O que são métodos ágeis?

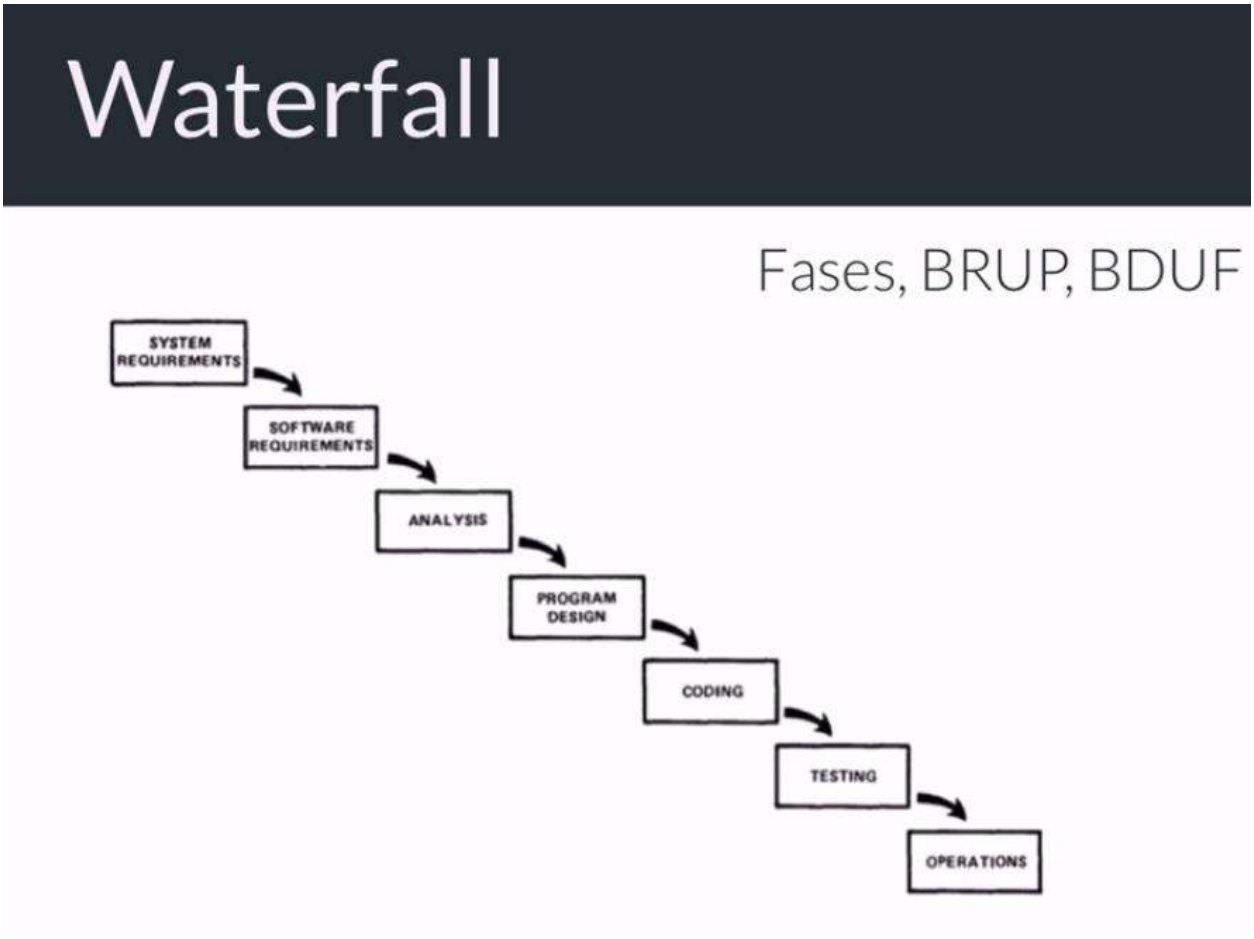
Esse é o curso de **Planejamento Ágil** da **Alura**. Nesta primeira parte, nós vamos introduzir um pouco sobre os conceitos de métodos ágeis: Por que eles apareceram? Em qual contexto? O que existia antes dos métodos ágeis? Essa será a introdução do curso antes de avançarmos no conteúdo.

