FACULDADE UNIGUAÇU

IGUAZU SCIENCE

EDITORA UNIVERSITÁRIA UNIGUAÇU

v. 3, n. 7, jun. 2025
ISBN 978-65-83057-12-9

ARTIGO CIENTÍFICO ACESSO LIVRE

METODOLOGIAS ÁGEIS PARA O DESENVOLVIMENTO DE SOFTWARE

Kaio Rodrigo dos Santos Britzke*; Henderson Augusto Gasparin*; Leonardo Gomes Guidolin**

*Acadêmico de Engenharia de Software, *kaiorodrigobritzke@gmail.com; hendersonaugusto5@gmail.com*.
**Mestre em Tecnologias Computacionais pro Agronegócio – UTFPR Medianeira, *leonardo.gguidolin@gmail.com*.

INFORMAÇÕES

Histórico de submissão:

Recebido em: 20 maio 2025
Aceite: 10 jun. 2025
Publicação *online*: jun 2025

RESUMO

Há diversas metodologias ágeis surgindo nos últimos anos, e cada vez mais se entra em discussão sobre a validade e efetividade delas, muitas empresas já as adotaram e, boa parte, relata ter ótimos resultados. Este trabalho visa detalhar algumas das principais metodologias ágeis (Scrum, XP, FDD, Lean Software Development e Crystal Family) e fazer a comparação dos processos das que mais estão sendo utilizadas hoje em dia (Scrum, XP e FDD), além de relatar alguns casos de sucesso e fracasso das metodologias ágeis em geral. Dessa forma, será possível conhecer essas metodologias além de identificar os pontos que tornam cada uma delas interessantes e também verificar a efetividade que está sendo relatada por quem as utiliza.

Palavras-chave: Metodologias Ágeis; Desenvolvimento de Software; Scrum; Extreme Programming; Gestão de Projetos; Engenharia de Software.

ABSTRACT

There have been several agile methodologies emerging in recent years, and there has been increasing discussion about their validity and effectiveness. Many companies have already adopted them, and many of them report excellent results. This paper aims to detail some of the main agile methodologies (Scrum, XP, FDD, Lean Software Development, and Crystal Family) and compare the processes of those most widely used today (Scrum, XP, and FDD), in addition to reporting some success and failure cases of agile methodologies in general. This way, it will be possible to learn about these methodologies, identify the points that make each of them interesting, and also verify the effectiveness reported by those who use them.

Keywords: Agile Methodologies; Software Development; Scrum; Extreme Programming; Project Management; Software Engineering.

Citação: BRITZKE, Kaio Rodrigo dos Santos; GASPARIN, Henderson Augusto; GUIDOLIN, Leonardo Gomes. Metodologias ágeis para o desenvolvimento de Software. **Iguazu Science**, São Miguel do Iguaçu, v. 3, n. 7, p. 135-144, jun 2025.

INTRODUÇÃO

A era da computação iniciou-se nos anos 40, e os maiores investimentos eram voltados a hardware. No início dos anos 50, passou-se a ter o domínio da tecnologia de hardware e então os investimentos se voltaram ao desenvolvimento dos sistemas operacionais o que possibilitou o surgimento das primeiras linguagens de programação de alto nível, permitindo assim que usuários pudessem se concentrar mais no desenvolvimento sem se preocuparem com questões técnicas do funcionamento do hardware. O surgimento de sistemas operacionais com multiprogramação, no início dos anos 60, possibilitou um aumento na eficiência destes, contribuindo para a queda de preço dos hardwares. Com o desenvolvimento dos hardwares e dos sistemas operacionais, passou-se a haver a necessidade de sistemas mais complexos e maiores em substituição aos pequenos aplicativos que existiam até o momento e foi aí que se iniciou a "crise do software" pela incompatibilidade dos métodos utilizados até então com os métodos necessários para a criação desses sistemas (Pressman, 2011).

Em 1968 foi realizada uma conferência pelo Comitê de Ciência da NATO (North Atlantic Treaty

Organization) com o nome “Engenharia de Software” e foi aonde, pela primeira vez, foi utilizado esse termo. Nessa conferência foi discutido sobre a existência de uma real crise de software, e o que se constatou foram, ao menos, os seguintes problemas: cronogramas não observados, projetos com tantas dificuldades que são abandonados, módulos que não operam corretamente quando combinados, programas que não fazem exatamente o que era esperado, programas tão difíceis de usar que são descartados, programas que simplesmente param de funcionar. E foi a partir desse cenário que passou a se desenvolverem os processos e metodologias de desenvolvimento de software (Pressman, 2011).

Um dos primeiros processos, que surgiu em 1970, foi o que ficou conhecido como processo em cascata, ele possui sete fases e uma só poderia ser iniciada após o término da anterior, daí o nome, isso acaba causando alguns transtornos, afinal em muitas vezes o cliente gostaria de fazer alguma mudança no sistema, porém descobre isso quando a fase de requerimentos já passou o que acaba deixando o projeto defasado ou mais caro por não prever de início esse tipo de mudança. Nos anos 80, surgiu outro processo para desenvolvimento de software, o processo em espiral, que apareceu como a solução para os problemas existentes no modelo em cascata, a exemplo do “ciclo de Demming” (ciclo PDCA), esse modelo surgiu com apenas quatro fases (Planejamento, Avaliação, Análise de Risco e Engenharia), o processo inicia no planejamento, vai para a avaliação, análise de risco, engenharia e, seguindo a ideia do espiral, volta para o planejamento fazendo todo o ciclo novamente, idealizando assim um modelo iterativo e incremental, permitindo que os erros ocorridos em uma fase possam ser revistos (Pressman, 2011).

