Desenvolvimento de Sistemas

Desenvolvimento ágil: cultura ágil, metodologias, rotinas, *frameworks*

O que é desenvolvimento ágil?

Você aprendeu, logo no início de seus estudos, as metodologias ágeis Scrum e *kanban*, que são frequentemente utilizadas. Esses conjuntos de orientações e ferramentas que organizam um modo de trabalhar (*frameworks*) são formas dinâmicas de desenvolver um projeto, seja um projeto de *software*, seja um projeto pessoal.

Ainda que seja muito útil conhecer vários tipos de processos e de ferramentas disponíveis, também é imprescindível conhecer a estrutura de princípios e valores que fundamenta tais metodologias, pois, para se tornar um desenvolvedor ágil, é necessário compreender a visão geral de agilidade. Pode-se dizer, portanto, que o desenvolvimento ágil se trata de um conjunto de fundamentos e processos que guia a execução das metodologias ágeis.

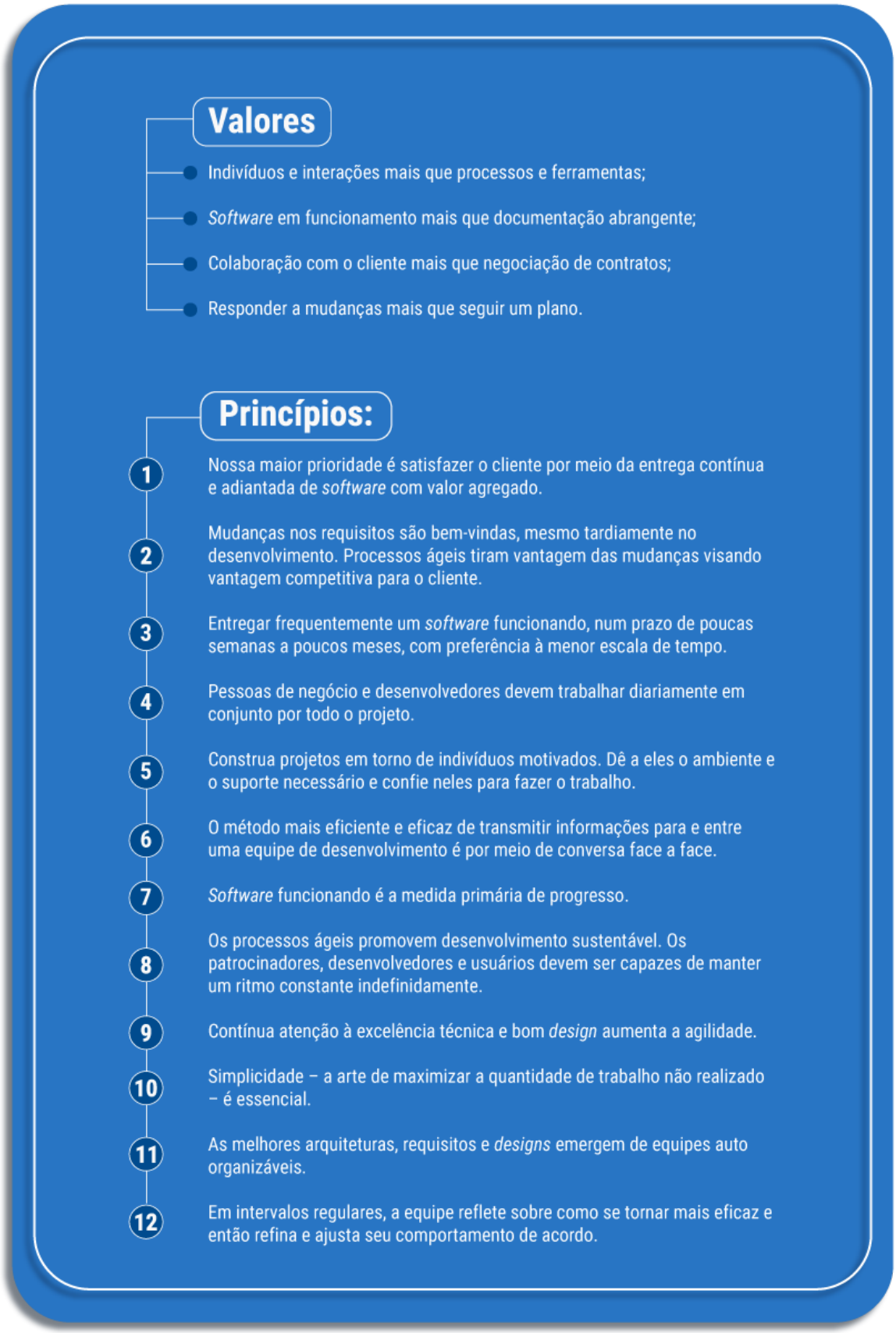
Vale também ressaltar que, embora seja possível aplicar metodologias ágeis a outros projetos que não sejam de desenvolvimento de *software*, o contexto deste conteúdo abrange a perspectiva do desenvolvimento de *software*.

Em 2001, um grupo de desenvolvedores se reuniu para criar uma declaração contendo uma representação de qual seria a melhor forma de desenvolver um *software*. Naquele período, as metodologias utilizadas não

estavam sendo satisfatórias, portanto, filtrar e agrupar aspectos que promovessem eficiência e eficácia nos projetos era uma forma de produzir novos e melhores resultados.

