

[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-
Compartilhagual 3.0 Não Adaptada.**

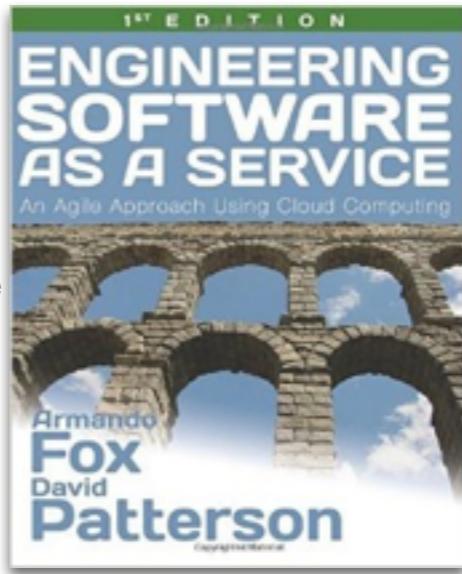
Mais informações visite

<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/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/>



Outline

- (ESaaS 1.2,10.7) SW Development Processes: Plan & Document
- (ESaaS 1.3) SW Development Processes: Agile Manifesto
- (ESaaS 1.12) Fallacies and Pitfalls
- (ESaaS 10.1) It Takes a Team: Two-Pizza and Scrum



© Photo by Ken Geiger/National Geographic Magazine

Waterfall vs. Spiral vs. RUP

Plan-and-Document processes

Como evitar o descrédito do SW?

- Não podemos tornar a construção de software tão previsível em termos de cronograma, custo e qualidade como a construção de uma ponte ou um edifício?
- Se sim, que tipo de processo de desenvolvimento devemos utilizar para torná-lo previsível a este ponto?

Engenharia de Software

- Trazer a disciplina **engenharia** para o SW
 - Termo cunhado ~20 anos após o 1º computador
 - Encontrar métodos de desenvolvimento de SW tão previsíveis em qualidade, custo e tempo assim como engenharia civil
- “Plan-and-Document” (Dirigido a Plano)
 - Antes de codificar, o gerente de projeto constrói o plano
 - Escrever uma documentação detalhada de todas as fases do plano
 - Progresso medido/monitorado em relação ao plano
 - Mudanças no projeto devem ser refletidas na documentação e possivelmente no plano

1º Processo de Desenvolvimento: Waterfall (1970)

- 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

8

Like falling down a waterfall

Lots of planning,

6-12 months or more per step

Often steps done by different people

Throw over the wall

Lots of documentation in case people leave

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

9



Com o que cascata funciona bem?

“Plan to throw one [implementation] away; you will, anyhow.”

Fred Brooks, Jr.

(recebeu em 1999 o Prêmio Turing por suas contribuições a arquitetura dos computadores, sistemas operacionais e engenharia de software)



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

10

May apply to your project
Downside to waterfall?

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.

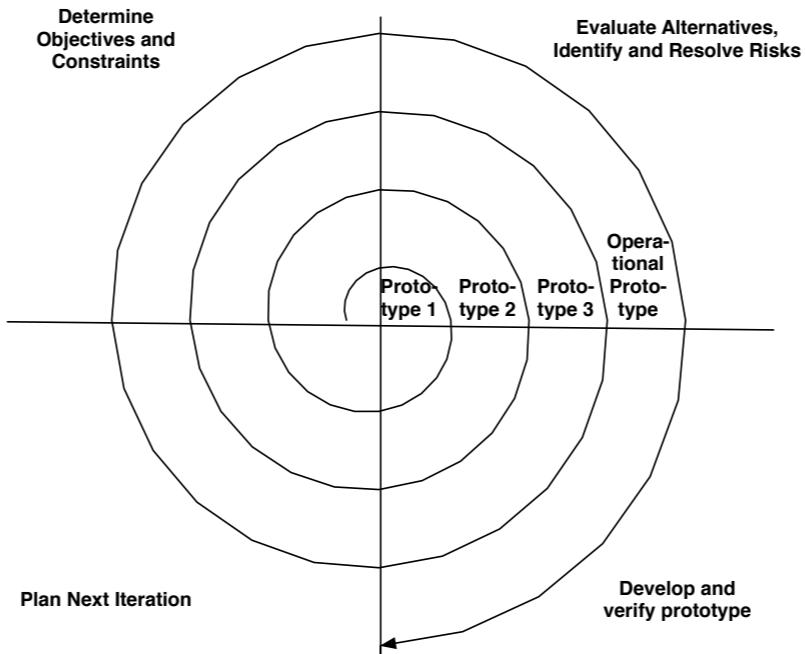
Ciclo de Vida Espiral (1986)

