

# **FGA206 – Engenharia de Produto de Software-EPS**

## **Introdução aos Métodos Ágeis: Sistemas Kanban**

**Prof. Hilmer Rodrigues Neri**

**As Origins**

# O Sistema de Produção da Toyota

- Taiichi Ohno
  - The Toyota Production System, 1988 (1978)
  - Elimine desperdícios
    - Fluxo “Just-in-Time”
  - Exponha os problemas
    - Stop-the-Line
    - Sistema Kanban



Claudia O. M; Goldman, A. Katayama E. T.; Uma introdução ao Desenvolvimento de Software Lean; Simpósio Brasileiro de Qualidade de Software, 2012

# O Sistema de Produção da Toyota

- Shigeo Shingo
  - formalizou o Sistema de Controle de Defeito Zero
  - Produção sem estoque
  - uso da inspeção na origem



# O Sistema de Produção da Toyota



- Com o passar dos anos, Ohno aplicou seu sistema de produção em supermercados americanos.
  - todos os tipos de produtos foram colocados nas prateleiras em pequenas quantidade
  - quando um cliente retiravam um produto na prateleira, ela era rapidamente reabastecida
- Nesse **sistema**, conhecido como **Pull**, a necessidades linhas ditam o ritmo da produção. E essa forma de operar contrapõe o sistema tradicional, prescritivo, conhecido como **sistema Push**.
- Para operar o sistema de produção Pull, Ohno desenvolveu um conjunto de ferramentas:
  - dentre elas o **Kanban**, que fornece um mecanismo de comunicação visual que transmite informações dentro e entre processos, através de cartões de instruções

KATAYAMA, Eduardo Teruo. A contribuição da indústria da manufatura no desenvolvimento de software [doi:10.11606/D.45.2011.tde-11042012-102429]. São Paulo : Instituto de Matemática e Estatística, University of São Paulo, 2011. Master's Dissertation in Ciência da Computação.

# O Sistema de Produção da Toyota



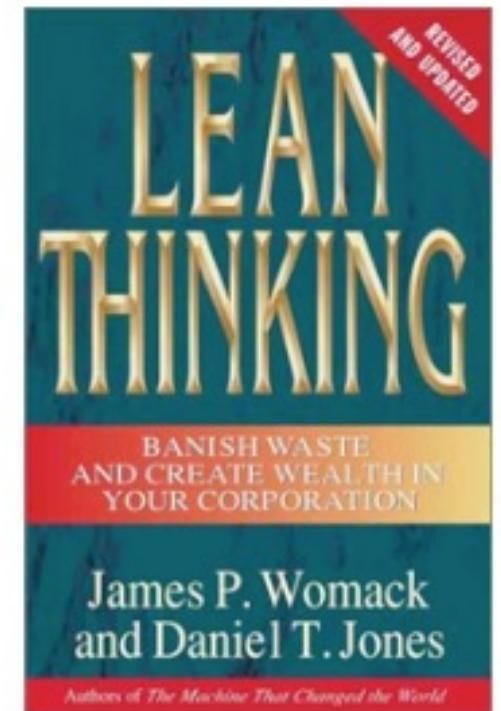
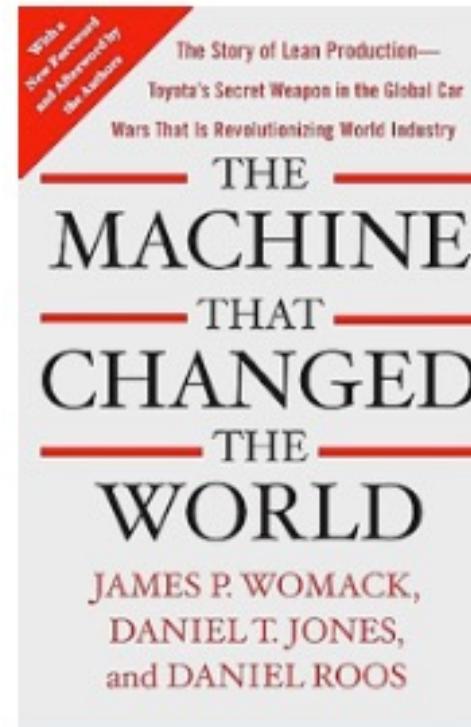
- Quando Shingo foi trabalhar com Ohno eles atacaram no problema de setup do sistema *Pull*:
  - reduziu o tempo médio de horas a minutos.
  - isso permitiu o uso de pequenos lotes e um quase perfeito **fluxo contínuo**



KATAYAMA, Eduardo Teruo. A contribuição da indústria da manufatura no desenvolvimento de software [doi:10.11606/D.45.2011.tde-11042012-102429]. São Paulo : Instituto de Matemática e Estatística, University of São Paulo, 2011. Master's Dissertation in Ciência da Computação.

# Valores LEAN

- Valor
- Eliminar Desperdício
- Fluxo Contínuo
- Pull Systems
- Busca da Perfeição



Claudia O. M; Goldman, A. Katayama E. T.; Uma introdução ao Desenvolvimento de Software Lean; Simpósio Brasileiro de Qualidade de Software, 2012

# Valores LEAN

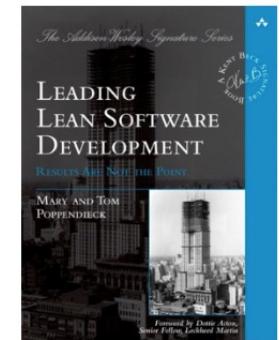
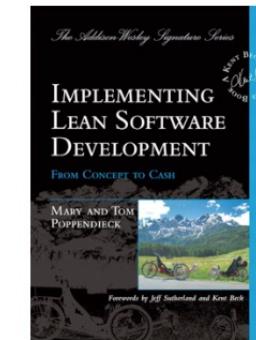
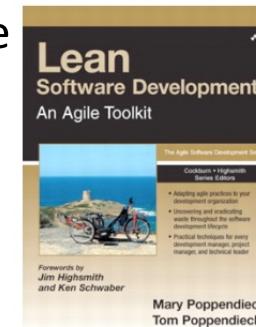
- Os valores foram expandidos na fábrica
  - Linda de Produção Lean
  - Operações Lean
  - Cadeia de suprimentos
  - Desenvolvimento de Produtos Lean
- Lembrete:
  - Desenvolver software = Criar novo Produto!

# **Desenvolvimento de Software LEAN**

# LEAN Software Development

## Origens do Kanban

- O movimento Lean revolucionou a manufatura!
  - e, mais recentemente, o desenvolvimento de produtos e o gerenciamento da cadeia de suprimentos (supply chain management).
- Mary e Tom Poppendieck traçaram os paralelos entre os valores e práticas Lean com o desenvolvimento de software
  - e, propuseram sete princípios

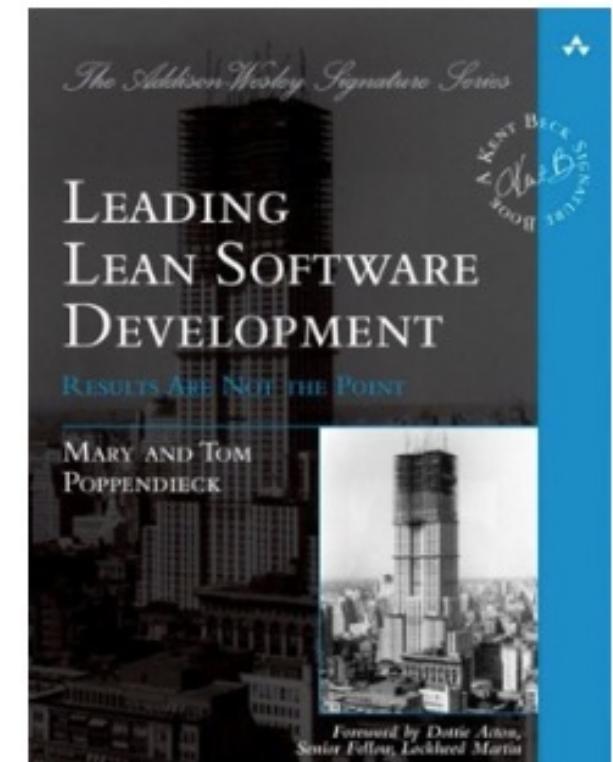
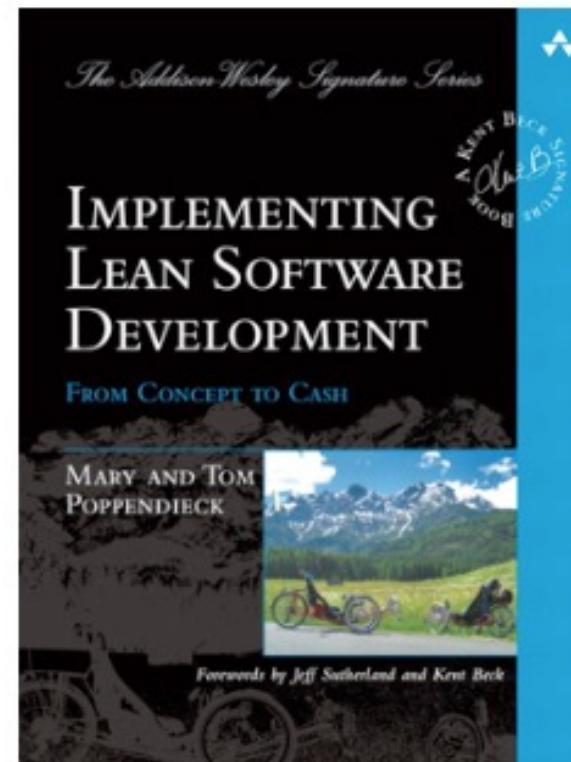
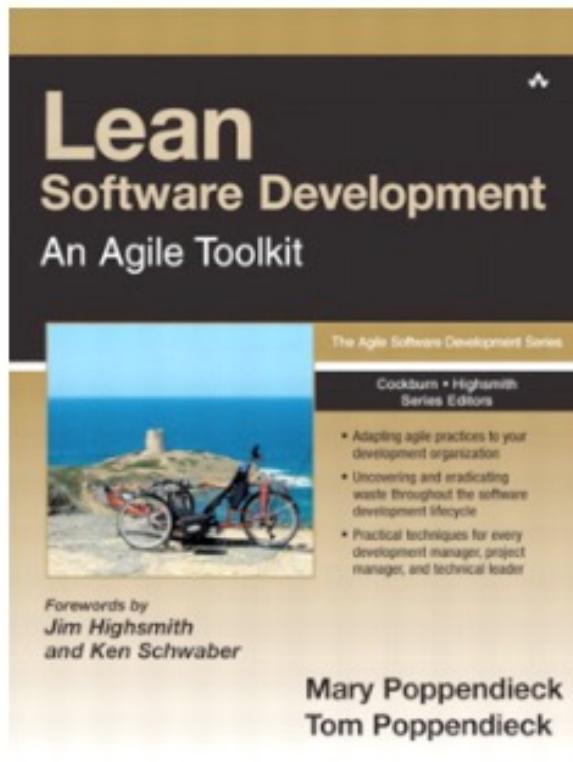


Taiichi Ohno. Toyota Production System: Beyond Large-Scale Production. Productivity Press, 1998

Mary Poppendieck and Tom Poppendieck. Lean Software Development: An Agile Toolkit for Software Development Managers. Addison-Wesley Professional, 2003

Mary Poppendieck and Tom Poppendieck. Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional, 2006.