Assim, com base nessa análise, foram definidos quatro valores e 12 princípios, os quais posteriormente fundamentariam as metodologias ágeis. O conjunto desses princípios e desses valores foi chamado de “manifesto ágil”.

Veja os valores e os princípios do manifesto ágil que guiam as práticas ágeis:


Valores
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.
Princípios:
1
Nossa maior prioridade é satisfazer o cliente por meio da entrega contínua e adiantada de software com valor agregado.
2
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3
Entregar frequentemente um software funcionando, num prazo de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4
Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5
Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6
O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face.
7
Software funcionando é a medida primária de progresso.
8
Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9
Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10
Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11
As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis.
12
Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Além de conhecer os fundamentos do desenvolvimento ágil, é muito importante entender o real significado de **agilidade** dentro do contexto de desenvolvimento de *software*.

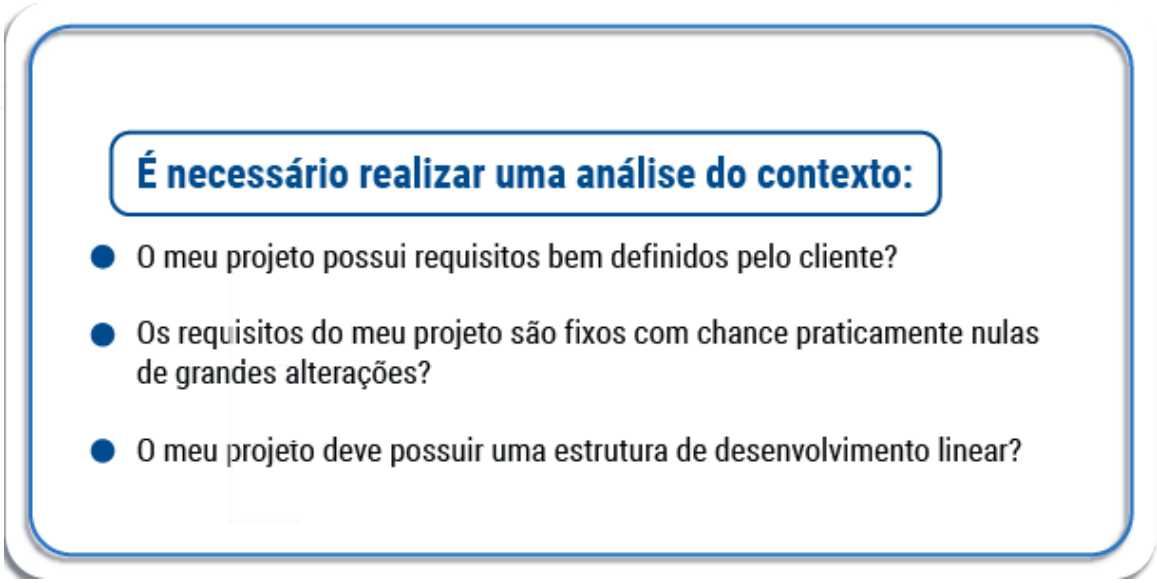
Em um primeiro contato com o termo, é comum relacionar a ideia de desenvolvimento ágil com a rapidez na entrega do *software*, com um desenvolvimento muito mais rápido em relação a outras metodologias. Entretanto, essa visão limitada pode acabar frustrando empresas, desenvolvedores e clientes, que passarão a ter uma expectativa irreal.

Ainda que possa de fato acelerar o processo de desenvolvimento, ao analisar o manifesto ágil, é possível observar que muitos dos princípios e dos valores estão relacionados à entrega contínua do projeto funcional em prazos menores, à aceitação de mudanças de projeto e à adaptação rápida a elas, à realização de análises frequentes de como o desenvolvimento está ocorrendo, à simplificação de processos burocráticos, entre outras.

Todos esses aspectos fazem com que as dificuldades e os problemas que possam surgir sejam solucionados mais rapidamente. É um trabalho em equipe que requer práticas simplificadas dentro da rotina. Portanto, não se trata somente de uma possível rapidez de entrega, mas, sim, da adoção de hábitos ágeis, da facilidade de processos, entre outros.

“Entrega contínua” é a capacidade de o sistema ser constantemente atualizado para o cliente, incluindo novas funcionalidades ou correções de defeitos.

Ainda, embora as metodologias ágeis possam chamar muita atenção como algo dinâmico e moderno, elas não serão obrigatoriamente a melhor solução para o desenvolvimento de um projeto.



Embora não seja regra, se você respondeu “sim” para as perguntas, provavelmente a metodologia em cascata ou outras convencionais se encaixem melhor no desenvolvimento do seu projeto.

Sistemas que envolvam licitações ou serviços específicos para órgãos públicos se beneficiam dessas metodologias, pois os requisitos do sistema provêm de regras previamente definidas e bem estruturadas. Outros sistemas mais simples, como gerenciamento de estoque, gerenciamento de vendas em supermercado e sistema de elaboração de orçamentos, são exemplos de projetos compostos de procedimentos padronizados com baixa tendência de mudanças ou atualizações, nos quais também não há uma real necessidade de aplicar metodologias ágeis.

