

[IF977] Engenharia de Software

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: [@vinicius3w](https://twitter.com/vinicius3w) :: assertlab.com

Licença do material

Este Trabalho foi licenciado com uma Licença

Creative Commons - Atribuição-NãoComercial-Compartilhalgual 3.0 Não Adaptada.

Mais informações visite

<http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt>





Proibido jogar durante a aula!

1º Aviso, eu vou confiscar o celular e vou gastar **TODAS** as suas pokebólas!

Provavelmente em um Pidgey ou Rattata... Mas eu vou errar. E de novo, de novo...

2º Aviso, depois de gastar as pokebólas eu vou usar todo o incenso e os lucky eggs!

Vai estar cheio de pokémons em volta, mas você não vai ter como capturá-los

3º Aviso, eu vou trocar os seus pokémons de level mais alto e mais raros! Talvez até 2... ou 3 deles.

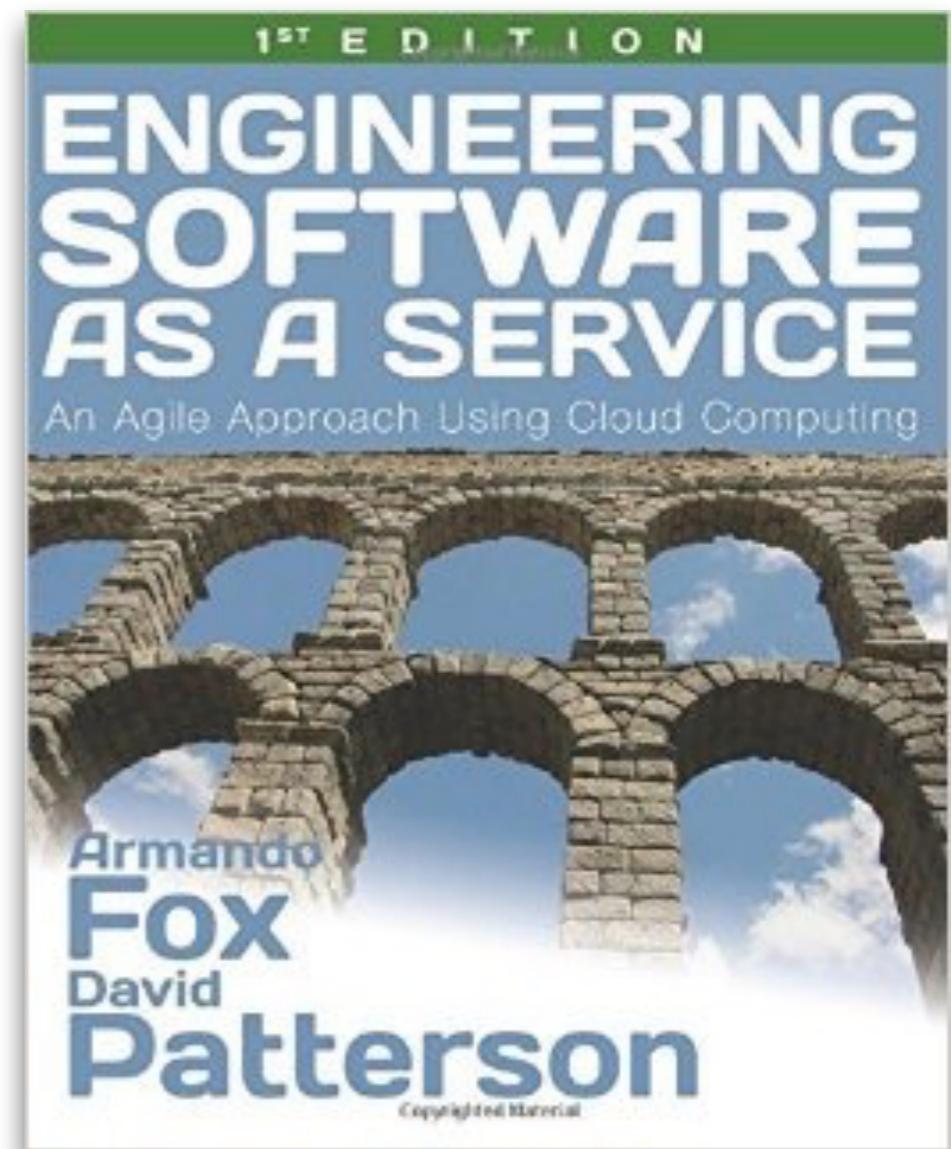
1 candy por uma Dragonas 1453 CP. Parece justo :D

Se não posso capturá-los, você também não pode!



Referências

- A biblioteca do Desenvolvedor de Software dos dias de hoje
 - <http://bit.ly/TDOA5L>
- SWEBOK
 - Guide to the Software Engineering Body of Knowledge (SWEBOK): <http://www.computer.org/web/swebok>
- Engineering Software as a Service: An Agile Approach Using Cloud Computing (Beta Edition)
 - <http://www.saasbook.info/>



Organização do Curso

- (ESaaS 1.1) Introdução a Engenharia de Software
- (ESaaS 1.4) Arquitetura Orientada a Serviço
- (ESaaS 1.5) Software como Serviço
- (ESaaS 1.6) Computação em Nuvem
- (ESaaS 1.7) Código Bonito vs Código Legado
- (ESaaS 1.8) Garantia de Qualidade de Software: Testes
- (ESaaS 1.9) Produtividade: Clareza via Concisão, Síntese, Reuso, & Ferramentas

Objetivos do Curso

- Discutir os [**Princípios**] da Engenharia de Software compreendendo os novos desafios, oportunidades e problemas **AINDA** em aberto
 - SaaS/SOA, Computação em Nuvem, Código Legado, Testes, Produtividade...
- Desenvolver um sistema de informação [**SaaS**] da sua concepção à implantação
- Investigar como solucionar problemas não-técnicos dos clientes
- Novas abordagens: Ruby on Rails, BDD, TDD
- E do lado do cliente: HTML, CSS, AJAX, JavaScript
- Implantação por meio da Computação em Nuvem

Comunicação

- Site da disciplina [**material** e **informações**]:
 - <http://www.cin.ufpe.br/~if977>
- Para facilitar a comunicação via **email** que não acontecer pela lista, utilizem [**if977**] como parte do **assunto** do email.
- Comunidade da if977 no **Facebook**:
 - [if977] Engenharia de Software (BSI, CIn-UFPE)
 - <http://www.facebook.com/groups/229376600477111/>
 - <http://if977-2099-1.slack.com>
 - <https://github.com/IF977>

Exercício

- Configuração básica do curso
- Slack team: <http://if977-2017-2.slack.com> (@cin.ufpe.br)
- Proceda com a configuração correta do seu perfil Slack, fornecendo uma imagem, ajudará o professor e monitores a aprender seu nome. Carregue uma imagem atual sua (não mais ninguém, nem uma foto de desenho animado de você, etc.) em seu perfil. Além disso, certifique-se de ter seu primeiro e último nome como parte do seu perfil.

Avaliação

- A avaliação neste curso se dará da seguinte forma:
 - os aspectos teóricos serão avaliados por meio de **dois exercícios escolares**;
 - a prática será avaliada por meio de **projeto de software desenvolvido em equipe**.
- A média da disciplina será calculada da seguinte forma:
 - **Média = (3.5 * NotaEE1 + 3.5 * NotaEE2 + 3 * NotaProjeto) / 10**, onde
 - **NotaEE1** é a nota do Primeiro Exercício Escolar;
 - **NotaEE2** é a nota do Segundo Exercício Escolar;
- A nota do projeto (**NotaProjeto**) será calculada assim:
 - **NotaProjeto = (NotaEntrega1 + NotaEntrega2 + NotaEntrega3 + NotaEntrega4) / 4**, onde
 - **NotaEntregaX (X = 1 .. 4)** corresponde à nota referente à entrega de partes do projeto, sendo quatro ao todo.