# LEAN Software Development



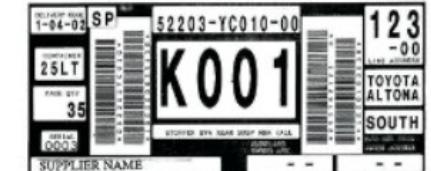
Claudia O. M; Goldman, A. Katayama E. T.; Uma introdução ao Desenvolvimento de Software Lean; Simpósio Brasileiro de Qualidade de Software, 2012

# LEAN Software Development

## Princípios

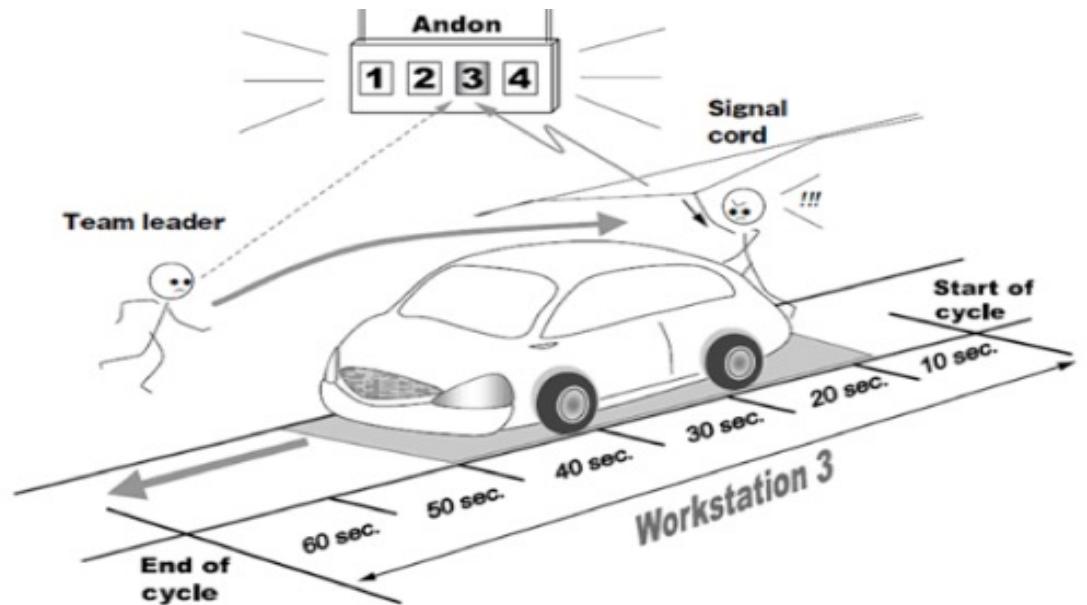
- Elimine desperdícios
- Inclua a qualidade no processo
- Crie conhecimento
- Adie comprometimentos
- Entregue rápido
- Respeite as pessoas
- Otimize o todo

Claudia O. M; Goldman, A. Katayama E. T.; Uma introdução ao Desenvolvimento de Software Lean; Simpósio Brasileiro de Qualidade de Software, 2012



# Kanban

- Sistemas Kanban propõe uma abordagem que se concentra em criar um **fluxo contínuo de trabalho** através de pequenos cartões sinalizadores chamados Kanban.
  - Visibilidade
  - Cadência contínua do fluxo de trabalho
  - Limitar o trabalho



**KATAYAMA, Eduardo Teruo.** A contribuição da indústria da manufatura no desenvolvimento de software. São Paulo : Instituto de Matemática e Estatística, University of São Paulo, 2011. Master's Dissertation in Ciência da Computação.

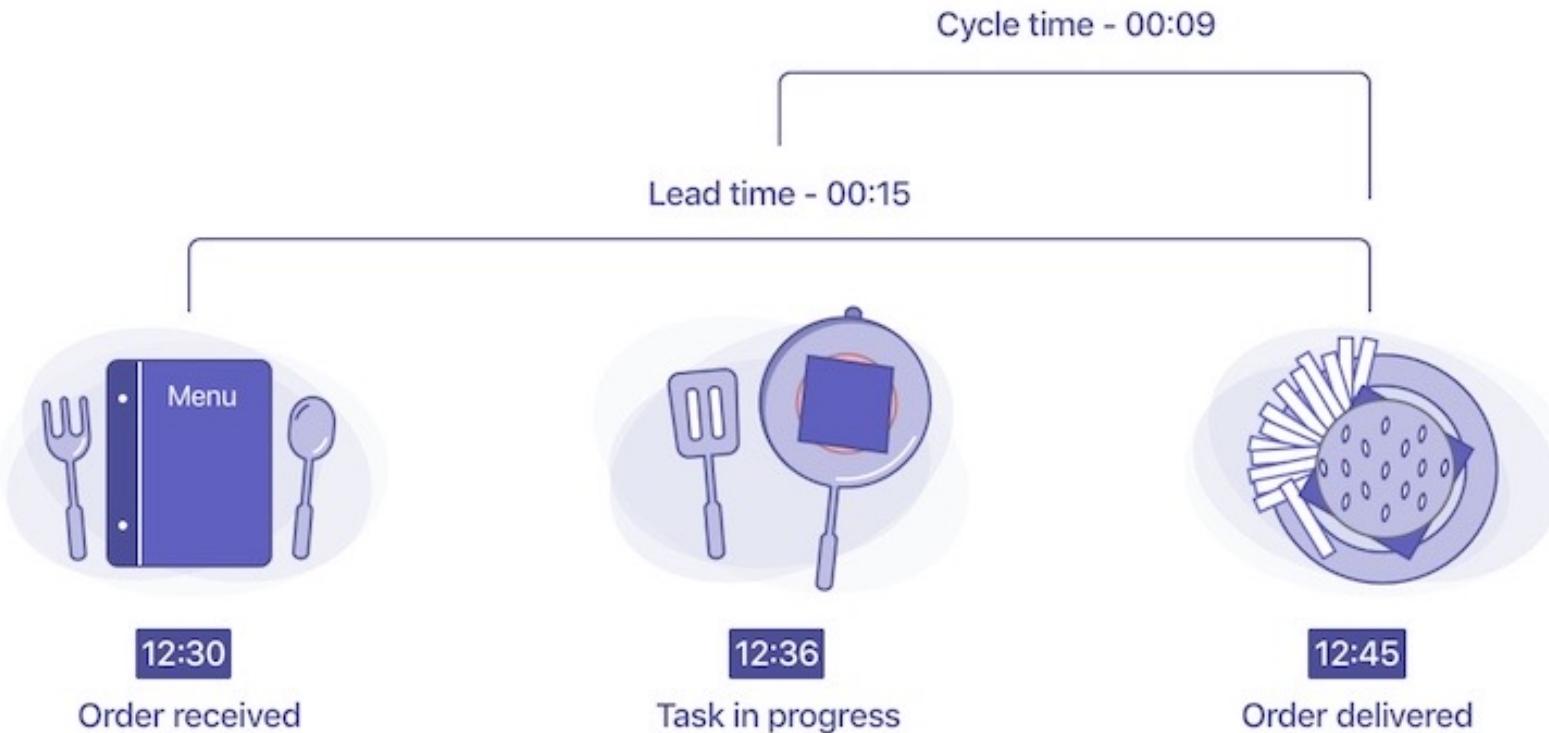
# Kanban



- No desenvolvimento de software, cada cartão contém uma pequena quantidade de trabalho a ser feita, geralmente, requisitos e suas respectivas tarefas desenvolvimento, testes, além do que deve ser implantado e disponibilizado para uso.

- Os cartões são alocados nos quadros Kanban, indicando em qual etapa(passo) de desenvolvimento a atividade está.
- A quantidade de tarefas em cada etapa(passo) do desenvolvimento indica o tamanho da fila.

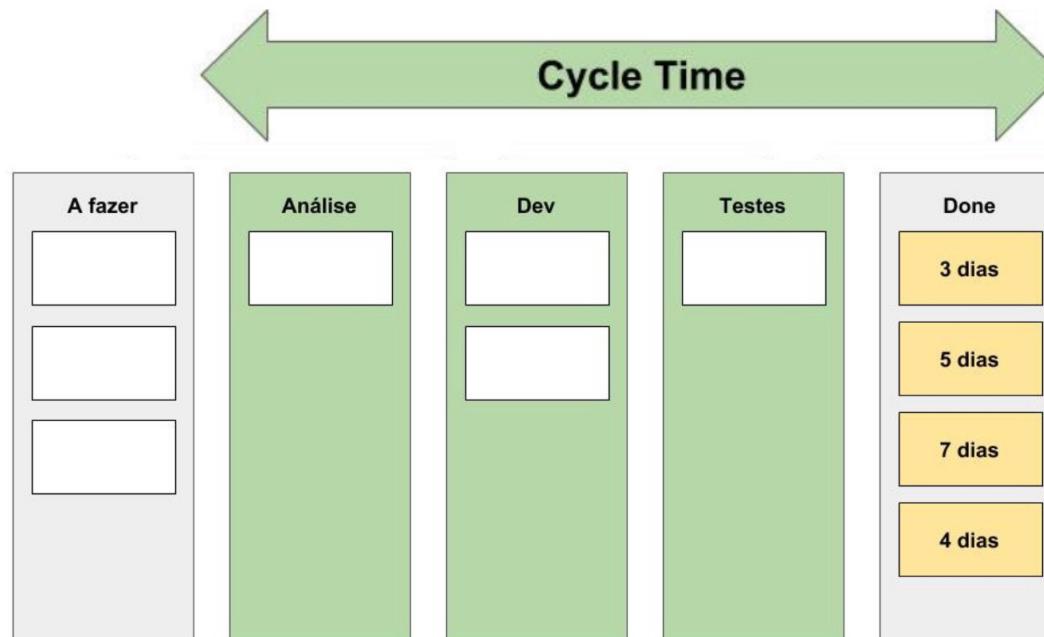
# Kanban – Lead Time X Cycle Time



Fonte imagem:<https://help.zenhub.com/support/solutions/articles/43000300345-use-control-charts-to-review-issue-cycle-lead-time>