Lembre-se de que, no método em cascata, cada etapa (de requisitos, de projeto, de implementação, de testes e de implantação) só pode iniciar quando a anterior estiver concluída. Outras metodologias, como a em espiral, que prevê períodos cíclicos de desenvolvimento (como as *sprints* do método Scrum), não são ágeis pelo excesso de burocracia e de documentação envolvidas.

Existem casos em que o tempo de colocação de um produto no mercado é um requisito importante, visto que, muitas vezes, determinado projeto depende do momento de entrada no mercado para prosperar e reagir contra a

competitividade da concorrência.

Se o prazo de entrega passar desse período, talvez o projeto de *software* perca o sentido e o valor para o cliente. Diante dessa perspectiva, para que o produto se destaque no mercado, muitas vezes os requisitos precisam ser alterados para que não haja prejuízos para o cliente.

Em outros casos, os requisitos podem nem estar completamente definidos logo no início do projeto, seguindo o fluxo de necessidades, e é aí que surge o seguinte questionamento: como criar um processo capaz de administrar a imprevisibilidade?

Os métodos ágeis, então, entram em ação e se apresentam como uma vantagem competitiva. É necessário ter agilidade para se adaptar a mudanças e apresentar entregas de valor para o cliente e para o ambiente de negócios.

Sistemas de redes sociais, de lojas, de tele-entrega, de transporte e outros que necessitam frequentemente de novas funcionalidades e atualizações são exemplos de aplicações que, muitas vezes, são influenciadas até mesmo por questões econômicas, sociais e sazonais. Esses são apenas alguns exemplos de sistemas que seriam beneficiados pelo uso de metodologias ágeis, nos quais cada entrega particionada acrescenta valor para o cliente.

Como foi possível observar, dois dos principais aspectos da metodologia ágil são a dinamicidade e a fluidez ao entrar em cenários de mudança. Em contextos normais, mudanças são caras, principalmente se não forem controladas e se não forem bem administradas, existindo processos adequados para lidar com a situação. Uma das características mais atrativas da metodologia ágil é justamente a capacidade de reduzir os custos da mudança no processo de *software*.

Observe a seguir um gráfico comparativo entre a agilidade e o custo das mudanças em relação a processos convencionais:

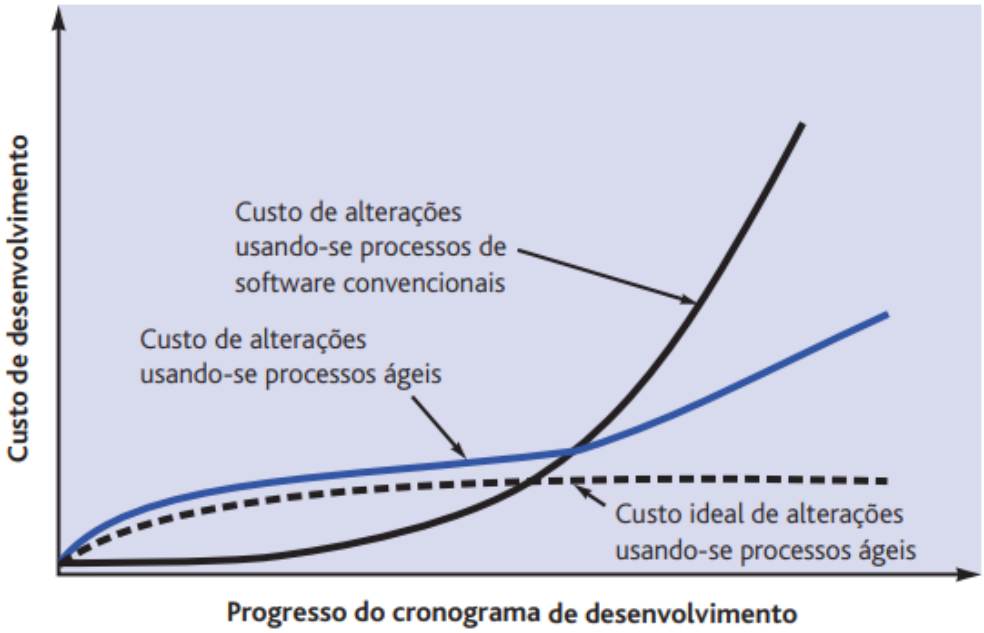


Figura 1 – Custos de alterações em relação ao tempo de desenvolvimento
Fonte: Pressman; Maxim (2016)

No processo de reunir requisitos logo no início do projeto, se houver a necessidade de alterar um detalhamento ou aumentar uma lista de funções, o projeto não será afetado negativamente, pois ainda será possível adaptar os processos.

Quanto mais perto do fim do projeto, mais difícil se torna para as metodologias mais “rígidas” alterarem sua estrutura, fazendo com que o processo de modificação seja mais custoso. Isso porque, por vezes, as mudanças podem surgir quando o sistema já está em etapa de testes e acaba envolvendo alterações em diversas áreas em que o desenvolvimento já foi concluído.

As metodologias ágeis tendem a reduzir esse problema, pois as alterações ao longo do desenvolvimento se dividem em tarefas menores conforme surgem. Além disso, o impacto das modificações reduz o custo geral desse processo.

