

# [IF977] Engenharia de Software

---

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

**AssertLab**  
Advanced Software and Systems  
Engineering Research Technologies



## Licença do material

---

Este Trabalho foi licenciado com uma Licença

**Creative Commons - Atribuição-NãoComercial-  
CompartilhaIgual 3.0 Não Adaptada.**

Mais informações visite

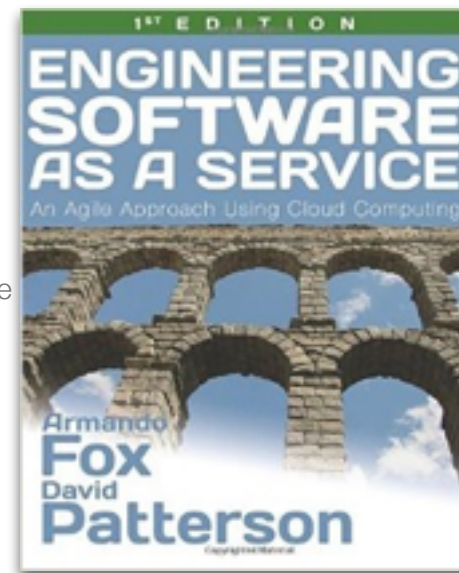
<http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt>



## Referências

---

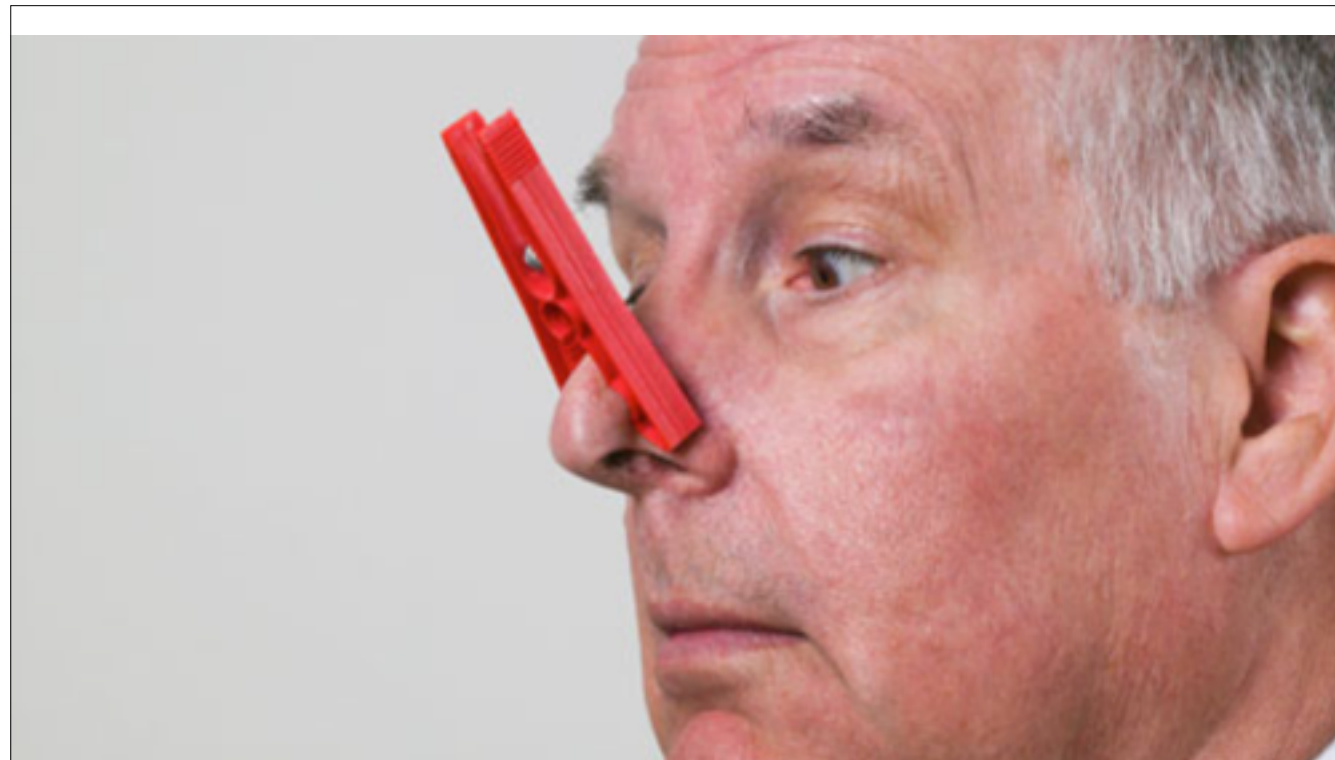
- A biblioteca do Desenvolvedor de Software dos dias de hoje
  - <http://bit.ly/TDOA5L>
- SWEBOK
  - Guide to the Software Engineering Body of Knowledge (SWEBOK): <http://www.computer.org/web/swebok>
- Engineering Software as a Service: An Agile Approach Using Cloud Computing (Beta Edition)
  - <http://www.saasbook.info/>