# Kanban – Cycle Time

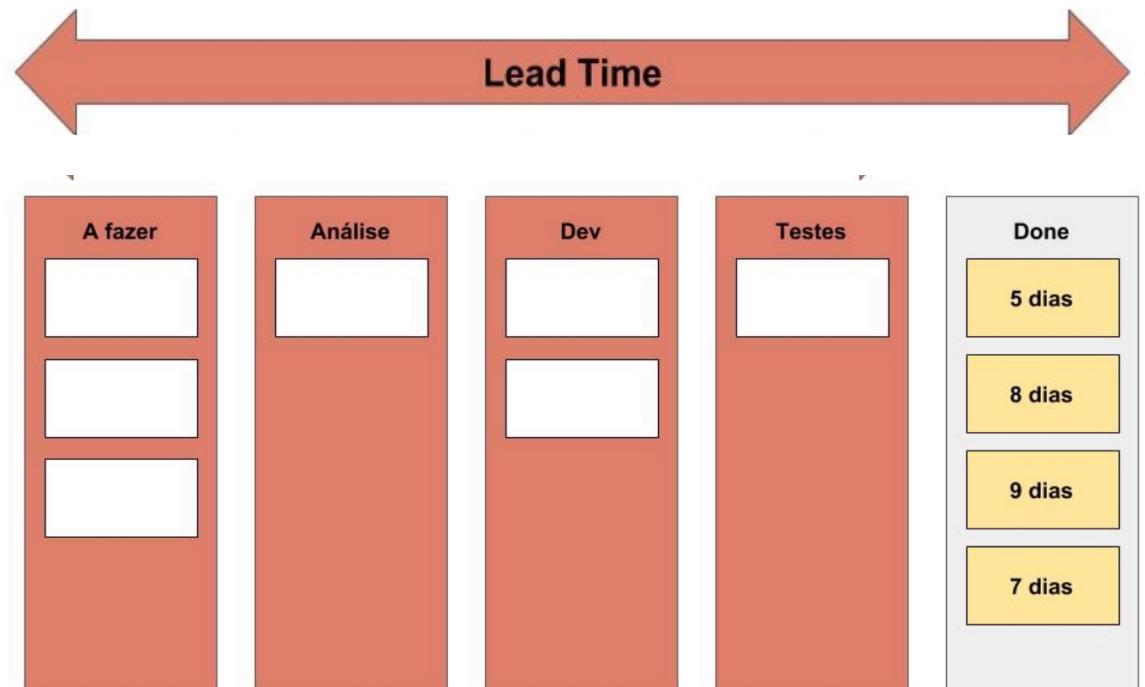
- O tempo de Ciclo (CT) é o tempo que um cartão(tarefa) leva para ser entregue a partir do momento em que começou a ser trabalhado.



Fonte imagem: <https://medium.com/@raphaelbatagini/kanban-e-suas-principais-m%C3%A9tricas-228d938326fb>

# Kanban – Lead Time

- O tempo de espera *lead time*- *LT*:
  - define quanto tempo, em média, uma tarefa vai ficar em cada etapa(passo) do Quadro Kanban
  - **Definição de Feito (DoD)** do Scrum!
- O LT inclui o tempo em fila - o tempo que a tarefa vai ficar na 2a sub-coluna dos passos do Quadro Kanban(**To-Do** no Scrum) aguardando ser puxada para o passo seguinte.



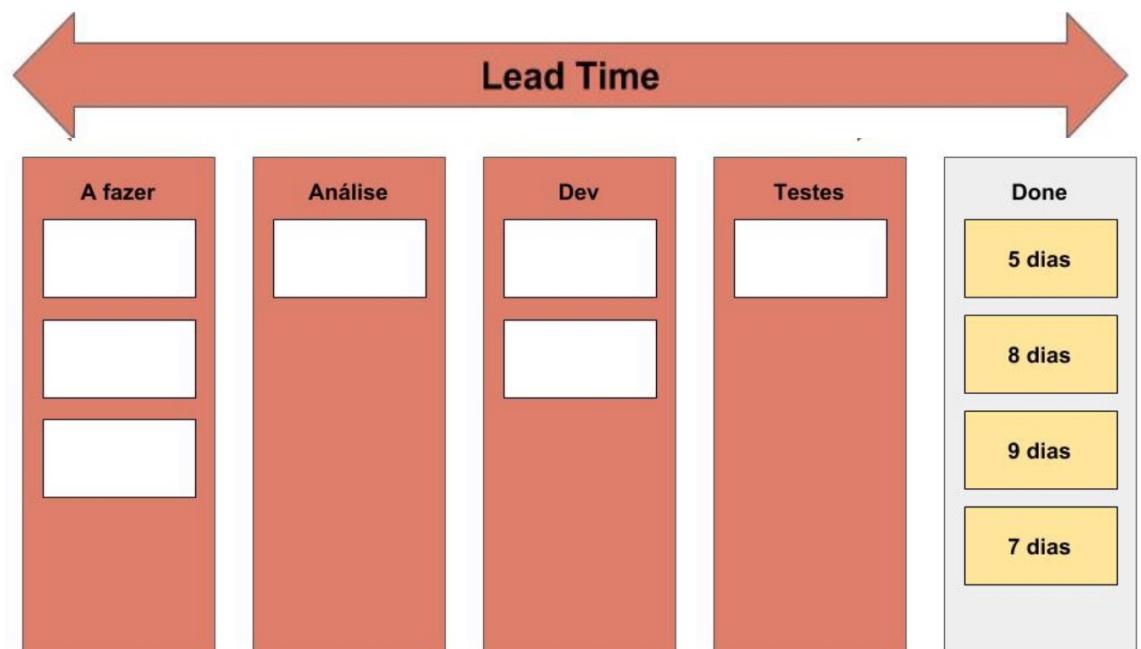
Fonte imagem: <https://medium.com/@raphaelbatagini/kanban-e-suas-principais-m%C3%A9tricas-228d938326fb>

Eric Brechner. 2015. Agile Project Management with Kanban (1st. ed.). Microsoft Press, USA.

# Kanban – Lead Time

- Essa estimativa considera uma tarefa de duração média, pois sabemos que vão existir tarefas mais complexas e mais simples.

- LT (análise) = 5 dias
- LT (implementação) = 12 dias
- LT (testes) = 6 dias
- LT (revisão)
- ...



Eric Brechner. 2015. Agile Project Management with Kanban (1st. ed.). Microsoft Press, USA.

# Kanban - Throughput

- A **Vazão** (*throughput-tp*) consiste em ver o **quanto de trabalho**, um time ou pessoa, consegue **entregar** em um período de tempo. E isso é percebido por meio da quantidade de cartões de tarefas relacionadas as histórias além, de atividades gerenciais,
  - quantidade de pessoas que atravessa a faixa de pedestre no tempo do semáforo
  - quantos litros de água uma cachoeira dá vazão por minuto
  - pontos por história ao longo do tempo ?!?!?

- Defina um período de tempo:
  - dentro deste período, quantos cartões/issues são concluídas, finalizadas.

Semana	1	2	3	4
Cards entregues	10	8	12	10



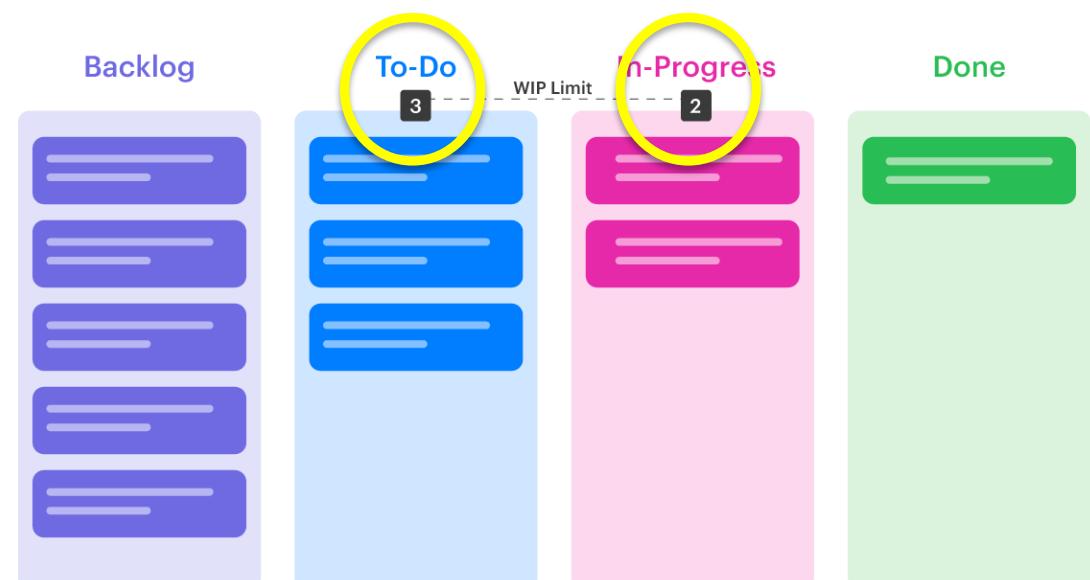
# Kanban - Throughput

$$\text{Throughput} = \frac{\text{qtde Cartões}}{\text{período Tempo}}$$

- Estime o *throughput* (**TP**) do passo com maior lead time
- No exemplo do livro (<https://engsoftmoderna.info/cap2.html>), e na maioria dos projetos de desenvolvimento de software, esse passo é a Implementação.
- Assim, suponha que o time seja capaz de sustentar a implementação de 8 **tarefas/cartões/issue** por mês.
  - **TP** (desse passo) é então:  $8 / 21(\text{úteis}) = 0.38 \text{ tarefas/dia.}$

