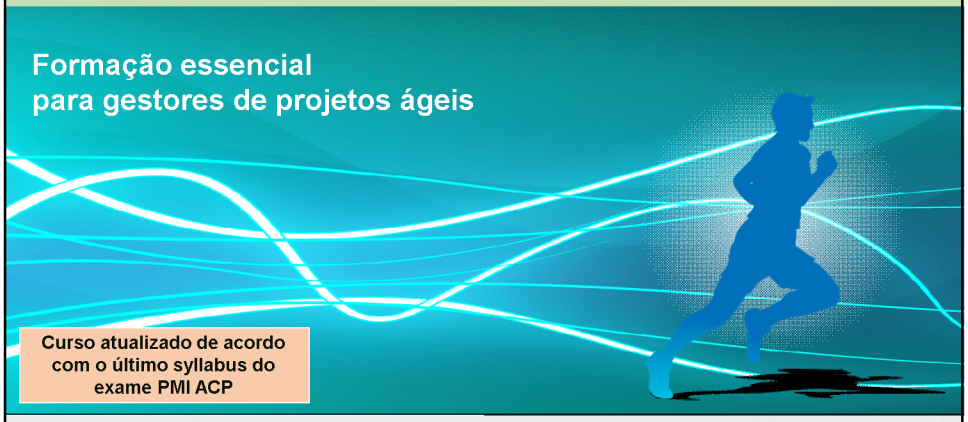
TI.exames

EDUCAÇÃO E TREINAMENTO
PARA ALAVANCAR A SUA CARREIRA

# Formação em Gerenciamento Ágil de Projetos

Preparatório para o exame PMI Agile Certified Practitioner (PMI ACP)®

## Formação essencial para gestores de projetos ágeis



**Curso atualizado de acordo com o último syllabus do exame PMI ACP**



Versão: 2.0 Liberação: 10/03/17

## Aviso de marcas registradas e direitos autorais

- Todos os direitos reservados. Nenhuma parte deste material poderá ser reproduzida ou transmitida em qualquer ou por qualquer meio sem a permissão escrita da TIEXAMES Consultoria e Treinamento Ltda.
- A TIEXAMES não licencia o uso de seu material para outras empresas. Se você encontrar outra empresa utilizando este material ou parte dele em treinamentos, por favor, denuncie pelo e-mail contato@tiexames.com.br.
- Algumas marcas registradas podem aparecer no decorrer deste curso. O uso destas marcas e logotipos é apenas para fins editoriais, em benefício exclusivo do proprietário da marca registrada, sem intenção de infringir as regras de sua utilização.

## Módulo 6

# Execução de projetos ágeis

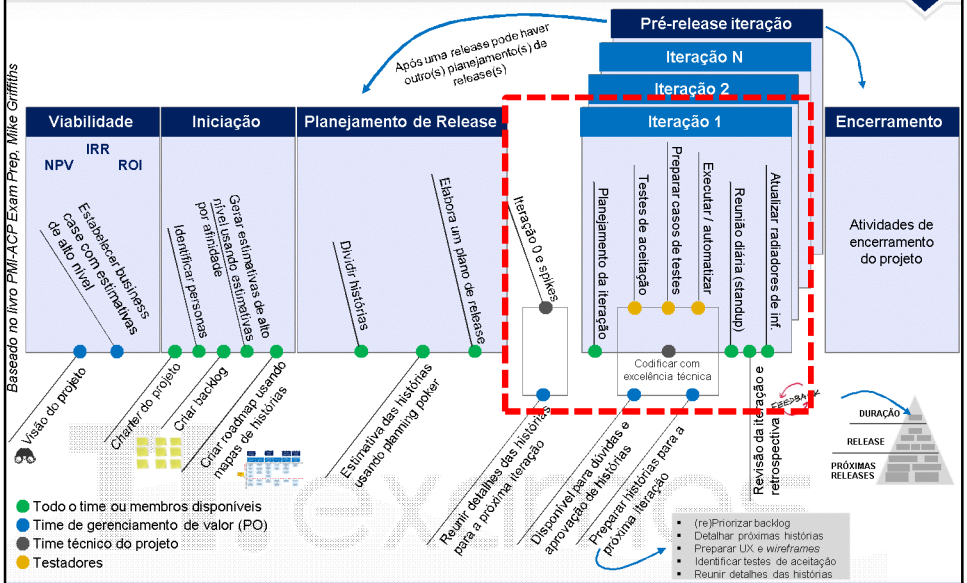
**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas



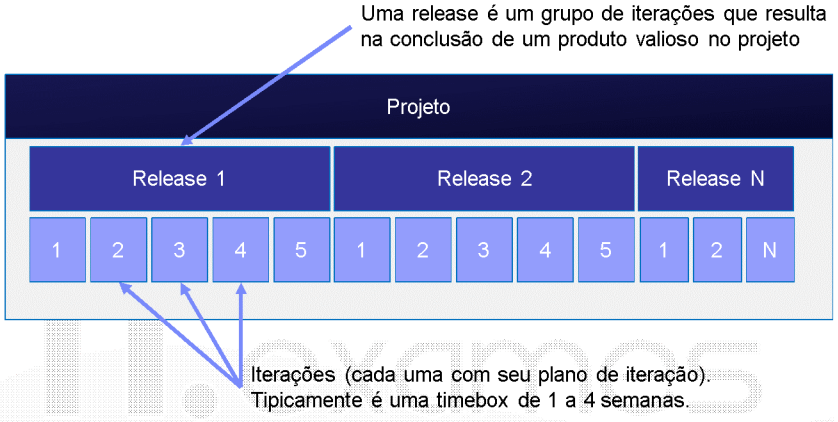


## Onde estamos?



## Projeto, releases e iterações

- Os projetos ágeis são divididos em releases (uma ou mais).
- As releases são divididas em iterações.

