Curso de Especialização em Engenharia de Software

Disciplina:

Engenharia de Software

Prof. Dr. Rodolfo Miranda de Barros

Metodologias Ágeis - Scrum

Metodologias Ágeis

- Reação às metodologias tradicionais;
- “Manifesto ágil“ (2001): http://www.agilemanifesto.org/
  - Movimento iniciado por programadores experientes e consultores em desenvolvimento de software;
  - Questiona e se opõe a uma série de mitos/práticas adotadas em abordagens tradicionais de Engenharia de Software e Gerência de Projetos.

Metodologias Ágeis

- Princípios do Manifesto Ágil:
  - **Indivíduos e interações** são mais importantes que **processos e ferramentas**;
  - **Software funcionando** é mais importante que **documentação completa (abrangente)**;
  - **Colaboração do cliente** é mais importante que **negociação de contratos**;
  - **Adaptação às mudanças** é mais importante que **seguir um plano**;
- Isto é, ainda que haja valor nos últimos itens, valorizamos mais os primeiros itens.

Metodologias Ágeis

Segundo Pressman:

- A engenharia de software ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento;
- A filosofia encoraja a satisfação do cliente e a entrega incremental de software logo no início;
- Métodos informais;
- Produtos de trabalho de engenharia de software mínimos e simplicidade global de desenvolvimento;
- As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar destas atividades não serem desencorajadas);
- Comunicação ativa e contínua entre desenvolvedores e clientes.

Metodologias Ágeis – Características Gerais

- Procuram minimizar riscos desenvolvendo software em pequenos espaços de tempo (iterações);
- Cada iteração é como um pequeno projeto:
  - Planejamento, requisitos, projeto, codificação, testes...
  - Propõem o desenvolvimento de software de forma mais rápida, com um grande número de ciclos, mas com qualidade;
- Objetivo de cada iteração:
  - Produzir componentes de software (incrementos);

Metodologias Ágeis – Características Gerais

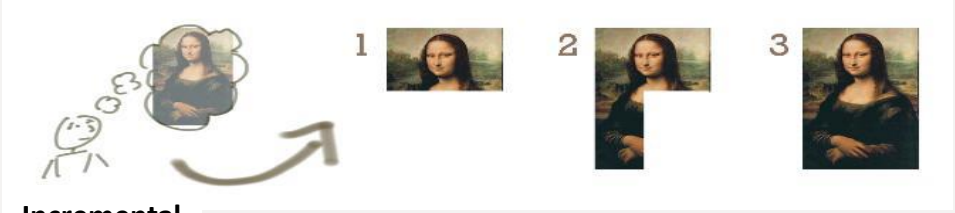
- Um catalizador efetivo para o feedback do cliente é um protótipo executável ou parte de um sistema operacional (“incrementos de software);
- “Incrementos de software” devem ser entregues em curtos períodos de tempo de modo que a adaptação acerte o passo com as modificações (imprevisibilidade);
- Essa abordagem iterativa habilita o cliente a avaliar o incremento de software regularmente, fornecer feedback necessário à equipe de software e influenciar as adaptações do processo feitas para acomodar o feedback.

SCRUM

O *Scrum* é um modelo ágil de processo que foi desenvolvido por Jeff Sutherland e por sua equipe no início da década de 90;

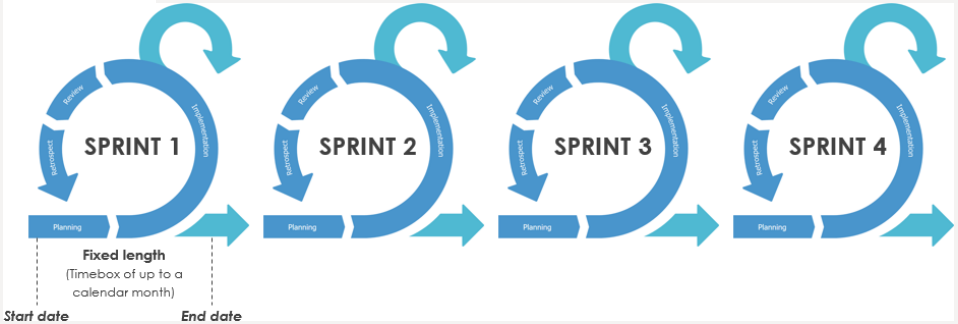
SCRUM

- O *Scrum* é um framework de processo ágil utilizado para gerenciar e controlar o desenvolvimento de um produto de software através de práticas iterativas e incrementais.
- É composto por um conjunto de boas práticas de gestão que admite ajustes rápidos, acompanhamento e visibilidade constantes e planos realísticos;
- Os projetos são divididos em ciclos chamados iterações, essas iterações são como miniprojetos que seguem todo um ciclo normal de desenvolvimento. Esses ciclos são chamados **Sprints**.



Incremental

Iterativo



SCRUM

- Os princípios *Scrum* são consistentes com o manifesto ágil:
  - Pequenas equipes de trabalho são organizadas de modo a maximizar a comunicação, minimizar a supervisão e maximizar o compartilhamento de conhecimento tácito informal;
  - O processo precisa ser adaptável tanto a modificações técnicas quanto de negócios, visando garantir que o melhor produto possível seja produzido;
  - O processo produz frequentes incrementos de software que podem ser inspecionados, ajustados, testados, documentados e expandidos.

