Engenharia de Software

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: @vinicius3w :: viniciusgarcia.me

[IF977] Engenharia de Software
http://bit.ly/vcg-es

Licença do material

Este Trabalho foi licenciado com uma Licença

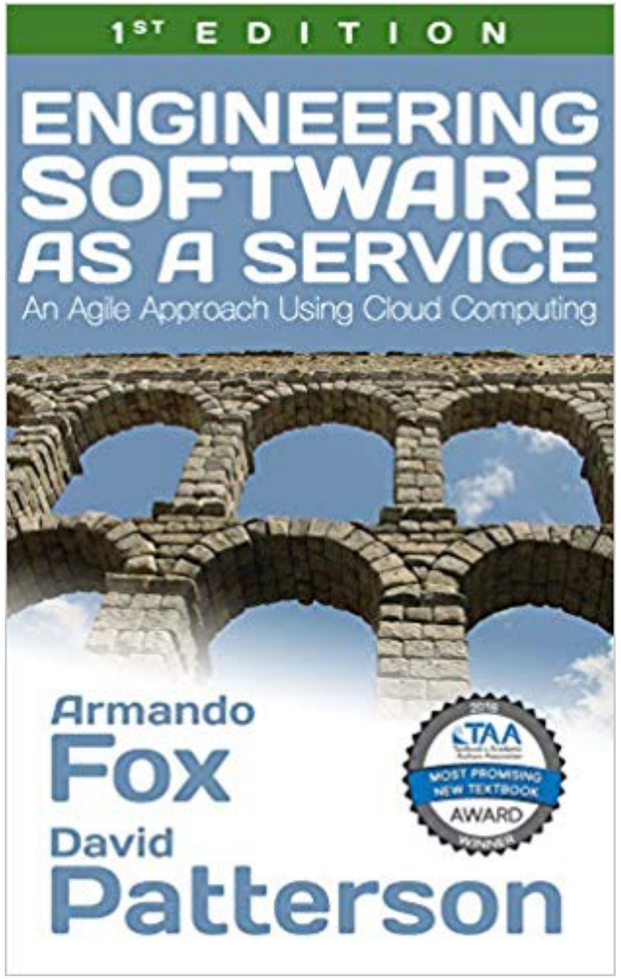
Creative Commons - Atribuição-NãoComercial-Compartilhalgual 3.0 Não Adaptada

Mais informações visite

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

Referências

- A biblioteca do Desenvolvedor de Software dos dias de hoje
  - http://bit.ly/31WYK5f
- 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
  - http://www.saasbook.info/
- Marco Tulio Valente. Engenharia de Software Moderna
  - https://engsoftmoderna.info/



Apertem os cintos!

Quem é?

github.com/chrislgarry/Apollo-11

Apollo 8 software team at MIT, c.1965

O que é Engenharia de Software (vs. programação)?

O que é Engenharia de Software (vs. programação)?

- "**Multi-person** development of **multi-version** programs"

- "…**combining separately** written programs and making them **suitable** for **use by people who had not written them**… These topics usually received little or no attention in traditional programming courses"

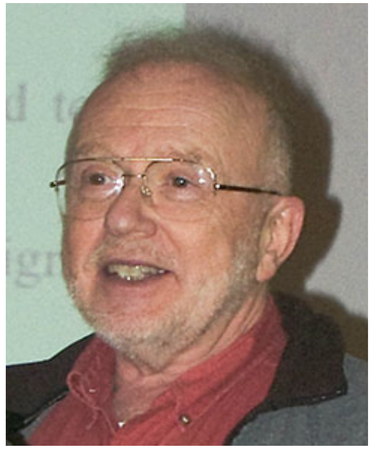


Photo credits: David Parnas: CC-BY-SA Hubert Baumeister, Wikimedia Commons ticket #2008072110008854. Fred Brooks: CC-BY-SA-3.0 ©SD&M, Wikimedia Commons

What Makes a Great Software Engineer?

- P. Li, A. Ko, J. Zhu, Microsoft & Univ. of Washington, Proc. ICSE 2015
- 59 interviews with recognized SE experts at Microsoft

Always improving
Passionate
Technically open-minded
Data-driven

Personal Characteristics

Decision Making

Knows people & organization
Updates mental models when learn new skills/facts/ context
Consider situation at multiple levels when making judgments
Can reason about complex & intertwining ideas

Helps others understand by tailoring explanations to them
Creates shared success possibly via personal compromises
Creates "safe haven" where others can learn from mistakes
Gives honest feedback

Team Skills

Technical Skills

Elegant solutions
Creative thinking when faced with limitations of current solutions
Anticipates technical needs based on prior experience

Our goal is for you to learn **disciplined**, **customer-focused**, **agile** software engineering of **information-system—as-a-service** (EISaaS)

Comunicação

- Site da disciplina [**material** e **informações**]:
  - https://github.com/IF977/IF977
- Para facilitar a comunicação via email que não acontecer pela lista, utilizem [if977] como parte do assunto do email.
- Comunicação
  - Canal da disciplina no **Telegram**
  - Repositório de projetos: https://github.com/IF977

Avaliações

- 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:
  - NotaIndivProjeto = (NotaProjeto/NumMembros) | NotaAttrib, onde
    - NumMembros = Número de membros na equipe
    - NotaProjeto >= NumMembros * 10
    - NotaAttrib = Nota atribuída pela equipe após a divisão