Sobre o Projeto



- Nota para Equipe!
 - Equipe quem vai dividir os pontos entre os membros.
 - Avaliação por interação, por artefatos e por execução.
- Temática: **Sistemas de Informação para/com Dados Abertos**



Se preparando para
o Curso

Warm up

Ambiente e ferramentas

- Para reduzir os obstáculos de instalar e configurar o mesmo ambiente de desenvolvimento em diferentes computadores com diferentes sistemas operacionais, é recomendado o uso de uma máquina virtual que contém todos os software necessários para o curso
 - <http://www.saasbook.info/bookware-vm-instructions>
 - Cloud9: <https://c9.io>
 - Vagrant: <https://www.vagrantup.com>
 - RubyMine: <https://www.jetbrains.com/ruby/>
 - Visual Studio Code: <https://code.visualstudio.com/>
 - Sublime Text: <https://www.sublimetext.com/>
- Também é recomendado, para quem não é familiarizado com Ruby, que percorra a trilha de Ruby do Code Academy:
 - <http://www.codecademy.com/tracks/ruby> (ou similar)

Controle de Versão

- Controle de versão e ferramentas de controle de versão (e.g., git) são essenciais na ES
 - A VISUAL GUIDE TO VERSION CONTROL
 - <http://betterexplained.com/articles/a-visual-guide-to-version-control/>
- BASH SHELL COMMANDS REVIEW
 - <http://bit.ly/1ltGFyD> and follow along in the command terminal in your VM.
- GIT IMMERSION TUTORIAL
 - Seções 1-35 do GitImmersion tutorial (http://gitimmersion.com/lab_01.html) na sua VM.

Máquina Virtual

- Baixe a Oracle Virtual Box
- <http://www.virtualbox.org>



VirtualBox

Download VirtualBox

Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox platform packages.** The binaries are released under the terms of the GPL version 2.
 - [VirtualBox 4.3.12 for Windows hosts](#) ↗x86/amd64
 - [VirtualBox 4.3.12 for OS X hosts](#) ↗x86/amd64
 - [VirtualBox 4.3.12 for Linux hosts](#)
 - [VirtualBox 4.3.12 for Solaris hosts](#) ↗amd64
- **VirtualBox 4.3.12 Oracle VM VirtualBox Extension Pack** ↗All supported platforms
Support for USB 2.0 devices, VirtualBox RDP and PXE boot for Intel cards. See this chapter from the [User Manual](#) for an introduction to this Extension Pack. The Extension Pack binaries are released under the [VirtualBox Personal Use and Evaluation License \(PUEL\)](#).
Please install the extension pack with the same version as your installed version of VirtualBox!
If you are using [VirtualBox 4.2.24](#), please download the extension pack ↗here.
If you are using [VirtualBox 4.1.32](#), please download the extension pack ↗here.
If you are using [VirtualBox 4.0.24](#), please download the extension pack ↗here.

[About](#) [Screenshots](#) [Downloads](#) [Documentation](#) [End-user docs](#) [Technical docs](#) [Contribute](#) [Community](#)

[Login](#) [Preferences](#) [search...](#)

Máquina Virtual

- Instale a VirtualBox
- Baixe a imagem da VM
 - URL: <http://www.saasbook.info/bookware-vm-instructions>
 - saasbook-0.10.3.1.zip
- Descompacte o arquivo [atenção para o caminho]
- Execute a VirtualBox e crie a nova VM
- Quando o VM Wizard aparecer, selecione as opções:
 - operating system: Linux
 - version: Ubuntu
 - RAM base memory: ao menos 1024 MB
 - Selecione "Use existing hard disk" e selecione o arquivo .vdi que você baixou
- O login e a senha são “saasbook”
- Esse screencast (<http://vimeo.com/34754478>), explica como configurar o boot e o login na VM.

Recursos Online (Ferramentas)

- Online "get started" tutorials on ruby in case want to get feet wet before Ruby lectures:
 - tryruby.org (beginner)
 - rubymonk.com (beginner/intermediate)
 - rubykoans.com (intermediate/advanced)
- Pointers to online HTML/CSS tutorials in in case want to get feet wet before Rails material:
 - W3 Schools-free online tutorials related to all Web technologies

Leitura obrigatória

[http://bit.ly/if977-
multitasking](http://bit.ly/if977-multitasking)

**“Multitasking Damages
Your Brain And Career,
New Studies Suggest”**



The Myth of Multitasking

- <https://youtu.be/PriSFBu5CLs>
- Clifford Nass, Stanford
- Allowing yourself to respond to distraction (incoming email, IM, etc.) triggers small dopamine release
- Over time, you get addicted to it
- Result: Multitaskers waste far more brainpower than monotaskers when actually distracted
- Long-term effects can be hard to reverse

So during lecture...





Introdução a Engenharia de Software

Practical Software Engineering

Pergunta

Which aspect of the software lifecycle consumes the most resources?

- A. Design
- B. Development
- C. Testing
- D. Maintenance

Pergunta

Which aspect of the software lifecycle consumes the most resources?

- A. Design
- B. Development
- C. Testing
- D. Maintenance

Ranking Top 200 Jobs ('12)

1. Software Engineer

28. Civil Engineer

34. Programmer

40. Physician

104. Airline Pilot

Researches, designs, develops and maintains software systems along with hardware development for medical, scientific, and industrial purposes.

Income: \$88,142 (+25%)

47. Accountant

Organizes and lists the instructions for computers to process data and solve problems in logical order.

Income: \$71,178

60. Nurse

73. Electrician

87. Attorney

173. Flight Attendant

85. Firefighter

96. Newspaper Reporter

200. Lumberjack

Engenharia de Software

- As economias de **TODAS** as nações desenvolvidas são dependentes de software.
- Cada vez mais sistemas são controlados por software.
- A engenharia de software se dedica às teorias, métodos e ferramentas para desenvolvimento de software profissional
 - Sistemas **não-triviais**
 - Com base em um **conjunto de requisitos**

Se [ES] é tão popular, por que tantos desastres?

- [Procure na Wikipedia para mais detalhes]
- Mariner I (O hífen mais caro da história)
 - 1962: menos de cinco minutos depois do início de voo, o Mariner I explodiu, dando um prejuízo de US\$80 milhões (<http://gizmodo.uol.com.br/hifen-nasa/>)
- Therac-25: sobredosagem de radiação letal
 - 1985: Reutilização de SW de uma máquina com HW de bloqueio em uma máquina sem: SW bug => **3 morreram**
- Desintegração da “Mars Climate Orbiter”
 - 1999: os engenheiros espaciais se esqueceram de converter as medidas do sistema imperial para unidades métricas (pound-seconds vs. newton-seconds) => desperdiçou \$325M
- Projeto “FBI Virtual Case File” abandonado
 - 2005: abandonado após 5 anos de trabalho => desperdício de \$170M
- Explosão do foguete Ariane 5
 - 1996 => desperdiçou \$370M

An Agenda for the Future of Software [Engineering in Brazil]

Silvio Meira, UFPE

INVITED TALK, SBES 2015

Abstract: Never so many people and things depended so much on software as now. In practice, almost everything depends on software, in all aspects of human activities. Not only in economy: the behaviors of each person and our, as a group and society, are performances over - sometimes inside of - software. Moreover, all signals show that software relevance is going to increase. At the same time, we face non-trivial difficulties in acquiring, developing, deploying, maintaining, and evolving software, in spite of everything we have done, as software engineers, in the last five decades. This keynote talk will try to discuss a possible agenda for the future of software and its engineering to the next 50 years, tackling the problem, in the Brazilian context, as an special case and of our main interest.

Bio: Silvio Meira is a [retired] Full Professor of Software Engineering at CIn/UFPE, Associate Professor at FGV DIREITO RIO, one of the founders of CESAR and PORTO DIGITAL and founder of IKEWAI.