Papéis no SCRUM

- **<u>Três papéis</u>**:
  - Product Owner;
  - Scrum Master;
  - Equipe (Team);

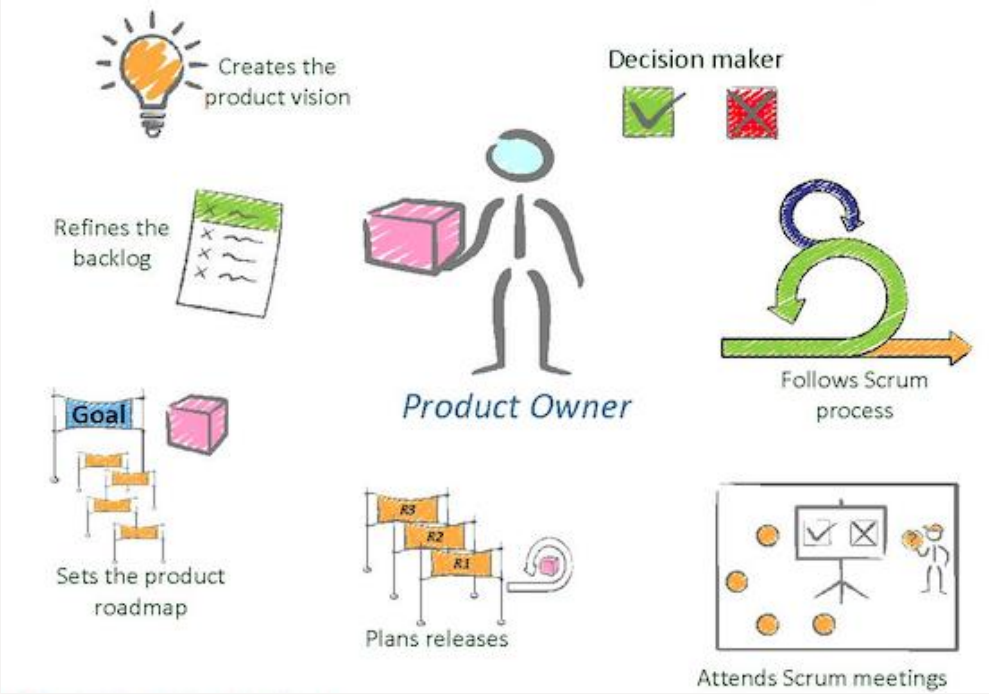
- Para o bom funcionamento do *Scrum* as pessoas responsáveis pelo projeto devem ter autoridade para fazer o que for necessário pelo seu sucesso;
- Pessoas não responsáveis não podem interferir no projeto:
  - Tal fato gera aumento de produtividade;
  - Evita situações constrangedoras para os envolvidos;
- Cada um conhece sua participação frente ao projeto e trabalha em conjunto para conseguir alcançar o objetivo definido.

Product Owner

- Responsável por apresentar os interesses de todos os stakeholders;
  - Dono do Produto do Software;
  - Usuário Final;
- Define fundamentos iniciais do projeto, objetivos e planos de release;
- Responsável pela lista de requisitos (Product Backlog);
  - Deve expressar claramente os itens do Product Backlog;
- Garante que as atividades com maior valor para o negócio são desenvolvidas primeiro;
- Garante que o Time de Desenvolvimento entenda os itens do Product Backlog no nível necessário;
- Para que o Product Owner tenha sucesso, toda a organização deve respeitar as suas decisões.

Características de um Product Owner

- Conhecimento do negócio;
- Capacidade de Negociação;
- Conhecimento do processo do Scrum;
- Acessível;
- Capacidade de Comunicação.