As entregas incrementais aos clientes se tornam bastante benéficas por essa perspectiva, pois permitem que o desenvolvimento seja avaliado e que possíveis mudanças sejam identificadas o mais breve possível, podendo moldar também a estrutura de como o projeto deve seguir posteriormente, evitando retrabalhos futuros.

Rotinas

É disseminado exageradamente que as práticas ágeis são contra processos, documentações e todo e qualquer processo burocrático, mas não é assim que funciona.

Desenvolvimento ágil não significa que nenhum documento é criado ou que processos não são utilizados; significa, na verdade, que somente os documentos que serão consultados mais adiante no processo de desenvolvimento serão criados, priorizando reais necessidades, da mesma forma que algum processo obrigatoriamente burocrático de uma empresa, se realmente necessário, poderá continuar sendo burocrático.

Outro aspecto muito importante é a percepção de que desenvolver um *software* de forma ágil não se trata somente de seguir uma metodologia ágil.

Não é possível simplesmente definir que serão utilizadas metodologias ágeis sem antes fazer com que práticas ágeis já façam parte da rotina. Quando a rotina é adaptada à visão de agilidade, uma estrutura adequada é desenvolvida para implementar metodologias, mesmo que outras áreas permaneçam mais burocráticas ou requeiram alguns procedimentos.

Veja a seguir algumas práticas que podem ser incorporadas no dia a dia para simplificar processos e proporcionar melhores resultados para a equipe:

Programação em par

Uma das práticas que podem trazer benefícios para a rotina dos programadores é a programação em par. A ideia é que duas pessoas trabalhem no desenvolvimento de um mesmo código, compartilhando monitor, teclado e *mouse*.

Um dos componentes da dupla (o “líder”) ficará responsável pela parte escrita do código, enquanto a outra pessoa (o “navegador”) terá a responsabilidade de observar, analisar e revisar o que está sendo desenvolvido. O papel de cada um da dupla poderá ser modificado com certa frequência caso necessário.

Ainda que pareça um trabalho rigoroso, ele traz diversos benefícios a longo prazo, pois melhora a qualidade do código com base na troca de experiências, conhecimentos e informações entre as pessoas, diminuindo a necessidade de uma futura refatoração do código.

Outro benefício proporcionado pelo uso dessa prática é o foco de desenvolvimento da equipe, pois todos estarão muito envolvidos nos códigos e em frequente observação.

É necessário, antes de aplicar a programação em par, analisar a quantidade de colaboradores disponível para realizar esse processo, bem como os prazos estipulados para o projeto.

Embora possa trazer benefícios a longo prazo (como redução de tempo em refatoração de código e maior compreensão do código inteiro), para situações que requeiram mais rapidez, talvez a

programação em par não seja a prática mais adequada.

Em substituição, pode ser usada a revisão de código (*code review*), uma evolução da programação em pares que define que todo código desenvolvido deve ser, sim, revisado, mas se possível em outro momento, de maneira assíncrona, por outro desenvolvedor.

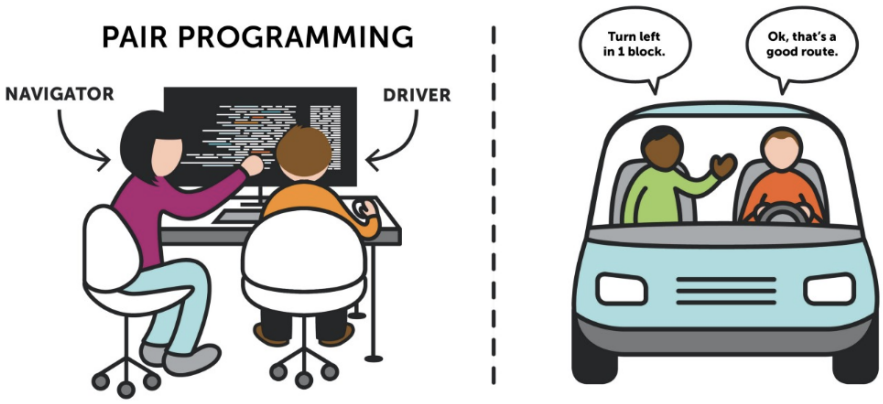


Figura 2 – Programação em par
Fonte: Mass (2020)

Veja mais alguns benefícios da programação em par:

- Muitos erros são identificados ao longo do desenvolvimento, e não nas etapas finais do projeto, como em estágios de teste.
- A identificação de erros no fim do projeto é estatisticamente menor.
- O código é menor e tem um *design* melhor devido às contribuições do observador.
- A equipe resolve problemas mais rapidamente.
- A equipe compreende melhor os problemas encontrados e as práticas de construção de *software*.
- O projeto de *software* acaba com várias pessoas conhecendo melhor cada peça da solução que foi desenvolvida, facilitando possíveis ajustes ou incrementos no código.
- A equipe desenvolve as habilidades de trabalho em equipe, comunicando-se com mais fluidez e dinamicidade.

Apesar de benéfica, essa prática realmente pode se tornar cansativa se realizada com frequência. Dessa forma, é importante realizar pausas ao longo do dia.

É importante também revezar duplas, para que toda a equipe fique mais engajada no desenvolvimento e melhore a comunicação.