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

Expectation setting

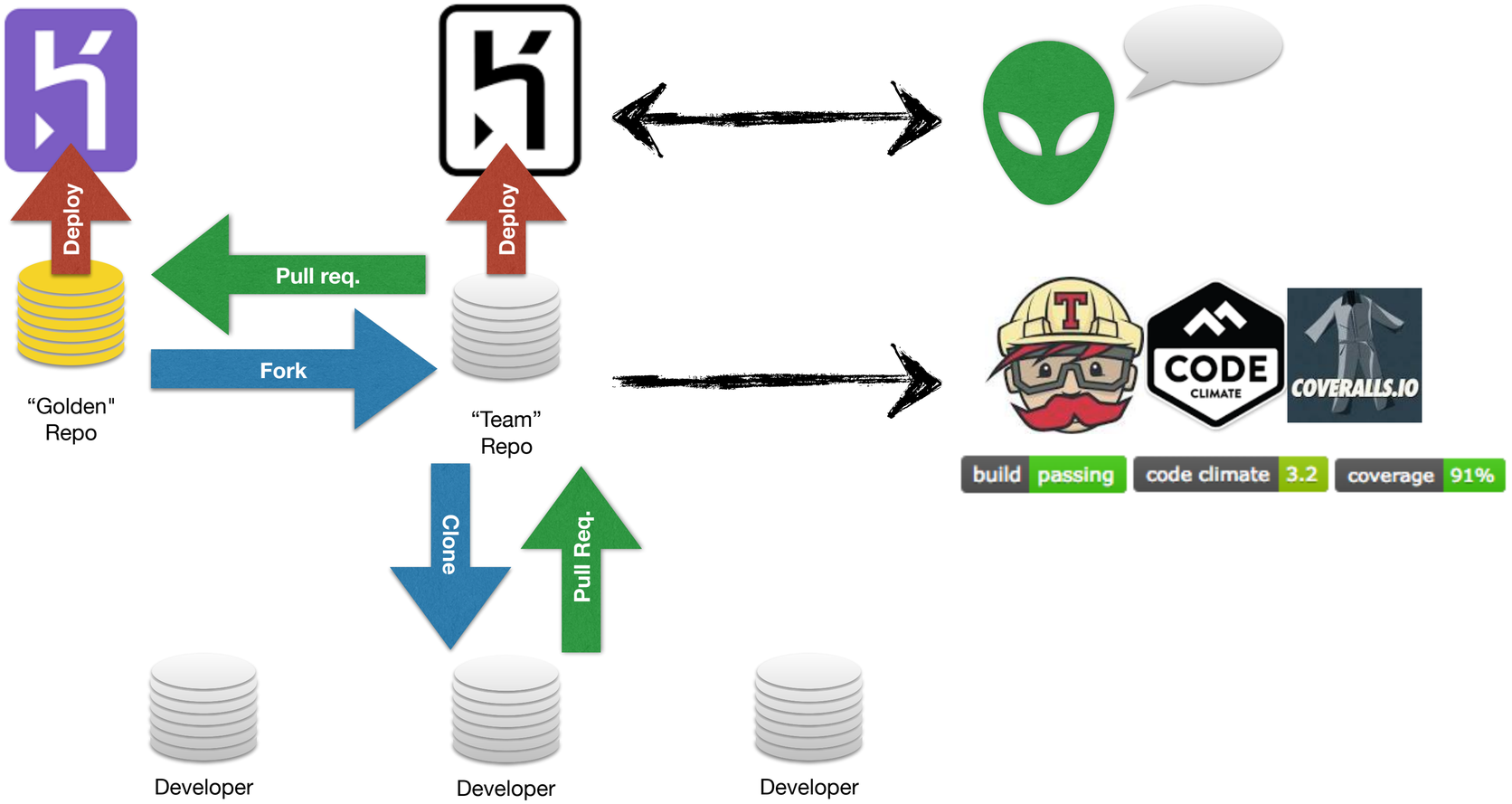
- Assumption 1: you know how to code
- Assumption 2: have GitHub (Student) account
- Goal: prepare you to function as member of professional software engineering community
  - Will this class teach me JS/React? Ruby/Rails?
  - Will I learn Ruby/Rails/JavaScript/React?
  - **Our goal: teach you how to learn them**
    - Hint 1: videos, reading, etc. are not sufficient
    - Hint 2: but they are necessary

What is missing from our student's technical skills?

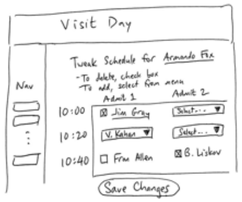
1. Legacy code
2. Working with non-technical customers
3. Testing

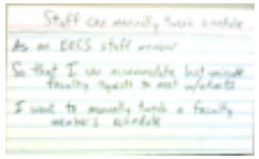
Project theme:
Do well by doing good

ESSI em dois slides



ESSI em dois slides





Conversar com cliente

Lo-Fi UI Mockup

User stories & scenarios

Legacy Code

Design Patterns

BDD / User Stories

Test-first Dev. (unit/func)