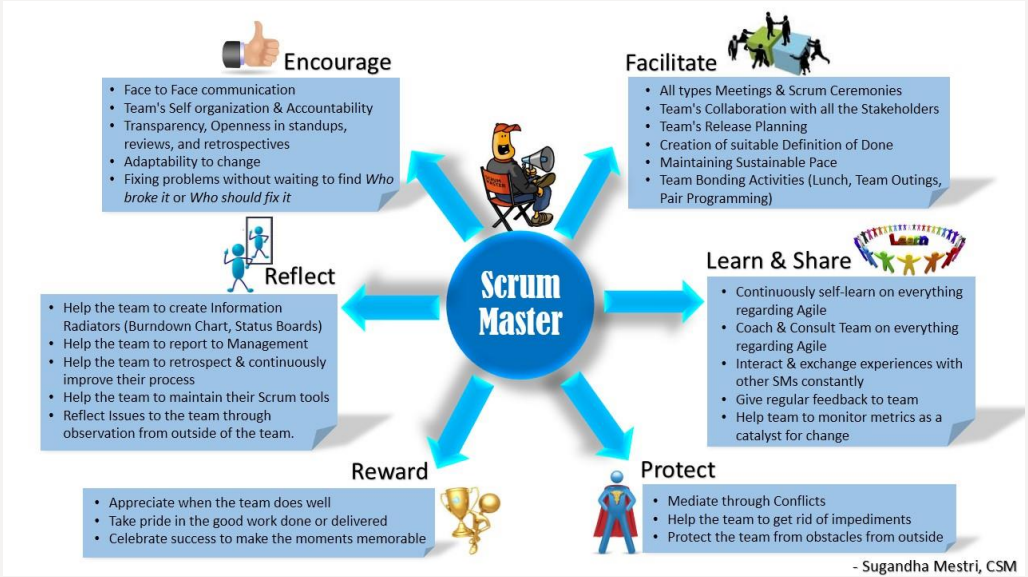


Scrum Master

- Responsável pelo sucesso do Scrum;
  - Os Scrum Masters fazem isso ajudando todos a entender a teoria, as práticas, as regras e os valores do Scrum.
- Garante que a equipe conheça e saiba trabalhar com Scrum;
- Implementar o Scrum na empresa de forma adaptada a sua cultura, para continuamente gerar benefícios;
- Certifica-se de que cada pessoa envolvida está seguindo seus papéis e as regras do Scrum;
- Certifica-se que pessoas não responsáveis não interfiram no processo;
  - Faz isso aqueles que estão fora do Time do Scrum a entender quais de suas interações com o Time do Scrum são úteis e quais não são.
- Remove impedimentos para o progresso do Time de Desenvolvimento;
- O principal papel do Scrum Master é servir o Time Scrum

Características de um Scrum Master

- Conhecimento da Equipe;
- Resolução de Problemas
- Conhecimento do processo do Scrum;
- Liderança;
- Capacidade de Comunicação.

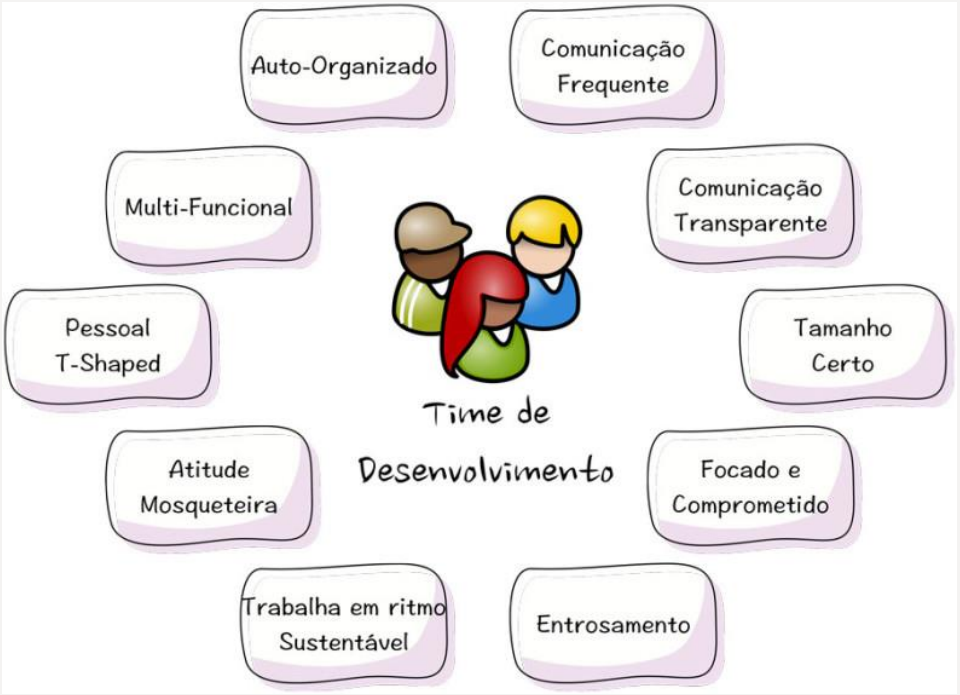


Time de Desenvolvimento

- Responsável por escolher as funcionalidades a serem desenvolvidas em cada interação e desenvolvê-las;
- O time se auto-gerencia, se auto-organiza;
- Todos os membros do time são coletivamente responsáveis pelo sucesso de cada iteração;
- Times de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto.
- Times de Desenvolvimento não contém sub-times dedicados a domínios específicos de conhecimento, tais como teste ou análise de negócios.
- O tamanho de um time de desenvolvimento pode alterar significativamente o sucesso do projeto;
  - Times muito pequenos vs times muito grandes.

