
Engenharia de Software e Sistemas (BCC35E)



Igor Scaliante Wiese



@IgorWiese



igor.wiese@gmail.com | igor@utfpr.edu.br |
igorwiese.com

Igor Scaliante Wiese, PhD.

Software Engineering Professor and Researcher at UTFPR - Brazil; Husband! Father! Corinthians fan! Chargers fan! Beer lover!

Home Publications Research Students Service Teaching



Igor Scaliante Wiese, PhD.

igor@utfpr.edu.br

I completed a PhD in Computer Science at the **Institute of Mathematics and Statistics** at **University of São Paulo** in 2016. Currently, I am a Assistant Professor at the **Federal University of Technology - Paraná (UTFPR)**, Brazil, working in the Academic Department of Computing (DACOM), where I am developing my research in the **Software Engineering and Collaborative Systems Research Lab**. I was a visiting scholar at the **University of California, Irvine** from July 2013 to January 2014. I am also a member of the **Software Engineering and Collaborative Systems Research Group** from **University of São Paulo**, and the Research Group on Information Systems from **State University of Maringá (UEM)**. Currently, I am at Northern Arizona University doing a post-doctoral.

I research the intersections of **Software Engineering (SE)** and **Computer Supported Cooperative Work (CSCW)**. Currently, I am working on help developers to find artifacts to perform a task correctly, using recommendation systems based on socio-technical aspects of software development. I am interested in Mining Software Repositories techniques, Recommendation Systems, Open Source Software, Human Aspects of Software Engineering and Empirical Software Engineering.



NAU
NORTHERN
ARIZONA
UNIVERSITY



Christoph Treude
University of Adelaide



Anita Sarma
Oregon State University



Igor Steinmacher - NAU



Marco Gerosa
Northern Arizona University



Alexander Serebrenik
Eindhoven University of Technology



Gustavo Pinto
Universidade Federal do Pará | Zup

Licença do material

Este Trabalho foi licenciado com uma Licença



**Atribuição-NãoComercial-
Compartilhagual 4.0 Internacional
(CC BY-NC-SA 4.0)**

Mais informações visite

https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt_BR

ENGENHARIA DE SOFTWARE MODERNA

Princípios e práticas para
desenvolvimento de software
com produtividade



MARCO TULIO VALENTE



Vinicius Garcia (He/Him) · 1º

Associate Professor at Centro de Informática (UFPE), Associate Research Fellow at SoftexRecife, Timbaleiro e Filho de Gandhi

Fala sobre #data, #vscode e #freecodecamp

Recife, Pernambuco, Brasil · [Informações de contato](#)



Universidade Federal de
Pernambuco



Fast MBA



Links uteis

The screenshot shows a GitHub repository page. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation is a search bar and a user profile icon. The main content area shows the repository details: 'igorwiese / Engenharia-de-Software-BCC35E' (Public). It lists several commits from 'igorwiese' with messages like 'Delete x.md', 'Update equipes.md', 'Create CODE_OF_CONDUCT.md', and 'Initial commit'. A 'Code' dropdown menu is open, showing options like 'Go to file', 'Add file', and 'Code'. To the right, there's an 'About' section with a description of the repository as belonging to the 'Engenharia de Software (BCC35E)' course at UFTM. Below the description are buttons for 'course' and 'softw'. Further down are links for 'Readme', 'MIT license', 'Code of conduct', '1 star', '0 watching', and '0 forks'.

<https://github.com/igorwiese/Engenharia-de-Software-BCC35E>

<https://discord.gg/GAjWC5dz>

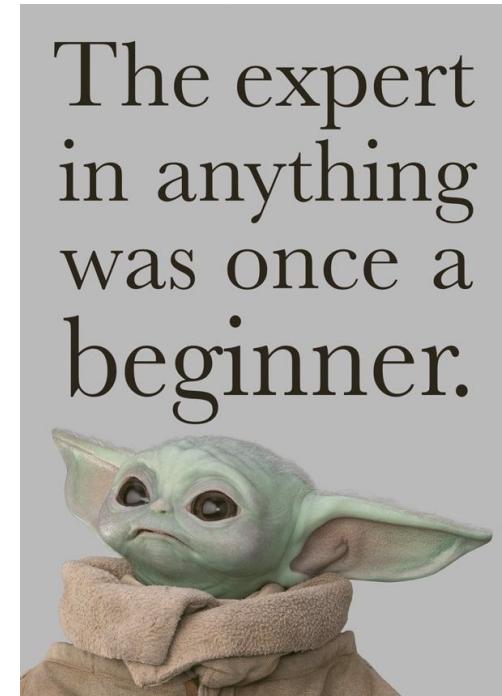
<https://app.strateegia.digital/dashboard/public-link/4qYOTW>

Objetivos

O objetivo principal deste curso é

- estudar
- analisar
- discutir, e
- aplicar os fundamentos de Engenharia de Software.