# Outline

---

- (ESaaS 9.5) Identifying What's Wrong: Smells, Metrics, SOFA
- (ESaaS 7.1) Intro to BDD & User Stories
- Administrivia
  - (ESaaS 7.3) Points, Velocity, and Pivotal Tracker
  - (ESaaS 7.2) SMART User Stories
  - (ESaaS 7.7) Lo-Fi User Interface Sketches and Storyboards
  - (ESaaS 7.4) Agile Cost Estimation
  - (ESaaS 7.10) Plan-and-Document Perspective



Identificando o que está  
errado: smells, métricas,  
SOFA

<http://pastebin.com/gtQ7QcHu>

## A (bad) Ruby calculator

---

```
1 class TimeSetter
2
3   def convert(d)
4     y = 1980
5     while (d > 365) do
6       if (y % 400 == 0 ||
7         (y % 4 == 0 && y % 100 == 0))
8         if (d > 366)
9           d -= 366
10          y += 1
11        end
12      else
13        d -= 365
14        y += 1
15      end
16    end
17    return y
18  end
19
20 end
```

## Além da Corretude

---

- Podemos dar feedback sobre beleza de software?

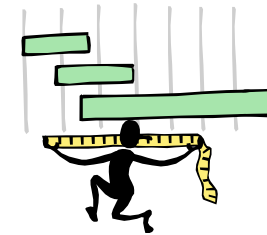
- Orientações sobre o que é belo?

- Avaliações qualitativas?

- Avaliações quantitativas?

- Se sim, quão bem eles funcionam?

- E Rails tem ferramentas para apoiá-los?



## Qualitativo: Code Smells

---

- CUPA (do inglês SOFA) captura sintomas que muitas vezes indicam o "cheiro" código:
- É **C**urto (**S**hort)?
- Ele faz apenas **U**ma (**O**ne) coisa?
- Será que ela tem **P**oucos (**F**ew) argumentos?
- É um nível consistente de **A**bstração (**A**bstraction)?
- A ferramenta **reek** do Rails encontra "cheiro" de código



8

Code smell alerts you that something may be wrong  
20 to 60 code smells depending on authority  
Code spend several lectures on them, but just give an idea now

Methods that don't follow this advice will often give off several of the smells

Short – depends on language (More than 1 screenful for old monitors) – try to do one thing, and quickly grasp

old terminals used to be 24 lines by 80 characters, so short fit in one screen  
Today windows are tremendously larger, so not a good guideline  
Want to look at method and quickly figure out what it is

One thing – may just call a bunch of methods do this



## Único Nível de Abstração

---

- Tarefas complexas precisam do “dividir para conquistar”
- Bandeira amarela para "encapsular essa tarefa em um método"
- Como uma boa notícia, as classes e métodos deve ler “de cima para baixo”!
  - Bom: começar com um resumo de alto nível dos pontos-chave, em seguida, entrar em cada ponto em detalhe
  - Bom: cada parágrafo lida com um tópico
  - Bad: divagar, saltando entre "níveis de abstração", em vez de refino progressivamente

9

Concentrate on Abstraction since correlated and Most rapidly makes your code start to make ugly, correleates well with other smells

Divide and Conquer: Don't have one method that does everything. Divide into understandable pieces, and have methods call others

One that orhestrates all the work

Symptom – 1 line of code says what to do, many lines of code around it saying how to do it

One of other SOFA is mulitple rguments

## Por que Muitos Argumentos é Ruim?

---

- Difícil de obter uma boa cobertura de testes
- Difícil de fazer mock/stub durante o teste
- Argumentos booleanos devem ser uma bandeira amarela
  - Se a função se comporta de forma diferente com base no valor booleano do argumento, talvez deva ser 2 funções
- Se os argumentos "viajam em um pacote", talvez você precisa extrair uma nova classe
  - Mesmo conjunto de argumentos para um monte de métodos

10

If behavior really depends on lots of arguments, then lots to test

Also makes it hard to mock/stub if many args interact

Boolean – made 1 method harder to test than 2 methods

# Program X & Smells

```
class TimeSetter
  def convert(d)
    y = 1980
    while (d > 365) do
      if (y % 400 == 0 ||
          (y % 4 == 0 && y % 100 != 0))
        if (d > 366)
          d -= 366
          y += 1
        end
      else
        d -= 365
        y += 1
      end
    end
    return y
  end
end
```

time\_setterTimeSetter#self.convert calls  
(y + 1) twice (Duplication)

.rb -- 5 warnings:

