PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

RENATA SEGARRO DE PAULA

COMPARAÇÃO DE METODOLOGIAS ÁGEIS

São Paulo, 2013

RENATA SEGARRO DE PAULA

COMPARAÇÃO DE METODOLOGIAS ÁGEIS

Monografia apresentada ao Curso de Especialização em Engenharia de Software da Pontifícia Universidade Católica de São Paulo, como requisito parcial para obtenção do título de Especialista em Engenharia de Software, orientado pelo Professor Mestre José Paulo Levandovski Papo.

São Paulo, 2013

COMPARAÇÃO DE METODOLOGIAS ÁGEIS

RENATA SEGARRO DE PAULA

Monografia defendida e aprovada, em 30 de dezembro de 2013, pela banca examinadora:

Professor Mestre José Paulo Levandovski Papo

Orientador

DEDICATÓRIA

Dedico este trabalho a todos aqueles que contribuíram de forma direta ou indireta para conclusão do mesmo.

AGRADECIMENTOS

Agradeço ao meu orientador Professor Mestre José Paulo Levandovski Papo por todo o apoio e orientação fornecidos durante a elaboração desse trabalho e de todas as suas aulas.

Agradeço minha família por toda a paciência e tolerância comigo, especialmente no desenvolvimento desse trabalho e também por todo o apoio e orientação no meu próprio desenvolvimento.

Agradeço a minha grande amiga Luciana dos Anjos Chaves por todas as dicas e apoio fornecidos.

"O analfabeto do século XXI não será aquele que não sabe ler e escrever, mas aquele que não consegue aprender, desaprender e aprender novamente"

Alvin Tofler

RESUMO

As metodologias de desenvolvimento de software são adotadas para que haja maior controle nessa tarefa que já é complexa por natureza. O problema que pode acontecer é que, dependendo do projeto, os métodos tradicionais podem deixar os desenvolvedores amarrados a requisitos desatualizados e que já não irão mais atender às reais necessidades dos clientes já que esses métodos são divididos em fases com processos bem definidos e são inflexíveis quanto a mudanças. Com o passar do tempo, os softwares estão adquirindo cada vez mais complexidade e necessitam serem desenvolvidos de forma mais flexíveis. As metodologias ágeis tem o objetivo de agilizar o processo de desenvolvimento, principalmente no que diz respeito à utilização do software pelo cliente. Cada vez mais são procuradas alternativas para o processo de desenvolvimento de sistemas de uma maneira rápida, eficiente e que atenda as reais necessidades dos clientes, mitigando as principais causas de fracasso dos projetos. Em mercados altamente competitivos, ou em momentos de crise econômica, a flexibilidade e a facilidade de mudar o rumo são qualidades muito valiosas para serem deixadas de lado. Este trabalho irá apresentar e detalhar três metodologias/framework para desenvolvimento ágil de softwares. Além disso, será feita uma comparação entre essas metodologias através de uma pesquisa descritiva e comparativa, na forma de estudos descritivos com base na pesquisa bibliográfica, permitindo ao leitor conhecer uma variedade de técnicas de modelagens e os formatos mais adequados para cada tipo de projeto.

Palavras-chave: Metodologia Ágil, Startup Enxuta, Scrum, XP

ABSTRACT

The software development methodologies are adopted for greater control in this task which is already complex in nature. Depending on the project, the problem that can happen is that the traditional methods can let the developers tied to outdated requirements and will no longer meet the actual needs of the customers as these methods are divided into phases with well-defined processes and are inflexible for changes. Over time, the software is getting increasingly complex and need to be developed more flexible manner. Agile methodologies aims to streamline the development process, especially with regard to the use of the software by the customer. Alternatives to the process of development of a fast, efficient and meets the actual needs of customers way systems, mitigating the main causes of failure of projects are increasingly sought after. In highly competitive markets, or in times of economic crisis, flexibility and ease of changing the course are very valuable qualities to be left aside. This paper will introduce and detail three methodologies / framework for a spry software development. Furthermore, a comparison of these methodologies will be taken through a descriptive and comparative research in the form of descriptive studies based on literature search, allowing the reader to know a variety of modeling techniques and the most suitable for each type of project formats.

Keywords: Agile Methodology, Lean Startup, Scrum, XP

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Este capítulo apresenta a motivação e a relevância do problema, os objetivos do trabalho, os resultados e contribuições, a metodologia adotada e a organização do trabalho.

1.1 Motivação

De acordo com a pesquisa realizada pelo *Standish Group, Chaos Report* (2001), apenas 28% dos projetos de TI são bem sucedidos. Periodicamente é realizada também no Brasil uma pesquisa semelhante, popularmente conhecida como “Pesquisa Archibald & Prado”, que ocorreram nos anos de 2005, 2006, 2008 e 2010, onde organizações de diversos tamanhos e natureza respondem um questionário a respeito de seus projetos realizados.

Nesta pesquisa, o resultado em 2010 foi mais otimista, apesar de ser ainda muito baixo, apresentando 56,7% de sucesso nos projetos, porém considerando que esta foi realizada com apenas 47 empresas, onde o conceito de projeto bem sucedido (ou sucesso completo ou apenas sucesso) utilizado na pesquisa foi: “o projeto terminou praticamente no prazo, escopo e orçamento previstos (diferenças insignificantes). O usuário ficou totalmente satisfeito, pois o produto/serviço que lhe foi entregue está sendo utilizado e realmente agregou valor ao seu trabalho” (PRADO, 2011).

Os resultados mostraram que as principais causas de fracasso são as seguintes, conforme podemos observar também pela Figura 1 (PRADO, 2011):

- Frequentes mudanças de escopo: 76%
- Falta de comprometimento das áreas: 38%
- Precariedade de metodologia de GP: 38%
- Alterações de Prioridade: 34%

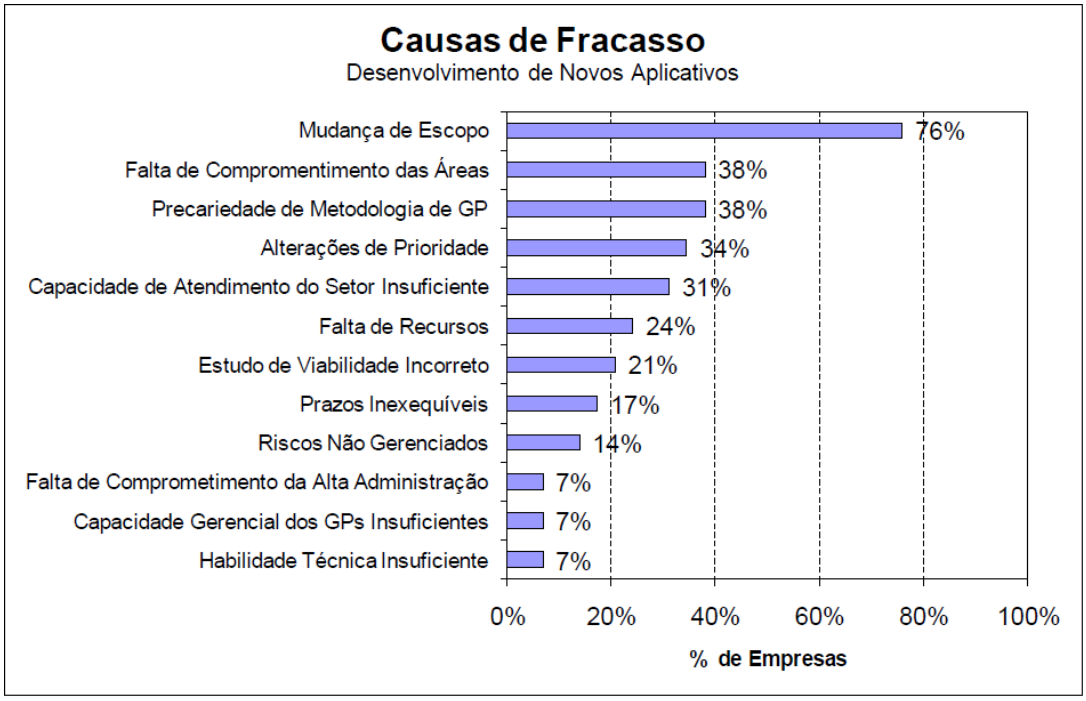

Figura 1. Causas de Fracasso dos Projetos

(PRADO, 2011)

Desta forma, cada vez mais são procuradas alternativas para o processo de desenvolvimento de sistemas de uma maneira rápida, eficiente e que atenda as reais necessidades dos clientes, mitigando as principais causas de fracasso dos projetos.

Por muito tempo o desenvolvimento de software seguiu uma metodologia tradicional e, mesmo hoje, ainda é muito utilizada, porém, segundo o Manifesto Ágil, escrito em fevereiro de 2001 por 17 praticantes de diversas metodologias diferentes, é preciso descobrir maneiras melhores de se desenvolver software, desenvolvendo e ajudando outras pessoas a desenvolver. Os softwares estão adquirindo cada vez mais complexidade e as metodologias ágeis tem o objetivo de agilizar o processo de desenvolvimento, principalmente no que diz respeito à utilização do software pelo cliente.

Conforme apresentado na figura 2, os resultados de projetos que utilizam metodologias clássicas, como Cascata, obtiveram um número superior de projetos fracassados e comprometidos (que tiveram significativas mudanças de prazo, tempo e orçamento e/ou a satisfação do usuário é parcial) comparados aos projetos que utilizaram metodologia ágil, o qual obteve 43% de projetos bem sucedidos, contra 26% dos projetos realizados pelo método Cascata.

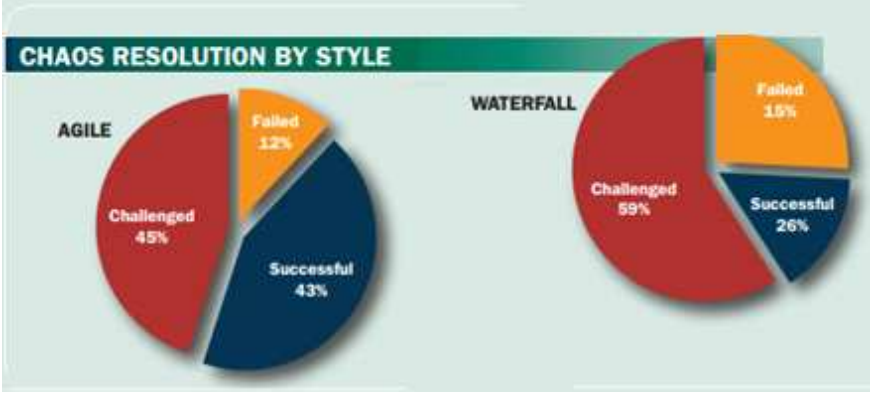

Figura 2. Chaos Resolution By Style

Fonte: Standish Group, Chaos Summary for 2010 (2010)

Desta forma, a metodologia ágil de desenvolvimento de software surgiu em resposta aos processos pesados e burocráticos de desenvolvimento de software, tendo como propósito central, de acordo com o Manifesto Ágil (2001):

- Indivíduos e interações ao invés de processos e ferramentas
- Software operante ao invés de documentações completas
- Colaboração do cliente ao invés de negociações contratuais
- Responder à mudanças ao invés de seguir um planejamento.

Segundo o mesmo manifesto (2001), os seguintes princípios são seguidos com metodologia ágil:

1. A prioridade mais alta é satisfazer o cliente através de entregas contínuas e antecipadas de software válido.
2. Mudanças nos requisitos são bem-vindas, mesmo as que chegam tarde no desenvolvimento. Processos ágeis asseguram a mudança como uma vantagem competitiva do cliente.
3. Entregar software produtivo frequentemente, de algumas semanas a alguns meses, de preferência os tempos mais curtos.
4. Pessoal de negócio e desenvolvedores trabalham juntos diariamente durante o projeto.
5. Criar projetos em torno de indivíduos motivados, proporcionar o ambiente e suporte que eles necessitam e confiar que eles farão o serviço.
6. O método mais eficiente e efetivo para transmitir informações entre e para a equipe de desenvolvimento é conversão cara a cara.

7. Software produtivo é a medida primária do progresso.
8. Processos ágeis promovem um desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Atenção contínua à excelência técnica e boa solução melhoram a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não feita – é essencial.
11. As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva e então sintoniza e ajusta seu comportamento de forma apropriada.

Se observarmos, os princípios apresentados demonstram soluções para as principais causas de fracasso em projetos, exibidos na figura 1, conforme apresentado na tabela 1:

Tabela 1. Causas do Fracasso X Princípios da Metodologia Ágil

Causas de Fracasso	**Resposta da Metodologia Ágil (Princípio)**
Mudança de Escopo (76%)	Princípio 2
Falta de Comprometimento das Áreas (38%)	Principio 4 e 5
Precariedade de Metodologia de GP (38%)	Conjunto de ações e princípios que tornam a metodologia de GP melhor e mais fácil, destacando os princípios 3, 6, 7, 8 10, 11 e 12.
Alterações de Prioridade (34%)	Princípios 1 e 2

Resumidamente, os métodos ágeis visam minimizar os riscos através desenvolvimento com entregas em curtos períodos, criando uma integração contínua com o cliente prezando a satisfação, garantindo a validação contínua do produto, na qual a equipe poderá trabalhar em melhorias internas com maior frequência para a garantia da qualidade durante todo o projeto e não somente no final, quando o software já foi entregue.

Diferente dos métodos tradicionais, a mudança dos requisitos são bem-vindas e são vistas como uma vantagem competitiva do cliente. A comunicação entre todas as partes envolvidas é permanente e presa-se pela comunicação cara a cara, por ser considerada mais eficiente.