Ainda assim, com o passar do tempo, a complexidade dos sistemas tem aumentado ainda mais, os sistemas possuem mais usuários e tornam-se cada vez mais importantes, podendo, em muitas vezes, gerenciar a vida ou a morte de pessoas, como é o caso de softwares que calculam quantidades de medicamentos a serem aplicados em pacientes ou que monitoram o estado de saúde dos pacientes, os sistemas também controlam a economia e as armas do mundo, é possível ver isso através do pânico que foi causado pela possibilidade do “bug” do milênio no final dos anos 90. Unindo-se aos problemas até então encontrados e à importância atual dos sistemas informatizados, surgiram alguns teóricos discordando da ideia de tratar o desenvolvimento de software como uma fábrica de produção em série, um exemplo é Cockburn que compara o desenvolvimento de software ao ato de escrever uma poesia épica em conjunto com diversas pessoas: seriam diversas pessoas, com diversos argumentos, tentando dar seu melhor sem talento, tempo ou recursos suficientes. Foi a partir desses pensamentos que surgiram as metodologias ágeis de desenvolvimento, a partir de 2001, com a assinatura do manifesto ágil, que estabeleceu os princípios das metodologias ágeis (Pressman, 2011).

O objetivo desta pesquisa é analisar e comparar as principais metodologias ágeis utilizadas no desenvolvimento de software, como Scrum, Extreme Programming (XP), Feature Driven Development (FDD), Lean Software Development e Crystal Family. Além de detalhar os princípios e processos de cada abordagem, o estudo busca compreender como essas metodologias contribuem para a melhoria dos resultados nos projetos de software, bem como identificar fatores de sucesso e causas comuns de falhas em sua adoção. Por meio de levantamento bibliográfico e análise de casos práticos, pretende-se fornecer uma visão abrangente e crítica sobre a aplicabilidade das metodologias ágeis no cenário atual da engenharia de software.

METODOLOGIA

Essa pesquisa científica foi feita a partir de pesquisas em sites, leitura de artigos, leitura de matérias e palestras online sobre como as metodologias ágeis no desenvolvimento de software vem evoluindo nos últimos anos e como isso colabora para o mesmo. Para que eu conseguisse chegar em um nível de pesquisa interessante e obtivesse conhecimento do mesmo, ao ler artigos, pesquisar sites, ler matérias e assistir palestras online eu estava sempre anotando e fixando os pontos que eu achava mais importante ou relevantes para a minha pesquisa. Após realizar todos esses procedimentos, coletar partes importantes de artigos e matérias, fui montando a pesquisa com esses principais pontos que coletei ou anotei durante a fase de estudos.

SCRUM

História

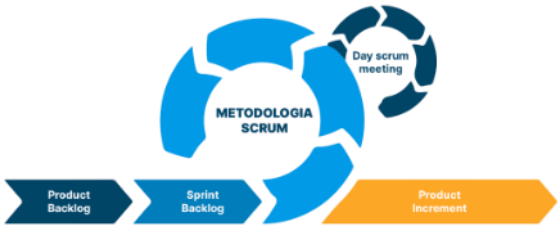
No início dos anos 90, mais precisamente em 1991, Peter DeGrace e Leslie Hulet Stahl, utilizaram o termo “abordagem Scrum” para fazer referência aos processos e métodos, outrora descritos por Takeuchi e Nonaka, em seu artigo "Wicked Problems, Righteous Solutions - A Catalogue of Modern Software Engineering Paradigms” (Problemas cruéis, soluções justas – Um catálogo dos paradigmas da engenharia de software moderna). Ainda nos anos 90, Ken Schwaber começa a utilizar essa abordagem em sua empresa, Jeff Sutherland também desenvolve uma abordagem similar, a qual se refere apenas como “Scrum”. A apresentação oficial do Scrum, bem como sua publicação ocorreu em 1995, na OOPSLA (Object-Oriented Programming, Systems, Languages & Applications – Programação, Sistemas, Linguagens e Aplicações orientadas a objeto), e então passaram a haver atualizações constantes com contribuições

significativas nos cinco anos seguintes de Mike Beadle e Martine Devos (Schwaber, 2011).

Valores

Conforme Schwaber e Sutherland (2011), ele é baseado nas teorias empíricas de controle de processo e, como tal, acredita que o conhecimento vem da experiência e da tomada de decisões baseadas naquilo que é conhecido. Assim como na maioria das metodologias ágeis, o Scrum prega um processo iterativo e incremental para o desenvolvimento de um projeto, de forma que se torne mais previsível e tenha riscos mais controláveis. Dentre os diversos valores e regras que regem o Scrum, há três que são mais importantes: Transparência: Todos os aspectos mais relevantes do processo tem que estar ao alcance dos responsáveis pelos resultados, sendo esses definidos de uma forma que estejam claros e causando o mesmo entendimento para estes, ou seja, todos devem "falar a mesma língua". Inspeção: Os usuários da metodologia devem verificar, com certa frequência, se os mecanismos Scrum e o progresso do projeto para identificar qualquer andamento que não esteja indo de acordo com o esperado. O ideal é que tal verificação seja feita por inspetores especializados em tal trabalho para que tenha um resultado melhor e também para que não venha a causar prejuízo de tempo àqueles que deveriam estar executando outras tarefas. Adaptação: Caso seja identificado, durante a inspeção, que os aspectos de algum processo foram desviados além do que é aceitável, assim o será o produto, portanto o processo deve ser adaptado o quanto antes para eliminar possíveis futuros desvios. Há quatro oportunidades formalmente descritas pelo Scrum em que a inspeção e a adaptação podem ser aplicadas: Reunião de planejamento do Sprint, Reunião diária (Daily Scrum), Reunião de revisão do Sprint, Retrospectiva do Sprint (Schwaber, 2011). Na figura abaixo (Figura 1) ela está demonstrando como funciona o fluxo simples de um Scrum totalmente cru.