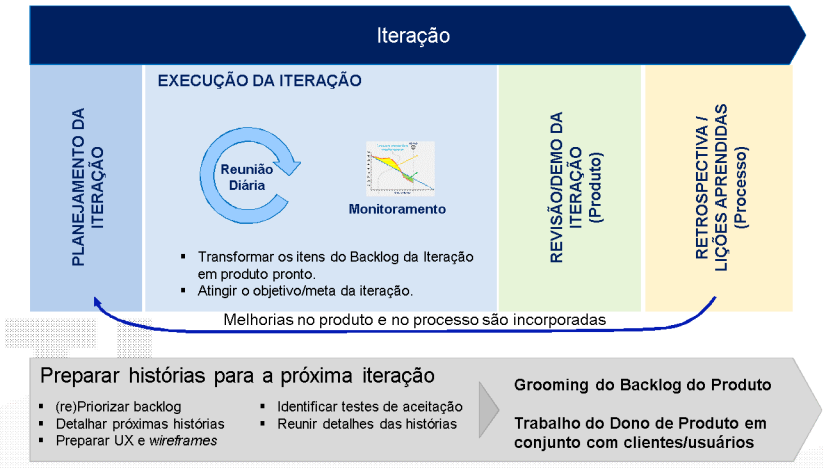






## Por dentro da iteração

- A execução de um projeto ágil se difere da de projetos tradicionais. Planejamento, execução, inspeção do produto e do processo ocorrem ao longo de toda a execução.



## Módulo 6

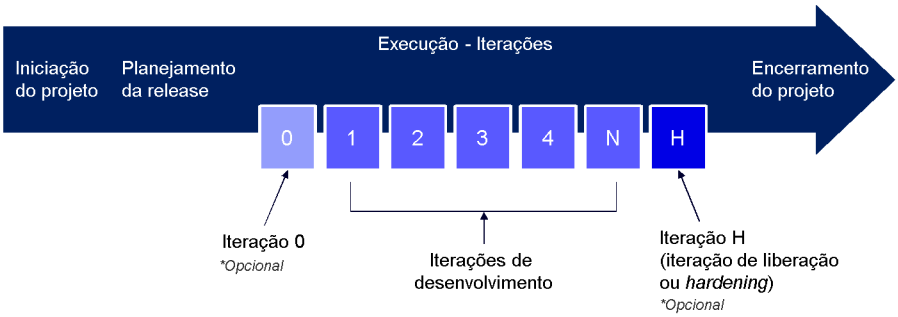
# Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas




## Tipos de iterações




**Iteração 0**

É uma iteração opcional que pode ser usada para preparar o cenário para os esforços de desenvolvimento. Por definição, a iteração 0 geralmente não envolve a construção de quaisquer entregas para o cliente.

**Iteração H**

O trabalho realizado nesta iteração pode incluir a estabilização do código (produto), a documentação do produto, a compilação dos conjuntos finais ou a conclusão de testes adicionais. Se um projeto tiver várias versões, talvez seja necessário programar uma iteração de endurecimento (hardening) antes de cada versão.

## Módulo 6

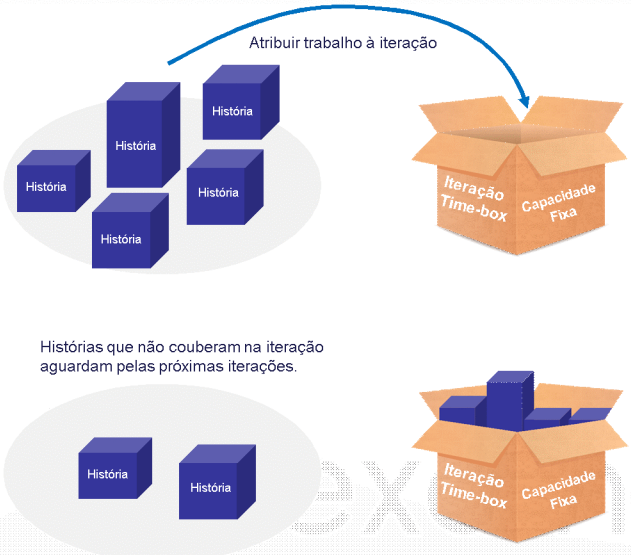
# Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas




## Time-box




- Uma time-box é um período de tempo com duração pequena e fixa no qual uma lista de atividades predefinidas é executada.
- No momento que o tempo se encerra, o evento está automaticamente finalizado. Ou seja, não pode ser estendido.
- As time-boxes proveem *checkpoints* (inspeções) frequentes para que o time possa calibrar o seu progresso e replanejar as abordagens que estão sendo utilizadas.

## Síndrome do estudante e lei de Parkinson

- **A síndrome do estudante**
  - → Significa adiar o trabalho até o último momento possível.
  - → Outras tarefas pressionam mais e, portanto, você atrasa o início de uma determinada tarefa até o último momento para lhe dar tempo de completar outro trabalho mais importante, muito provavelmente também sendo completado no último momento.
- **A lei de Parkinson**
  - → Contribui negativamente fazendo com que as atividades se estendam por todo o tempo dado para a execução das atividades.

**As time-boxes criam uma tensão nas datas de entrega diminuindo os efeitos.**





## Módulo 6

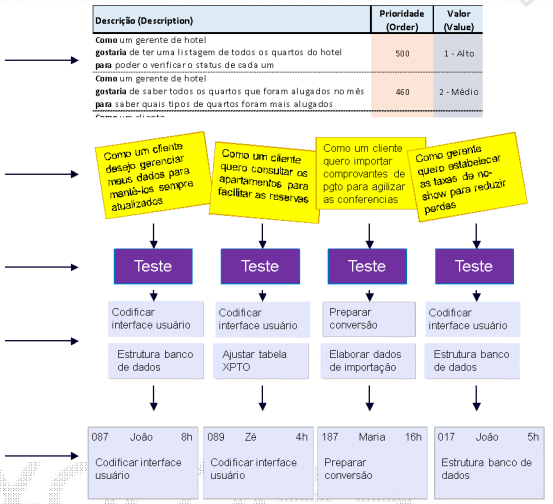
## Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas

## Planejamento da iteração

1. Debater (dono de produto e time de entrega) sobre as histórias de usuário mais prioritárias que estão no backlog do produto e qualquer dependência.
2. Time de entrega seleciona as histórias de usuário para a iteração (de acordo com a prioridade das histórias e sua capacidade).
3. Time de entrega define os critérios de aceitação e escreve / refina os testes de aceitação para cada história (em conjunto com dono de produto).
4. Time de entrega decompõe as histórias de usuário em tarefas.
4. Time de entrega estima cada tarefa (utilizando dias/horas ideais). Verifica se há capacidade para entregar todos os itens.