- TimeSetter#self.convert calls  
(y + 1) twice (Duplication)
- TimeSetter#self.convert has approx 6 statements  
(LongMethod)
- TimeSetter#self.convert has the parameter name  
'd' (UncommunicativeName)
- TimeSetter#self.convert has the variable name  
'd' (UncommunicativeName)
- TimeSetter#self.convert has the variable name  
'y' (UncommunicativeName)

A little Ruby goes a long way; if stare at a long time and can't figure it out, its too long

Not DRY (y+1)

Uncommunicative variables

- Suppose hebrew calendar, suppose not english

## Quantitativo: Complexidade ABC

---

- Contagens de Assignments, Branches e Condições
  - Branching - :and, :case, :else, :if, :or, :rescue, :until, :when, :while
- Pontuação = raiz quadrada ( $A^2 + B^2 + C^2$ )
- NIST (National Inst Stds. & Tech...):  $\leq 20$  / método
- Ferramenta flog do Rails verifica a complexidade ABC

12

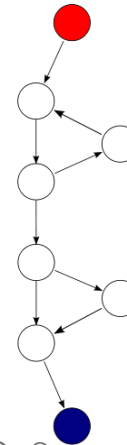
Branching terms are :and, :case, :else, :if, :or, :rescue, :until, :when, :while.

NIST – standard for gov't contracts

## Quantitativo: Complexidade Ciclômática (McCabe)

- # de caminhos linearmente independentes no código =  $E - N + 2P$  (edges, nodes, connected components)

```
def mymeth
  while(...)
    ....
  end
  if (...)
    do_something
  end
end
```



Ferramenta saikuro do Rails  
calcula a complexidade  
ciclômática

- No exemplo,  $E=9$ ,  $N=8$ ,  $P=1$ , so  $CC=3$
- NIST (Natl. Inst. Stds. & Tech.):  $\leq 10/\text{módulo}$

a graphical measurement of the number of possible paths through the normal flow of a program

Node is basic block

Connected components

Also called McCabe Complexity, after the inventor

## Quantitativo: Métricas

Metric	Tool	Target score
Code-to-test ratio	rake stats	$\leq 1:2$
C0 (statement) coverage	SimpleCov	90%+
Assignment-Branch-Condition score	flog	< 20 per method
Cyclomatic complexity	saikuro	< 10 per method (NIST)

- “Hotspots”: lugares onde várias métricas **levantam bandeiras vermelhas**
  - `add require 'metric_fu' to Rakefile`
  - `rake metrics:all`
- Considere as métricas com um grão de sal
  - Como cobertura, melhor para identificar onde há oportunidade de melhoria

Several tools point to difficulties than get your attention

# Ano Bissexto & Quantitativo

---

```
class TimeSetter
  def convert(d)
    y = 1980
    while (d > 365) do
      if (y % 400 == 0 ||
          (y % 4 == 0 && y % 100 != 0))
        if (d > 366)
          d -= 366
          y += 1
        end
      else
        d -= 365
        y += 1
      end
    end
    return y
  end
end
```

- ABC contou 23 (>20 então é um problema))
- Complexidade do código 4 ( $\leq 10$  então não é um problema)

## Ano Bissexto **Revisado** & Métricas

---

```
class TimeSetter
  def self.convert(day)
    year = 1980
    while (day > 365) do
      if leap_year?(year)
        if (day >= 366)
          day -= 366
        end
      else
        day -= 365
      end
      year += 1
    end
    return year
  end
end
```

```
private
def self.leap_year?(year)
  year % 400 == 0 ||
    (year % 4 == 0 && year % 100 != 0)
end
end
```

- **Reek**: No Warnings
- **Flog (ABC)**:
  - TimeSetter.convert = 11
  - TimeSetter.leap\_year? = 9
- **Saikuro (Code Complexity)** = 5



## Pergunta

---

Que orientação CUPA é mais importante para o teste de nível de unidade?

- A. Curto
- B. Fazer uma coisa
- C. Ter poucos argumentos?
- D. Ater-se a um nível de abstração

17

Opinion vs. fact, so can make arguments for several, and could come up with counterexamples for specific programs why one is more important than the other

Short – long and straightline could still be easy to test

One Thing – if doing multiple things, but each is simple, could still be easy to test

Answer: Our opinion is few arguments – if arguments matter, then lots of degrees of behaviors, lots to test

(could be just a lot of arguments just go into one formula that is easy to test, OK, but probably should be own new class, that can test separately)

4. One level abstraction means hard to understand, but not necessarily hard to test

## Pergunta

---

Que orientação CUPA é mais importante para o teste de nível de unidade?