- Combina **Big Design Up Front** com protótipos
- Ao invés de documentar **todos os requisitos no início**, vai desenvolvendo o documento de requisitos através de **iterações** de protótipos à medida que eles são **necessários** e evoluem com o projeto

12



Ciclo de vida Espiral



(Figure 1.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, 2nd Beta edition, 2013.)

13

O processo é representado como uma espiral ao invés de uma sequência de atividades com realimentação.

Cada loop na espiral representa uma fase no processo.

Sem fases definidas, tais como especificação ou projeto – os loops na espiral são escolhidos dependendo do que é requisitado.

Os riscos são explicitamente avaliados e resolvidos ao longo do processo.

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**

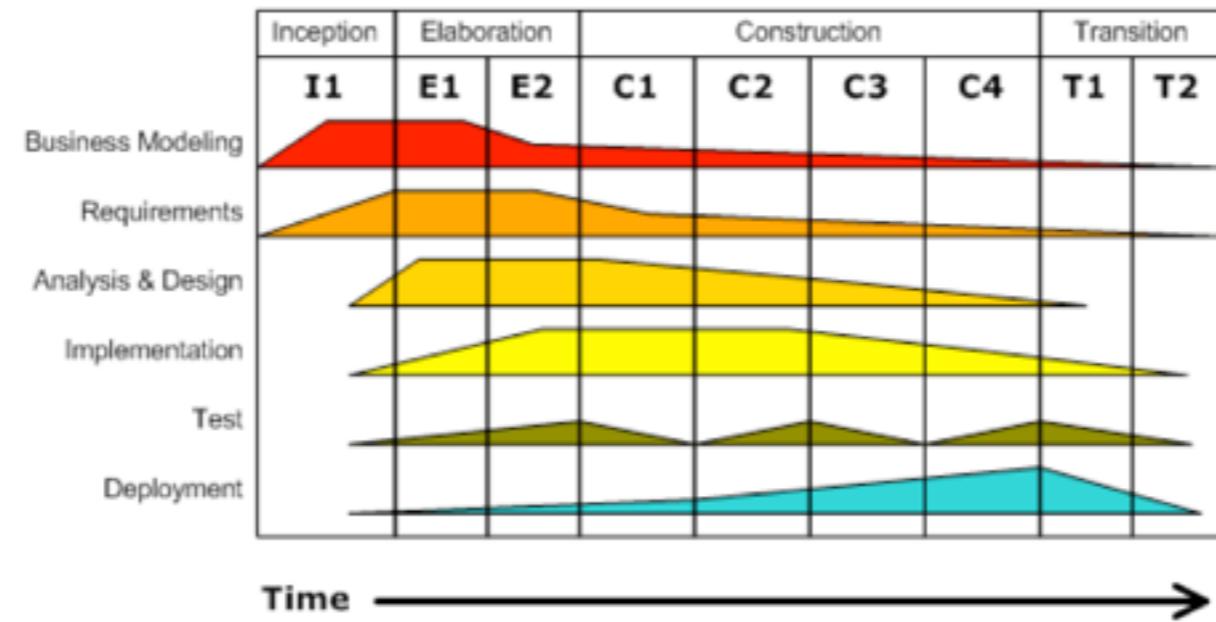


Espiral Good & Bad



-
- 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

Ciclo de Vida do Rational Unified Process (RUP) (2003)



(Figure 1.5, *Engineering Long Lasting Software* by Armando Fox and David Patterson, 2nd Beta edition, 2013.)

16

Fases do RUP

- 4 Fases, centrado no gerenciamento de projetos

1. Concepção

- Estabelecer o business case para o sistema.

2. Elaboração

- Desenvolver um entendimento do domínio do problema, arquitetura do sistema e identificar riscos

3. Construção

- Projeto, programação, teste de sistema e documentação

4. Transição

- Implantar o sistema no seu ambiente operacional.