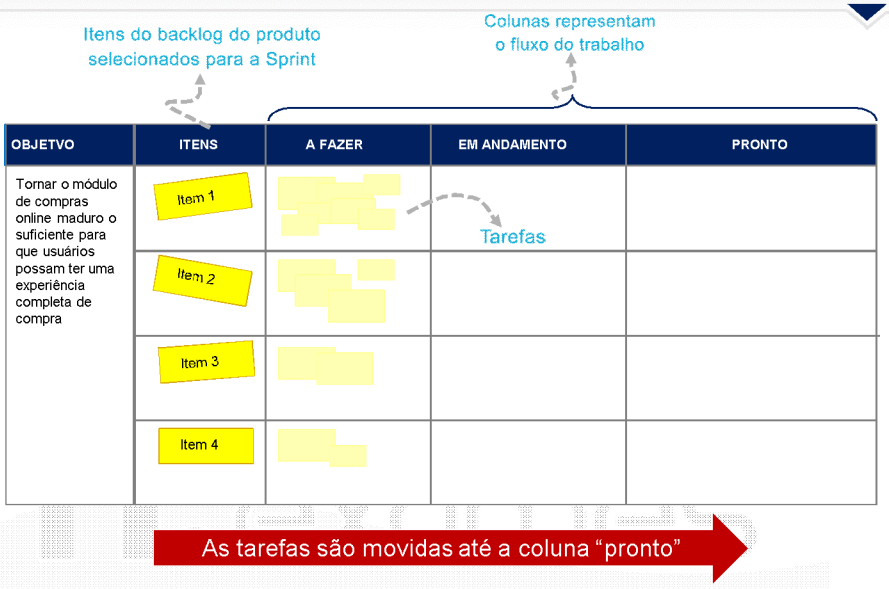
## Backlog da iteração (ou sprint)

- Os itens selecionados do Backlog do Produto, mais as tarefas necessárias para transformá-los em incremento software "pronto" e potencialmente usável, é chamado de **Backlog da Iteração (ou Sprint).**
- É um artefato vivo que nasce durante a reunião de planejamento da iteração, mas perdura por toda a iteração.

| Item do Backlog do Produto (1) | Valor | Pontos | Prioridade | Tarefas (2) | Estimativa (horas) | Horas Restantes | Executado por |
|---|---|---|---|---|---|---|---|
| Como um cliente registrado, eu quero solicitar on-line o aumento do meu limite de crédito para comprar mais | 50 | 21 | Crítico | Atualizar a tabela de Conta de Crédito com 2 campos a mais | 2 | 2 | Em aberto |
| | | | | para entrada e validação da solicitaçao | 8 | 8 | Em aberto |
| | | | | Integrar a tela com tabalea do banco por meio do serviço existente | 6 | 6 | Em aberto |
| | | | | Executar testes conforme critérios de aceitação | 8 | 8 | Em aberto |
| Como um cliente registrado, eu quero solicitar a partir do meu celular o aumento do meu limite de crédito | 30 | 13 | Importante | Refatorar a tabela do banco para adicionar a origem da solicitação | 1 | 1 | Em aberto |
| | | | | Adicionar uma tela no aplicativo mobile para entrada e validação | 4 | 4 | Em aberto |
| | | | | com as camadas de serviços existentes | 4 | 4 | Em aberto |
| | | | | Executar testes conforme critérios de aceitação | 1 | 1 | Em aberto |
| | | | | Executar testes de integração | 6 | 6 | Em aberto |

**É a soma dos:** (1) itens selecionados para a iteração (sprint) (2) tarefas necessárias para desenvolver os itens

## Backlog da iteração - utilizando o quadro de tarefas







## Módulo 6

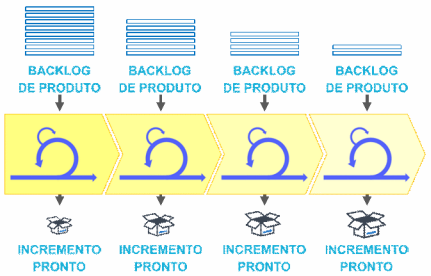
## Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas

## “Pronto”, um conceito-chave

- Ao final de cada Sprint, um incremento de software tem que estar pronto.





O incremento é a soma de todos os itens do Backlog do Produto prontos durante uma iteração, mais todos os anteriores já finalizados.

Pronto significa que pode ser usado e posto em produção imediatamente. Se não for possível colocá-lo em produção, não está pronto.

**PRONTO =** Requisitos finalizados. Sem trabalho remanescente. + Boa qualidade para ser usado em produção. + Em condições de ser liberado para produção.

+ **Ser transparente. Todos sabem o que significa.**




## Definição de “Pronto”

- Quando um item do Backlog do Produto ou um incremento é descrito como **“pronto”**, todos devem entender o que **“pronto”** significa.
- Para criar uma maior transparência, deve ser então criada uma **“Definição de Pronto (Definition of Done – DoD)”.**

**DEFINIÇÃO DE PRONTO** = Descrição de tudo **o que** o Time de Desenvolvimento **deve fazer** para entregar um **incremento pronto**, de **boa qualidade** e **potencialmente liberável**. Permite que todos tenham **claro entendimento** do que significa “pronto”.