Evidências apresentadas através de pesquisas demonstram como o método mostrar-se eficaz também na prática e não somente na teoria. Dados mais atuais, como os do Agile Survey (2013), exibem resultados bastante animadores sobre a metodologia ágil, o qual diz que ¾ dos participantes responderam que seus projetos ágeis foram concluídos com sucesso e ¼ tiveram sucesso em 100% de seus projetos ágeis.

Contudo, é preciso que fique claro, que como qualquer transformação dentro de uma organização, é necessário comprometimento, apoio e mudanças para adaptação a metodologia ágil. Para utilizar e aplicar seus conceitos existem diversos métodos e frameworks disponíveis no mercado, entre eles, SCRUM, Lean Startup e XP, que serão apresentadas neste trabalho.

1.2 Objetivos

O objetivo deste trabalho é apresentar e realizar comparações entre diferentes metodologias e frameworks ágeis para desenvolvimento de software, exibindo ao leitor uma variedade de opções e formatos para cada tipo de organização.

A metodologia é a ciência que estuda os métodos de desenvolvimento de software. A agilidade no método de desenvolvimento é uma necessidade atualmente, pois os projetos hoje estão sujeitos à mudanças de última hora (LOBO, 2008).

1.3 Resultados Esperados

Este trabalho resultará em uma análise comparativa de diferentes metodologias e frameworks ágeis de desenvolvimento de projetos com o objetivo de possibilitar ao leitor mais facilidades na escolha da metodologia mais adequada para cada tipo de organização.

Demonstrando as características de cada método ágil (Scrum, Lean Startup e XP) para que o leitor do trabalho possa identificar qual o melhor método a ser utilizado em cada desenvolvimento.

Entre os benefícios apresentados com a utilização do desenvolvimento ágil de software, pode-se destacar o fato da prioridade mais alta ser a satisfação do cliente, com sua constante colaboração e a ocorrência das mudanças de requisitos serem bem-vindas, se opondo a regra geral presente nas metodologias clássicas.

1.4 Metodologia

O intuito deste trabalho é levantar, analisar, apresentar e comparar metodologias e frameworks de desenvolvimento ágil.

Será utilizado primeiramente, como procedimento técnico, a pesquisa e a revisão bibliográfica, realizando análises em livros, artigos, trabalhos acadêmicos, entre outros, com o objetivo de revisar a literatura existente, adquirindo conhecimento inicialmente básicos e delimitados, para posterior enriquecimento e documentação do trabalho através de contribuições secundárias.

A pesquisa bibliográfica é a mais simples técnica de pesquisa. Ela explica um problema fundamentando-se apenas nas contribuições secundárias, ou seja, nas informações e dados extraídos de leitura corrente e de referências, de revistas impressas e virtuais, material audiovisual, entrevistas, documentos etc. de diferentes autores que versam sobre o tema selecionado para o estudo (REIS, 2010).

Para a descrição do trabalho, quanto aos objetivos, será utilizada a pesquisa descritiva, pois a mesma é utilizada para descrever fenômenos existentes, situações presentes e eventos, identificar problemas e justificar condições, e principalmente, porque a técnica permite comparar e avaliar o que os outros estão desenvolvendo em situações e problemas similares, visando aclarar situações para futuros planos e decisões (GRESSLER, 2004).

Abaixo é apresentado um resumo com a visão geral das pesquisas e estratégias que serão utilizadas para desenvolvimento do trabalho:

- Fundamentos iniciais e básicos referentes a metodologias diversas com foco posterior na metodologia ágil.
- Pesquisa sobre os métodos ágeis Scrum, Lean Startup e XP.
- Comparação entre as técnicas apresentados, proporcionando ao leitor o confronto dos prós e contras das técnicas.

1.5 Organização do trabalho

O trabalho está organizado da seguinte forma:

O capítulo 1, Introdução, é a motivação para este trabalho, bem como seus objetivos, resultados esperados, a metodologia e sua organização.

O capítulo 2, Fundamentos e Estudo da Arte, apresenta os conceitos necessários para o leitor compreender o trabalho.

O capítulo 3, Scrum, apresenta e explica este framework.

O capítulo 4, Startup Enxuta, apresenta e explica essa metodologia.

O capítulo 5, Programação Extrema, apresenta e explica essa metodologia.

O capítulo 6, Comparação entre os métodos e framework, apresenta uma análise comparativa dos itens relatados nos capítulos 3, 4 e 5.

O capítulo 7, Considerações Finais, apresenta a conclusão do trabalho e possíveis extensões futuras.

O capítulo 8, Referências Bibliográficas, apresenta a lista de referências bibliográficas utilizadas para o desenvolvimento desse trabalho.

2 Fundamentos e Estudo da Arte

Este capítulo apresenta alguns conceitos como base técnica para o desenvolvimento do trabalho e que são importantes para que o leitor possa entendê-lo. O item 2.1 apresenta os conceitos que serão abordados. O item 2.2 apresenta o conceito de processos, metodologia, métodos de desenvolvimento de software e framework. O item 2.3 apresenta o conceito de métodos clássicos. O item 2.4 apresenta o conceito de metodologia ágil.

2.1 Introdução

Os conceitos que serão abordados são: processos, metodologia, métodos de desenvolvimento de software, framework, métodos clássicos e metodologias ágeis.

2.2 Processos, Metodologia, Métodos de desenvolvimento de software e Framework

Os termos descritos neste item são conhecidos, mas ocasionam grande confusão até mesmo entre os mais experientes na área de tecnologia de informação. Desta forma, como muitos desses termos serão citados no trabalho, é de extrema importância, explica-los e diferenciá-los para melhor entendimento.

O termo processo, isoladamente, significa um “conjunto de atos por que se realiza uma operação qualquer” (DICIONÁRIO ONLINE, 2009). Voltado para o ambiente de TI, processos de software, são atividades em fases ordenadas, que tem a finalidade de definir e organizar o ambiente de desenvolvimento de software, com foco no processo. São normalmente pesados ou customizáveis (LOBO, 2008).

Processo de software ainda é definido como uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade (PRESSMAN, 2011)

Assim, um processo de software define a abordagem adotada focada nos processos, sem definir as tecnologias que fazem parte deste processo, como procedimentos técnicos e ferramentas.

Metodologia é definida como uma ciência que estuda os métodos aos quais ela se liga ou de que se utiliza (DICIONARIO ONLINE, 2009). Sendo assim, a metodologia de desenvolvimento de software é a ciência que estuda os métodos de desenvolvimento de software.

De acordo com o dicionário online de português (2009), método é maneira de dizer, de fazer, de ensinar uma coisa, segundo certos princípios e em determinada ordem. Maneira de agir. Obra que reúne de maneira lógica os elementos de uma ciência, de uma arte etc.

Voltado para o ambiente de software, assim como os processos, os métodos também são definidos como são atividades em fases ordenadas, que tem a finalidade de definir e organizar o ambiente de desenvolvimento de software, focalizado no processo, tendo praticamente a mesma definição, porém sempre enfatizando que o processo é considerado pesado.

Framework é um conjunto de classes cooperativas que implementam os mecanismos que são essenciais para um domínio de problemas específicos (HORSTMANN, 2006). A qualidade distintiva de um framework é que ele fornece uma implementação para as funções básicas e invariantes e inclui um mecanismo para permitir que o desenvolvedor se conecte as diversas funções ou as reutilize (LARMAN, 2005).

Pensando em framework na definição de processos de software, o livro UML Guia do Usúário (BOOCH; RUMBAUGH; JACOBSON, 2006), traz um significado mais especifico para framework o definindo como um padrão de arquitetura que fornece um *template* extensível para aplicações dentro de um domínio. O framework é maior do que um mecanismo.

2.3 Métodos Clássicos

Os métodos clássicos ou tradicionais são muito conhecidos, pois são aplicados já há muito tempo. Eles têm como principal característica serem divididos em fases com processos bem definidos e por serem chamados de “pesados”.

Para melhor entendimento desses métodos, serão apresentados os conceitos básicos dos principais métodos tradicionais:

- Cascata: utilizando este método, é necessário obedecer uma sequência obrigatória para o desenvolvimento de software. Este é um dos

processos mais inflexíveis, pois não permite o retorno para fases já implementadas. Por exemplo, estou na fase de desenvolvimento do software e foi identificada uma nova necessidade para o cliente, ou seja, um novo requisito, o modelo não permite retornar para a fase de requisitos. Sendo assim, ele só pode ser utilizado em projetos onde os requisitos estão bem definidos e são estáveis.

As suas fases clássicas são (REZENDE, 2005): definição e análise dos requisitos, projeto do software, implementação e testes, implantação e integração do produto. Em cada uma das fases são estabelecidas suas respectivas atividades as quais geram produtos de saída pré-definidos.

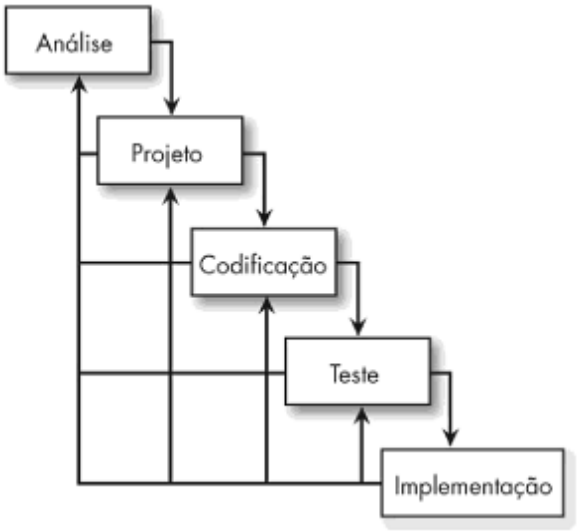

Figura 3. Modelo ciclo de vida Cascata Clássico

(AUDY; PRIKLADNICKI, 2007)

- Incremental: como o próprio nome diz, neste método, o software é construído por incrementos. Invés de se construir todo o software de uma única vez como no modelo cascata, o projeto é desenvolvido, testado e entregue em partes, ocorrendo suas fases diversas vezes durante o projeto.

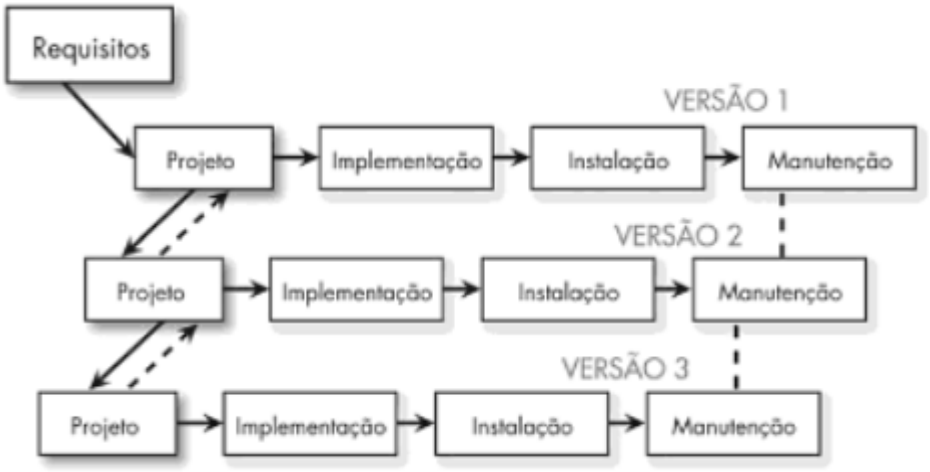

Figura 4. Modelo ciclo de vida Incremental

(AUDY; PRIKLADNICKI, 2007)

- Espiral: este modelo foi desenvolvido para englobar as melhores características dos ciclos de vida cascata e incremental, ao mesmo tempo que adiciona um novo elemento, a análise de risco, que não existe nos modelos anteriores, possuindo quatro importante atividades: planejamento, análise de riscos, engenharia e avaliação do cliente.

Figura 5. Modelo ciclo de vida Espiral

É possível identificar claramente que uma das principais características dos métodos clássicos é que os requisitos devem ser bem definidos no início do projeto, dificultando as alterações de requisitos no decorrer do projeto. Em princípio, é difícil para o cliente especificar os requisitos explicitamente como um todo.

2.4 Metodologias Ágeis

A expressão “Metodologias Ágeis” tornou-se conhecida em 2001, quando especialistas em processos de desenvolvimento de software representando entre outros, os métodos Scrum e Extreme Programming (XP), foram estabelecidos princípios e características comuns destes métodos. Assim foi criada a “Aliança Ágil” e efetuou-se o estabelecimento do “Manifesto Ágil” (REIS, 2013) .

Metodologia Ágil é uma forma de gestão e desenvolvimento de Software que usa uma abordagem de planejamento e execução iterativa e incremental voltado para processos empíricos - complexos, caóticos ou com muita incerteza, tem mudança ao longo do processo, não são repetitivos e são imprevisíveis (STEFFEN, 2012).

A metodologia divide o problema em produtos menores visando entregar software funcionando regularmente, visa a aproximação e maior colaboração do time de desenvolvimento com os especialistas de negócios, comunicação face a face, redução dos riscos associados às incertezas dos projetos. Procura aceitar e responder as mudanças de forma mais rápida e natural e é claro, visa principalmente a satisfação final dos clientes por meio da adoção de práticas de gestão e de engenharia de software com foco nos valores e princípios do *agile*. Seguindo desta forma o propósito central da metodologia ágil, de acordo com o Manifesto Ágil (2001):

- Indivíduos e interações ao invés de processos e ferramentas
- Software operante ao invés de documentações completas
- Colaboração do cliente ao invés de negociações contratuais
- Responder à mudanças ao invés de seguir um planejamento.