Vamos falar sobre:

- Scrum;
- XP;
- Kanban;
- Agile;
- Problemas e resultados;

Quais eram os problemas com os chamados métodos tradicionais de desenvolvimento de software?

Basicamente, os métodos tradicionais seguiam o modelo conhecido como *Waterfall*, ou, o modelo Cascata, no qual se tinham diversas fases de um mesmo projeto. Observe o esquema abaixo:



Primeiro, era realizado um levantamento detalhado de tudo o que deveria ser desenvolvido, eram feitos levantamentos de requisitos com o cliente e, inclusive, muitos contratos eram fechados em cima desses requisitos, os chamados "Contratos de escopo fechado". Para se definir qual o escopo do projeto, é preciso ter uma ideia de tudo o que o cliente necessita. Assim, o cliente precisava decidir tudo o que ele deseja e não deseja logo no início de um projeto. Mas, é no começo que o cliente ainda não teve a oportunidade de experimentar e, nesse caso, é justamente nesse instante que ele é obrigado a fazer uma série de decisões.Quem fazia esse levantamento é uma equipe de análise.

Depois, seguia para uma nova etapa, de *software requirement*, em que, uma outra equipe fazia a análise de todos esses requisitos e cada etapa desse processo costumava demorar muito tempo. Por exemplo, cada passo teria vários meses de duração.

Após o processo de análise, passava-se para uma outra equipe que fazia o design, a modelagem e para fazer isso utilizava recursos da UML (que em inglês significa *Unified Modeling Language*) para modelar. Depois de muito tempo, começava-se a codificar: uma equipe codificava, outra equipe fazia os testes - se fizesse -. E como os testes estão no final de um processo era muito comum se pensar "- Poxa, não dá tempo de testar, vamos fazer a implantação porque o prazo está acabando e vamos estourá-lo." Então, o prazo dos testes ficava apertado pois, estava no final de um processo e isso acabava gerando muitos problemas de qualidade também.

Perceba que são várias etapas bem definidas, geralmente realizadas por pessoas com *skills* diferentes, por equipes, inclusive, distintas. Nessa lógica, apenas quando uma etapa está completa, passa-se para a próxima, sem a possibilidade de se retornar a anterior, a ideia era que as etapas avançassem linearmente para as próximas.

O grande problema dessa abordagem é que quando o cliente obtinha o produto final, o software, ele manifestava descontentamentos, "Não foi isso que eu pedi","não era isso que eu queria", "não era isso que eu precisava" e "isso não me atende."

Se por alguma razão o cliente precisasse parar um projeto no meio, ele não teria nada funcionando, nenhum resultado, uma vez que o produto só ficava pronto ao final do projeto. Portanto, mesmo o *ROI* - retorno sobre o investimento - o cliente só recebia ao final do projeto.

Além disso, como eu disse antes, no começo do processo o cliente sabe muito pouco sobre o projeto. Não se conhece ainda aquela equipe, aquele contexto do negócio e do próprio projeto que está sendo realizado. Então, é nesse momento que se fazem as piores decisões e, ao mesmo tempo, as mais importantes também. Isso gerava dois problemas, conhecidos na comunidade como **Big Requirements Up Front**, que é a ideia de você levantar todos os requisitos de antemão e de **Big Design Up Front**, de você tentar fazer todo o design de antemão.

Veremos que, utilizando métodos ágeis, isso é feito de uma maneira bem diferente! O processo é contínuo e contêm todas as etapas já mencionadas (levantar requisitos, analisar, modelar, codificar e testar). A grande diferença é que ele é feito em ciclos, com intervalos muito menores, o que resolve grande parte desses problemas! A cada

ciclo o cliente recebe um produto e já pode ver ele funcionando, assim, pode dar um feedback que oriente a equipe sobre o que está sendo feito.

Na abordagem tradicional, os requisitos pouco claros acabavam criando uma série de problemas. E, como não se podia voltar idealmente de uma etapa pra outra, tudo tinha que ser muito bem feito, muito bem levantado, o que acabava levando muito tempo.