Measure Velocity

Deploy

How to Have a Bad Experience in this course

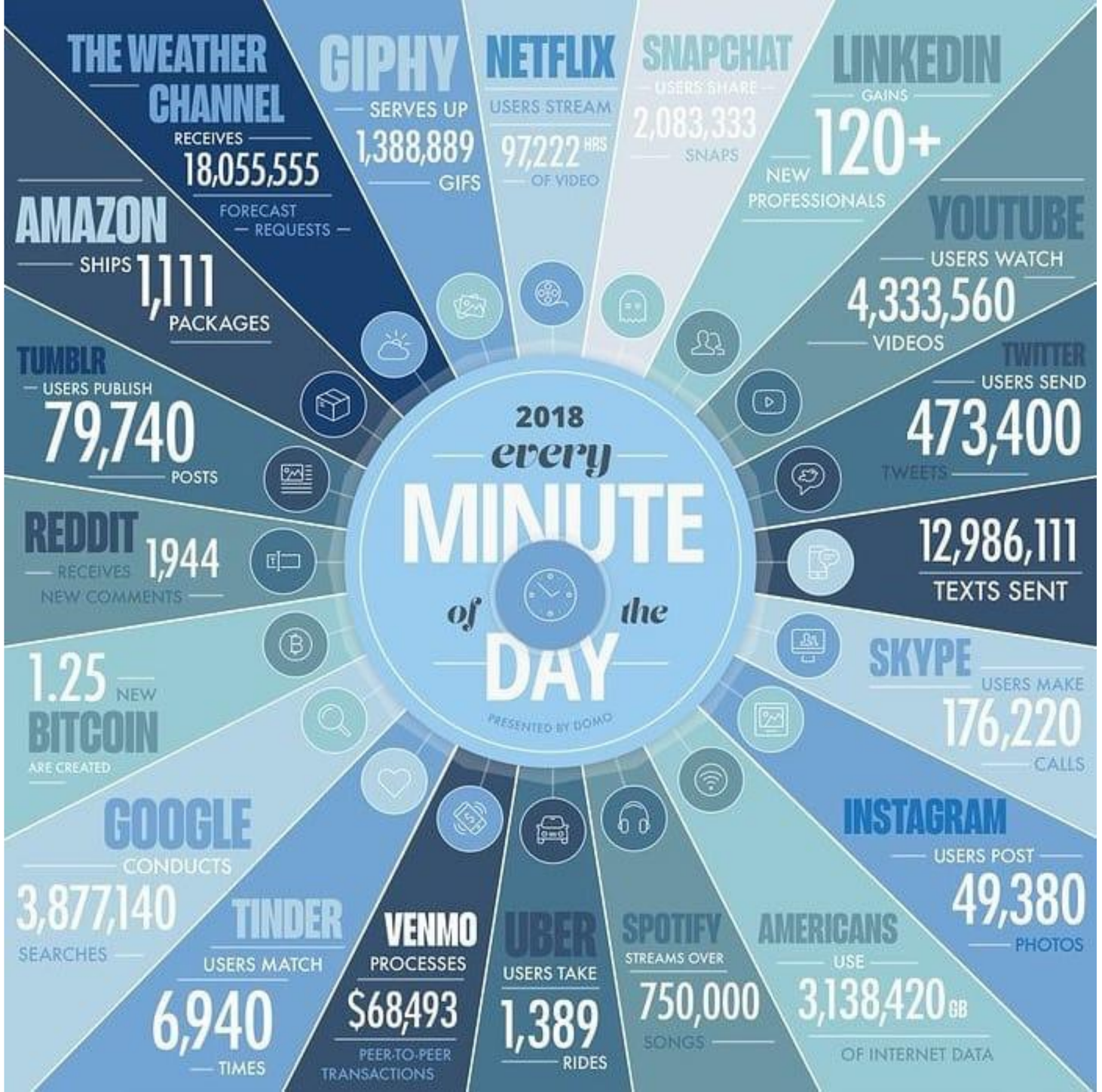
- Skip lecture/section (physically or mentally)
- Blow off/copy/don't do own work on HWs
- Work alone, don't engage peers
- Resist new tools/techniques
- Ignore process, code as you always have
- Go for points vs. learning
- Assume reading/lecture/videos == learning
- Cheat (violate Honor Code)
- Multitask (email, IM, etc.) while coding

Introdução a Engenharia de Software

Practical Software Engineering

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**


2018
every
MINUTE
of the
DAY
PRESENTED BY DOMO
THE WEATHER CHANNEL
RECEIVES
18,055,555
FORECAST REQUESTS
GIPHY
SERVES UP
1,388,889
GIFS
NETFLIX
USERS STREAM
97,222 HRS
OF VIDEO
SNAPCHAT
USERS SHARE
2,083,333
SNAPS
LINKEDIN
GAINS
120+
NEW PROFESSIONALS
AMAZON
SHIPS 1,111 PACKAGES
YOUTUBE
USERS WATCH
4,333,560
VIDEOS
TUMBLR
USERS PUBLISH
79,740
POSTS
TWITTER
USERS SEND
473,400
TWEETS
REDDIT
RECEIVES 1,944
NEW COMMENTS
12,986,111
TEXTS SENT
1.25 NEW
BITCOIN
ARE CREATED
SKYPE
USERS MAKE
176,220
CALLS
GOOGLE
CONDUCTS
3,877,140
SEARCHES
INSTAGRAM
USERS POST
49,380
PHOTOS
TINDER
USERS MATCH
6,940
TIMES
VENMO
PROCESSES
$68,493
PEER-TO-PEER TRANSACTIONS
UBER
USERS TAKE
1,389
RIDES
SPOTIFY
STREAMS OVER
750,000
SONGS
AMERICANS
USE
3,138,420 GB
OF INTERNET DATA

Software’s greatest hits

- June 4, 1996 -- Ariane 5 Flight 501
  - https://youtu.be/kYUrqdUyEpI
- First Flight 501's backup computer crashes
  - https://youtu.be/2eQpUgHkBcg
- For more: History's Worst Software Bugs
  - https://www.wired.com/2005/11/historys-worst-software-bugs/

