Scrumban como metolodologia ágil de desenvolvimento de software para desenvolvedores individuais: Um estudo de caso

Paulo Nunes de Azevedo
Universidade de São Paulo
pnunesdeazevedo@usp.br

Resumo

O estudo dos métodos de desenvilvimento ágil para engenharia de software são métodos criados para maximizar a entrega da solução para um determinado problema, no mínimo tempo possível. No entanto, é notório que para os casos em que o desenvolvimento ocorre por um único desenvolvedor, eles demandam um certo grau de adaptação, e a pesquisa a seguir tem como foco estudar a aplicação de alguns deles utilizando casos reais de desenvolviemento de software de maneira individual.

O Scrumban é uma metodologia que equilibra entregas com organização de fluxo de trabalho, com objetivo de ter bons resultados de forma mais sustentável.

Introdução

Na engenharia de software é muito comum de existirem uma série de problemas, desde falhas de arquitetura, código legado sem documentação devida e que dificulta demais projetos em uma empresa, dívida técnica em projetos passados, dificuldades com entregas de software funcional, dentre vários outros.

Como forma de lidar com tantos problemas, e continuar a desenvolver softwares que cumpram com o seu objetivo e sejam fáceis de se fazer manutenção, surgiu em 2001, um grupo de dezessete desenvolvedores que formaram a chamada *The Agile Alliance*, que compuseram o *The Agile Manifesto*, no qual foi estabelecido alguns dos valores fundamentais, que, desde então, orientam a filosofia ágil [1].

Diante de tantos cenários que variam desde as tecnologias utilizadas, os tipos de clientes da empresa e até mesmo a cultura de onde os desenvolvedores estão inseridos como membros da sociedade, foram desenvolvidos diversos tipos de metodologias ágeis para melhorar o gerenciamento e a produtividade das equipes, que podem ser seguidas tanto de maneira purista, como de forma a aproveitar o melhor de cada uma para determinado cenário. Elas costumam fazer separações de tarefas por membros da equipe e separações de momentos de foco em determinadas atividades, afim de diminuir problemas relacionados à produtividade, como por exemplo, deixar um desesnvolvedor responsável ao mesmo tempo com muitas tarefas ao invés de o permitir focar em somente uma por vez, como reduzir problemas mais técnicos, como o incentivo a documentação e a refatoração, afim de diminuir a dívida técnica e facilitar futuras manutenções, como sugerem metodologias como o XP (Extreme Programming).

Esses problemas, que já são um verdadeiro desafio em projetos com diversas equipes de desenvolvedores com diferentes níveis de expertises e com uma variedade de especialidades, aumentam ainda mais complexidade quando falamos de projetos feitos por um único desenvolvedor. Para esse caso, vemos a necessidade de adaptar as metodologias conhecidas e aclamadas pelo mercado para essas situações, e até mesmo de se pensar em novas metodologias para esse cenário.

Tal situação se encontra como algo comum, quando pensamos em casos como, por exemplo:

Surgimento de startups

No estágio inicial de uma startup temos poucos sócios e funcionários, e pode ser necessário designar um único desenvolvedor para alguns projetos, e será necessário que ele tenha em mente conceitos como a entrega do Produto Mínimo Viável (MVP) e de saber conversar com o cliente final, ou o outros membros da empresa que não sejam técnicos, mas represantariam a imagem de

um cliente, nas abordagens das metodologias em geral.

Algumas startups que podem ser citadas como um exemplo de que, projetos de software com poucos desenvolvedores podem ser muito expressivas para a economia são: Ifood, TRACTIAN, Nubank, PagSeguro, 99 e Loggi.

Projetos Open Source

Muitas das vezes, antes de um projeto Open Source ganhar relevância e inúmeros colaboradores ao redor do globo, o idealizador precisa desenvolver ele sozinho por um período considerável de tempo, ou com poucos colaboradores, com cada um fazendo um projeto menor dentro do inicial, e aplicação de metodologias solo para essas situações podem maximizar o pouco tempo que muitos deles tem para se dedicar a esse tipo de trabalho.

