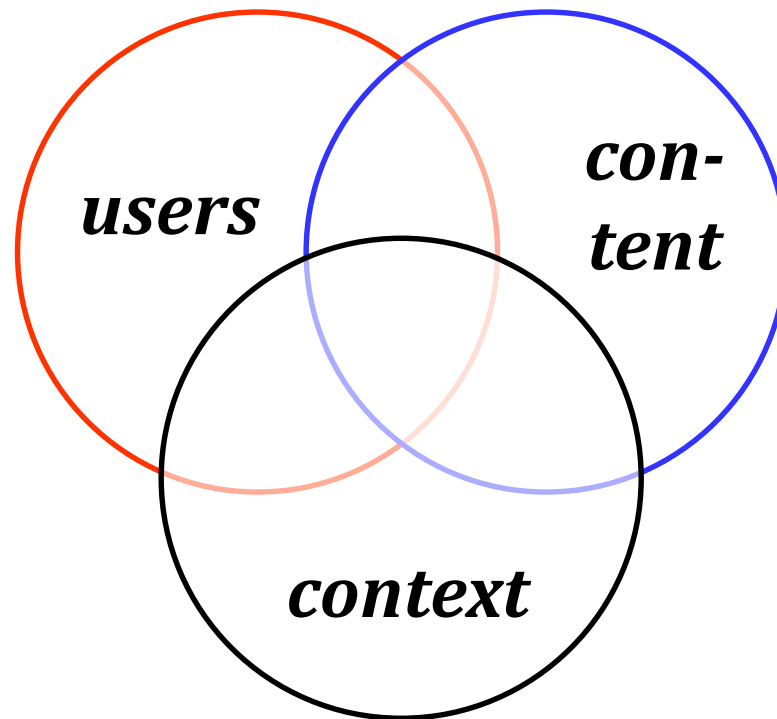
be consistent, not uniform

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

cu atragerea experților 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: example

Viziune (*big idea*)

Basecamp: “*project management is communication*”

Flickr: “*online photo management & sharing application*”

Ta-da List: “*competing with a post-it note*”

cerințe: example

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 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

Calitative

funcționalitate

fiabilitate

utilizabilitate

eficiență (performanță)

mentenabilitate

independența de platformă

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

responsive Web design

noi tipuri de cerințe

Referitoare la evoluție

utilizatorul final exploatează aplicația Web fără a trebui s-o (re)instaleze pe calculator

noi tipuri de cerințe: aspecte de interes

inițial:

oferirea funcționalităților esențiale (*less is more*)

versiuni ulterioare:

extinderea aplicației Web

- pe baza unei interfețe de programare (API) publice –
ce încurajează dezvoltarea de soluții date 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*)

folosirea unor soluții „la cheie”: biblioteci, *framework*-uri
management de proiecte
etc.

arhitecturi web: componente tipice

client(i)

firewall

proxy

middleware

server(e) Web

server(e) de aplicații

framework-uri, biblioteci, alte componente

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

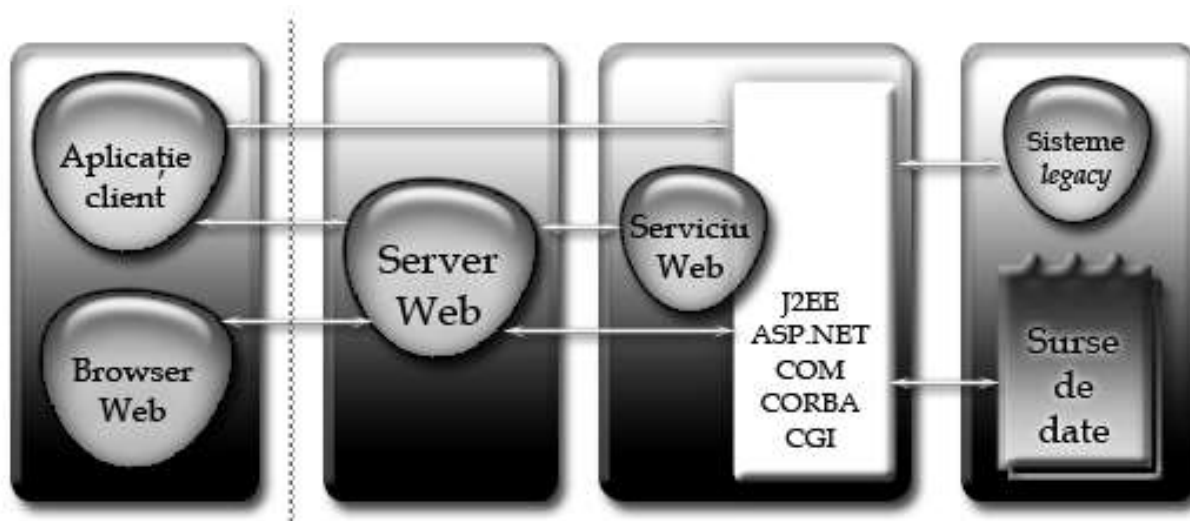
server(e) de conținut multimedia

server(e) de management al conținutului (CMS)

aplicații/sisteme tradiționale

arhitecturi web

Uzual, se adoptă arhitecturi stratificate
(*N-tier Web applications*)



Există anumite „rețete”
privind dezvoltarea de aplicații Web?

proiectare

0 problemă – oricare ar fi aceasta –
poate apărea frecvent

proiectare

Cei experimentați au găsit diverse soluții pentru problema în cauză, reușind să recunoască problema și să aleagă soluția (optimă) care poate fi aplicată într-un anumit context

proiectare

Pattern (șablon)

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

inițial, cu utilizare în arhitectură
Christopher Alexander, 1979

proiectare

Pattern (șablon)

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



proiectare

Tradițional, *pattern*-urile se utilizează în proiectarea de software

pattern \equiv “mind sized” chunk of information

lucrarea de referință:

E. Gamma *et al.*, *Design Patterns*, Addison-Wesley, 1995

proiectare

Pattern-uri de proiectare au fost folosite,
ulterior, în alte arii

interacțiune dintre om-calculator

- ▶ design și interacțiune Web, *mobile computing*

modelare conceptuală

- ▶ proiectarea bazelor de date, ontologii,...

proiectare

Un *pattern* poate descrie cunoștințele unui expert
(pe baza experienței sale personale)
în domeniul unei probleme în ceea ce privește
recunoașterea problemei, a contextului și
a soluției la acea problemă

proiectare

Un *pattern* nu reprezintă o regulă fermă

uneori nu trebuie aplicat! ► *anti-patterns*

proiectare

Este necesară adoptarea unui vocabular comun
corespunzător domeniului problemei

► *pattern language*

proiectare

Pattern-uri privitoare la:

proiectare

arhitectură

analiză

dezvoltare

structură

comportament

...

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

Șablon de specificare a unui *pattern*:

numele

rezumatul

problema

contextul

soluția

exemplele

utilizările

proiectare

Exemple de colecții de șabloane
(*patterns repositories*)

privind proiectarea de software

<http://c2.com/cgi/wiki?DesignPatterns>

patterns of enterprise application architecture

<http://martinfowler.com/eaCatalog/>

interacțiunea cu utilizatorul

<http://profs.info.uaic.ro/~evalica/patterns/>

proiectare

Șabloane de proiectare tradiționale

creaționale

Builder, Prototype, Singleton

structurale

Adapter, Bridge, Decorator, Façade, Flyweight, Proxy

comportamentale

Command, Iterator, Mediator, Observer, State, Visitor

proiectare

Web Patterns

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

proiectare

Session State Patterns

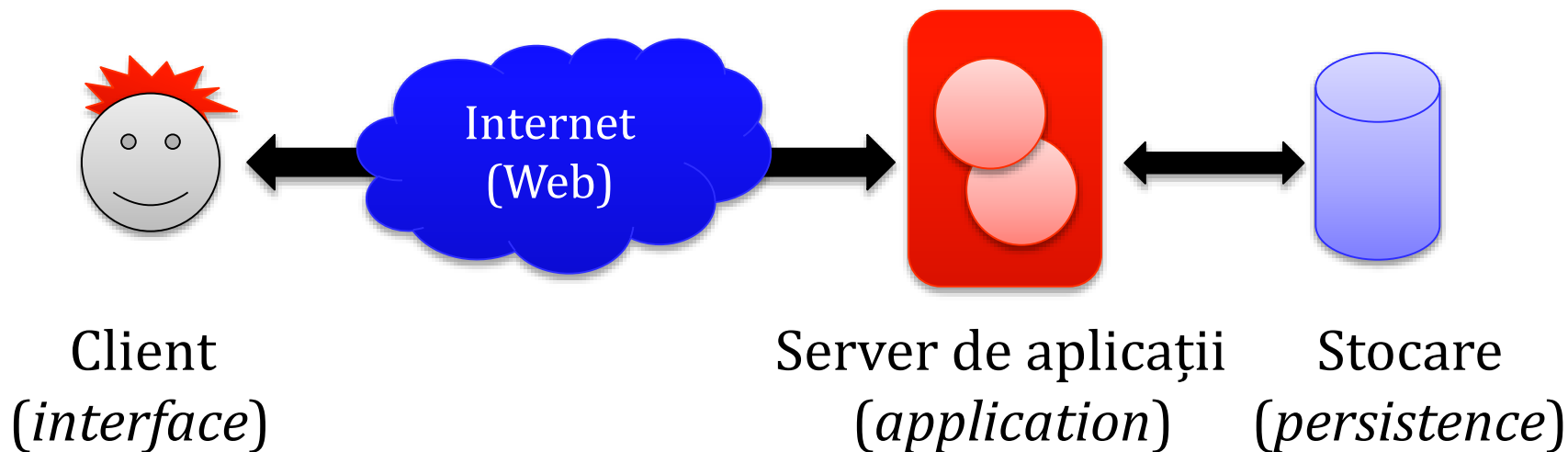
Client Session State
Server Session State
Database Session State