Investments of federal agencies on IT projects in 2014

Dept. of the Interior - \$1.1B

Dept. of State - \$1.4B

NASA - \$1.5B

Dept. of Energy - \$1.5B

Social Security Admin. -
\$1.9B

Dept. of Commerce -
\$1.9B

Dept. of Agriculture -
\$2.7B

Dept. of Justice -
\$2.7B

Dept. of
Transportation -
\$3.2B

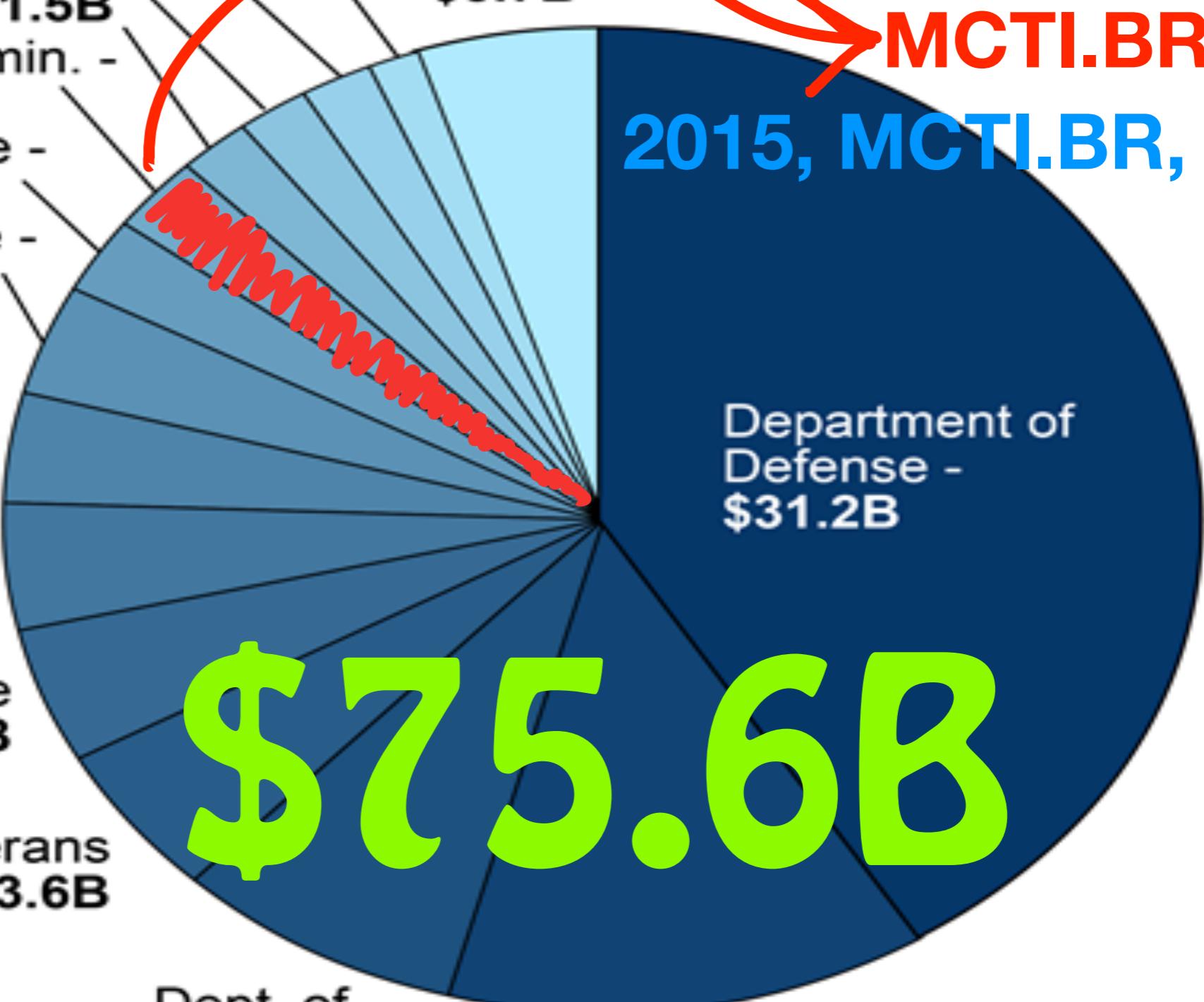
Dept. of the
Treasury - \$3.5B

Dept. of Veterans
Affairs - \$3.6B

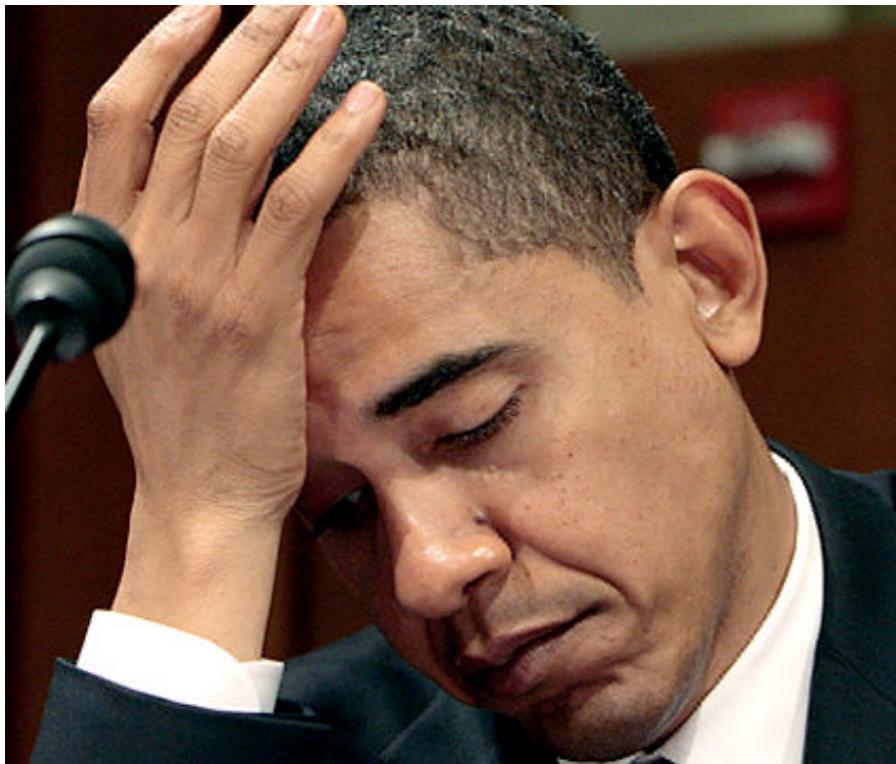
Dept. of
Homeland
Security -
\$6B

Dept. of Health and
Human Services -
\$9.8B

Other -
\$3.7B



Poor President Obama...



HealthCare.gov Learn Get Insurance Log in Espanol

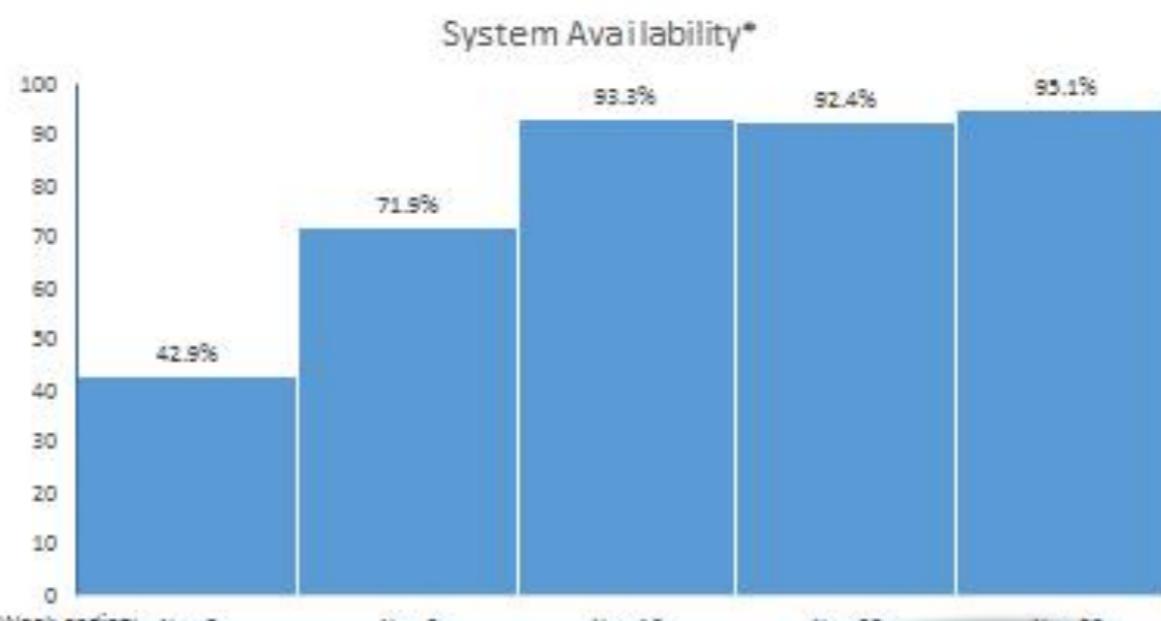
Individuals & Families Small Businesses All Topics Search SEARCH

The System is down at the moment.

We're working to resolve the issue as soon as possible. Please try again later.