De forma simplificada, podemos dizer que seu principal objetivo é entregar o produto que o cliente realmente deseja com qualidade e que lhe será útil.

A seguir, são apresentadas algumas vantagens das metodologias ágeis do ponto de vista do cliente e das equipes de desenvolvimento de software (STEFFEN, 2012):

Vantagens (Cliente):

- Foco e maximização do ROI (Retorno do Investimento) e do Valor de Negócio;
- Entregas do produto mais rápidas, frequentes e regulares;

- Aceleração do *Time-to-marke*t o que se traduz em ganho de competitividade;
- Maximização do *Value-to-Makert*: Foco no que é prioritário e traz mais valor para o usuário, o que se traduz em ganho de usabilidade;
- Transparência e visibilidade do status do projeto;
- Flexibilidade para mudanças de requisitos e prioridades além de maior agilidade na tomada de decisões;
- Melhoria da Qualidade do produto final;
- Produtividade;
- Redução dos riscos e das indesejáveis surpresas.

Vantagens (gestor e equipes)

- Escopo e objetivos claros e priorizados;
- Equipes auto gerenciáveis, maior autonomia, disciplina e regularidade;
- Maximização do comprometimento;
- Melhoria na comunicação. A comunicação intensa com o cliente e a gestão de suas expectativas são partes do processo;
- Inspeção e Adaptação constantes do processo em busca da melhoria contínua e a redução dos desperdícios;
- Antecipação dos problemas e maior agilidade na tomada de ações.

3 Scrum

Esse capítulo apresenta o framework Scrum onde:

- Item 3.1 apresenta uma introdução ao Scrum;
- Item 3.2 apresenta a teoria do Scrum;
- Item 3.3 apresenta os papéis necessários e utilizados no Scrum;
- Item 3.4 apresenta os artefatos elaborados;
- Item 3.5 apresenta os eventos necessários e utilizados no Scrum.

3.1 Introdução

O termo Scrum é derivado do rugby. Esse termo vem do artigo “The New New Product Development Game”, de Hirotaka Takeuchi and Ikujiro Nonaka (1986), onde é feita a analogia ao esporte.

O Scrum foi desenvolvido por Ken Schwaber e Jeff Sutherland e, segundo eles, Scrum não é um processo ou uma técnica para construção de produtos, mas sim, é um framework no qual você pode empregar diversos processos e técnicas. Tem sido usado para gerenciar o desenvolvimento de produtos complexos desde o início dos anos 1990. Scrum deixa clara a relativa eficácia de sua gestão de produtos e práticas de desenvolvimento para que você possa melhorar (SCHWABER; SUTHERLAND, 2012).

O Scrum tem base no empirismo que afirma que o conhecimento vem da experiência e a tomada de decisões é baseada no que sabemos. Consiste em Equipes do Scrum associadas a seus papéis, eventos, artefatos e regras e, cada componente dentro desse framework tem uma função específica e é essencial para o uso e o sucesso do Scrum.

Nas próximas seções são apresentadas as funções específicas de cada componente envolvido no Scrum.

3.2 Teoria do Scrum

Segundo Schwaber e Sutherland (2012), Scrum emprega uma abordagem interativa e incremental para otimizar a previsibilidade e controle de risco. É baseado

em três pilares: transparência, inspeção e adaptação (SCHWABER; SUTHERLAND, 2012).

- Transparência: mesma visão para todos os participantes seja processo, definições ou outros aspectos. Por exemplo, é preciso que aqueles que realizam o trabalho e os que aceitam o resultado do trabalho tenham o mesmo entendimento do que é uma entrega pronta considerando que o propósito de cada Sprint é entregar incrementos de funcionalidades utilizáveis e que atendam a essa definição de pronto, entendida e conhecida pelo time Scrum.

- Inspeção: acompanhamento dos artefatos e da evolução para alcançar o objetivo, é preciso ter cautela para não atrapalhar o andamento.

- Adaptação: ajustes no processo devem ser feitos sempre que necessário e o mais breve possível para minimizar possíveis impactos.

Para sustentar esses pilares, o Scrum determina quatro eventos formais: reunião de planejamento de Sprint, reuniões diárias, revisão de sprint e retrospectiva do sprint.

Abaixo está o fluxo do Scrum, dessa forma podemos visualizar seu funcionamento.

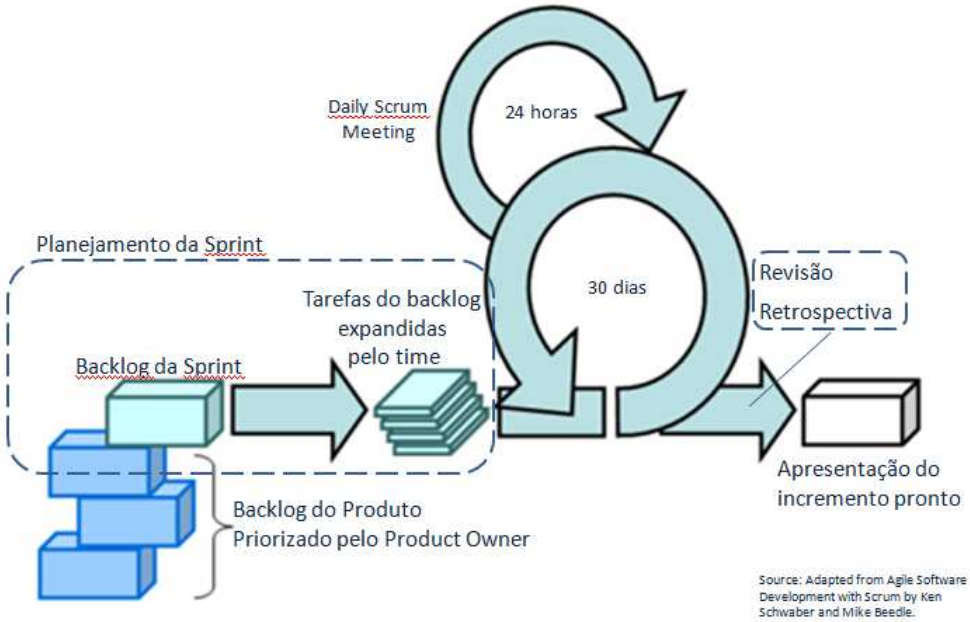

Figura 6. Fluxo do Scrum

De forma resumida, o Scrum se baseia em uma lista de requisitos que são priorizados pelo Product Owner (backlog do produto). Após isso é definido o que

será tratado na Sprint (backlog da Sprint) e a equipe expande as tarefas (planejamento da Sprint). Durante o período de duração da Sprint onde o desenvolvimento é feito são realizadas reuniões diárias para acompanhamento. Ao término da Sprint são realizadas as reuniões de revisão e retrospectiva juntamente com a entrega do incremento pronto. Ao término de uma Sprint inicia-se o planejamento da próxima.

Como já dito antes, para o sucesso desses eventos, o Scrum se baseia em papéis e artefatos específicos e bem definidos, os quais são descritos detalhadamente nos próximos itens.

3.3 Papéis

Os times de Scrum devem ter 2 características fundamentais, serem auto-organizáveis e multifuncionais, onde:

- Auto-organizáveis: a equipe define qual a melhor forma de trabalharem

- Multifuncionais: a equipe não depende de outras pessoas, externas ao time, para finalizar seu trabalho, todas as competências necessárias para isso estão dentro do próprio time Scrum.

O Scrum é composto de três papéis, product owner, equipe de desenvolvimento e o Scrum Master.

3.3.1 Product Owner

O Product Owner ou Dono do Produto é uma única pessoa, mas ele pode ser um centralizador caso esteja representando um comitê.

É o responsável por elaborar o Product Backlog, deve conhecer as necessidades do cliente. É ele que levanta, detalha, prioriza e apresenta os requisitos, sendo o responsável também pelo entendimento desses requisitos por todo o time, esclarecendo dúvidas sempre que preciso. Além disso, também é o responsável por garantir o retorno de investimento e integrar os ambientes com mais de um cliente Scrum Guide (SCHWABER; SUTHERLAND, 2011).

O product owner deve ser uma figura respeitada por todos, pois ele é o único que poderá falar com a equipe de desenvolvimento sobre as priorizações e o que deve ser feito. Em suma, é o gerenciador do backlog de produto.

3.3.2 Equipe de desenvolvimento

São os profissionais que vão executar o trabalho para entregar cada versão pronta ao final das Sprints.

A equipe de desenvolvimento tem estrutura e autoridade para organizar e gerenciar seu próprio trabalho, não havendo intervenção nem mesmo do Scrum Master. Não há sub-equipes dentro da equipe de desenvolvimento e, independente da característica de cada profissional, todos são considerados desenvolvedores e a equipe como um todo é responsável pelo trabalho.

Segundo o Scrum Guide (SCHWABER; SUTHERLAND, 2011), o tamanho ideal para a equipe de desenvolvimento é de 3 a 9 pessoas. Menos que 3 pode haver restrição de habilidades gerando entregas não utilizáveis e, mais que 9 integrantes é necessário que haja muita coordenação e acaba gerando maior complexidade para o processo empírico gerenciar.

3.3.3 Scrum Master

O Scrum Master é o responsável por garantir que o Scrum aconteça, garantindo a aderência do time às teorias, práticas e regras do Scrum. Segundo o Scrum Guide (SCHWABER; SUTHERLAND, 2011), é ele que orienta a equipe a entender quais as suas interações com o Time Scrum são úteis e quais não são. O Scrum Master ajuda todos a mudarem estas interações para maximizar o valor criado pelo Time Scrum.

Além de garantir o uso do Scrum, é de sua responsabilidade remover os impedimentos do time e minimizar ao máximo as interferências externas. O scrum master oferece apoio a todo o time Scum e para a organização.

3.4 Artefatos

Os artefatos do Scrum servem de base para os três pilares desse framework (transparência, adaptação e inspeção). Segundo o Scrum Guide (SCHWABER; SUTHERLAND, 2011), os artefatos definidos para o Scrum são especificamente projetados para maximizar a transparência das informações chave necessárias para assegurar que o Time Scrum tenha sucesso na entrega do incremento.

3.4.1 Backlog do Produto

O Backlog do Produto ou Product Backlog é uma lista de requisitos priorizados. Os requisitos do topo da lista são os mais conhecidos e que possuem maior detalhe e são esses que serão os próximos a serem desenvolvidos, para isso é necessário que cada requisito tenha detalhe suficiente para que possa atender ao conceito de pronto entendido pelo time Scrum.

Segundo o Scrum Guide (SCHWABER; SUTHERLAND, 2011), o Backlog do produto evolui enquanto houver evolução no produto e no ambiente em que esse produto será utilizado, o que o torna dinâmico e vivo. Essa lista, além da ordenação, possui a descrição e a estimativa de cada requisito e é nela que estão listadas todas as características, funções, requisitos, melhorias e correções que formam as mudanças que devem ser feitas no produto nas futuras versões.

Para a priorização dos requisitos considera-se o valor, custo, necessidade e risco de cada um.

A responsabilidade desse artefato é do Product Owner, porém a estimativa é dada pela equipe de desenvolvimento sempre com o apoio do Product Owner para esclarecimentos e decisões.

Através desse artefato é possível monitorar quanto falta para concluir o trabalho previsto e essa informação deve ser transparente para todos.

3.4.2 Backlog da Sprint

É o Backlog da Sprint que define qual trabalho será feito pela equipe de desenvolvimento para atender os itens do Backlog do Produto.

Segundo o Scum Guide (SCHWABER; SUTHERLAND, 2011), o Backlog da Sprint ou Sprint Backlog é um conjunto de itens do Backlog do Produto selecionados para a Sprint, juntamente com o plano de entrega do incremento do produto para atingir o objetivo da Sprint.

Esse artefato é alterado durante toda a Sprint pela equipe de desenvolvimento, adicionando ou complementando as informações, inclusive a estimativa real e é utilizado nas reuniões diárias.

Da mesma forma que o Backlog do Produto, no Backlog da Sprint é possível monitorar a evolução dos trabalhos e saber quanto ainda está pendente, esse monitoramento é feito nas reuniões diárias.

3.4.3 Incremento

São todos os itens do Backlog do Produto. Lembrando que esse artefato foi complementado durante o andamento de todas as Sprints. O incremento deve sempre atender ao conceito de pronto conhecido pelo time Scrum, mesmo que este não venha a ser liberado pelo Product Owner.

3.5 Eventos

Os eventos do Scum servem para criar uma rotina do dia-a-dia do time Scrum, minimizando a necessidade de eventos não planejados e que possam impactar nas tarefas planejadas. No Scrum para todos os eventos há uma duração máxima determinada, esse tipo de evento é conhecido como evento time-boxing e é utilizado para que não seja gasto mais tempo do que o necessário, não interferindo na produtividade.

Todos os eventos determinados no Scrum possuem sempre o objetivo de ter transparência e permitir a inspeção e adaptação.

3.5.1 Sprint

Considerada o coração do Scrum a Sprint é um evento com tempo máximo de duração determinado de um mês. Ao final da Sprint deve ser entregue uma versão pronta do produto, possível de utilização e é iniciada a próxima Sprint.

