**UNINOVE – Universidade nove de julho**

**CENTRO DE PÓS-GRADUAÇÃO "LATO SENSU" EM MASTER IN PROJECT MANAGEMENT**

**METODOLOGIA ÀGIL: Análise dos benefícios para o processo de desenvolvimento de software**

**GILCICLEIDE MORERIA DE ANDRADE VECHI**

**Centro de Pós – Graduação - Memorial**

**2017**

**GILCICLEIDE MORERIA DE ANDRADE VECHI**

**METODOLOGIA ÀGIL: Análise dos benefícios para o processo de desenvolvimento de software**

Trabalho de Conclusão de Curso apresentado no curso de Pós-Graduação "Lato Sensu" em Master In Project Management – nível especialização modalidade Presencial Centro de Pós-Graduação - Memorial. Orientado pelo professor [Norival Mantovani](http://turismo.cps.sp.gov.br/ead/user/view.php?id=435&course=15) como requisito parcial para obtenção do título de Especialização em Gerenciamento de Projeto.

**São Paulo – SP**

**2017**

**RESUMO**

Esta pesquisa foi realizada com método qualitativo, a fim de colher informações sobre novos métodos de gerenciamentos, quais empresas utilizam seus conceitos, quais vantagens e desvantagem de se aplicar e se a metodologia Ágil pode substituir as metodologias tradicionais.

O conteúdo desta pesquisa conseguiu um vasto detalhamento sobre conceitos, documentos e ferramentas mais utilizados, principais papeis dos funcionários dentro da metodologia ágil e realizou comparações com o modelo cascata, considerado metodologia tradicional. Neste trabalho é possível entender quais as principais diferenças entre os dois métodos de gerenciamento, com detalhes sobre riscos, vantagens e desvantagens de cada uma.

Com isso, o autor conseguiu chegar à conclusão que as empresas trabalham com metodologias variadas, suas escolhas e aplicações dos métodos de gerenciamento que são usados, dependem da necessidade do cliente e do projeto e foi possível entender que não existem certo ou errado, os projetos devem passar por uma profunda análise e assim definir os métodos utilizados.

**LISTA DE FIGURAS**

Figura 1 - Entregas por etapas conforme ocorre no conceito ágil............................................07

Figura 2 - Melhorias contínuas conforme ocorre no conceito ágil...........................................08

Figura 3 - Visão geral do Scrum no ágil...................................................................................09

Figura 4 – Cenário das funcionalidades BDD..........................................................................11

Figura 5 – Quando de tarefas, Kanban......................................................................................14

Figura 6 – O modelo em cascata ou Waterfall..........................................................................18

**SUMÁRIO**

**1 INTRODUÇÃO--------------------------------------------------------------------------------06**

1.1 PROBLEMATIZAÇÃO------------------------------------------------------------------------06

1.2 HIPÓTESE---------------------------------------------------------------------------------------06

1.3 JUSTIFICATIVA E OBJETIVO-------------------------------------------------------------07

**2 FUNDAMENTAÇÃO TEÓRICA----------------------------------------------------------07**

2.1 HISTÓRIA DA METODOLOGIA ÁGIL---------------------------------------------------07

2.2 CONCEITO E CARACTERÍSTICAS DO ÁGIL------------------------------------------08

**2.2.1 Os 12 princípios do Ágil----------------------------------------------------------------------10**

2.3 VANTANGENS DA APLICAÇÃO DA METODOLOGIA ÁGIL----------------------11

**2.3.1 Automação de Teste em Ágil----------------------------------------------------------------12**

**2.3.2 Documentações usadas no modelo Ágil----------------------------------------------------12**

2.3.2.1 Método de gerenciamento de projetos ágeis-------------------------------------------------14

**2.3.3 Principais funções dos recursos no Ágil---------------------------------------------------15**

**2.3.4 Riscos da aplicação da Metodologia Ágil--------------------------------------------------16**

**2.3.5 As principais Empresas que trabalham com ágil no Brasil---------------------------17**

2.4 VANTAGENS DO MODELO CASCATA-------------------------------------------------17

**2.4.1 Documentações usadas no modelo cascata--------------------------------------------18**

**2.4.2 Principais riscos no modelo cascata--------------------------------------------------------19**

**3 METODOLOGIA-----------------------------------------------------------------------------19**

**4 ANÁLISE DOS RESULTADOS------------------------------------------------------------19**

4.1 ANÁLISE SWOT DA METODOLOGIA ÁGIL-------------------------------------------20

4.2 ANÁLISE SWOT DA METODOLOGIA CASCATA------------------------------------21

**5 CONCLUSÃO----------------------------------------------------------------------------------22**

**6 BIBLIOGRAFIA-------------------------------------------------------------------------------23**

**7 GLOSSÁRIO DA ÁREA DE INFORMÁTICA-----------------------------------------25**

**1 INTRODUÇÃO**

A partir dos anos 90, começaram a surgir processos alternativos de desenvolvimento de software, com novos processos e foram apelidados de “leves” *lightwei­ght*, em oposição aos métodos anteriores, “pesados” *heavyweight* e só passaram a ser chamadas de ágeis após 2001.

Com a inovação tecnológica, cada vez mais rápida, as empresas perceberam que precisavam diminuir a burocracia em seus processos, estão deixando de usar os modelos tradicionais de gerenciamentos, os quais são conside­rados excessivamente regrados, lentos, burocráticos e inadequados, com processos engessados e os requisitos totalmente definidos para recorrer a um novo método de desenvolvimento de projetos, a Metodologia Ágil, a qual tem como proposta principal, diminuindo a burocracia para realizar mais entregas, fazendo com que grandes projetos sejam granulado/divididos em pequenas etapas e cada uma destas etapas seja entregáveis, os quais podem ser implantados para o ambiente de produção individualmente.

No ágil, os desenvolvedores e os testes trabalham em conjunto com o time do cliente, participam das reuniões desde a ideia inicial, ondem todos podem expor seus pontos de vistas e agregar valor ao produto. O time de qualidade realiza as escritas dos cenários e execuções dos testes, são responsáveis pela a cobertura das variáveis dos testes, também, são a ponte entre o que está sendo desenvolvido e o que foi solicitado. O gestor escreve as histórias representando os desejos e necessidades dos usuários do software. Os testes são realizados durante o desenvolvimento, conforme são desenvolvidas etapas das atividades e todos do time são responsáveis pela qualidade da entrega final, não apenas a equipe de testes, portanto todos da equipe são alto gerenciáveis, responsáveis pelo término das suas atividades.

**1.1 PROBLEMATIZAÇÃO**

Em um ambiente de muita concorrência e com a inovação tecnológica a todo vapor, é natural que as empresas busquem aperfeiçoarem seus processos e diminuam suas burocracias. Assim conseguem acompanhar ou até ultrapassar seus concorrentes, realizando mais entregas com sistemas mais modernos e em um menor prazo.

Dado este contexto, propôs-se a investigação do seguinte problema de pesquisa: Será que todo projeto pode utilizar a metodologia ágil, as quais diminuem os processos burocráticos, tem maior flexibilidade, com entregas parciais e contínuas ou alguns projetos ainda precisam ser gerenciados com formas mais definidas e seguras como nos modelos tradicionais, mais conhecidos como cascatas?