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

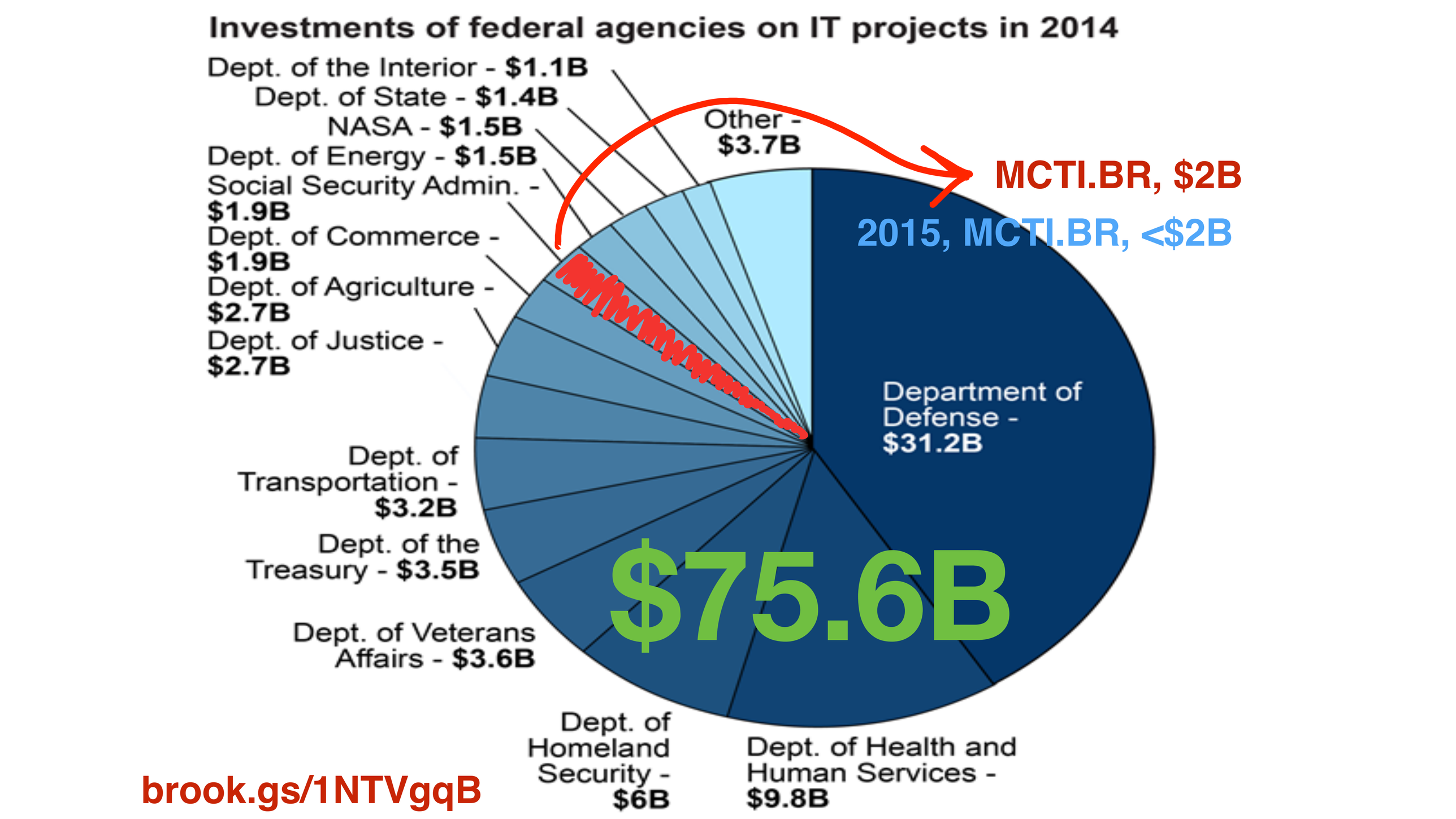
Silvio Meira, UFPE

INVITED TALK, SBES 2015

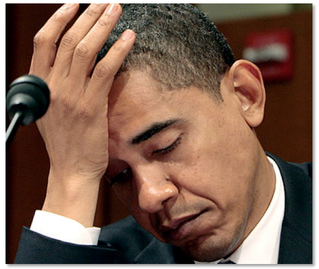
Abstract: Never so many people and things depended so much on software as now. In practice, almost everything depends on software, in all aspects of human activities. Not only in economy: the behaviors of each person and our, as a group and society, are performances over - sometimes inside of - software. 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.

https://www.youtube.com/watch?v=eujhiejLL7c


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
Other - $3.7B
MCTI.BR, $2B
2015, MCTI.BR, <$2B
Department of Defense - $31.2B
Dept. of Transportation - $3.2B
Dept. of the Treasury - $3.5B
Dept. of Veterans Affairs - $3.6B
$75.6B
Dept. of Homeland Security - $6B
Dept. of Health and Human Services - $9.8B
brook.gs/1NTVgqB

Poor President Obama...