Os demais eventos do Scrum acontecem durante a execução da Sprint: reunião de planejamento da Sprint, reuniões diárias, revisão da Sprint e retrospectiva da Sprint. Também faz parte da Sprint o trabalho de desenvolvimento.

Caso seja necessário, o Product Owner é a única pessoa que pode cancelar uma Sprint. Cancelamentos não são comuns visto que o tempo de duração da sprint é curto e é preciso deslocar todo o time para o planejamento de uma nova sprint.

Há apenas algumas situações onde o cancelamento é feito, como por exemplo, o objetivo da sprint não fazer mais sentido devido a uma mudança na estratégia da organização, do mercado ou de alguma tecnologia. Caso isso aconteça, é necessário revisar os itens do backlog, mesmo aqueles que já estão prontos.

3.5.1.1 Reunião de Planejamento de Sprint

A Reunião de Planejamento de Sprint ou Sprint Plannig Meeting é um evento com duração de oito horas considerando uma sprint de um mês. Caso a sprint tenha uma duração menor, o tempo da reunião deve ser proporcionalmente menor.

Essa reunião é divida em duas partes onde, em cada parte, é utilizado metade do tempo total da reunião de planejamento da Sprint. Tem como entrada o backlog do produto e como saída gera-se o backlog da sprint. Vejamos o que é feito em cada uma das partes.

- Primeira parte: O product owner apresenta o backlog do produto, já priorizado anteriormente por ele, para todo o time scrum. Com isso, a equipe de desenvolvimento irá estimar o tempo necessário para atender cada um deles e definir quais itens eles podem atender na sprint planejada. Posteriormente, é definida a meta da Sprint que é uma justificativa do porque a equipe irá trabalhar com o incremento escolhido.
- Segunda parte: A equipe de desenvolvimento define como ela irá trabalhar para gerar o incremento pronto ao final da sprint. Com isso é gerado o backlog da sprint. Antes de encerrar a reunião, ainda é preciso que a equipe de desenvolvimento defina qual trabalho será feito nos primeiros dias da sprint. Nesta segunda parte da reunião, a presença do product owner não é obrigatória, fica a cargo da equipe de desenvolvimento decidir se ele participa ou não, sua presença é válida caso ainda haja necessidade de esclarecimentos dos requisitos e seja identificado o excesso ou falta de trabalho para a sprint em planejamento.

Ao final da reunião, a equipe de desenvolvimento apresenta ao product owner e ao scrum master como será realizado o trabalho para atender o objetivo.

3.5.1.2 Reuniões Diárias

As Reuniões Diárias ou Daily Scrum Meeting é um evento com duração de quinze minutos e apenas a equipe de desenvolvimento participa, porém eles contam com o apoio e orientação do scrum master

que os ensina como conduzir a reunião e não estourar o tempo planejado para o evento. Não é uma reunião de status.

O objetivo dessa reunião é acompanhar a evolução das atividades planejadas, saber o que mudou do dia anterior para o atual e planejar o dia seguinte. São as três perguntas abaixo, extraídas do Scrum Guide (SCHWABER; SUTHERLAND, 2011), que devem ser esclarecidas por cada um da equipe de desenvolvimento:

- O que foi completado desde a última reunião?
- O que será feito até a próxima reunião?
- Quais os obstáculos que estão no caminho?

Dessa forma é possível inspecionar se o objetivo da sprint será atendido conforme o planejado.

Conforme consta no Scrum Guide (SCHWABER; SUTHERLAND, 2011), essas reuniões são chave para a inspeção e adaptação, melhoram as comunicações, eliminam outras reuniões, identificam e removem impedimentos para o desenvolvimento, destacam e promovem rápidas tomadas de decisão, e melhoram o nível de conhecimento da Equipe de Desenvolvimento.

3.5.1.3 Revisão da Sprint

A Revisão da Sprint ou Sprint Review é um evento realizado ao término da sprint com duração máxima de quatro horas considerando uma sprint de um mês e, da mesma forma que a reunião de planejamento da sprint, caso a sprint tenha uma duração menor, o tempo da reunião deve ser proporcionalmente menor.

O objetivo dessa reunião é apresentar ao product owner o resultado final da sprint para que ele verifique o incremento pronto e tire dúvidas e é uma nova oportunidade para que a equipe de desenvolvimento troque experiências do que foi positivo e/ou negativo na sprint. Com isso o backlog do produto pode ser incrementado e adaptado caso haja necessidade e todo o time contribui para o planejamento da próxima sprint.

3.5.1.4 Retrospectiva da Sprint

A Retrospectiva da Sprint ou Sprint Retrospective é realizada entre a reunião de revisão da sprint e a reunião de planejamento da próxima sprint e tem duração de três horas considerando uma sprint de um mês, também deve ter tempo proporcional caso a sprint tenha uma duração menor.

O objetivo dessa reunião é o time analisar como foi o trabalho de cada um, como foi a sprint e identificar melhorias (inspeção e adaptação), seja de pessoas, processos, relações ou ferramentas. Após isso é feito um plano para atender essas melhorias que devem fazer parte da próxima sprint. Dessa forma o time é motivado a sempre melhorar aumentando a qualidade do produto entregue e, caso seja necessário, a definição de pronto feita pelo time pode ser adaptada também.

4 Startup Enxuta (Lean Startup)

Esse capítulo apresenta o método Startup Enxuta onde:

- Item 4.1 apresenta uma introdução a Startup Enxuta;
- Item 4.2 apresenta a teoria da Startup Enxuta;
- Item 4.3 apresenta o funcionamento do método.

4.1 Introdução

Não podemos dizer que a metodologia *Startup* Enxuta é exclusiva para desenvolvimento de sistemas. Na verdade é um método para desenvolvimento de empresas e produtos que visa criar novos negócios e novos mercados, porém essa metodologia visa eliminar o desperdício de tempo e dinheiro o que faz com que tenhamos agilidade nos processos. Com isso, seus conceitos podem ser aplicados em projetos ágeis.

O Sebrae define que *startups* são “as empresas de pequeno porte, recém-criadas ou ainda em fase de constituição, com atividades ligadas à pesquisa e desenvolvimento de ideias inovadoras, cujos custos de manutenção sejam baixos e ofereçam a possibilidade de rápida e consistente geração de lucros”. Eric Ries, criador do método da *Startup* Enxuta, trabalha com uma definição mais ampla, ele considera que uma *startup* “é uma instituição humana projetada para criar novos produtos e serviços sob condições de extrema incerteza.” (RIES, 2011). Para este trabalho utilizaremos a segunda definição.

A palavra enxuta, segundo o próprio Ries (2011), é derivada do processo de manufatura enxuta originado no Japão com a revolução que Taiichi Ohno e Shigeo Shingo promoveram na Toyota alterando radicalmente a condução das cadeias de suprimento e dos sistemas de produção entre outras características.

O pensamento enxuto tem como uma das suas principais características aproveitar o conhecimento adquirido e focar nas tarefas de maior valor. “A aposta-chave é que entendendo o que é valor para o cliente você será capaz de identificar e eliminar os desperdícios, via o melhoramento contínuo dos processos de produção, e assim alavancar a sua posição competitiva, em particular no que se refere à fatores como a velocidade no atendimento aos clientes, a flexibilidade para se

ajustar ao seus desejos específicos, a qualidade e o preço do produto ou serviço ofertados" (COSTA; JARDIM, 2010). O pensamento enxuto trabalha, basicamente, com a identificação do que tem valor para o cliente e dos desperdícios no fluxo de produção, com isso esse fluxo é ajustado e implantado, podendo sempre ser aperfeiçoado.

Com esses conceitos e após algumas tentativas que fracassaram, em 2008, Eric Ries criou a *Startup* Enxuta.

O método *Startup* Enxuta oferece *startups* com um framework para geração de hipóteses de produtos, para rapidamente testar essas hipóteses com os clientes e, em seguida, determinar se será refinado ainda mais o produto ou haverá mudança de recursos em direção a um novo produto. Este processo destina-se a aumentar a velocidade do aprendizado e diminuir o desperdício de recursos valiosos (PRESS, 2012).

As *startups* enxuta são elaboradas com base em três pilares (RIES, 2008):

- **Comoditização da tecnologia**: É cada vez mais fácil e barato para as empresas levar produtos ao mercado, alavancando software livre e de código aberto, a computação em nuvem, os dados sociais abertos (Facebook, OpenSocial) e de distribuição livre (AdWords, SEO). *Startups* enxutas têm a capacidade de usar essa pilha de commodities para reduzir os custos e, mais importante, reduzir o tempo para o mercado.
- **Desenvolvimento de software ágil**: Agilidade permite às empresas construir software de qualidade superior mais rápido. Isso acelera o ciclo de feedback Ideias-Código-Dados. Combinado com as tendências tecnológicas acima, ele também permite estratégias de implantação rápida, como o *just-in-time*.
- **Desenvolvimento do cliente**: Não é suficiente apenas construir um produto com grandes recursos - você tem que descobrir se há um mercado para isso. A única maneira de fazer isso é ir para fora e testar suas hipóteses contra a realidade. A maior fonte vantajosa de custo/hora que todas as empresas enxutas têm é de evitar desenvolvimentos com características que os clientes não querem.

4.2 Teoria da Startup Enxuta

A teoria da *startup* enxuta se baseia em 5 princípios, os quais são esclarecidos nos tópicos abaixo:

1. **Empreendedores estão em toda parte** - Segundo Connolly (2012), a ideia que a maioria das pessoas tem de um empreendedor é a de um Steve Jobs, tipo visionário criativo. Na realidade, ter a capacidade para implementar uma ideia e gerenciar as competências é tão importante quanto a criatividade. A abordagem de *startup* enxuta pode funcionar em empresas de qualquer tamanho, mesmo numa de grande porte, em qualquer setor ou atividade (RIES, 2011). Empreendedores podem ser quaisquer pessoas que trabalham na incerteza para criar novos produtos e serviços.
2. **Empreendedorismo é administração** - Como uma *startup* é considerada uma instituição e não um produto, ela requer um tipo específico de gestão voltada especificamente para seu contexto de incerteza (RIES, 2011).
3. **Aprendizagem validada** - Connolly (2012) menciona que as startups não existem apenas para construir produtos, mas para construir um negócio sustentável. Ries (2011) afirma que elas existem para aprender a desenvolver um negócio sustentável. Essa aprendizagem pode ser validada cientificamente por meio de experimentos frequentes que permitem aos empreendedores testar cada elemento de sua visão.
4. **Construir – Medir – Aprender** - A atividade fundamental de uma *startup* é transformar ideias em produtos, medir como os clientes reagem, e, então aprender se é o caso de pivotar ou perseverar. Todos os processos de *startup* bem-sucedidos devem ser voltados a acelerar esse ciclo de *feedback* (RIES, 2011). O termo pivotar e perseverar serão muito utilizados neste capítulo. Para o termo pivotar utilizaremos como definição a necessidade de ir para outra direção e testar novas hipóteses. Perseverar é o oposto, quando devemos seguir com as hipóteses já testadas.
5. **Contabilidade para inovação** - Embora possa não ser empolgante, o gerenciamento de processos de uma empresa pode ser essencial para seu sucesso (CONNOLLY, 2012). Para as startups é necessário um tipo específico de contabilidade. É preciso definir como medir o progresso, marcos e como priorizar o trabalho, com isso é possível melhorar os resultados do empreendedorismo e atribuir responsabilidade aos inovadores (RIES, 2011).

4.3 Entendendo o método

Como todo processo de condução de um novo negócio ou produto, as *startups* também possuem uma visão clara do objetivo final: criar um negócio próspero e capaz de mudar o mundo (RIES, 2011). Para atender uma visão é preciso ter uma estratégia e gerar um resultado final, chamado de produto. Exemplificado na ilustração abaixo:

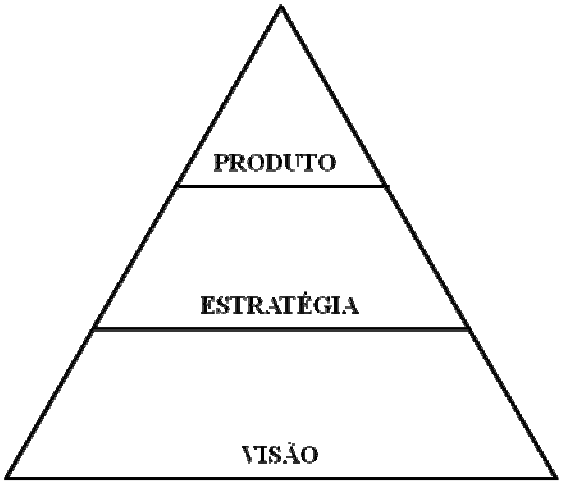

Figura 7. Cadeia para gerar um produto

(RIES, 2011)

No caso das startups, o produto sempre poderá ser otimizado de acordo com o aprendizado adquirido no processo de medição da aceitação dos clientes e, caso seja necessário uma mudança na estratégia, é feito o chamado pivô. Conforme apresentado na ilustração abaixo.