**1.2 HIPÓTESE**

Se os conceitos da metodologia Ágil forem aplicados da forma correta, as empresas conseguem entregar mais desenvolvimentos de *softwares*, do que entregariam nos modelos de tradicionais de gerenciamentos, por exemplo, no modelo cascata mais conhecido como *Waterfall*, os quais precisam ter concluído o desenvolvimento de todo o escopo funcional, para ser enviados à equipe responsável pela cobertura dos testes de qualidade, então são realizadas as correções ou melhorias identificadas, são realizados novos testes após as correções e se não houverem mais alterações, é realizada uma reunião de aceite e após a conclusão de todas as etapas, os projetos são enviados para implantação.

Ou mesmo com todas as vantagens proporcionadas pelo Ágil, os métodos de gerenciamentos tradicionais precisam continuar sendo aplicados conforme hajam necessidades dos projetos.

**1.3 JUSTIFICATIVA E OBJETIVO**

O autor pretende entender melhor os conceitos, objetivos, pontos fracos e fortes da aplicação da metodologia Ágil e também das metodologias tradicionais de gerenciamento, dentro das grandes empresas, as quais estão adotando a metodologia Ágil e pregando como a melhor solução para os projetos de desenvolvimento.

O objetivo é conhecer as vantagens e desvantagem deste novo método de gerenciamento, entender seus conceitos as aplicações dentro das empresas. Com isso, analisar se o Ágil se trata do melhor método para os dias de hoje, já que, as grandes Organizações vêm adotando esta nova forma de gerenciar os desenvolvimentos dos *softwares*. As empresas estão proporcionando mais inovações para seus clientes, em menos tempo e de forma contínua.

**2 FUNDAMENTAÇÃO TEÓRICA**

**2.1 HISTÓRIA DA METODOLOGIA ÁGIL**

A partir dos anos 90, começaram a surgir processos alternativos de desenvolvimento de software, com novos processos e só passaram a ser chamadas de ágeis após 2001. Com o surgimento do movimento ágil, realizado por 17 profissionais da área de desenvolvimento de sistemas os quais se uniram em 2001 e realizaram o “Manifesto Ágil”. De acordo com Manifesto Ágil (2001), foram descobertas melhores maneiras para o desenvolvimento de *softwares* com as seguintes premissas: Indivíduos e interações sobre processos e ferramentas, software de trabalho sobre documentação abrangente, colaboração do cliente em relação à negociação de contratos e respondendo a mudança ao seguir um plano. Ou seja, as pessoas são mais importantes que os processos e as documentações.

De acordo com Sabbagh (2013), o Manifesto Ágil reconhece que a utilização de processos, ferramentas, documentação, contratos e planos pode ser importante para o sucesso do projeto, mas são ainda mais importantes os chamados valores Ágeis: os indivíduos e interações entre eles, software (ou produto) em funcionamento, colaboração com o cliente e responder a mudanças.

* **Autores do Manifesto Ágil**

Kent Beck: Criador da Extreme Programming – XP e JUnit, framework usado no desenvolvimento de testes de unidade e uma das maiores referências do mundo ágil.

Jeff Sutherland e Ken Schwaber: Inventores do Scrum.

Martin Fowler: Autor dos livros “Analysis Patterns”, “Planning Extreme Programming” e “Refactoring”, importante referência em design para desenvolvedores.

Dave Thomas e Andrew Hunt: Coautores do livro “O Programador Pragmático”, pregam a simplicidade e leveza no desenvolvimento, além de metodologias centradas em pessoas.

Alistair Cockburn: Criador da família de métodos ágeis chamada de Crystal.

Ward Cunningham: Criador do método de design CRC e contribuidor para outras metodologias, incluindo XP.

Arie van Bennekum: Ativamente envolvido no consórcio DSDM (Dynamic Systems Developmen tMethod).

Brian Marick : Representante da comunidade de testes e das ideias do que o Agile Testing pode ser.

Jim Highsmith: Autor do método Adaptive Software Development (ASD) e do livro com mesmo nome.

Robert C. Martin: Experiente em XP, autor do livro “Principles, Patterns, and Practicesof Agile Software Developmente”.

Ron Jeffries: Primeiro coach em XP, proprietário do XProgramming.com e coautor do livro “*Extreme ProgrammingInstalled*”.

Jon Kern: Programador e arquiteto experiente em diversas linguagens.

Peter Coad: Dono da *TogetherSoft* e um dos criadores da FDD.

Mike Beedle: Adotou *Scrum* e XP como metodologias ágeis com sucesso em diversos projetos, coautor do livro “*Scrum, Agile Software Development*”, com Ken Schwaber.

Stephen J. Mellor: Também conhecido como Steve Mellor, autor de “*Executable* UML” e “MDA*Distilled*”.

James Grenning: Um dos criadores da técnica conhecida como *Planning Poker*. Autor de “*Test-DrivenDevelopment*”.

**2.2 CONCEITO E CARACTERÍSTICAS DO ÁGIL**

**Satisfação do cliente através da entrega contínua:** Em projetos desenvolvidos com métodos tradicionais (Cascata / *Waterfall*), as entregas são demoradas, pois as implantações ocorrem após as colusões de todas as etapas do projeto. Sendo assim, o cliente ver o resultado após muito tempo e se ele precisar fazer uma alteração ou melhorias vai precisar passar por todo o processo novamente.

Já no ágil, as implantações são realizadas pelas etapas que vão sendo concluídas e com isso, em pouco tempo o cliente já ver o resultado do que está sendo desenvolvido e já pode ir incluindo suas alterações e melhorias durante esse processo de desenvolvimento. Sendo assim, como as entregas ocorrem de forma contínua, à satisfação do cliente é muito mais garantida.

**Entrega de software funcional com frequência:** No método ágil, as entregas são formadas por conjuntos de pequenas histórias que formam uma *Sprint*. A Sprint pode ser realizada de 1 a 4 semanas, isso contando o desenvolvimento e os testes de qualidade o quais, precisam formar entregas que agregam valor ao cliente, ou seja, funcionalidades independentes de outras etapas que faltam ser desenvolvidas. Sendo assim, são implantadas para uso do cliente, no ambiente de produção, em curto prazo e com frequência.

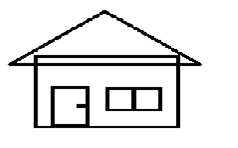
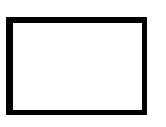


Figura 1 – Entregas por etapas conforme ocorre no conceito ágil.

Fonte: Elaborado pelo autor, 2017.

**Mudanças são bem-vindas:** As mudanças no Escopo inicial é uma das grandes vantagens do método ágil, o cliente ou até mesmo a equipe, conseguem ter um retorno rápido do sistema que está sendo construído, com isso, conseguem implantar melhorias ou mudanças para o projeto de acordo com a necessidade ou observações dos usuários.

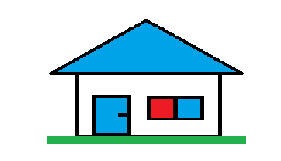
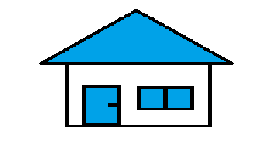


Figura 2 – Melhorias contínuas conforme ocorre no conceito ágil.

Fonte: Elaborado pelo autor, 2017.

**Executivos e desenvolvedores trabalham em conjunto:** Para os sistemas serem desenvolvidos com qualidade, nos projetos que estão trabalhando com o método ágil, as equipes de desenvolvimento, de testes, de gerenciamento e os clientes/usuários, costumam ficar fisicamente em um mesmo local, para ocorrer reuniões e alinhamentos de formas mais rápida, sendo assim todos se tornam um grande time.