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

6 Software Development Lessons From Healthcare.gov's Failed Launch, CIO, 11/2013, bit.ly/1WkOgo5

~1968: bring engineering discipline to software

- Why can’t building software be like building a bridge? Safe, predictable cost & quality…
  - (though 90% infrastructure projects late/over $)
- “Plan-and-Document”
  - Before coding, project manager makes plan
  - Write detailed documentation all phases of plan
  - Progress measured against the plan
  - Changes to project must be reflected in documentation and possibly to plan

1º Processo de Desenvolvimento: Waterfall (1970), 5 fases

- 5 fases do ciclo de vida Waterfall ou Processo de Desenvolvimento em Cascata
  - A.K.A. “Big Design Up Front” ou BDUF

- Análise e definição de requisitos
- Projeto de sistema e software
- Implementação e teste de unidade
- Integração e teste de sistema
- Operação e manutenção

- Primeiro modelo a organizar as atividades de desenvolvimento
- Uma fase tem de estar completa antes de passar para a próxima.
- Saídas das fases são acordadas contratualmente!
- Todas as fases envolvem atividades de validação

Com o que cascata funciona bem?

- Especificações que não mudam: ônibus espacial da NASA, controle de aeronaves...
- Muitas vezes, quando o cliente vê, quer mudanças
- Muitas vezes, depois de construído a primeira vez, os desenvolvedores aprendem o caminho que eles deveriam ter seguido

Was this a successful approach for SW development?

- ~420KLOC, ≤1 bug in last 3 versions
- 260 SW engineers working for one of only 4 dev shops to receive highest “maturity rating” from **SW Engineering Institute@CMU**
- Rigorous adherence to waterfall
  - ex: 6 KLoC change = 2500 pages of docs

They Write the Right Stuff. FastCompany, 1996.
Photo: Flickr user Matthew Simantov, CC-BY-SA

Was this a successful approach for SW development?

- And the users exclaimed with a laugh and a taunt: "It's just what we asked for, but not what we want." —Anonymous

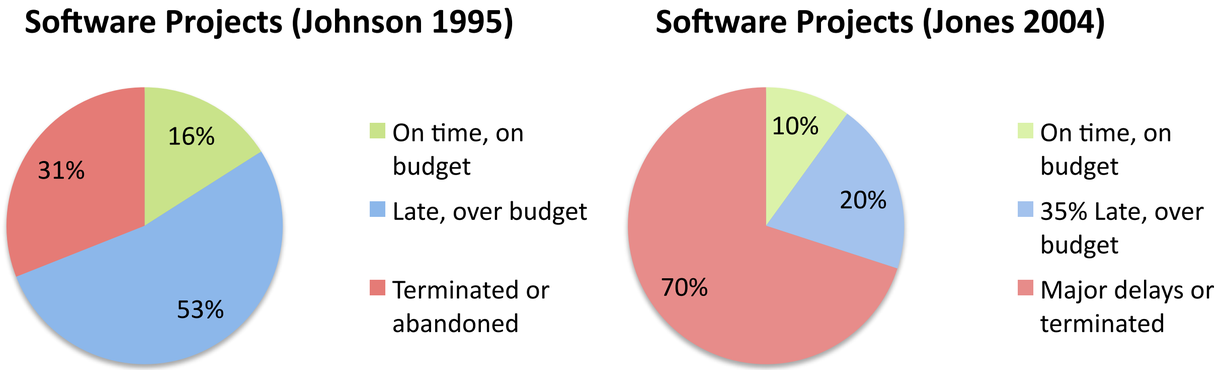
- Software's strength is its evolvability — **its ability to change and adapt** — but that's a poor fit for "big design up front" (BDUF) or "top down" approaches

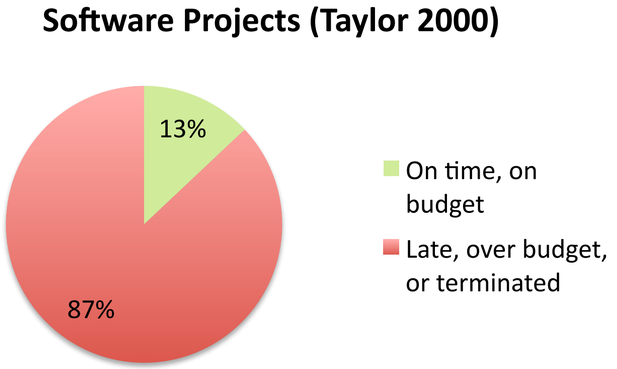
Problemas do modelo cascata

- **Particionamento** inflexível do projeto em estágios
  - Dificulta a resposta aos requisitos de mudança do cliente.
- Documentos “completamente elaborados” são necessários para fazer as transições entre estágios
- Apropriado somente quando os requisitos são bem compreendidos e quando as **mudanças são raras**
  - Poucos sistemas de negócio têm requisitos estáveis.

Quão preciso são os processos dirigido a planos?

- IEEE Spectrum “Software Wall of Shame”
- 31 projetos: desperdiçaram $17B





3/~500 novos projetos cumprem os custos e o cronograma

(Figure 1.6, Engineering Long Lasting Software by Armando Fox and David Patterson, 2nd Beta edition, 2013.)

Como lidar com mudanças?