Please include the reference ID below if you wish to contact us at 1-800-318-2596 for support.
Error from: https://www.healthcare.gov/marketplace/global/en_US/registration%23signUpStepOne
Reference ID: 0.cdd73b17.1380636213.edae7e9

Health Insurance Marketplace 181 DAYS LEFT TO ENROLL OCT 1 Open Enrollment Began JAN 1 Coverage Can Begin MAR 31 Open Enrollment Closes



* Excluding scheduled maintenance

3.555 projetos com custo **≥\$10M**,
2003-2012

apenas **6.4%** foram considerados um
“**sucesso**”

52% “**marromeno**”

41.6% “**fracasso**” total

"I spent \$319 million on a website and all I got was this bad press."

— Someone, somewhere in the U.S. Department of Health and Human Services (HHS)

Large federal information technology purchases have to end. Any methodology with a 94 percent chance of failure or delay, which costs taxpayers billions of dollars, doesn't belong in a 21st-century government.

Why the Government Never Gets Tech Right, NYT, 10/2013, nyti.ms/1YBZOVW; wapo.st/1VaXDtn; bit.ly/1WkOgo5

e
quando
roda



The devastating hack of the federal Office of Personnel Management, explained

Updated by Timothy B. Lee on September 23, 2015, 12:55 p.m. ET ▾ tim@vox.com

The Office of Personnel Management has been **forced to admit**, yet again, that the devastating hack of its computer systems was worse than the agency had previously acknowledged. The latest revelation: Fingerprint data from 5.6 million people has been stolen, a fivefold increase from the agency's previous estimate of 1.1 million. Overall, the agency estimates that hackers took information related to 22 million people.

bit.ly/1MKPJ5s

UK, HO, IMMIGRATION CASE WORK

Começou em 2010...

encerrado @ £350M, 2014

novo sistema encomendado...

gastará +£209M, 2016/7

“I just don’t think the UK government should be allowed to buy IT at all. Maybe give them abacuses, but they could still get those wrong.”



BR, SRF, SIC: Sistema Integrado de Crédito

**atrasado em 2011, quando a...
data limite ficou pra 2012;**

daí, foi pra 13, 14, 15, está em 16⁺...

**rodou em 4 unidades do SERPRO
tá de mudança pra BHZ!...**



Sistema que **subsidia todos os outros sistemas que necessitam efetuar cálculos de acréscimos legais, com base em toda a legislação e regras de acréscimos aplicáveis a créditos tributários, depósitos judiciais, depósitos sub júdice, direitos creditórios, compensações, restituições, resarcimentos, penalidades da rede arrecadadora, tratamentos de juros para tributos não vencidos, permitindo a realização de consolidações e imputações.**

~40KPF; esforço \approx 1GD, 2{LX, iP...}

“desenvolvimento” \approx 500.000 horas

custa ~R\$35K por UC

SRF \leadsto SERPRO = R\$120M

estimativa até 2014: R\$250M

perdas: 4 anos de SIMPLES*

1. sprints + iterations ≠ agile
healthcare.gov nunca foi testado de fato
2. um ‘sistema’ RESULTA de um ‘SISTEMA’
{não havia como ‘parar’ healthcare.gov...}
3. TESTE tem que ser parte da ENTREGA
e não apagar incêndios depois da ‘entrega’
4. de nada adianta INFORMATIZAR O CAOS
se os processos não estão bem definidos...
5. não há sistema PERFEITO; nem POR LEI...
governos tendem a insistir nisso; #FAIL!...
6. sistemas -online, high profile- São ALVOS
healthcare.gov nunca foi olhado assim

Como evitar o descrédito?

- Quais são as lições de 60 anos de Desenvolvimento de Software?
- Vamos revisar algumas abordagens ressaltando pontos fortes e fracos
- Experiência prática com uma boa abordagem para Software como Serviço



Software como Serviço

Software as a Service (SaaS)

Alvos do software

- SW Tradicional: código binário instalado e rodando totalmente no dispositivo do cliente
- Os usuários precisam atualizar repetidamente
 - Muitas versões de hardware e muitas versões de SO
 - Novas versões precisam passar por um ciclo de **release** extensivo para garantir a compatibilidade
- Uma alternativa onde o desenvolva-se o SW que necessita rodar em apenas uma plataforma de HW&SO?
 - Ciclo de releases rápido, sem atualizações dos usuários?

Software como Serviço (SaaS)

- SaaS entrega SW & dados como serviço através da rede via clientes “magros” (e.g., browser) rodando no cliente
 - Busca, redes sociais, streaming de vídeo
- Versões SaaS de SW tradicionais
 - Microsoft Office 365, Salesforce
- SaaS é um caminho [**revolucionário**] sem volta!

6 Razões para SaaS

1. Sem preocupações com o HW sobre instalação ou SO
2. Sem preocupações sobre perda de dados (no site remoto)
3. Fácil interação para grupos (mesmos dados)
4. Se o volume de dados é grande ou muda frequentemente, basta manter uma cópia no sitio central
5. Uma cópia do SW, ambiente de HW controlado => sem aborrecimentos com compatibilidade [desenvolvedores]
6. Uma cópia 1 => simplifica atualizações para desenvolvedores e sem CRs de atualização dos usuários

SaaS **S2** Metodologias Ágeis & Rails

- Atualizações frequentes => Ciclo de Vida Ágil
- Muitos frameworks para Ágil/SaaS
 - Django/Python, Zend/PHP, Spring/Java
- Vamos utilizar Ruby on Rails (“Rails”)
- Rails é um framework muito popular que utiliza Ruby (e.g., Twitter, Cloudfoundry, OpenNebula)
- Ruby, uma linguagem de script moderna: OO, funcional, gerenciamento automático de memória, tipos dinâmicos, reuso via mix-ins, síntese via metaprogramação

Por que gastar tempo com Rails?

- 15 semanas para aprender:
 - SaaS, Agile, Pair Programming, Behavior Driven Design, LoFi UI, Storyboards, User Stories, Test Driven Development, Enhance Legacy Code, Scrum, Velocity, JavaScript, Design Patterns, UML, Deployment, Security
- A única esperança é contar com um ambiente altamente produtivo (linguagens, ferramentas, frameworks...): Acredito que Rails é a melhor escolha
 - Sugestão: “Crossing the Software Education Chasm”, A. Fox, D. Patterson, Comm. ACM, May 2012
 - <http://bit.ly/1b9QbFj>