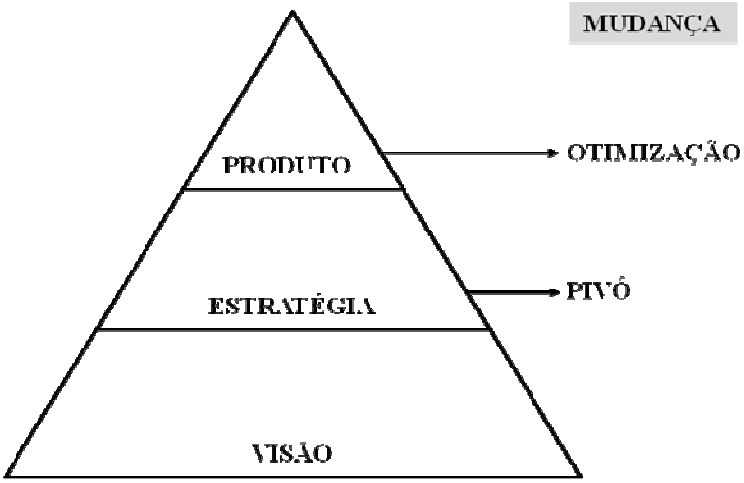

Figura 8. Cadeia no Startup para gerar um produto

(RIES, 2011)

Para iniciar esse método é preciso ter uma ideia de produto ou negócio como ponto de partida se enquadrando no conceito de visão para alcançar um objetivo final.

O método não se baseia simplesmente em sair desenvolvendo esta ideia, ignorando gestão, processo e disciplina, o que conduz muito mais ao insucesso do que ao sucesso, fazendo com que novos produtos não tenham saída nas lojas, novos negócios vão à falência e sistemas não sejam utilizados. Esses insucessos, além do prejuízo financeiro ocasionado, são um grande desperdício de recursos como tempo e pessoas.

Um dos objetivos da *startup* enxuta é evitar o insucesso. Para isso, o método se baseia no conceito de aprendizagem validada. Segundo Ries (2011), a aprendizagem validada é um processo para demonstrar empiricamente as descobertas realizadas por uma equipe a respeito das perspectivas de negócios atuais e futuras de uma *startup*. Com esse conceito, podemos identificar e eliminar fontes de desperdício.

A visão que Ries (2011) defende é a ideia de uma nova disciplina de administração empresarial, pois pelos métodos tradicionais não seria possível eliminar o desperdício. É preciso que as pessoas meçam sua produtividade de maneira diferente para descobrir a coisa certa a criar e pela qual os clientes se interessem e se disponham a pagar por ela. A *startup* enxuta é uma nova maneira de considerar o desenvolvimento de produtos novos e inovadores, que enfatiza a interação rápida e percepção do consumidor, uma grande visão e grande ambição, tudo ao mesmo tempo.

Para garantir a interação rápida e, conforme mencionado, sem sair desenvolvendo tudo, é preciso realizar experimentos que são os produtos desenvolvidos por uma *startup*, seu resultado é a aprendizagem de como desenvolver uma empresa de forma sustentável. Para isso precisamos também definir qual o mínimo que temos que apresentar a um cliente para que possamos aprender se é válido continuarmos com base na ideia inicial ou se é necessário mudar radicalmente.

Esse mínimo que foi mencionado é o chamado MVP, do inglês *Minimum Viable Product* e, traduzido para o português, Produto Mínimo Viável. Com a definição

desse conceito temos que o MVP é a versão do produto que permite uma volta completa no ciclo construir - medir - aprender, com o mínimo de esforço e o menor tempo de desenvolvimento. Ou seja, é o produto mínimo que a *startup* tem que gerar para poder validar as hipóteses e iniciar o processo de aprendizagem, possibilitando a escolha entre pivotar ou perseverar. O MVP tem como lição que qualquer trabalho adicional ao que é necessário para validar a aprendizagem é um desperdício, independentemente da importância que lhe foi dada em um determinado momento (RIES, 2011).

Segundo o manual da startup (RIES, 2010), o MVP toma qualquer forma que seja necessária para garantir um aprendizado relevante que ajude a Startup a caminhar em direção ao sucesso. Mais do que a forma específica que o MVP toma para um determinado experimento, o mais importante é a criação da cultura de experimentação que permite o aprendizado, gastando a menor quantidade de recursos e tempo possível.

São duas as hipóteses validadas com o MVP (RIES, 2011):

- **Hipótese de Valor** – É formulada para testar se o produto ou serviço de fato fornece valor aos clientes no momento em que o estão utilizando.
- **Hipóteses de Crescimento** – É formulada para testar como os novos clientes descobrirão um produto ou serviço.

Essas hipóteses são as duas suposições mais importantes das *startups* pois elas que geram variáveis de ajustes que controlam o motor de crescimento da startup e em cada iteração são melhoradas possibilitando uma evolução.

O motor de crescimento de uma *startup* é composto pelos ciclos de *feedback* e, para que esse motor de crescimento funcione perfeitamente, é preciso que o tempo entre um ciclo e outro seja precisamente gerenciado.

Com base no que vimos até agora, podemos perceber que o fundamental para o sucesso de uma *startup* enxuta são os ciclos de *feedback* que consistem em sabermos o mínimo que temos que desenvolver para podermos medir e aprender com o resultado de cada experimento. Esse será nosso próximo assunto.

4.3.1 Ciclo de Feedback (Construir – Medir – Aprender)

Já sabemos que o fundamental para o sucesso de uma *startup* enxuta são os seguintes itens que compõem o ciclo construir - medir - aprender:

- Saber qual o produto mínimo viável (MVP) que vamos utilizar como experimento;
- Saber o que vamos medir com o resultado desse experimento;
- Aprender com o resultado gerado

Como já vimos anteriormente, é nesse ciclo que as ideias são transformadas em produtos e, conforme os clientes interagem com os produtos, são gerados os *feedbacks* e dados que servem para o aprendizado. O *feedback* pode ser de dois tipos:

- **Qualitativo** – Se os clientes estão ou não gostando do produto
- **Quantitativo** – Qual o volume de clientes que está utilizando o produto e quantos deles consideram que esse produto tem valor

Esse ciclo está no centro do modelo da *startup* enxuta e é através dele que temos as informações necessárias para sabermos se a *startup* está no caminho certo ou não, ou seja, se temos que pivotar ou perseverar.

Vejamos graficamente esse ciclo:

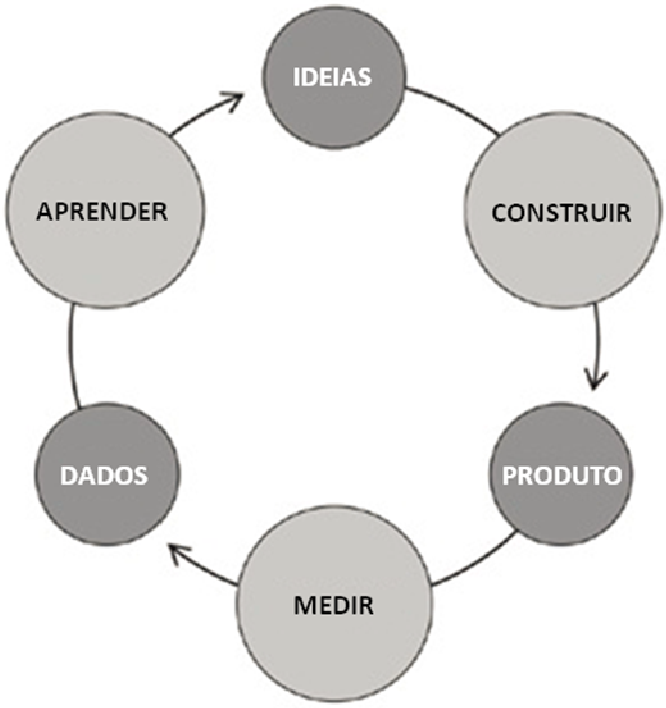

Figura 9. Ciclo de feedback Construir – Medir – Aprender

(RIES, 2011)

A ideia principal desse ciclo de *feedback* não é focar em uma tarefa específica, mas sim focar em minimizar o tempo total gasto nele.

4.3.1.1 Construir

Antes de definir qual será o produto mínimo viável que será desenvolvido para iniciar o ciclo, é preciso definir as hipóteses. Essas hipóteses são definidas com base no que Ries (2011) chama de atos de fé, que são os elementos mais arriscados da *startup*, pois são suposições das quais todo o empreendimento depende delas. Caso essas suposições sejam verdadeiras, uma grande oportunidade estará à espera e, se forem falsas, a startup se arrisca ao total fracasso.

Com as hipóteses de valor e de crescimento definidas e originadas as variáveis de ajuste onde, a cada iteração será uma tentativa para impulsionar o crescimento da *startup*, chega o momento de desenvolver o mais rápido possível o produto mínimo viável (MVP). Com ele podemos iniciar o quanto antes o processo de aprendizagem com o mínimo de esforço necessário dando uma volta completa no ciclo de *feedback*, possibilitando identificar se o produto irá criar valor para o cliente ou não.

Com base no que acabamos de falar, é importante ressaltar que o mínimo produto viável não precisa ser perfeito para poder testar as hipóteses fundamentais do negócio. Antes da divulgação para todo o mercado, os testes iniciais são feitos com clientes especiais conhecidos como adotantes iniciais. Segue algumas características apresentadas por Ries (2011) sobre eles:

- São aqueles que sentem a necessidade pelo produto de modo mais aguçado, não precisando estar com a solução perfeita para conseguir capturar o interesse deles;
- Utilizam sua imaginação para completar o que falta do produto;
- Estão dispostos a utilizar o produto mesmo antes de ele estar pronto;
- Sua principal preocupação é serem os primeiros a utilizar ou adotar o novo produto/tecnologia;
- Tendem a ser mais tolerantes com os erros;
- São em particular ávidos em fornecer *feedback*.

Não há um padrão de como deve ser desenvolvido um produto mínimo viável. Sua complexidade varia para cada *startup*, ficando essa decisão para os empreendedores e desenvolvedores. Caso haja dúvidas, por exemplo, de quantas

funcionalidades devem ser apresentadas em um MVP, deve-se sempre deixar o mais simplificado possível.

Como exemplo de possíveis tipos de produto mínimo viável que podem ser utilizados temos o protótipo que não precisa ser elaborado com uma tecnologia avançada, vídeo de apresentação de uma tecnologia ou produto e o MVP com *concierge*. Este último, segundo Croll e Yoskovitz (2013) é um tipo de MVP utilizado quando o investimento não compensa. A verificação para saber se há uma necessidade real do serviço é feita manualmente com os primeiros clientes, o que possibilita também validar o que de fato será utilizado e possibilita o aperfeiçoamento sem escrever sequer uma linha de código e sem contratar nenhum empregado. Ries (2011) afirma que num MVP com *concierge*, o serviço manual/personalizado, não é o produto, mas uma atividade de aprendizagem elaborada para testar as suposições no modelo de crescimento da empresa.

O produto mínimo viável é apenas o primeiro passo para a aprendizagem. Pode acontecer de, após diversas iterações, identificarmos que o produto ou estratégia não é adequado e tenhamos a necessidade de uma mudança, o chamado pivô. Com essa mudança a estratégia é alterada radicalmente.

4.3.1.2 Medir e Aprender

Na fase de medir o maior desafio é determinar se os esforços de desenvolvimento do produto levarão a um progresso real (RIES, 2011). Precisamos saber se estamos melhorando nosso produto, para isso não podemos apenas nos basear no aumento dos números de vendas, a contabilidade padrão, é preciso de uma contabilidade específica para *startups*, com foco na inovação, é a chamada contabilidade para inovação.

Segundo Ries (2011), a contabilidade para inovação funciona em três passos:

- Primeiro, utiliza um produto mínimo viável para estabelecer dados reais a respeito de onde a empresa está naquele exato momento (*baseline*). Sem um quadro claro do *status* corrente – não importa a distância que se está no objetivo –, é impossível acompanhar seu progresso;
- Segundo, as *startups* devem tentar regular o motor a partir da *baseline* na direção ideal. Isso pode exigir muitas tentativas. Após a *startup* fazer todos os ajustes e otimizações de produto possíveis para mover sua *baseline* rumo ao

ideal, a empresa alcança um ponto de decisão. Este é o terceiro passo: pivotar ou perseverar.

- Se a empresa estiver fazendo um bom progresso na direção do ideal, isso significa que está aprendendo de forma apropriada e utilizando aquela aprendizagem de maneira efetiva. Nesse caso, faz sentido continuar. Caso contrário, a equipe gerencial deve acabar concluindo que sua estratégia de produto corrente é imperfeita e requer uma mudança importante. Quando uma empresa pivota, esta começa o processo de novo, estabelecendo uma nova *baseline* e, em seguida, ajustando o motor a partir dali. O sinal de pivô bem-sucedido é que essas atividades de ajuste do motor são mais produtivas após o pivô do que antes.

Através do MVP que é possível obter os dados reais necessários para a definição da *baseline*, alguns desses dados são quantidade de novos cadastros e valor do tempo de vida do cliente. Esse tipo de informação é de extrema importância para aprender sobre os clientes e suas reações com os produtos. É importante, em primeiro lugar, testar as suposições mais arriscadas, porém, para que isso faça sentido, é preciso mitigar os riscos para construir um negócio sustentável. Se isso não for possível não é válido testar as demais suposições.

Na etapa de ajuste do motor de crescimento, é preciso definir além do que será ajustado, como podemos medir se esse ajuste trouxe melhora no comportamento do cliente. Após um ajuste, podemos citar como exemplos duas variáveis que podem refletir essa melhora: aumento do número de clientes e aumento do tempo que o cliente passa utilizando o produto. Esses dados em comparação com a baseline é que nos possibilitam decidir entre pivotar ou perseverar.

Para analisar a evolução da startup uma das ferramentas mais importantes é conhecida como análise de coorte. Análise de coorte consiste em uma série de levantamentos realizados em intervalos de tempo apropriados nos quais a coorte serve como unidade básica de análise. Coorte é um grupo de entrevistados que experimentam o mesmo evento no mesmo intervalo de tempo. A expressão análise de coorte refere-se a qualquer estudo no qual existam medidas de algumas características de uma ou mais coortes em dois ou mais momentos no tempo (MALHOTRA, 2004).