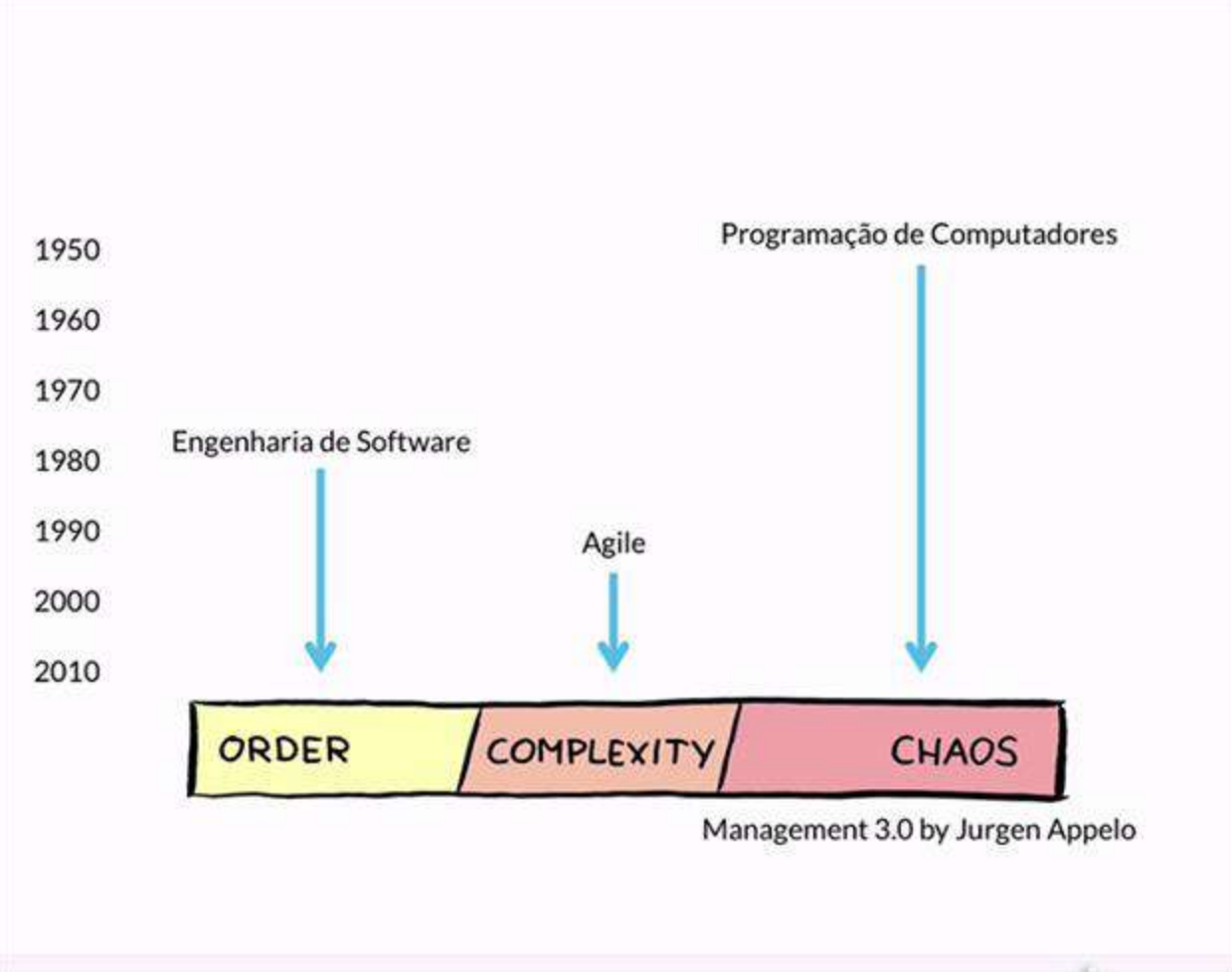
A resistência às mudanças, portanto, era grande. Justamente porque o escopo era fechado, assim, se o cliente reivindicasse alguma alteração, tinha que ocorrer uma negociação, uma *change request*, o que aumentava o custo de se ter aquele cliente. O processo em si, o método em si, resistia às mudanças e não incentivava o cliente a mudar para que aquele sistema atendesse melhor aos seus negócios, pelo contrário, não eram desejadas mudanças. O que era decidido no início devia ser mantido até o fim.

