



Devo usar Scrum?





Disciplina:

Métodos e Ferramentas de Engenharia de Software
Instituto de Informática, Engenharia de Software
Universidade Federal de Goiás
Goiânia, 2018/2

Alunos:

João Pedro Arruda Vieira
Larissa Chyevena Lopes de Mello
Natália Lopes da Silva
Nelson William Viana de Siqueira
Sofia Martins Moraes

Professor:

Fábio Lucena



Contexto

Contexto

- ➔ Surgido na década de 90, o **Scrum** é um **framework** estrutural, utilizado para gerenciar projetos, que se tornou extremamente popular, sendo adotado em peso no planejamento e gestão de projetos de **software**.

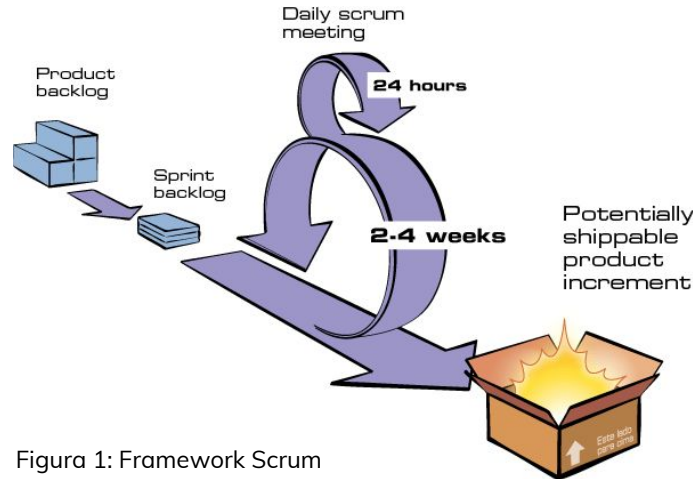


Figura 1: Framework Scrum





Contexto

- ➡ Mas, para entender o **porquê** de tantos projetos utilizarem Scrum, deve-se entender como esse framework **auxilia** o processo de desenvolvimento de software.
- ➡ Para isso, é necessário entender a **origem** da própria Engenharia de Software [1].





Contexto

- ➔ Na década de 60, a indústria se deparou com a chamada **“Crise do Software”**, ocasionada por um acúmulo de **problemas** no processo de desenvolvimento, tais como:
 - imprecisão nas estimativas de prazo e custo, insatisfação do cliente com o sistema entregue, qualidade de software menor que a adequada, dificuldade de manutenção do software, dentre outros [2].
- ➔ Para se resolver esses problemas, surgiu a **Engenharia de Software**.



Contexto

Figura 2: “Não há bala de prata”, representando o famoso artigo de Brooks.



- ➔ Porém, a Engenharia de Software **não foi suficiente** para se resolver todos esses problemas. Isso porque, segundo Frederick P. Brooks, o software enfrenta **dificuldades essenciais** (pertinentes à sua natureza) e **acidentais** (pertinentes à sua produção).
- ➔ A Engenharia criada resolve **somente** problemas acidentais.





Contexto

- ➔ Dessa forma, permaneceram problemas como:
- **Complexidade** — o software pode possuir grande número de estados;
 - **Conformidade** — padrões desenvolvidos por pessoas diferentes em tempos diferentes;
 - **Alterabilidade** — um produto pode passar por mudanças constantes;
 - **Invisibilidade** — software não é um produto tangível e visualizável, não pode ser representado através de uma abstração geométrica [3].



Contexto

- ➔ Logo, a fim de se **resolver** problemas relacionados à Engenharia de Software corrente e modelos de desenvolvimento tradicionais como o Cascata, considerados lentos, burocráticos e inflexíveis, foi concebido o conceito de **desenvolvimento ágil** de software.