# Kanban - WIP

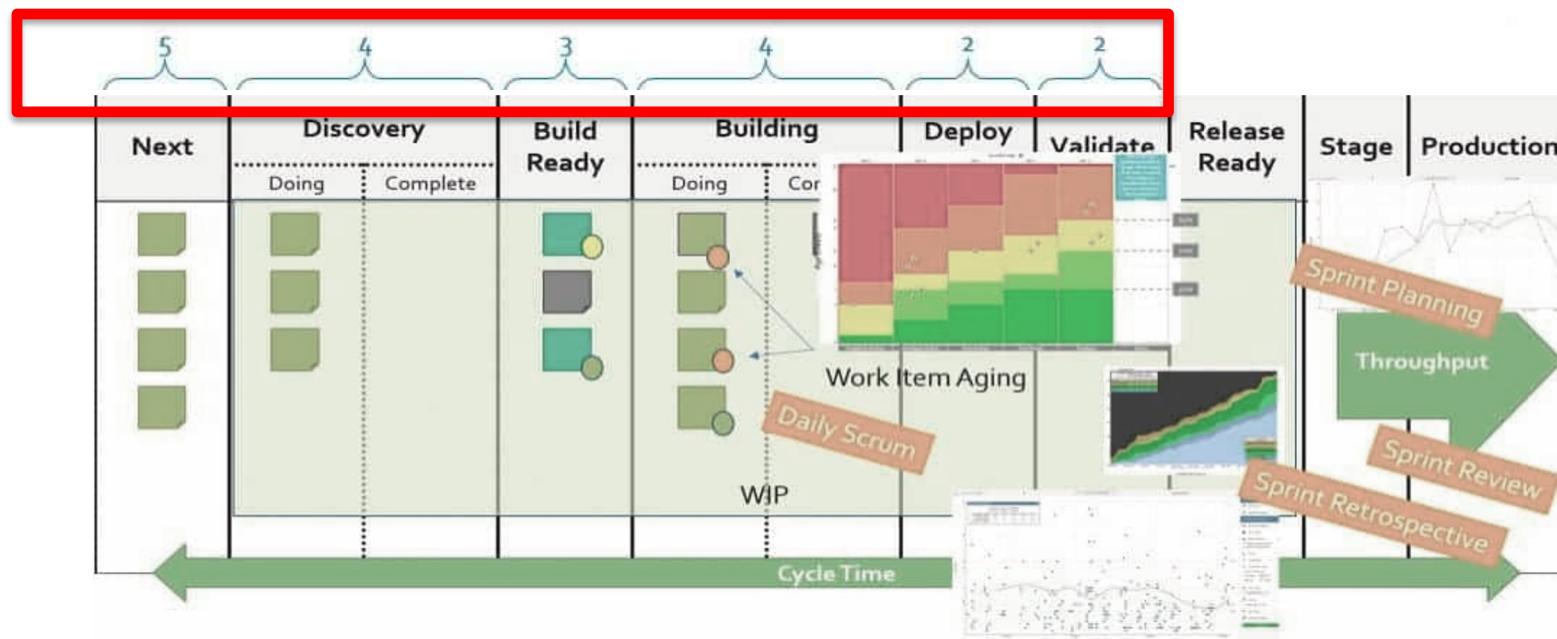
- Os sistemas Kanban foram idealizados de forma a limitar o estoque em produção.
  - (WIP - work in progress), pois quanto maior a quantidade de WIP, mais lento é o fluxo do sistema.
- O WIP deve ser limitado e uma nova tarefa só deve começar a ser executada quando existe capacidade disponível.
- Limites WIPs são o recurso oferecido no Kanban para garantir um ritmo de trabalho sustentável e a entrega de sistemas de software com qualidade.



[https://www.google.com/url?sa=i&url=https%3A%2F%2Fkissflow.com%2Fproject%2Fagile%2Fwip-limits-in-kanban%2F&psig=AOfVawOOh22UZFSjcAe\\_t02IN75X&ust=1650039366464000&source=images&cd=vfe&ved=2ahUKEwiikNnj-ZP3AhWfrZUCHX8CAZOQjRx6BAgAEAs](https://www.google.com/url?sa=i&url=https%3A%2F%2Fkissflow.com%2Fproject%2Fagile%2Fwip-limits-in-kanban%2F&psig=AOfVawOOh22UZFSjcAe_t02IN75X&ust=1650039366464000&source=images&cd=vfe&ved=2ahUKEwiikNnj-ZP3AhWfrZUCHX8CAZOQjRx6BAgAEAs)

# Kanban - WIP

- O papel desses limites é contribuir para que os desenvolvedores não fiquem sobrecarregados de tarefas e, consequentemente, propensos a baixar a qualidade de seu trabalho.



<https://www.scrum.org/resources/blog/4-key-flow-metrics-and-how-use-them-scrum-events>

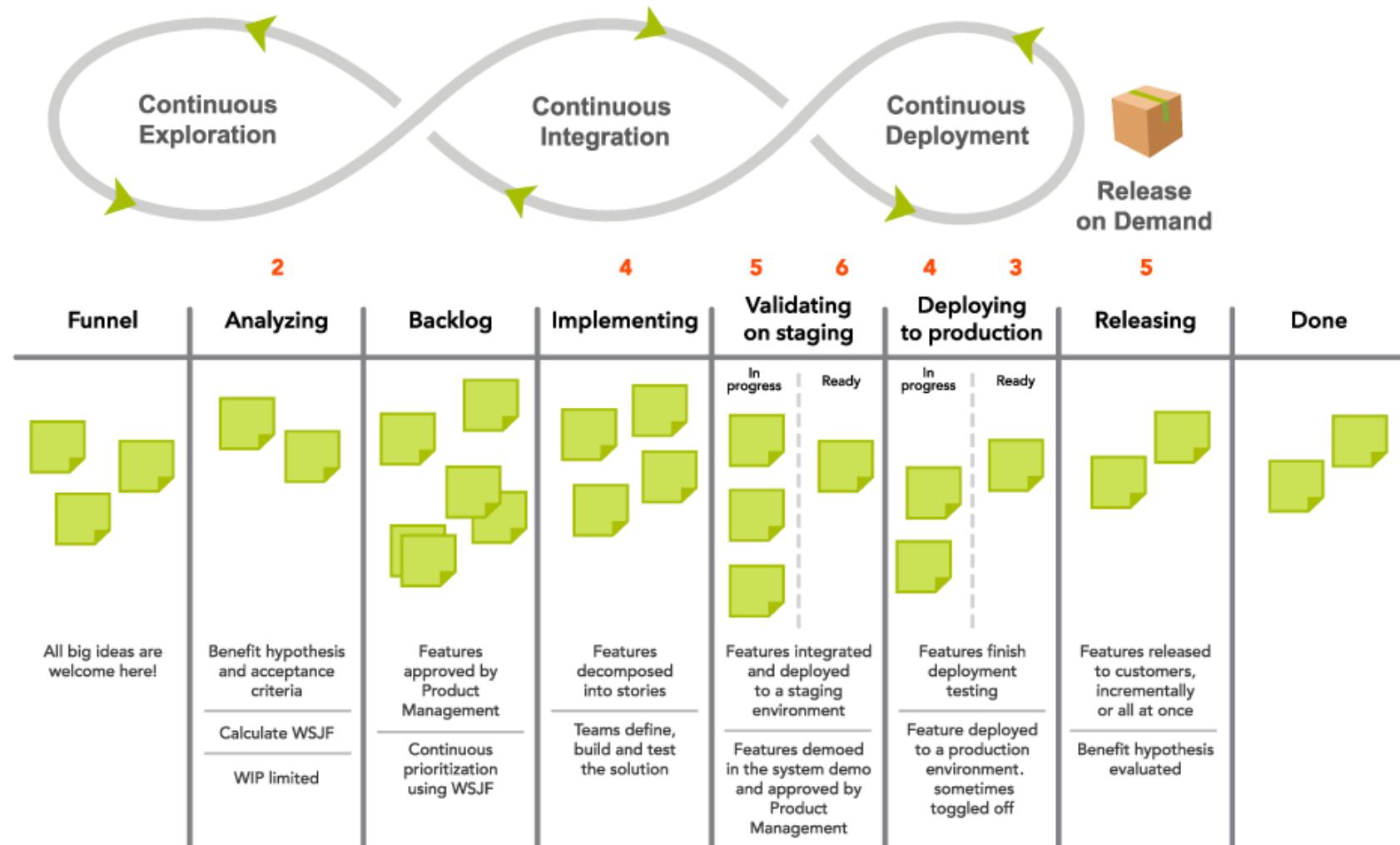
$$\text{WIP}(\text{passo}) = \text{TP} * \text{LT}(\text{passo})$$

# Kanban - WIP

Backlog	Especificação 2		Implementação 5		Revisão de Código 3	
<b>H3</b>	em espec.	especificadas <b>T8 T9 T10 T11 T12</b>	em implementação <b>T4 T5 T6 T7</b>	implementadas	em revisão <b>T3</b>	revisadas <b>T1 T2</b>

- Ainda segundo o exemplo do livro, teremos os seguintes resultados:
  - $WIP(\text{especificação}) = 0.38 * 5 = 1.9$
  - $WIP(\text{implementação}) = 0.38 * 12 = 4.57$
  - $WIP(\text{revisão}) = 0.38 * 6 = 2.29$
- Portanto, os resultados finais(arredondados para cima) ficam assim:
  - $WIP(\text{especificação}) = 2$
  - $WIP(\text{implementação}) = 5$
  - $WIP(\text{revisão}) = 3$
- Eric Brechner, sugere ainda, adicionar uma margem de erro de 50% nos WIPs calculados, para acomodar variações no tamanho das tarefas, tarefas bloqueadas devido a fatores externos, etc.

# Kanban



© Scaled Agile, Inc.