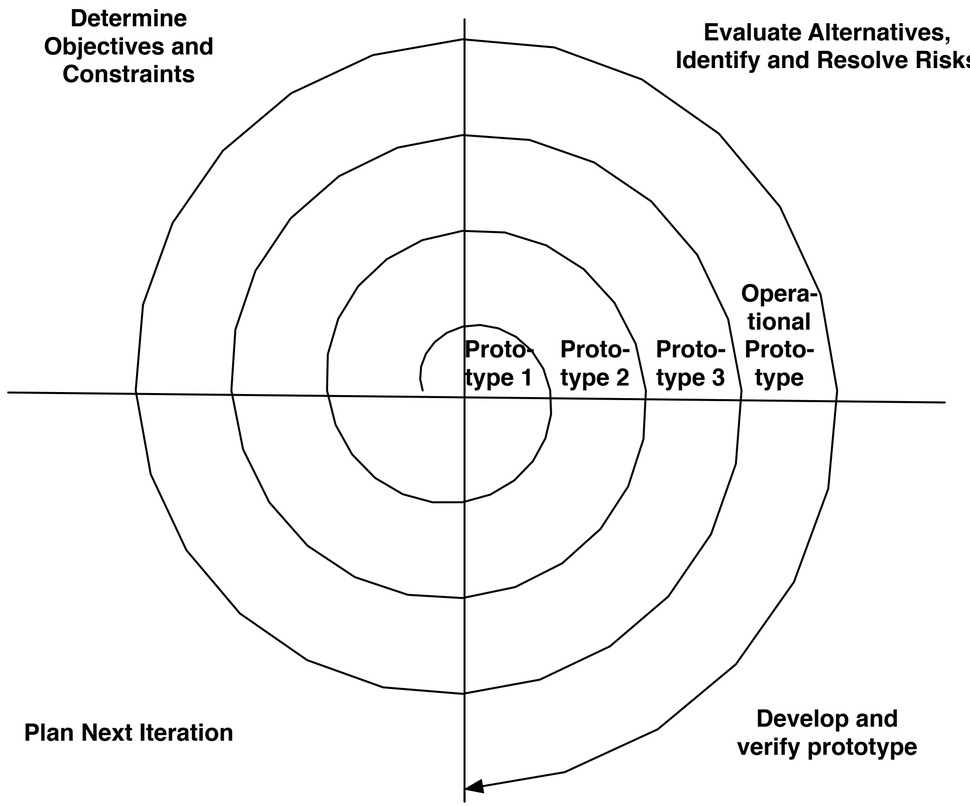
- “Plan to throw one [implementation] away; you will, anyhow.”
- Fred Brooks, Jr. (1999 Turing Award winner)

- Often after build first one, developers learn right way they should have built it

(Photo by Carola Lauber of SD&M www.sdm.de. Used by permission under CC-BY-SA-3.0.)

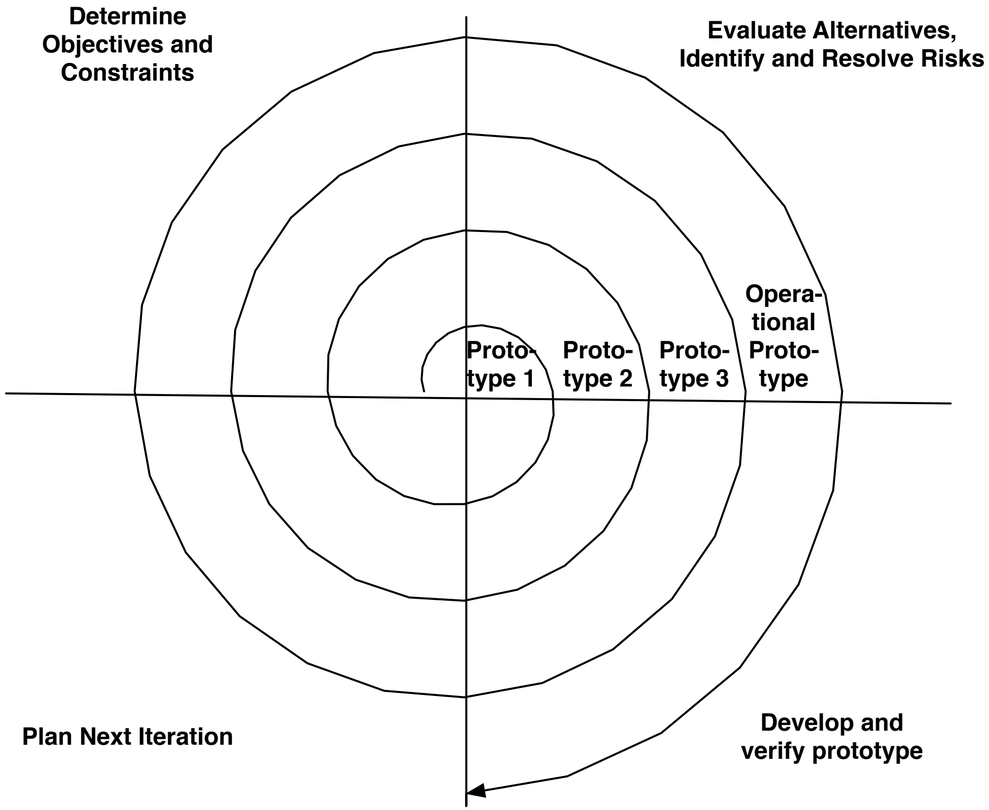
Ciclo de Vida Espiral (1986)

- Use prototypes to get customer feedback until “final” version built
  - each “iteration” delivers a new prototype
  - iterations may be far apart in time
- Later variant: Rational Unified Process
  - overlaps planning & execution stages of multiple iterations



Setores do modelo espiral

- Definição de objetivos
  - Objetivos específicos para a fase são identificados.
- Avaliação e redução de riscos
  - Riscos são avaliados e atividades são realizadas para reduzir os riscos-chave.
- Desenvolvimento e validação
  - Um modelo de desenvolvimento para o sistema, que pode ser qualquer um dos modelos genéricos, é escolhido.
- Planejamento
  - O projeto é revisado e a próxima fase da espiral é planejada.