- É uma lista de atividades que agregam valor ao desenvolvimento do software.
- Quanto mais harmonioso for o conjunto de atividades, mais produtivo será o time.

  Intimamente relacionada às práticas de engenharia de software.
- Tudo o que estiver na definição de “pronto” deve ser feito antes que o incremento ou o item seja dado como “pronto”.

## Definição de “Pronto" – Exemplo

- A definição de pronto é uma lista de atividades que agregam valor ao desenvolvimento do software.
- **Tudo o que estiver na definição de “pronto” deve ser feito antes que o incremento ou o item seja dado como “pronto”.**

### DoD - Exemplo 1

- ✓ Código completo de acordo com os padrões de codificação.
- ✓ Código coberto por testes unitários.
- ✓ Código revisado por pares (peer review).
- ✓ *Build* automático.
- ✓ Não há tarefas pendentes para a funcionalidade.
- ✓ Sem defeitos conhecidos abertos.
- ✓ Todos os testes de aceitação executados.
- ✓ Manual de usuário atualizado.
- ✓ Testes de integração realizados.
- ✓ Testes de regressão realizados.

### DoD - Exemplo 2

- ✓ Código desenvolvido e testado.
- ✓ Testes unitários com cobertura de 80% (novos desenvolvimentos).
- ✓ Testes unitários executados e sem falhas.
- ✓ Bugs corrigidos e re-testados.
- ✓ Funcionalidades revisadas pelo Dono do Produto.
- ✓ Implantado em ambiente de homologação.
- ✓ Scritps ajustados para *deploy*.
- ✓ Release notes atualizadas.




## Definição de “pronto” - Benefícios

Entre outros benefícios, destacam-se:

| | |
|---|---|
| **Aumenta a transparência** | • Todos os stakeholders sabem exatamente o que significa um produto pronto.<br>• Serve como um acordo entre o time de entrega e o Dono do Produto, garantindo que todo o produto gerado pelo projeto estará dentro dos padrões de qualidade estabelecidos entre eles. |
| **Aumenta a previsibilidade** | • As entregas passam a ser mais previsíveis em termos de qualidade.<br>• Ajuda a assegurar que o incremento é realmente liberável ao final de uma iteração (ou sprint).<br>• Ajuda o time de entrega a planejar quantos itens de backlog do produto consegue entregar na próxima iteração. |

## Módulo 6

# Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas




## Reuniões diárias (stand-up meeting)

- É uma prática chave dos times ágeis.
- São curtas, focadas e eliminam a necessidade de outras reuniões de status.
- Duram 15 minutos ou menos (time-box).
- Os participantes respondem apenas a três perguntas:
  - →**O que foi feito desde a última reunião?**
  - →**O que você planeja completar hoje?**
  - →**Quais são os impedimentos no trabalho?**

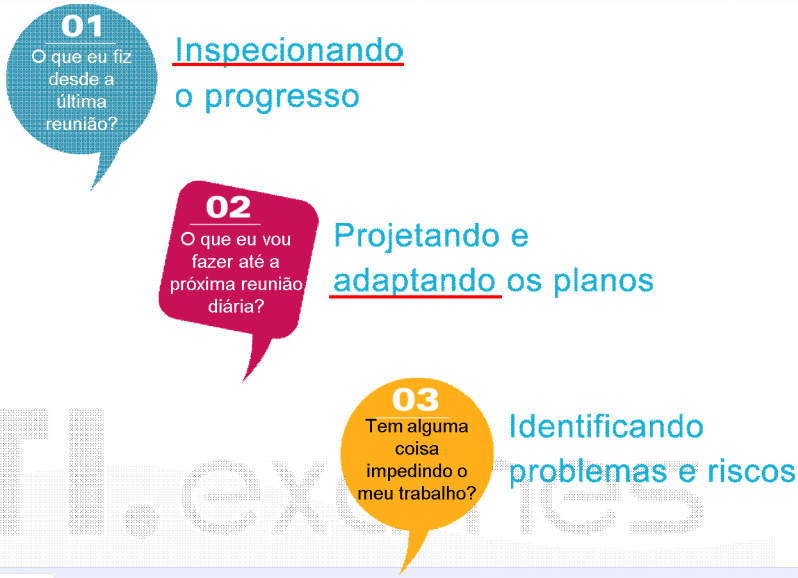

- São reuniões para o time e realizadas pelo time para ajudar a todos a ficarem focados no escopo e na meta da iteração acordados.
- Não são reuniões para reportar a situação do projeto ao líder/gerente de projeto.
- Devem ser realizadas preferencialmente no mesmo local e horário para reduzir a complexidade.
- Para manter a time-box de 15 minutos, convém que conversas sobre como resolver uma questão sejam realizadas fora desta reunião.

## Regras para a reunião diária (stand-up meeting)

- Se você tem uma tarefa, deve participar.
- Somente pessoas com tarefas podem falar.
- Fale para o time, não para o líder / Scrum Master / gerente de projetos.
- Sem conversas paralelas.
- Caso surjam novas tarefas, adicione-as no backlog da iteração.
- Discuta a resolução de problemas depois da reunião diária.
- Resolva os problemas após a reunião diária (não durante).




## Reuniões diárias (stand-up meeting)



## Impedimentos

- Impedimentos são continuamente identificados ao longo da iteração e se tornam transparentes durante a reunião diária.
- Há diversas maneiras de tratar os impedimentos, entre elas podemos usar as seguintes diretrizes:

1

Fazer com que os impedimentos fiquem visíveis.

2

Resolver o que estiver dentro do seu contexto.

3

Solicitar ajuda ao Agile Practitioner / Scrum Master / Agile Coach / Agile Project Manager se estiver fora do seu contexto.

4

Envolver o Dono de Produto (ou representante do cliente) se o impedimento persistir.