Code review

Uma prática importante que deve ser implementada em equipes é a colaboração no código. Técnicas como a revisão de código (*code review*) auxiliam a equipe no desenvolvimento do senso de responsabilidade coletiva pelo que está sendo criado e, ao mesmo tempo, aumentam a base de conhecimento de todos os envolvidos.

Tal como o nome sugere, o objetivo dessa prática é realizar uma revisão a cada versionamento do projeto, não devendo ser um processo muito demorado, visando a encontrar falhas, *bugs* e possíveis pontos de melhoria no desenvolvimento.

O programador, assim que conclui uma funcionalidade e está pronto para incluí-la no código principal do projeto (e no repositório), submete esse arquivo a outro desenvolvedor, que analisa o código e aponta melhorias, caso necessário. O arquivo então retorna ao programador, que realiza os ajustes e o submete novamente à revisão até que não haja mais ajustes necessários.

O ideal é que, no ambiente da equipe, haja tempo para analisar o código que está sendo produzido e dar um retorno sobre ele.

Há ainda outros pontos positivos que podem ser observados na code *review*:

- Aumento da qualidade do código
- Maior confiança no momento de entrega do projeto
- Ganho de conhecimento tanto por parte do desenvolvedor quanto por parte de quem está realizando a revisão

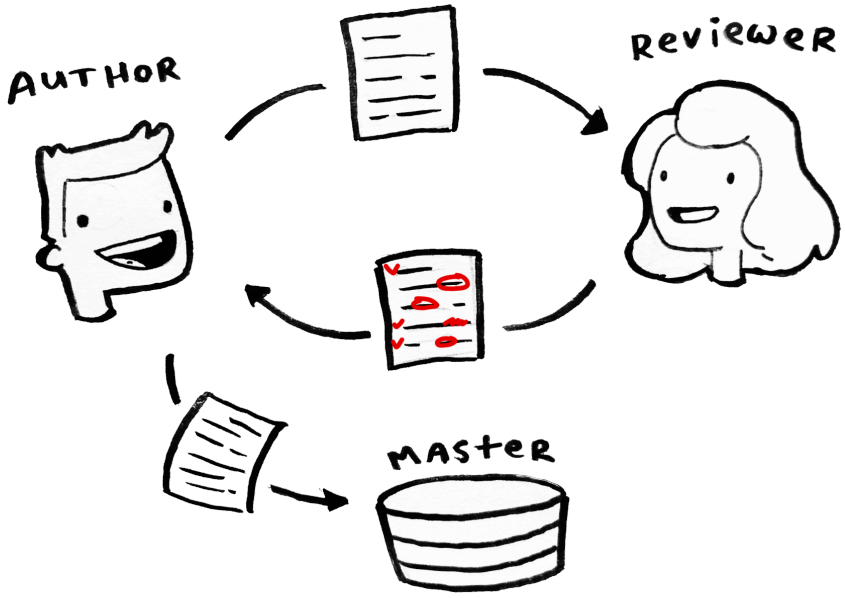


Figura 3 – Processo de *code review*

Fonte: Matic (2020)

TDD

Outra prática interessante a ser adotada é o *test-driven development* (desenvolvimento guiado por testes), ou TDD. Trata-se de uma prática que depende de testes automatizados, classes e métodos específicos que invocam métodos de outras classes do sistema, verificando se o resultado gerado por eles é o esperado.

Veja um exemplo de teste automatizado:

```
public class ClasseDoSistema{
    public int calculaSoma(int a, int b){
        int soma = a + b;
        return soma;
    }
}
```

```
public class TestesDoSistema{
  @Test
  public void testaSoma(){
      int resultado = calculaSoma(2,2);
      //verifica se o resultado é 4, o esperado para 2+2
      assertEquals(4,resultado);
  }
}
```

Essas classes de teste podem ser executadas automaticamente, permitindo detectar erros e alguma alteração realizada no código.

No TDD, primeiro são desenvolvidos os métodos de teste e, depois, os de funcionalidade. No exemplo anterior, primeiramente foi desenvolvido **testaSoma()** para depois ser implementado **calculaSoma()**. A primeira execução do teste acusará falha, pois não há implementação no método testado. Diz-se, nesse momento, que os testes estão “vermelhos”, ou seja, falhos.

Com as definições dos resultados esperados, completa-se o código da funcionalidade e se executa novamente o teste até que a execução fique “verde”, ou seja, seja realizada com sucesso (o resultado esperado é obtido). O objetivo a partir daí será desenvolver um código completo que resulte em sucesso nos testes anteriormente idealizados. Após, poderá ser realizada a refatoração do código para aprimorá-lo.

Dessa maneira, é possível observar todas as possíveis falhas do código, podendo resolver todos os problemas, e guiar o modo como o *software* é desenvolvido. Além disso, na hora de realizar testes de *software*, provavelmente o código necessitará de menos ajustes, pois a análise de falhas e testes já ocorreu durante o desenvolvimento.