Figura 1. Infográfico Scrum



Fonte: BLOG, 2022

Papeis

Os times Scrum foram feitos para se organizarem sozinhos e terem diversas funções. A ideia é que uma equipe que se organiza por si só saiba definir de forma melhor como completarem seu trabalho em relação a quando são dirigidos por uma pessoa de fora. Quanto às diversas funções, a ideia é que todas as habilidades necessárias para resolver todas as questões que surgirem estejam dentro da equipe, sem a dependência de outros que não participam ativamente da equipe. Os profissionais integrados a esse time são divididos em três papeis:

Product Owner ("Dono do produto")

Ele é o responsável por fazer com que o produto tenha o maior valor possível. As estratégias para que esse objetivo seja atingido podem ser as mais variadas possíveis e devem levar em consideração os times Scrum, suas organizações e os indivíduos que fazem parte. É também o Product Owner o responsável por atualizar e gerenciar o Backlog do produto (termo explicado a frente), o que inclui, conforme o manual do Scrum:

- Expressar claramente os itens do Backlog do Produto;
- Ordenar os itens do Backlog do Produto para alcançar melhor ascmetas e missões;
- Garantir o valor do trabalho realizado pelo Time de Desenvolvimento;
- Garantir que o Backlog do Produto seja visível, transparente, claro para todos, e mostrar o que o Time Scrum vai trabalhar a seguir; e,
- Garantir que a Equipe de Desenvolvimento entenda os itens do Backlog do Produto no nível necessário (Schwaber, 2011).

É importante ressaltar que o Product Owner é responsável por esses itens, porém não necessariamente precisa executá-los, podendo delegá-los, por exemplo, à equipe de desenvolvimento. É de extrema importância para o bom desenvolvimento e sucesso do produto que a figura do Product Owner seja respeitada bem como suas decisões, ninguém poderá alterar prioridades ou dar ordens à equipe de desenvolvimento sem passar por ele, assim como a equipe de desenvolvimento não pode acatar a alguma ordem que não venha dele (Schwaber, 2011).

Equipe de Desenvolvimento

A equipe de desenvolvimento é responsável pela realização das tarefas "executáveis" do Backlog, deixando-as "prontas" e entregando uma versão incrementada utilizável do produto ao final de cada Sprint, ela é estruturada e tem autonomia de organizar e gerenciar seu próprio trabalho como considerar melhor. Conforme Schwaber e Sutherland (2011), uma equipe de desenvolvimento deve possuir as seguintes características:

"- Elas são auto-organizadas. Ninguém (nem mesmo o Scrum Master) diz à Equipe de

Desenvolvimento como transformar o Backlog do Produto em incrementos de funcionalidades potencialmente utilizáveis;

- Equipes de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto.
- O Scrum não reconhece títulos para os integrantes da Equipe de Desenvolvimento que não seja o Desenvolvedor, independentemente do trabalho que está sendo realizado pela pessoa; Não há exceções para esta regra.
- Individualmente os integrantes da Equipe de Desenvolvimento podem ter habilidades especializadas e área de especialização, mas a responsabilidade pertence à Equipe de Desenvolvimento como um todo;
- Equipes de Desenvolvimento não contém sub equipes dedicadas a domínios específicos de conhecimento, tais como teste ou análise de negócio."

Para o bom funcionamento da metodologia, é recomendado que as equipes de desenvolvimento possuam entre três e nove integrantes, menos que isso pode significar uma menor interação que pode resultar menor produtividade, além de, por possuir poucos integrantes, pode apresentar uma deficiência nas habilidades necessárias para que uma versão seja entregue realmente funcional. Mais de nove integrantes em uma equipe de desenvolvimento pode gerar uma complexidade acima da capacidade de um processo empírico gerenciar (Schwaber, 2011).

Scrum Master

O Scrum Master é algo parecido com um auditor da metodologia, o qual fará parte do time para assegurar que a teoria, a prática e as regras do Scrum estão sendo aplicadas de forma correta. Além desse papel de auditor, o Scrum master é o elo entre o time e o mundo exterior, permitindo que apenas ocorram interações que de fato agreguem valor ao produto.