Disciplinas de Engenharia do RUP

- 6 disciplinas das pessoas através do ciclo de vida do projeto
 1. Business Modelling
 2. Requirements
 3. Analysis and Design
 4. Implementation
 5. Test
 6. Deployment

Boas práticas do RUP

- Desenvolver o software iterativamente
- Gerenciar requisitos
- Usar arquiteturas baseadas em componentes
- Modelar o software visualmente
- Verificar a qualidade de software
- Controlar as mudanças do software



RUP Good & Bad



- Práticas de negócios vinculados a processo de desenvolvimento
- Lotes de ferramentas da Rational (agora IBM)
- Ferramentas de apoio a melhoria gradual do projeto
- Ferramentas são caras (!open source)
- Muitas opções para adaptar o RUP para uma empresa
- Apenas bom para projetos de média a larga escala

Gerência de Projetos Dirigida a Plano (DP)

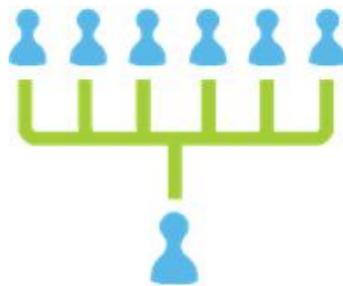
- DP depende do **Gerente de Projeto**
 - Escrever contrato para ganhar/vender o projeto
 - Recrutar equipe de desenvolvimento
 - Avaliar o desempenho dos engenheiros de software, que estabelece o salário
 - Estimar os custos, manter o cronograma, avaliar os riscos e superá-los
 - Documentar o plano de gerenciamento de projetos
 - Ganhar o crédito pelo sucesso ou culpa se os projetos estão atrasados ou acima do orçamento

Tamanho do Time de DP

“Adding manpower to a late software project makes it later.”

Fred Brooks, Jr., The Mythical Man-Month

- Leva tempo para os novos membros do time aprenderem sobre o projeto
- A comunicação do time aumenta, deixando menos tempo para trabalhar efetivamente
- Para projetos grandes, o Ideal é ter pequenos grupos (4-9 pessoas) hierarquicamente compostos



Pergunta

O que **NÃO** é uma diferença chave entre os ciclos de vida Cascata, Espiral e Ágil?

- A. Cascata utiliza fases longas, Ágil utiliza iterações rápidas e Espiral iterações longas
- B. Cascata não possui código funcional antes do fim, Espiral e Ágil tem código funcional em cada iteração
- C. Cascata e Espiral utilizam requisitos formalmente escritos porém Ágil não utiliza nada escrito
- D. Cascata e Espiral possuem fases de projeto arquitetural porém você não pode incorporar a arquitetura de SW no ciclo de vida Ágil

23

Waterfall uses written requirements, but Agile does not use anything written down is the answer to question