Porém, cabe um ponto de atenção: **essa prática requer um período de adaptação para que seja aplicada.**

O TDD traz os seguintes benefícios:

- Mais clareza do código que já foi produzido
- Mais qualidade do produto, pois os testes automatizados darão segurança de que uma nova funcionalidade incluída no sistema não prejudicará outras funcionalidades já existentes
- Refinamento do código de forma mais rápida e objetiva às necessidades

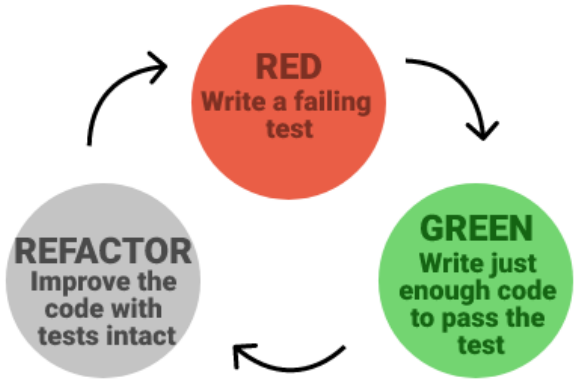


Figura 4 – Processo de TDD

Fonte: Stemmler (2021)

Design incremental

O *design* incremental da base de código consiste em uma técnica que visa a uma estrutura de código simples desde a sua concepção. O ponto-chave é que, embora simples, a estrutura pode ser melhorada de forma contínua, progressivamente.

A parte do *design*, no caso de *softwares* mais complexos, é muito importante para garantir robustez e qualidade futura, afinal, por eles terem um período mais longo de desenvolvimento e um conjunto maior de requisitos, é possível que mudanças sejam solicitadas com mais frequência conforme as necessidades do cliente.

O *design* incremental é uma forma de economizar tempo, direcionando o desenvolvimento diretamente para as demandas necessárias com mais frequência.

Alguns dos benefícios em utilizar tal técnica são:

- Um *design* que é frequentemente aprimorado desenvolve uma estrutura melhor para aqueles que trabalham no desenvolvimento dele.
- Diversas situações tornam difícil a tentativa de prever detalhes do *design*. Logo, não há como definir rapidamente um *design* final para a estrutura de código. O uso do *design* incremental faz com que não seja gasto mais tempo do que o necessário em algo que possivelmente será alterado.
- Em códigos antigos, reescrevê-los utilizando o *design* incremental aumenta a vida útil do *software*.

Considerando que as metodologias ágeis buscam um processo de entrega contínua, novas decisões na estrutura do *software* podem ser definidas após o projeto ser disponibilizado em sua primeira versão para os clientes.

Padronização do código

Mais uma prática bastante interessante é a padronização do código desenvolvido. São consideradas boas práticas de programação: realizar a documentação de código; definir estilos e padrões de *design*; indentar etc. Quando você desenvolve um código, por exemplo, precisa considerar que ele não passará somente pelas suas mãos. Portanto, é imprescindível que o código seja de fácil compreensão e mantenha sempre padrões de semântica.

Há ainda outros benefícios em utilizar a padronização de códigos:

- A leitura do código se torna mais fácil para todos os desenvolvedores e para si mesmo.
- A aplicação de melhorias ao código e a manutenção dele se tornam tarefas menos “pesadas”, fluindo com mais facilidade.
- A adoção da padronização do código torna os algoritmos mais profissionais.
- Reduz-se a necessidade de manutenção do código.

Time box

A *time box* (caixa de tempo) tem um conceito simples: é uma forma de gerir o tempo mediante a definição de uma meta e de um intervalo de tempo para a realização de uma tarefa. Isso evita a procrastinação, pois você já definiu o que vai fazer e quanto tempo terá disponível para tanto.

É importante, inclusive, que o tempo predefinido seja adequado à tarefa, afinal, caso venha a ser estabelecido mais tempo do que o necessário, o objetivo real dessa prática acabará não sendo atingido.

São alguns dos benefícios da utilização da *time box*:

- Maior foco ao desenvolver uma tarefa
- Otimização do tempo, evitando passar mais que o necessário em um mesmo código
- Aplicação em diversos contextos além da programação
- Conclusão mais frequente das metas, o que gera motivação

Comunicação

A comunicação com os clientes e com a própria equipe, dentro do mundo ágil, objetiva praticidade e objetividade.

É importante ser acessível e facilitar a troca de informações sempre que possível. Reuniões auxiliam bastante no processo de comunicação, mas devem seguir os padrões de agilidade, tendo um tempo predefinido (*time box*) e objetividade.

Por causa do cenário atual, o trabalho é, em diversas situações, realizado a distância. Então, é interessante investir também em ferramentas que auxiliem no processo de comunicação, tais como Slack, Teams, Discord, entre outras.

Mesmo que não existam muitas técnicas-padrão a serem seguidas, é importante sempre ter bom senso. As pessoas não são iguais; elas têm capacidades e personalidades diferentes. É necessário respeitar os colegas em todos os processos de comunicação para um bom ambiente de equipe.

Buscando a melhoria contínua, você pode observar o manifesto ágil, em especial o item “indivíduos e interações mais que processos e ferramentas”. Ao analisá-lo, é possível entender que, embora práticas ágeis sejam muito benéficas, é imprescindível melhorar a interação entre as pessoas para alcançar o sucesso no time e para ter uma *performance* melhor.

Para finalizar, lembre-se de que as práticas mencionadas, ainda que sejam utilizadas dentro de algumas metodologias ágeis, podem ser aplicadas individualmente.