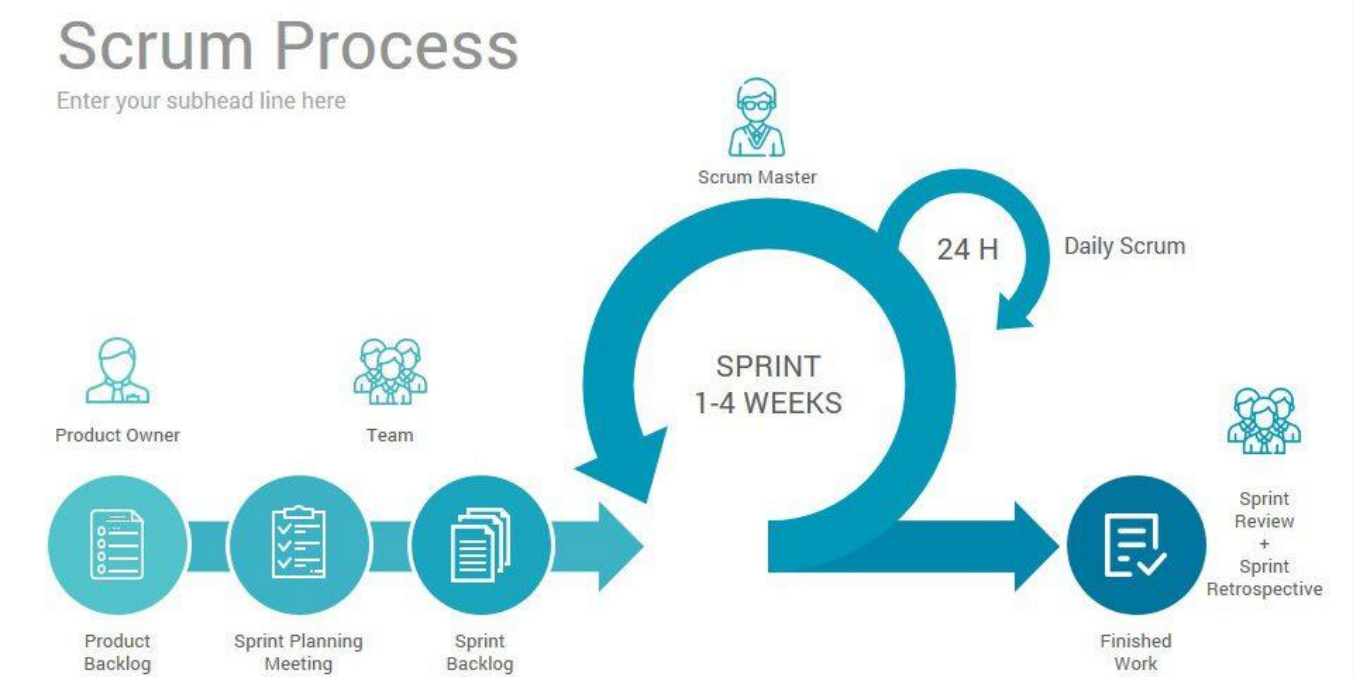
Características do Time de Desenvolvimento

- Conhecimento do Scrum;
- Proatividade;
- Colaboração;
- Trabalho em Equipe;
- Multidisciplinar;
- Auto-organizável.



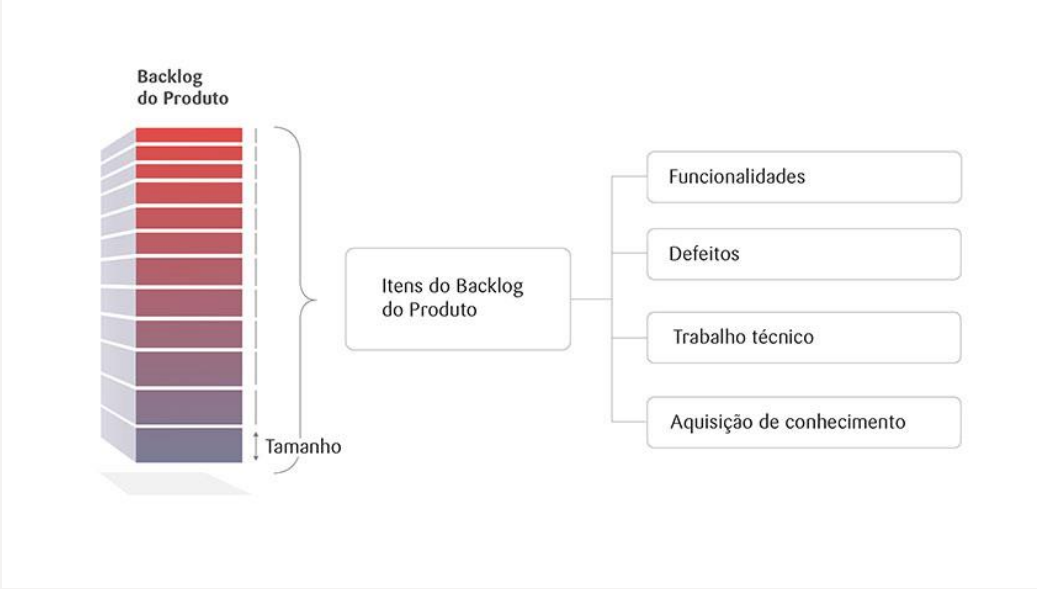
SCRUM - Fases

- Planejamento
- Sprints
  - Reuniões Diárias
  - Revisão
  - Retrospectivas
- Encerramento