Com essa ferramenta, não analisamos apenas os totais, podemos dividir em alguns grupos específicos, de acordo com a startup e analisar quanto cada grupo

desses representa no valor total. Como exemplos de coortes que podem ser utilizados temos: quantidade de clientes que apenas entrou ou abriu o sistema (nosso produto), quantidade de clientes que entrou/abriu o sistema e que clicou em um determinado menu, etc.

As métricas nos ajudam a medir a evolução da startup. É necessário cautela para não sermos iludidos com as métricas conhecidas como de vaidade. Esse tipo de métrica dá sensação de movimento para frente mesmo que a empresa esteja fazendo pouco progresso Ries (2011). Ela apresenta uma imagem positiva, mas não necessariamente exibe a realidade da startup. Quando esse tipo de métrica mostra evolução nos números, podemos associar esse progresso com as atividades de melhorias que estão sendo desenvolvidas/aplicadas naquele momento, o que pode não ser verdade e, assim, gerando uma ilusão.

Para evitar essa ilusão é preciso trabalhar com as métricas que possuem os três As. São eles:

- **Acionável** – São relatórios que devem demonstrar causa e efeito claros, ajudando a analisar o comportamento do cliente. Dessa forma, sabe-se que ações são necessários para reproduzir os resultados. Quando a causa e o efeito são entendidos com clareza, as pessoas são mais capazes de aprender a partir de suas ações (RIES, 2011).

  - **Acessível** – Os relatórios devem ser elaborados do modo mais simples possível possibilitando o entendimento de todos. Ries (2011) explica que é necessário trabalhar com relatórios baseados em coortes, pois com cada análise de coorte podemos saber entre as pessoas que utilizaram o produto em um período específico, quando delas tiveram os comportamentos que nos interessam. Dessa forma, com o relatório ficando acessível à todos os funcionários, é possível um entendimento comum do relatório.

- **Auditável** – Devemos assegurar que os dados sejam confiáveis para os funcionários, evitando questionamentos quanto a veracidades dos dados. Precisamos ser capazes de testar os dados dos relatórios pessoalmente no mundo real conversando com clientes. Os gerentes precisam da capacidade para conferir os dados por amostragem com os clientes (RIES, 2011).

Conforme já dito anteriormente a contabilidade, muitas vezes, não é algo empolgante, pois geralmente está associada à geração de relatórios financeiros e auditorias, mas ela é fundamental para o sucesso das startups.

Com a contabilidade conseguimos medir com precisão como está a evolução do negócio e, com isso, aprender de forma mais rápida se devemos ou não seguir em frente, ou seja, pivotar ou perseverar. Para tomar essa decisão também é importante considerar quanto tempo temos, geralmente esse tempo é definido pelo dinheiro que é investido na *startup* e quanto está sendo gasto mensalmente. Para prolongar esse período, chamado por Ries (2011) de pista de decolagem, é possível conseguir recursos adicionais ou cortar gastos. Deve-se ter atenção para que não aconteça de, simplesmente, deixar a startup falir de forma mais desacelerada. Ries (2011) afirma que a verdadeira medida da pista de decolagem é saber quantos pivôs ainda é possível de acontecer na *startup*: a quantidade de oportunidades que possui para realizar uma mudança fundamental em sua estratégia empresarial. O objetivo sempre é alcançar a aprendizagem validada com o menor custo.

Muitas vezes os pivôs demoram a acontecer e há alguns motivos para que isso aconteça. Podemos citar as falsas conclusões geradas pelas métricas de vaidade e quando há confusão no entendimento da hipótese. Além disso, consideramos um fator importante o medo que os empreendedores têm, nem tanto de assumir que suas visões estão equivocadas, mas sim de que sejam consideradas equivocadas sem poderem ter uma oportunidade real de serem aprovadas. Tudo isso gera uma perda de foco na validação da hipótese, podendo postergar ainda mais a decisão de pivotar. Sempre devemos considerar o ato de pivotar quando os experimentos com os produtos possuem uma eficácia decrescente e quando há a sensação generalizada de que o desenvolvimento do produto pode ser mais produtivo.

Muitas vezes o termo pivô é utilizado incorretamente para tratar de qualquer mudança. Um pivô é um tipo específico de mudança e tem o objetivo de testar uma nova hipótese fundamental a respeito do produto, do modelo de negócios e do motor de crescimento. Ries (2011) apresenta os seguintes tipos de pivôs:

- **Zoom-in** – O que antes era considerado um recurso isolado num produto torna-se o produto todo.

- **Zoom-out** – É o oposto do *Zoom-in*. O que era considerado o produto completo torna-se um recurso isolado de um produto maior. Pode acontecer de um único recurso ser insuficiente para suportar o produto completo.
- **Segmento de clientes** – A hipótese de produto é confirmada parcialmente, pois a empresa percebe que o produto soluciona um problema real para clientes reais, porém não são o tipo de clientes que a empresa planejou atender.
- **Necessidade de cliente** – A hipótese de produto é confirmada parcialmente, pois por conhecermos os clientes, percebe-se que o problema que estamos tentando solucionar não é muito importante para eles e podemos identificar novos problemas, mais importantes para serem solucionados, o que pode exigir até a construção de um produto completamente novo.
- **Plataforma** – Refere-se a uma mudança de um aplicativo para uma plataforma, e vice-versa.
- **Arquitetura de negócios** – Esse pivô se apropria do conceito de Geoffrey Moore, que observou que as empresas de modo geral seguem uma de duas arquiteturas principais de negócio, sendo elas a de alta margem e baixo volume (modelo de sistemas complexos) e a de baixa margem e alto volume (modelo de operações em volume). Neste pivô, a startup troca de arquitetura.
- **Captura de valor** – É parte da hipótese de produto. O valor criado pelas empresas pode ser capturado de diferentes maneiras. Os métodos são conhecidos como modelos de monetização (independente do produto) e receita. O pivô refere-se a troca da forma de captura de valor que pode gerar consequências de longo alcance para o restante do negócio, do produto e das estratégias de marketing.
- **Motor de crescimento** – A empresa muda a estratégia de crescimento para buscar um crescimento mais rápido ou mais lucrativo. Pode requerer uma mudança na maneira de capturar o valor.
- **Canal** – Canal de vendas ou também conhecido como canal de distribuição é uma terminologia tradicional de vendas que se refere ao

mecanismo pela qual a empresa fornece seus produtos para os clientes. O pivô de canal é o reconhecimento de que a mesma solução básica pode ser fornecida através de um canal distinto com maior eficácia.

- **Tecnologia** – Este pivô é uma maneira da empresa alcançar a mesma solução usando uma tecnologia completamente diferente. A nova tecnologia m em relação a preço e/ou desempenho em comparação com a tecnologia usada antes.

Um pivô é considerado uma hipótese estratégica, que irá exigir um novo produto mínimo viável. Como vimos, é uma mudança estruturada para testar um nova hipótese do produto, do modelo de negócio e do motor de crescimento e é necessário termos as ferramentas necessárias para identificar com agilidade se a *startup* está no caminho certo e/ou identificar novos caminhos a seguir.

4.3.2 Aceleração e crescimento da Startup

No *Lean Startup* o objetivo sempre é aprender a desenvolver um negócio sustentável o mais rápido possível. Para isso há algumas técnicas que permitem às startups enxutas acelerarem através do ciclo de feedback construir - medir - aprender tão rápido quanto possível, mesmo durante sua expansão.

Uma dessas técnicas é trabalhar com lotes pequenos. Trabalhar dessa forma, para a maioria das pessoas não é a melhor opção para realizar uma tarefa, mas essa abordagem tem muitas vantagens se comparado com a realização de um desenvolvimento em grandes lotes. Na manufatura enxuta, esse lote pequeno é conhecido como “fluxo de peça única”, pois ele tem o tamanho de lote de uma unidade.

Mesmo que tenham o mesmo tempo de duração nas tarefas individualmente, ao trabalharmos com o lote pequeno, temos os seguintes benefícios que são citados por Ries (2011):

- É possível realizar entregas intermediárias para o cliente com muito mais rapidez;
- Caso o cliente mude de ideia ou não estamos fazendo conforme o cliente espera, conseguimos descobrir da maneira mais rápida, pois com a primeira entrega intermediária ele já valida o que está sendo feito;

- Podemos identificar falhas nos processos com muito mais rapidez já que passamos pelo processo todo logo no início da execução, o que faz com que a qualidade seja melhorada.

Todas essas características dos lotes pequenos faz com que não haja retrabalho e consequentemente evita gastos de tempo e dinheiro. Essa abordagem também é importante para a implantação contínua. Esse conceito tem como ideia principal que a implantação deve acontecer automaticamente e com frequência, não sendo um evento único do projeto. Dessa forma podemos aprender mais rápido com nosso cliente, o que nos gera uma vantagem competitiva essencial para as *startups*.

Para o crescimento da startup não podemos deixar de considerar que novos clientes chegam à empresa através de outros clientes. Segundo Ries (2011) essa ação pode acontecer de quatro maneiras principais:

- Boca a boca: É o crescimento natural que acontece com base no entusiasmo dos clientes satisfeitos com o produto;
- Efeito colateral do uso dos produtos: Quando produtos de moda ou *status* são expostos para outros clientes durante o seu uso influenciando-os a utilizar o produto;
- Através de publicidade financiada: Nesse caso não há interação entre os clientes, a empresa faz publicidade para incluir novos clientes. É importante destacar que para ser uma fonte de crescimento sustentável, a publicidade deve ser paga como resultado da receita recorrente e não com outras fontes, como o capital de investimento;
- Através da compra ou do uso repetido: Esses são os tipos de produtos projetados para serem comprados mais de uma vez, outro são projetados como eventos ocasionais como, por exemplo, um casamento.

Além dessa forma de crescimento, para as *startups*, temos que considerar a importância do motor de crescimento que, como já vimos, validam as métricas definidas e proporcionam a aprendizagem validada. É o motor de crescimento que nos mantém focados nas métricas de importância. Segundo o criador do método *Startup* Enxuta (RIES, 2011), são três os tipos de motor de crescimento:

- Motor de crescimento recorrente: Monitoram a taxa de atrito ou taxa de rotatividade com muita atenção. Essa taxa é definida como a fração de clientes de qualquer período que não permanecem comprometidos com o

produto da empresa. As regras desse motor de crescimento são muito simples. Se a taxa de aquisição de novos clientes for maior que a taxa de rotatividade, o produto crescerá e a velocidade de crescimento é determinada pelo que Eric Ries chama de taxa de composição, que é a taxa de crescimento natural menos a taxa de rotatividade;

- Motor de crescimento viral: O conhecimento do produto se espalha com rapidez de pessoa para pessoa, da mesma forma que um vírus se torna epidêmico. É diferente do boca a boca, pois esse motor de crescimento depende da interação com outras pessoas para o uso normal do produto. É acionado por um ciclo de *feedback* denominado ciclo viral e sua velocidade de crescimento é determinada pelo coeficiente viral, que mede quantos novos clientes utilizarão um produto como consequência de cada novo cliente inscrito. Para o negócio ser sustentável e crescer exponencialmente esse coeficiente deve ser maior que 1,0;
- Motor de crescimento pago: A empresa tem um gasto para conseguir novos clientes. Também é acionado por um ciclo de *feedback*. Cada cliente paga uma determinada soma em dinheiro para o produto durante seu “tempo de vida” como cliente. O valor do tempo de vida do cliente é o que ele paga com a dedução dos custos variáveis. Para saber a velocidade de crescimento da empresa precisamos conhecer, além de quanto a empresa ganha com cada cliente cadastrado, quanto custa a aquisição de um novo. O custo com a aquisição deve ser menor que o que a empresa ganha com cada novo cliente. Com isso há duas maneiras para aumentar a taxa de crescimento, aumentar a receita por cliente ou reduzir o custo de aquisição do novo cliente. A receita dos clientes pode ser utilizada para reinvestir na aquisição de novos.

É através dos motores de crescimento que sabemos se a *startup* alcançou o encaixe produto/mercado, isto é, a *startup* encontrou um conjunto grande de clientes em potencial que querem exatamente o produto da *startup*.

Ries (2011) aponta que um grande desafio das *startups* é manter o motor de crescimento funcionando. Com o tempo, o grupo de clientes que utiliza o produto da *startup* esgota-se, isso pode levar muito ou pouco tempo. Para a empresa não entrar em crise, é preciso que trabalhe com o portfólio de atividades, ao mesmo tempo

regulando seus motores de crescimento e desenvolvendo novas fontes de crescimento para quando os motores se esgotarem de modo inevitável.

Para a aceleração e o crescimento da startup quanto antes conseguimos identi ficar falhas em nosso processo, antes conseguimos corrigi-lo e antes conseguimos voltar ao nosso processo normal e até melhorá-lo. Para agilizar essa identificação do real problema há uma técnica conhecida como Cinco Porquês, que afirma que temos que repetir perguntas de “por que” cinco vezes para podermos identificar a causa real de um problema.

Além de todas essas técnicas mencionadas não podemos esquecer-nos da importância da inovação que é fundamental para o crescimento contínuo de uma startup ou de outras novas startups na organização.