SAFE - <https://www.scaledagileframework.com/program-and-solution-kanbans/>

# Q-Rapids Throughput

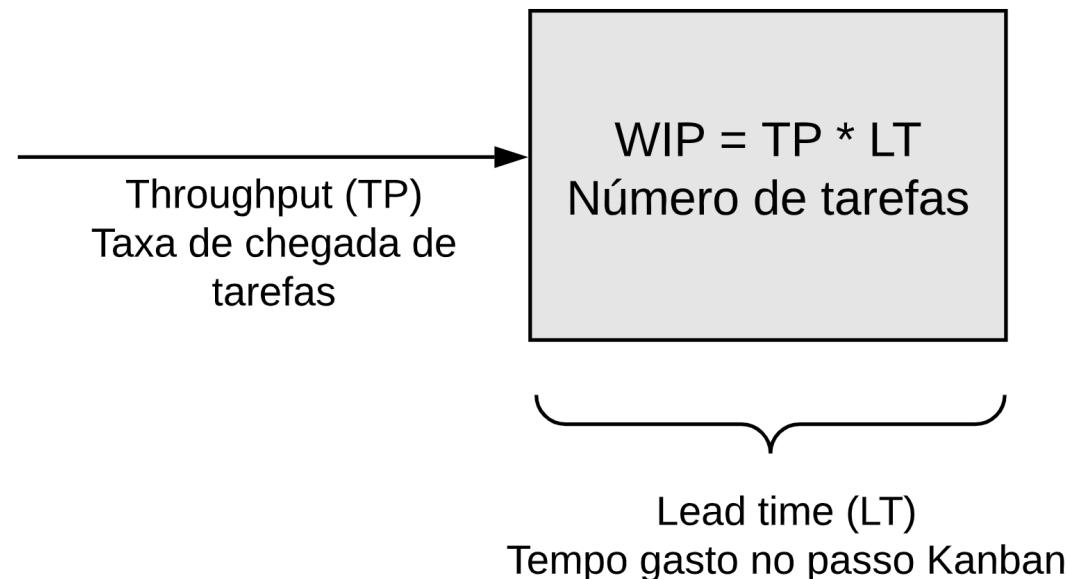
- Já conforme a definição no modelo(referência) de qualidade Q-Rapids, o throughput é observado de duas maneiras

Productivity	Issues' Velocity	Resolved Issues' Throughput	Density of issues whose resolution didn't take longer than the defined duration threshold	$= \frac{\text{Number of issues resolved under the duration threshold}}{\text{Total number of issues in a given timeframe}}$ <p style="text-align: center;">where the duration threshold is user defined</p>	Total number of issues with resolved status (e.g. sprint, week, month) and total issues in timeframe
		Issues-type in a timeframe	Density of issues of a specific type within a defined timeframe	$= \frac{\text{Number of issues of a specific type}}{\text{Total number of issues in a given timeframe}}$ <p style="text-align: center;">where type = resolved, open, task, bugs, etc.</p>	Total number of issues with type (e.g. open, closed, resolved, task, bugs ) and total issues in timeframe (sprint, week, month)
		Team Throughput	Density of issues resolved by a team in a given timeframe	$= \frac{\text{Number of resolved issues}}{\text{Total number of issues in a given timeframe}}$	Total number of issues (a.k.a. story points) with status resolved/completed and total issues in timeframe (sprint, week, month)

[https://www.q-rapids.eu/\\_files/ugd/d88d32\\_5923394075584d5185758549dc0fb21a.pdf](https://www.q-rapids.eu/_files/ugd/d88d32_5923394075584d5185758549dc0fb21a.pdf)

# A Lei de Little

- A Lei de Little diz que o número de itens em um sistema de filas é igual à taxa de chegada desses itens multiplicado pelo tempo que cada item fica no sistema.
  - Traduzindo para o nosso contexto, o sistema é um passo de um processo Kanban e os itens são tarefas.

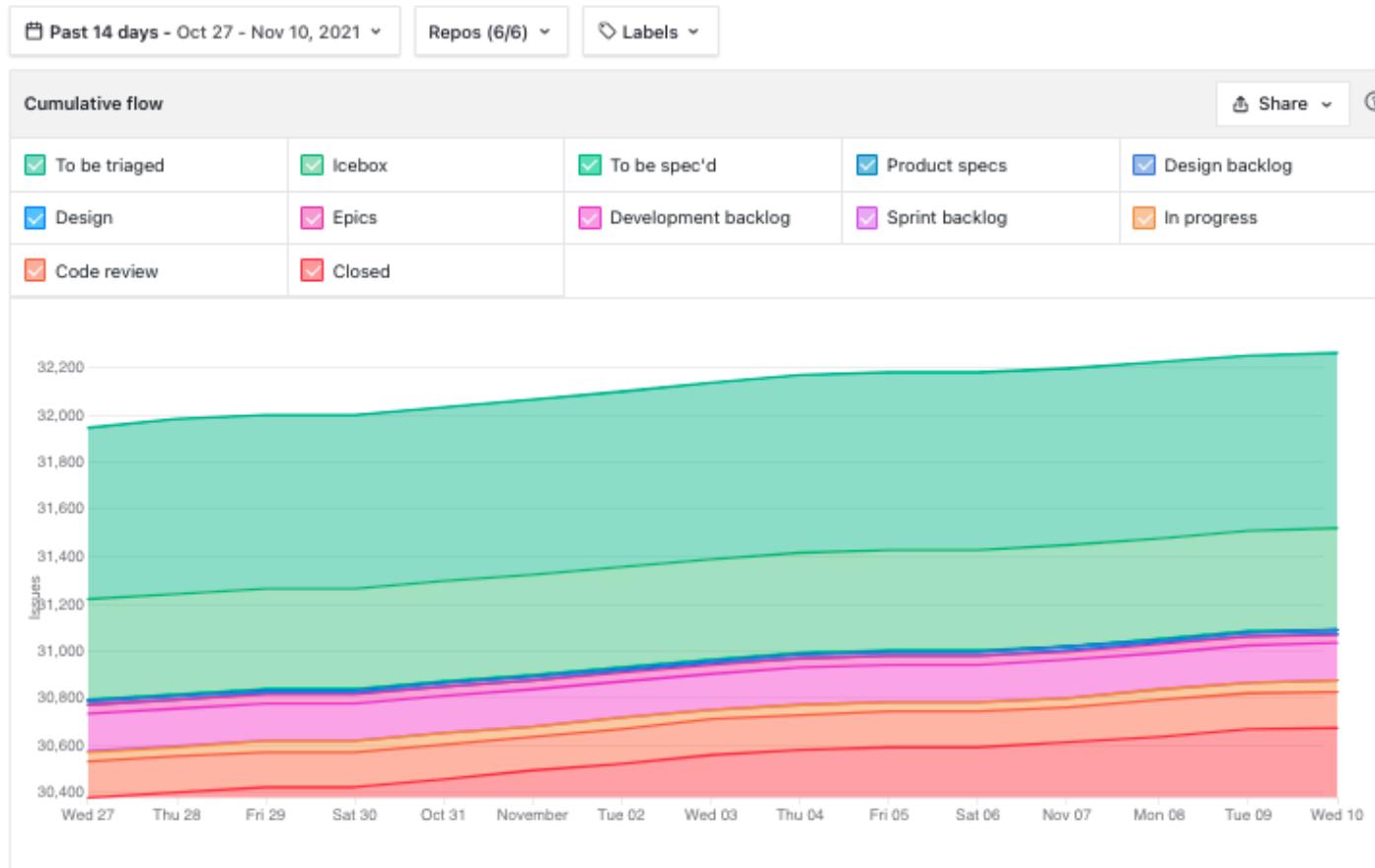


Lei de Little:  $WIP = TP * LT$

# Algumas percepções via zenhub

## Throughput

- Os diagramas de fluxo cumulativos são utilizados como visualização da movimentação de *issues*(cartões) em cada pipeline/passo/raia, ao longo do tempo.



# Algumas percepções via zenhub

## Throughput

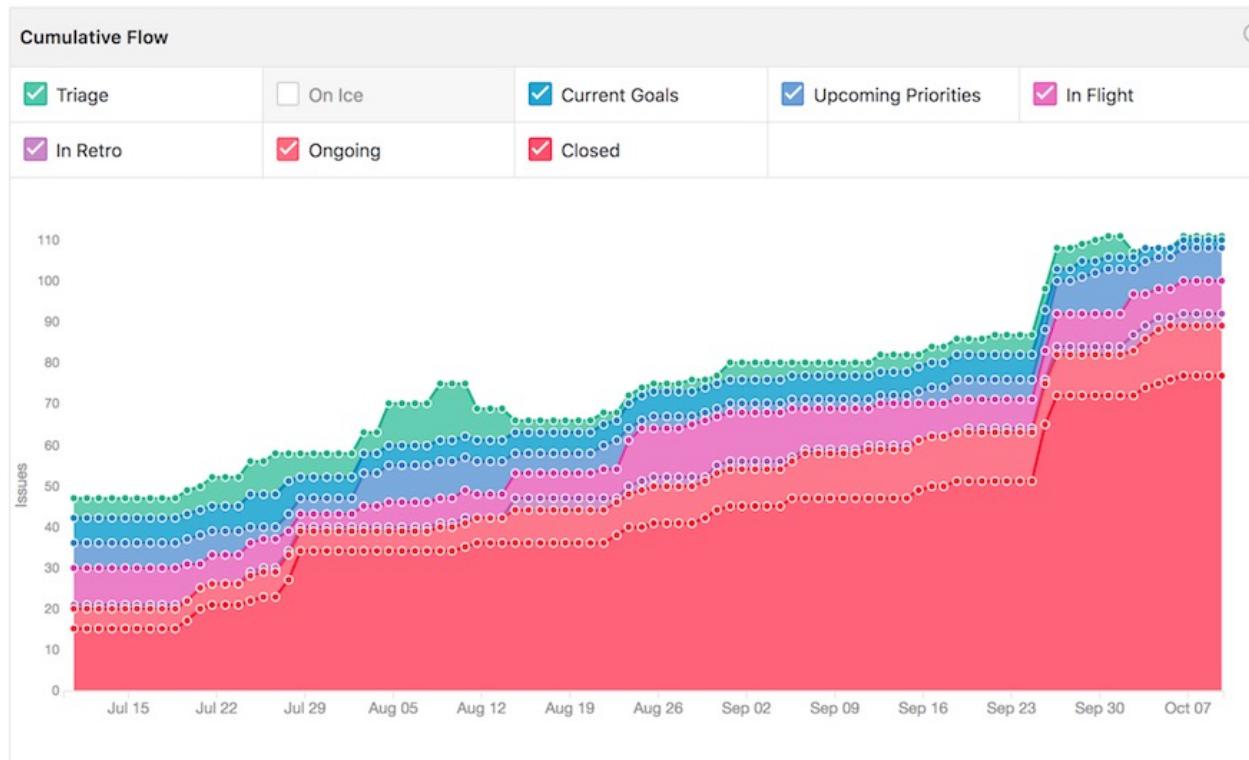
- Além dos diagramas a quantidade de *issues/cartões* em cada pipeline/passo/raia, também é exibido de forma tabular

Issues in pipeline	November 7th	November 8th	November 9th	November 10th
To be triaged	748	742	743	741
Icebox	426	426	426	428
To be spec'd	5	5	5	5
Product specs	0	0	0	0
Design backlog	15	15	15	15
Design	2	2	2	2
Epic	37	37	37	37
Development backlog	160	155	156	157
Sprint backlog	1	1	1	1
In progress	39	41	46	47
Code review	149	159	151	151
Closed	30,612	30,637	30,669	30,676