## Módulo 6

# Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas

## Spikes

- É um esforço curto (normalmente timeboxed) que é dedicado a explorar uma abordagem, investigar um problema ou reduzir um risco de projeto.
- Embora os spikes possam ser feitos a qualquer momento durante um projeto, eles geralmente assumem a forma de breves iterações exploratórias ou esforços de comprovação de conceito que são feitos no início de um projeto, antes do início do esforço de desenvolvimento.
  → Quando usado dessa maneira, o spike pode parecer muito semelhante à iteração 0.

Tipos especializado de spikes

### Spikes arquiteturais

- Spike arquitetura é um esforço curto para verificar se a abordagem que o time espera usar irá funcionar para o projeto.
- Exemplo: "vamos passar uma semana testando o desempenho dos drivers de banco de dados nativos antes de tomar uma decisão sobre a conectividade".
- A ideia é explorar a viabilidade de uma abordagem ou solução candidata em um curto espaço de tempo.

### Spikes baseados em riscos

- O spike baseado em risco é um esforço para investigar - e esperamos reduzir ou eliminar – um risco ou ameaça ao projeto.
- Essas curtas experiências para investigar porções de risco do projeto são uma ferramenta chave para o gerenciamento de riscos.




## Módulo 6

# Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas

## Demonstrando o produto (obtendo feedback)

# Loops de Feedback são essencias para projetos ágeis

- Permitem a inspeção e a adaptação dos processos e do produto.
- São fundamentais para o aprendizado do time do projeto e stakeholders.
- São fundamentais para o engajamento dos stakeholders.
- Mostram o progresso real do projeto.




## Revisão da iteração

**Demonstração dos itens prontos e obtenção de feedback**

Inspeção

**Time de Entrega**

**Atualização do Backlog do Produto**

Adaptação

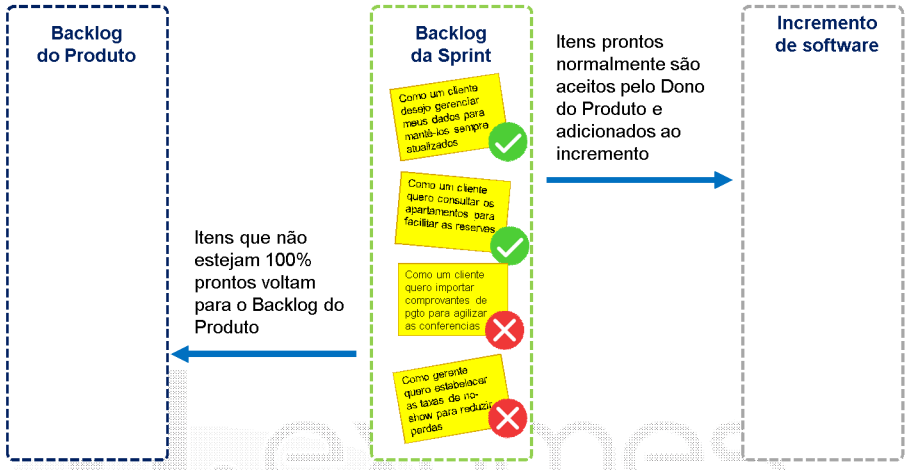
**Dono de Produto / Equipe de valor**

**Monitoramento do progresso do projeto**

Projeção

**Dono de Produto / Equipe de valor**

## Somente os itens prontos devem ser apresentados







## Módulo 6

# Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas

## Melhoria contínua

- Fundamentalmente, uma iteração deve resultar em duas coisas:

INCREMENTO PRONTO

EQUIPE MELHOR

A reunião de retrospectiva da iteração tem papel fundamental nesses objetivos, pois permite a **inspeção** do que não está bom, do que pode melhorar e qual o **plano (adaptação)** para melhorar.

>> Iremos discutir este assunto amplamente durante o módulo 8.



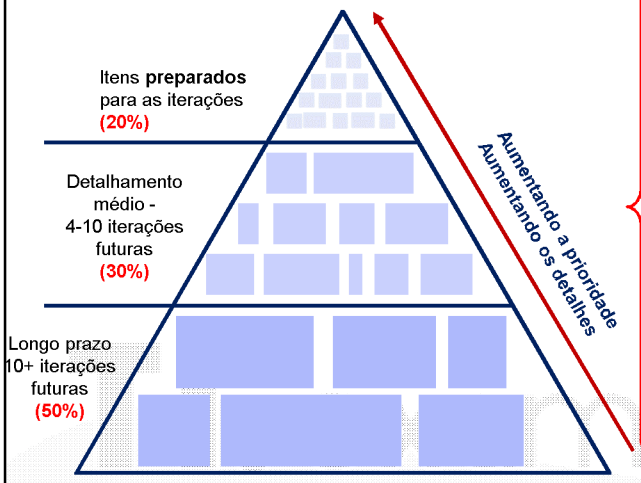

## Módulo 6

## Execução de projetos ágeis

**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas

## Detalhamento do Backlog do Produto - relembrando



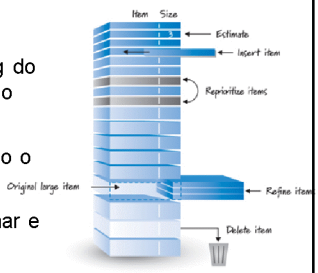

- O dono de produto é o responsável por priorizar, detalhar e refinar o backlog do produto. Ele faz isso ao longo das iterações.
- O dono de produto vai precisar da ajuda do time de entrega para fazer isso, pois envolve questões técnicas e de estimativa.
- O dono do produto vai fazer isso com o time durante o planejamento da iteração, porém, não vai dar tempo de fazer essas atividades pensando na iteração atual e em futuras.
- E o que fazer então?

**Grooming do Backlog**




## Grooming do Backlog do Produto

- É a preparação e refinamento do Backlog do Produto.
- Tradução de *Groom* = preparar, arrumar.
- O Dono do produto é o responsável por realizar o Grooming do Backlog do Produto, mas pode contar com o apoio de todo o time para realizar isto.
- Normalmente é realizado por meio de uma reunião com todo o time de entrega.
- O objetivo desta reunião é ajudar o Dono do Produto a refinar e preparar os itens do Backlog do Produto para entrarem nas iterações futuras.
- Durante o *grooming*, o time avalia a granularidade dos itens do Backlog do Produto e, se necessário, os quebra em itens menores.
- Uma vez que itens do Backlog do Produto estejam claros o suficiente, o time de entrega pode estimar a complexidade de cada um.