Indústria de Jogos Indie

O surgimento de pequenos estúdios de jogos tem ganhado muita popularidade nos últimos anos, e ultimamente, tem conseguido competir igualmente com grandes indústrias dos jogos eletrônicos como a *Electronic Arts*, *Ubisoft* e a *Activision*, sendo responsáveis por algumas das premiações na *The Game Awards 2024* em títulos como *Balatro* na categoria de melhor jogo mobile.

Projetos Freelancer

Muitas das vezes, o mais adequado para uma empresa é a contratação de programadores freelancer para o desnvolvimento de alguns softwares específicos, e nesse cenário, é muito conveniente que esse desenvolvedor tenha em mente metodologias que otimizem o seu trabalho e as entragas para seus clientes, uma vez que esse tipo de profissional demanda, muitas vezes da indicação de clientes satisfeitos e um maior número de trabalhos realizados com sucesso em uma margem de tempo menor o quanto for possível.

Casos Históricos de desenvolvimento solo

Na história podemos citar alguns softwares de suma importância para o mundo como conhecemos hoje, como o Git, o principal software de versionamento de projetos, desenvolvido inicialmente por Linus Torvalds [2], [3], e também a primeira versão do JavaScript, o Mocha, desenvolvido por Brendan Eich [4], e que hoje em dia é uma das linguagens de programação mais utilizadas no mundo e a que está presente em praticamente quase todas as aplicações web no planeta.

Objetivo

A pesquisa tem como objetivo entender um pouco melhor sobre como as metologias ágeis podem ser adaptadas para o desenvolvedor individual de forma a melhorar o seu desmpenho e as suas entregas, tendo cuidado, ao mesmo tempo, com sobrecarga cognitiva e com burnout.

Função das Metodologias ágeis para o desenvolvedor solo

Enquanto na maior parte dos cenários, temos as metodologias ágeis para dividir o trabalho em equipe, para o desenvolvedor individual, elas atual o auxiliando na autogestão e a evitar uma sobrecarga desnecessária, sem abrir mão de entregas significativas.

Em muitos casos, o desenvolvedor que atua sozinho precisa se dividir em diversos papéis, como o de entender o problema que o cliente quer que seja solucionado, e de gerenciar o fluxo de trabalho, além de manufaturar o código propriamente dito. Cada uma dessas etapas costumam ser desafiadoras indivivualmente, e tal fato, aumentam as chances de sobrecarga e problemas psicológicas atribuidos a isso, como burnout para o desenvolvedor solo, caso não haja uma estrutura eficiente para organizar todas essas tarefas.

Sobre a importância da saúde mental do desnvolvedo solo

Projetos de software, costumam exigir muito da saúde física e mental dos seus desenvolvedores, em muitos dos casos, especialmente quando a equipe é composta por um único indivíduo, que além da questão técnica precisa estar atento à questão mercadológica do produto que está desenvolvendo e da organização de todo o projeto, o que é algo perigoso para a questão mental.

Hoje em dia tem se tornado mais comum o alerta aos cuidados com a saúde mental no trabalho, e colaboradores com maior acúmulo de tarefas costumam ser os maiores alvos de problemas como burnout [5].

Com isso em mente, esse estudo pode contribuir com um ambiente de trabalho mais humanizado, o que para o lado dos desenvolvedores solo melhoram a sua qualidade e expectativa de vida, e para o lado de empresas que tenham esses desenvolvedores como funcionários, evitar de perder esses trabalhadores pelo tempo de sua recuperação em casos mais sérios, ou até mesmo processos judiciais para casos com suspeita de abusos.

Algumas metodologias de desenvolvimento ágil e suas adaptações para o cenário individual

Scrum

O Scrum é uma das metodologias ágeis para desenvolvimento de software mais conhecidas de utilizadas do mundo e ela se resumo em dividir uma equipe em três partes, o Product Owner (PO), Scrum Master e a equipe de desenvolvimento, nos quais temos o primeiro faz o intermediário com o cliente para saber o que ele precisa e o que seria o produto mínimo viável para ele, enquanto que o scrum master orienta como pode ser organizado as sprints para o time de desenvolvimento, de fato, manufaturar o software, em que as sprints seriam um período de cerca de 3 a 5 semanas em média, em que ao final dela é entregue alguma adição funcional do software em questão.