A. Curto

B. Fazer uma coisa



C. Ter poucos argumentos?

D. Ater-se a um nível de abstração

18

Opinion vs. fact, so can make arguments for several, and could come up with counterexamples for specific programs why one is more important than the other

Short – long and straightline could still be easy to test

One Thing – if doing multiple things, but each is simple, could still be easy to test

Answer: Our opinion is few arguments – if arguments matter, then lots of degrees of behaviors, lots to test

(could be just a lot of arguments just go into one formula that is easy to test, OK, but probably should be own new class, that can test separately)

4. One level abstraction means hard to understand, but not necessarily hard to test

Comentários devem descrever coisas que **NÃO** são óbvias a partir do código

"por quê", não "o quê"

AssertLab

- \* This means 2 things for writing comments:
  - \* Don't just repeat what's obvious from the code
  - \* Do think about what's not obvious (low level and high level):
    - \* The units for variables, invariants
    - \* Design issues that went through your mind while you were writing the code: why the code is written this way.
    - \* Subtle problems that required a particular implementation
  - \* Abstractions:
    - \* Goal is to write classes and other code that hides complexity: easier to use than to write
    - \* Abstractions won't be obvious from implementation; comments should describe these (should be simple).
    - \* For example, what do I need to know to invoke a procedure? I shouldn't have to read the code of a procedure before calling it.
- \* "Obvious" refers to someone who will come along later and read your code, not you.

Must write comments as you code; if you come back later you will

## Bad Comments

---

```
// Add one to i.  
i++;
```

```
// Lock to protect against concurrent access.  
SpinLock mutex;
```

```
// This function swaps the panels.  
void swap_panels(Panel* p1, Panel* p2) {...}
```

20

- \* Most software is poorly documented
- \* A lot of code has no comments at all
- \* Many open source projects set a very bad example
- \* When present, comments often not helpful
- \* Many people think comments are a bad idea
  - \* Not useful
  - \* Out of date
  - \* Clutter the code
  - \* "Just let me read the code!"

## Comments (cont)

---

- Comentários devem ter um nível de abstração maior do que o código

```
# Scan the array to see if the symbol exists
```

**not**

```
# Loop through every array index, get the  
# third value of the list in the content to  
# determine if it has the symbol we are looking  
# for. Set the result to the symbol if we  
# find it.
```

21

- \* The biggest problem is that most people don't know how to write comments.
- \* "My code is self-documenting" (one of the great lies)

There are many other facets of writing good documentation, but this one rule will get you most of the way.

Armando agrees with Why?

- \* Alas, can't scale



## Introdução a Behavior-Driven Design & User Stories

Next Chapter

Been preparing for building SaaS apps; now ready to build

## Por que projetos de SW falham?

---

- Não atendem a necessidade dos clientes
  - Não é o que eles pediram/quêrem
- Ou os projetos estão atrasados/atrasam
- Ou estão acima do orçamento (custo)
- Ou são difíceis de manter e evoluir
- Ou todas as alternativas acima
- Como o Metodologias Ágeis tentar **evitar** este fracasso?