- Processo de Desenvolvimento vs. Processo de Gerenciamento

Espiral
Good & Bad

- Iterações envolvem o cliente antes do produto estar completo
  - Reduz as chances de mal entendidos
- Gerenciamento de Riscos no ciclo de vida
- Monitoramento do projeto facilitado
- Cronograma e custo mais realista através do tempo

- Iterações longas de 6 a 24 meses
- Tempo para os clientes mudarem de ideia
- Muita documentação por iteração
- Muitas regras para seguir, pesado para todo projeto
- Custo alto do processo
- Complicado de atingir metas de investimento e marcos no cronograma

Peres's Law

"If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time."

— Shimon Peres

(vencedor do Prêmio Nobel da Paz de 1994)

(Photo Source: Michael Thaidigsmann, put in public domain,
See http://en.wikipedia.org/wiki/File:Shimon_peres_wjc_90126.jpg)

Alternative Process?

- P&D requires extensive documentation and planning
- P&D depends on an experienced manager
- Can we build software effectively without careful planning and documentation?
- How to avoid “just hacking”?

Agile Manifesto, 2001

- “We are uncovering better ways of developing SW by doing it and helping others do it. Through this work we have come to value
  - **Individuals and interactions** over processes & tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan
- That is, while there is value in the items on the right, we value the items on the left more.”

Agile lifecycle

- Embraces change as a fact of life: continuous improvement vs. phases
- Developers continuously refine working but incomplete prototype until customers happy, with customer feedback on each **Iterations** (1-2 weeks)
- All lifecycle elements in every iteration
- Agile emphasises **Test-Driven Development (TDD)** to reduce mistakes, written down **User Stories** to validate customer requirements, **Velocity** to measure progress

“Extreme Programming” (XP), uma versão do ciclo de vida ágil (Kent Beck et al.)

- If short iterations are good, make them as short as possible (weeks vs. years)
- If simplicity is good, always do the simplest thing that could possibly work
- If testing is good, test all the time. Write the test just before writing the code to be tested.
- If code reviews are good, review code continuously, by programming in pairs, taking turns looking over each others’ shoulders.

Agile Then and Now

- Controversial in 2001
  - "… yet another attempt to undermine the discipline of software engineering… nothing more than an attempt to legitimize hacker behavior."
  - Steven Ratkin, "Manifesto Elicits Cynicism," IEEE Computer, 2001
- Mainstream for last several years
  - 2012 study of 66 projects found majority using Agile, even for distributed teams

[Sommerville, 2011]

Sim: Dirigido a Plano; Não: Ágil

- É importante ter uma especificação e projeto?
- Os clientes não estão disponíveis para feedback?
- O Sistema a ser desenvolvido é grande?
- O Sistema é complexo (ex. tempo real)?
- Vai ter um ciclo de vida longo?
- Está utilizando ferramentas “ruins”?
- O time está geograficamente distribuído?
- A cultura do time é orientada a documentação?
- O Time tem um perfil “fraco” de desenvolvimento?
- O sistema está sujeito a regulamentação externa?

Pergunta

Qual afirmação é **VERDADEIRA**?

A. A grande diferença entre Ágil e DP é que Ágil não usa requisitos
B. A grande diferença entre Ágil e DP é medir o progresso de encontro a um plano
C. Você pode construir apps SaaS utilizando Ágil mas não com DP
D. A grande diferença entre Ágil e DP é a construção de protótipos e a interação com os clientes durante o processo

Pergunta

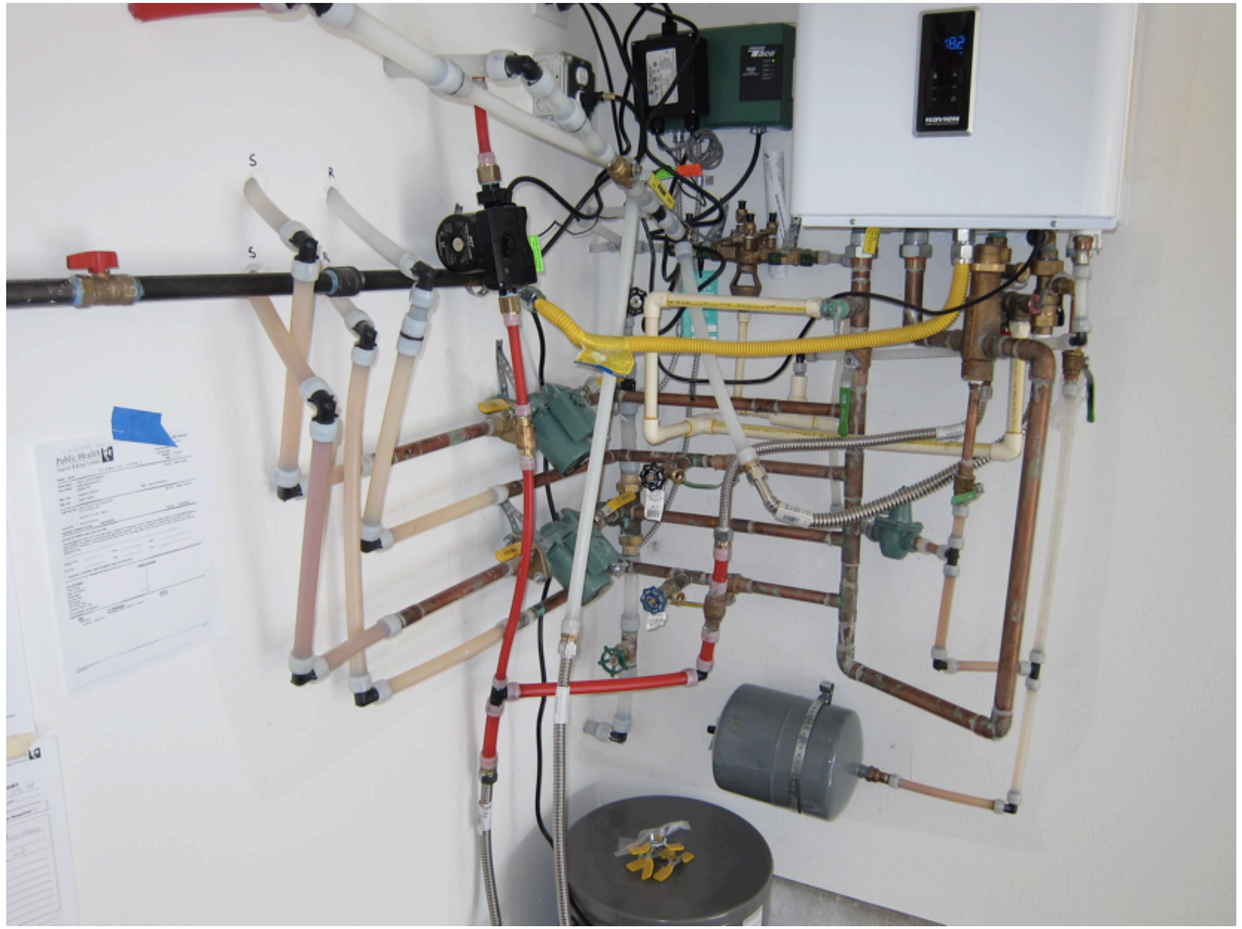
Qual afirmação é **VERDADEIRA**?