Pergunta

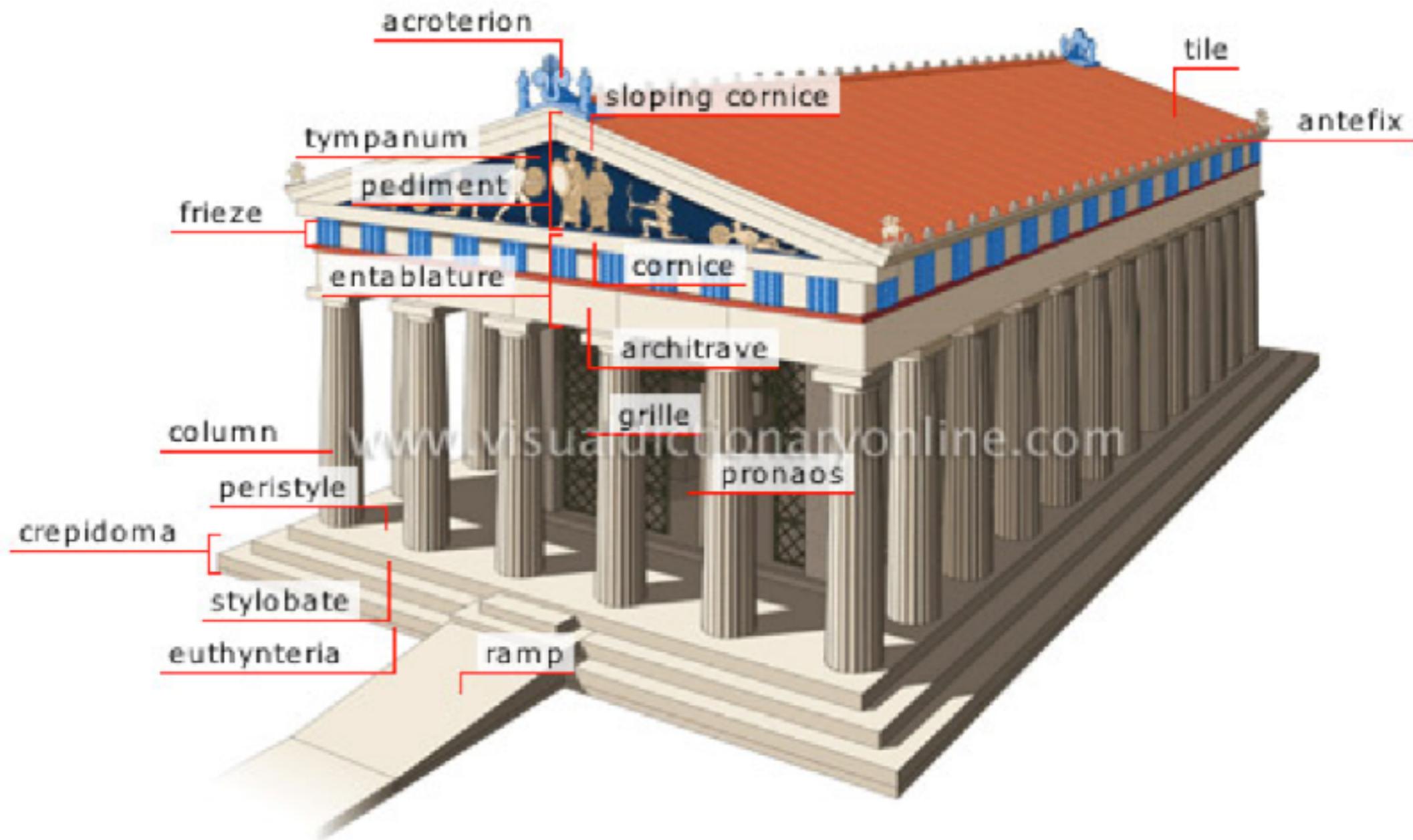
Qual o argumento mais **FRACO** para a popularidade das apps como SaaS do Google ?

- A. Não perca dados: Gmail
- B. Grupos cooperativos: Documentos
- C. Grandes conjuntos de dados: Youtube
- D. Sem atualizações intrusivas ao melhorar o app: Busca

Pergunta

Qual o argumento mais **FRACO** para a popularidade das apps como SaaS do Google ?

- A. Não perca dados: Gmail
- B. Grupos cooperativos: Documentos
- C. Grandes conjuntos de dados: Youtube
- D. Sem atualizações intrusivas ao melhorar o app: Busca



Arquitetura Orientada a Serviço

Software Oriented Architecture
(SOA)

Arquitetura de Software

Você pode/consegue projetar um software de forma que seja possível recombinar módulos independentes para ofertar diferentes apps sem muito esforço de programação?

Arquitetura Orientada a Serviço

- SOA: Arquitetura de Software onde todos os componentes são projetados como serviço
- Apps são compostas por serviços interoperáveis
 - Fácil de derivar uma nova versão para um conjunto de usuários
 - Também é fácil para se recuperar de um erro de projeto
- Contraste para “SW silo” sem uma API interna

CEO: Amazon shall use SOA!

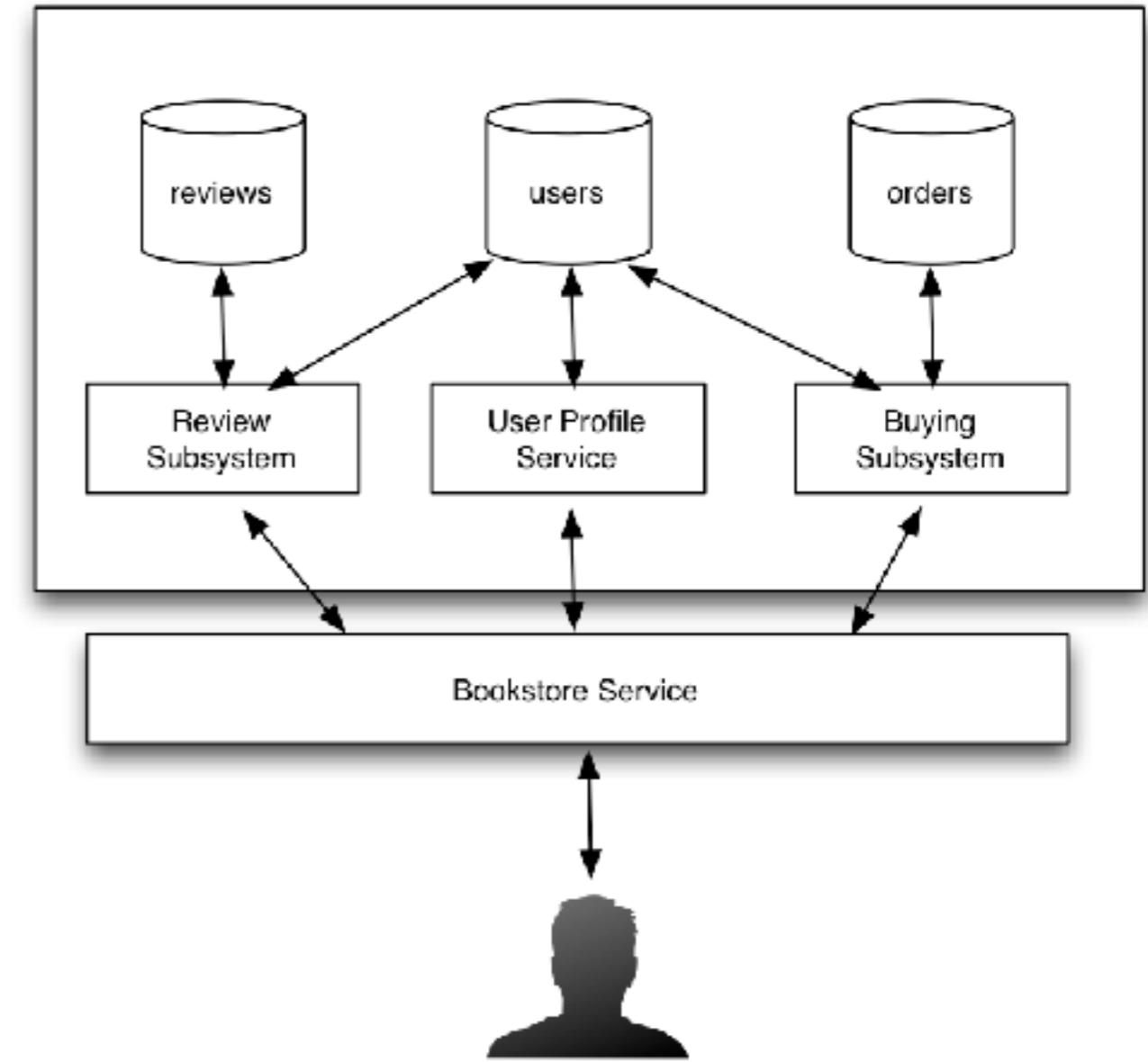
1. “All teams will henceforth expose their data and functionality through service interfaces.
2. “Teams must communicate with each other through these interfaces.
3. “There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

CEO: Amazon shall use SOA!

4. “It doesn't matter what [API protocol] technology you use.
5. “Service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
6. “Anyone who doesn't do this will be fired.
7. “Thank you; have a nice day!”