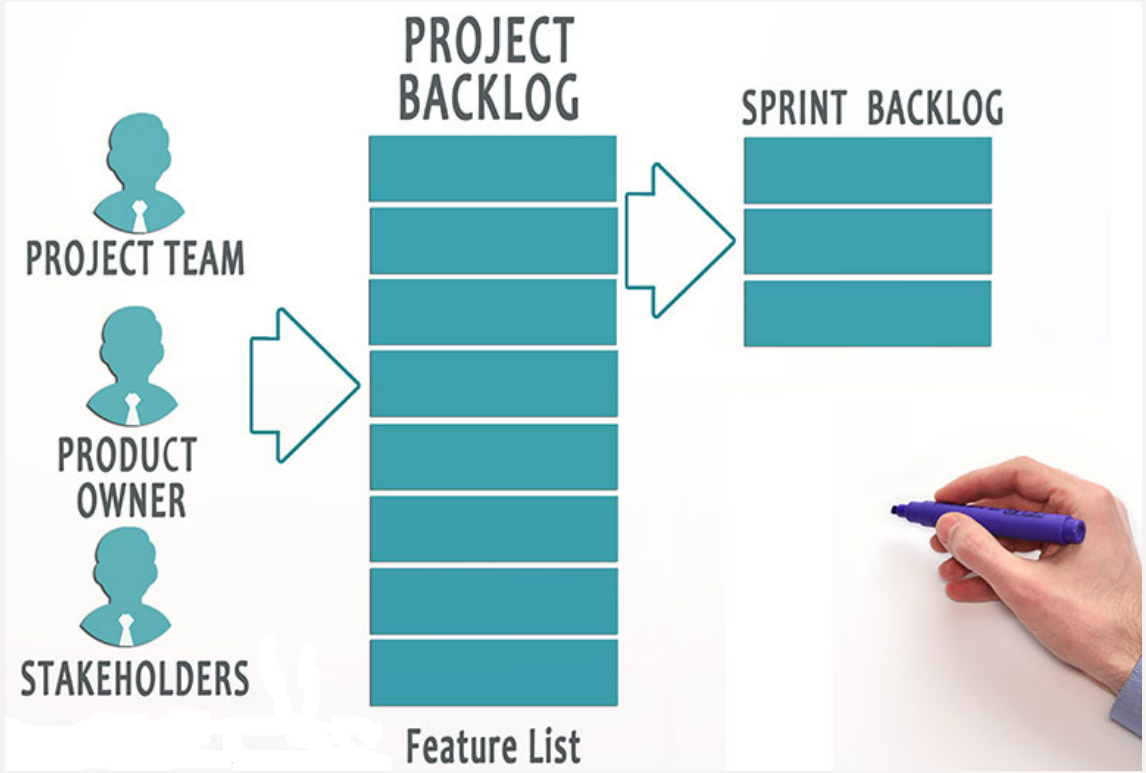
SCRUM - Planejamento

- Relativamente curto;
- Projeto da arquitetura do sistema;
- Estimativas de datas e custos;
- Criação do ***backlog***:
  - Participação de clientes para o Levantamento dos requisitos e atribuição de prioridades;
- Definição de equipes e seus líderes;
- Definição de pacotes a serem desenvolvidos;



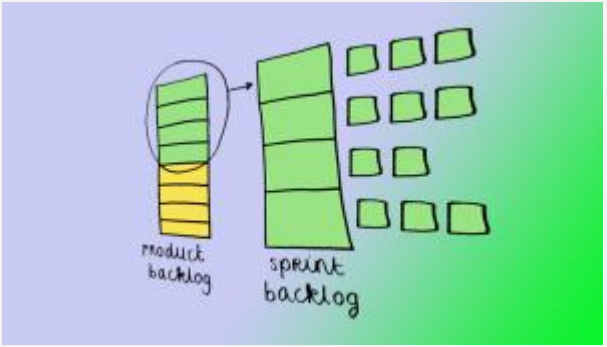
SCRUM – PLANEJAMENTO - Product Backlog

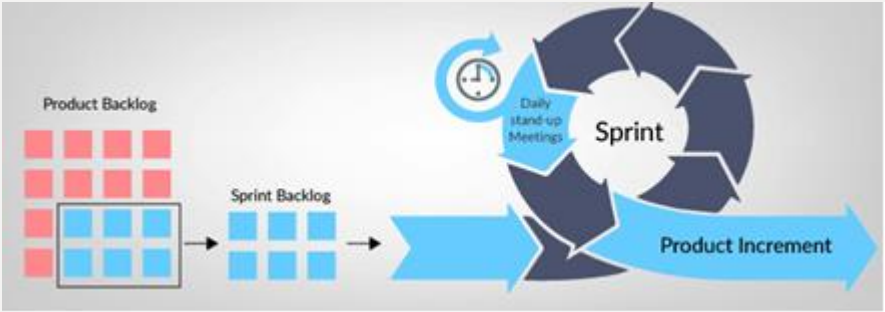
- Lista de todas as funcionalidades desejadas;
- É gerada incrementalmente:
  - Começa pelo básico, o extra aparece com o tempo;
  - Novos requisitos aparecem quando o cliente vê o produto;
- Pode conter:
  - Tarefas diretas, casos de uso e histórias;
- A lista é priorizada pelo *Product Owner*, que deve entender cada item da lista (história);
- Deve conter características que agreguem algum valor de negócio ao produto.