proiectare

Data Source Architectural Patterns

Table Data Gateway
Row Data Gateway
Active Record
Data Mapper

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





Fruit / **Presentation**

Cream / **Markup**

Custard / **Page Logic**

Jelly / **Business Logic**

Sponge / **Database**

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



Fruit / **Presentation**

Cream / **Markup**

Custard / **Page Logic**

Jelly / **Business Logic**

Sponge / **Database**

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

arhitecturi web

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

arhitecturi web

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

principiu: *separation of concerns*

arhitecturi web: **mvc**

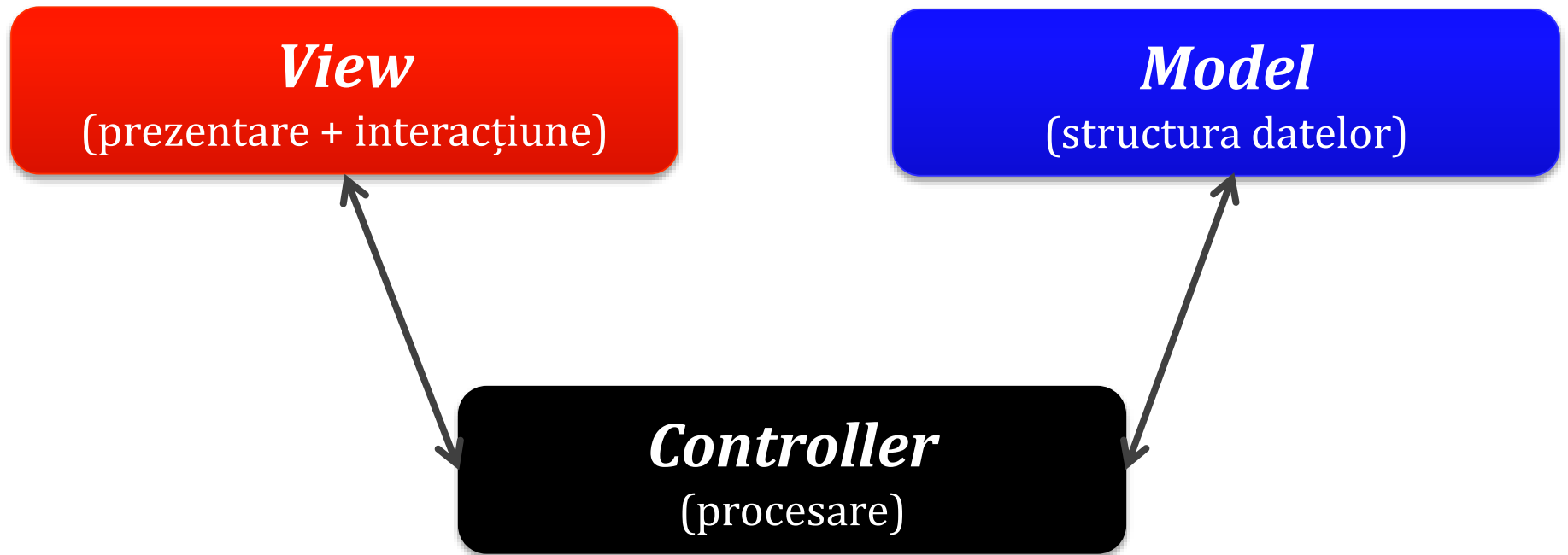
Majoritatea aplicațiilor Web sunt dezvoltate conform **MVC** (***Model-View-Controller***)

arhitecturi web: **mvc**

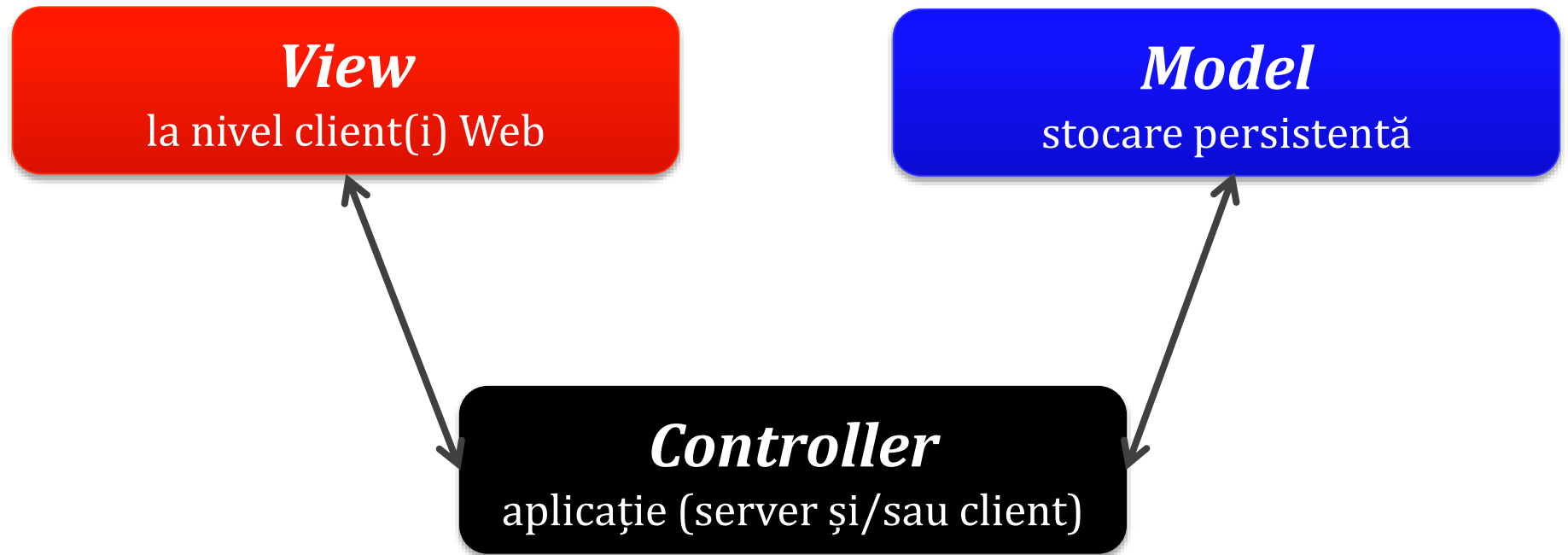
Șablon arhitectural

descriș în premieră în 1979 în contextul interacțiunii
dintre om și calculator – Smalltalk (Xerox PARC)

arhitecturi web: **mvc**



arhitecturi web: **mvc**



arhitecturi web: **mvc**



HTML, CSS, SVG,
MathML, WebGL etc.

View

la nivel client(i) Web



(No)SQL, JSON,
XML (XQuery), RDF (SPARQL),...

Model

stocare persistentă



servere de aplicații, *framework*-uri

Controller

aplicație (server și/sau client)



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 va selecta un *view* corespunzător

arhitecturi web: **mvc**

Model

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

desemnează datele și regulile (*i.e.* restricțiile)
referitoare la date ► concepte vizând aplicația Web