Do ponto de vista prático, os conceitos estudados serão aplicados no desenvolvimento
de um projeto de um sistema de informação simples.



Nossos princípios

- Conhecimento como “obra aberta” (em vez de “mensagem fechada”);
- Curadoria de conteúdos, sínteses e roteiros de trabalho (em vez da produção de conteúdos próprios para EAD, por exemplo);
- Ambiências computacionais diversas (em vez de se restringir aos serviços do Ambiente de Aprendizagem);
- Aprendizagem em colaborativa (em vez de aprendizagem isolada);
- Divergência, convergência e conversação entre todos, em interatividade (em vez de apresentação de conteúdos);
- Atividades autorais inspiradas nas práticas da cibercultura (em vez de “estudo dirigido”);
- Mediação online para colaboração (em vez de “tutoria reativa”); e,
- Jornada formativa e colaborativa, baseada em competências (em vez de apenas exames presenciais).

Ambiência computacional

- Trabalharemos com missões
- Relacionadas aos nossos tópicos de aulas
- Publicados no repositório da disciplina:
 - Engenharia de Software: <https://github.com/igorwiese/Engenharia-de-Software-BCC35E>
- Utilizamos um servidor do **Discord** para comunicação **assíncrona** e **ambientes de reunião** (sala de audio e video) para **encontros fortuitos**
 - <https://discord.gg/GAjWC5dz>
- **Mural** para dinâmicas de trabalho síncrono (Github discussions também!)
- E para as missões utilizados a plataforma **Strateegia** (<https://strateegia.digital/>)
 - <https://app.strateegia.digital/dashboard/public-link/4qYOTW>

Missão

Para cada missão os alunos devem [individualmente ou em grupo, isso vai se ajustando ao longo da jornada - inicialmente individual] produzirem algo [um texto, um sistema, uma imagem, um infográfico, um vídeo, etc.] que vai ser definido em função de cada missão.

Metodologia



Avaliação

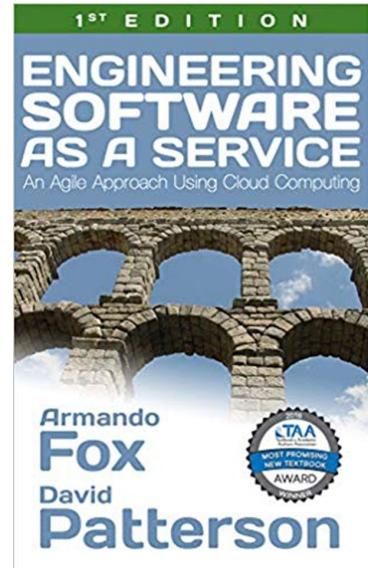
- Os aspectos **teóricos** serão avaliados por meio das **missões** com caráter de avaliação **individual**, mas vale ressaltar que as missões possuem uma natureza de execução prática
 - Pontos de Divergência, Convergência e Hackenges
- A consolidação dos aspectos teóricos discutidos ao longo da disciplina serão avaliados por meio de um **exercício prático e em equipe** que consistirá no desenvolvimento de um projeto de sistema de informação simples
 - $\text{Media} = (3 * \text{NotaMissoes} + 7 * \text{NotaProjeto}) / 10$, onde
 - NotaMissoes é a nota individual do estudante referente às missões;
 - NotaProjeto é a nota do projeto será calculada com base nos seguintes aspectos:
 - avaliação do desempenho nas iterações; escopo do cenário; identificação das pessoas; concepção do Plano de MVP; especificação da Proposta de Solução; projeto da Arquitetura; especificação e implementação de Testes; Implantação da solução; Relato de Lições Aprendidas & Decisões de Projeto.
- A não participação de uma destas atividades, reprova automaticamente.