Beck: Part of coping with change is making change at architectural level -

Pergunta

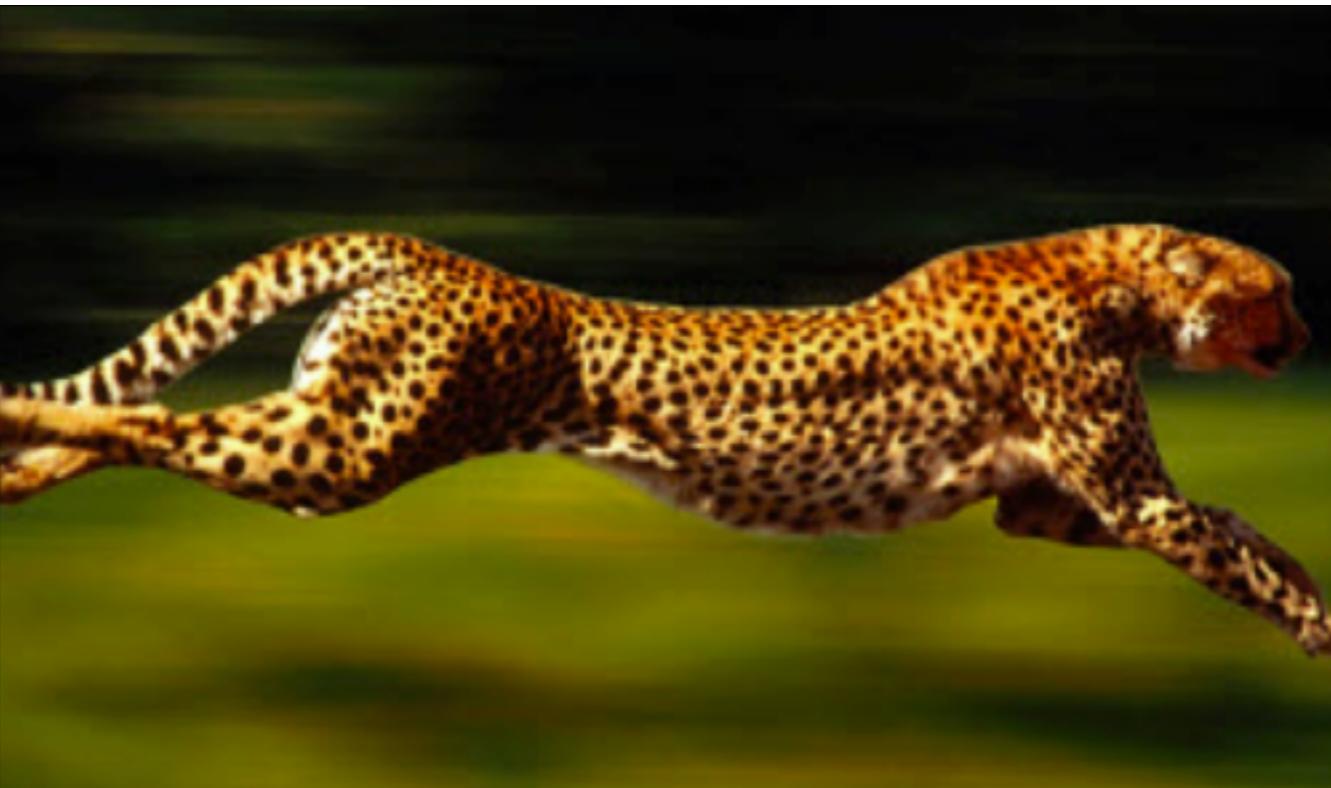
O que **NÃO** é uma diferença chave entre os ciclos de vida Cascata, Espiral e Ágil?

- A. Cascata utiliza fases longas, Ágil utiliza iterações rápidas e Espiral iterações longas
- B. Cascata não possui código funcional antes do fim, Espiral e Ágil tem código funcional em cada iteração
-  C. Cascata e Espiral utilizam requisitos formalmente escritos porém Ágil não utiliza nada escrito
- D. Cascata e Espiral possuem fases de projeto arquitetural porém você não pode incorporar a arquitetura de SW no ciclo de vida Ágil

24

Waterfall uses written requirements, but Agile does not use anything written down is the answer to question

Beck: Part of coping with change is making change at architectural level -



Agile

Development processes

earliest ever lecture prep!
follow along download slides

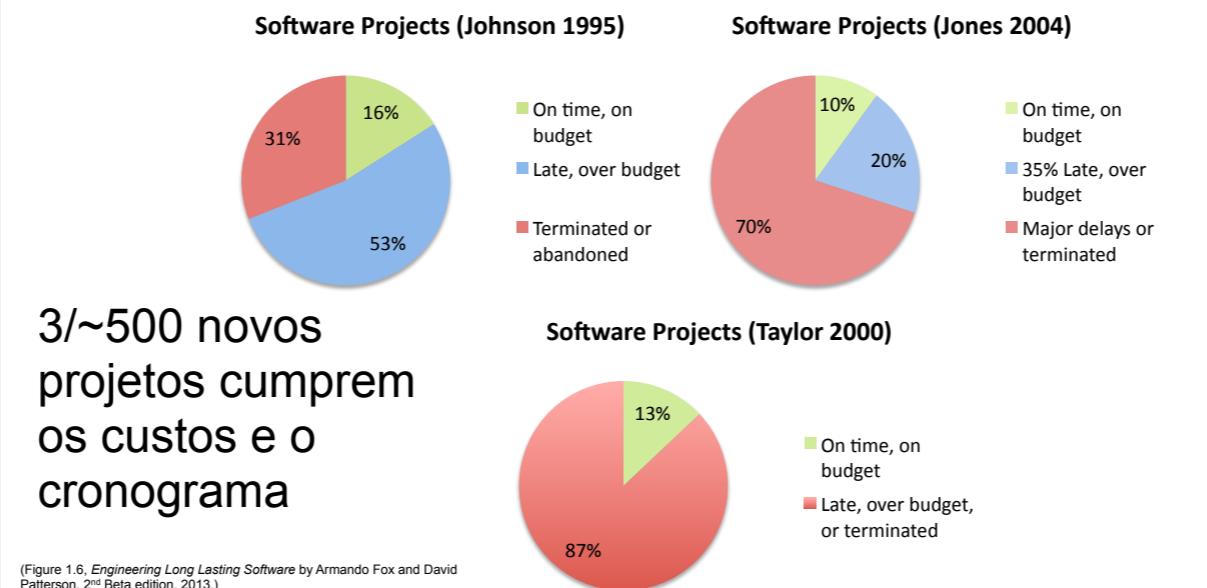
Processo Alternativo?