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

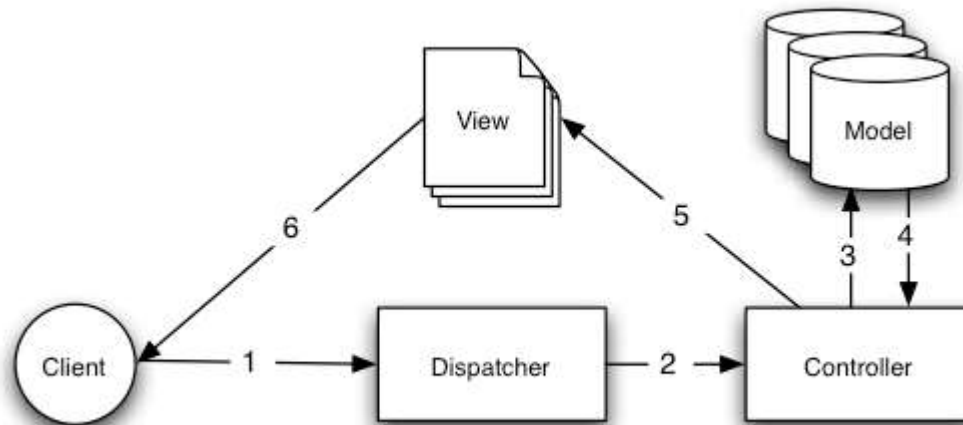
arhitecturi web: **mvc**

View

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

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

arhitecturi web: mvc



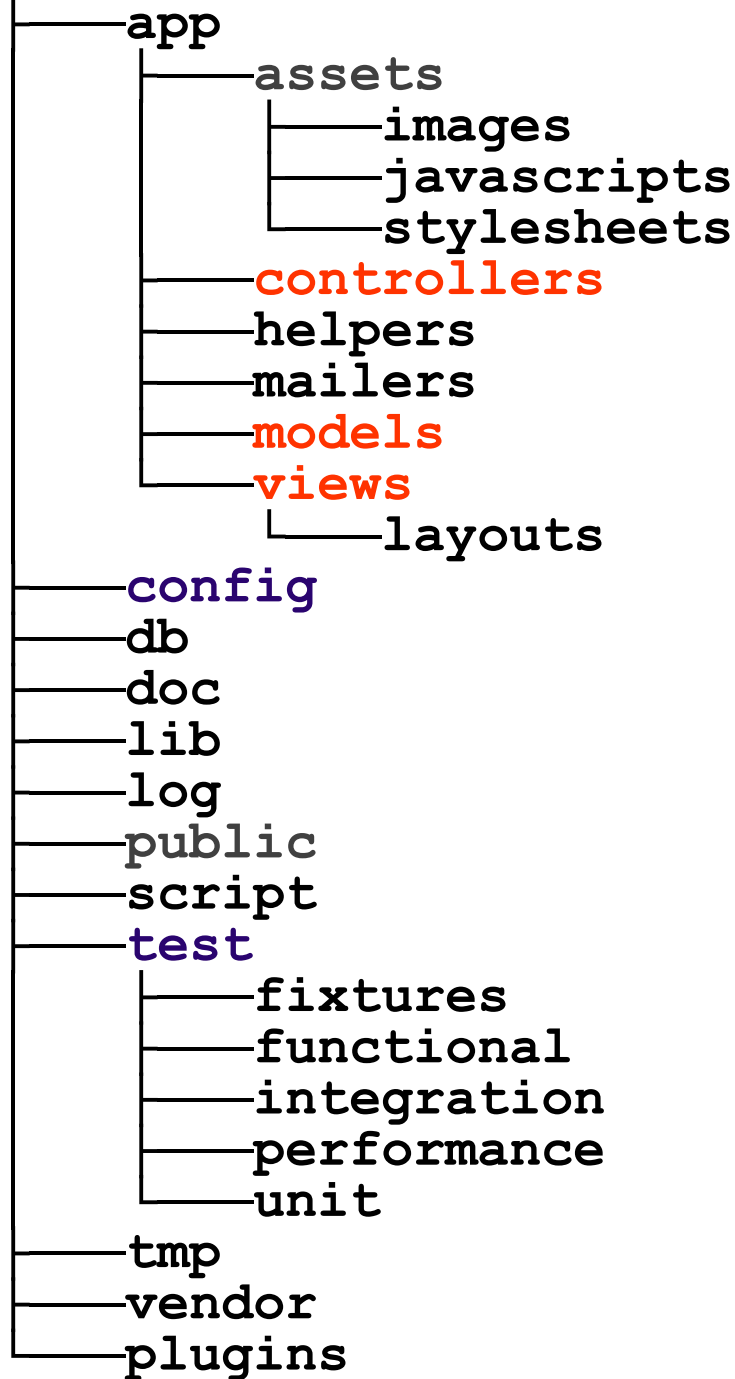
etape tipice:

- (1) cerere trimisă de client – *e.g.*, navigator Web,
- (2) dirijare (*routing*) a cererii către *controller*,
- (3) recurgera la un *model*, (4) furnizare reprezentare,
- (5) selectare a unui *view*, (6) prezentare 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



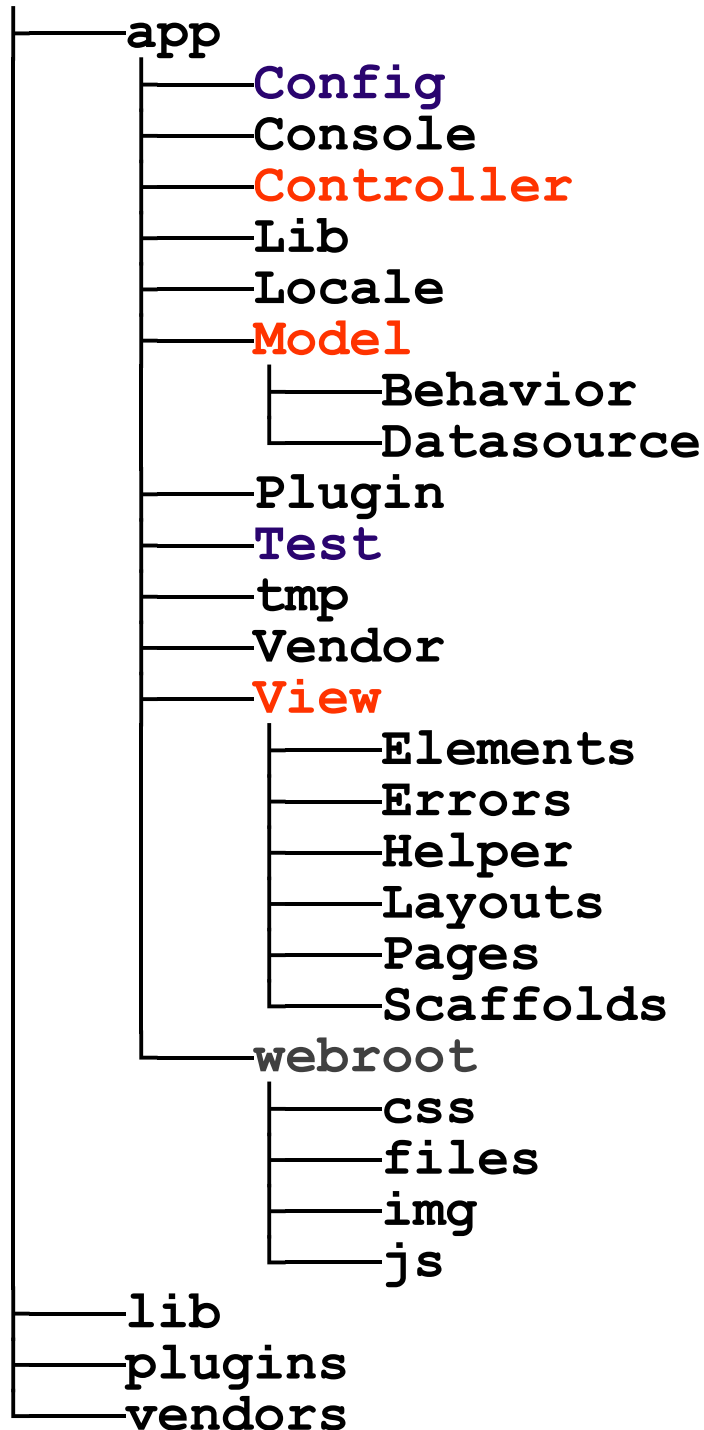
„scheletul” unei aplicații Web
 create în **Ruby on Rails**
<http://rubyonrails.org/>