Plano de Trabalho 2022.2

- Apresentação da Disciplina + Introdução a Engenharia de Software
 - Processos de Software
 - Engenharia de Requisitos
 - Teste A/B
 - Pensamento de Design
 - Princípios de Projeto
 - Padrões de Projeto
 - Arquitetura de Software
 - Testes de Software
 - DevOps
-
- Git, Linear, Node, Vuejs, NuxtJs,, Heroku...
 - CBL bigIdea [definir o tema central do projeto]
 - CBL essentialQuestion [definir a questão essencial do projeto]
 - CBL essentialQuestion [definir a questão essencial do projeto]
 - CBL challenge [definir o desafio do projeto]
 - CBL guidingQuestions [definir questões norteadoras do projeto]
 - CBL guidingActivities [definir atividades e recursos norteadores do projeto]
 - CBL analysis [analisar as lições aprendidas a partir das atividades norteadoras]
 - CBL evaluation [avaliar os aprendizados resultantes do projeto]

Referências

- Marco Túlio Valente. Engenharia de Software Moderna
 - <https://engsoftmoderna.info/>
- 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/>



Mãos à obra!

<https://bit.ly/if977-22-1>

Apertem os cintos!



Quem é?

github.com/chrislgarry/Apollo-11



Eu ouvi ela no ICSE 2018 na Suécia... <https://www.youtube.com/watch?v=ZbVOFoUk5lU>

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

O que faz um ótimo engenheiro de software?

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

Sempre melhorando

Apaixonado

Tecnicamente de mente aberta

Orientado por dados

Características Pessoais

Tomada de Decisão

Conhece pessoas e organização

Atualiza **modelos mentais** quando aprende novas habilidades/fatos/contexto

Considera a situação em vários níveis ao fazer julgamentos

Pode raciocinar sobre idéias complexas e entrelaçadas

Ajuda os outros a aprender, adaptando explicações a eles

Cria **sucesso compartilhado**, possivelmente por meio de compromissos pessoais

Cria "**porto seguro**" onde outras pessoas podem aprender com os erros
Dá feedback honesto

Habilidades de trabalhar em Time

Habilidades Técnicas

Soluções elegantes

Pensamento criativo diante das limitações das soluções atuais

Antecipa **necessidades técnicas** com base em experiências anteriores

**Nosso objetivo é que você
aprenda engenharia de software
ágil, disciplinada e focada no
cliente, de sistemas de
informação como serviço (EISaaS)**

Sobre o Projeto



- Nota para Equipe! Equipe que vai dividir os pontos entre os membros.
- Avaliação por interação, por artefatos e por execução.
- Temática: LIVRE

Configuração de expectativas

- Suposição 1: você sabe codificar
- Suposição 2: ter conta no GitHub (Student)
- Objetivo: prepará-lo para atuar como membro da comunidade profissional de engenharia de software
 - Esta turma me ensinará JS/React? Ruby/Rails/Go/Flutter/Vuejs?
 - Vou aprender Ruby/Rails/JavaScript/React/Go/Flutter/Vuejs?
- Nosso objetivo: **ensinar vocês a aprendê-los**
 - Dica 1: vídeos, leitura etc. não são suficientes
 - Dica 2: mas eles são necessários

Como ter uma experiência ruim neste curso

- I. Pular/faltar aula/seção (física ou mentalmente)
- II. Copiar/não fazer o próprio trabalho dos HWs e do projeto
- III. Trabalhe sozinho, não envolva colegas
- IV. Resistir a novas ferramentas/técnicas
- V. Ignore o processo, codifique da forma que quiser
- VI. Trabalhar por para pontos x aprendizagem
- VII. Suponha leitura/palestra/vídeos == aprendizagem
- VIII. Fraude (violar código de honra)
- IX. Multitarefa (email, mensagens instantâneas, etc.) durante a aula/codificação

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

O que se estuda em ES?

1. Engenharia de Requisitos
2. Projeto de Software
3. Construção de Software
4. Testes de Software
5. Manutenção de Software
6. Gerência de Configuração
7. Gerência de Projetos
8. Processos de Software
9. Modelos de Software
10. Qualidade de Software
11. Prática Profissional
12. Aspectos Econômicos



Eu ouvi ele no ICSE 2018 na Suécia -

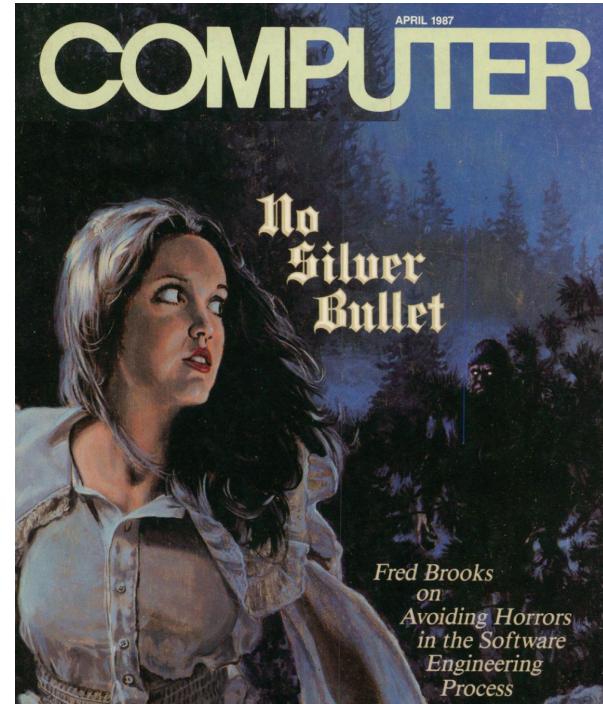
<https://www.youtube.com/watch?v=StN49regNq8>