Bookstore: Silo

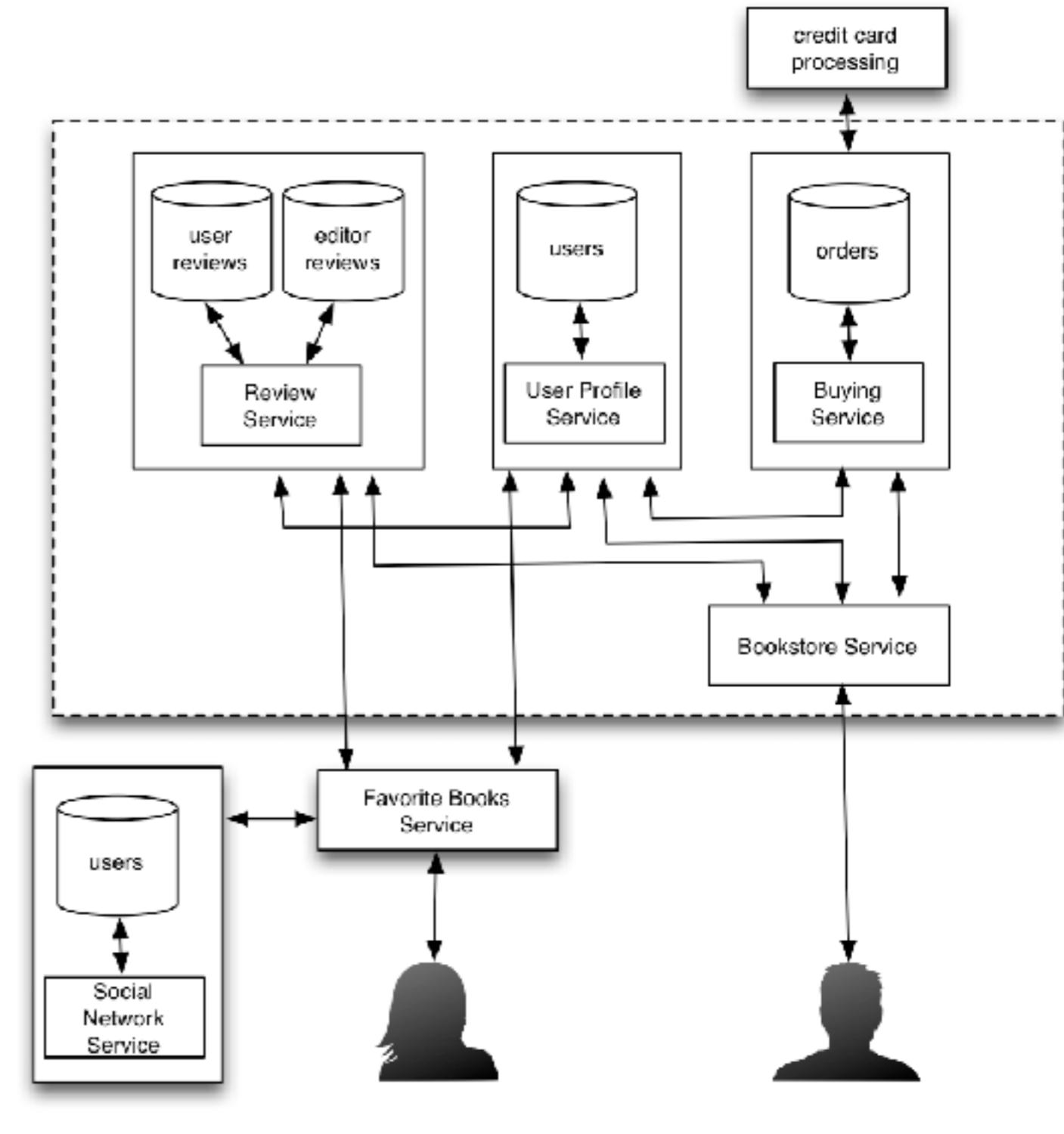
- Subsistemas internos podem compartilhar dados diretamente
 - User profile Service
- Todos os subsistemas “dentro” de uma única API (“Bookstore”)



(Figure 1.3, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Beta edition, 2012.)

Bookstore: SOA

- Subsistemas independentes, como se estivessem em datacenters separados
 - User Service API
- Pode recombinar serviços para criar novos serviços (“Favorite Books”)



(Figure 1.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Beta edition, 2012.)

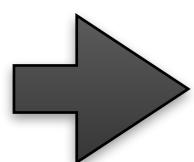
Pergunta

Quais afirmações **NÃO** são verdades sobre SOA?

- A. SOA não afeta desempenho
- B. Sistemas silo provavelmente ficam fora do ar com uma falha, SOA deve saber lidar com falhas parciais
- C. SOA melhora a produtividade dos desenvolvedores por meio do reuso
- D. Nenhum serviço pode acessar os dados de outro serviço; pode apenas fazer requisições para estes dados através de uma API

Pergunta

Quais afirmações **NÃO** são verdades sobre SOA?

- 
- A. SOA não afeta desempenho
 - B. Sistemas silo provavelmente ficam fora do ar com uma falha, SOA deve saber lidar com falhas parciais
 - C. SOA melhora a produtividade dos desenvolvedores por meio do reuso
 - D. Nenhum serviço pode acessar os dados de outro serviço; pode apenas fazer requisições para estes dados através de uma API



Computação em Nuvem

Cloud Computing

Qual o HW ideal para SaaS?

- Amazon, Google, Microsoft... Desenvolveram sistemas de HW para rodar SaaS
- O Que eles usam, em termos de infraestrutura de HW: Mainframes? Supercomputers?
- Como desenvolvedores de SW independentes constroem apps SaaS e competem sem um investimento em HW similar a Amazon, Google, Microsoft?

Infraestrutura SaaS?

- 3 Demandas de infraestrutura para SaaS
 1. Comunicação: permitir que os consumidores interajam com o serviço
 2. Escalabilidade: flutuações na demanda durante adição de novos serviços para novos clientes rapidamente
 3. Confiabilidade: disponibilidade contínua de serviço e comunicação 24x7

Serviços em Clusters

- Cluster: aglomerado de computadores conectados
 1. Mais escalável do que servidores convencionais
 2. Mais barato do que servidores convencionais
 - 20X para equivalentes vs. Servidores maiores
 3. Poucos operadores para 1000s servidores
 - Seleção cuidadosa de HW e SW idênticos
 - Monitores de Máquinas Virtuais simplifica a operação
 4. Confiabilidade via redundância extensiva

Warehouse de Computadores Escaláveis

- Clusters cresceu de 1000 para 100 mil servidores com base na demanda do cliente para aplicativos SaaS
- Economias de escala empurraram o custo dos grandes data centers em fatores de 3x a 8x
 - Comprar, hospedar, operar 100K vs 1K
- Data centers tradicionais utilizados, cerca de 10% - 20%
- Modelos de negócio baseados em pay-as-you-go trazem benefícios concretos

Computação Utilitária / Computação em Nuvem Pública

- Oferece computação, armazenamento, comunicação a centavos por hora
- Sem adicionais por escalabilidade
 - 1000 computadores @ 1 hora, ou
 - 1 computador @ 1000 horas
- Ilusão de infinita escalabilidade para o usuário
 - Tantos computadores quanto você puder dispor
- Exemplos: Amazon Web Services, Google App Engine, Microsoft Azure

2013 AWS Instances & Prices

Instance Type	Per Hour	\$ Ratio to small	Virtual Cores	Compute Units	Memory (GiB)	Storage (GB)
m1.small	\$0.06	1.0	1	1.0	1.7	1 x 160
m1.medium	\$0.12	2.0	1	2.0	3.8	1 x 410
m1.large	\$0.24	4.0	2	4.0	7.5	2 x 420
m1.xlarge	\$0.48	8.0	4	8.0	15.0	4 x 420
m3.xlarge	\$0.50	8.3	4	13.0	15.0	EBS
m3.2xlarge	\$1.00	16.7	8	26.0	30.0	EBS
c1.medium	\$0.15	2.5	2	5.0	1.7	1 x 350
c1.xlarge	\$0.58	9.7	8	20.0	7.0	4 x 420
cc2.8xlarge	\$2.40	40.0	32	88.0	60.5	4 x 840
m2.xlarge	\$0.41	6.8	2	6.5	17.1	1 x 420
m2.2xlarge	\$0.82	13.7	4	13.0	34.2	1 x 850
m2.4xlarge	\$1.64	27.3	8	26.0	68.4	2 x 840
cr1.8xlarge	\$3.50	58.3	32	88.0	244.0	2 x 120 SSD
hi1.4xlarge	\$3.10	51.7	16	35.0	60.5	2 x 1024 SSD
hs1.8xlarge	\$4.60	76.7	16	35.0	117.0	24 x 2048
t1.micro	\$0.02	0.3	1	varies	0.6	EBS
cg1.4xlarge	\$2.10	35.0	16	33.5	22.5	2 x 840

Computação Utilitária

- Competição dos Top 500 supercomputadores
- 532 Eight Extra Large (@ \$2.40/hour), 17000 cores = 240 TeraFLOPS
- 72nd/500 supercomputer @ ~\$1300 per hour
- Cartão de crédito => até 1000s computadores
- FarmVille no AWS
 - Primeiro grande jogo online: 5M usuários
 - E se a startup tivesse que construir o data center?
 - 4 dias = 1M; 2 meses = 10 M; 9 meses = 75M

IBM Watson para alugar?