Frameworks e metodologias

Como visto ao longo deste conteúdo, o desenvolvimento ágil consiste em um conjunto de princípios e valores utilizado como forma de nortear todo o processo de organização e de desenvolvimento de tarefas.

Quando se fala em metodologias e *frameworks* ágeis, está-se falando em formas de **aplicar** os princípios ágeis a um projeto, ou seja, formas de sair somente da teoria e vincular a agilidade a um projeto.

Pode-se dizer então que *frameworks* são um conjunto de ferramentas úteis que, quando aplicadas à rotina, trazem determinados resultados que inclusive podem ser aplicados a contextos diferentes.

Já uma metodologia é um conjunto de métodos (incluindo ferramentas) que fazem parte de um processo bem definido, orientando os passos que o projeto deve tomar. Ainda que os conceitos tenham suas diferenças, muitas vezes o termo “*framework*” é difundido como metodologia, ou então o termo “metodologia” é difundido como *framework*. Sendo assim, talvez você encontre em livros e na Internet os processos e as ferramentas que serão apresentados sendo definidos das duas formas.

Algumas das metodologias mais utilizadas no mercado você já deve conhecer. Que tal recapitular?

Scrum

O que é

Scrum é considerado um *framework* que auxilia pessoas, times e organizações a gerarem valor por meio de soluções adaptativas para problemas complexos.

Processo

Em resumo, o método Scrum requer que um *scrum master* promova um ambiente em que ocorram os seguintes processos:

- Um *product owner* ordena o trabalho para problemas complexos em um *product backlog*.
- O *scrum team* transforma uma seleção do trabalho em um incremento de valor durante uma *sprint*.
- O *scrum team* e os seus stakeholders inspecionam os resultados e se preparam para a próxima sprint por meio de reuniões frequentes.
- O processo é repetido até a finalização do projeto.

Para relembrar conceitos gerais do Scrum de forma mais detalhada, leia a obra *The Scrum Guide*, um guia oficial desenvolvido pelos criadores do Scrum, gratuito e disponível em português.

Kanban

O que é

O *kanban* é uma estratégia para otimizar o fluxo de valor por meio de um processo que utiliza facilitação visual e limitação de *work in progress* – WIP (trabalho em progresso) de um sistema puxado. Essa facilitação visual ocorre por quadros com colunas e cartões.

Processo

O processo deve ocorrer da seguinte forma:

- Visualização do *workflow*: clareza do fluxo de trabalho, por meio de quadros, para que todos possam enxergá-lo.
- Limitação do *work in progress*: definição – por estado(s) do trabalho, por pessoa, por faixa, por tipo de trabalho, para todo o sistema *kanban* etc. – do número máximo de itens de trabalho permitidos de cada vez.
- Gestão ativa dos itens de trabalho em andamento: monitoramento frequente da execução de processos.
- Inspeção e adaptação da definição de *workflow* do time: verificação da eficácia do fluxo de trabalho e realização de ajustes conforme necessário.

Para relembrar conceitos gerais do *kanban* de forma mais detalhada, leia a obra *The Official Guide to The Kanban Method*, um guia oficial desenvolvido pelos criadores do *kanban*, gratuito e disponível em português.

XP

O que é

A *extreme programming* (programação extrema), ou XP, envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento, projeto, codificação e testes. O propósito é que, se uma prática está sendo benéfica, ela deve ser executada várias e várias vezes, trazendo mais qualidade ao código.

Processo

O processo deve ocorrer da seguinte forma:

- Planejamento: história de usuário, valores, critérios de teste de aceitação, plano de iteração.
- Projeto: projeto simples, cartões CRC (classe, responsabilidade, colaboração), soluções pontuais, protótipos.
- Codificação: programação em pares, testes de unidades, integração contínua.
- Testes: testes de unidades, integração contínua.

A XP não conta com um guia especialmente desenvolvido para ela, mas, para conhecê-la ainda mais, acesse o *site* oficial da metodologia para acompanhar as regras de cada uma das etapas.

Apesar dos aspectos únicos de cada metodologia, é possível combiná-las conforme necessário. Com o avanço do tempo, as organizações precisam estar adaptadas a novas realidades, pois, consequentemente, manter o mesmo conjunto de práticas nem sempre é a melhor alternativa para o negócio.

Nessa perspectiva, é sempre importante buscar a metodologia (ou as metodologias) que se aplica melhor aos processos. Também é válido considerar como a equipe se adaptará às mudanças de papéis e à estrutura de trabalho. O processo poderá ser desafiador, mas com certeza trará benefícios.

Cultura ágil e processo de transição

A cultura dentro de uma empresa é caracterizada pelo conjunto de comportamentos, atitudes, missão, visão e valores que guiam uma organização. Assim, a cultura ágil consiste em inserir os conceitos de agilidade como parte do que guia a organização, intrínsecos aos colaboradores.

Para empresas muito focadas em procedimentos burocráticos e estruturas hierárquicas, a inovação parece algo distante. Muitas empresas – e não somente equipes de desenvolvimento – têm buscado, com mais frequência, formas de utilizar metodologias ágeis para melhorar seus processos de negócio. Quando as organizações adotam essa perspectiva, criam aberturas para transformar e atualizar o ambiente de trabalho, trazendo melhores resultados para os clientes externo e interno.