avansat

structura de directoare
în cazul unei aplicații Web
folosind *framework*-ul

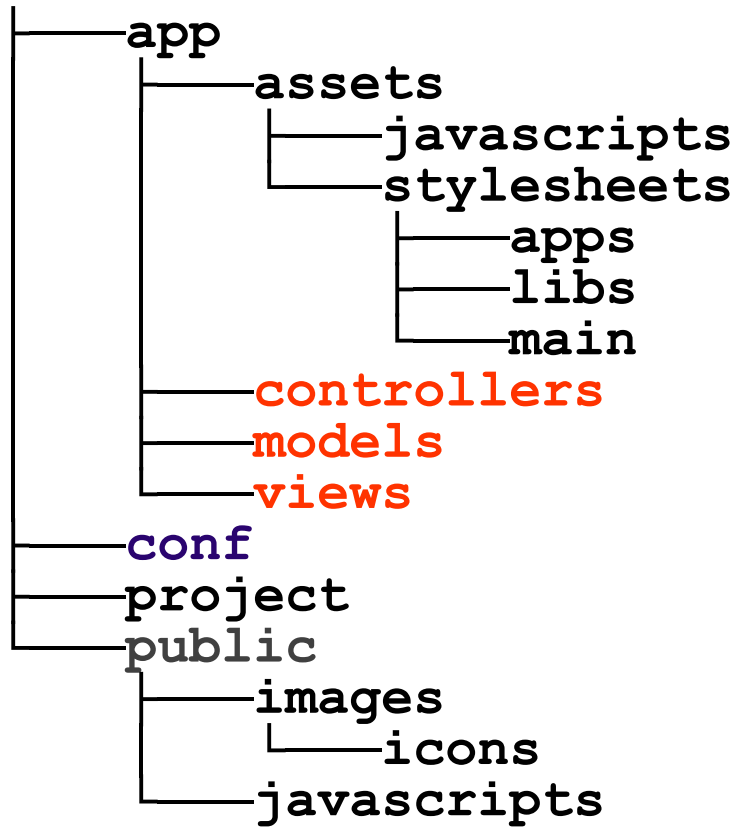
CakePHP

<http://cakephp.org/>



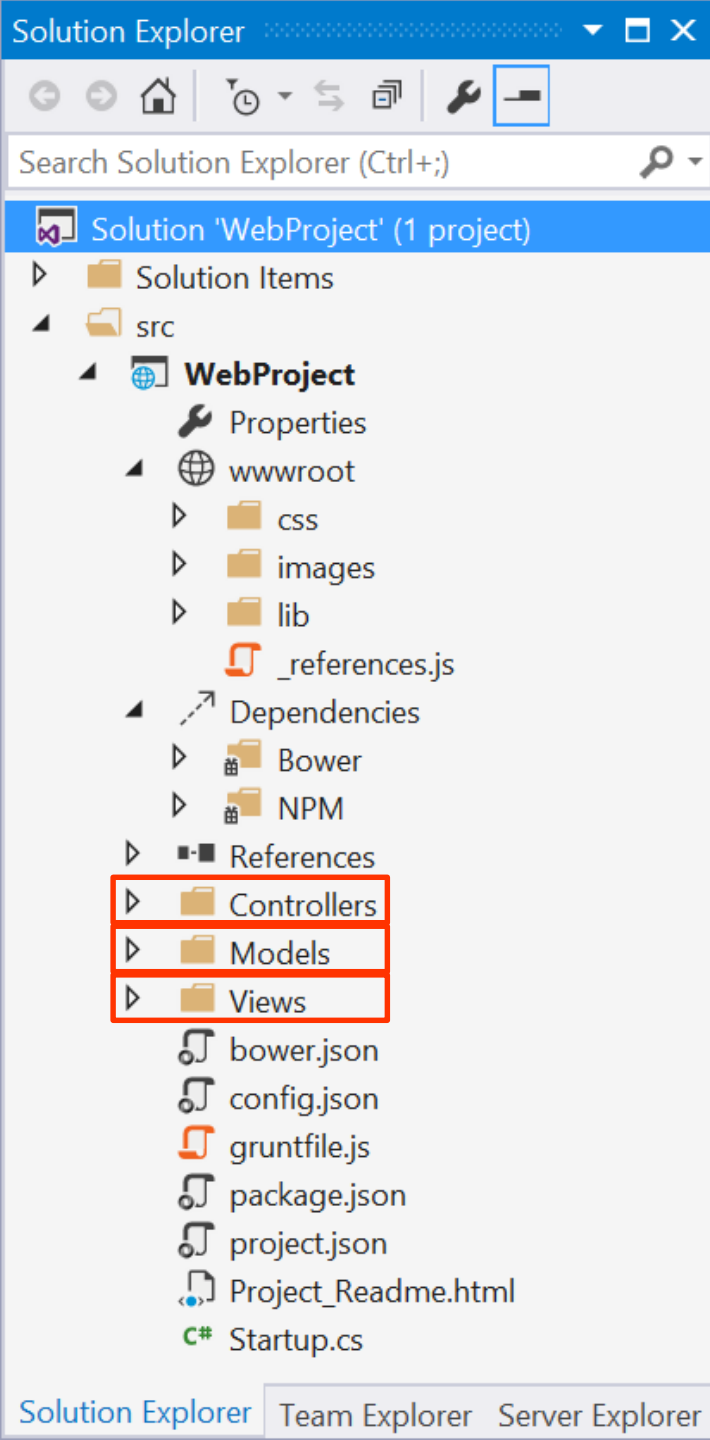
framework-uri PHP similare

www.phpwact.org/php/mvc_frameworks

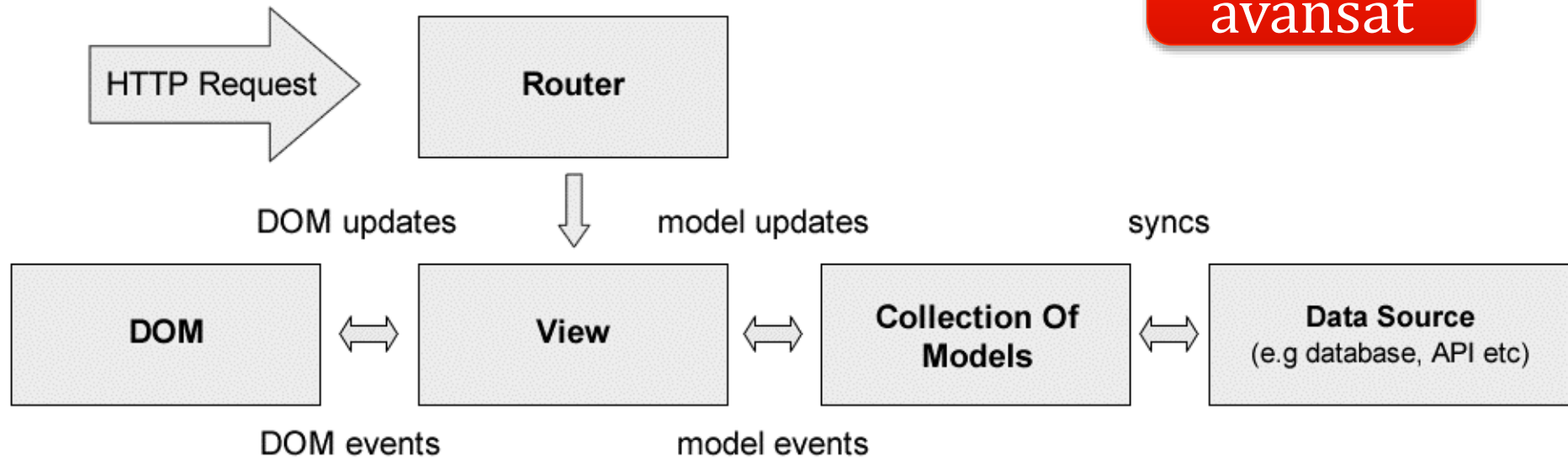


structura de directoare în cazul unei aplicații Web
ce recurge la *framework*-ul **Play** pentru Java și Scala

<http://www.playframework.org/>



structura de directoare în cazul
unei aplicații **ASP.NET MVC**
<http://www.asp.net/mvc>



flux de activități într-o aplicație MVC la nivel de client

utilizare pragmatică
via biblioteci ori *framework*-uri JavaScript

Angular – <https://angularjs.org/>

Backbone – <http://backbonejs.org/>

Ember – <http://emberjs.com/>

Mithril – <http://lhorie.github.io/mithril/>

arhitecturi web: mvc

Variante derivate:

MVVM (*Model View ViewModel*)

MVP (*Model View Presenter*)

Passive View

Supervising Controller

PAC (*Presentation Abstraction Control*)

Prin ce mijloace poate fi implementată
o aplicație Web?

implementare

Server de aplicații Web

scop:

eficientizarea proceselor de dezvoltare
a aplicațiilor Web de anvergură

implementare

Server de aplicații Web

se bazează pe **interfețe de programare (API-uri)**
și/sau pe **componente reutilizabile**

puse la dispoziție de server ori de alți ofertanți

implementare

Server de aplicații Web

poate fi integrat în unul/mai multe servere Web

de asemenea, poate oferi propriul server Web
sau mediu de execuție

implementare

Server de aplicații Web

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

situație tipică:
MVC ori variații

implementare

Server de aplicații Web

simplifică maniera de invocare
de programe (*script-uri*) ale unei aplicații Web

- ▶ generarea de conținut dinamic pe partea de server

implementare

Server de aplicații Web

limbaj(e) de programare

API-ul de bază

stocare persistentă a modelelor de date (relaționale, XML)

interacțiune Web

cookie-uri & sesiuni

medii de dezvoltare + cadre de lucru

caracteristici particulare

implementare

Server de aplicații Web

limbaj(e) de programare

C# și alte limbaje *.NET Framework* – ASP.NET

Java – AppFuse, Play, Wicket etc.

JavaScript – Node.js + *framework-uri*: Express, Locomotive etc.

PHP – PHP + *framework-uri*: CakePHP, Laravel, Symfony,...

Python – Django, Flask, Grok, Pyramid, Zope

Ruby – Ruby on Rails, Sinatra

implementare

Server de aplicații Web

limbaj(e) de programare

pot fi dinamice – *e.g.*, Python, Ruby

interpretate sau compilate

uzual, se preferă generarea de cod intermediar:

IL (*Intermediate Language*) – C#, Java

implementare

Server de aplicații Web

API de bază

contribuie la „puterea” limbajului + serverului de aplicații
(via funcții/clase predefinite)

securitate, consistență,
acces la resursele mediului de operare/rulare,
asigurarea independenței de platformă

implementare

Server de aplicații Web

suport pentru stocare persistentă
în baze de date relaționale – via **SQL**

exemplu: funcții/module PHP predefinite
pentru o pleiadă de sisteme de baze de date
(Firebird, MySQL, PostgreSQL,...)

biblioteci incorporate (**SQLite** + **mysqli**) sau diverse extensii

implementare

Server de aplicații Web

suport pentru stocare persistentă
în baze de date relaționale – via **SQL**

ORM (*Object-Relational Mapping*)

ADO.NET pentru ASP.NET

JDBC (*Java DataBase Connectivity*) pentru Java (JSP)

Sequelize – bibliotecă pentru Node.js

implementare

Server de aplicații Web

suport pentru stocare persistentă
în baze de date relaționale – via **SQL**

eventual, *framework*-uri adiționale
implementând șablonul *Active Record*

exemple: **active_record** (modul Node.js), **Castle Project** (.NET),
Doctrine (PHP), **Play Framework** (Java, Scala), **Rails** (Ruby)

implementare

Server de aplicații Web

suport pentru stocare persistentă
pe baza modelelor arborescente: XML

date (semi)structurate

transformări în alte formate: XPath, XSLT

procesări: DOM, SAX, SimpleXML etc.

validări de date: DTD, XML Schema, RELAX,...

interogări: XQuery

cursurile
viitoare

implementare

Server de aplicații Web

suport pentru stocare persistentă
recurgând la alte paradigme non-relaționale
(bazate pe grafuri și/sau cheie—valoare),
distribuite la nivel de Internet, scalabile – NoSQL
<http://nosql.mypopescu.com/>

exemplificări:

Cassandra, CouchDB, Hadoop, MarkLogic, MongoDB, Neo4j etc.

implementare

Server de aplicații Web

suport pentru interacțiunea Web

interacțiunea e facilitată de controale specificate
în cadrul codului-sursă rulat la nivel de server

emulează câmpurile din formularele HTML și/sau
oferă controale noi – *e.g.*, calendar, *slideshow*,...