E também cuidado: Não Existe Bala de Prata

Frederick Brooks. No Silver Bullet -
Essence and Accidents of Software
Engineering. IEEE Computer, 1987.

Imagen de:

https://twitter.com/zeljko_obren/status/909014656802574336



No Silver Bullet

Essence and Accidents of Software Engineering

Frederick P. Brooks, Jr.

University of North Carolina at Chapel Hill

Fashioning complex conceptual constructs is the essence; accidental tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.

Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, one seeks bullets of silver that can magically lay them to rest.

The familiar software project, at least as seen by the nontechnical manager, has something of this character; it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet—something to make software costs drop as rapidly as computer hardware costs do.

But, as we look to the horizon of a decade hence, we see no silver bullet.

There is no single development, in either technology or in management technique, that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity. In this article, I shall try to show why, by examining both the nature of the software problem and the throughs—and indeed, I believe such to be inconsistent with the nature of software—many encouraging innovations are under way. A disciplined, consistent effort to develop, propagate, and exploit these innovations should indeed yield an order-of-magnitude improvement. There is no royal road, but there is a path.

The first step toward the management of disease was replacement of demon theories and humours theories by the germ theory. That very step, the beginning of hope, in itself dashed all hopes of magical solutions. It told workers that progress would be made stepwise, at great effort, and that a persistent, unremitting care would have to be paid to a discipline of cleanliness. So it is with software engineering today.

Does it have to be hard?—Essential difficulties

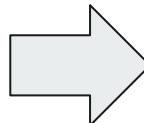
Motivo: Dificuldades Essenciais

Complexidade - construir um satélite?

Conformidade - leis mudam - imposto,
leis físicas não mudam

Facilidade de Mudanças - demanda por
mudança contínua

Invisibilidade - natureza abstrata, é
difícil visualizar o tamanho e
consequentemente estimar o esforço



**Tornam Engenharia
de Software diferente
de outras engenharias**

Falha Famosa: Explosão do Ariane 5 (1996)



30 segundos depois



Custo do foguete e satélite:
US\$ 500 milhões

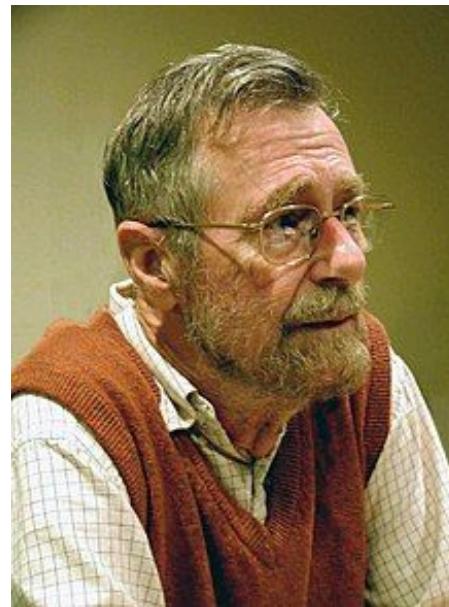
Relatório do Comitê de Investigação

- Explosão foi causada por uma falha de software
- Conversão de um real de 64 bits para um inteiro de 16 bits
- Como o real não "cabia" em 16 bits, a conversão falhou
- Recomendação: rever todo software de bordo!

Frase famosa

Testes de software
mostram a presença de
bugs, mas não a sua
ausência.

-- Edsger W. Dijkstra



Outra observação importante



Allen Holub
@allenholub

...

All software is tested. The real question is whether the tester is you or your users.