**Interação face a face:** Essa interação face a face ajuda diminuir a burocracia e os prazos de entregas já que, os desenvolvedores e os testadores, conseguem alinhar duvidas ou melhorias, falando diretamente com o cliente e ter o retorno imediato, sem a necessidade de esperar decisões ou aprovações executivas para pequenos detalhes nas construções de serviços ou telas.

**Desenvolvimento de projetos em torno de pessoas motivadas:** O conhecimento das pessoas é mais importante para o ágil, do que as documentações e os processos. Com isso, a equipe fica mais motivada por ver que suas ideias e suas observações são analisadas, então todos se sentem importantes e valorizados dentro do projeto.

**Progresso medido através de softwares funcionais:** Para gerenciamento dos projetos pelos métodos Ágeis, muitas empresas estão adotando *softwares* mais simplificados, onde pode ser parametrizado de acordo com a necessidade. Os *softwares* que vem sendo mais usado pelas empresas são:

* **Jira:** Ele permite o monitoramento de tarefas e acompanhamento de projetos garantindo o gerenciamento de todas as suas atividades em único lugar.
* **Confluence:** É um *software* de colaboração de conteúdo, onde são arquivados de evidências, os critérios de aceite com as histórias do projeto e o mapeamento dos testes.

De acordo com Atlassian (2017), a [integração do *Confluence* com o Jira permitem que os projetos tenham mais transparência e a abertura dos erros encontrados ficam vinculados de forma automática no Jira.](https://br.atlassian.com/software/confluence/jira-integration)

**Times auto gerenciáveis:** O nível de maturidade das pessoas e da equipe precisa ser alto, pois no ágil, o time é auto gerenciável e cada um tem a reponsabilidade de gerenciar seu próprio trabalho, más, ao mesmo tempo todos são responsáveis por olhar e orientar o trabalho do outro.

**Visão geral do *Scrum* no Ágil:** No ágil, os projetos são divididos em ciclos mensais, mais conhecidos como ***Sprint***, que é o prazo para realizar o conjunto de atividades estimadas durante a reunião de planejamento. Durante o ciclo vigente são feitas reuniões diárias com os envolvidos, onde cada um conta o que fez o que vai fazer e quais são seus impedimentos. As atividades não realizadas durante este ciclo vão para uma lista de pendências também, chamadas de *backlog*.

No final do Sprint, são apresentadas as funcionalidades desenvolvidas para o dono do projeto, que avalia se está de acordo com o que esperava e se podem ser implantadas as funcionalidades. Para encerrar as atividades desta *Sprint*, é realizada uma reunião de retrospectiva, onde todos avaliam os acontecimentos que ajudaram ou atrapalharam suas atividades e o projeto como um todo.

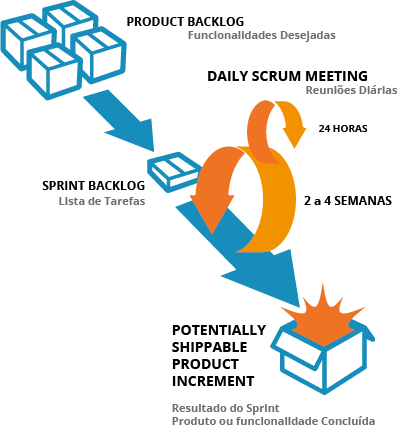


Figura 3 – Visão geral do *Scrum* no ágil.

Fonte: BRQ, 2017.

**2.2.1 Os 12 princípios do Ágil**

* A maior prioridade é satisfazer o cliente com entregas contí­nuas e adiantadas de software com os valores agregados.
* Mudanças nos requisitos são bem-vindas, mesmo tardiamente no de­senvolvimento. Os processos ágeis tiram vantagens das mudanças, visando as vantagens competitivas para os clientes.
* Entregar frequentemente o software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
* Pessoas de negócios e desenvolvedores devem trabalhar diariamen­te em conjunto por todo o projeto.
* Construir projetos em torno de indivíduos motivados. Dê a eles o am­biente e o suporte necessário e confie neles para realizar o trabalho.
* O método mais eficiente e eficaz de transmitir informações, para a equi­pe e entre a equipe de desenvolvimento é a conversa frente a frente.
* Software funcional é a medida primária de progresso.
* Os processos ágeis promovem desenvolvimento sustentável. Os pa­trocinadores, desenvolvedores e usuários devem ser capazes de man­ter um ritmo constante sempre.
* A atenção à excelência técnica e bom design aumenta a agi­lidade.
* Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
* As melhores arquiteturas, requisitos e design emergem de times auto gerenciáveis.
* Os intervalos regulares, o time reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

**2.3 VANTANGENS DA APLICAÇÃO DA METODOLOGIA ÁGIL**

De acordo com Engelbreth (2017), na metodologia ágil, as empresas normalmente são flexíveis e se adaptam às mudanças de forma rápida. Ela implementa mais rápido e é capaz de aproveitar novas oportunidades à medida que aparecem. A tomada de decisão ocorre naturalmente, pois nesta metodologia o projeto não tem um escopo fechado e podem sofrer alterações.

O projeto é acompanhado de perto pelo cliente e pelos gerentes que estão fisicamente próximo ao time de desenvolvimento e de qualidade que realizam os testes. Ocorrem reuniões diárias, os riscos e o andamento do projeto são diariamente estudados e gerenciados através das *daily*, reuniões de 15 minutos que ocorrem com os envolvidos, todos os dias onde a comunicação é simples e direta.

**2.3.1 Automação de Teste em Ágil**

De acordo com Nogueira (2016), a automação de teste no ágil é obrigatória, independente se esta ação será feita por um testador ou pelos membros do time. O importante é planejar a automação em cada fase. Com automação trabalhando em conjunto com o time de testes funcionais e com os desenvolvedores, os projetos conseguem ter muito mais cenários testados, a qualidade passa ter filtro maior e consequentemente menos erros nas implantações. Há quatro níveis que identifica o planejamento da automação de teste em projetos ágeis.

**Nível de *Release*:** Onde se tem a visão do todo e planejamento em alto nível sobre automação, onde o Time já tem uma boa visão do todo e acaba sendo o melhor momento para pensar em ferramentas, ambiente e técnicas de testes, fazer experimentos, levantar possíveis problemas de *design* e os riscos dos projetos.

**Nível de *Feature:*** Visão de uma funcionalidade e planejamento específico.

**Nível de *Story*:** Onde a visão é macro do que será feito no planejamento, quais serão os cenários cobertos pela automação. **Exemplo:** O arquivo **"**<nome\_funcionalidade>. *feature*" contém o BDD a ser testado e deve ser salvo na estrutura do projeto.

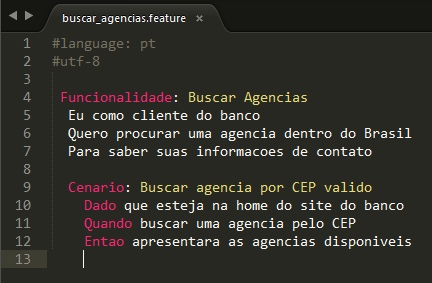
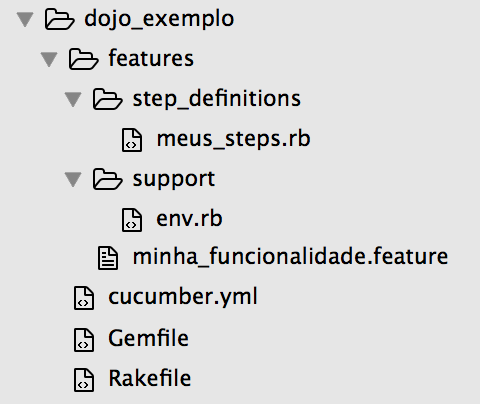


Figura 4 – Cenário das funcionalidades BDD.