Scrum de Um. Assim é chamada a sua versão para o cenário individual, em que todos as partes do time de desesnvolvimento são o mesmo desenvolvedor que atua em cada momento como cada um desses membros da equipe, hora atuando da forma que se esperaria de um PO, conversando com o cliente para saber o que ele espera, hora como scrum master, traçando objetivos e metas para a equipe de desenvolvedores e fazendo revisões do andamento do projeto, e hora como o time de desenvolvimento fazendo os códigos em si.

Kanban

O Kanban é uma metodologia focada em fluxo de trabalho visual, na limitação do trabalho de progresso (WIP) e na melhoria contínua do sistema. Essa metodologia se encontra muito comumente sendo usada junto com alguma outra, como o scrum.

Seu foco é de organizar o fluxo de trabalho de forma visual, seja com quadro de anotações em uma empresa, ou com softwares de próprios para isso, como o Trello. A ideia central é categorizar cada tarefa como “A fazer”, “Em andamento” ou “Feita”, e minimizar a quantidade de tarefas em andamento ao mesmo tempo, para que haja um foco maior em cada atividade individualmente.

Personal Kanban. Como uma metodologia mais focada na gestão de tarefas de uma equipe, ele serve de maneira praticamente direta para o desenvolvedor solo, usando de suas ideias para a autogestão do projeto.

Scrumban [6]

O Scrumban é a junção do Scrum e do Kanban como uma metodologia própria. Ele trabalha com organização visual do fluxo de trabalho e com o WIP, e não é tão rigoroso quanto as sprints do scrum clássico.

É uma das metodologias mais usadas também, na indústria da engenharia de software por unir vantagens de outras duas metodologias clássicas, e podendo oscilar entre, por exemplo, usar um pouco mais o scrum puro, sendo mais rógido quanto às entregas e às revisões, ou menos protocolar e focar em finalizar uma tarefa em andamento por vez mais do que realizar as dailys solo por questões protocolares por si só.

Extreme Programming (XP)

O XP tem o foco em produzir softwares com maior qualidade técnica, maior capacidade de resposta e de mudança nos requisitos do cliente, além de ter maior facilidade de realizar futuras manutenções quando necessárias. Essa metodologia, embora não seja a escolhida para esse

estudo de caso, pode ter traços que possamos utilizar futuramnente.

Ela valoriza alguns pilares, como: Comunicação, simplicidade, feedback, coragem e respeito.

O XP também conta com algumas práticas centrais como o TDD (Test-Driven Development), que consiste em escrever um teste automatizado antes de escrever o código em si, e ir elaborando o código até ter a versão mínima que passe nesse teste, e após isso, é elaborado um novo teste com mais algumas funções e o código passa por refatoramentos até que passe no novo teste. Esse ciclo é chamado de "Red, Green, Refactor" [7]:

- Red: Pensar sobre o que o quer ser desenvolvido, e daí elaborar testes do que deve ser feito.
- Green: Pensar em como passar nos testes feitos. Elaborar o código mínimo que passe nos testes.
- Refactor: Pensar em como melhorar a implementação existente, muitas vezes refatorando o código para permitir uma nova implementação.

Como prós do TDD temos uma alta cobertura de testes e confiança na alterações do código, tendo os testes como espécia de documentação sobre cada alteração feita durante o projeto e como desvantágem temos a sua curva de aprendizagem, que pode tornar a sua proeficiência difícil de ser dominada.

XP para o desenvolvedor solo. Ao usar o XP para o desenvolvimento solo podemos fazer as seguintes adaptações de seus pilares:

- Comunicação: Ter comunicação clara com o cliente, com usuários beta e com si próprio, fazendo uso de boas práticas de programação e documentação clara sempre que possível.
- Simplicidade: Procurar fazer o que for mais simples e que funcione, evitando implementações complexas sem necessidade, mais conhecido como o conteito *"You Ain't Gonna Need It"* (YAGNI).
- Feedback: Receber feedbacks de testes automatizados, de entregas de protótipos para o cliente ou usuários beta e auto-reflexão sobre o que foi feito até o momento.
- Coragem: Ter a coragem de muitas das vezes refatorar partes do seu código, afim de tornar possível a impelementação de alguma feature ou de tornar mais fácil de realizar futuras manutenções.
- Respeito: Respeitar o próprio trabalho, o tempo e o cliente no sentido de facilitar manutenções futuras no código aplicando boas práticas de programação.