- Quão **preciso** o Dirigido a Plano pode ser em relação a custo, cronograma e qualidade?
- DP requer **extensiva documentação e planejamento** e depende da **experiência** do gerente
 - É possível construir software efetivamente, e com sucesso, sem um cuidadoso planejamento e documentação?
 - Como evitar o “**basta cortar**”?

Quão preciso são os processos dirigido a planos?

- IEEE Spectrum “Software Wall of Shame”

- 31 projetos: desperdiçaram \$17B



27

Figure 1.6: The first study of software projects found that 53% of projects exceeding their budgets by a factor of 2.9 and overshot their schedule by a factor of 3.2 and another 31% of software projects were cancelled before completion (Johnson 1995). The estimated annual cost in the United States for such software projects was \$100B. The second survey of 250 large projects, each with the equivalent of more than a million lines of C code, found similarly disappointing results (Jones 2004). The final survey of members of the British Computer Society found that only 130 of 1027 projects met their schedule and budget. Half of all projects were maintenance or data conversion projects and half new development project, but the successful projects divided into 127 of the former and just 3 of the latter (Taylor 2000).

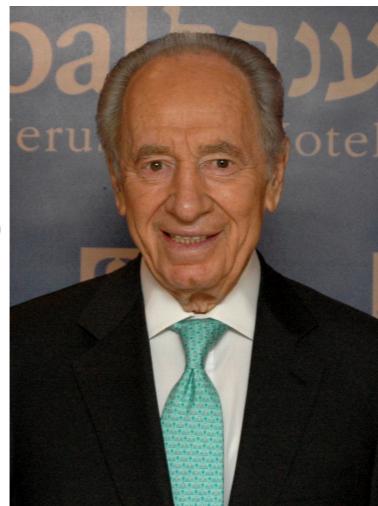
Peres's Law

"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)



28

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.”

“Extreme Programming” (XP), uma versão do ciclo de vida ágil

- 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 code before you write the code to test.
- If code reviews are good, review code continuously, by programming in pairs, taking turns looking over each other's shoulders.

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 **Iteration** (every ~1 to 2 weeks)
- Agile emphasises **Test-Driven Development** (TDD) to reduce mistakes, written down **User Stories** to validate customer requirements, **Velocity** to measure progress

31

Working but incomplete

More iterations, the more you'll learn

Write Tests before write code

Customer helps write SW requirements

Agilidade então... E agora?

Controverso em 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

Aceito em 2003

Estudo de 2012 com 66 projetos confirmaram que a maioria usava Metodologias Ágeis, mesmo com times distribuídos

Sim: Dirigido a Plano
Não: Ágil

[Sommerville, 2011]

1. É importante ter uma especificação e projeto?
2. Os clientes estão disponíveis para feedback?
3. O Sistema a ser desenvolvido é grande?
4. O Sistema é complexo (ex. tempo real)?
5. Vai ter um ciclo de vida longo?
6. Está utilizando ferramentas “ruins”?
7. O time está geograficamente distribuído?
8. A cultura do time é orientada a documentação?
9. O Time tem um perfil “fraco” de desenvolvimento?
10. 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

34

VertLeftWhiteCheck1

While Agile does not develop extensive requirements documents as does Waterfall, the interactions with customers lead to the creation of requirements as user stories, as we shall see in Chapter 7

Both build working but incomplete prototypes that the customer helps evaluate. The difference is that customers are involved every two weeks in Agile versus up to two years in with Spiral.