# Algumas percepções via zenhub

## Throughput

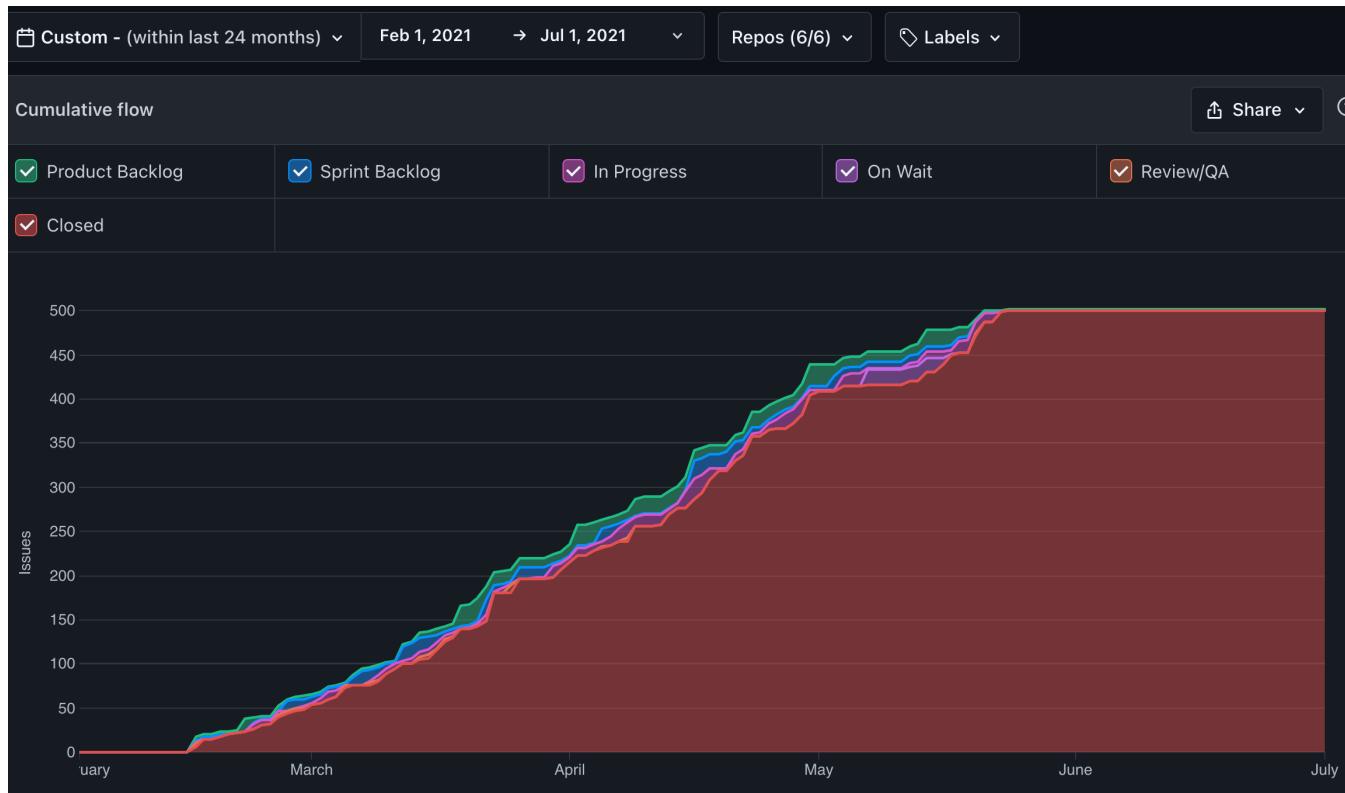
- O pipeline *closed* deve estar continuamente em crescimento
- Uma inclinação ascendente para a direita com faixas dos pipelines uniformes, indicam bom balanceamento.
- Por exemplo, se seu e backlog do sprint está crescendo, mas o que está em andamento não, isso significa que você está aceitando mais trabalho em seu sprint, sem entregar, finalizar.



# Algumas percepções via zenhub

## Throughput

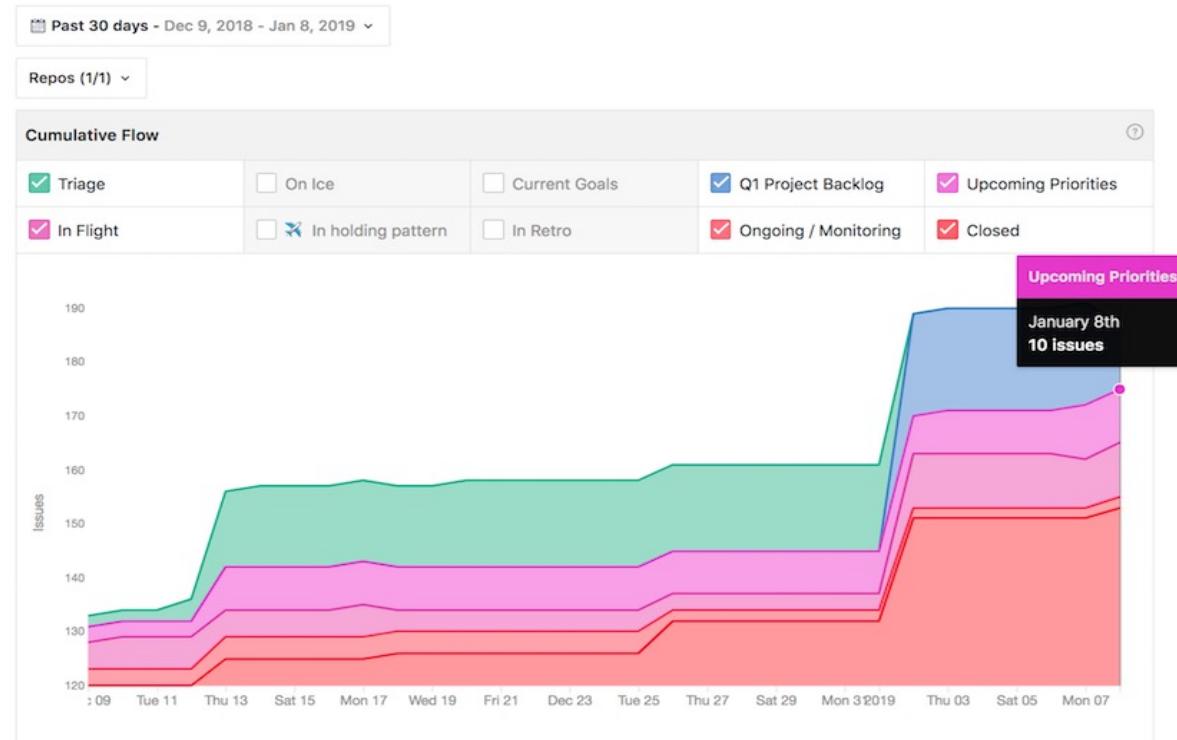
- Ao longo do tempo, os pipelines devem se “encontrar” com o pipeline *closed*



# Algumas percepções via zenhub

## Throughput

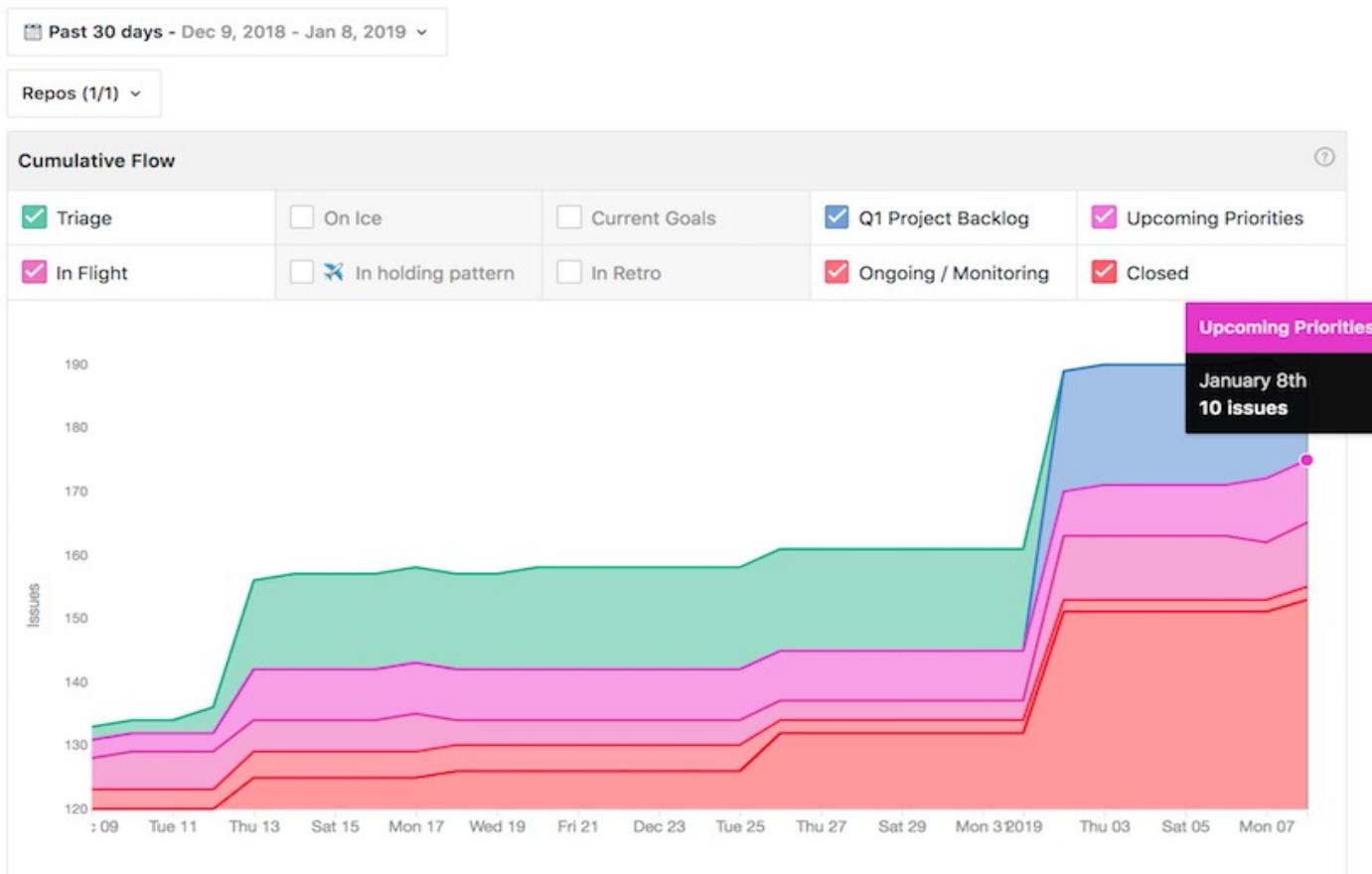
- Analisando o gráfico abaixo é possível observar, por exemplo:
  - possível indicação de gargalo já que, a maioria dos pipelines não apresentam variação durante a maior parte do tempo. Tarefas muito complexas? Trabalhando em muitas issues?
- Variação significativa observada no final do período
  - fechando muitas issues no mesmo dia?
  - esperando a conclusão de um sprint?
  - esperando uma release estar pronta!?