5 Programação Extrema (XP - eXtreming Programming)

Esse capítulo apresenta a metodologia Programação Extrema onde:

- Item 5.1 apresenta uma introdução ao XP;
- Item 5.2 apresenta a teoria do XP;
- Item 5.3 apresenta os valores do XP;
- Item 5.4 apresenta os princípios do XP;
- Item 5.5 apresenta as práticas do XP;
- Item 5.6 apresenta os papéis do XP;
- Item 5.7 apresenta o ciclo de vida do XP.

5.1 Introdução

Do inglês *eXtreme Programming*, a Programação extrema, também conhecida como XP, foi proposta por Kent Beck nos anos 90, como uma resposta aos problemas encontrados no desenvolvimento de software, pois estes eram normalmente associados a metodologia de desenvolvimento.

Kent Beck descreve a Programação eXtrema como uma maneira leve, eficiente, de baixo risco, flexível, previsível, científica e divertida de desenvolver software. A XP reconhece que os projetos precisam dedicar-se à obtenção da redução dos custos e tirar vantagem daquilo que foi economizado (BECK, 2000).

5.2 Teoria da Programação Extrema

De acordo com Beck (2000), a missão da Programação Extrema, é aceitar o risco do projeto como o problema a ser resolvido, defendendo um estilo de desenvolvimento de software que trate os riscos. Sendo necessário para isso, definir recomendações (*guidelines*) para adaptá-lo às condições locais.

O modelo de desenvolvimento de software é baseado em um sistema de variáveis de controle. Neste modelo existem quatro variáveis no desenvolvimento de software (BECK, 2000):

- Custo
- Tempo

- Qualidade
- Escopo

A interação entre estas variáveis definem os valores das mesmas. Podemos aumentar o valor de uma e diminuir o valor de outra ou vice versa, assim como podemos aumentar o valor de uma e agregar o valor em outra. Desta forma, é importante que as quatro variáveis sejam visíveis para todos os envolvidos dos projetos, incluindo programadores e clientes, pois somente desta forma todos poderão escolher a melhor forma de controlar estas variáveis.

Para melhor entendimento sobre esta interação, Beck (2000) cita alguns exemplos:

-Custo: Mais investimento pode melhorar as outras variáveis, porém muito dinheiro no começo do projeto, cria mais problemas do que resolve.

-Tempo: Mais tempo para entrega do produto pode aumentar a qualidade e o escopo.

-Qualidade: Poderão haver ganhos de tempo a curto prazo com o sacrifício da qualidade, porém a qualidade é terrível como variável de controle, pois o custo (humano, comercial e técnico) é enorme.

-Escopo: Um escopo menor faz com que seja possível fornecer maior qualidade. Além de permitir entregar o produto mais cedo (tempo) e mais barato (custo).

O ciclo de desenvolvimento do XP é fundamentalmente baseado em *pair programming* e testes (BECK, 2000):

- Pares de programadores programa juntos.
- O desenvolvimento é voltado a testes. Primeiramente devem ser efetuados os testes e somente então a codificação. Até que todos os testes rodem, a funcionalidade não terminou. Somente após todos os testes aplicados com sucesso e não houver nenhum outro teste que poderia falhar, a funcionalidade pode ser considerada adicionada e terminada.
- Os pares de programadores não aplicam somente os casos de testes. Eles também evoluem o projeto do sistema. Alterações não são restritas a nenhuma área em particular. Os programadores em pares acrescentam valor à análise, ao projeto, à implementação e ao teste do

sistema. Eles podem agregar este valor onde quer que o sistema precise.

- A integração ocorre imediatamente após o desenvolvimento, incluindo teste de integração.

A Programação Extrema é baseada também em valores, princípios e práticas, que serão apresentadas nas próximas seções.

5.3 Valores

Os quatro valores fundamentais da metodologia XP são: comunicação, simplicidade, feedback e coragem. Estes valores definem as atitudes das equipes e as prioridades da metodologia.

5.3.1 Comunicação

Grande parte dos problemas encontrados em projetos ocorrem devido a falha na comunicação ou a falta dela. Existem diversos aspectos que podem levar a isso: o desenvolvedor possui o conhecimento técnico e o cliente o conhecimento do negócio. É necessário que ambas as partes se conversem e consigam entender o problema um do outro. O desenvolvedor precisa entender a importância do negócio para o cliente e é preciso que o cliente entenda os desafios técnicos para implementação do negócio. Muitas vezes o cliente ou o programador informa algo importante para o projeto, mas a informação parece ser ignorada. Desta forma, a comunicação é considerada o primeiro valor da XP.

A XP procura manter as comunicações certas fluindo através do emprego de muitas praticas que não podem ser feitas sem comunicação. São praticas que fazem sentido a curto prazo, como testar, programar em pares e estimar. É a comunicação entre programadores, clientes e gerentes (BECK, 2000).

Além disso, a XP possui um treinador que possui o trabalho de perceber quando as pessoas não estão se comunicando corretamente e restabelecer a comunicação entre elas.

5.3.2 Simplicidade

A XP valoriza a simplicidade no sentido de desenvolver com simplicidade as funções, de acordo com a necessidade do cliente. Não vale a pena implementar

uma tela totalmente configurável para ser utilizada por um único usuário, sendo que algo mais simples já atenderia.

De acordo com Beck (200), a XP está fazendo uma aposta. Está apostando que é melhor fazer uma coisa simples hoje e pagar um pouco mais amanhã para fazer alguma modificação nela se for preciso, do que fazer uma coisa mais complicada hoje que talvez nunca será usada.

5.3.3 Feedback

O Feedback funciona em diferentes escalas de tempo (BECK, 2000), ele pode funcionar na escala de minutos e dias. Os desenvolvedores definem testes unitários para toda a lógica do sistema onde pode haver algum problema, desta forma eles têm o feedback concreto sobre o estado do sistema. Quando os clientes escrevem histórias, caracterizando o sistema, os desenvolvedores as avaliam, assim os clientes têm o feedback concreto sobre a qualidade de suas histórias.

O Feedback também pode funcionar em escalas de maior tempo, como de semanas e meses. Os clientes e os testadores escrevem testes para as histórias, tendo desta forma o feedback sobre o estado atual do sistema. Além disso, os cliente podem acompanhar o cronograma e visualizar a velocidade do time de desenvolvimento, tendo visão se o planejamento terá sucesso.

5.3.4 Coragem

A coragem é o quarto valor da XP, de acordo com Beck (2000), se você não tiver definidos os três primeiros valores, coragem por si só é apenas fuçar em seu próprio sistema. No entanto, quando combinada com comunicação, simplicidade e feedback concreto, a coragem se torna extremamente valiosa.

Quando falamos de coragem na XP, são citados exemplos, como coragem de jogar um código pronto fora e iniciar novamente, pois o mesmo está ruim suficiente para atrapalhar o desenvolvimento e velocidade do projeto. É ter coragem para seguir e corrigir uma falha impeditiva no fim do projeto.

5.4 Princípios

Os princípios fundamentais da XP são (BECK, 2000):

- Feedback rápido: é necessário obter feedback, interpretá-lo e aplicar o que é aprendido no sistema o mais rápido possível.
- Simplicidade Presumida: a XP defende que os problemas devem ser resolvidos da forma mais simples possível. Por mais que isso seja difícil, principalmente para os desenvolvedores, não deve-se planejar o futuro, pensando em reuso, devemos implementar a solução que resolva o problema hoje.
- Mudanças incrementais: as mudanças devem ser feitas divididas em pequenas partes. Grandes mudanças realizadas de uma vez só não funcionam.
- Aceitação das mudanças: as mudanças precisam ser aceitas e de certa forma até bem vindas. Quanto mais opções disponíveis para resolver o problema, mais rápido haverá solução.
- Alta qualidade: se não houver qualidade no trabalho, as chances de falha do projeto crescem muito ou são quase certeiras.

A seguir alguns princípios menos fundamentais, mas que também ajudarão na tomada de decisão e resolução de problemas (BECK, 2000):

- Ensinar aprendendo;
- Investimento inicial pequeno;
- Jogar para ganhar;
- Experimentação concreta;
- Comunicação honesta e franca;
- Trabalhar a favor dos instintos do pessoal, e não contra eles;
- Aceitação de responsabilidades;
- Adaptação local;
- Viajar com pouca bagagem;
- Métricas genuínas.

5.5 Práticas

Para aplicar corretamente os valores e princípios da XP, são propostas algumas práticas. Há uma confiança muito grande na sinergia entre elas, os pontos fracos de cada uma são superados pelos pontos fortes de outras (BECK, 2000).

- Jogo de Planejamento (Planning Game): Os clientes e desenvolvedores se reúnem para priorizar e estimar as atividades. Deve-se iniciar o desenvolvimento com um plano simples e refiná-lo constantemente ao longo do projeto, trabalhando com pequenos ciclos de entrega.
- Ciclos de Entregas Curtos (Small Releases): sempre é possível e interessante fazer versões pequenas, com entregas constantes para aceitação e acompanhamento do cliente.
- Metáfora (Metaphor): para facilitar a comunicação com o cliente e fazer os desenvolvedores e clientes falarem a mesma língua.
- Design Simples (Simple Design): Simplicidade é sempre citado em XP, pois é um de seus princípios, é comum haver confusão sobre código simples e código fácil. É necessário entender que nem sempre o código fácil levará a solução simples para o projeto.
- Time Coeso (Whole Team): o time é multidisciplinar, e é formado por pessoas engajadas.
- Testes de Aceitação (Customer Tests): são testes elaborados pelo cliente junto aos testadores do sistema para aceitar determinada funcionalidade do sistema.
- Semana de 40 horas (Sustainable Pace): é defendido um horário de trabalho saudável, com prioridade para qualidade de vida, evitando-se as horas extras.
- Reuniões em pé (Stand-up Meeting): as reuniões devem ser efetuadas em pé para evitar assuntos desnecessário e agilizar o processo.
- Propriedade Coletiva (Collective Ownership): o objetivo é todos conhecerem as partes do sistema, sem a necessidade de somente uma pessoa alterar parte específica.
- Programação Pareada (Pair Programming): é a programação em pares ou dupla evitando desta forma a possibilidade de problemas e a falta de padronização, já que o código é revisado por duas pessoas.
- Padronização do Código (Coding Standards): é necessário haver padrões de desenvolvimento de código, para que todos possam entender o que foi feito.

- Desenvolvimento Orientado a Testes (Test Driven Development): O desenvolvimento deve ser efetuado sobre os testes criados, de forma que o código funcione para os testes que serão executados. Os testes unitários são essências para garantir a qualidade do projeto.
- Refatoração (Refactoring): neste processo é efetuada a melhoria contínua do desenvolvimento, melhorando a clareza do código, facilitando o entendimento e evitando duplicações.
- Integração Contínua (Continuous Integration): sempre deve-se integrar os novos desenvolvimentos a versão atual do sistema, diminuindo a possibilidade de erros que ocorrem em uma integração posterior.

5.6 Papéis

No XP há papéis definidos porém é importante ressaltar que o objetivo não é simplesmente que as pessoas preencham os papéis determinados, mas sim fazer com que cada membro da equipe contribua com o melhor de si para o projeto. Aqui serão detalhados apenas os principais papéis que precisam ser preenchidos no XP, segundo Beck (2000):

- Programador: Este é o coração do XP. Além de fazer o papel normal de programador, seu principal valor é a comunicação com as outras pessoas. Caso o programa feito ainda depende de uma comunicação que não está pronta, o programador deve definir e escrever testes que testem essa comunicação, garantindo que seja atendida;
- Treinador: É o responsável pelo processo como um todo. É a pessoa que irá manter a equipe focada, deve conhecer toda a aplicação, se é identificado um erro no projeto é o treinador que irá decidir se deve haver alguma interferência ou não e caso seja necessário interferência é preciso fazê-la com cuidado para que a equipe não perca a autoconfiança. Com o amadurecimento da equipe o papel do treinador diminui;
- Rastreador: Seu objetivo é coletar métricas sobre o que está sendo desenvolvido e confrontar com as métricas estimadas verificando possíveis divergências. O rastreador deve tomar cuidado para não

perturbar muito os programadores pedindo por métricas. Deve ser definida uma frequência para essa interferência;

- Clientes: É ele que define o que deve ser feito e com que prioridade.

5.7 Ciclo de Vida

Segundo Beck (2000), o ciclo de vida da XP possui algumas fases durante seu ciclo de vida, são elas:

- Exploração: São definidas as *user stories* iniciais e o rascunho da arquitetura. Do ponto de vista de requisitos, é interessante detalhá-los de forma que se tenha uma visão suficiente para começar o projeto, com uma definição básica para as estimativas;
- Planejamento: É feita entre cliente e programadores para que seja definido qual o maior conjunto de histórias podem ser atendidas no menor prazo possível para a primeira *release* (primeira grande entrega). As *releases* tem duração de 2 a 6 meses;
- Iterações: Após a definição da *release*, esta é quebrada em pequenas entregas, chamadas iterações onde já são construídas algumas das histórias, são entregues e validadas com o cliente. A iteração é composta por diferentes tarefas, entre elas está a escrita dos casos de testes, refatoramento, codificação, realização dos testes e integração. Cada iteração tem duração máxima de 4 semanas;
- Produção: É nessa fase que se certifica que o software está pronto para entrar em produção. É comum a realização de novos testes para provar que o sistema está estável o suficiente;
- Manutenção: É o estado normal de um projeto XP onde alterações são constantes. Deve-se simultaneamente produzir novas funcionalidades, manter o sistema existente rodando, substituir membros do time que partem e incorporar novos membros ao time;
- Morte: É o término do projeto, quando não há mais histórias para serem implementadas. É o momento de escrever uma breve documentação sobre o sistema para auxiliar no entendimento de outras pessoas caso surja alguma necessidade de mudança.