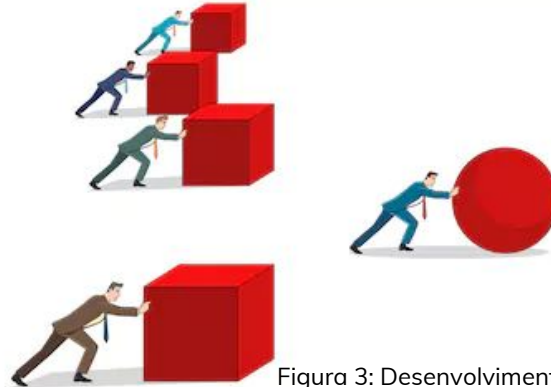


Figura 3: Desenvolvimento ágil





Contexto

- ➡ Foi publicado então o **Manifesto Ágil**, contendo os seguintes princípios:
- Indivíduos e interações mais que processos e ferramentas;
 - Software em funcionamento mais que documentação abrangente;
 - Colaboração com o cliente mais que negociação de contratos;
 - Responder a mudanças mais que seguir um plano [4].





Contexto

- ➡ Com o manifesto ágil, o Scrum rapidamente ganhou **popularidade** mas não necessariamente todo o público compreende sua aplicação.
- ➡ Hoje, há uma **noção equivocada** de que o Scrum é adequado para todos os projetos, similar uma “*bala de prata*”.
- ➡ É importante entender em **quais** projetos é realmente coerente seguir o que é proposto pelo Scrum [5].





Projetos adequados



Projetos adequados

- ➡ O Scrum tem uma base **empírica** voltada ao **controle de processo**. Ou seja, é fundamental que as experiências e decisões anteriores sirvam de base para a evolução do processo.
- ➡ Para apoiar essa perspectiva e obter os benefícios do Scrum, é necessário que o projeto possa apoiar três pilares importantes: **transparência, inspeção e adaptação** [6].





Projetos adequados

➡ Transparência

- O desenvolvimento do projeto deve ser transparente, expondo e discutindo as eventuais **adversidades**.
- Os interessados no projeto devem ter meios de **visualizar** aspectos significativos e especificações do processo.





Projetos adequados

➡ Inspeção

- Deve haver a possibilidade de **inspecionar** com frequência os produtos gerados, considerando o processo e seus objetivos e **sem interromper** as demais tarefas.
- O **Product Owner**, como conhecedor do domínio, é responsável por inspecionar os resultados e garantir que as necessidades estejam sendo atendidas. Suas decisões devem ser respeitadas pelo resto da equipe.





Projetos adequados

➡ Adaptação

- Deve haver a necessidade de **adaptações constantes** dos requisitos do projeto, seja por falta de definições claras ou pela volatilidade do negócio.
- O software sendo produzido deve ser prontamente **ajustado** para comportar as novas necessidades, de forma a evitar impactos negativos.





Requisitos



Requisitos

➔ Logo após a confirmação quanto à adequação do projeto ao Scrum, o próximo passo é estabelecer um ambiente ágil apropriado. Ignorar essa etapa aumentam as chances do projeto ser levado ao fracasso, portanto, é preciso estar atento ao requisitos essenciais para criar este ambiente [7]. São eles:

- **Uma equipe ágil**
- **Um produto viável**
- **Uma empresa ágil**
- **Um cliente ágil**





Requisitos

↳ Equipe ágil

- O **Scrum Master** deve dominar o framework e estar a par das soluções mais efetivas para a esfera do projeto. Além de atuar como um facilitador.
- O **Product Owner** deve ter conhecimento sobre a metodologia ágil e estar disponível para esclarecer as dúvidas que surgirem durante o desenvolvimento do projeto.
- Além de ter experiência, a **Equipe de Desenvolvimento** precisa ser auto-organizável e multifuncional a fim de entregar o produto de modo incremental e iterativo.





Requisitos

➡ Produto adequado

- O produto a ser desenvolvido deve obedecer à característica incremental e iterativa do Scrum, ou seja, tem que ser analisado logo no início do projeto se o mesmo pode ter suas funcionalidades divididas a fim de formarem mais de uma entrega.