Fonte: Matsumoto, 2015

**Nível de tarefas**: Visão micro e aplicação prática de todo o planejamento, onde são realizadas as automações dos testes passo a passo nas funcionalidades desenvolvidas. Os comandos do *Capybara* devem ser inseridos dentro dos *Steps* gerados pelo *Cucumber*.

**2.3.2 Documentações usadas no modelo Ágil**

De acordo com Renzi (2014), os principais documentos funcionais, dentro da metodologia ágil são:

* **Backlog**

Para iniciar o processo de desenvolvimento da forma correta, dentro da metodologia ágil, o Analista Funcional deve acompanhar o PO, Dono do Produto, na criação do backlog todos os critérios de aceite. Então, a partir daí o Analista Funcional deve entender melhor as necessidades do projeto e documentar este entendimento de uma forma resumida e apresentar ao time de desenvolvimento, nesta apresentação devem ser tiradas as dúvidas e nivelando o conhecimento entre todos. Após o alinhamento e aprovações de todos envolvidos, a base do sistema inicia a arquitetura.

* **Critério de Aceite**

O Analista Funcional precisa criar uma documentação completa, detalhada e assim chances de erros. Após criar a documentação de um Sprint de ser feito um alinhamento com o Líder Técnico para antecipar possíveis imprevistos, como por exemplo alguma limitação técnica para o desenvolvimento de uma funcionalidade. Após o alinhamento com o líder técnico deve ser feita uma validação com o P.O, Dono do Produto, para que alinhada as regras de negócio, padrões de sistemas e se o documento atende a necessidade do projeto. Após estas etapas, a documentação funcional fica pronta e deve ser passada aos desenvolvedores, analistas de testes e todos envolvidos do projeto.

**Funcionalidade:** É um comportamento ou ação que o sistema oferece ao seu usuário baseada em uma entrada e uma saída, que contém um ou mais cenários diferentes. O nome é determinado pela combinação de um verbo e um substantivo e ela costuma ser escrita pelo PO e/ou Funcional, dentro critério de aceito.

Exemplo de funcionalidade: Buscar Agências.

Eu como cliente do banco.

Quero procurar uma agência dentro do Brasil.

Para saber suas informações de contato.

* ***Planning***

Nesta etapa o time já deve ter a documentação funcional em suas mãos e o *planning* serve para realizar o entendimento desta documentação para o time e o PO e também, para ser passado as prioridades do *backlog* e o time entende e expressa a complexidade das histórias e volta para que seja definido o que cabe dentro do prazo da *Sprint* para ser desenvolvido e homologado.

* ***BDD***

É técnica de desenvolvimento ágil que objetivo a integrar regras de negócios com linguagem de programação, focando o comportamento do software. As escritas das funcionalidades são realizadas pelo PO, Dono do Projeto ou pelo Analista funcional. Já os cenários, na maior parte dos projetos ágeis, são criados por quem vai testar a aplicação, os analistas de testes.

**Cenários de testes:** Descreve um conjunto ordenado de comportamentos baseado em uma entrada para alcançar um resultado específico de uma funcionalidade. As palavras chaves do cenário (Dado / Quando / Então / E) são base para o funcionamento do teste.

Exemplo de um Cenário: Buscar agência por CEP válido.

Dado que esteja na home do site do banco.

Quando buscar uma agência pelo CEP.

Então apresentará as agências disponíveis

* **Tipos de Testes**

Existem diversos tipos de testes, mas, nem todos são aplicados dentro dos projetos. Exemplo dos principais testes usados no ágil:

**Funcional:** Também chamado como teste da caixa preta e tem com foco a análise das entradas e saídas de dados dentro de um programa ou sistema. No teste funcional, a aplicação é analisada com o olhar do cliente, ou seja, o analista de testes não valida os códigos, os seus testes são baseados nas experiências de diferentes perfis de usuários.

**Não funcional:** Os testes não funcionais são usados para prever possíveis erros que os usuários podem cometer e com a identificação prévia destes erros, a equipe pode encontrar e corrigir falhas no desenvolvimento do sistema.

**Unidade:** São os testes de cada pedaço da funcionalidade desenvolvida e os testes podem ser realizados conforme são desenvolvidas as telas. Nestes testes são validados nomes dos campos, limites máximos e mínimos de caracteres, mascaras dos campos e qualquer tipo de detalhe específico.

**Integração:** São os testes de cada pedaço da funcionalidade desenvolvida e os testes podem ser realizados conforme são desenvolvidas as telas. Nestes testes são validados nomes dos campos, limites máximos e mínimos de caracteres, mascaras dos campos e qualquer tipo de detalhe específico.

**Aceitação:** O teste é direcionado a um grupo de usuários finais que irá utilizar o sistema. É um dos últimos testes aplicados no software, servindo para avaliar o nível de aceitação dele perante o público. Os avaliadores observam como o usuário mexe no sistema, navegam, quais são as suas dificuldades, se o sistema responde adequadamente ao que os usuários fazem etc. São realizados cenários que simulam as rotinas onde o software seria usado e os testes ficam mais próximos da realidade. Também útil para avaliar quais requisitos o usuário esperava que o programa atendesse, especialmente se ele for um cliente que encomendou o software.

**Usabilidade:** O teste com o usuário no início significa a criação inicial de protótipos, normalmente esboços e modelos descritos como protótipos de baixa fidelidade e os protótipos de alta fidelidade devem ser criados posteriormente. Os testes de usabilidade devem ocorrer logo no início, mas, deve avaliar o sistema do ponto de vista do usuário final.

* **Criticidade de testes.**

Alta: Teste de suma importância para as funcionalidades desenvolvidas, essencial para o escopo solicitado no critério de aceite.

Média: Teste de cenários que ocorrem com menos frequência no ponto de vista dos usuários, mas, que são necessários para uma boa cobertura de testes.

Baixa: Testes de situações específicas, raros de acontecer com os usuários do sistema e que podem ficar de fora da cobertura caso não haja tempo hábil para testar.

* **Status de testes.**

Passou: Cenários de testes que foram concluídos com sucesso.

Falhou: Cenários de testes que apresentaram erros que precisam de correções.

Bloqueado: São cenários que estão impactados por erros impeditivos.

N/A: Cenários que estão ou ficaram fora do escopo final.

Não executado: Cenários de teste não iniciados.

* **Confluence.**

É uma ferramenta que armazena os documentos gerados durante a *Sprint*, ele proporciona um lugar centralizado para manter o trabalho da equipe organizado e acessível, facilita localizar as informações necessárias e pode ser personalizado de modo a se adaptar a todo e qualquer tipo de necessidade do projeto. Em geral, as empresas que trabalham com ágil, costumam subir os documentos de evidências, cenários e critérios de aceite após o termino da *Sprint*.