Perceba que esse "até o fim" podia ser um espaço muito grande de tempo, então, as mudanças iam encarecendo cada vez mais. Os projetos eram longos demais e tinham pouquíssimos *feedbacks* dos clientes. Assim, o mercado mudando, o cliente alterando suas ideias iniciais, o mundo ficando diferente... Esse processo passou a ser pouco aderente a esses câmbios.

O **Standish Group** afirma em relatório que apenas 50% desses requisitos eram implementados, ou seja, muita coisa acabava ficando de fora. Perceba que o levantamento detalhado feito no início nunca era implementado por completo. O que isso significa? Desperdício de dinheiro e tempo.

E 64% daquilo que foi implementado raramente era utilizado. Então era mais desperdício ainda! Utilizando **métodos ágeis** tentaremos endereçar todos esses problemas e evitar que esses desperdícios aconteçam.

Vamos falar de métodos ágeis. Observe o desenho do publicado por Jurgen Appelo, no livro "Management 3.0":



Essa figura mostra o seguinte:

- 1950: Existia "Programação de computadores"

A programação de computadores era algo caótico, não tinha ordem e tão pouco processo. Era tudo muito novo! A medida que a computação foi evoluindo, houve a necessidade de se ordenar melhor o processo de desenvolvimento dos *softwares*. Como se conheciam bem as engenharias, como a engenharia civil e engenharia mecânica muito das ideias dessas ciências foram levadas para os *softwares.*

Esse processo evoluiu até atingir a Engenharia de Software, que tentou deixar o *software*algo mais ordenado, nesse processo se compreendeu que as coisas eram

bastante previsíveis. Por exemplo, se compreendeu que se poderia tomar decisões de antemão, e manter essas decisões num ciclo grande de anos.

Os **métodos ágeis** são justamente o meio termo, onde você tem uma certa previsibilidade, mas não uma previsibilidade total. Por isso, você trabalha em interações em ciclos de *feedback* contínuo, pra que você possa rapidamente ir respondendo às mudanças que você não pode prever desde o início do projeto.

Esse foi o tema do "Manifesto Ágil" que deu origem também ao grupo de pessoas, que recebeu o mesmo nome, muito respeitadas na comunidade de desenvolvimento de software em geral, com competências distintas, em desenvolvimento, programação, design e várias áreas de *software*. Eles disseram o seguinte:

"Estamos descobrindo maneiras melhores de desenvolver softwares fazendo nós mesmos e ajudando outros a fazê-los. Através desse trabalho, passamos a valorizar:

- Indivíduos e a interação entre eles, mais do que processos e ferramentas. A questão não é que processos e ferramentas não são importantes - eles são importantes - mas as pessoas são mais importantes! Essa é a ideia: elas têm que ser colocadas em primeiro lugar.
    - O **Software em funcionamento** é mais importante que a documentação abrangente. O cliente não recebe valor diretamente pela documentação em si, mas ele recebe pelo software em funcionamento, que é o que ele precisa no fim das contas. Não é que nós não precisamos escrever documentação - muito pelo contrário, documentação que agrega valor tem que ser escrita -, o que nós não precisamos é daquela documentação que apenas vai ser um desperdício.
    - A **Colaboração com o cliente** é mais importante do que a negociação contratual. Não vale ter um contrato com um cliente e forçá-lo para manter o contrato, isto é, o escopo fechado até o final. Nós queremos construir *software*sque agreguem valor aos clientes, por isso, nós temos que colaborar com o cliente.
    - É necessário **responder às mudanças** mais que seguir um plano. Temos que estar em constante estado de inspeção, de

> adaptação, de responder às mudanças do mundo, às mudanças das necessidades do nosso cliente."

De uma maneira simplista, esse é o "Manifesto Ágil". À medida que formos avançando no treinamento, retornaremos para essas ideias e vamos mostrando como, na prática, o desenvolvimento ágil transforma esses princípios em realidade e ação.

Além do Manifesto, nós temos mais 12 princípios ágeis:

1) Satisfazer o cliente.