6 Comparação entre os processos

Esse capítulo irá tratar o framework Scrum e os métodos da Startup Enxuta e da Programação Extrema como processos, de acordo com a definição apresentada no capítulo 2 desse mesmo trabalho.

Este capítulo apresenta uma comparação entre os três processos descritos nos itens anteriores.

Inicialmente é apresentado um quadro resumo com algumas das características de cada um dos processos apresentados:

Tabela 2. Comparação entre Scrum, *Startup* Enxuta e XP

	Scrum	**Startup Enxuta**	**XP**
O que é	Um framework.	Uma metodologia	Uma metodologia
Quem define os requisitos	Product Owner	Inicialmente os empreendedores e após o início do ciclo de *feedback* são os adotantes iniciais	Equipe de desenvolvimento junto com o cliente
Possui iterações	Sim	Sim	Sim
Como são chamadas as interações	Sprint	Ciclo de *feedback* construir – medir – aprender	Iterações
Há um tempo máximo para a duração das interações	Sim, de 30 dias.	Não, porém são realizados no menor tempo possível e com o mínimo de esforço.	Sim, até 4 semanas.
Equipe	Multidisciplinar	Multidisciplinar	Multidisciplinar

Nos próximos subitens abordaremos os tópicos apresentados neste quadro.

6.1 Requisitos

Nos três processos há uma fase determinada para levantamento e definição dos requisitos.

No Scrum esse levantamento é feito antes do planejamento da Sprint (iteração). Os requisitos são registrados de forma já priorizada no artefato conhecido como backlog do produto e no planejamento da Sprint são definidos quais requisitos serão atendidos e os mesmos são registrados no artefato backlog da Sprint. A

validação dos requisitos desenvolvidos é feita ao término da iteração, na reunião de revisão da sprint.

Na Startup Enxuta, não há nenhuma documentação específica para o registro dos requisitos. Esses são identificados na fase inicial do experimento, antes do desenvolvimento do MVP. A validação dos requisitos é feita também após a entrega na fase conhecida como aprender, após a fase de medição.

No XP, o registro dos requisitos é feito através do plano de testes elaborado antes da construção e a validação dos requisitos é feita quando todos os testes são feitos no sistema e são satisfatórios, não geram erros.

A grande diferença da Startup Enxuta com os outros processos é que para a validação dos requisitos nem sempre é necessário que o produto esteja construído 100%.

6.2 Iterações

No Scrum e no XP as iterações tem uma duração limite, sendo de no máximo 30 dias para o Scrum e em média de 1 a 4 semanas para o XP. Para a Startup Enxuta não há essa definição, porém a equipe sempre trabalha com o menor tempo possível para poderem o quanto antes validar a aprendizado.

As iterações são controladas e acompanhadas de formas diferentes nos três processos. No Scrum ocorre durante as reuniões diárias e na revisão do Sprint pela validação dos requisitos e dos pontos que podem ser melhorados.

No XP esse controle e acompanhamento são feitos com base nas métricas que são atualizadas periodicamente com o time. Na Startup Enxuta também são utilizadas métricas para medir a evolução geral da startup, porém a iteração (experimento) é validado pelas hipóteses de valor e de crescimento que foram definidas inicialmente, é através dessas hipóteses que é possível identificar se a startup está caminhando da melhor maneira e se deve perseverar ou pivotar.

6.3 Equipe

Os três processos que estão sendo abordados nesse trabalho defendem a necessidade de uma equipe multidisciplinar. Esse tipo de equipe faz com que não seja necessário buscar informações fora, o que gera agilidade.

Diferente da Startup Enxuta, no Scrum e no XP há alguns papéis definidos na equipe. A diferença entre o Scrum e o XP é que no XP há flexibilidade com relação à existência desses papéis e, desde que seja para um melhor desempenho da equipe não há restrições quanto uma pessoa atuar em dois papéis distintos.

6.4 O que há em comum entre Scrum, Startup Enxuta e XP

Há diversas características comuns entre Scrum, *Startup* Enxuta e o XP, são elas:

- O entendimento da equipe e do cliente deve sempre estar alinhado: em todos os três processos apresentados há uma grande e importante atuação no papel do cliente, é com base no retorno dele que podemos identificar se serão necessárias mudanças no projeto ou não. O Scrum e o XP têm definido o cliente como um de seus papéis e o cliente sabe de sua atuação no decorrer do desenvolvimento do projeto. Na *startup* enxuta a atuação do cliente é um pouco diferenciada, pois acontece do cliente não saber das informações que está gerando para a equipe melhorar ou desistir do desenvolvimento, a equipe obtém as informações do comportamento do cliente além de pesquisas realizadas com eles;
- Trabalham com feedback: é ao término das iterações que são gerados os feedbacks. Na caso da *startup* enxuta, o feedback possui uma importância ainda mais relevante, pois, diferente do Scrum e do XP que têm o cliente para definir exatamente o que será construído, na *startup* ainda não há a certeza de que o cliente irá querer ou precisar do que está em desenvolvimento e são os feedbacks que possibilitam o aprendizado e a tomada de decisão para perseverar no que está sendo feito ou se o melhor é pivotar;
- Trabalham com a entrega contínua: em todos os processos apresentados, um dos principais objetivos é realizar entregas o mais rápido possível para o cliente de modo que o mesmo já consiga validar;
- Flexíveis às mudanças: conforme o tópico acima da entrega contínua o cliente valida e gera *feedback* o mais breve possível. Na validação e

feedback são identificadas mudanças e melhorias que já podem ser priorizadas e podem ser implementadas sem que seja preciso o término do projeto como um todo;

- Podem ser utilizados para outras finalidades que não seja o desenvolvimento de software;
- Equipes de desenvolvimento são multidisciplinares;

Além dos itens mencionados acima, também há em comum entre esses três processos algumas características negativas:

- Resistência das pessoas na utilização das metodologias ágeis;
- Documentação escassa e/ou informal.

7 Considerações finais

Esse capítulo apresenta as conclusões finais sobre o trabalho desenvolvido e possíveis extensões futuras do mesmo.

7.1 Conclusões

As metodologias ou frameworks para desenvolvimento de software tem uma relevância muito grande, pois auxiliam no controle e acompanhamento dos projetos.

As metodologias ou frameworks ágeis surgiram com o intuito principal de tornar o processo de desenvolvimento de softwares mais flexível às mudanças para que na entrega final, o cliente obtenha o que de fato tenha valor e seja de utilidade para ele, dessa forma também não há o risco do sistema ser implantado e já estar defasado. É por esse motivo que um dos objetivos das metodologias ágeis é realizar entregas o mais breve possível ao usuário.

Entre as características e benefícios apresentados durante todo o decorrer do trabalho, podemos perceber que a utilização do desenvolvimento ágil de software tem como prioridade mais alta a satisfação do cliente e, por isso, conta com sua constante colaboração e recebe bem as mudanças de requisitos solicitadas e identificadas, o que se opõe a regra geral presente nas metodologias clássicas.

Através das pesquisas realizadas foi possível entender como deve ser o fluxo de trabalho do Scrum, Startup Enxuta e do XP e entender o nível de dificuldade de cada uma.

Um ponto que vale ressaltar é que para esses processos funcionarem bem e em direção aos seus objetivos, é fundamental que todos os passos propostos sejam seguidos rigorosamente, caso contrário a metodologia ou framework pode ser descaracterizado não gerando os resultados necessários e não atendendo as expectativas depositadas nesses processos.

O objetivo desse trabalho foi apresentar o Scrum, a *Startup* Enxuta e o XP de modo que os leitores tenham adquirido conhecimento o suficiente de todos eles permitindo que façam a escolha pela melhor opção de acordo com seu projeto, empresa e organização.

7.2 Extensões Futuras

Para esse trabalho, temos como extensões futuras as seguintes possibilidades:

- Realizar uma nova comparação das metodologias apresentadas com outras metodologias utilizadas no mercado;
- Realizar um estudo detalhado específico de uma das metodologias apresentadas incluindo sua aplicação em diferentes tipos de projetos e organizações.

8 Referências Bibliográficas

AUDY, J.; PRIKLADNICKI, R.. **Desenvolvimento Distribuído de Software: Desenvolvimento de software com equipes distribuídas**. Rio de Janeiro: Elsevier, 2007. Série Editora Campus – Sociedade Brasileira de Computação.

BECK, K.. **Programação Extrema (XP) Explicada: Acolha as mudanças**. Porto Alegre: Artmed, 2000. 186p.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D.. **The Agile Manifesto**. 2001. Acesso em: 18 mar. 2013.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I.. **UML: Guia do Usuário**. 2ª Edição. Rio de Janeiro: Elsevier, 2006. 479p.

CONNOLY, P.. **Five principles of the lean approach.** 2012. Disponível em: http://www.businesspost.ie/#!story/Home/News/Startup+Advice%3A+Five+principles+of+the+lean+approach/id/19410615-5218-50a0-c302-f2e4d4955746. Acesso em: 10 jun. 2013.

COSTA, R. S.; JARDIM, E.G.M.. **Os cinco passos do pensamento enxuto - NET**. Rio de Janeiro, 2010. Disponível em: http://www.trilhaprojetos.com.br. Acesso em: 05 jun. 2013.

CROLL, A.; YOSKOVITZ, B.. **Lean analytics : use data to build a better startup faster**. 1ª Edição. Sebastopol: Copyright, 2013. 412p.

DICIONÁRIO ONLINE, **Significado de Processo**. 2009. Disponível em: http://www.dicio.com.br/processo. Acesso em: 20 jun. 2013.

GRESSLER, L. A.. **Introdução à pesquisa: projetos e relatórios**. São Paulo: Loyola, 2004, 295p.

HORSTMANN, C.. **Padrões e Projetos Orientados a Objetos**. 2ª Edição. Porto Alegre, Artmed, 2006. 424p.

JONES, D.T.; WOMACK, J. P.. **A mentalidade enxuta nas empresas: elimine o desperdício e crie riqueza**. Elsevier, 2004. 408p.

LARMAN, C.. **Utilizando UML e padrões: Uma introdução à análise e ao projeto orientados a objetos e os desenvolvimento iterativo**. 3ª Edição. Porto Alegre, Artmed, 2005. 699p.

LOBO, E.J.R.. **Curso De Engenharia de Software: Métodos e processos para garantir a qualidade no desenvolvimento de softwares**. São Paulo: Digerati Books, 2008. 112p.

MALHOTRA, N. K.. **Pesquisa de Marketing: Uma Orientação Aplicada**. 4ª Edição. Porto Alegre: Artmed, 2004. 723p.

PRADO, D.. **Maturidade E Sucesso em TI: Relatório Resumido**. 2011. Disponível em: http://www.maturityresearch.com/novosite/2010/downloads/PesquisaMaturidade2010_Relatorio-T.I._VersaoResumida_V3.pdf. Acesso em: 08 abr. 2013.

PRESS, G.. **The Lean Startup ...in 30 Minutes: A Concise Summary of Eric Ries**. 2012. 46p.

PRESSMAN, R.S.. **Engenharia de Software: uma abordagem profissional**. 7ª Edição. Porto Alegre: AMGH, 2011, 780p.

REIS, D. F. **Conceitos básicos sobre Metodologias Ágeis para Desenvolvimento de Software**. DevMidia. Disponível em http://www.devmedia.com.br/conceitos-basicos-sobre-metodologias-ageis-para-desenvolvimento-de-software-metodologias-classicas-x-extreme-programming/10596. Acesso em 02 dez. 2013.

REZENDE, D. A.. **Engenharia de Software e Sistema de Informação**. 3ª Edição. Rio de Janeiro: Brasport, 2005. 316p.

RIES, E.. **O MVP: a ferramenta de experimentação e aprendizado da Startup.** 2010. Disponível em: http://www.manualdastartup.com.br/blog/o-mvp-a-ferramenta-de-experimentacao-e-aprendizado-da-startup. Acesso em: 15 jun. 2013.

RIES, E.. **Principles of Lean Startups, presentation for Maples Investments**. 2008. Disponível em: http://www.startuplessonslearned.com/2008/11/principles-of-lean-startups.html. Acesso em: 11 jun. 2013.

REIS, L. G.. **Produção de Monografia da Teoria à Prática: o método educar pela pesquisa (MEP)**. Brasília: Senac, 2010. 180p.

RIES, E.. A **The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses**. 1ª Edição. New York: Copyright, 2011. 321p.

SCHWABER, K.; SUTHERLAND, J.. **Guia do Scrum**. 2011. Disponível em: http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20Portuguese%20BR.pdf#zoom=100. Acesso em: 29 abr. 2013.

STANDISH. **Extreme Chaos Report 2001**. 2001. Standish Group. Disponível em: http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf Acesso em: 18 de abril de 2013.

STEFFEN, J. B. **O que são essas tais de metodologias Ágeis?** Developer Works IBM, 2012. Disponível em https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_tais_de_metodologias__c3_a1geis?lang=en. Acesso em 02 dez. 2013.

TAKEUCHI, H.; NONAKA, I.. **The New New Product Development Game**. 1986. Disponível em: http://mis.postech.ac.kr/class/MEIE780_AdvMIS/paper/part3/32_The%20new%20product%20development%20game.pdf. Acesso em: 22 abr. 2013.

VERSION, **7th Annual State of Agile Development Survey**, Version One, 2013. Disponível em: http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf. Acesso em: 04 abr. 2013.