**2.3.2.1 Método de gerenciamento de projetos ágeis.**

No ágil, existem dois métodos mais usados nos gerenciamentos dos projetos. Algumas empresas preferem que as tarefas dos seus projetos fiquem em um quadro, onde os status das tarefas ficam visíveis e gerenciadas pelo time, como o ocorre com o *Kaban*. Já outras empresas preferem que as tarefas sejam gerenciadas por um sistema que conseguem gerar relatórios por status, por funcionalidade e da forma que atender melhor ao projeto, o sistema que mais se adaptam com as necessidades dos projetos ágeis é o Jira.

* **Kanban**

É um quadro de trabalho que organiza as tarefas da Sprint, são basicamente quatro estados, pode ocorrer uma variação para cada projeto.

*Baklog* ou A Fazer: Tarefas ainda não iniciadas.

Fila ou Em Andamento: Tarefas em desenvolvimento.

Fazendo ou Em Teste: Tarefas desenvolvidas e sendo testadas.

Concluído: Tarefas desenvolvidas e sendo testadas.

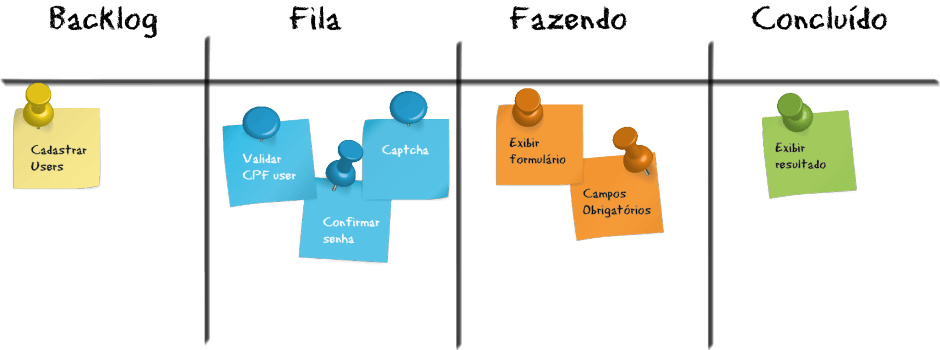


Figura 5 – Quando de tarefas, Kanban.

Fonte: BRQ, 2017.

* **Jira.**

O sistema de gerenciamento de atividades, muito usado na metodologia ágil, por se tratar de uma ferramenta simples que organiza as tarefas e os erros encontrados no projeto. É de fácil visualização para acompanhamento dos status e das horas gastas por cada atividade cadastrada.

* **Campos de pendências no Jira.**

Campo: Descrição

Projeto: Nome do projeto principal a qual a pendência pertence.

Chave: Um identificador exclusivo para essa pendência, exemplo: PFX-120.

Resumo: Um breve resumo de uma linha da pendência.

Tipo: Selecionar opção da lista de tipos.

Afeta a versão: A [versão](https://confluence.atlassian.com/jsbr/gerenciar-versoes-920354914.html) do projeto em que a pendência ocorre ou ocorreu.

Corrigido na versão: A [versão](https://confluence.atlassian.com/jsbr/gerenciar-versoes-920354914.html) do projeto em que a pendência será corrigida ou foi corrigida.

Status: Selecionar opção da lista de status classificar o ciclo de vida do projeto.

* **Prioridade:** A importância da pendência em relação às demais.

Prioridade: Descrição

Mais alta: Prioridade máxima. Indica que essa pendência tem precedência sobre todas as outras.

Alta: Indica que essa pendência está causando um problema e requer atenção urgente.

Média: Indica que essa pendência tem um impacto significativo.

Baixa: Indica que essa pendência tem um impacto relativamente reduzido.

Mais baixa: Prioridade mínima.

* **Resolução:** Selecionar opção da lista de resoluções a qual classifica se foi concluída.

Resolução: Descrição

Concluído: A pendência será realizada.

Não realizar: A pendência não será realizada.

Duplicada: A pendência já em análise.

Não é possível reproduzir: Normalmente uma instabilidade a qual não é possível ser reproduzida.

* **Afeta a versão:** A [versão](https://confluence.atlassian.com/jsbr/gerenciar-versoes-920354914.html) do projeto em que a pendência ocorre ou ocorreu.

Corrigido na versão: A [versão](https://confluence.atlassian.com/jsbr/gerenciar-versoes-920354914.html) do projeto em que a pendência será corrigida ou foi corrigida.

* **Status das atividades:**

Aberta: Tarefas ainda não iniciadas.

Em Andamento: Tarefas em desenvolvimento.

Em Teste: Tarefas desenvolvidas e sendo testadas.

Reaberto: Tarefas devolvidas para ajuste após testes.

Em Reteste: Tarefas em testes após correções.

Fechada: Tarefas desenvolvidas e com testes concluídos e aprovados.

**2.3.3 Principais funções dos recursos no Ágil**

No ágil existem alguns papeis chaves que costuma ter na maior parte dos projetos que trabalham com esta metodologia.

**Product Owner (PO):** Este é um dos papeis fundamentais dentre do ágil, porque representa o desejo do cliente, considerado o dono do produto e quem define e prioriza as funcionalidades para desenvolver dentro da *Sprint,* isto conforme o capacite do time e também, tem o importante papel de dar o De Acordo final, no que foi desenvolvido e pode aprovar ou recusar ou solicitar ajustes antes, durante e após o desenvolvimento do projeto.

**Analista Funcional**: Ele também, representa o desejo do cliente, deve atuar como um braço direito do (PO), auxiliar nas definições das funcionalidades, escrever o critério de aceite com as histórias que devem ser desenvolvidas, ajudar nos alinhamentos das informações entre os gerentes, áreas técnicas, desenvolvedores e analistas de qualidade.

**Scrum Master**: Ele deve atuar como um facilitador e responsável por remover os obstáculos que impactam o avanço das atividades e do projeto. Também deve proteger a equipe assegurando que ela não se comprometa excessivamente com relação àquilo que é capaz de realizar durante um Sprint.

**Arquiteto Funcional**: Temo papel de criar a estrutura do que será desenvolvido, analisar se a funcionalidade pode ser desenvolvida sem prejuízos a outras funcionalidades existentes ou que estão sendo desenvolvidas no mesmo momento por outras áreas.

**Desenvolvedores**: Como o próprio nome revela, eles são os responsáveis por desenvolver os códigos dos serviços e das telas conforme alinhado nas reuniões, documentado critério de aceite e desejo do (PO).

**Analista de Qualidade**: Também conhecido como *Tester*, tem o papel mapear e escrever os cenários de testes, responsável por realizar os testes funcionais, as regras de negócios, garantir que a funcionalidade desenvolvida está de acordo com a solicitada, registrar os erros encontrados no Jira, colher as evidências dos testes realizados, solicitar as correções erros e melhorias encontradas no projeto, responsável por dá o De Acordo no que foi desenvolvido corretamente e solicitar o De Acordo do (PO) para as funcionalidades que foram desenvolvidas conforme alinhado na reuniões ou padrões dos sistemas.