Suas interações com o Product Owner ocorrem, entre outras maneiras, das seguintes:

- "Encontrando técnicas para o gerenciamento efetivo do Backlog do Produto;
- Claramente comunicar a visão, objetivo e itens do Backlog do Produto para a Equipe de Desenvolvimento;
- Ensinar a Time Scrum a criar itens de Backlog do Produto de forma clara e concisa;
- Compreender a longo-prazo o planejamento do Produto no ambiente empírico;
- Compreender e praticar a agilidade; e,
- Facilitar os eventos Scrum conforme exigidos ou necessários." (Schwaber, 2011)

Já quando se trata da equipe de desenvolvimento, essas interações ocorrem, entre outras maneiras, das seguintes:

- "Treinar a Equipe de Desenvolvimento em autogerenciamento e interdisciplinaridade;
- Ensinar e liderar a Equipe de Desenvolvimento na criação de produtos de alto valor;
- Remover impedimentos para o progresso da Equipe de Desenvolvimento;
- Facilitar os eventos Scrum conforme exigidos ou necessários;
- Treinar a Equipe de Desenvolvimento em ambientes organizacionais nos quais o Scrum não é totalmente adotado e compreendido." (Schwaber, 2011)

Enfim, com a organização, as interações, devem ocorrer das seguintes maneiras, entre outras:

- "Liderando e treinando a organização na adoção do Scrum;
- Planejando implementações Scrum dentro da organização;
- Ajudando funcionários e partes interessadas a compreender e tornar aplicável o Scrum e o desenvolvimento de produto empírico;
- Causando mudanças que aumentam a produtividade do Time Scrum;
- Trabalhando com outro Scrum Master para aumentar a eficácia da aplicação do Scrum nas organizações." (Schwaber, 2011)

Eventos Scrum

Para uma melhor organização, o Scrum se divide em eventos com duração máxima estabelecida (Schwaber, 2011).

Sprint

O Sprint é um dos principais componentes do Scrum, esse evento tem como missão entregar uma versão incremental utilizável ao seu final, para que isso ocorra, o Sprint deve possuir uma definição do que deve ser construído e um plano para que isso ocorra. Ele possui duração máxima de um mês, afinal em um tempo maior que esse poderia ocorrer mudanças no que deveria ser construído, aumentando a complexidade e o risco, além disso, se houver qualquer problema apenas o orçamento de um mês estará comprometido e não todo ele. Conforme Schwaber e Sutherland (2011) há algumas regras que devem ser seguidas para que um Sprint tenha o andamento correto:

- "Não são feitas mudanças que podem afetar o objetivo do Sprint;
- A composição da equipe de desenvolvimento permanece constante;

- As metas de qualidade não diminuem; e,
- O escopo pode ser clarificado e renegociado entre o ProductOwner e a equipe de desenvolvimento quanto mais for aprendido."

Um Sprint pode ser cancelado a qualquer momento, antes de sua finalização, pelo Product Owner, que normalmente o faz quando a organização muda sua direção ou quando as condições do mercado ou da tecnologia mudam, fazendo com que o objetivo do Sprint se torne obsoleto. Há quatro outros eventos em que se divide o Scrum, são eles: reunião de planejamento do Sprint, reuniões diárias, revisão do Sprint e retrospectiva do Sprint (Schwaber, 2011).

Reunião de planejamento do Sprint

Todo o time Scrum participa da reunião de planejamento do Sprint, que deve durar, no máximo, 8 horas (considerando um Sprint de um mês) para responder a duas perguntas: "O que ficará 'pronto' nesse Sprint?" e "Como ficará 'pronto' o que foi escolhido para esse Sprint?". O objetivo é que ao final da reunião a equipe de desenvolvimento possa explicar ao Product Owner e ao Scrum Master como pretende trabalhar de forma auto-organizada para atingir o objetivo do Sprint e criar um incremento previsto (Schwaber, 2011).

O que ficará 'pronto' nesse Sprint?

A equipe de desenvolvimento faz uma previsão de tudo que conseguirá desenvolver durante o Sprint levando em consideração e selecionando os itens do Backlog do produto. Feito isso o time Scrum determina a meta do Sprint, que é o objetivo, dentro do Sprint, da implementação de cada item selecionado do Backlog do produto (Schwaber, 2011).

Como ficará "pronto" o que foi escolhido para esse Sprint?

A equipe de desenvolvimento deve determinar como executará e quando entregará cada um dos itens selecionados para a entrega da versão ao fim do Sprint, o conjunto desse plano de entrega com os itens selecionados é chamado de Backlog do Sprint.

É interessante que durante essa parte da reunião o Product Owner esteja presente para negociar, junto à equipe de desenvolvimento, o que entrará no Sprint

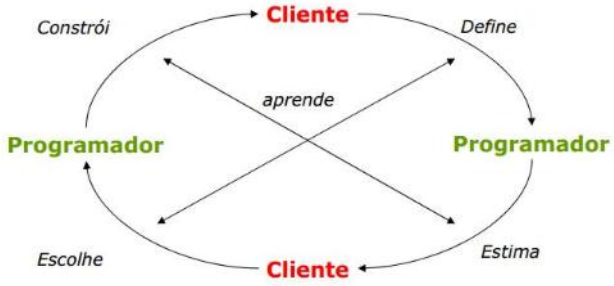
e o que pode ser trocado, além de ajudar a esclarecer quaisquer itens que não estejam tão claros quanto deveriam (Schwaber, 2011).

XP (Extreme Programming)

História

Os principais colaboradores e criadores da XP são Kent Beck e Ward Cunningham, diversas das principais características da XP foram herdadas da comunidade que ambos faziam parte, a comunidade SmallTalk (linguagem de programação orientada a objetos criada em meados dos anos 1960), características como adaptação à mudança, desenvolvimento iterativo, ênfase nos testes, integração contínua, programação pareada e refatoração. No ano de 1999, foi a primeira vez em que o termo "XP" (extreme programming) foi utilizado, quando o programador Kent Beck trabalhava como líder de um projeto na "Chrysler Comprehensive Compensation", um projeto de longo prazo que visava reescrever a aplicação de folha de pagamento da Chrysler Corp. Segundo Wells (1999), certo dia Beck se perguntou "Há algumas atividades que contribuem para o sucesso do desenvolvimento de software, o que aconteceria se as fizéssemos o mais intensamente possível?" e daí veio o nome "Programação Extrema". Após essa experiência, em 1999, Beck lançou o primeiro livro sobre XP, "Extreme Programming: Explained: Embrace Change", o qual recebeu o prêmio JOLT de produtividade da revista "Software Development" (Wells, 1999). Na imagem abaixo (Figura 2) está apresentando como funciona a metodologia aplicada dentro de um processo empresarial, que é basicamente programação nua e crua, o mais intenso possível.