The answer is measuring progress against a plan

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

35

VertLeftWhiteCheck1

While Agile does not develop extensive requirements documents as does Waterfall, the interactions with customers lead to the creation of requirements as user stories, as we shall see in Chapter 7

Both build working but incomplete prototypes that the customer helps evaluate. The difference is that customers are involved every two weeks in Agile versus up to two years in with Spiral.

The answer is measuring progress against a plan



Fallacies and Pitfalls

Chapter 1 Summary

(Engineering Long Lasting Software §§1.12, 1.13)

Falácia e Armadilhas

- **Falácia:** Se um projeto de SW está falhando em relação ao cronograma, melhore-o adicionando pessoas!
 - Adicionar mais pessoas piora!
 - Curva de aprendizado para novas pessoas
 - Comunicação aumenta quando o projeto cresce, o que reduz a disponibilidade para concluir o trabalho

“Adding manpower to a late software project makes it later.”

Fred Brooks, Jr. The Mythical Man Month

37

Fallacy – sounds true, but false

Pitfall – know it might happen, but it happens anyways

If behind, throw stuff out

Falácia e Armadilhas

- **Falácia:** O ciclo de vida Ágil é melhor para o desenvolvimento de SW
 - Bom para alguns projetos, melhor para outros
 - Mas não é uma boa escolha para foguetes da NASA
- Pode-se aprender os princípios da ES de muitas maneiras, e pode-se aplicá-los em tantas outras, escolha a sua forma (Ágil/SaaS)
 - Nota mental: *Na sua vida profissional você irá se deparar com inúmeras outras oportunidades, espera-se que você sempre aprenda algo novo*

38

Fallacy – sounds true, but false

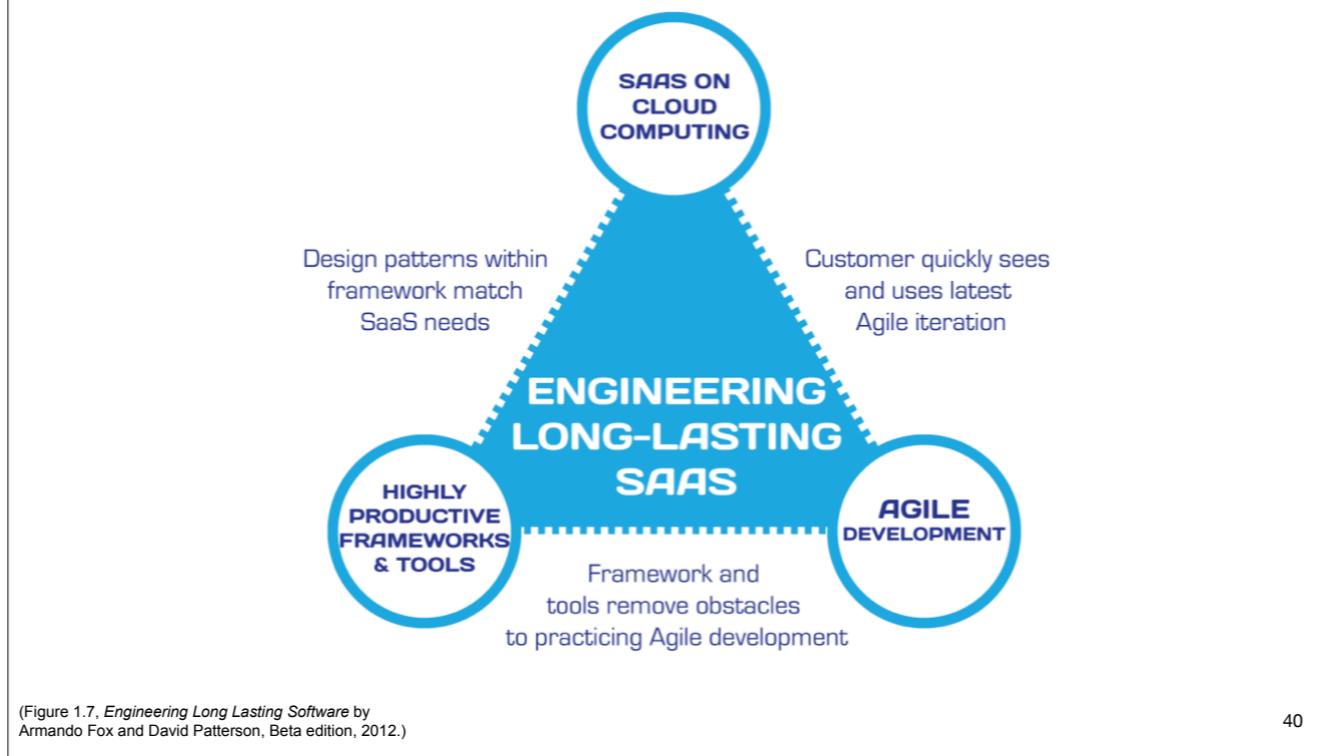
Pitfall – know it might happen, but it happens anyways