SCRUM - Sprint

- O time recebe uma parte do *backlog* para desenvolvimento;
- O backlog não sofrerá modificações durante o Sprint;
- Duração de 1 a 4 semanas;
- Sempre apresentam um executável ao final.





Tamanho - *Story Points*

- Story points:
  - Medida relativa do tamanho de uma história;
  - Não existe fórmula;
  - É resultado do agrupamento de fatores: esforço envolvido no desenvolvimento da funcionalidade, a complexidade, riscos, etc.

Tamanho - *Story Points*

- Usualmente, para cada story do backlog deverá ser atribuído um valor da série aproximada de Fibonacci (1,2,3,5,8,13,21, ...) (Planning Poker);
- O significado dos valores é relativo, onde uma story de pontuação 8 demanda aproximadamente quatro vezes mais esforço que uma story de pontuação 2;
- Essa atribuição deve ser feita pelo time e não por uma única pessoa;
- Para começar a pontuar as stories de um product backlog, pega-se a story que o time julga ser a de menor esforço e atribui pontuação 2. As demais stories deverão seguir uma pontuação relativa a essa primeira.

Velocidade

- É o total de story points (sp) entregues em uma iteração pela equipe;
- Exemplo: Se uma equipe completa em uma interação 3 histórias, uma estimada em 5 sp, uma outra em 3 sp e a terceira em 5 sp, a velocidade do time é de 13;

Velocidade

- Velocidade da equipe = 20
- Total de Story Points = 100

- Total de iterações do projeto = 100/20 = 5 iterações

- Duração da iteração = 3 semanas

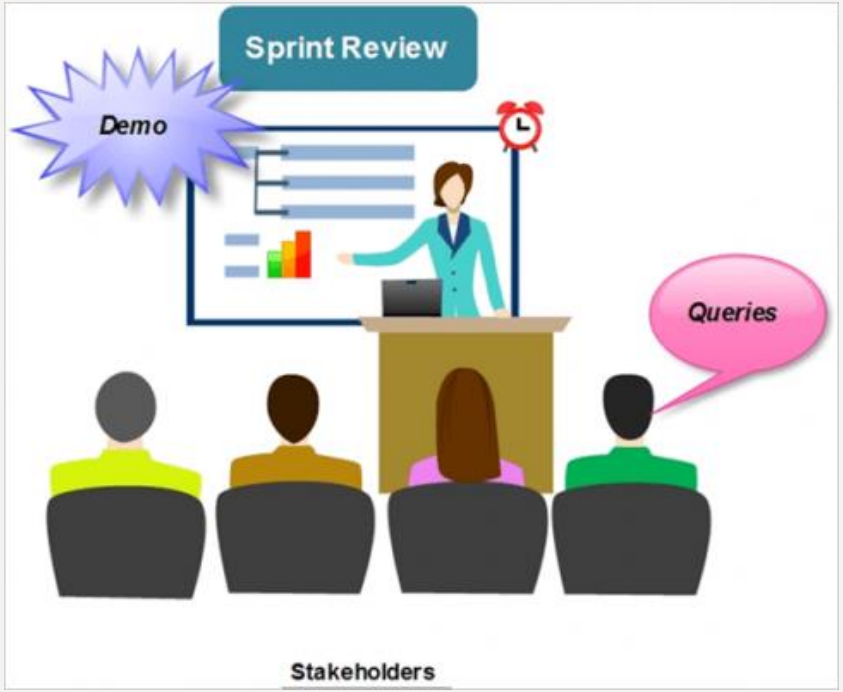
- Estimativa de entrega do produto = 15 semanas

SCRUM - SPRINT – Reuniões Diárias

- Cerca de 15 minutos de duração;
- Todos respondem às perguntas:
  - O que você realizou desde a última reunião?
  - Quais problemas você enfrentou?
  - Em que você trabalhará até a próxima reunião?
- Benefícios:
  - Maior integração entre os membros da equipe;
  - Rápida solução de problemas: promove o compartilhamento de conhecimento;
  - Progresso medido continuamente: Minimização de riscos;

SCRUM - SPRINT – Revisão

- Deve obedecer à data de entrega:
  - Permitida a diminuição de funcionalidades;
- Apresentação do produto ao cliente:
  - Sugestões de mudanças são incorporadas ao *backlog;*
- Benefícios:
  - Apresentar resultados concretos ao cliente;
  - Integrar e testar uma boa parte do software;
  - Motivação da equipe;
  - Cada stakeholder fala suas impressões e sugere mudanças com suas respectivas prioridades;
  - Possíveis modificações no Product Backlog são discutidas entre o Product Owner e o time;
  - ScrumMaster anuncia a data e o local da próxima reunião de revisão do Sprint ao Product Owner e a todos stakeholders;



SCRUM - SPRINT – Retrospectiva

- Sprint Retrospectiva é rito de avaliação do Sprint que acabou de se encerrar;
- Participam desta reunião:
  - Time, ScrumMaster e, opcionalmente, Product Owner;
- Os membros do time devem responder a duas questões:
  - O que aconteceu de bom durante o último Sprint?
  - O que pode ser melhorado para o próximo Sprint?
- ScrumMaster escreve as respostas e prioriza na ordem que deseja discutir as potenciais melhorias;
- ScrumMaster nesta reunião tem o papel de fazer com que o time encontre melhores formas de aplicar o Scrum.