# Algumas percepções via zenhub

## Throughput

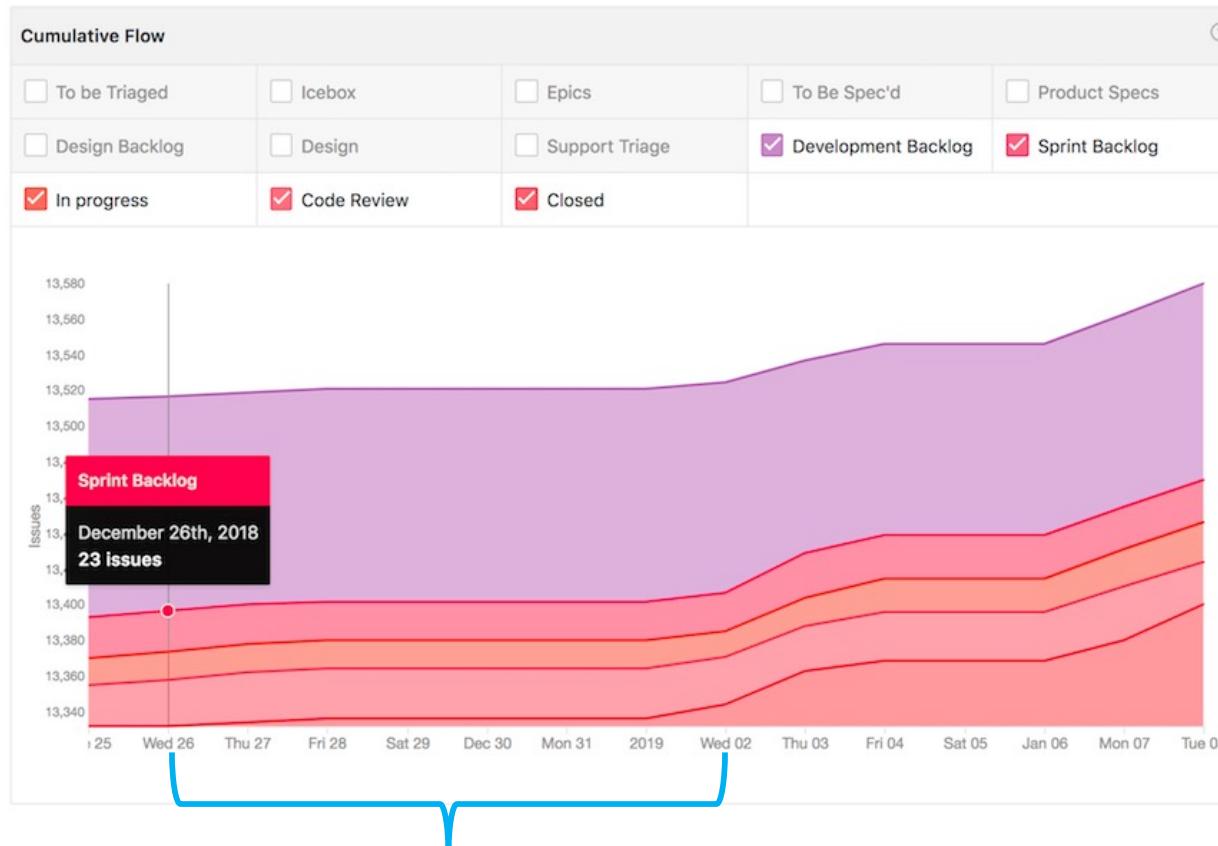
- Quando os pipelines permanecem com a mesma largura, significa que o trabalho não está ficando acumulado em nenhum pipeline específico



# Algumas percepções via zenhub

## Throughput

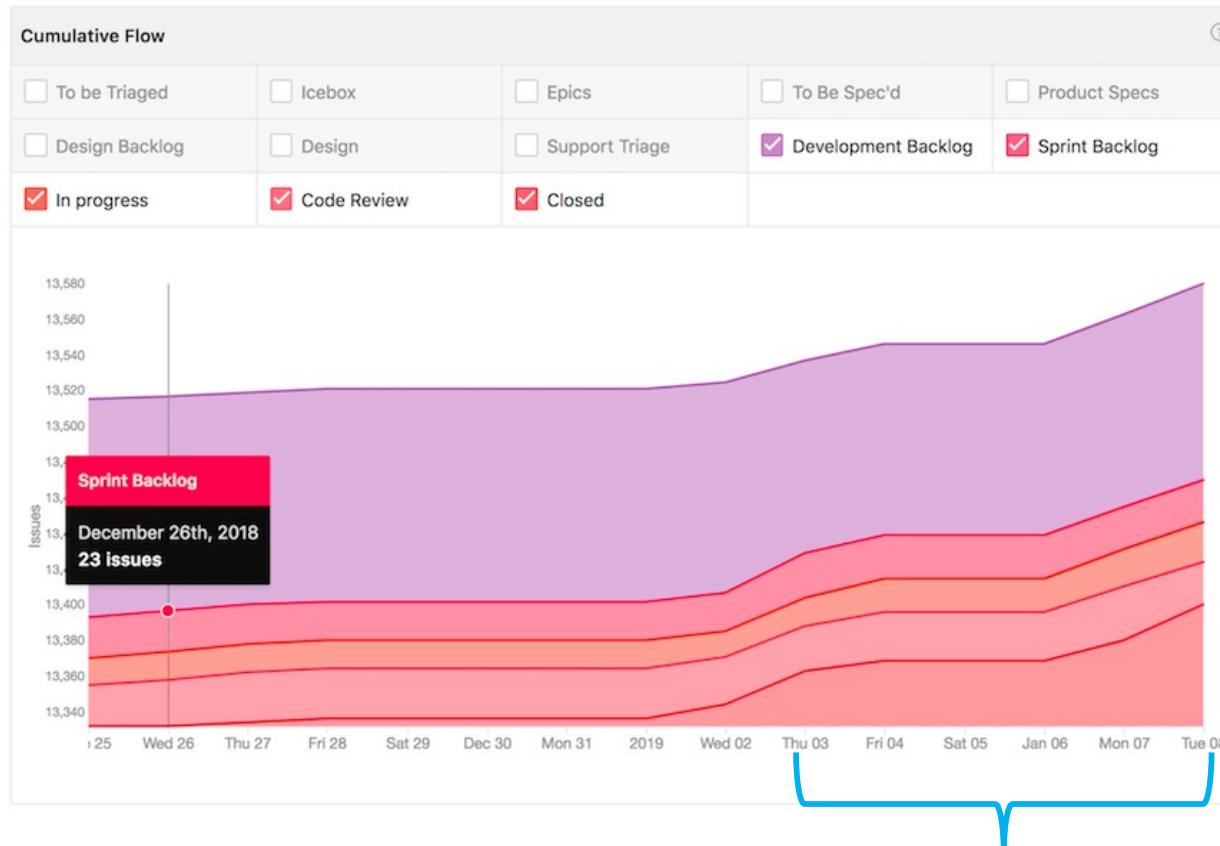
- Do dia 26 ao dia 2, o pipeline do sprint backlog não varia, e isso pode significar:
  - que o trabalho não ficou pronto durante o sprint?
  - você assumiu issues/cartões maiores do que o tamanho de um sprint?



# Algumas percepções via zenhub

## Throughput

- Já do dia 3 ao dia 8, uma curva ascendente e o pipeline closed está ficando maior.
  - Isso é positivo e indica que o trabalho está sendo concluído, ficando pronto.



# Algumas percepções via zenhub

## Lead Time X Cycle Time

- **Lead Time:**
  - o pipeline inicial que captura o momento em que a issue foi criada
  - até o pipeline final que captura o momento em que o trabalho é **concluído, finalizado.**
  - Isso reforça a importância da definição no projeto do conceito **DoD** e o se essa definição escolhida por seu time coincide com o pipeline *closed* ou não.  
Lembrando que no scrum **feito = software em produção!**
  - o intervalo de pipelines **New Issue** → **Closed** costuma ser o mais comum para a maioria das equipes.
- **Cycle Time:**
  - o pipeline inicial que captura quando um cartão/issue de trabalho realmente começa.
  - até o pipeline final que captura o momento em que o trabalho é concluído
  - o intervalo de pipelines **In Progress** → **Closed** costuma ser o mais comum para a maioria das equipes.

# Algumas percepções via zenhub

## Lead Time X Cycle Time

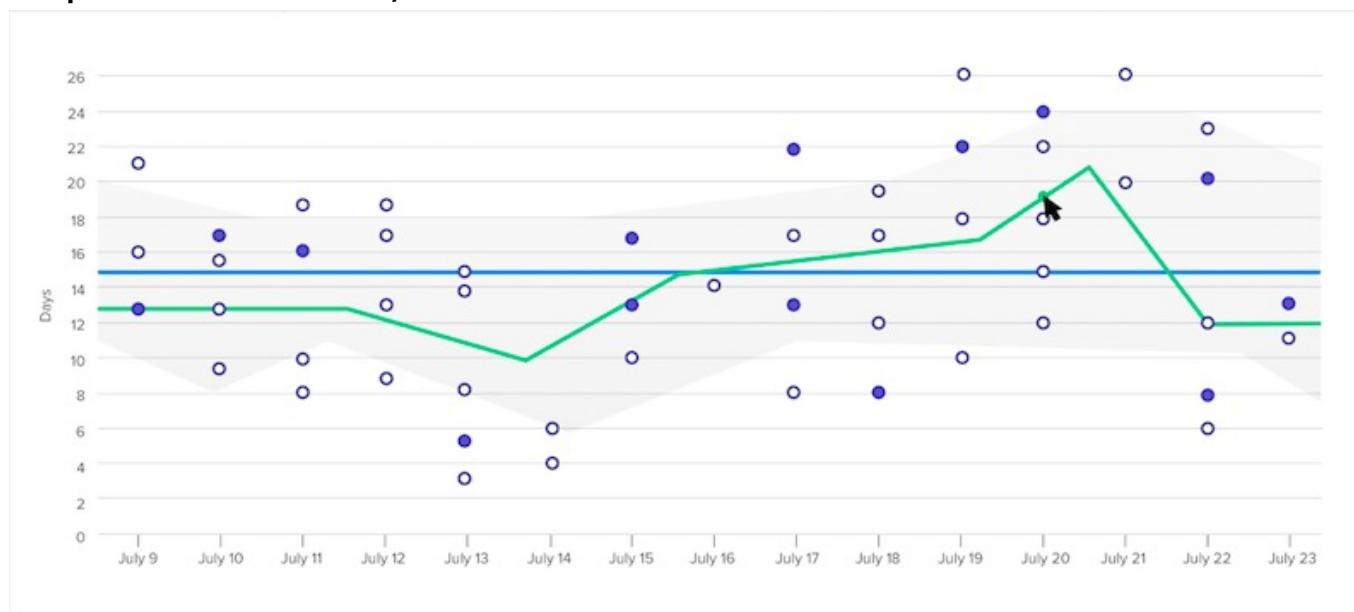
- São visualizados pelo gráfico de controle-CCR:
  - eixo X representa a data que a issue foi fechada e o eixo Y, a quantidade de dias para que a issue fosse fechada.
  - cada ponto representa uma issue, como observado no gráfico abaixo
  - um grupo de issues fechadas no mesmo dia é exibido como ●
  - linha azul, média de dias para uma issue ser fechada
  - linha verde, média móvel dos últimos dias (?!)
  - área em cinza, o desvio padrão observado no período, considerando a média móvel
  - os pontos fora da área do desvio padrão representam os *outliers*.