### Riscos da aplicação da Metodologia Ágil

Nos projetos que trabalham com ágil um grande esforço para gerenciar riscos é desnecessário, já que as iterações são curtas, a ênfase em qualidade e entregas frequentes, faz com que as equipes evitem a maioria dos riscos encontrados nos modelos tradicionais. Sendo assim, muitos projetos ágeis não possuem uma forma de gestão de riscos documentados e muitos projetos ágeis podem ser realizados sem uma gestão de riscos e outros com esta gestão mais formalizada, vão depender da necessidade de cada projeto. Os principais riscos do modelo Ágil são:

* A perda de bons profissionais no meio do processo, levando conhecimentos adquiridos ou até mesmo, atividades inacabadas.
* Contratações de funcionários que não tem posturas profissionais quando não tem gerentes lhe cobrando ou direcionando para a próxima atividade a ser realizada.
* Falta de documentações detalhadas de processos.
* O projeto ficar com um custo inviável e o valor do financiamento terminar antes do projeto ser concluído, já que, o custo total não estimado previamente assim como ocorrem nos métodos tradicionais.

**2.3.5 As principais Empresas que trabalham com ágil no Brasil**

**Atlântico:** É uma Instituição de Ciência e Tecnologia, há mais de 15 anos atua no desenvolvimento de softwares específicos para áreas de gestão e engenharia. Atua nas áreas de: Financeiro, setor elétrico, saúde, órgãos de fomento, corporativo, indústria e Telecom, é um dos principais ICTs do Nordeste em Pesquisa e possui excelência em pesquisa e desenvolvimento utilizando metodologias ágeis e tradicionais**.**

**BRQ:** É uma empresa que nasceu em 1993, na cidade de São Paulo, vem atuando a mais de 24 anos com soluções digitais próprias e customizadas, possui mais de 3000 profissionais distribuídos em oito unidades no país e uma nos Estados Unidos, é reconhecida pelo IDC como a maior empresa de serviços no gerenciamento e desenvolvimento de aplicações para o setor financeiro. Atua nas seguintes áreas: Financeiros, bancos, seguros, Telecom setor, grandes redes de supermercados.

**Inmetrics S/A:** É uma empresa de soluções de eficiência em TI, atua na qualidade dos desenvolvimentos de *software,* atende mais de 100 empresas dos mais variados segmentos de negócios, como: bancos, meios de pagamento, Telecom, seguradoras, varejo, indústria e energia. Está investindo cada vez mais nos métodos ágeis como uma alternativa às metodologias de projetos e desenvolvimento tradicionais e nos seus processos são utilizadas diversas ferramentas, tais como *Scrum*, *Safe@Oficce,* XP, DAD, *Lean IT*, *DevOps* e *Design Thinking.*

### RSI Informática: É uma empresa pioneira nos testes de qualidade de *software, desde* 1993, tem quase 1.000 colaboradores e seus principais clientes são: bancos, meios de pagamento, Telecom, seguradoras, varejo, indústria e energia. Desde 2016, vem inserindo a metodologia ágil para seus projetos e lançou uma melhoria no processo de validação dos testes automatizando, usando o BDT **(**Behaviour Driven Testing**)**,**** conceito foi criado em 2009 por *Mark Crowther*, onde os testes automatizados possuem foco nos testes funcionais, consideram cenários positivos e negativos e assim, usa a Análise de Risco para a tomada de decisão.

### Stefanini: Fundada em 1987, como uma empresa de treinamentos, em 1989 foi aberto o primeiro escritório e em 1990 início na área de desenvolvimento de sistemas. Possuem escritórios no Brasil, Argentina, Chile, México, Estados Unidos, Peru, Colômbia, Espanha, Portugal, Itália, Londres, Índia, Canadá e África do Sul. Consolida-se como uma empresa global, chega 30 países. A Stefanini possui experiência no gerenciamento Ágil de Projetos e tem um centro de excelência da metodologia *Scrum*, oferece gerenciamento ágil de projetos com abordagens do *Kanban*, conforme o perfil e a necessidade de cada cliente.

**2.4 VANTANGENS DO MODELO CASCATA**

De acordo com Ramos (2010), o modelo clássico mais conhecido como modelo cascata, foi proposto por Royce em 1970 e em até meados da década de 1980, foi modelo exclusivo de aceitação geral. A ideia principal seria a sequência de saída da primeira etapa para a segunda etapa e a saída da segunda etapa para a terceira e assim por diante. As atividades são agrupadas em tarefas, executadas sequencialmente, de forma que uma tarefa só poderá ter início quando a anterior tiver terminado.

No método cascata o processo de desenvolvimento estruturado, existe maior enfoque na etapa de planejamento, com o objetivo de eliminar os riscos e a entrega do produto sempre é entregue em sua totalidade sem variações de custo.



Figura 6 – O modelo em cascata ou waterfall.

Fonte: Royce, 1970.

**2.4.1 Documentações usadas no modelo cascata**

**Análise e definição dos requisitos:** Deve conter os requisitos do produto que será desenvolvido como, limitações e objetivos do software, os requisitos devem ser definidos de uma maneira apropriada para que sejam úteis na etapa seguinte. Esta etapa deve ter a documentação e o estudo da facilidade e viabilidade do projeto, pode ser vista como uma concepção de um produto de software e também como o início do seu ciclo de vida.

**Implementação:** Esta é a etapa em que são criados os programas. Se o projeto possui um nível de detalhe elevada. Inicialmente aconselha-se ser realizado teste unitário, que são, unidades de códigos produzidos e devem ser testadas individualmente antes de passar a etapa de integração e teste global. Quando iniciam os testes de sistema, são analisadas as soluções dos erros e assegura que as entradas definidas produzam resultados reais que coincidam com os requisitos especificados.

**Mapa de cobertura:** Para ser criado um mapa de cobertura, inicialmente são realizadas reuniões de entendimentos com todas as áreas envolvidas, onde são tiradas todas as dúvida e após esta reunião, o analista de testes deve mapear todas as funcionalidades de forma detalhada, prevendo os riscos, fatores externos e internos que possam causar atrasos nas entregas, nesta cobertura todas as variações de casos de testes devem ser documentadas.

Após o mapeamento detalhado, é realizada reunião para apresentar os documentos gerados, casos haja correções solicitadas, todo o processo deve ser repetido e se não houver correções, o mapa será considerado aprovado então, é serializada a WBS, a qual contém todos os custos previstos para o projeto, após a aprovação da WBS devem-se iniciar as escritas dos passos a passos, scripts, de cada item previamente mapeado e aprovado.

* + 1. **Principais riscos no modelo cascata**

O ciclo de vida no modelo Cascata é o paradigma mais visto e mais empregado nos projetos de desenvolvimentos, mas, sua aplicabilidade vem sendo questionada e na verdade, os projetos raramente seguem o fluxo sequencial que este modelo propõe, normalmente sua metodologia si misturam a outros métodos gerenciamento moderno.

Em princípio, é difícil para o cliente especificar os requisitos e isto acarreta a incerteza natural do início de qualquer projeto. Mediante estes fatores, o cliente deve ser paciente, pois uma versão funcional não estará disponível até o final do desenvolvimento e qualquer erro ou mal-entendido, se não for detectado até que o software seja revisado, pode ser desastroso, pois quando ocorre um atraso todo o processo é afetado.

1. **METODOLOGIA**

Este trabalho foi realizado através de pesquisas de livros, *blogs, sites* de profissionais renomados nas áreas de gerenciamentos de projetos de TI, também, foram realizadas pesquisas das grandes empresas que trabalham com desenvolvimento e a qualidade de Softwares, analisando os métodos gerenciais mais usados, as vantagens e desvantagens dos métodos antigos e mais modernos e se tem alguma forma de gerenciamento considerada eficaz e que possa ser aplicadas em todos os projetos, desconsiderando assim, as demais metodologias aplicadas e a realização dos estudos aplicados nesta pesquisa segue a mesma metodologia de trabalhos científicos, com consulta de sites especializados, blogs e estudos das empresas que trabalham com este método. O conhecimento científico permite que existam formas de entendimentos e interpretações diferentes para cada pessoa, pois, não se trata de uma obra pronta ou acabada.