SCRUM - Encerramento

- Finalização do projeto
- Atividades:
  - Testes de integração;
  - Testes de sistema;
  - Documentação do usuário;
  - Preparação de material de treinamento;
  - Preparação de material de marketing.

Kanban - O Que é?

- *Kanban* é um termo de origem japonesa e significa literalmente "cartão" ou "sinalização". Este é um conceito relacionado com a utilização de cartões (post-it e outros) para indicar o andamento dos fluxos de produção em empresas de fabricação em série;
- Segundo Highsmith (2009), esse termo se deriva do processo Japonês de melhoria contínua presente na filosofia "Just in Time". Essa filosofia foi desenvolvida pela Toyota a fim de melhor visualizar e controlar o fluxo da sua linha de montagem, pois permite uma clara visualização das fases dentro do processo de produção.

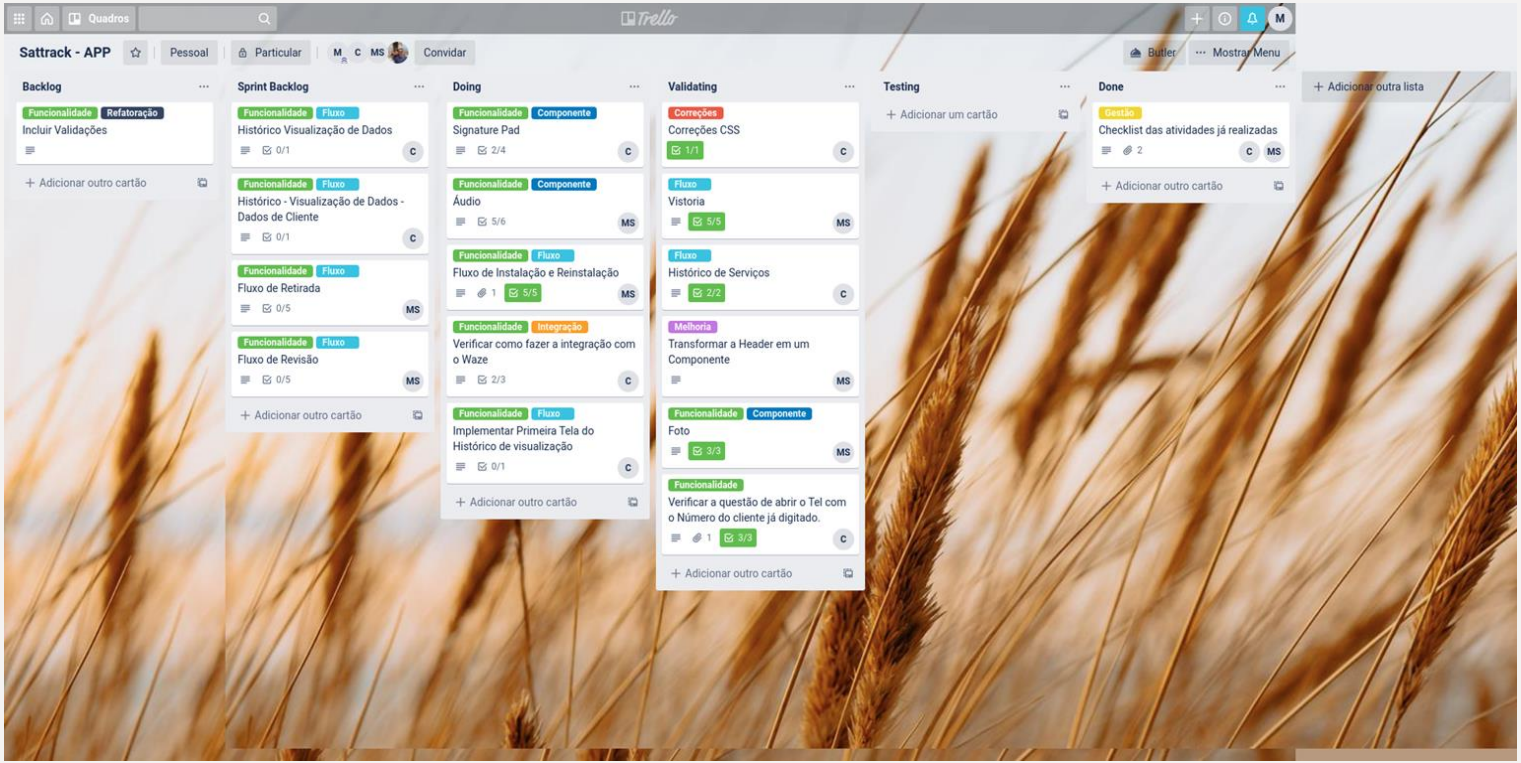
Kanban – O Que é?