12:49 PM · May 28, 2020 · TweetDeck

Verificação vs Validação

- **Verificação:** estamos implementando o sistema corretamente?
 - De acordo com os requisitos e especificações
- **Validação:** estamos implementando o sistema correto?
 - O sistema que os clientes querem
 - Testes de aceitação com os usuários

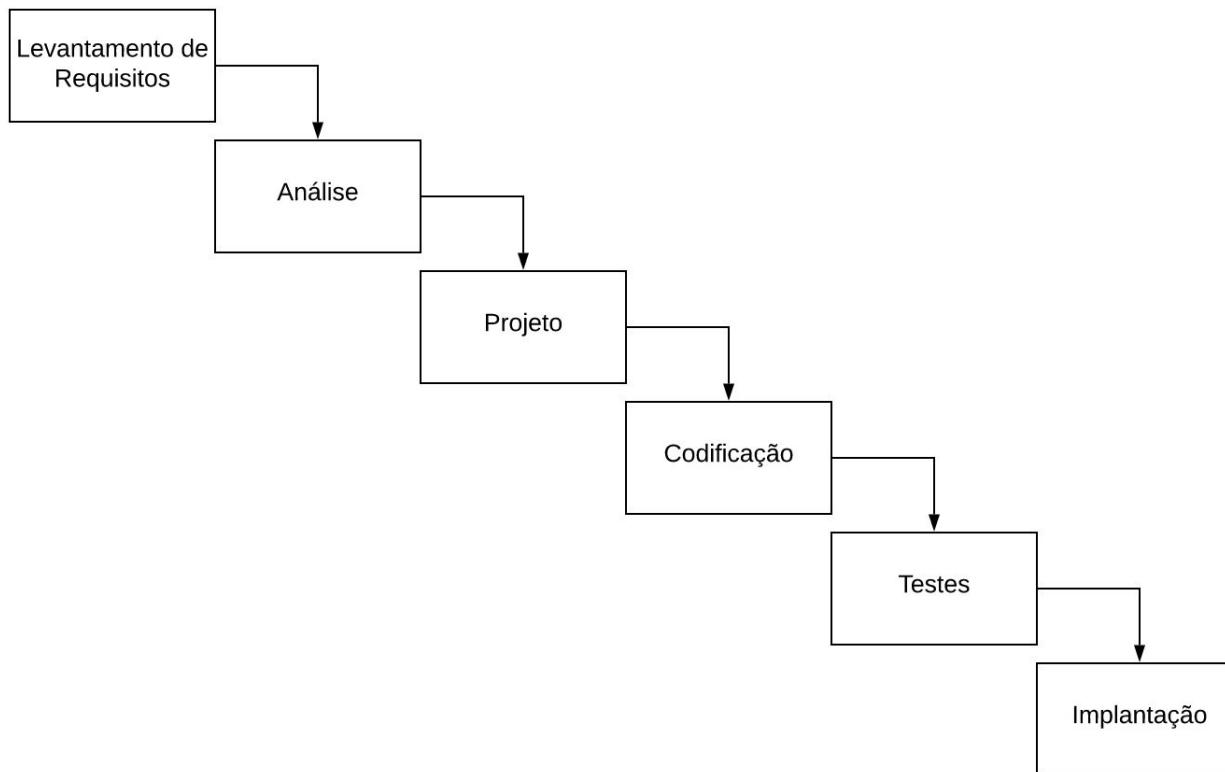
Processos de Desenvolvimento de Software

- Um processo de software define quais atividades devem ser seguidas para construir um sistema de software
- Dois principais modelos:
 - Waterfall ("cascata")
 - Ágil (ou incremental ou iterativo)

Modelo em Cascata

- Inspirado em processos usados em engenharias tradicionais, como Civil, Mecânica, Elétrica, etc
- Proposto na década de 70 e muito usado até ~1990
- Adotado pelo Depto de Defesa Norte-Americano (1985)

Modelo em Cascata



Problemas com Modelo Waterfall

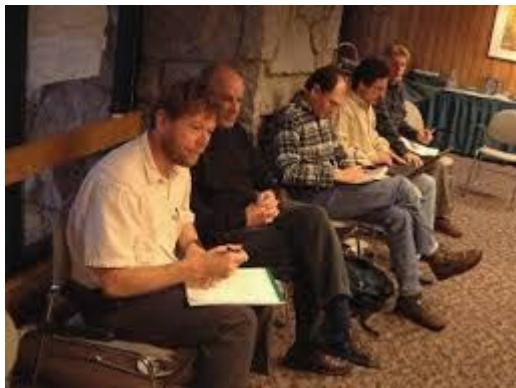
- Requisitos mudam com frequência
 - Levantamento completo de requisitos demanda tempo
 - Quando ficar pronto, o "mundo já mudou"
 - Clientes às vezes não sabem o que querem
- Documentações de software são verbosas
- E rapidamente se tornam obsoletas