Como abordagem da pesquisa foi utilizada método de natureza qualitativa a qual, teve como características o caráter de exploratório, realidades múltiplas e o autor conhece o espaço da pesquisa e convive com os objetivos pesquisados e como contexto, foi realizada análise de materiais didáticos, revisões sistemáticas de artigos, através de pesquisas em sites especializados, blogs de autores que escrevem e atuam na área de desenvolvimento gerenciamento de *softwares*. Também foram realizadas pesquisas de várias empresas que atuam no ramo e seguem as metodologias abordadas pela pesquisa, assim como, conversas informais com profissionais de TI.

**4 ANÁLISE DOS RESULTADOS**

Foi possível observar um consenso em que as metodologias Ágeis focam em acelerar o desenvolvimento de software e numa interação maior entre as equipes. Nesta pesquisa o autor buscou conhecer as vantagens, desvantagens, riscos e mercados das metodologias tradicionais (Cascatas) e das metodologias Scrum Ágeis, analisou os seus conceitos as aplicações dentro das empresas para adquirir a reposta ao problema e a hipótese levantada inicialmente.

**4.1 ANÁLISE SWOT DA METODOLOGIA ÁGIL**

|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **Strengths** | **F** | **Força** |
| * Mais independência e produtividade para a equipe * Maior liberdade no planejamento do projeto e em cada etapa de trabalho. * Melhor definição do objetivo * Projetos são discutidos e flexibilizados em conjunto. * Equipe trabalha mais unida e a divisão do trabalho é realizada de acordo com as habilidades de cada membro do time. * Existe uma participação mais ativa do cliente em todas as etapas do projeto, através de feedbacks. * Melhor atendimento ao cliente. * Investimento em profissionais qualificados. * Mais foco na automação dos testes. | | | |
| **W** | **Weaknesses** | **F** | **Fraquezas** |
| * O produto é entregue por partes, o que pode não ser vantajoso para o cliente que precisa de um projeto 100% pronto. * Planejamento extenso, que exige várias análises em cada etapa do projeto. * Os projetos necessitam de profissionais com mais experiências no trabalho em equipe, em processos, reponsabilidades com suas tarefas. * As equipes ágeis são alocadas em projetos exclusivos e cada profissional é um custo fixo do seu respectivo projeto. * Pode ter um custo mais alto do que um projeto realizado com metodologias tradicionais. | | | |
| **O** | **Oportunities** | **O** | **Oportunidades** |
| * Avanço tecnológico rápido. * Necessidade desenvolver projetos antes da concorrência. * Grande aceitação dos clientes. | | | |
| **T** | **Threats** | **A** | **Ameaças** |
| * Crise econômica no Brasil. * A perda de bons profissionais no meio do processo, levando conhecimentos adquiridos ou até mesmo, atividades inacabadas. | | | |

**4.2 ANÁLISE SWOT DA METODOLOGIA CASCATA**

|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **Strengths** | **F** | **Força** |
| * Maior percepção por parte do cliente referente ao valor total do projeto. * Todas as atividades identificadas nas fases do modelo são fundamentais e estão na ordem certa. * Entrega do produto em sua totalidade. * Maior enfoque na etapa de planejamento, com o objetivo de eliminar os riscos. * Projetos são realizados de forma mais rápida. * Torna o processo de desenvolvimento estruturado. * Tem uma ordem sequencial de fases. * Cada fase cai em cascata na próxima e cada fase deve estar terminada antes do início da seguinte. | | | |
| **W** | **Weaknesses** | **F** | **Fraquezas** |
| * Não é interessante para clientes que querem sempre inovar e adicionar melhorias nos seus produtos. * Não fornece feedback entre as fases e não permite a atualização ou redefinição das fases anteriores. * Não suporta modificações nos requisitos. * Não prevê a manutenção. * Não permite a reutilização. * É excessivamente sincronizado. * Se ocorrer um atraso todo o processo é afetado; * Faz aparecer o software muito tarde. * Todo o processo é centralizado no gerente de projetos. | | | |
| **O** | **Oportunities** | **O** | **Oportunidades** |
| * Crise econômica no Brasil, o qual os custos dos projetos são uns dos principais critérios para investimentos em novos projetos ou melhorias de seus sistemas. * Empresas que preferem investir em projetos mais definidos e com resultados mais seguros. * Baixo risco na entrega do produto final ou projeto. | | | |
| **T** | **Threats** | **A** | **Ameaças** |
| * Avanço tecnológico rápido * Necessidade desenvolver projetos antes da concorrência. * Grande aceitação dos clientes as novas metodologias Ágil. * Não prevê a manutenção dos projetos após sua implantação | | | |

**5 CONCLUSÃO**

Com as pesquisas realizadas sobre as metrologias tradicionais, mais conhecidas como cascatas, e as metodologias ágeis, o autor pode entender os conceitos, as vantagens e desvantagens de cada modelo. As duas metodologias têm suas características que podem ser boas ou ruins dependendo a necessidade de cada projeto. Com isto as empresas que trabalham com desenvolvimentos de *softwares* e com homologação ou testes de qualidade de sistemas, não trabalham com uma metodologia ou outras, normalmente, estas empresas trabalham com vários métodos de gerenciamentos, inclusive a metodologia ágil e o modelo cascata.

O modelo cascata dominou por décadas os gerenciamentos de desenvolvimentos de TI, suas características tradicionais, metódicas, com análise detalhadas dos riscos e custos totais mais baixos, são ideais para empresas e gestores que possuem certo temor as mudanças. Já o modelo ágil, com método de gerenciamento flexível e inovador são ideais para as empresas competitivas que precisam estar sempre inovando seus sistemas e produtos.

Então, se a empresa possui um cliente que quer o projeto entregue 100% concluído, possui um orçamento e escopo fechado, ela precisa utilizar a metodologia tradicional, pois, não faz sentido utilizar metodologias ágeis que são entregues por partes e que precisam de uma interação do cliente e a equipe do projeto e com isso comprometeria o cronograma e o escopo, além do aumento de custos. Mas, se a empresa precisa ter sistemas em constantes melhorias, possui um escopo flexível e quer ter softwares competitivos e inovadores, esta empresa precisa trabalhar com a metodologia ágil, que se encaixa melhor nas necessidades de seus projetos.

Não existe modelo de gestão certo ou errado, o que existe são novas formas necessidades de gerenciamentos e/ ou de desenvolvimento, os quais surgem ao longo dos anos, que agregam e possibilitam mais opções para clientes e empresas prestadoras de serviços. O que deve ser analisado não são as vantagens dos métodos e sim qual é o modelo de gerenciamento que o projeto necessita, bom senso acima de tudo para que não haja insatisfação dos clientes e nem desperdícios de recursos.

**6 BIBLIOGRAFIA**

ATLASSIAN. **Confluence é um software de colaboração de conteúdo que revoluciona a forma de trabalho das equipes.** Disponível em: <<https://br.atlassian.com/software/confluence>>. Acesso em: 15 out. 2017.

ATLASSIAN. **Colaboração: o seu caminho.** Acesso em: <<https://br.atlassian.com/software/confluence/features>>. Acesso em: 15 out. 2017.