Figura 2. Infográfico Extreme Programming



Fonte: DEVMEDIA, 2022

Valores

Os valores aplicados pelo XP são nada mais do que uma forma de trabalhar em harmonia com a equipe e com os valores da empresa. Ele ainda permite que, além de seus valores padrões, cada um que for aplicar a metodologia incorpore seus valores a ela, conforme as mudanças que forem feitas nas regras do XP. Os valores padrões são:

Simplicidade: Não se deve ser feito nada além do que foi solicitado, afinal se deve trabalhar sobre a solicitação pela qual foi feita o investimento, isso aumentará o valor gerado até a data final. Dessa forma, pode-se aos poucos caminhar rumo ao objetivo analisando calmamente cada uma das falhas que ocorrerem durante o percurso (Wells, 1999).

Comunicação: Os membros da equipe devem trabalhar próximos, comunicando-se diariamente e pessoalmente, todos contribuindo desde o levantamento de requisitos até a codificação e entrega ao cliente (Wells, 1999).

Retorno: Devem ser entregues versões funcionais do programa periodicamente. Essas versões periódicas serão demonstradas para que todas as alterações nos processos ou no projeto sejam feitas conforme a necessidade (Wells, 1999).

Respeito: Todos os membros da equipe devem ser igualmente respeitados, assim como seus conhecimentos. Dessa forma o cliente deve respeitar os desenvolvedores acerca de seus conhecimentos técnicos, assim como os desenvolvedores devem respeitar o cliente quanto aos seus conhecimentos do negócio (Wells, 1999).

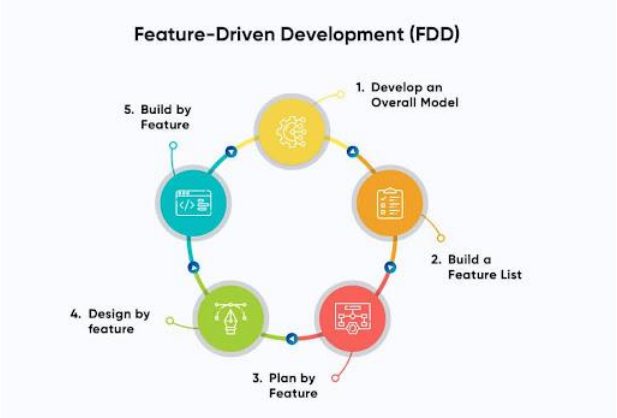
Coragem: Sempre se deve dizer a verdade e trabalhar em cima dela. Deve-se ter a coragem também de fazerem as mudanças necessárias para que o projeto tenha sucesso (Wells, 1999).

Feature Driven Development (FDD)

História

O FDD (Desenvolvimento dirigido a funcionalidade) foi criado ao longo de um projeto, em 1997, de desenvolvimento de software em Singapura que contou com Jeff de Luca como gerente de projeto, Peter Coad como arquiteto chefe e Stephen Palmer como gerente de desenvolvimento. Era um projeto ambicioso, porém complexo e que já havia falhado uma vez, o que deixou os usuários céticos e uma equipe de desenvolvimento desmoralizada e frustrada. A partir do desenvolvimento e utilização dos processos desta metodologia no projeto, os funcionários passaram a se divertir com o que estavam aplicados e o trabalho passou a fluir muito melhor, e na Figura 3 isso fica claro como funciona o FDD aplicado, que seria basicamente você entregar pequena features ao cliente ou pequenas funcionalidades de acordo com o que você vai desenvolvendo (Palmer, 2002).

Figura 3. Infográfico Feature-Driven Development



Fonte: CDN, 2024

Abordagem

Segundo Palmer e Felsing (2002), muitos dos processos existentes até a época focavam em documentações e outras coisas que faziam as pessoas se importarem muito mais em seguir os processos descritos na metodologia utilizada do que realmente no desenvolvimento do software. Para ele, um desenvolvimento bem feito não deve ter diversos processos que tem que ser consultados diversas vezes durante todo o processo, mas sim apenas alguns que consigam ser decorados e, assim, enraizados no desenvolvedor, além de muita comunicação. O FDD foi pensado para projetos de software de tamanho moderado a grande e ele tem como foco uma "característica" que, neste caso, é uma função valorizada pelo cliente e que pode ser implementada em uma semana ou menos. Essas características são pequenos blocos de funcionalidades que serão entregues periodicamente (em iterações), sendo assim, é mais fácil para os usuários descrevê-las, revisá-las e relacioná-las. Como se tratam de pequenos blocos, os projetos e codificações são mais simples e fáceis de monitorar (Palmer, 2002).