A cluster of hexagons in various shades of blue and teal, some solid and some outlined, arranged in a geometric pattern in the top-left corner.

Requisitos

➡ **Empresa ágil**

- De forma geral, a empresa deve estar consciente e entender sobre os conceitos da metodologia ágil, além de apoiar e estarem dispostos à capacitar a equipe de modo que o Scrum funcione adequadamente.





Requisitos

↳ Cliente ágil

- A colaboração com o cliente é fundamental para o Scrum, sendo ele a essencial fonte de feedback sobre as constantes adaptações do projeto, a fim de que se garanta a certeza de que suas necessidades estejam sendo atendidas.





Erros



Erros

➡ Ao se utilizar o Scrum, os erros mais comuns que podem ocorrer são [8]:

- Esperar que a implementação seja fácil;
- Fazer as práticas sem os princípios;
- Scrum Master ser um gerente de projeto;
- Scrum Master ser o único responsável pela comunicação;
- Não envolver o PO;
- Não realizar reuniões diárias;
- Não levantar obstáculos cedo o bastante;
- Não realizar reuniões retrospectivas.



Erros

➔ Esperar que a implementação seja fácil

- Utilizar o Scrum não é apenas transformar requisitos em histórias de usuário, realizar reuniões e fazer as sprints, trata-se de uma mudança cultural dentro da organização, e isto demanda tempo. Para isso não deve haver falta de comunicação ou de responsabilidade, nem desconfiança entre o pessoal envolvido no projeto.



Figura 4: Implementação



Erros

➡ Fazer as práticas sem os princípios

- Os princípios do Scrum são o alicerce para um bom funcionamento e estabilidade da metodologia. Usar as técnicas e ferramentas do Scrum sem de fato entender o motivo por trás das práticas tornam a os esforços insustentáveis a longo prazo.

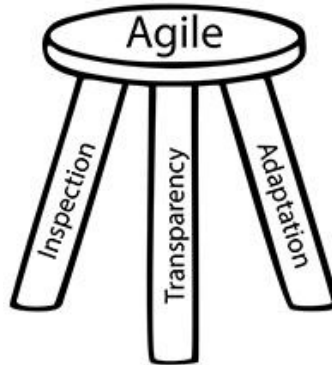


Figura 5: Pilares



Erros

➡ Scrum Master ser o gerente de projeto

- A equipe de desenvolvimento deve ser auto-organizável, portanto não cabe ao Scrum Master agir como um gerente de projetos e ditar as tarefas e atribuir esforços. Sua função é agir como um facilitador através de inspeções sobre quaisquer problemas que atrapalhem o andamento do projeto.



Figura 6: Scrum Master



Erros

➔ Scrum Master ser o único responsável pela comunicação

- A comunicação é um princípio crucial para as metodologias ágeis, e embora o Scrum Master seja responsável por promover interações entre os envolvidos no projeto, muitas vezes é mais produtivo que própria equipe desenvolvimento fale diretamente o Product Owner.



Figura 7: Comunicação



Erros

➡ Não envolver o PO

- A colaboração é um aspecto crucial do Scrum. Empresários e desenvolvedores necessitam de trabalhar juntos para produzir o software que o cliente deseja. Isso pode ser aprimorado por meio da colaboração, além da comunicação frequente, envolvendo feedback para validar e realizar correções.

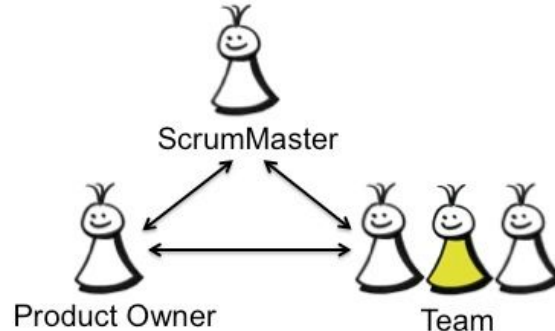


Figura 8 - SM, PO e time



Erros

➡ Não realizar reuniões diárias

- O *stand-up meeting* é demasiado importante em vários aspectos, pois são nessas reuniões que a comunicação e a colaboração dão visibilidade e transparência ao projeto.