## Dicas para o Grooming

- Não tente agendar a reunião de grooming durante os primeiros ou últimos 20% da iteração.
- Trate a reunião de grooming igual à primeira parte da reunião de planejamento da iteração.
- O Dono do Produto deveria apresentar trabalho suficiente para planejar ao menos duas iterações, além da iteração atual.
- Tenha certeza que todos entendem que a estimativa não é a definitiva até que a reunião de planejamento da iteração aconteça.




## Reunião de Grooming

### Proposta

- Prover entradas para o dono de produto sobre o status atual dos próximos itens de backlog.
- Inclui esclarecer e analisar dependências, riscos, premissas, critérios de aceitação e estimar (em alto nível), a fim de informar a priorização e manutenção contínua do backlog do produto.

### Agenda

- Abertura
- Grooming dos itens de backlog
  - ✓ Revisar as próximas histórias
  - ✓ Ajustar as histórias
  - ✓ Fornecer estimativas de alto nível
- Retrospectiva da reunião
- Encerramento

### Descrição da reunião

- Esta reunião deveria fornecer ao dono de produto informações suficientes pra manter o backlog do produto saudável.
- Esta reunião fornece pontos de sincronização com o time de desenvolvimento para garantir que os itens de backlog do produto estão entendidos e preparados para entrarem nas próximas iterações.
- O time inteiro deveria ser envolvido nesta reunião.
- Especialistas são encorajados a participar.

### Participantes

- Dono de produto
- Time de entrega
- Scrum Master / gerente de projetos
- Especialistas nos assuntos
- Clientes
- Etc.

### Saídas

- Itens do backlog esclarecidos e comentados com informações relevantes.
- Estimativas de alto nível para implementação dada pelo time de entrega.
- O dono de produto é equipado com informações para ajudar com as decisões de priorização.

## Módulo 6

# Execução de projetos ágeis

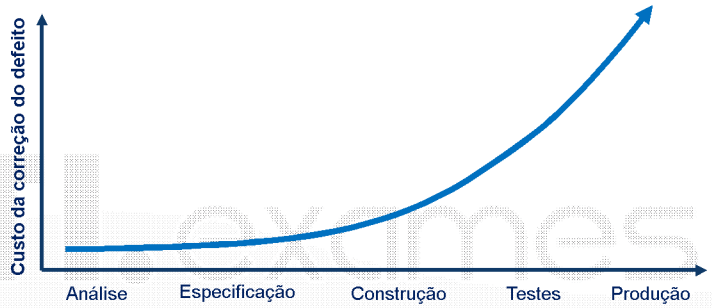
**Este módulo cobre:**

- A execução de projetos ágeis
- Tipos de iterações
- Time-box
- Planejamento da iteração
- Definição de pronto
- Executando a iteração
- Spikes
- Demonstrando o produto e obtendo feedback
- Retrospectiva da iteração
- Grooming do backlog
- Detecção de problemas




## Detecção de problemas

- O objetivo desta seção é apresentar ferramentas para identificação e resolução de problemas.
- Sabemos que problemas irão ocorrer no projeto. O quão efetivamente o time lida com problemas tem um impacto crítico sobre o sucesso ou fracasso do projeto.
- Para minimizar os impactos destes problemas, devemos encontrá-los o mais rápido possível.
- Detectar problemas de forma antecipada diminui o retrabalho, aumenta a confiabilidade do projeto e as chances de ser um projeto de sucesso.



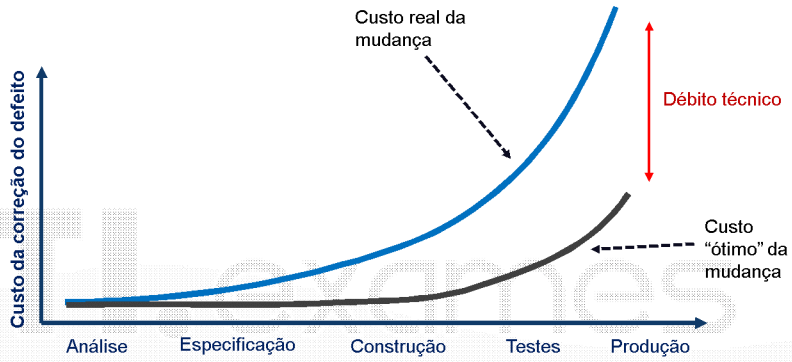
## Encontrando os problemas





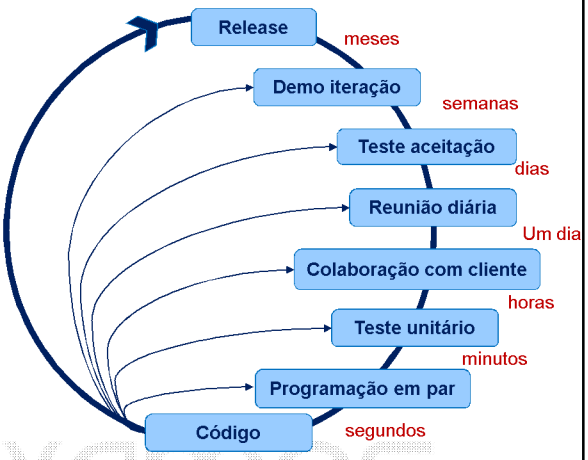
## Débito técnico

- Débito técnico é tudo aquilo que reduz a velocidade do time.
  - →Por exemplo: erros no sistema, ausência de testes automatizados, alta complexidade do código, ausência de testes unitários, arquitetura ruim, ausência de padrões de design e código, entre outras coisas.
- É necessário saber que débito técnico é algo inevitável, sempre vai existir, mas deve ser em baixo nível. E se não for pago, o débito tende a aumentar com o tempo.