- Depois de algum tempo, esse contexto foi aplicado a outros tipos de produção e também a gestão de projetos, sendo muito útil ao ajudar o gestor de projetos a controlar as suas tarefas nas diferentes fases de um projeto, bem como a planejar e dividir essas tarefas de forma eficiente, já que uma suas bases são times de trabalho auto organizáveis, iterações curtas, rápida e fácil reação a mudanças e melhoria contínua (Highsmith, 2009);
- No Kanban, cada tarefa ou História passa por diversas fases durante o seu ciclo de vida, tendo cada fase a sua função dentro desse ciclo de vida. Uma tarefa só é considerada pronta quando passa por todo esse ciclo e uma outra tarefa só é iniciada quando a anterior estiver finalizada (Highsmith, 2009).

Utilizando o Kanban para controlar o Projeto

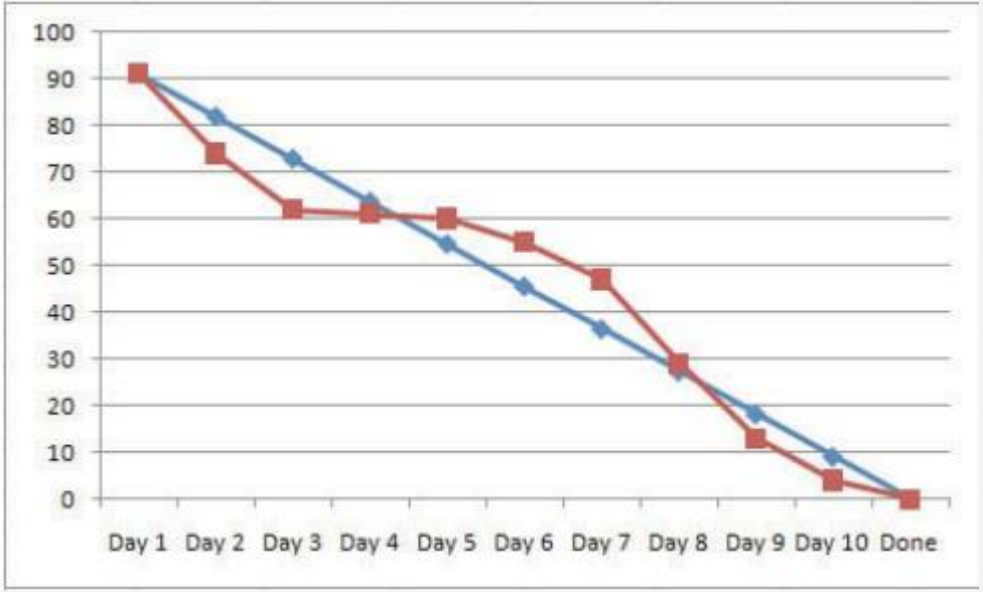
- O Kanban é o método mais comum para gestão visual de projetos ágeis de *software*;
- Geralmente, o Kanban para o Scrum trabalha com de 4 a 6 colunas, sendo:
  - Product Backlog;
  - Sprint Backlog;
  - Doing;
  - Tests;
  - Validation;
  - Done.

Exemplo Kanban para o desenvolvimento de Software



Burndown Chart

- Gráfico que permite monitorar o progresso do time de desenvolvimento;
- Mostra a quantidade de trabalho que já foi feita e o trabalho restante em um espaço temporal.



Conclusões sobre as aulas 2 e 3 (Modelos de Ciclo de Vida e Metodologias ágeis)

- Não fique limitado a uma “arma” ou técnica em particular;
- Metodologias diferentes são necessárias para diferentes tipos de projetos
  - Fatores a serem considerados:
    - Número de Pessoas envolvidas no Projeto;
    - Criticidade do Sistema;
    - Prioridades do Projeto;
- Construa a sua “caixa de ferramentas”;

Conclusões Módulos 1 e 2

- No modelo ágil há um conjunto de “ideias” que representam um afastamento significativo da engenharia de software convencional;
- Segundo Pressman, muitos conceitos ágeis são simples adaptações de bons conceitos de engenharia de software;
- Conclusão segundo Pressman: “há muito a ser ganho considerando o melhor de ambas as escolas, e quase nada a ser ganho denegrindo qualquer uma dessas abordagens.”

REFERÊNCIAS

- Highsmith, J. R. Agile project management: creating innovative products. Pearson Education, 2009.
- KNIBERG, Henrik. Scrum e XP Direto das Trincheiras. 2007. Disponível em: https://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches. Visitado em: 16/06/2020.
- Laboratório GAIA – http://www.gaia.uel.br. Visitado em: 18/06/2020.
- Pressman, Roger S. Engenharia de Software - Uma Abordagem Profissional, Amgh Editora, 8ª Ed., 2016.

BOM ESTUDO!

Em caso de dúvidas, não deixe
de perguntar aos tutores.

CAPES | UAB UNIVERSIDADE ABERTA DO BRASIL | NEAD Núcleo de Educação a Distância | Universidade Estadual de Londrina