Figura 9 - Stand-up meeting



Erros

➡ Não levantar obstáculos cedo o bastante

- Uma das principais responsabilidades do Scrum Master é remover obstáculos para a equipe continuar sendo produtiva. Entretanto, caso não haja o levantamento de obstáculos através de uma boa comunicação, o Scrum Master não tem como resolvê-los.



Figura 10 - Obstáculos



Erros

➡ Não realizar reuniões retrospectivas

- A *Sprint Retrospective* é essencial para que a equipe reflita sobre como se tornar mais eficaz e como entrar em sintonia com seus integrantes, além de ajustar seu comportamento para os próximos sprints.

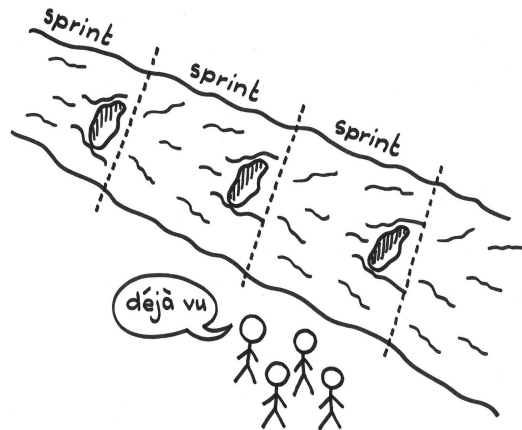


Figura 11 - Sprint Retrospective





Armadilhas



Armadilhas

➔ Serão listadas, a seguir, as principais armadilhas que podem ocorrer em determinados projetos [9]:

- Excesso de planejamento inicial;
- Falha ao reiniciar sprints;
- Deixar de aprender;
- Mudar o time constantemente.



Armadilhas

➡ Excesso de planejamento inicial

- É necessário evitar que as reuniões de *Sprint Planning* sejam longas demais. A equipe deve iniciar o projeto a partir das primeiras necessidades que forem identificadas pelo Product Owner.



Figura 12 - Sprint Planning

Armadilhas

➡ Falha ao reiniciar Sprints

- Ainda que seja incomum, caso um Sprint seja cancelado, a equipe deve retornar ao trabalho imediatamente, mesmo que seja mais conveniente seja esperar até que todos os erros se resolvam.



Figura 13 - Erros



Armadilhas

↳ Deixar de aprender

- O Scrum consiste no desenvolvimento constante e, para isso, é recomendado que os membros da equipe aprendam de maneira igualmente constante. A busca por conhecimento deve ser estimulada, com o objetivo de garantir que a equipe sempre tenha as habilidades necessárias para prosseguir com o projeto.



Figura 14 - Evolução



Armadilhas

➡ Mudar o time constantemente

- As equipes ágeis são auto-organizáveis, sendo recomendado que estas tenham alto desempenho para gerar valor. Mudar membros com frequência pode diminuir a eficiência e produtividade, podendo contribuir para a não permanência de outros membros no projeto.



Figura 15 - Organização do time





Estudo de caso



Sistema de Gestão de Estoque

- ➡ Estudo de caso realizado na Universidade Federal de São Carlos (UFSCar) com o objetivo de apresentar, num contexto de um ambiente acadêmico, a aplicação da metodologia ágil Scrum.
- ➡ Foram descritas as atividades realizadas, sua implantação e sua adaptação, destacando os principais desafios, dificuldades, benefícios, e lições aprendidas na aplicação do framework.