Lean Software Development

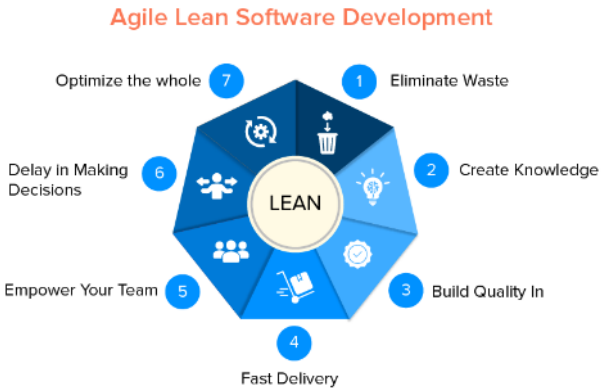
História

A "Lean Software Development" (Desenvolvimento "Magro" de Software) tem sua origem, como o nome sugere, nas "lean techniques" (técnicas "magras") do ambiente industrial sendo aplicadas ao desenvolvimento de software. Mary Poppendieck, que idealizou a metodologia, juntamente com Tom Poppendieck, foi programadora por diversos anos, até se deparar com uma oportunidade e seguir a carreira de gestora de TI. Após alguns anos, ao voltar ao cenário de desenvolvimento de software, deparou-se com algo que considerou assustador: uma ênfase em definição e detalhamento de processos sendo justificada pelo movimento de manufatura "magra" (o qual ela acompanhou na própria indústria). Como sabia que as técnicas que estavam sendo aplicadas eram exatamente o oposto do que a manufatura "magra" pregava ("um fluxo rápido e essencial", "as pessoas devem pensar mais para fazerem seus trabalhos", "testar incessantemente é a melhor forma de fazer as coisas funcionarem corretamente"), Mary resolveu tentar mudar o paradigma do desenvolvimento de software escrevendo um livro que explica a forma correta de aplicar as técnicas "magras" ao desenvolvimento de software como cita a Figura 4 (Poppendieck, 2003).

Abordagem

O desenvolvimento de software é algo muito abrangente, indo desde "Web Design" até o envio de um satélite em órbita, e sendo assim tão abrangente e variável, há diversas práticas que podem ser aplicadas a alguns casos e a outros não. Esta metodologia cuida de aplicar as práticas corretas para cada caso levando em conta os princípios "magros" (Poppendieck, 2003).

Figura 4. Infográfico Lean Software Development



Fonte: MEDIA, 2025

Crystal Family

História

A família "Crystal" foi desenvolvida a partir de uma necessidade da IBM em 1991. Ela precisava de uma metodologia "object-technology" e para isso contratou Alistair Cockburn como seu desenvolvedor.

Cockburn iniciou o processo de desenvolvimento da metodologia a partir de entrevistas com times de projetos e percebeu que as práticas utilizadas eram bem diferentes das descritas nos livros da época, como o uso excessivo de comunicação próxima e direta, moral, diálogo com o usuário final, etc. Pelo que pode perceber, esses aspectos diferenciavam os projetos que obtinham sucesso dos que não. Utilizando-se dessas técnicas, Cockburn foi consultor de um projeto com preço e escopo fixo, e as teve como principal fator de sucesso. Partindo das lições aprendidas com a experiência deste e dos projetos que participou posteriormente (Cockburn, 2004).

Aspectos

"Crystal" é uma família de metodologias com um código genético comum, que tem ênfase em entrega frequente, comunicação próxima e melhoria recíproca. Deve-se lembrar de que "Crystal" não é simplesmente uma metodologia, mas sim uma família, portanto existe uma metodologia diferente para cada tipo de projeto, sempre utilizando o "código genético" da família. O nome "Crystal" parte de duas dimensões dos projetos: tamanho e criticidade, correspondendo, respectivamente, as propriedades cor e rigidez do mineral (cristal). Projetos maiores, que exigem maior coordenação e comunicação, correspondem a cores mais escuras (branco, amarelo, laranja, vermelho e assim por diante). Projetos para sistemas que possuem um maior número de regras de validação e verificação correspondem a cristais mais rígidos. A metodologia quartzo é utilizada para sistemas inofensivos com poucos desenvolvedores, já um mesmo time criando um sistema para um reator nuclear exige uma metodologia diamante, demandando assim por verificações repetitivas tanto no design quanto na implementação dos algoritmos (Cockburn, 2004).

Crystal Clear

A "Crystal Clear" é a variação mais conhecida da família Crystal, trata-se de um aperfeiçoamento que pode ser aplicado em equipes que contenham de 2 a 8 pessoas, instaladas em uma mesma sala ou escritórios adjacentes. A comunicação próxima é reforçada, ocorrendo de uma forma natural e quase imperceptível, afinal os integrantes acabam participando como ouvintes de discussões que não fariam parte (Cockburn, 2004).

Dextra

Uma empresa brasileira, a Dextra, relatou, em 2010, ter alcançado cem mil horas de projetos com metodologias ágeis após ter sido a pioneira na adoção do Scrum no Brasil e afirma ter obtido resultados impressionantes. Dentre esses projetos, destacam-se sistemas para a GLOBOSAT, a Força Aérea Brasileira e o Grupo Confidence, nos quais a empresa afirma ter obtido sucesso e satisfação totais de seus clientes. Na comunidade ágil, há um caso famoso de um projeto que foi "salvo" pelas metodologias ágeis, o projeto um sistema de gerenciamento de casos ("Sentinela" - o qual passou a ser desenvolvido após os atentados da cidade de Oklahoma). Houve um projeto desenvolvido, inicialmente, com metodologias tradicionais, o qual foi cancelado em 2005 com um orçamento de cento e setenta milhões de dólares. Após esse fracasso, foi feita uma nova tentativa utilizando ainda a metodologia em cascata, agora com o orçamento de quatrocentos e vinte e cinco milhões de dólares, em 2010 foi verificado que o orçamento estouraria em trezentos e trinta e um milhões de dólares e o prazo aumentaria mais seis anos, ainda com a possibilidade de o sistema não estar completo.