# Algumas percepções via zenhub

## Lead Time X Cycle Time

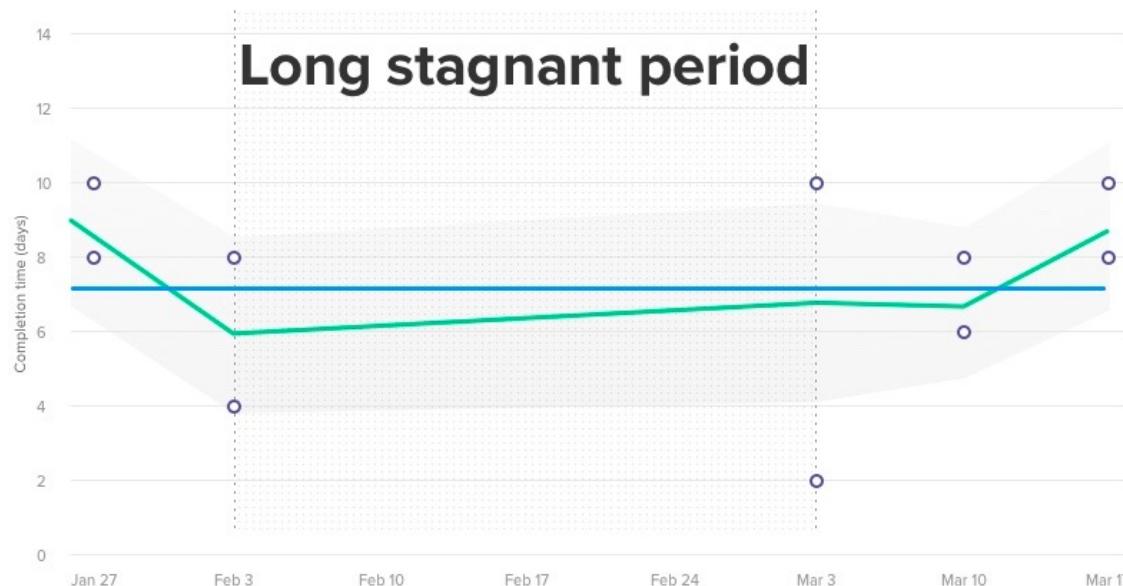
- Situação desejada, ideal:
  - variação limitada - quanto menor for a área em cinza, o desvio padrão, mais estável é o seu processo. Menos variação significa melhor fidedignidade, confiança nas suas estimativas e entregas
  - estabilidade na média móvel: a linha da média móvel (verde) deve ter um comportamento sem muitas variações. Por exemplo, as alterações na média móvel podem indicar que os cartões/isses estão demorando mais para serem fechados, mas também podem indicar que os problemas estão ficando mais complexos. Portanto, conversem entre si!



# Algumas percepções via zenhub

## Lead Time X Cycle Time

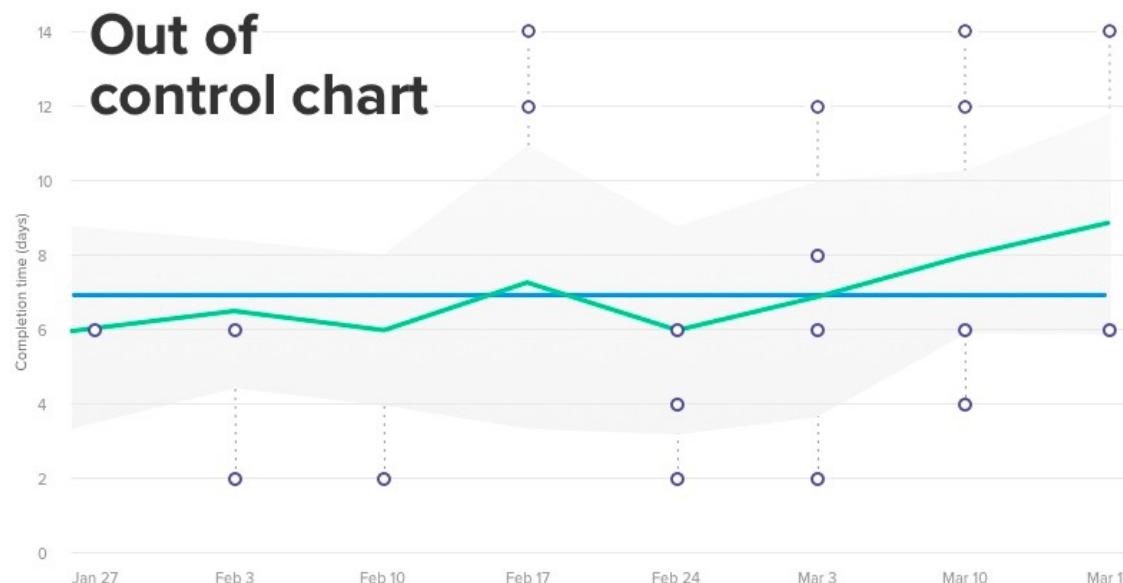
- O Fluxo de trabalho estagnado, é percebido por longo período de tempo em que os cartões/issues não são concluídas:
  - tarefas muito complexas?
  - havia um conjunto de issues que estavam acumuladas e que foram encerradas no mesmo dia?
  - que mudanças podem ser feitas para distribuir o trabalho a ser entregue de forma mais uniforme?



# Algumas percepções via zenhub

## Lead Time X Cycle Time

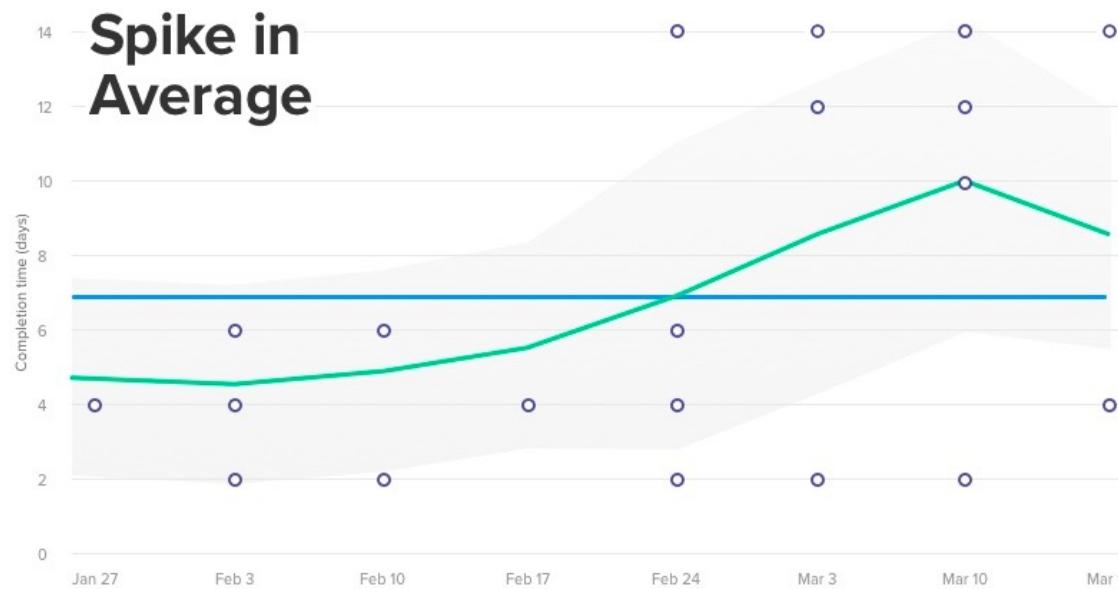
- Lead Time e Cycle Time imprevisíveis - um número significativo de *outliers* (pontos acima ou abaixo da área em cinza) indica que o um processo está “fora de controle”.
  - isso significa que há uma grande variação em quanto tempo leva para fechar os cartões/issues
  - inviabiliza qualquer previsão
  - Por que há tanta variabilidade?



# Algumas percepções via zenhub

## Lead Time X Cycle Time

- Gargalos em seu fluxo de trabalho: o pico na média móvel(verde) pode ser um alerta antecipado sobre problemas no desenvolvimento.
  - mesmo que o tempo médio geral pareça aceitável, um pico na média móvel não deve ser ignorado.
  - o quê aconteceu no período?
  - o que possivelmente influenciou esse pico?



# Referências

- Marco Túlio Valente. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade, 2020, disponível em:  
<https://engsoftmoderna.info/cap2.html>
- KATAYAMA, Eduardo Teruo. A contribuição da indústria da manufatura no desenvolvimento de software
- Claudia O. M; Goldman, A. Katayama E. T.; Uma introdução ao Desenvolvimento de Software Lean; Simpósio Brasileiro de Qualidade de Software, 2012
- Taiichi Ohno. Toyota Production System: Beyond Large-Scale Production. Productivity Press, 1998

# Referências

- Mary Poppendieck and Tom Poppendieck. Lean Software Development: An Agile Toolkit for Software Development Managers. Addison-Wesley Professional, 2003
- Imagens lead time, cycle time, throughput e WIP, disponível em: <https://medium.com/@raphaelbatagini/kanban-e-suas-principais-m%C3%A9tricas-228d938326fb>
- Visualizando o throughput de issues no zenhub:  
<https://help.zenhub.com/support/solutions/articles/43000038850>
- Visualizando led-time e cycle time no zenhub:  
<https://help.zenhub.com/support/solutions/articles/4300030345-use-control-charts-to-review-issue-cycle-lead-time>