If behind, throw stuff out

Falácia e Armadilhas

- **Armadilha:** Ignore o custo do projeto de SW
 - Desde ≈ 0 é o custo de fabricar um SW, pode-se inferir que ≈ 0 é o custo de refazer o SW da forma que o cliente quer
 - Ignore o custo de projetar e testar
 - Abordagem da pirataria: *ninguém deveria pagar pelo desenvolvimento, apenas pela fabricação?*

Concluindo: ES é muito mais do que Programação



Virtuous triangle: 3 jewels that course is based on

Top: SaaS revolutionize SW, depends on Cloud

Right: Agile, invented independent of other jewels, just a great match

Left: Ruby independent but Rails created to do Agile

Chain between the jewels:

Cloud & Agile: deployed on cloud so customers sees every 2 weeks

Cloud & Frameworks: frameworks have design within framework that matches SaaS needs

Tools & Agile: Not preaching, but tools to implement Agile principles - TDD



(Engineering Software as a Service §10.1)

Eng SW como um time de futebol

- **Era dos Programadores Super-Heróis**
- **Aumento de qualidade e funcionalidades**



- O programador não tem como resolver tudo sozinho

- **Carreira de sucesso**

- Ninguém ganha jogo sozinho, quando vence, vencem todos

*“There are no winners on a losing team,
and no losers on a winning team.”*

Fred Brooks, Jr.

42

Early in my career knew superhero programmers – go away for weekend and build amazing breakthrough

Ken Thompson, co-inventor of UNIX, won Turing Award, was a Berkeley student

Bill Joy, led BSD UNIX, which was first open source project, hero programmer – wrote vi over long weekend

No longer individuals building amazing things – Linus Torvald leads open source around Linux, but not done by himself

Early – bar was low

Not that many superheroes

What does team need to win (do you need to do more than your part)

Time: Maior campeão da UFSCAR

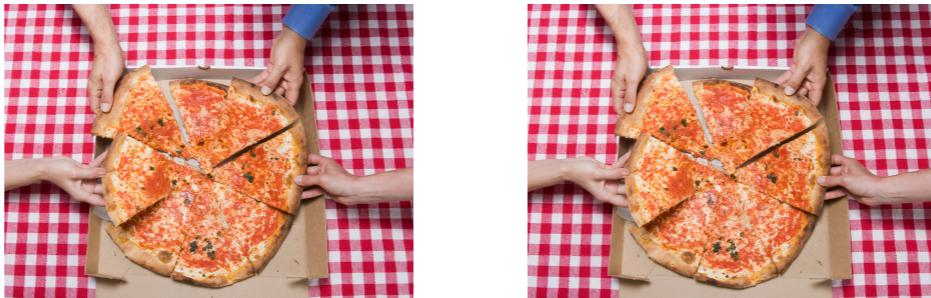


43

Organização alternativa para o time?

- Processo Dirigido a Planos requer uma **extensiva** documentação e planejamento e seu **sucesso** depende da experiência do gerente
- Como devemos organizar um time Ágil?
- Existe alguma alternativa a organização **hierárquica** com um **gerente** que coordena o projeto?

SCRUM: Organização do time



- Time para “Duas Pizzas” (4 a 9 pessoas)
- Scrum é inspirado por pequenas reuniões frequentes
 - 15 minutos todo dia, no mesmo lugar e hora
 - Para aprender mais: Agile Software Development with Scrum by Schwaber & Beedle

45

Bezos, lots of companies, (2 pizza teams in more people)

Need a way to self-organize

Scrum – penalty

Standup Meeting

Its 9:08, over by 9:23

What happens at meeting – go

Daily SCRUM Agenda

- Responda 3 perguntas nas “*daily scrums*”:
 1. O que foi feito ontem?
 2. O que está planejado para hoje?
 3. Há algum impedimento ou obstáculo?
- Ajuda as pessoas a identificar o que elas precisam