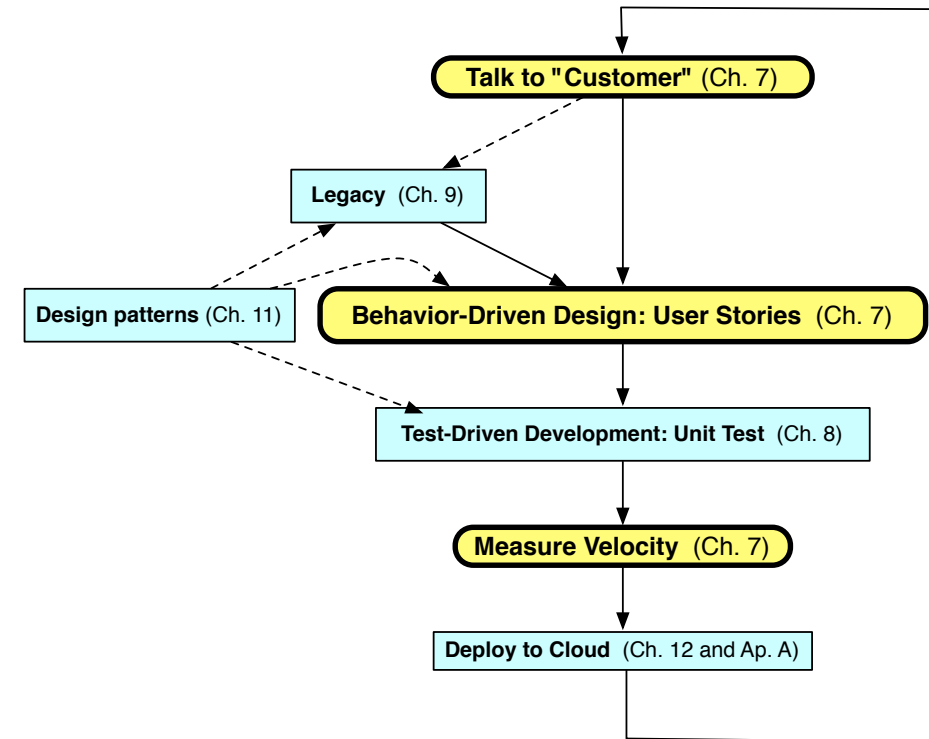
## Ciclo de Vida Ágil :: Revisão

---

- Trabalhar em **estreita colaboração**, de forma **contínua** com as partes interessadas na elaboração de requisitos, testes
  - Usuários, clientes, desenvolvedores, programadores de manutenção, operadores, gestores de projectos...
- Manter um protótipo de trabalho (**funcional**) durante a implantação de novos recursos a cada **iteração**
  - Tipicamente a cada 1 ou 2 semanas
  - Em vez de cinco fases principais, meses de duração
- Verifique com as partes interessadas sobre o que está próximo, para garantir a construção da coisa certa (vs. verificar)



# Iteração Ágil



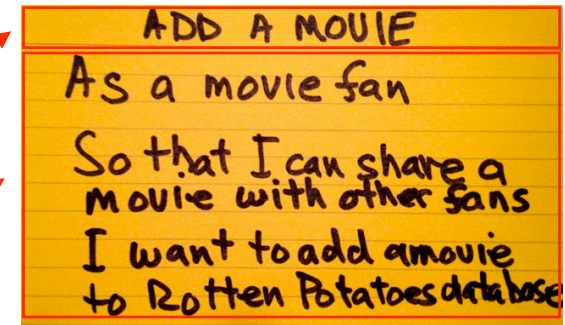
## Behavior-Driven Design (BDD)

---

- BDD faz perguntas sobre o comportamento da app *antes e durante o desenvolvimento*, para reduzir falhas de comunicação
  - Validação vs. Verificação
- Requisitos escritos como *histórias de usuário*
  - Leves descrições de como a app é utilizada
- BDD concentra-se no *comportamento* de execução da aplicação ao invés da sua *implementação*
  - Test-Driven Design ou TDD (em breve) testa a implementação

# User Stories

- 1-3 sentenças em linguagem natural (cotidiana)
  - Se encaixa em um cartão 3"x5"
  - Escrito *por/com* cliente
- Formato "connextra":
  - nome do recurso/característica (feature)
  - **Como um** [tipo de stakeholder],
  - **Para que** [eu possa alcançar algum objetivo],
  - **Eu quero** [fazer alguma tarefa]
  - As 3 frases devem estar lá, podem estar em qualquer ordem
- Ideia: histórias de usuário podem ser formuladas como testes de aceitação antes que o código seja escrito



Dave Patterson created this image  
From HCI community

## Por que cartão 3x5?

---

- (a partir da comunidade de User Interface)
- Nonthreatening => todos os stakeholders participam do brainstorming
- Fácil de reorganizar => todos os stakeholders participam na definição de prioridades
- Como histórias devem ser curtas, fica fácil de mudar durante o desenvolvimento
  - Muitas vezes temos novos *insights* durante o desenvolvimento

## Diferentes stakeholders podem descrever o mesmo comportamento de formas diferentes

---

- *Veja quais dos meus amigos estão indo para um show*
    - Como admirador de show
    - Para que eu possa apreciar o show com meus amigos
    - Eu quero ver qual dos meus amigos do Facebook estão indo a um determinado show
- 
- Mostrar patrono de amigos do Facebook
    - Como gerente de bilheteria
    - Para que eu possa induzir um patrono para comprar um bilhete
    - Eu quero mostrar-lhe qual de seus amigos do Facebook está indo para um determinado show

## Product Backlog

---

- Sistemas reais tem 100s de histórias de usuário
- Backlog: histórias de usuário ainda não concluídas
  - (Vocês viram Backlog no Pivotal Tracker)
- Priorizar os itens mais valiosos
- Organizar para que eles correspondam ao SW liberado ao longo do tempo

## Problemas Relacionados (Related Issue)

---

- **Spike**
  - Investigação corriqueira em uma técnica ou problema
  - e.g. spike on recommendation algorithms
- Após a spike concluída, o código deve ser descartado
- "Agora que sei o que utilizar, vou fazer certo!"

## Pergunta

---

Que expressão sobre BDD e histórias de usuários é **FALSA**?

- A. BDD é projetado para ajudar com a validação (construir a coisa certa), complementar a verificação
- B. BDD deve testar a implementação da app
- C. Histórias do Usuário são o equivalente a requisitos de projeto na abordagem Dirigida a Plano
- D. Esta é uma História de Usuário válida:  
“Search TMDb  
I want to search TMDb  
As a movie fan  
So that I can more easily find info”



## Pergunta

---

Que expressão sobre BDD e histórias de usuários é **FALSA**?