- Campeão do Jeopardy
- Hardware: 90 IBM Power 750 servers
- 3.5 GHz 8 cores/server
- $90 @ \$2.40/\text{hour} = \$200/\text{hour}$
- Custo de um advogado ou contador...
- Para quais tarefas uma máquina com IA seria tão boa quanto uma pessoa muito bem treinada @ \$200/hora?
- O que isso pode significar para a sociedade?

Pergunta

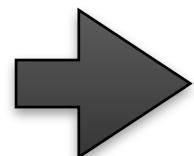
Quais afirmações **NÃO** são verdade sobre SaaS, SOA e Computação em Nuvem?

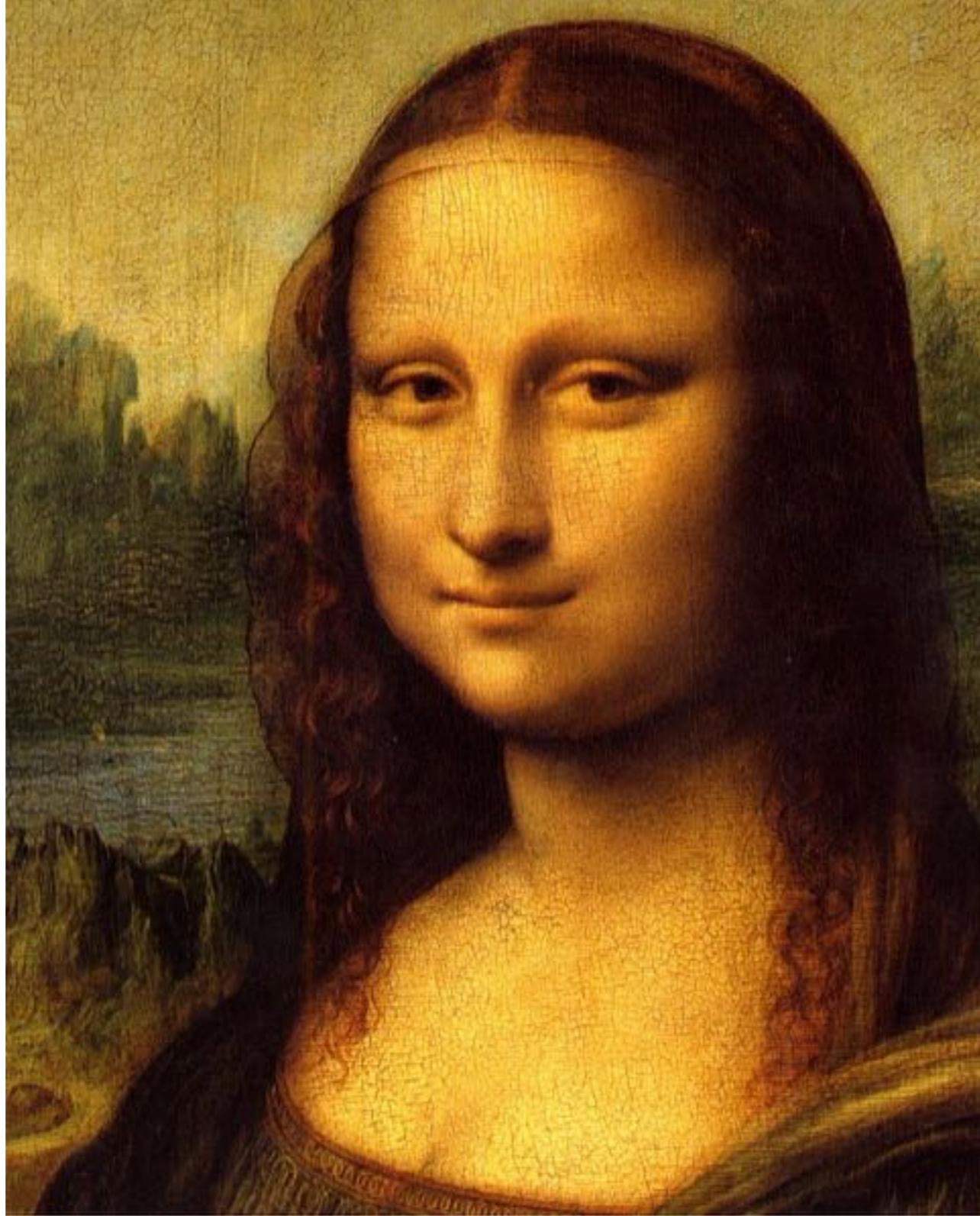
- A. Clusters são coleções de servidores comuns conectados por switches em uma LAN
- B. A Internet provê a comunicação para SaaS
- C. Computação em Nuvem utiliza clusters de HW + camadas de SW utilizando redundância para confiabilidade
- D. Data centers privados podem custar o mesmo que um Warehouse de Computadores Escaláveis se eles apenas se comprarem o mesmo tipo de HW

Pergunta

Quais afirmações **NÃO** são verdade sobre SaaS, SOA e Computação em Nuvem?

- A. Clusters são coleções de servidores comuns conectados por switches em uma LAN
- B. A Internet provê a comunicação para SaaS
- C. Computação em Nuvem utiliza clusters de HW + camadas de SW utilizando redundância para confiabilidade
- D. Data centers privados podem custar o mesmo que um Warehouse de Computadores Escaláveis se eles apenas se comprarem o mesmo tipo de HW





SW Legado vs SW Bonito

Estética de programação

- Eu me importo com o que os outros pensam do meu código?
- Se funcionar, não importa com o que o código se parece?

SW Legado vs SW Bonito

- **Código legado:** SW antigo que continua em uso (atende aos requisitos dos clientes) mas de difícil evolução [problemas de projeto ou tecnologia ultrapassada]
 - **60%** de custo adicional de manutenção para novas funcionalidades em SW legado
 - **17%** de custo adicional para correção de bugs
- **Código bonito:** atende as necessidades dos clientes e é fácil de evoluir

Código legado: vital porém ignorado

- Não existe nos cursos tradicionais e nos livros de ES
- Principal resposta a uma pesquisa feita com experts da indústria sobre o que eles acham que deveria ser visto em um novo curso de ES
 - Salve trabalho pela reutilização de código existente (e.g., open source)
- Iremos falar mais sobre SW legados e padrões de programação mais tarde, no curso
 - Irão ajudar a entender como construir um código bonito

Pergunta

Que tipo de SW é considerado uma falha épica?

- A. Código Bonito
- B. Código Legado
- C. Código inesperadamente com vida curta
- D. Tanto alternativas 2 e 3

Pergunta

Que tipo de SW é considerado uma falha épica?

A. Código Bonito

B. Código Legado

→ C. Código inesperadamente com vida curta

D. Tanto alternativas 2 e 3



Garantia de Qualidade e Testes

Qualidade de Software

- Qualidade do produto (em geral): “adequação ao uso”
 - Valor de negócio para o cliente e o produtor
 - Garantia de Qualidade: processos/padrões => produtos com alta qualidade & melhoria da qualidade
 - Quality Assurance
- Qualidade de SW
 1. Satisfaz necessidades do cliente: fácil de usar, obtém respostas corretas, não falha, ...
 2. Fácil para o desenvolvedor debugar e evoluir
- SW QA: garante a qualidade e melhora os processos organização

Garantia [Assurance]

- **Verificação**: “construímos a coisa **certa**? ”
 - Está de acordo com a especificação?
 - Ex: inspeção de código, análise estática
- **Validação**: “construímos **certa** a coisa?”
 - É o que o cliente quer? A especificação está correta?
 - Exs: testes, animação de especificações
- Geralmente HW foca em **verificação**
- Geralmente SW foca em **validação**
- Testes -> SW Quality Assurance

Testes

- É inviável termos testes exaustivos [\$\$\$]
- “***Divide et Conquer***”: realizar diferentes testes em diferentes fases do desenvolvimento
 - Um nível superior não refaz os testes do inferior

Níveis de Testes

- Testes de **sistema** ou **aceitação**: programa [integrado] atende suas especificações
- Teste de **integração**: as interfaces entre as unidades possuem pressupostos consistentes, comunicam-se corretamente
- Teste de **módulo** ou **funcional**: através das unidades individuais
- Teste de **unidade** ou **unitário**: o método faz o que era esperado fazer