Nesse momento, Chad Fulgham, o CIO, decidiu pausar o projeto e passar a utilizar o Scrum. Para isso ele reduziu a equipe de desenvolvimento para cinquenta e cinco pessoas (antes havia cento e vinte e cinco) e organizou todo o trabalho em 60 "histórias de usuários" e determinou a quantidade dos Sprints que seriam realizados e a duração de cada um. Como resultado dessa mudança, o projeto foi entregue e disponibilizado para todos os funcionários do FBI em primeiro de julho de dois mil e doze, conforme memorando publicado no site do FBI (Dextra, 2012).

Casos de fracasso

A maior parte dos casos de fracasso que é possível encontrar nos sites e livros é relatada por defensores das metodologias ágeis, ou seja, em todos esses casos há alguma falha na utilização dos métodos por parte dos fracassados, ou algum tipo de resistência apresentada pela equipe. Ainda assim, esses casos podem, ao menos, deixar claros os erros mais comuns

que ocorrem e dá uma ideia de como impedi-los de acontecerem.

Blass - Projeto canon

Essa empresa foi extremamente ambiciosa no Projeto Canon, tendo sido criada no mês de Julho, comprometeu-se com a entrega completa de um site

totalmente funcional para o dia trinta de setembro. Após um dia de apresentação acerca da metodologia XP por Laurent Bossavit (autor do caso) a empresa decidiu utilizar a metodologia para parte do projeto. Inicialmente, essa iniciativa mostrou-se efetiva, permitindo até mesmo que algumas iterações fossem entregues sem grandes problemas, porém os verdadeiros obstáculos apareceram quando foi necessário expandir o processo para o restante do projeto após serem verificados diversos problemas de qualidade nessas outras partes, além do fato de o cliente não estar ciente de tudo isso que estava acontecendo (as diferenças nos processos de trabalho entre as equipes e a consequente falha na qualidade e nos prazos). Basicamente, o autor resume o fracasso desse projeto em três características que conflitaram, em parte, com os princípios do XP: paixão, ousadia e glamour. A paixão fez com que acreditassem poder desenvolver projetos grandes para grandes companhias. A ousadia fez com que aceitassem um projeto muito complexo em um prazo extremamente curto e irreal. Finalmente o glamour fez com que dessem mais atenção a "perfumarias" em vez de realmente se preocuparem com a estrutura interna do sistema, além de fazer com que deixassem a coragem (um dos valores pregados pelo XP) de lado para acomodarem-se em linguagens e sistemas de gerenciamento de banco de dados tradicionais em vez de realmente procurarem o que melhor os atenderia para o projeto em questão. Ainda assim, o que acabou fazendo com que o fracasso fosse iminente foi a má administração da empresa, que acabou deixando um pouco de lado a opinião e o pensar dos colaboradores, para tentar resolver tudo "de cima" (algo não recomendado por todas as metodologias ágeis).

Caso de sucesso

Após falarmos sobre todas as metodologias de desenvolvimento citadas acima, agora irei citar um caso de sucesso de uma empresa que aderiu à modalidade "Scrum" como metodologia para seu desenvolvimento. Uma empresa de tecnologia financeira que pertence a um grupo empresarial, ou um conglomerado paranaense com atuação em educação, cooperativas de crédito e agricultura familiar. Fundada a partir de um projeto acadêmico na Universidade Tecnológica Federal do Paraná (UTFPR) em Medianeira, consolidou-se como um caso de sucesso ao oferecer soluções digitais inovadoras para o setor de crédito cooperativo. A metodologia Scrum aplicada a essa empresa funciona mais ou menos da seguinte maneira:

- O Cliente entra em contato com o suporte, suporte coleta a necessidade do cliente e modula um "protótipo" do que eles querem que seja desenvolvido.
- Suporte faz a abertura de um chamado descrevendo a necessidade e anexando esse "protótipo" ao mesmo, encaminhando isso para o Product Owner (P.O) de desenvolvimento.
- O P.O faz a análise de requisitos gerais do chamado e sabe para qual desenvolvedor encaminhar a demanda.
- O desenvolvedor realiza o desenvolvimento, faz o deploy das alterações que foram feitas em um ambiente de testes e envia a demanda juntamente com o chamado para os testers
- O testers realiza os testes e se estiver de acordo com o que o cliente pediu, a demanda é enviada novamente para o P.O para que ele possa fazer o deploy das alterações no ambiente de produção, que é onde o cliente vai poder usufruir do desenvolvimento feito.

Abaixo irei demonstrar na Figura 5 como mais ou menos funciona o procedimento e a esteira de desenvolvimento usando o Scrum como metodologia

Figura 5: Exemplo de Scrum aplicado

Fonte: Autoria própria.

RESULTADOS E DISCUSSÃO

Os resultados observados na aplicação das metodologias ágeis demonstram ganhos significativos em termos de flexibilidade, qualidade do produto e satisfação do cliente. Empresas como a Dextra reportaram melhorias na entrega de valor ao cliente após a adoção do Scrum, alcançando centenas de milhares de horas de desenvolvimento com sucesso comprovado. Casos como o projeto Sentinel do FBI reforçam a eficácia das práticas ágeis em contextos complexos e críticos. Entretanto, a pesquisa também evidencia que falhas na implementação, resistência cultural e falta de alinhamento com os princípios ágeis podem comprometer os resultados, como exemplificado no caso da Blass com o projeto Canon. O estudo reforça que a adoção bem-sucedida das metodologias ágeis requer não apenas conhecimento técnico, mas também uma mudança organizacional.