- A. BDD é projetado para ajudar com a validação (construir a coisa certa), complementar a verificação
- ➔ B. BDD deve testar a implementação da app
- C. Histórias do Usuário são o equivalente a requisitos de projeto na abordagem Dirigida a Plano
- D. Esta é uma História de Usuário válida:  
“Search TMDb  
I want to search TMDb  
As a movie fan  
So that I can more easily find info”



## Points, Velocity, and Pivotal Tracker

## Produtividade e Ferramentas

---

- Queremos evitar grande esforço de planejamento nas Metodologias Ágeis? Se sim, como estimar o tempo sem um plano?
- Histórias do Usuário pode ser usado para medir o progresso no projeto?
- O que uma ferramenta deve fazer para ajudar a medição de progresso para Metodologias Ágeis?

## Medindo Produtividade



- Uma medida de produtividade da equipe: calcular a média de histórias/semana?
  - Mas algumas histórias são muito mais difíceis do que outras
- Avalie cada história de usuário com antecedência em uma escala numérica simples
  - 1 para histórias simples, 2 para histórias médias, 3 para histórias muito complexas
- **Velocidade**: número médio de pontos/semana

36

Start with this 3 point scale

Why good idea?

See what you actually accomplish in points per week vs. what you guess as optimistic programmers

Teams learns to get estimate difficulty of user story (gave a story 1 point but took 2 weeks; why was it hard? Learn from mistakes)

## Mais sobre Pontos

---

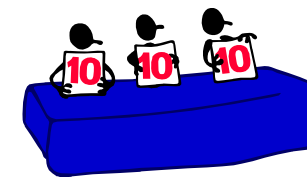
- Depois de ganhar a experiência, a escala Fibonnaci é uma boa sugestão: 1, 2, 3, 5, 8
  - (Cada novo número é a soma dos dois anteriores)
  - Na Pivotal Labs, 8 é extremamente **rara**
- Equipes atribuem valor: voto, mantendo-se os dedos simultaneamente, tirar média
  - Se uma grande discordância aparece (2 e 5), tem que discutir mais sobre a história



37

How decide points – rock, paper, scissors – everyone says how many points a story is, and see if agree  
Group says average and record so can see how well you did - rather

## Mais sobre Pontos



- $\geq 5 \Rightarrow$  divida a história em histórias mais simples
  - backlog não muito exigente
- Não importa se a velocidade é 5 ou 10 pontos por iteração
  - Desde que seja consistente
- A ideia é melhorar a auto-avaliação e sugerir um número de iterações para um dado conjunto de features

38

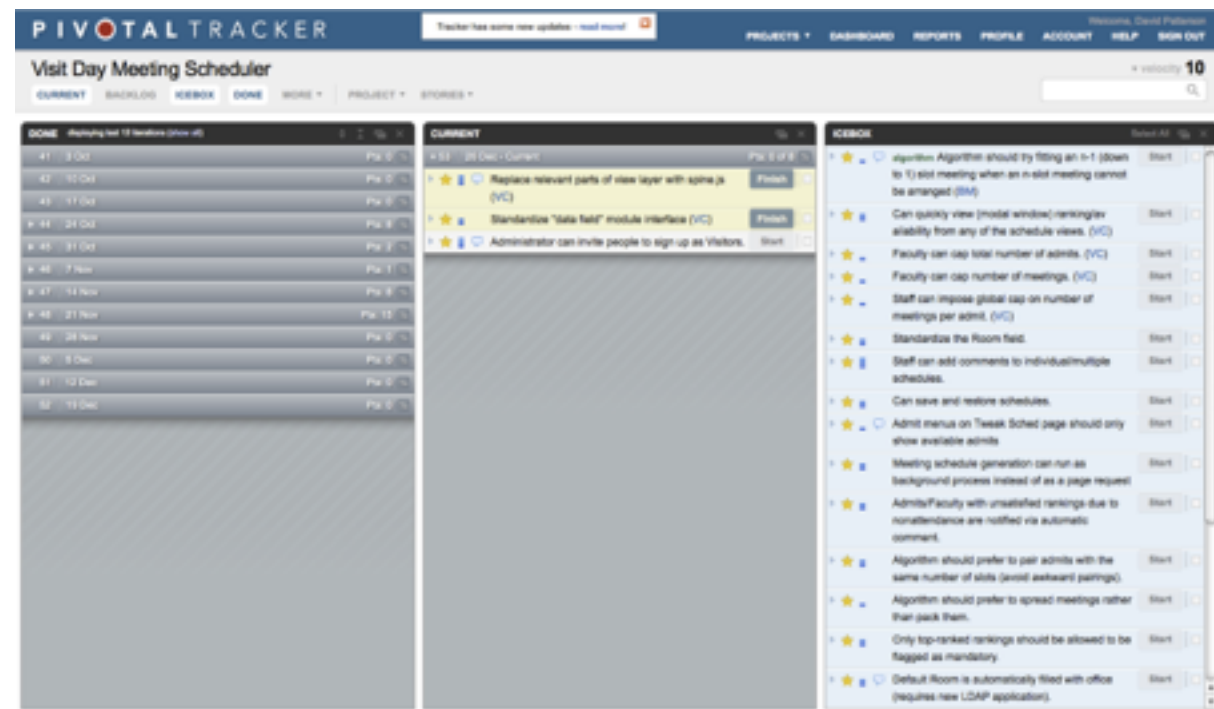
At Pivotal Labs, break into multiple stories