Mais Testes

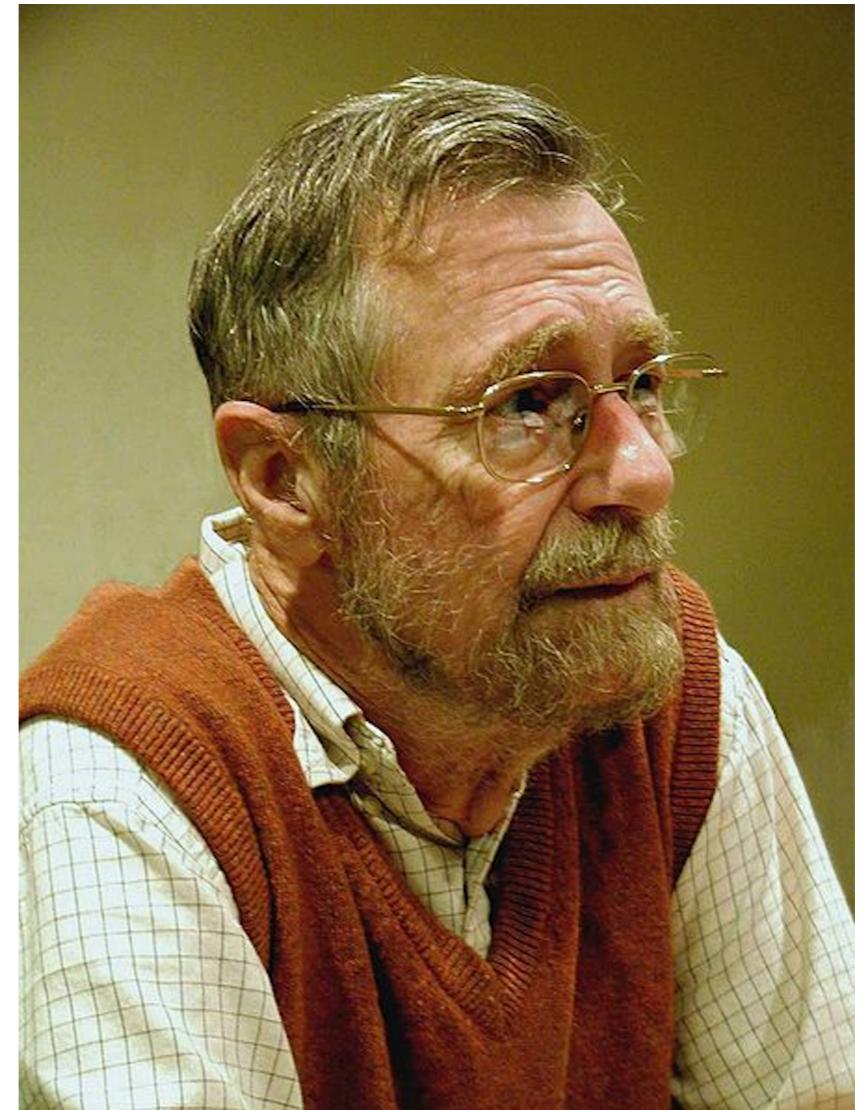
- **Cobertura**: % de código coberto pelos testes
- **Testes de Recessão**: automaticamente reexecuta testes passados para atestar que modificações não quebram o que estava ok
- **Teste de Integração Contínua**: testes de regressão contínua vs. Fases finais
- Ágil -> Test Driven Design (TDD)
 - Escreva os testes **ANTES** de escrever o código que você espera (testes guiam a codificação)

Limites dos Testes

Teste de programas podem ser utilizados para mostrar a **presença** de bugs, mas nunca para mostrar sua **ausência!**

Edsger W. Dijkstra

(recebeu em 1972 o Prêmio Turing por sua contribuição fundamental para o desenvolvimento das Linguagens de Programação)



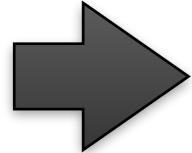
Pergunta

Qual afirmação **NÃO** é verdade sobre testes?

- A. Com melhor cobertura de testes, você tem mais chances de capturar falhas
- B. Embora difícil de alcançar, testes com cobertura de 100% garantem a confiabilidade do projeto
- C. Cada teste de um nível superior delega mais testes detalhados para os níveis inferiores
- D. Testes unitários funcionam com uma única classe e teste de módulo funciona através das classes

Pergunta

Qual afirmação **NÃO** é verdade sobre testes?

- A. Com melhor cobertura de testes, você tem mais chances de capturar falhas
-  B. Embora difícil de alcançar, testes com cobertura de 100% garantem a confiabilidade do projeto
- C. Cada teste de um nível superior delega mais testes detalhados para os níveis inferiores
- D. Testes unitários funcionam com uma única classe e teste de módulo funciona através das classes

Produtividade

Produtividade: Concisão,
Síntese, Reutilização e
Ferramentas

Produtividade

- Lei de Moore: 2X recursos/1.5 ano
 - Projetos de HW ficam maiores
 - Processadores mais rápidos e memórias maiores
 - Projetos de SW ficam maiores
 - Difícil de melhorar a produtividade de HW e SW
- 4 técnicas
 1. Clareza via concisão
 2. Síntese
 3. Reutilização
 4. Automação e ferramentas

Clareza via concisão

1. Sintaxe: curta e de fácil leitura

assert_greater_than_or_equal_to(a,7)

vs _____

Clareza via concisão

1. Sintaxe: curta e de fácil leitura

```
assert_greater_than_or_equal_to(a,7)
```

vs **a.should be ≥ 7**

2. Aumente o nível de abstração

- Linguagens de Programação de Alto Nível (HLL) vs. Linguagens assembly
- Gerenciamento automático de memória (Java vs C)
- Linguagens de script: reflexão, metaprogramação (síntese)

Síntese

- Síntese de Software
 - BitBit: geração de código para atender alguma situação e remoção de testes condicionais
- Estágio da pesquisa: Programação por Exemplo

Reúso

- Reutilizar código ou escrever código?
- Técnicas (cronologicamente)
 1. Procedures e funções
 2. Bibliotecas padronizadas (reuso de uma vez só)
 3. POO: reuso e gerenciamento de coleções de tarefas
 4. Padrões de projeto: reuso de uma estratégia geral independente da implementação

Automação e ferramentas

- Substituição de tarefas tediosas e manuais por automação para economizar, melhorando a acurácia
 - Novas ferramentas podem tornar a vida melhor! (i.e. Make, Ant)
- Preocupações com novas ferramentas: Confiabilidade, Qualidade de UI...
- Bons desenvolvedores devem aprender repetidamente como usar novas ferramentas: **curva de aprendizado**
 - Muitas chances na nossa proposta: Cucumber, RSpec, Pivotal Tracker, Trello

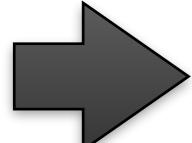
Pergunta

Qual afirmação é VERDADE sobre produtividade?

- A. Copiar e colar código é outra boa alternativa para reutilização
- B. Metaprogramação ajuda a produtividade através da síntese de programas
- C. Entre as 4 razões para produtividade, a primeira para HLL é reuso
- D. Em uma sintaxe concisa é mais provável ter menos erros e ter melhor manutenibilidade

Pergunta

Qual afirmação é VERDADE sobre produtividade?

- A. Copiar e colar código é outra boa alternativa para reutilização
-  B. Metaprogramação ajuda a produtividade através da síntese de programas
- C. Entre as 4 razões para produtividade, a primeira para HLL é reuso
- D. Em uma sintaxe concisa é mais provável ter menos erros e ter melhor manutenibilidade

Dry

“Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.”

• Andy Hunt and Dave Thomas, 1999

- **Don't Repeat Yourself (DRY)**
 - Não queira encontrar uma série de locais onde a mesma correção deva ser aplicada!
 - Refatore sempre seu código!

Sumarizando...

- Neste curso veremos Princípios da Engenharia de Software demonstrados de forma prática
 - Proposta de projeto para times de até 5 pessoas para lidar com desenvolvimento de um app no paradigma SaaS
- Ciclo de Vida Ágil – trabalhando em iterações e protótipos funcionais
- *Test Driven Development*: unitários a aceitação
- Produtividade: Concisão, Síntese, Reuso, Ferramentas & Automação
- SaaS traz menos aborrecimentos para desenvolvedores e usuários