CONCLUSÕES

A intenção deste trabalho foi demonstrar algumas das diversas metodologias ágeis que já surgiram e os diversos processos que elas propõem não apenas para administrar um projeto, mas sim para toda uma forma diferente de se trabalhar e valorizar o profissional. O XP é uma das metodologias ágeis mais completas se considerarmos todas as linhas que suas regras abrangem, mencionando desde o relacionamento com o cliente, até o desenvolvimento do sistema e sua entrega, o Scrum não possui tantos detalhamentos como o XP (não especifica processos para a codificação em si, por exemplo), porém passou a ser uma das metodologias mais utilizadas, afinal suas regras para a administração de um projeto são bem definidas e por não terem especificações tão "fortes" em algumas áreas (como é o caso da mencionada codificação) enfrenta menos resistência para ser implantada, além de poder ser utilizada com mais facilidade em qualquer tipo de projeto (não apenas em desenvolvimentos de sistemas), enfim, o Scrum pode servir como a porta de entrada menos traumática para as metodologias ágeis. Assim como o XP, o FDD também é muito abrangente em suas regras, detalhando minuciosamente cada etapa do desenvolvimento do projeto e, portanto, pode passar pelos mesmos problemas do XP. Apesar de terem muito em comum, não há uma metodologia ágil que seja boa para todos, aqueles que desejam utilizá-las devem estudar o cenário e as culturas em que serão implantadas e então tentar identificar qual melhor se encaixa.

É importante ressaltar também que não necessariamente devem-se seguir todas as regras de apenas uma metodologia ágil, podendo, como acontece muito com o Scrum, mesclar regras e princípios de diversas metodologias ágeis, conforme o que melhor se adaptar para o ambiente em que for implantado. Se compararmos com o tempo que utilizamos as metodologias tradicionais e o surgimento das metodologias ágeis, realmente elas ainda possuem um longo caminho a percorrer antes de dar um veredito sobre a sua real efetividade, porém devem ser observados os resultados (assertividade de prazos, satisfação do cliente, satisfação dos colaboradores) que as empresas que as utilizam relatam e também a quantidade de pessoas e empresas que estão seguindo essa tendência. É importante lembrar que já houve uma quebra de paradigma em outro momento no mundo do desenvolvimento de sistemas no momento em que se passou a utilizar mais a orientação a objeto do que a programação estruturada, e se deve lembrar que essa existia desde meados dos anos 60.

REFERÊNCIAS

BECK, Kent *et al*. **Manifesto para Desenvolvimento Ágil de Software.** 2001. Disponível em: http://www.agilemanifesto.org/iso/ptbr/ Acesso em: 03 de Março de 2025.

BOSSAVIT, Laurent. The Unbearable Lightness of Programming: a tale of two cultures. Cutter IT **Journal, Massachusetts**, v.15, n.9, 2002. Disponível em: http://cf.agilealliance.org/articles/system/article/file/1052/file.pdf. Acesso em: 18 de Março de 2025.

COCKBURN, Alistair. Crystal Clear: **A Human-Powered Method Small Teams**. New Jersey: Addison Wesley, 2004.

KOSCIANSKI, André; SOARES, Michel dos Santos. Capítulo 10: **Metodologias ágeis**. In: Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 1 ed. São Paulo: Novatec, 2006.

PALMER, Stephen R.; FELSING, John M. **A Practical Guide to Feature-Driven Development**. New Jersey: Prentice Hall PTR, 2002.

POPPENDIECK, Mary; POPPENDIECK, Tom**. Lean Software Development:** An Agile Toolkit. New Jersey: Addison Wesley, 2003.

PRESSMAN, Roger. Capítulo 4: **Desenvolvimento Ágil.** In: Engenharia de Software. 6 ed. São Paulo: McGraw Hill Interamericana, 2006.

SÃO PAULO. Dextra**. Dextra alcança 100 mil horas de projetos com metodologias ágeis com resultados impressionantes**, 2012. Disponível em: http://www.dextra.com.br/noticias/Dextra-alcanca-100-mil-horas-de-projetos-com-metodologias-ageis-com-resultados-impressionantes.htm. Acesso em: 03 de Março de 2025.

SÃO PAULO. USP. **Cooperativa de Desenvolvimento Ágil de Software** – AgilCoop. Instituto de Matemática e Estatística (IME), 2012. Disponível em: http://ccsl.ime.usp.br/agilcoop/casos_de_sucesso. Acesso em: 10 de Março de 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide** – The Definitive Guide to Scrum: The rules of the game. Estados Unidos, 2011. Disponível em: http://www.scrum.org/Portals/0/Documents/Scr

um%20Guides/Scrum_Guide.pdf. Acesso em 20 de Abril de 2025.

WASHINGTON DC. **FBI**. FBI Announces Deployment of Sentinel, 2012. Disponível em: http://www.fbi.gov/news/pressrel/press-releases/fbi-announces-deployment-of-sentinel. Acesso em: 15 de Março de 2025.

WELLS, Don. **The Rules of Extreme Programmin.** Estados Unidos, 1999. Disponível em: http://www.extremeprogramming.org/rules.html . Acesso em 11 de Março de 2025.