Metodologia

A metodologia adotada para a pesquisa será um estudo de caso. Será produzido um software utilizando o scrumban e será regitrado o seu progresso, por testes de software, documentação do projeto e pelos feedbacks tanto do desenvolvedor quanto do cliente.

Coleta de Informações

Durante o desenvolvimento, serão coletados dados que indiquem sobre o andamento do projeto e como a metodologia escolhida está influenciando o seu desenvolvimento, assim como foi a relação entre o desenvolvedor solo e o cliente, tanto em relação às impressões, quanto às entregas durante o processo.

A coleta será feita através do resultado dos testes que serão realizados durante o desenvolvimento do software, que medirão a sua evoluação durante o processo de desenvolvimento, e pelos feedbacks do cliente e do desenvolvedor, que irão informar como foi o processo, tanto no quesito humano, em relação aos fatores como sobrecarga de tarefas, quanto no sentido de satisfação do cliente com as entregas.

Cronograma

Definição dos requisitos do software a ser desenvolvido - 1 mês

Será feito um estudo sobre os requisitos do software a ser desenvolvido, onde será enfatizado a atuação do que seria o PO do scrum original.

Será elaborado um documento em que contêm o problema do cliente a ser resolvido pelo desenvolvedor para que, posteriormente, sob a atuação do scrum master, seja elaborado um plano de sprint inicial para elaboração do produto mínimo viável, mas sem o mesmo rigor de sprints e reuniões diárias do scrum original, sendo substituido pelo foco em separar o projeto em pequenas tarefas e minimizar ao máximo as que estarão em andamento ao mesmo tempo.

Aplicação o método no desenvolvimento do software - 8 meses

O desenvolvimento do software propriamente dito. Aqui a metodologia será usada ao eu máximo, e é possível que seja necessário usar algum traço de alguma outra além do scrumban, e será também documentado até que ponto o scrumban por si só foi mais relavante do que quando somado a mais uma metodologia.

Avaliação dos resultados - 1 mês

Será feito um levantamento sobre os resultados obtidos com o desenvolvimento do software e quais as influências que o scrumban teve no seu desenvolvimento, tanto em relação à qualidade de vida que o desenvolvedor experimentou nesse período, quanto a satisfação do cliente ao receber as entregas.

Preparação dos relatórios - 2 meses

Será elaborado um relatório final com o as conclusões sobre o estudo de caso, tanto pela funcionalidade do software entregue, quanto pelos feedbacks do desenvolvedor e do cliente, como pessoas envolvidas no processo.

Referências

[1] K. Beck *et al.*, «Manifesto for Agile Software Development». Acedido: 11 de junho de 2025. [Online]. Disponível em: https://agilemanifesto.org/

[2] S. Vaughan-Nichols, «História do Git». Acedido: 2 de junho de 2025. [Online]. Disponível em: https://www.zdnet.com/article/linus-torvalds-built-git-in-10-days-and-never-imagined-it-would-last-20-years/

[3] L. Torvalds, «Código-fonte do Git». Acedido: 2 de junho de 2025. [Online]. Disponível em: https://github.com/git/git/commits/f35ca9ed3ef02541a938f3bd43cad83af93eb94f/?since=2005-04-07&until=2005-04-07

[4] Cybercultural, «1995: The Birth of JavaScript». Acedido: 2 de junho de 2025. [Online]. Disponível em: https://cybercultural.com/p/1995-the-birth-of-javascript/

[5] A. K. Pikos, «The causal effect of multitasking on work-related mental health: The more you do, the worse you feel». Acedido: 15 de junho de 2025. [Online]. Disponível em: https://www.econstor.eu/bitstream/10419/172863/1/dp-609.pdf

[6] R. P. P. - PR, «Scrumban: a metodologia que une o melhor do Scrum e do Kanban». Acedido: 15 de junho de 2025. [Online]. Disponível em: https://posdigital.pucpr.br/blog/scrumban

[7] C. Team, «TDD - Red, green, refactor». Acedido: 15 de junho de 2025. [Online]. Disponível em: https://www.codecademy.com/article/tdd-red-green-refactor