Manifesto Ágil (2001)

Encontro de 17 engenheiros de software em Utah

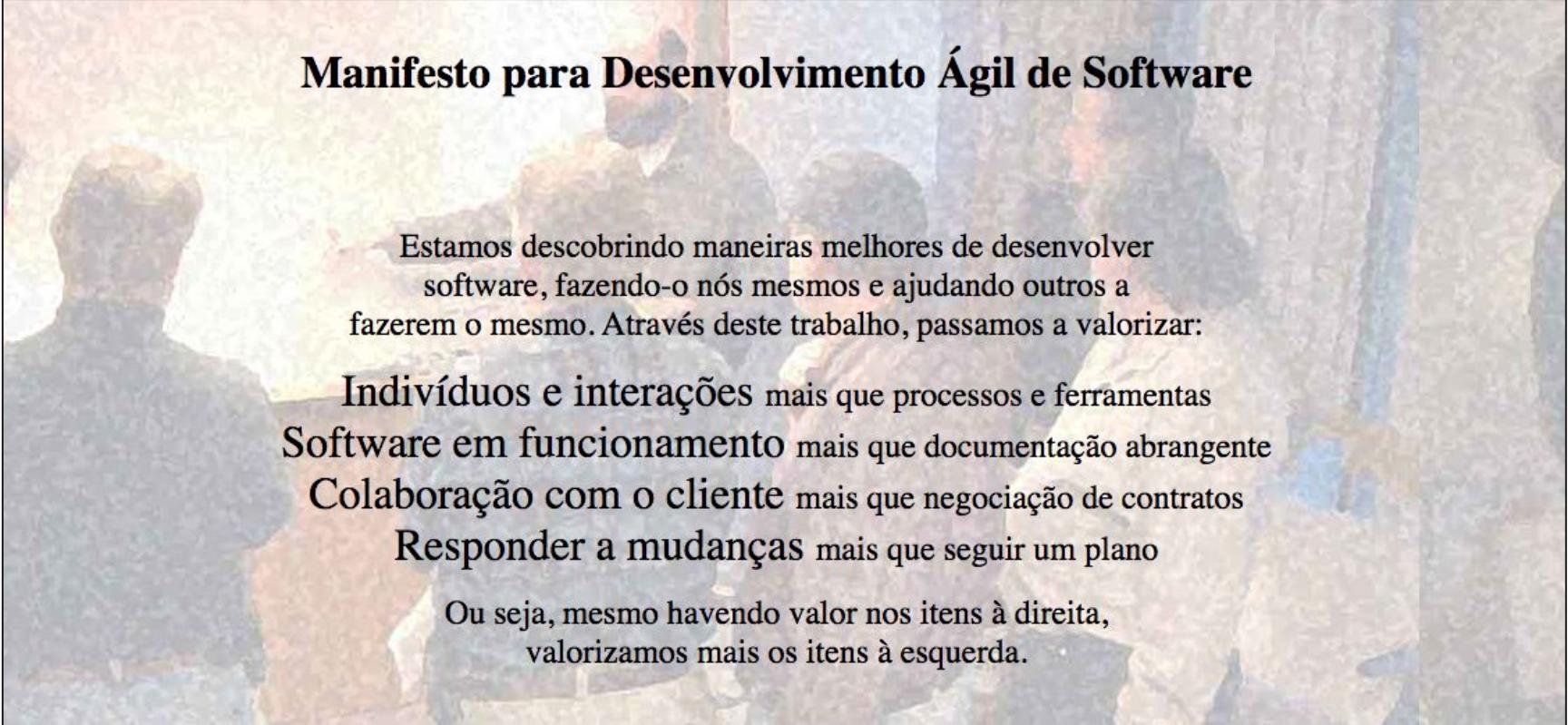
Crítica a modelos sequenciais e pesados

Novo modelo: incremental e iterativo



<https://siamchamnankit.co.th/history-some-pictures-and-pdfs-of-the-agile-manifesto-meeting-on-2001-a33c40bcc2b>

Manifesto para Desenvolvimento Ágil de Software



Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

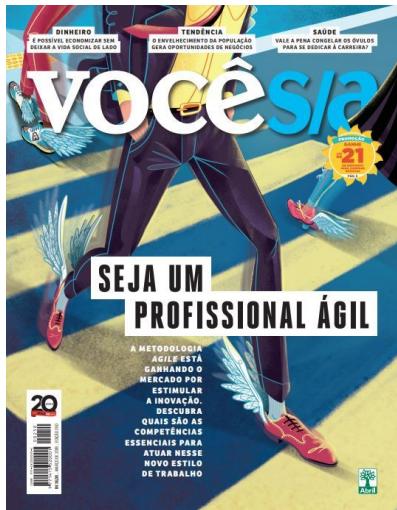
Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Desenvolvimento Ágil