A. A grande diferença entre Ágil e DP é que Ágil não usa requisitos

B. A grande diferença entre Ágil e DP é medir o progresso de encontro a um plano

C. Você pode construir apps SaaS utilizando Ágil mas não com DP

D. A grande diferença entre Ágil e DP é a construção de protótipos e a interação com os clientes durante o processo



SW Legado vs SW Bonito

Maintenance != bug fixes

- Enhancements: 60% of maintenance costs
- Bug fixes: 17% of maintenance costs

- Hence the “60/60 rule”:
  - 60% of software cost is maintenance
  - 60% of maintenance cost is enhancements.

- Glass, R. Software Conflict. Englewood Cliffs, NJ: Yourdon Press, 1991

Legacy Code Matters

- Since maintenance consumes ~60% of software costs, it is probably the most important life cycle phase of software...
- “Old hardware becomes obsolete;
  old software goes into production every night.”
  - Robert Glass, Facts & Fallacies of Software Engineering (fact #41)
- How do we understand and safely modify legacy code?

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

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]
- **Código bonito**: atende as necessidades dos clientes e é fácil de evoluir
  - Insight: Agile techniques for embracing change also useful to manage legacy code!
  - Legacy code can be beautiful code

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

Learning New Languages and Stacks

Frameworks Often Drive Language Adoption

- cgi-bin, mod_perl, PHP, JSP, ASP.net…
- Good news: imperative OO languages more alike than different
- Great developers learn new stacks rapidly
- Three-stage plan:
  I. Learn the language (8-step plan)
  II. Understand the application architecture implied by the framework
  III. Understand mapping between language's features & framework's abstractions

Language 8-step plan

1. Types & typing
2. Primitive types/ops, control flow, strings…
3. Methods (functions, procedures)
4. Abstraction & encapsulation
5. Idioms
6. Libraries & dependency management
7. Debugging (language & framework)
8. Testing facilities

Pitfalls

- “You can write FORTRAN in any language”

- A koan from the venerable MIT AI Lab…
  - Don’t blindly copy-paste code you don’t understand. You don’t know where it’s been.

Advice: learn from the Internet

- Google the nouns and verbs, to see examples of what's idiomatic, before coding it yourself
  - "js hash list keys", "js array search element"
  - What are the class names or libraries to lookup?
  - Not unusual: research time >> coding time
- Vet answer: Lots of up-votes? Reputable site?
- Read real docs at ruby-doc.org to learn more
  - Node core library & "standard library"
- Don't copy-paste code you don't understand
  - To know if understand, try to break it and see if it breaks as predicted
  - Use debug to stop a program and inspect state

Agenda de Iteração

1ª Semana

Reunião com Cliente	Planejamento da Iteração	Produção
Reunião de retrospectiva		

2ª Semana

Produção	Auto-avaliação
Reuniões diárias (standup meeting)	

Atividades Importantes

Reunião com cliente

- Coletar feedback
- Coletar novos recursos
- Compreender os valores de negócios

Planejamento da Iteração

- Compreender as histórias do usuário
- Prioritizar
- Estimativa
- Selecione a lista de pendências de uma iteração

Auto-avaliação

- Práticas
- Trabalho em equipe

Produção

- Alta qualidade
- Entrega rápida
- Integração Contínua
- Use as ferramentas

Reunião diária

Reunião de retrospectiva

- Reflita sobre a iteração passada
- Resolva problemas da equipe
- Melhore a produtividade da equipe

In conclusion…

- Unlike HW, SW can evolve to match customer needs over time
- …if development process embraces change
- Agile is one way to embrace change, evolved from Big Design Up Front (BDUF) Plan&Doc models
- Natural extension to managing legacy code: it's just change over longer time scales
- Learning to learn languages >> knowing language: plan to spend as much or more time researching examples as writing the code yourself
- For Thurs: start learning our Javascript stack this way! Expect to read a lot, but also expect must learn by doing!