2) Dar boas-vindas às mudanças.

3) Entregar com frequência resultados.

4) Trabalhar como um time, não apenas um grupo de pessoas reunidas, ou seja, ser uma equipe de verdade.

5) Motivar as pessoas.

6) Comunicação face a face é preferida em relação a outros tipos de comunicação, como comunicação escrita, a troca de e-mail, telefone. Assim se potencializa a comunicação entre as pessoas.

7) Medir o software em funcionamento.

8) Manter um ritmo sustentável: Equipes exageradamente ágeis não são equipes que fazem milhares de horas extras, viram madrugadas e etc. É necessário um ritmo sustentável e saudável, que possa ser mantido ao longo do tempo e com qualidade de vida para as pessoas envolvidas.

9) A qualidade deve ser algo bem visto, bem preparado e importante também. Então, "deixar os testes para o final e fazer se der tempo" não acontece com métodos ágeis. Os testes devem fazer parte do dia a dia e não devem ser apenas uma fase final.

10) Manter as coisas sempre simples.

11) Os designs têm que ser evolutivos. Então, não é uma coisa também resolvida de antemão, é algo constantemente adaptado.

12) Refletir regularmente sobre o processo, sobre o que se está fazendo e analisar de que maneira nós podemos melhorar continuamente. Essa é a ideia de melhoria contínua, a de refletir regularmente. "

E, para implementar essas ideias, existem vários métodos ágeis e vários processos ágeis. Alguns desses processos já existiam até antes do manifesto. Esse é o "guarda-chuva ágil".

Não existe uma única maneira de se utilizar "Ágil", de se fazer "Ágil". Na verdade, existem várias abordagens. A essência é a mesma, mas a forma pode mudar e, por isso, é preciso conhecer alguns desses métodos para que se entenda qual deles vai se adaptar melhor às suas necessidades, ao seu contexto, sua equipe e seu time.

Nós vamos falar um pouco sobre **Scrum**, **kanban** e **XP**, mas existem outros que você pode pesquisar depois.

É importante ver esses métodos como ferramentas. Muitas vezes nos deparamos com discussões de que "Scrum é melhor que XP", "XP é melhor que Scrum" ou "kanban é o melhor". Na verdade, são métodos distintos. Então, dependendo da sua necessidade, você vai utilizar essas ferramentas de maneiras diferentes.

Métodos podem ser prescritivos e quanto mais eles são, mais coisas bem definidas encontraremos no método. RUP é extremamente prescritivo - ele define vários papéis, artefatos e práticas que devem ser realizadas - são 120 no total. O XP só tem 13, o Scrum tem 9, o Kanban tem 3.

Então, a grande questão é a seguinte, o **kanban**, por exemplo, é um dos métodos mais adaptativos que existem, portanto, as empresas que utilizam o **kanban**, provavelmente possuem no dia a dia, poucas coisas em comum, ou seja, possuem a necessidade de adaptar constantemente o seu cotidiano. Já as empresas que usam o **RUP** vão ter muita coisa em comum, isto é, suas necessidades mudam.

Dessa forma, percebemos que os métodos variam dentro da linha de prescrição - de ser mais prescritivo ou menos prescritivo e, naturalmente, mais adaptativos.

Da mesma forma que qualquer ferramenta pode ser mal utilizada, muitas vezes, o insucesso de um projeto pode não ser por demérito da metodologia ou do processo empregado. Você pode simplesmente estar utilizando, por exemplo, uma serra elétrica como se ela fosse um machado, o que provavelmente acarretará a quebra da serra elétrica em vez do corte da árvore. Esse é um exemplo que o Henrik Kniberg da comunidade ágil sempre dá: "Se você usar a serra elétrica para ser serra elétrica, muito provavelmente, você obterá os resultados esperados. Caso contrário, ao utilizar uma ferramenta, por melhor que ela seja, de forma inadequada ou fora das condições em que ela é indicada, o resultado obtido, provavelmente, será bem diferente do que era esperado."

Portanto, é preciso ter muito cuidado com isso! Como são ferramentas, você pode usar todas elas para resolver vários problemas. Uma não é melhor que a outra, elas são apenas ferramentas diferentes.