Profundo impacto na indústria de software

Hoje, tudo é ágil... Talvez adjetivo até desgastado



Março 2019



Maio 2020

Exemplo: Waterfall vs Agile (apenas para ilustrar)

- Problema: construir uma ponte
- Solução Waterfall (*plan-and-document*):
 - Projeto preliminar e requisitos (largura etc)
 - Maquete e projeto de engenharia, estrutural, etc
 - Simulação em um túnel de vento
 - Construção da ponte
 - Entrega e inauguração

Exemplo: Waterfall vs Agile (cont.)

- Solução ágil (incremental, iterativo):
 - Constrói-se uma primeira versão, com uma única pista
 - Em seguida, constrói-se uma segunda pista
 - Depois, duplicam-se as duas pistas, etc
- Para projetos com grau de "incerteza", método ágil faz mais sentido

From Amazon to Zoom: What Happens in an Internet Minute In 2021?

<https://bit.ly/3ITh0II>



Para finalizar: Tipos ABC de sistemas

- Classificação proposta por Bertrand Meyer
- Três tipos de software:
 - Sistemas C (Casuais)
 - Sistemas B (Business)
 - Sistemas A (Acute)

Sistemas C (Casuais)

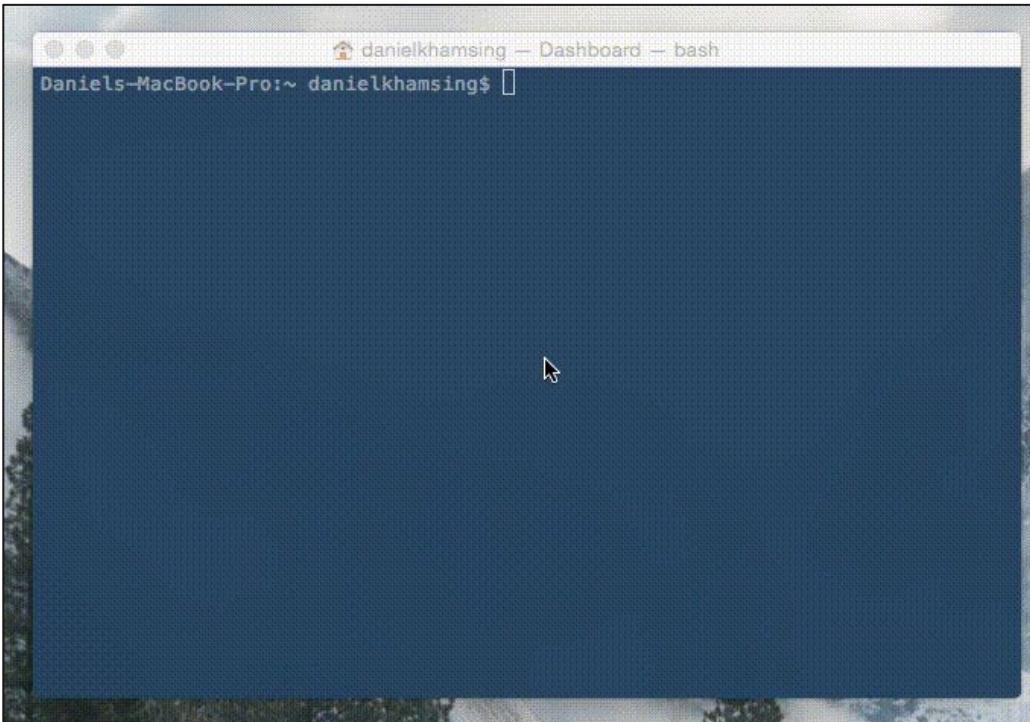
- Tipo muito comum de sistema
- Sistemas pequenos, sem muita importância
- Podem ter bugs; às vezes, são descartáveis
- Desenvolvidos por 1-2 engenheiros
- **Não se beneficiam dos princípios e práticas deste curso**
- Risco: "over-engineering"