Sistema de Gestão de Estoque

- ➡ Como se passou em um ambiente acadêmico algumas adaptações foram necessárias como:
 - Ampliação do período das Sprints
 - Rotatividade do papel de scrum master
 - Reuniões semanais não presenciais
- ➡ Ao longo de 7 Sprints o time desenvolveu o projeto de forma contínua e evolutiva. A cada Sprint o time registrava suas dificuldades e lições aprendidas.





Sistema de Gestão de Estoque

- ➡ Ao final do projeto pode-se concluir que:
- O Scrum, mesmo contendo alguns padrões e regras pode ser adaptado ao projeto.
 - Mesmo tendo um time composto por alunos que não tinham conhecimento profundo sobre o assunto, ele foi implementado gerando um retorno satisfatório.
 - A dinâmica do Scrum fez com que o grupo estivesse mais unido e focado no objetivo





Sistema de Gestão de Estoque

- O caráter iterativo e evolutivo do Scrum deu aos alunos mais chance de desenvolverem suas habilidades;

Mesmo surgindo vários empecilhos a ferramenta foi implementada com sucesso, e projetos futuros irão estender sua utilização.





Fábrica de desenvolvimento distribuído de software

- ➡ Estudo de caso realizado por uma fábrica de desenvolvimento de software open source [O3S 2007]. Esta fábrica é formada por dez alunos do curso de Pós-Graduação do Centro de Informática da Universidade Federal de Pernambuco.
- ➡ O projeto executado está inserido na área de saúde coletiva/pública, denominado ANKOS (A New Kind Of Simulator) [ANKOS 2007], que surge como um sistema capaz de organizar as informações coletadas por pesquisadores em áreas de estudo da esquistossomose.





Fábrica de desenvolvimento distribuído de software

- ➔ Em relação ao estudo do projeto anterior podemos perceber que se trata de algo mais complexo, até por que ele é baseado em políticas [Johnson K 2001] que definem as diretrizes básicas a serem adotados por todos os projetos da fábrica de software.

Será que o Scrum também se encaixou nesse projeto?





Fábrica de desenvolvimento distribuído de software

- ➡ Nesse cenário, diferente do estudo anterior, o time já havia um conhecimento amplo do assunto, e antes do Scrum utilizavam outras metodologias em seus projetos.
- ➡ E por esse motivo adaptações foram necessárias:
 - Reuniões diárias substituídas pela comunicação assíncrona.
 - O Scrum master além de estar no gerenciamento dos impedimentos, também fazia parte do time de desenvolvedores.





Fábrica de desenvolvimento distribuído de software

➡ Ao final do estudo pode-se perceber que:

- Algumas premissas importantes do Scrum como a presença física do time com as iterações diárias, não puderam ser cumpridas devido às características do projeto.
- A parte do Scrum ser focado em equipes auto-organizadas e auto-gerenciáveis colaborou e muito para o projeto pois são aspectos semelhantes ao desenvolvimento distribuído.





Fábrica de desenvolvimento distribuído de software

- A parte do Scrum ser focado em equipes auto-organizadas e auto-gerenciáveis colaborou e muito para o projeto pois são aspectos semelhantes ao desenvolvimento distribuído.

Mesmo não cobrindo todas as características específicas para equipes geograficamente distribuídas, foi possível fazer uso de diversos aspectos de desenvolvimento ágil sem comprometer o projeto.



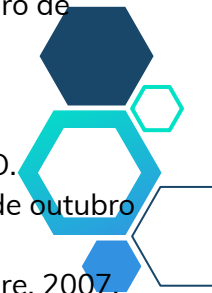


Obrigado!