BLOG ARCHIVE**. Definição: Modelo Cascata ou Clássico**. Disponível em: <http://modelocascata.blogspot.com.br/2010/>. Acesso em: 06 nov. 2017.

BLOG DIGITAL TECHNOLOGIES. **Definição: Metodologias ágeis x metodologias tradicionais**. Disponível em: <http://blog.dtidigital.com.br/metodologias-ageis-x-metodologias-tradicionais/>Acesso em: 08 nov. 2017.

BRQ. **Metodologias ágeis de desenvolvimentos de software.** Disponível em: <<http://www.brq.com/metodologias-ageis/>>. Acesso em: 25 set. 2017.

CUNNINGHAM, Ward. **Manifesto para o desenvolvimento ágil de software.** Disponível em: <<http://www.manifestoagil.com.br/>>. Acesso em: 20 set. 2017.

DENIS, Pedro. **Scrum: Dobro do Trabalho na Metade do Tempo.** Disponível em: <<http://www.mindmaster.com.br/como-implantar-scrum/>>. Acesso em: 20 set. 2017.

GITHUB. **Features**. Disponível em: <https://github.com/features>. Acesso em: 28 out. 2017.

[INMETRICS](https://www.implementingscrum.com/).**Fábrica de soluções e evolução do modelo.** Disponível em: <<http://www.inmetrics.com.br/fabrica-de-solucoes/>>. Acesso em: 26 out. 2017.

# [IMPLEMENTING SCRUM](https://www.implementingscrum.com/).Scrum: Implementing Scrum in the Real World. Disponível em: <<http://www.implementingscrum.com/>>. Acesso em: 22 set. 2017.

# [LEAN](https://www.implementingscrum.com/) INSTITUTE BRASIL Definição: Lean é uma filosofia de gestão inspirada em práticas e resultados do Sistema Toyota. Disponível em: <<https://www.lean.org.br/o-que-e-lean.aspx>>. Acesso em: 01 out. 2017.

NOGUEIRA, Elias. **O que é Agile Testing.** Disponível em: <<https://imasters.com.br/desenvolvimento/o-que-e-agile-testing/?trace=1519021197&source=single>>. Acesso em: 25 set. 2017.

PRIKLADNICKI, Rafael. **Métodos Ágeis para desenvolvimento de software.** Porto Alegre: BOOKMAN, 2014.

# RENZI, Wagner. A importância da documentação funcional na Metodologia Ágil. Disponível em:<http://blog.rsinet.com.br/index.php/novidades/openbdt-framework-open-source-baseado-no-conceito-de-bdt/>. Acesso em: 31 out. 2017.

RSI. Framework open source baseado no conceito de BDT. Disponível em: <<https://www.tiespecialistas.com.br/2014/03/importancia-da-documentacao-funcional-na-metodologia-agil/>>. Acesso em: 28 out. 2017.

# SABBAGH, Rafael. Scrum: Gestão ágil para projetos de sucesso. Rio de janeiro: Casa do Código, 2013.

# SANTOS, Cleber. Rush - Metodologia Ágil de gerenciamento de projetos priorizando prazo. Disponível em: <http://pmpath.com.br/rush-metodo-agil-de-gerenciamento-de-projetos-priorizando-prazo>. Acesso em: 01 out. 2017.

SCRUM ALLIANCE. **Agile Atlas.** Disponível em: <<https://www.scrumalliance.org/why-scrum/agile-atlas>>. Acesso em: 10 out. 2017.

SUTHERLAND, Jeff. **Scrum: A arte de fazer o dobro do trabalho na metade do tempo**. Estados Unidos: Leya, 2014.

STEFANINI. **Scrum e suas vantagens.** Disponível em: <<https://stefanini.com/br/2014/01/scrum-vantagens/>>. Acesso em: 24 set. 2017.

VIEIRA, Denisson. **Scrum: A Metodologia Ágil Explicada de forma Definitiva.** Disponível em: <<http://www.mindmaster.com.br/scrum/>>. Acesso em: 20 set. 2017.

XAVIER, Eneida. **Agile x Scrum x Lean, o que cada um é?** Disponível em: <<http://www.youwilldobetter.com/2017/04/agile-x-scrum-x-lean-o-que-cada-um-e/>>. Acesso em: 01 out. 2017.

**7 GLOSSÁRIO DA ÁREA DE INFORMÁTICA**

Os projetos realizados com a metodologia ágil têm um vasto vocabulário em inglês, os quais não são traduzidos nos projetos e todos acabam se familiarizando com as palavras mais usuais. Exemplos:

***Scrum*:** É uma [metodologia ágil](http://www.desenvolvimentoagil.com.br/xp/manifesto_agil) para gestão e planejamento de projetos.

***Daily:*** São reuniões rápidas que ocorrem diariamente, onde cada um da equipe fala o que fez, o que pretende fazer e se tem algum impedimento.

***Product Backlog:*** É uma lista de funcionalidades com os requisitos que o cliente espera receber ao final do projeto, descrito com sua própria linguagem.

***Product Owner (PO):*** É o dono do produto, a pessoa que define as funcionalidades prioritárias para desenvolver na *sprint*.

***Release Burndown:*** Ele mostra, através de relatórios gerados, o que foi feito e o que resta fazer, isso pode ocorrer de acordo a necessidade do projeto, diariamente, semanalmente ou nos finais das *sprints*.

**Scrum Master:** Atua como um facilitador e torna-se responsável por remover quaisquer obstáculos que sejam levantados pela equipe, também, o importante papel de proteger a equipe assegurando que ela não se comprometa excessivamente com relação àquilo que é capaz de realizar durante um Sprint.

***Scrum Team:*** É a equipe de desenvolvimento, em média possui 6 a 10 pessoas, más, também existem equipes bem maiores.

***Sprint Backlog:*** É uma lista de tarefas que a equipe se compromete a fazer em um *sprint.*

***Plannig:*** É uma reunião que ocorre no início dos *sprints,* com todo o time e onde o [*Product Owner*](http://www.desenvolvimentoagil.com.br/scrum/product_owner) descreve as funcionalidades de maior prioridade para a equipe e desta reunião saem as tarefas que serão realizadas no [*Sprint Backlog*](http://www.desenvolvimentoagil.com.br/scrum/sprint_backlog)*.*

***Retrospective:*** É uma reunião com toda equipe, que ocorre no fim da Sprint, onde são avaliados pelo time o que deu certo, o que pode ser melhorado e que ações serão tomadas para melhorar.

***Review:*** É uma reunião com todo o time, ela ocorre no final do *sprint* e sua função é mostrar as novas funcionalidades ou melhorias desenvolvidas.

***Waterfall:*** Significa cascata e é considerada a forma mais tradicional de gerenciar projetos, pois, os requisitos são totalmente definidos no início do projeto e geralmente sofrem pouca ou nenhuma alteração durante sua execução.

***Ruby for Windows:*** linguagem de programação utilizada nos testes.

***cmder for Windows:*** Sistema que traz as funcionalidades *bash*(Terminal) para o *Windows.*

***Bash***: É um interpretador de comandos.

***DevKit:*** *Kit* de ferramentas que o sistema operacional precisa para que o desenvolvimento funcione.

***Kit:*** É uma série de ferramentas digitais para a criação de aplicativos

***Chromedriver:*** Driver do navegador que será utilizado na automação

***SublimeText:*** Editor de texto com funções úteis para escrever o código da automação de testes.