Exemplos de Sistemas Casuais

SL(1): Cure your bad habit of mistyping

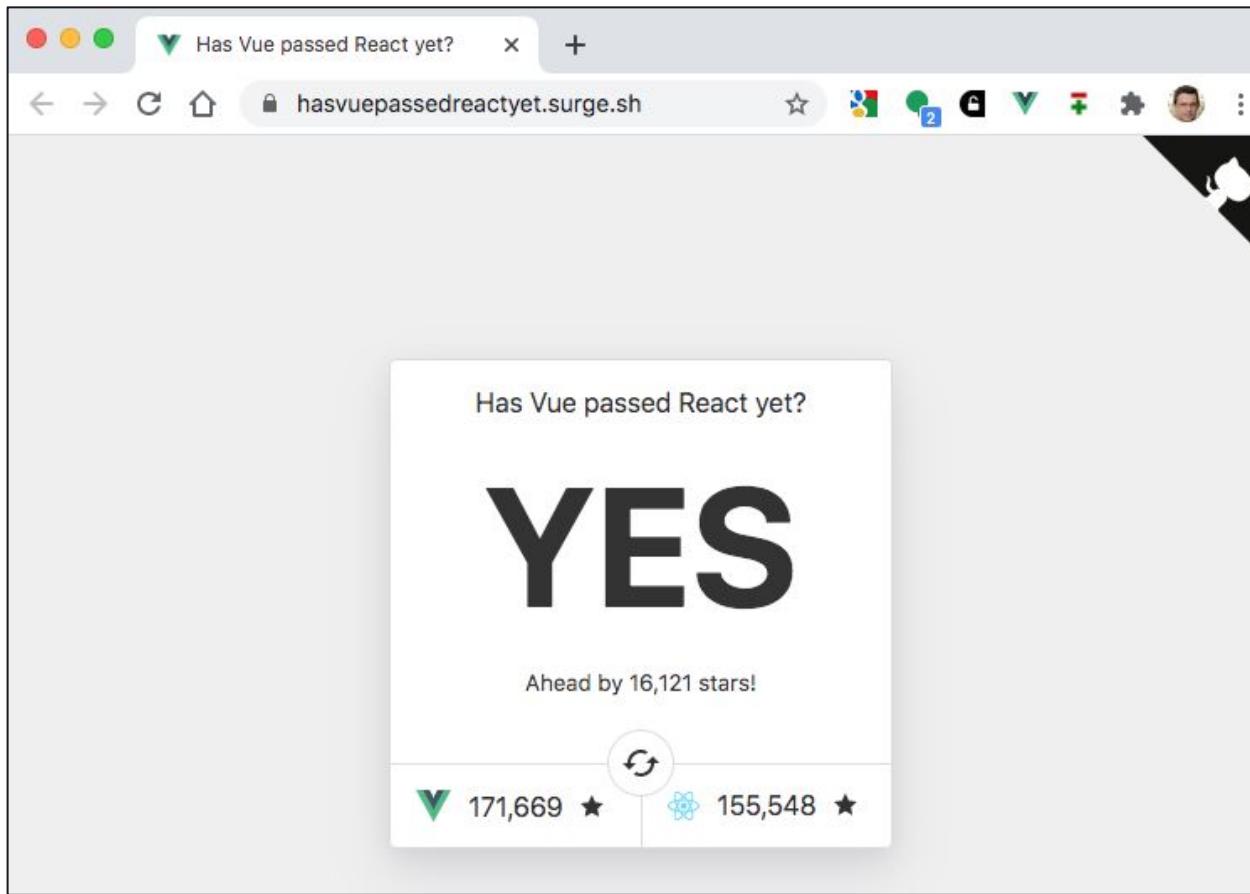
SL (Steam Locomotive) runs across your terminal when you type "sl" as you meant to type "ls". It's just a joke command, and not useful at all.

Copyright 1993,1998,2014 Toyoda Masashi (mtoyoda@acm.org)



<https://github.com/mtoyoda/sl>

<https://hasvuepassedreactyet.surge.sh>



Sistemas B (Business)

- Sistemas importantes para uma organização
- **Sistemas beneficiam-se do que veremos no curso**
- Risco: não usarem técnicas de ES e se tornarem um passivo, em vez de um ativo para as organizações

Sistemas A (Acute)

- Sistemas onde nada pode dar errado, pois o custo é imenso, em termos de vidas humanas e/ou \$\$\$
- Também chamados sistemas de missão crítica



Metrô



Aviação



Medicina

Sistemas A (Acute)

- Requerem certificações
- **Fora do escopo do
nosso curso**

Document Title

DO-178C - Software Considerations in Airborne Systems and Equipment Certification

Description

This document provides recommendations for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. Compliance with the objectives of DO-178C is the primary means of obtaining approval of software used in civil aviation products.

Document Number

DO-178C

Format

Hard Copy

Committee

SC-205

Issue Date

12/13/2011