Every time there is a SW concept, some builds a tool to implement idea to make it easy to do

# Pivotal Tracker



- Calcula a velocidade para o time e gerencia as HU: Current, Backlog, Icebox



39

Pivotal Labs brought us Pivotal Tracker

Bookkeeping for user stories, as long as doing that, might as well calculate velocity

Upper right corner is team velocity (10)

Very easy to use

Current

Done

Backlog (not shown) – what work on next

Icebox – invent more stories than will ever implement (frozen)

Pivotal will move things from backlog to current automatically and assign people to the tasks

Top to bottom is prioritized order (except in icebox, it is random)

## Pivotal Tracker



- Priorizar histórias de usuários pela localização: Current, Backlog, Icebox
- Quando concluído, mova para o painel Done
- Pode adicionar pontos de release lógicos, assim podemos descobrir quando um release realmente vai acontecer
  - pontos pendentes / velocidade
- **Epic** (com o próprio painel)
  - Combine as histórias de usuário relacionadas
  - Ordenadas independente das histórias do usuário no Backlog

40

Epic: give software engineers the big picture of where the application is in the development process with regard to big feature

Will put into backlog with Release Point

Tracker will estimate when Release Point looks to be

Tell customer as go along



## Rastreie papéis

---

- Desenvolvedores não decidem quando a história foi completada
  - Clique no botão Finish, que a envia ao "**Product Owner**" (como em uma equipe Scrum)
- PO avalia a história do usuário e então decide
  - Aceitar, que marca história de usuário como feita, ou
  - Rejeitar, que marca história com a necessidade de ser reiniciada pelo desenvolvedor

# Pivotal Tracker: Features vs. Chores

---

- Features
  - Histórias de usuários que fornecem valor de negócio verificável para o cliente
  - "Adicionar checkbox de concordo à página"
  - Pontos valiosos e, portanto, deve ser estimado
- Chores
  - HU que são necessárias, mas não fornecem nenhum valor óbvio diretamente ao cliente
  - "Descubra porque a suíte de testes é tão lenta"
  - Não há pontos

## Team Cyberspace Whiteboard

---

- Tracker permite anexar documentos às HUs (e.g., LoFi UI)
- Wiki no repositório Github
- Google Drive: criação colaborativa e visualização de diagramas, apresentações, planilhas e documentos de texto
- Campfire: web-based service for password-protected online chat rooms (<http://campfirenow.com>)
  - Telegram (<http://telegram.org>); Slack (<http://slack.com>); Google Hangout...

## Pergunta

---

Qual afirmação em relação a Pontos, Velocidade, e Tracker é **VERDADEIRA**?

- A. Quando comparar dois times, o com maior velocidade é mais produtivo
- B. Quando você não sabe como abordar uma HU, basta dar 3 pontos
- C. Com Tracker, desenvolvedores pegam as HUs e marcam como Aceitas quando concluídas
- D. Tracker ajuda a priorizar e acompanhar as HUs e seu status, calcula a velocidade e prevê o tempo de desenvolvimento de software

44

- 1 is false Since each team assigns points to user stories, you cannot use velocity to compare different teams. However, you could look over time for a given team to see if there were some iterations that were significantly less or more productive
2. User stories should not be so complex that you don't know have an approach to implementing it. If they are, you should go back to your stakeholders to refactor the user story into a set of simpler tasks that you do know how to approach
3. Product owners say when done, not developers
4. True

Velocity is self-assessment


Not supposed to not know what you're doing

10 minutes for this section

## Pergunta

---

Qual afirmação em relação a Pontos, Velocidade, e Tracker é **VERDADEIRA**?