## Validação e Verificação frequente

- As técnicas ágeis são projetadas para resolver os problemas o mais rápido possível, antes que eles possam crescer mais e subir a curva de custo de mudança.
- O Ágil usa testes regulares, pontos de verificação e revisões para resolver os problemas antes que eles se tornem maiores.
- Esta prática é chamada de **verificação e validação frequentes**.
- A figura ao lado ilustra os ciclos de verificação e validação que ocorrem em um projeto de desenvolvimento de software com XP.







## Testes exploratórios e de usabilidade

- O conteúdo do exame PMI-ACP menciona dois tipos especializados de teste: testes exploratórios e de usabilidade.
- Esses tipos de testes são comumente usados no desenvolvimento de softwares e são mais fáceis de entender nesse contexto, embora não estejam necessariamente limitados a projetos deste tipo.

Tipos especializado de testes

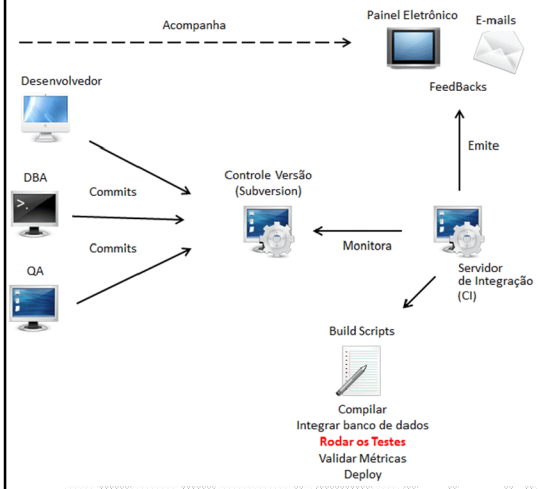
### Testes exploratórios

- Os testes exploratórios diferem dos testes de script, os quais tentam exercitar todos os componentes funcionais de um sistema.
- Em vez disso, confiam na autonomia, habilidade e criatividade do testador na tentativa de descobrir problemas e comportamento inesperado.
- Estes são um complemento aos testes de script, e não de forma alguma um substituto para eles.
- O uso de uma combinação de teste de script (manual ou automatizado) juntamente com testes exploratórios não-scriptados aumenta a cobertura do teste e reduz o risco de que um defeito não seja detectado.

### Testes de usabilidade

- Testes de usabilidade tentam responder à pergunta: "Como um usuário final responderá ao sistema em condições realistas?"
- O objetivo desse tipo de teste é diagnosticar como é fácil usar o sistema e ajudar a descobrir onde há problemas que podem precisar de redesenho ou alterações.
- Isso normalmente envolve a observação de usuários, pois são eles que vão interagir com o sistema pela primeira vez.
- Por exemplo, onde eles fazem uma pausa e têm de pensar sobre como fazer alguma coisa? Onde eles ficam presos e pedir ajuda?
- Os dados podem ser coletados filmando os usuários, usando ferramentas de rastreamento ocular ou realizando entrevistas pós-teste.

## Integração contínua



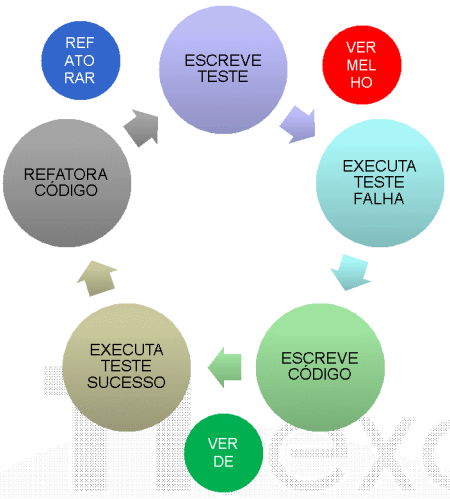

A integração contínua ajuda a minimizar os problemas de integração oriundos de várias pessoas fazendo alterações incompatíveis no mesmo código-fonte (produto).

**Benefícios:**

- A grande vantagem da integração contínua está no feedback instantâneo.
- O time recebe um aviso (feedback) de forma antecipada sobre os problemas causados.
- Problemas de integração são resolvidos antes e com custos menores.
- Reduz riscos, pois como o sistema é integrado continuamente e rapidamente, os erros também são detectados na mesma velocidade.



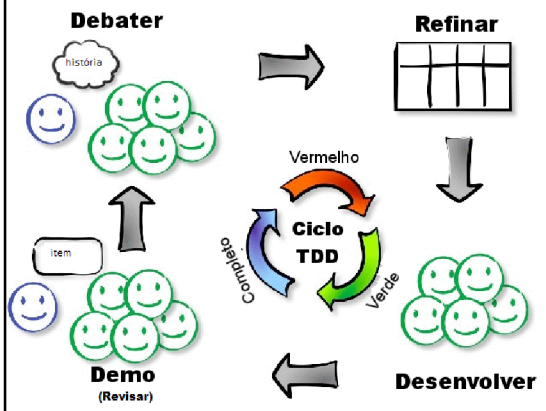

## Desenvolvimento dirigido a testes (TDD - Test Driven Development)




É uma técnica de desenvolvimento de software que se baseia em um ciclo curto de repetições:

- Primeiramente o desenvolvedor escreve um caso de teste automatizado que define uma melhoria desejada ou uma nova funcionalidade.
- Então, é produzido código que possa ser validado pelo teste para posteriormente o código ser refatorado sob padrões aceitáveis.

## Desenvolvimento dirigido por testes de aceitação (ATDD – Acceptance Test-Driven Development)




É uma prática de obtenção de requisitos de forma colaborativa aplicada por times ágeis na qual exemplos concretos e testes automatizados são utilizados para especificar os requisitos, tornando-os mais claros.