► generare de cod HTML (+JavaScript) în funcție de client

implementare

Server de aplicații Web

suport pentru interacțiunea Web

exemplificări:

ASP.NET (<**asp:control**> – *e.g.*, FileUpload, ListBox, Table,...)

framework-ul **PRADO** (PHP)

formidable, **form-data**, **forms** – module Node.js

similar, pentru platforma Java: *e.g.*, **JSF** (*JavaServer Faces*)

implementare

Server de aplicații Web

suport pentru interacțiunea Web

încurajarea folosirii de machete de vizualizare (*templates*)
pe baza unui procesor specific – ***Web template system***

implementare

Server de aplicații Web

suport pentru interacțiunea Web

Web template system

utilizând specificații de prezentare a conținutului
(*Web template*), datele persistente
(*e.g.*, preluate dintr-o bază de date) sunt folosite
de un procesor (*template engine*)
pentru a genera documente HTML ori alte formate

implementare

Server de aplicații Web

suport pentru interacțiunea Web

Web template system

la nivel de server

Haml (Ruby), **Mustache** (C++, JS, PHP, Python, Scala,...),
Smarty (PHP), **Velocity** (Java), **XSLT** (XML) etc.

implementare

Server de aplicații Web

suport pentru interacțiunea Web

Web template system

la nivel de client

disponibile pentru JavaScript:

Dust.js, EJS, HandleBars, Mustache, Nunjucks,...

implementare

Server de aplicații Web

suport pentru interacțiunea Web

transfer asincron de date via suita de tehnologii Ajax



vezi cursurile
viitoare

eventual, via *framework*-uri/module/clase adiționale

implementare

Server de aplicații Web

suport acordat inginerilor software

aplicații *N*-tier

se încurajează folosirea șabloanelor de proiectare:

*Container, MVC (Model-View-Controller),
Proxy, Configuration Parameters, Invocation Context,...*

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

implementare

Framework (cadru de lucru)

clasificare:

de uz general

management de conținut
(CMS – *Content Management System*)

la nivel de intranet – *e.g.*, portal organizațional

implementare

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

ASP.NET: ASP.NET 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

Biblioteca Web (*library*)

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

implementare

Biblioteca 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

implementare

Serviciu Web

software – utilizat la distanță de alte aplicații/servicii –
oferind o funcționalitate specifică,
a cărei implementare nu trebuie cunoscută de dezvoltator



detalii în
cursurile viitoare

implementare

Serviciu Web

software – utilizat la distanță de alte aplicații/servicii –
oferind o funcționalitate specifică,
a cărei implementare nu trebuie cunoscută de dezvoltator

recurge la tehnologii Web deschise
(adresare via URI, acces prin HTTP,
formate de date: CSV, JSON, XML,...)

implementare

API (*Application Programming Interface*)

“any well-defined interface that defines the service that one component, module, or application provides to other software elements”



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

Alexa Web	Web traffic data	API Documentation	Python wrapper for Alexa Web
Amazon	Online Shopping	API Documentation	Python wrapper for Amazon.com
AWS	Cloud computing platform	API Documentation	Python wrapper for AWS
Archive.org	Internet Archive	API Documentation	Python wrapper for archive.org
Balanced	Payments for Marketplaces	API Documentation	Python wrapper for Balanced
BigML	Machine Learning Made Easy	API Documentation	Python wrapper for BigML
Bing	Microsoft search engine	API Documentation	Python wrapper for Bing search
Bitly	URL shortener	API Documentation	Python wrapper around bit.ly
Blogger	Blog-publishing service	API Documentation	Python wrapper around Blogger
Box	Online file sharing	API Documentation	Python wrapper for Box
Braintree	Accept Payments Online	API Documentation	Python wrapper for Braintree
Carriots	M2M Application Platform	API Documentation	Python wrapper for Carriots
Close.io	Sales communication platform	API Documentation	Python wrapper for Close.io
Coinbase	Bitcoin Wallet	API Documentation	Python wrapper for Coinbase

exemplu: acces la API-uri în Python – www.pythonapi.com

implementare

Web component

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

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

implementare

Web component

dezvoltare bazată pe o bibliotecă/*framework*

soluții „tradiționale” – uzual, la nivel de client:

Dojo Toolkit

jQuery UI

...

implementare

Web component

dezvoltare bazată pe o bibliotecă/*framework*

cadrul general:

Web Components (în lucru la Consorțiul Web)
recurgând la diverse tehnologii HTML5

<http://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,...

implementare

(Web) app

a distributed computer software application designed for optimal use on specific screen sizes and with particular interface technologies

Robert Shilston, 2013

implementare

(Web) app

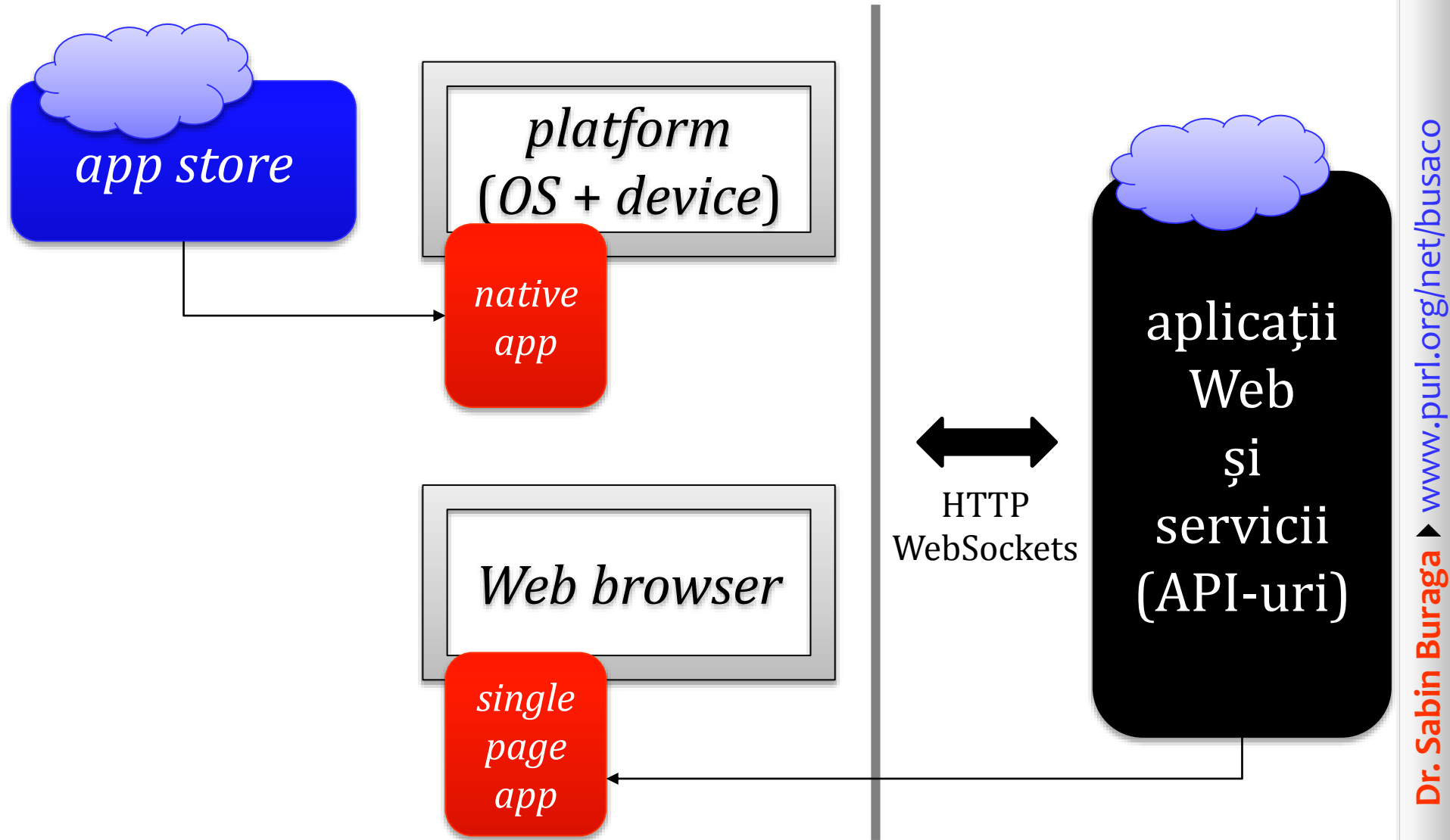
uzual, se poate obține via un *app store*
(centralizat sau descentralizat)

exemple notabile:

Chrome Apps

aplicații **Windows** dezvoltate în JavaScript

aplicații Web mobile pentru **Firefox OS, Kindle Fire,...**



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
comerciale
versus
educaționale, gratuite sau *open-source*

dezvoltare

Recurgerea la medii de dezvoltare

comerciale

versus

educaționale, gratuite sau *open-source*

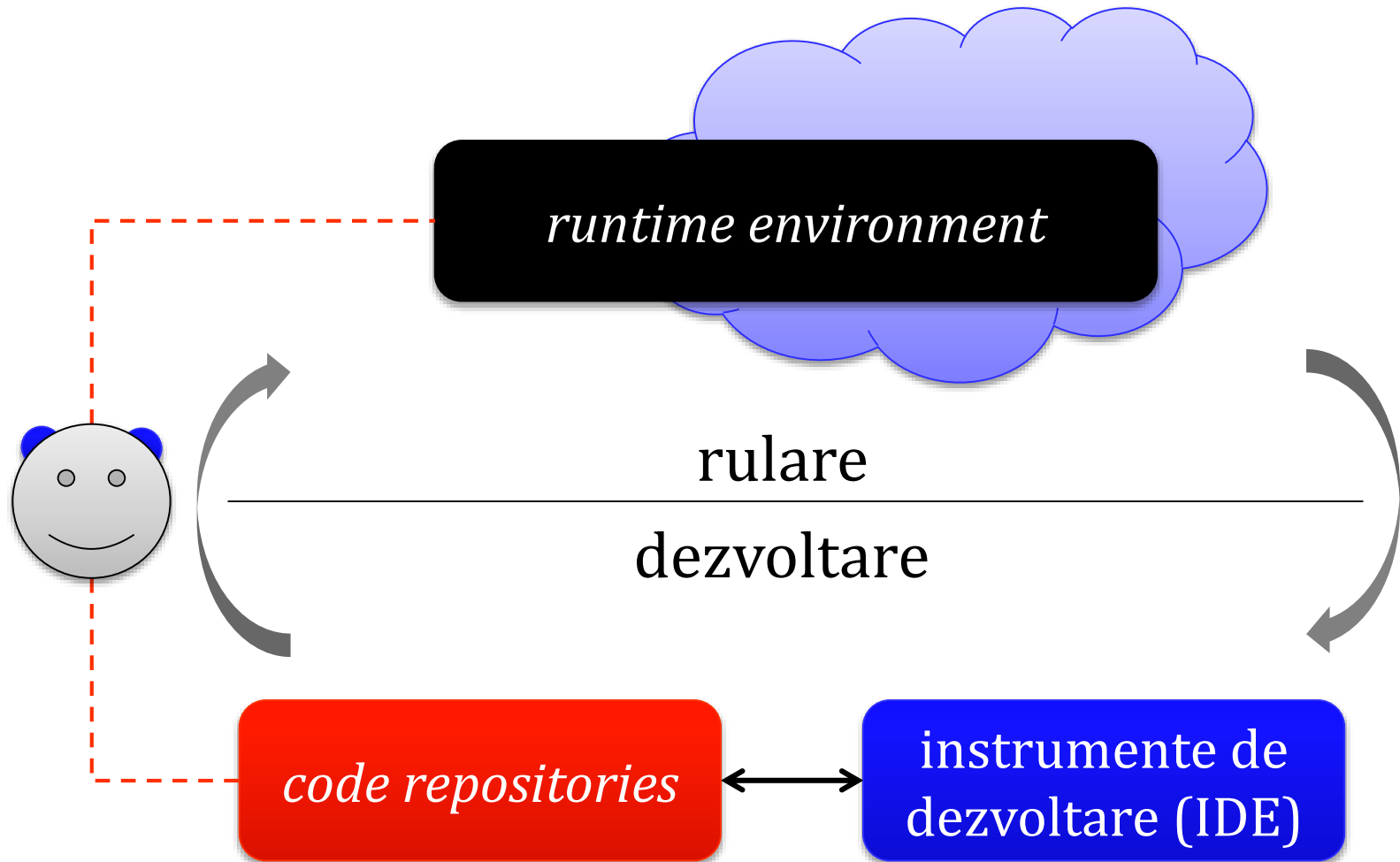
exemplificări:

Anjuta, Aptana Studio, Eclipse, Emacs, IntelliJ IDEA,
KomodoIDE, Padre, PHPStorm, PyCharm, RubyMine,
Visual Studio, Zend Studio

dezvoltare

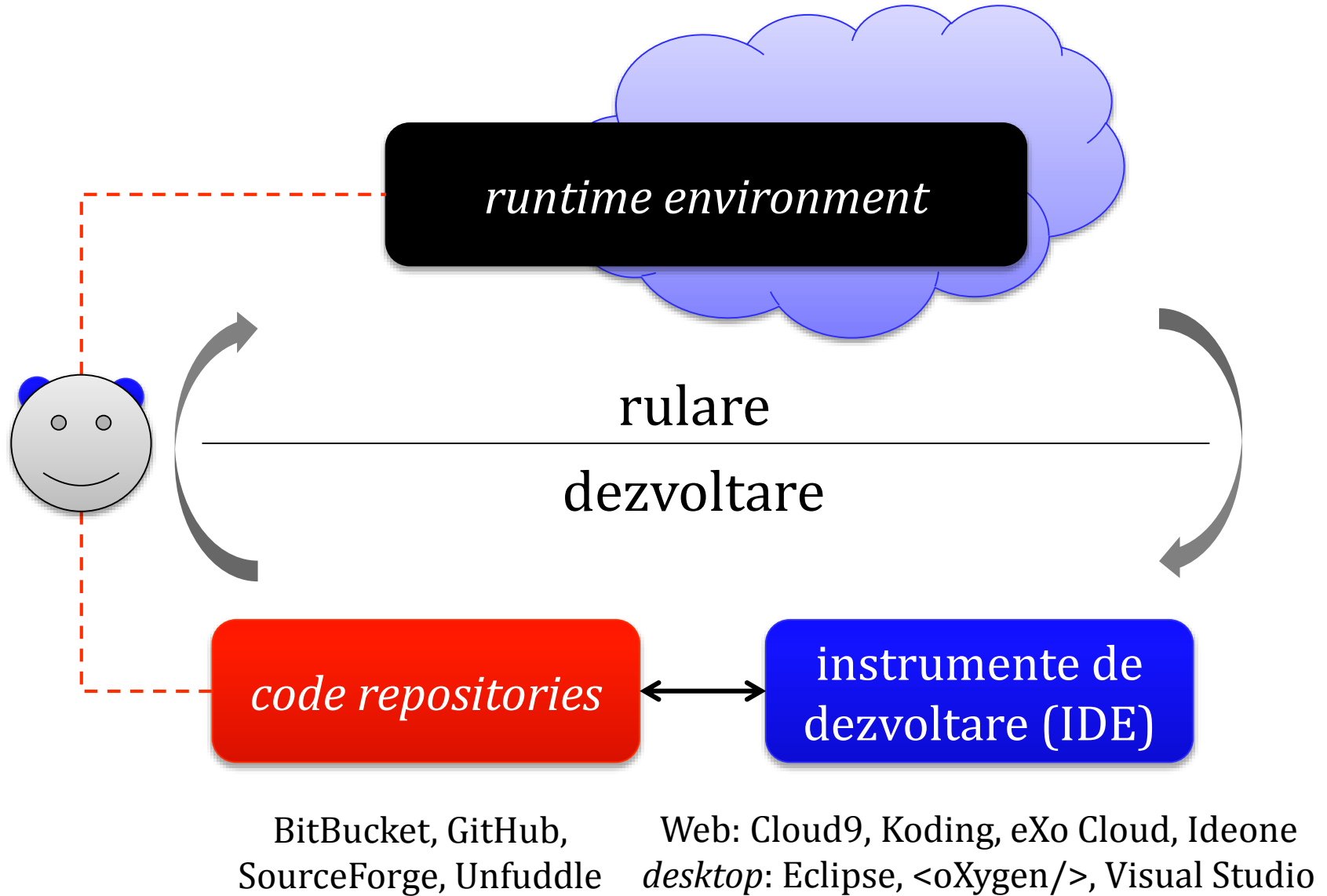
Recurgerea la medii de dezvoltare
disponibile ca aplicații Web

Cloud9 IDE – <https://c9.io/>
Codenvy – <https://codenvy.com/>
CodePen – codepen.io
Compilr – <https://compilr.com/>
JSFiddle – <http://jsfiddle.net/>
PhpFiddle – <http://phpfiddle.org/>



Development as a Service

Google App Engine, Heroku,
Jelastic, Windows Azure



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 aceleiași suite de programe (*codebase*)

Instrumente client/server:

Apache Subversion – SVN

Microsoft Team Foundation Server – TFS

Soluții distribuite:

Git (implementat în bash, C și Perl)

<http://git-scm.com/>

Mercurial (dezvoltat în Python)

<http://mercurial.selenic.com/>

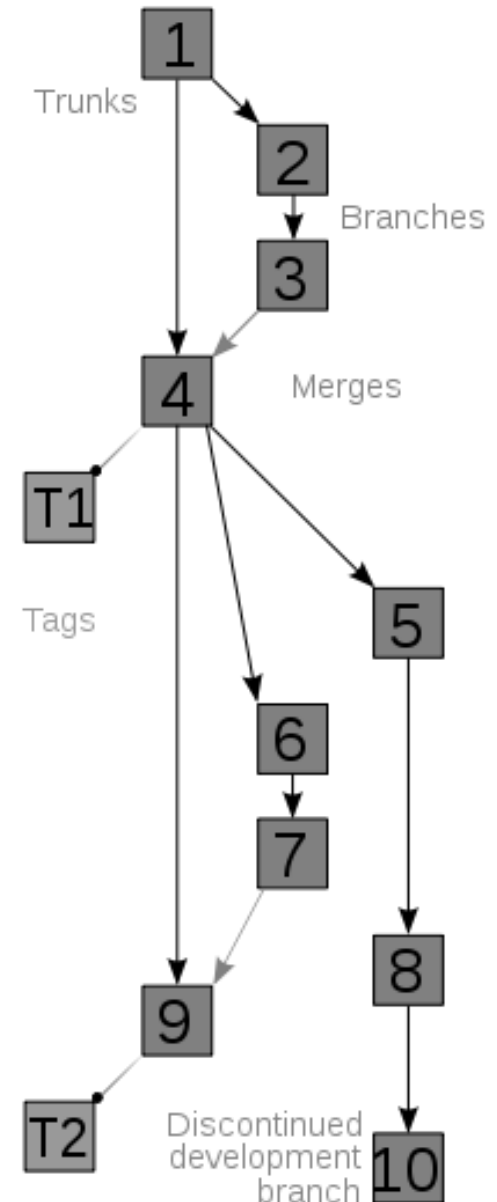
Rational Team Concert (oferit de IBM)

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

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

BitBucket – <https://bitbucket.org/>

GitHub – <https://github.com/>



dezvoltare

Încurajarea/impunerea
unui stil de redactare a codului-sursă

exemple:

C# – <https://github.com/dennisdooomen/csharpguidelines>

Java – <http://checkstyle.sourceforge.net/>

JavaScript – <https://github.com/rwaldron/idiomatic.js/>

Perl – <http://perldoc.perl.org/perlstyle.html>

PHP – <http://pear.php.net/manual/en/standards.php>

Python – <https://www.python.org/dev/peps/>

Ruby – <https://github.com/styleguide/ruby>

dezvoltare

Management de pachete software

căutare, instalare, compilare, verificare a dependențelor

exemplificări:

Bower, Composer, npm, NuGet, RubyGems

de studiat <https://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, NUnit (.NET),

Test::Class (Perl), unittest (Python), Unit.js

+

JSUnit, FireUnit, Mocha, Selenium

la nivel de client

de vizitat și <http://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

referitoare la **securitate**

testare: exemplu – codul-sursă

Documente HTML – validator.w3.org, instrumentul Tidy

Foi de stiluri CSS – CSS Lint: <http://csslint.net/>

Documente XML – bine-formatate / valide

Script-uri pe partea client (JavaScript) via JSLint, JSHint

Programe rulate la nivel de server – xUnit

Integritatea și accesul la sistemul de fișiere

Integritatea și accesul la bazele de date

Suport oferit de navigatorul Web – <http://caniuse.com/>

Probleme de securitate – www.owasp.org

Rezolvarea scalabilității aplicației Web

exploatare

Publicarea sitului

server dedicat

vs.

furnizor de găzduire Web (*hosting*)

soluție gratuită vs. comercială

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

fără *Under construction* ori *404 Not found* nicăieri!