Mesmo com tantos benefícios, a falta de preparo e o não entendimento da cultura ágil na empresa podem fazer com que todo esse processo de transição se torne, no lugar de uma solução, um problema ainda maior. Alguns dos problemas que podem ser vistos quando a cultura ágil não está sendo realmente entendida na empresa são os seguintes:

- Crença dos gestores de que não existe mais liderança
- Insatisfação quando orientações mais tradicionais não ágeis são rejeitadas
- Dificuldade de se “desprender” de procedimentos burocráticos

A cultura ágil não busca necessariamente aplicar agilidade em todos os setores da empresa, quebrar todas as regras, mas sim, como visto anteriormente, trazer um equilíbrio entre ter agilidade nos locais certos e seguir normas que são realmente importantes. Adequar-se a esse domínio requer empenho de todos que fazem parte da empresa.

Veja a seguir como adotar essa visão no seu cotidiano e como outros setores influenciarão no processo:

Auto-organização

“Uma equipe ágil é auto-organizada e tem autonomia para planejar e tomar decisões técnicas” (PRESSMAN, 2016). Quando se fala em metodologias ágeis, pode-se ter a perspectiva de que a auto-organização é não linear, ou seja, de que não há um cronograma fixo do que deve ser feito.

Logo, por haver alterações frequentes e mais dinamicidade no projeto, é necessária uma organização maior para que as tarefas não se tornem uma enorme bagunça. Veja algumas dicas:

- Seja proativo para executar tarefas pendentes. Não espere ordens.
- A comunicação com a equipe deve ocorrer frequentemente, inteirando-se mais das necessidades dela.
- Questione várias vezes sobre suas tarefas até que compreenda completamente o que deve ser desenvolvido. Não fique com dúvidas.
- Registre todas as informações que achar necessárias para a tarefa, evitando perdê-las.
- Atualize frequentemente os *status* do seu progresso na tarefa para uma melhor organização da equipe.

A auto-organização é um processo de cada um, que não conta com uma autoridade ou com elementos externos para ser colocada em prática. Para que a cultura ágil funcione, é necessário que toda a equipe siga princípios básicos de auto-organização por conta própria.

Visão de time

“A filosofia ágil enfatiza a competência individual (de cada membro da equipe) combinada com a colaboração em grupo como fatores críticos de sucesso para a equipe” (PRESSMAN, 2016). Em um contexto de equipe, existe uma descentralização de autoridade, permitindo que haja uma maior participação dos integrantes no andamento do projeto. Porém, não se engane: ainda haverá um gestor para orientar o time.

A diferença, nesse caso, é que a equipe poderá opinar mais sobre como os procedimentos serão realizados e terá mais autonomia para tomar decisões técnicas e para organizar e direcionar as tarefas conforme a especialidade da equipe. A estrutura-base do projeto será responsabilidade da liderança, enquanto a dinamicidade referente à organização e à delegação de tarefas caberá à equipe.

Confira algumas ações que auxiliam na introdução da cultura ágil pela perspectiva de equipe:

- A organização do trabalho deve ocorrer em grupo para uma melhor delegação de tarefas.
- A comunicação entre a equipe deve ser frequente, auxiliando nas dificuldades e disseminando informações importantes sobre o projeto.
- A equipe deve ser capaz de sincronizar as atividades que serão realizadas naquele dia e ter uma visão geral do projeto.

Comunicação com outros setores

“Suporte inadequado pode fazer com que até mesmo os bons fracassem na realização de seus trabalhos” (PRESSMAN, 2016). Embora as metodologias ágeis sejam vistas com mais frequência na área da programação, muitas empresas estão ampliando o uso dessas práticas para diversos setores.

Em razão da necessidade de aumentar a eficiência dos processos, compreender melhor as necessidades dos clientes e atender às novas demandas que surgem em um ritmo rápido, vem se tornando frequente o “ágil em escala”. “Ágil em escala” é quando um conjunto de práticas ágeis visa a abranger um número maior de grupos –nesse caso, outros setores da empresa.

Esses outros setores da empresa, ainda que passem a ter mais autonomia, seguirão com uma liderança, tal como os times de desenvolvedores. A nova visão agora é que todos os setores se comuniquem de forma a facilitar o progresso das atividades da empresa, alinhando-se para um mesmo objetivo.

Seguem alguns aspectos que tornam outros setores ágeis:

- Outros setores também passarão a ter mais autonomia e descentralização de informações, da mesma forma que já ocorre com as equipes de desenvolvimento.
- Se os setores da empresa já estiverem adaptados aos objetivos do time, ficará muito mais fácil diminuir o tempo de execução de determinados processos, pois os setores terão informações-base para dar retorno com mais agilidade.
- Os setores devem estar engajados entre si para aumentar a produtividade. Dessa forma, as demandas ficam “travadas” por menos tempo.
- A comunicação deve ser direta e objetiva entre os setores, auxiliando na redução de ruídos de comunicação.

Grande parte da cultura ágil se refere a ignorar parte do controle e confiar que todos querem ter o melhor desempenho possível, cada um fazendo sua parte e participando das reuniões diárias, trabalhando como equipe. Para auxiliar nesse processo de adaptação, também é válido aplicar as rotinas apresentadas neste conteúdo.