**1 -** As histórias de usuário são refinadas durante o grooming do backlog do produto. Devem ser elencados exemplos de utilização das histórias de usuário de tal forma que esses cenários possam ser escritos como testes.

2 - O próximo passo é organizar os testes de aceitação em um formato requerido pelo framework de automação de testes.

3 - O próximo passo é implementar a funcionalidade para fazer com que o teste de aceitação passe.

4 - Após os testes passarem com sucesso, a história é verificada pelo dono de produto durante a reunião de revisão.




## Work in Progress

**Qual o problema com este projeto?**

## Limite WIP

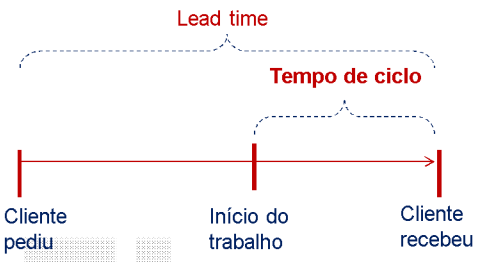
- O Kanban limita o trabalho em progresso (Work In Progress – WIP) para ajudar a identificar problemas e minimizar o desperdício e custos associados a mudanças durante o desenvolvimento.





## Tempo de ciclo (Cycle time)

- **É uma métrica que apresenta o quanto de tempo demora para termos um item pronto.**
- Esse tempo inicia a partir do momento em que **o time começa a trabalhar em um pedaço do projeto** (uma história, por exemplo), até que o item **esteja finalizado, aceito e possa entregar valor de negócio.**




**WIP excessivo:**

- O WIP representa o dinheiro investido sem ainda retorno de investimento.
- O WIP oculta gargalos no processo e isso mascara a eficiência.
- O WIP representa os riscos na forma de retrabalho potencial.

*Quanto menor, melhor*

- Tempo de ciclo está intimamente ligado ao "Work In Progress".
- **Longos tempos de ciclo levam ao aumento da quantidade de WIP.**

## Tempo de ciclo (Cycle time)

- Para ajudar a manter o foco na redução do tempo de ciclo, muitos times anotam a "data inicial" e "data final" nos cartões (post-it):

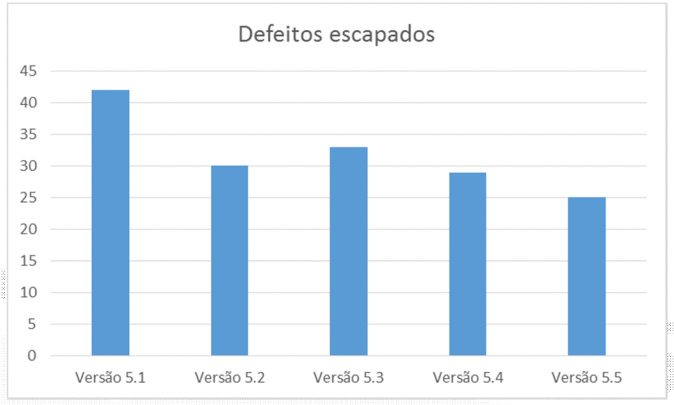

- O tempo de ciclo do projeto é a média do tempo de ciclo dos itens.
- **Ou em outras palavras, é o tempo médio para produzir cada item!**

$$\text{Tempo de ciclo} = \frac{\text{WIP}}{\text{Taxa de entrega (Throughput)}}$$




## Defeitos escapados (escaped defects)

- São os problemas que foram encontrados pelo cliente durante o período de utilização do produto.
- É importante monitorar a quantidade de defeitos escapados reportados contra a sua origem.



## Análise de variação

**Variação**

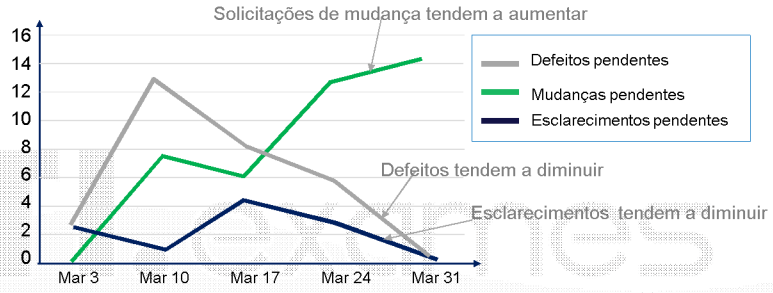
- É a medida de quão distante as coisas estão ou o quanto elas variam uma da outra.
- Exemplo: Se você pedir para várias pessoas estimarem o mesmo trabalho, então haverá uma variação (diferença) entre suas estimativas. E uma vez que o trabalho for executado, haverá uma variação (diferença) também entre a estimativa e o realizado.

- Ao avaliar o desempenho ou acompanhar o resultados, devemos entender que sempre existirá uma certa variação devido a flutuação normal.
- Muitas vezes vamos simplesmente aceitar as variações devido a estarem relacionadas a causas comuns.
- Somente vamos precisar investigar ou tomar ações para variações **com causas especiais**. E, com o apoio das reuniões diárias, podemos ter informações de quaisquer questões ou impedimentos que podem atrapalhar o rendimento do time, levando a assim a uma questão especial que precisa ser resolvida.
  - → Exemplo: Um desenvolvedor está tendo baixo rendimento devido a seu computador apresentar travamentos frequentes.




## Análise de tendência

- É uma ferramenta importante para a detecção de problemas porque fornece *insights* sobre questões futuras antes que elas ocorram.
- A figura abaixo mostra os defeitos, solicitações de mudança e esclarecimentos (questões como "como eu uso o produto X?") que foram registrados pelo time durante o mês de março:
  - → Podemos notar que houve um aumento do volume de solicitações de mudança e o time não teve capacidade para processá-las na mesma velocidade. Além de ter aumentando o WIP, isto pode nos dizer que o time talvez não tenha gastado o tempo suficiente para a validação dos requisitos antes de desenvolvê-los.



## Fim do módulo