Referências:

- [1] SCHWABER, K; SUTHERLAND, J. Um guia definitivo para o Scrum: as regras do jogo.
 - [2] DIJKSTRA, Edsger – The Humble Programmer. 1972.
 - [3] BROOKS, Frederick P. No Silver Bullet – Essence and Accident in Software Engineering. 1987.
 - [4] MANIFESTO PARA O DESENVOLVIMENTO ÁGIL DE SOFTWARE. Disponível em: <<http://www.manifestoagil.com.br/index.html>>. Acesso em: 01 de outubro de 2018.
 - [5] SCRUM. Disponível em: <<https://www.desenvolvimentoagil.com.br/scrum/>>. Acesso em: 01 de outubro de 2018.
 - [6] COMMON MISINTERPRETATIONS OF SCRUM. Disponível em: <<https://www.scrum.org/resources/blog/common-misinterpretations-scrum>>. Acesso em 01 de outubro de 2018.
 - [7] SCRUM PREREQUISITES. Disponível em: <<https://mplaza.pm/scrum-prerequisites/>>. Acesso em: 01 de outubro de 2018.
 - [8] IMPLANTAÇÃO DE SCRUM: 10 ERROS QUE VOCÊ NÃO PODE COMETER. Disponível em: <<https://www.scrum.org/resources/blog/common-misinterpretations-scrum>>. Acesso em: 01 de outubro de 2018.
 - [9] 15 ARMADILHAS COMUNS DO SCRUM. Disponível em: <<https://sitecampus.com.br/15-armadilhas-comuns-do-scrum/>>. Acesso em: 01 de outubro de 2018.
 - [10] DESENVOLVIMENTO DE SOFTWARE USANDO O FRAMEWORK SCRUM: UM ESTUDO DE CASO. Disponível em: <<http://www.revistatis.dc.ufscar.br/index.php/revista/article/view/81>>. Acesso em: 01 de outubro de 2018.
 - [11] SOARES, F. et al. Adoção de SCRUM em uma Fábrica de Desenvolvimento Distribuído de Software. 2007.
- 



Imagens:

Figura 1. Disponível em <https://www.desenvolvimentoagil.com.br/images/scrum/ciclo_scrum.gif>

Figura 2. Disponível em

<<http://labs.sogeti.com/wp-content/uploads/2017/11/no-silver-bullet-e1509987415294.png>>

Figura 3. Disponível em

<<https://image.shutterstock.com/image-vector/business-concept-cartoon-businessman-pushing-260nw-1023393298.jpg>>

Figura 4. Disponível em: <<https://www.joebarrios.com/scrum-master-role-business-analyst/>>.

Figura 5. Disponível em: <<https://innolution.com/blog/how-safe-is-your-agile-environment>>.

Figura 6. Disponível em:

<<https://www.bitcoininsider.org/article/15963/5-scrum-masters-their-best-advice-leading-high-performing-teams>>.

Figura 7. Disponível em:

<<https://www.fiverr.com/arsalanxafar/be-your-scrum-master-agile-project-manager>>.

Figura 8. Disponível em:

<<http://www.agilebuddha.com/agile/agile-offshore-business-analyst-complementing-the-role-of-product-owner/>>.

Figura 9. Disponível em: <<https://www.infoworks-tn.com/daily-scrum/>>.

Figura 10. Disponível em: <<http://roneringa.com/leading-scrum-teams-to-maturity/>>.





Imagens:

Figura 11. Disponível em: <<http://scrumbook.org/value-stream/sprint/sprint-retrospective.html>>.

Figura 12. Disponível em: <<http://loonslab.com/2017/05/02/common-problems-with-the-daily-stand-up/>>.

Figura 13. Disponível em:

<<https://medium.com/the-liberators/myth-the-scrum-master-cant-remove-people-from-the-scrum-team-92fba0ad391b>>.

Figura 14. Disponível em: <<http://roneringa.com/leading-scrum-teams-to-maturity/>>.

Figura 15. Disponível em: <<https://controlyourchaos.wordpress.com/>>.

Figura 16. Disponível em

<<http://litterale.com.br/2018/04/12/estudo-de-caso-devido-a-um-conflito-mal-resolvido-dois-diretores-pedir-am-demissao/>>