- A. Quando comparar dois times, o com maior velocidade é mais produtivo
- B. Quando você não sabe como abordar uma HU, basta dar 3 pontos
- C. Com Tracker, desenvolvedores pegam as HUs e marcam como Aceitas quando concluídas
-  D. Tracker ajuda a priorizar e acompanhar as HUs e seu status, calcula a velocidade e prevê o tempo de desenvolvimento de software

45

- 1 is false Since each team assigns points to user stories, you cannot use velocity to compare different teams. However, you could look over time for a given team to see if there were some iterations that were significantly less or more productive
2. User stories should not be so complex that you don't know have an approach to implementing it. If they are, you should go back to your stakeholders to refactor the user story into a set of simpler tasks that you do know how to approach
3. Product owners say when done, not developers
4. True

Velocity is self-assessment

Not supposed to not know what you're doing

10 minutes for this section



## SMART User Stories

## Criando Histórias do Usuário

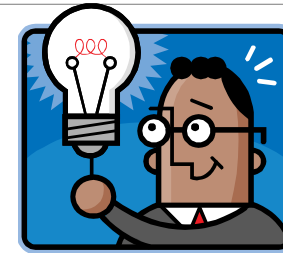
---

- Como você sabe se você tem uma história do usuário boa ou ruim?
  - Tamanho certo?
  - Não é muito difícil?
  - Vale a pena?

## SMART HU

---

- **S**pecific
- **M**easurable
- **A**chievable (idealmente, implementado em 1 iteração)
- **R**elevant (“os 5 porque's”)
- **T**imeboxed (saber quando desistir)





## Specific & Measurable

---

- Cada cenário deve ser testável
  - Implica em conhecer boas entradas e existir os resultados esperados
- Anti-exemplo: "UI deve ser user-friendly"
- Exemplo: Given/When/Then.
  1. Dada alguma condição de partida específica(s),
  2. Quando eu faço X,
  3. Em seguida, um ou mais coisas deve acontecer



## Achievable

---

- Completar em 1 iteração
- Always aim for working code @ end of iteration
- If  $<1$  story per iteration, need to improve point estimation per story
- Se não puder entregar recurso na 1 iteração, entregar subconjunto de histórias
- Sempre apontar para o código de trabalho @ final de iteração
- Se  $<1$  história por iteração, precisamos melhorar estimativa pontual por história



## Relevant: “business value”

---

- Descubra o valor de negócio, ou mate a HU:
  - Proteger as receitas
  - Aumentar a receita
  - Gerenciar o custo
  - Aumentar o valor da marca
  - Tornar o produto notável/reconhecido
  - Fornecer mais valor agregado aos seus clientes
- Um bom exemplo: <http://wiki.github.com/aslakhellesoy/cucumber>



## 5 Porquê's para encontrar relevância

---

- Show patron's Facebook friends
  - As a box office manager
  - So that I can induce a patron to buy a ticket
  - I want to show her which Facebook friends are going to a given show

1. Why?
2. Why?
3. Why?
4. Why?
5. Why?



52

Why add the Facebook feature? As box office manager, I think more people will go with friends and enjoy the show more.

Why does it matter if they enjoy the show more? I think we will sell more tickets.

Why do you want to sell more tickets? Because then the theater makes more money.

Why does theater want to make more money? We want to make more money so that we don't go out of business.

Why does it matter that theater is in business next year? If not, I have no job.

## 5 Porquê's para encontrar relevância

---

1. Why add the Facebook feature? As box office manager, I think more people will go with friends and enjoy the show more.
2. Why does it matter if they enjoy the show more? I think we will sell more tickets.
3. Why do you want to sell more tickets? Because then the theater makes more money.
4. Why does theater want to make more money? We want to make more money so that we don't go out of business.
5. Why does it matter that theater is in business next year? If not, I have no job.

## Timeboxed



- Pare a HU quando ultrapassar orçamento tempo
- Desista, divida em histórias menores ou reagende o que for deixado de lado
- Para evitar subestimar o tamanho do projeto
- Pivotal Tracker rastreia a velocidade, o que ajuda a evitar esta subestimação

## Pergunta

---

Qual feature abaixo é a **MENOS** SMART?

- A. Usuário pode buscar por um filme pelo título
- B. Rotten Potatoes deve ter um bom tempo de resposta
- C. Quando adicionar um filme, 99% das páginas "Add Movie" devem ser exibidas em 3 segundos
- D. Como cliente, eu quero ver os 10 filmes mais vendidos, listados por preço, para que eu possa comprar o mais barato primeiro

55

2nd is least smart

## Pergunta

---

Qual feature abaixo é a **MENOS** SMART?

- A. Usuário pode buscar por um filme pelo título
- ➔ B. Rotten Potatoes deve ter um bom tempo de resposta
- C. Quando adicionar um filme, 99% das páginas "Add Movie" devem ser exibidas em 3 segundos
- D. Como cliente, eu quero ver os 10 filmes mais vendidos, listados por preço, para que eu possa comprar o mais barato primeiro

56

2nd is least smart