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*)

exploatare: management

Privind utilizatorul:

interacţiune Web – *e.g.*, utilizabilitate

www.slideshare.net/busaco/interaciune-web-concepte-context-studii-de-caz

şabloane de proiectare a aplicaţiilor Web sociale

www.slideshare.net/busaco/hci06-design-patternssocialinteraction

performanţa Web la nivel de *browser*

www.slideshare.net/busaco/web12-performantaaplicatiilorwebclient

exploatare: analiza utilizării

Usage analysis

metode explicite

bazate pe date oferite de utilizator

e.g., chestionare & monitorizare (*user testing*),
analiza mesajelor de *e-mail* 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 de vizitare etc.

servicii de monitorizare/raportare

exemple: **Google Analytics**, **WordPress Statistics**

parametrii unui proiect web

obiectiv principal

durată

cost

abordare

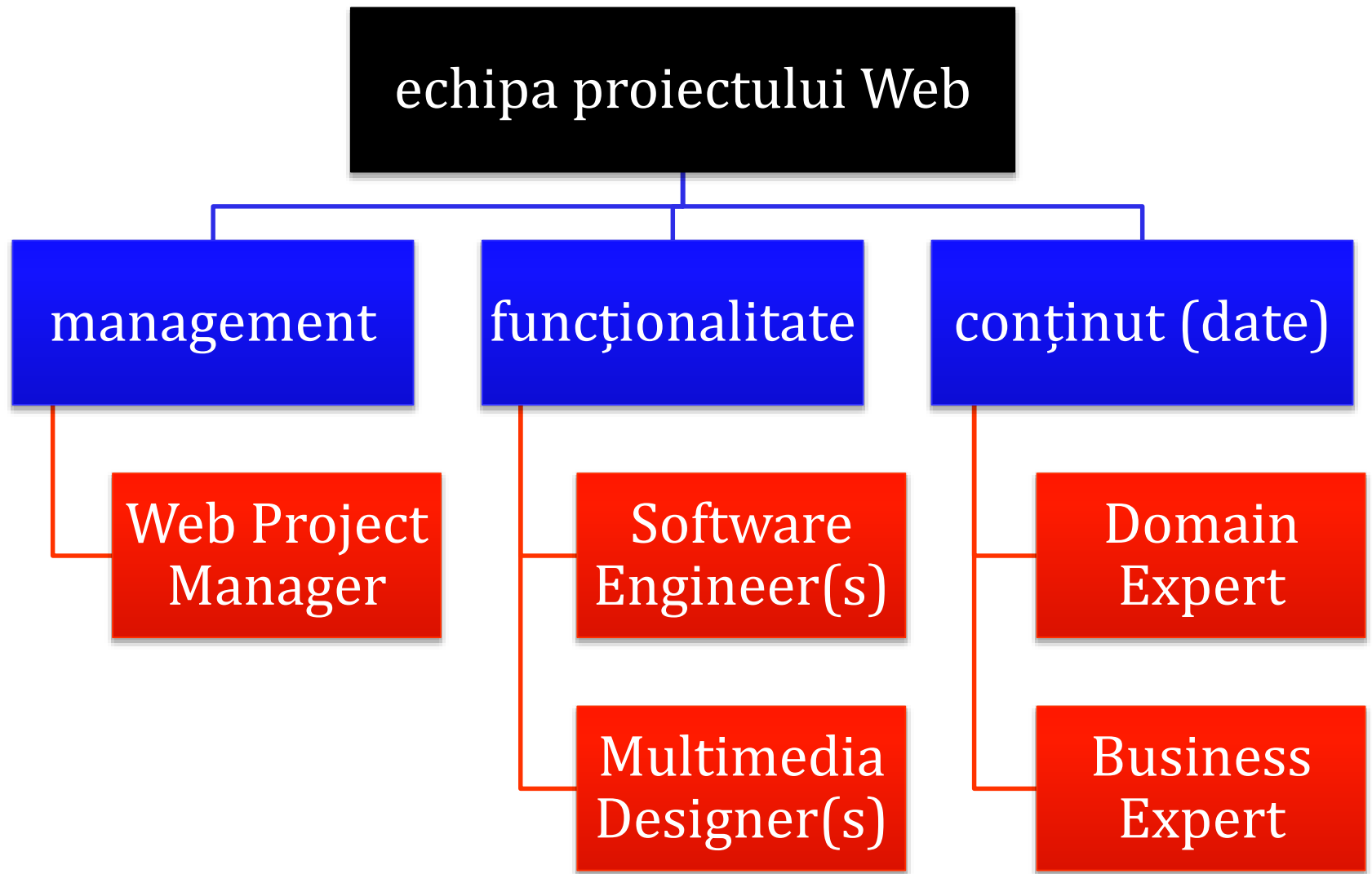
tehnologii

proces

rezultat

resurse umane

profilul echipei



vezi S. Buraga, „Ce înseamnă a fi dezvoltator Web” (2014)

www.slideshare.net/busaco/ce-nseamn-a-fi-dezvoltator-web

Câteva exemplificări
privind arhitectura unor aplicații Web?



Imagine Cup 2009

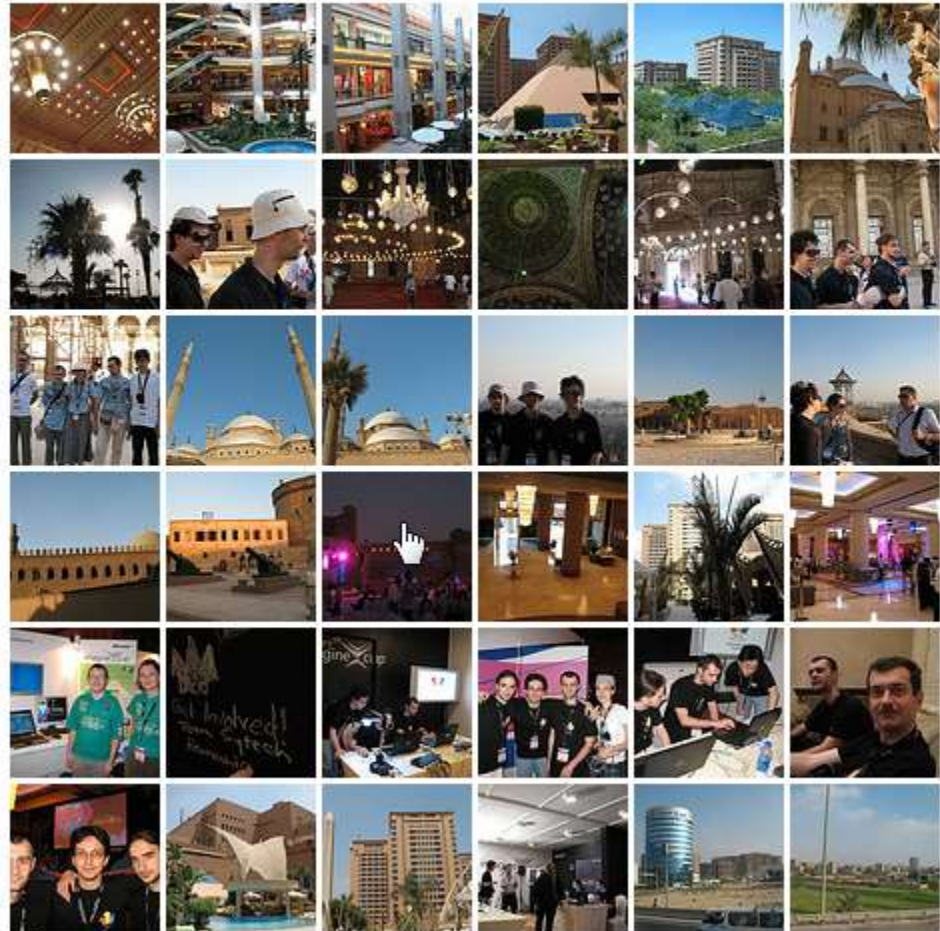
[Thumbnails](#) [Detail](#) [Comments](#)



Photos taken during the Imagine Cup 2009 competition, 02-08 July 2009, Cairo, Egypt.