46

Go around in a circle

By understanding what each team member was doing, the team can identify work that would help others make more rapid progress

This would unblock me if you would do this first

Most is common sense

Papéis do Scrum

- Time: times com tamanho 2-pizza que entregam SW
- **Scrum Master**, membro do time que
 - Atua como um facilitador que organiza reuniões diárias
 - Mantêm o *backlog* do trabalho a ser feito e o time focado
 - Grava decisões e mede o processo usando o *backlog*
 - Comunica-se com os clientes e a gerência fora da equipe.
 - Remove os impedimentos e obstáculos que impede o time de fazer progresso no projeto



47

Not the manager – SM is supposed to be the buffer so team can concentrate

Papéis do Scrum

- **Product Owner:** um membro do time (não é o *Scrum Master*) que representa a voz do cliente e prioriza as histórias do usuário



48

Team member, not the scrum master, represents customer at daily scrums

Also prioritises user stories

Resolvendo conflitos

- e.g. Diferentes visões em uma determinada decisão técnica
 - Primeiramente liste todos os itens em que os lados concordam
 - Ao invés de começar pelas diferenças
 - Descobriu que estão mais próximos do que imaginavam?
 - Cada lado articula argumentos do outro, mesmo que não concorde com eles
 - Evita confusões sobre termos ou suposições, que pode ser a verdadeira causa do conflito

Resolvendo conflitos

- Confronto construtivo (Intel)
 - Se você tem uma forte opinião que uma pessoa está propondo uma solução errada tecnicamente, você é obrigado a trazer isso à tona, mesmo que seja o seu chefe
- Discordo e compromisso (Intel)
 - Uma vez tomada uma decisão, todos precisam abraçá-la e seguir em frente
 - “*Eu discordo, mas eu vou ajudar mesmo que não concorde*”
- **Resolução de conflitos é muito útil inclusive na vida pessoal!**

Resumindo SCRUM



- O produto é dividido em um conjunto de partes gerenciáveis e inteligíveis.
- Requisitos instáveis não impedem o progresso.
- Toda a equipe tem visão de tudo e consequentemente a comunicação da equipe é melhorada.
- Os clientes recebem a entrega dos incrementos no tempo certo, além do *feedback* de como o produto funciona.
- Se estabelece a confiança entre os clientes e os desenvolvedores e se cria uma cultura positiva na qual todos acham que o projeto dará certo.

Pergunta

Qual afirmação é **VERDADEIRA**?

- A. Comparado ao Scrum, gerentes de projeto DP atuam tanto como Scrum Master e Product Owner
- B. Times devem tentar evitar conflitos a qualquer custo
- C. DP pode trabalhar com times maiores do que Scrum que reportam diretamente ao gerente de projeto
- D. Conforme estudos mostraram, 84%-90% são projetos dentro do prazo e orçamento, assim gerentes DP podem prometer aos clientes um conjunto de funcionalidades dentro de um prazo e custo acordado

52

1. True.
2. False. While conflicts need to be resolved, they can be helpful in finding the best path forward for a project.
3. False. While teams can be larger, each team is 2-pizza sized, and composed as a hierarchy (2 level management)
4. False. – 84% to 90% project are late and/or over budget

Pergunta

Qual afirmação é **VERDADEIRA**?

- A. Comparado ao Scrum, gerentes de projeto DP atuam tanto como Scrum Master e Product Owner
- B. Times devem tentar evitar conflitos a qualquer custo
- C. DP pode trabalhar com times maiores do que Scrum que reportam diretamente ao gerente de projeto
- D. Conforme estudos mostraram, 84%-90% são projetos dentro do prazo e orçamento, assim gerentes DP podem prometer aos clientes um conjunto de funcionalidades dentro de um prazo e custo acordado

53

1. True.
2. False. While conflicts need to be resolved, they can be helpful in finding the best path forward for a project.
3. False. While teams can be larger, each team is 2-pizza sized, and composed as a hierarchy (2 level management)
4. False. – 84% to 90% project are late and/or over budget

Desafio & Atividades

- CodeAcademy Ruby 11-19 (até D+7, 23:59)
- SaaS TV: Raffi Krikorian
 - <http://www.youtube.com/watch?v=IPAFuhSXBck>