110 photos | 384 views | [Add a comment?](#)

Items are from 13 Jul 2009.



studiu de caz: **Flickr**

studiu de caz: flickr

Scop:

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

aplicatie reprezentativă a Web-ului social

agregare de comunități – imaginea ca obiect social

suport pentru adnotari via termeni de continut (*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)

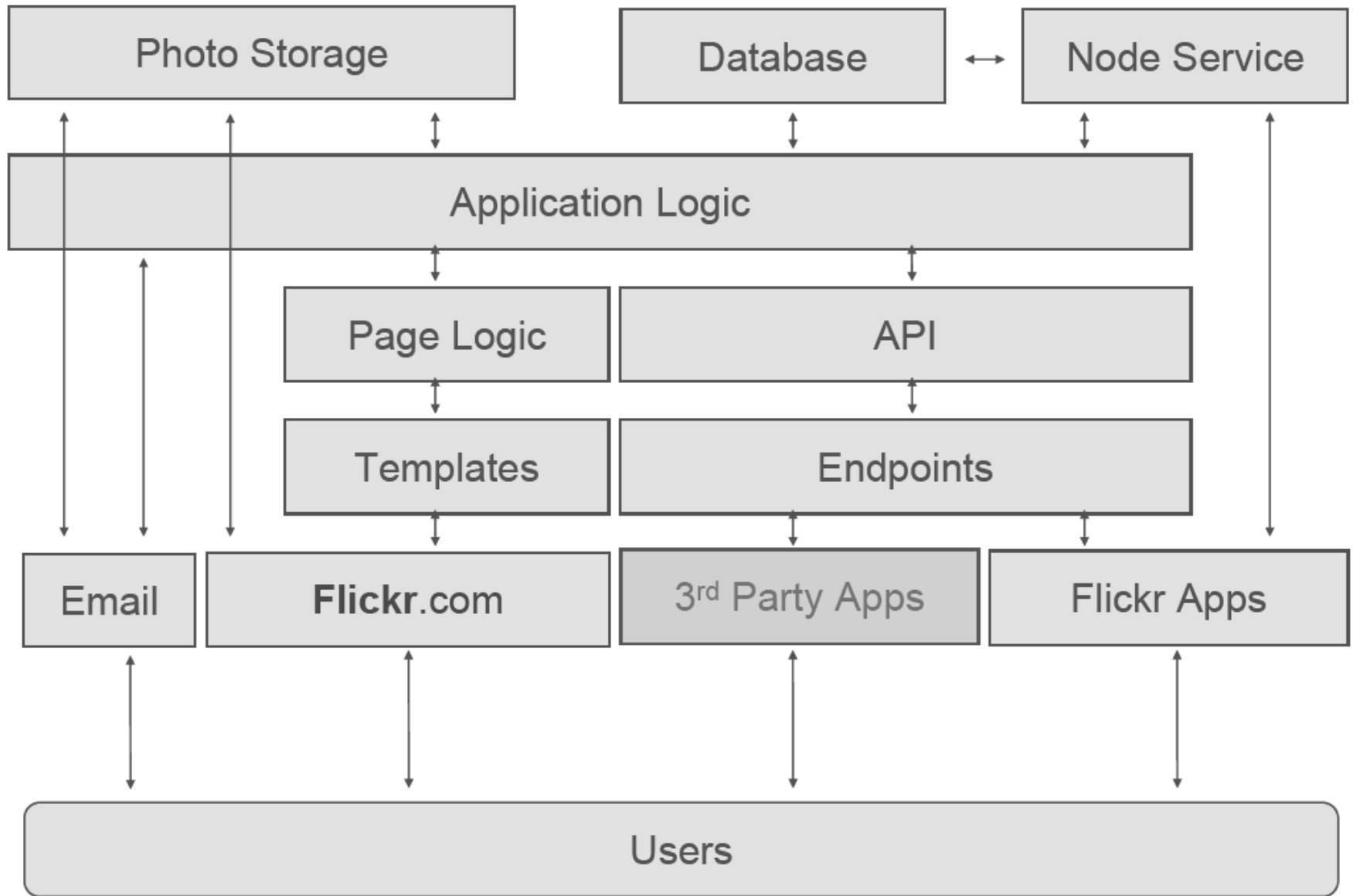
MySQL (stocare în format InnoDB)

ImageMagick (bibliotecă C de prelucrare de imagini)

Ajax (interacțiune asincronă)

Linux (platformă de rulare)

alte detalii la <http://highscalability.com/flickr-architecture>



arhitectura inițială – conform (Cal Henderson, 2007)

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, eventual,
pe alte platforme

www.flickr.com/services/api/

Open Source Iasi

Open Source conference

15 MARCH
2014

 Iasi
in Romania

 opensourceiasi.wordpress.com

 Save to iCal / iPhone / Outlook / GCal

[#opensourceiasi](https://twitter.com/opensourceiasi)

 lanyrd.com/cxgxd (short URL)

6 speakers



[Jakob Cosoroabă](#)

@jcsrb



[Sabin Buraga](#)



[Alex Lakatos](#)

@lakatos88

Mozilla Rep, JavaScript Dev



[Andreea Popescu](#)

@andreea_popescu

Mozilla Rep, Firefox user and
Firefox OS app reviewer



[Stefan Cosma](#)

@stefanbc

App builder extraordinaire,
community manager and developer



[Ioana Chiorean](#)

@ioana_cis

studiu de caz: **Lanyrd**

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 & audiență, *slide*-uri,...
+ calendare și localități de desfășurare

concepte importante: *conferences, user profiles, emails, dashboard, coverage, topics, guides*

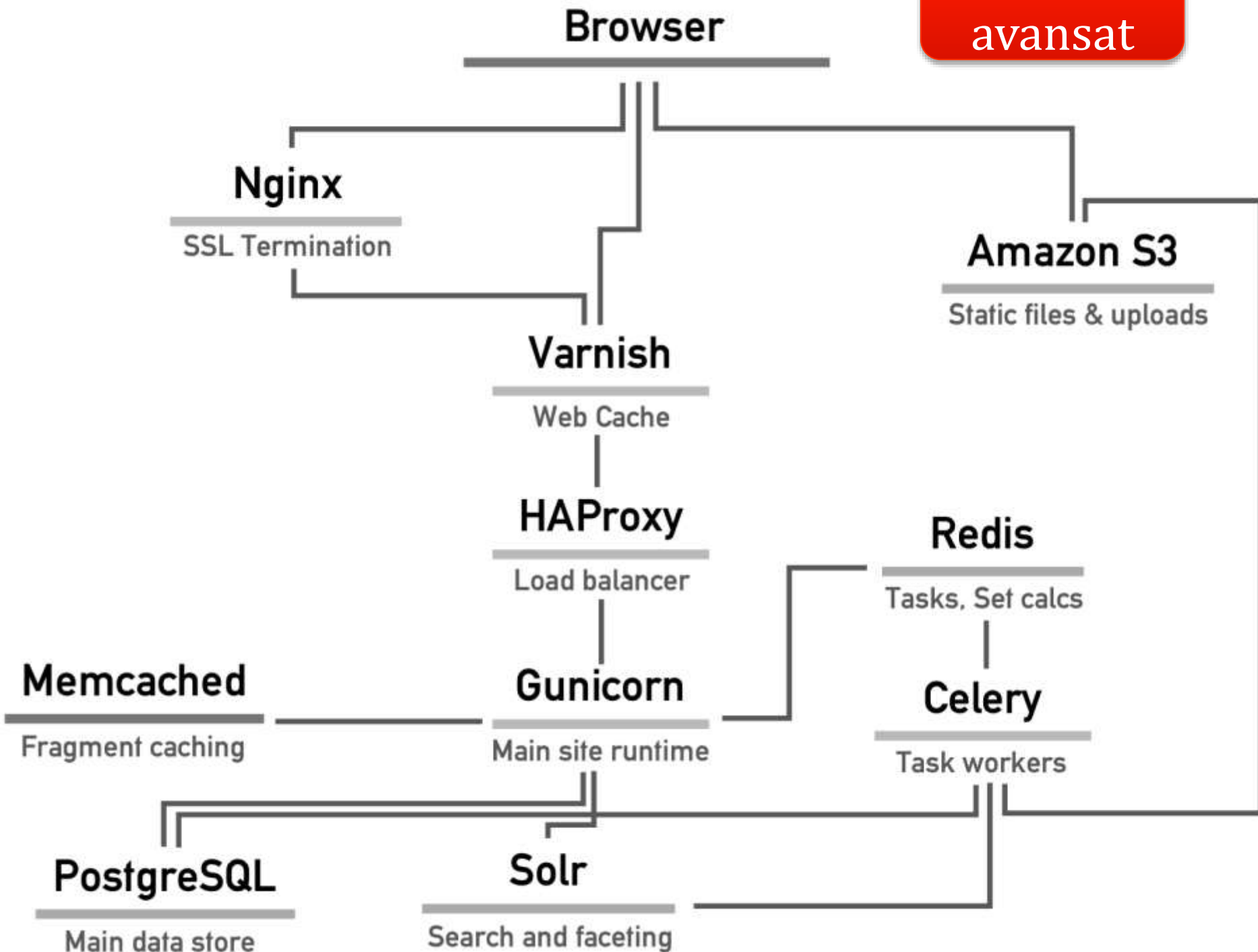
studiu de caz: lanyrd

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

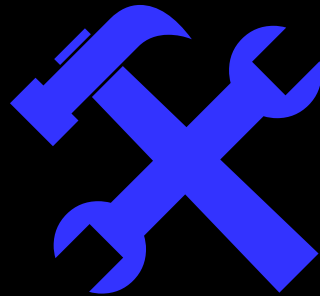
2½	<i>backend developers</i>
1¾	<i>frontend developers</i>
½	<i>mobile developers</i>
1½	<i>designers</i>
¾	<i>system administrators</i>
¾	<i>business operations</i>

A. Godwin, *Inside Lanyrd's Architecture*, QCon London, 2013

<http://www.infoq.com/presentations/lanyrd-architecture>



rezumat



programare Web ► inginerie Web
dezvoltarea aplicațiilor Web – aspecte esențiale

PhpFiddle

Sign In Sign UP


Web app demo → References → Aid coding tools → Gist interaction → Dropbox interaction →

Editor Window Res Links APIs LibFiddle Intro Doc

Analyze code → Run [F9] Save Update Export → Editor theme →

Run this code

```
1 <?php
2 // umplem un tablou cu valori de la 1 la 10
3 for ($contor = 1; $contor <= 10; $contor++) {
4     $valori[$contor] = $contor;
5 }
6 // realizam suma valorilor
7 $suma = 0;
8 foreach ($valori as $element)
9     $suma += $element;
10 // afisam suma obtinute la iesirea standard
11 // pentru a fi trimisa clientului Web
12 echo("<p>Suma de la 1 la 10 este <strong>" . $suma . "</strong>.</p>");
13 ?>
```



episodul viitor:

dezvoltarea de aplicații Web în PHP