Universidade Federal do Rio de Janeiro

Escola Politécnica

MBA em Governança, Projetos e Serviços de TI (MGPS)

Scaled Agile Framework: Aplicação de Framework Para Escalar Práticas Ágeis em Corporações e Projetos de Alta Complexidade

Autor:

_____

Daniel José Chaves Ferreira

Orientador:

_____

Prof. Edilberto Strauss, PH.D.

Examinador (es): _____

Prof. Claudio Luiz Latta de Souza, MSc

_____

Prof. Flávio Luis de Mello, DSc

_____

Prof. Manoel Villas Bôas Júnior, MSc

_____

Prof. Noberto Ribeiro Bellas, MSc

_____

Prof. Paulo Fernando Peixoto da Costa Fazzioni, MSc

MGPS
Maio de 2016

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

AGRADECIMENTO

Agradeço a minha família pelo apoio e aos professores pelo conhecimento passado durante todo o curso. Agradeço ao professor Strauss pela paciência e orientação no decorrer deste trabalho.

RESUMO

Este trabalho apresenta um estudo sobre o SAFE (*Scaled Agile Framework*) como alternativa para apoio aos negócios em corporações que utilizam metodologias ágeis para desenvolvimento dos seus produtos.

O estudo traz um levantamento de informações e conceitos sobre o metodologia *Lean*, bem como sua origem e história até as suas aplicações nos dias atuais. Apresenta também as Metodologias Ágeis, sua origem, suas metodologias mais utilizadas atualmente (Scrum e XP) e finaliza com a origem, conceitos, informações e benefícios do *Scaled Agile Framework* (SAFe) como solução de apoio para projetos complexos que envolvam arquitetura ágil.

Palavras-chave*: Lean*, Metodologias Ágeis, SAFe

ABSTRACT

This paper presents a study on the SAFE (Scaled Agile Framework) as an alternative to support businesses in corporations using agile methodologies for developing their products.

The study provides a survey of information and concepts of Lean methodology and its origin and history to their applications today. It also presents the Agile methodologies its origin , and its methodologies more used (Scrum and XP) and ends with the origin , concepts, information and benefits of the Scaled Agile Framework (SAFe) as a support solution for complex projects involving agile architecture.

Keywords: Lean, Agile Methodology, SAFe

Sumário

1 Introdução 1

1.1 - Motivação 1

1.2 - Justificativa 1

1.3 - Objetivo 2

1.4 - Metodologia. 2

1.5 - Estrutura do Trabalho 2

2 *Lean Manufacturing* (Produção enxuta) 3

2.1 - Origem 3

2.2 - Princípios do *Lean* 4

2.3 - Técnicas *Lean* 6

2.4 - Beneficios do *Lean*. 11

2.5 - Obstáculos para implantação 12

3 Metodologias Ágeis 13

3.1 - História 13

3.2 - Princípios 15

3.3 - SCRUM 16

3.3.1 - Sprint - Ciclo de desenvolvimento Scrum 17

3.3.2 - Papéis Scrum 18

3.3.3 - Artefatos 20

3.3.4 - Utilização 22

3.4 - *EXTREMME PROGRAMMING* (XP). 22
3.4.1 - Princípios 22
3.4.2 - Práticas 23
3.4.3 - Ciclo do XP. 25
3.5 - Desafios 22

4 *Scaled Agile Framework* (SAFe) 28

4.1 - Origem 28
4.2 - Princípios 28
4.2.1 - Ter visão econômica. 29
4.2.2 - Aplicar pensamento sistêmico. 29
4.2.3 - Supor variações e preservar opções. 30
4.2.4 - Incrementar com rapidez, ciclos de aprendizagem integrados. . 30
4.2.5 - Marco de base na avaliação objetiva dos sistemas de trabalho. 32
4.2.6 - Visualizar e limitar os processos, reduzir o tamanho dos lotes
e gerenciar o tamanho das filas 33
4.2.7 - Aplicar cadencia síncrona com planejamento. 34
4.2.8 - Destravar a motivação intrínseca dos trabalhadores. 35
4.2.9 – Descentralizar a tomada de decisão. 35
4.3 - Níveis 35
4.3.1 - Team 37
4.3.2 - Program 38
4.3.3 - Portfólio 41
4.4 - Métricas 43
4.4.1 - M1 - Métricas de iteração 44
4.4.2 - M2 – Auto avaliação do Team Ágil 45

4.4.3 - M3 – ART auto avaliação 45

4.4.4 - M4 – Atualização de relatórios de entrega 46

4.4.5 - M5 – Incremento de programas 46

4.4.5 - M5 – Incremento de programas 46

4.5 - Aplicações 47
.

4.6 - Casos de sucesso 48

4.6.1 - John Deere 48

4.6.2 - Infogain 49

4.6.3 - Accenture Technology 50

5 Conclusão e trabalhos futuros 52

Referências 54

Anexos 56

Lista de Figuras

Figura 1 - Visão geral do ciclo de vida do Scrum 26

Figura 2 - Scrum of Scrums 28

Figura 3 - Burndown Chart 30

Figura 4 - Ciclo de vida do XP 35

Figura 5 - Scaled Agile Framework 45

Figura 6 - Team 46

Figura 7 - Program 48

Figura 8 - Portfólio 50

Figura 9 - Metricas ART 53

Figura 10 - Métricas de iteração 54

Figura 11 - Radar Chart 54

Figura 12 - Release Train PI Burdown 55

Figura 13 - PI Metrics..... 56

Capítulo 1

Introdução

A crescente busca para entregar produtos com qualidade, de forma rápida e com baixo custo vem gerando diversos estudos baseados em metodologias atualmente utilizadas, e que geram valores substanciais para as empresas em que são aplicadas. Dentro deste contexto a metodologia *Lean* vem apoiando e sendo utilizada como modelo de estratégia de negócios para diversas áreas. Neste trabalho é abordado o *Scaled Agile Framework* (SAFe) com a finalidade de uso da metodologia *Lean* em equipes de desenvolvimento Ágil, projetos complexos e o uso em escala de aplicações que por conceito sempre se faz em pequenas proporções.

1.1 – Motivação

Atualmente mais de 90% das empresas adotam técnicas de desenvolvimento ágil de *software* de alguma forma [1].

1.2 – Justificativa

Embora com resultados positivos para maior parte das empresas, algumas não obtêm o mesmo resultado. Neste caso, as equipes trabalham duro, faz muito esforço para agilizar as entregas, mas acabam falhando. Este trabalho explora o SAFe como uma solução alternativa para escalar projetos complexos e que envolvam metodologias ágeis. A proposta é apoiar as empresas que adotam a metodologia ágil a gerenciar seus projetos complexos a diminuir os riscos de forma mais eficaz. Dentre as principais falhas na adoção de uma metodologia ágil estão [5]:

- Filosofia ou a cultura da empresa que conflitam com os princípios do ágil;

- Falta de suporte gerencial para apoiar as mudanças. É preciso desejar as mudanças;

- Falta de experiência ou treinamento insuficiente no novo processo. É preciso tornar-se capaz de trabalhar de maneira ágil;

- Falta de comprometimento da equipe. É preciso reconhecer que há espaço para melhorias e desejá-las.

1.3 – Objetivo

Muitas empresas estão aplicando técnicas ágeis em escala e estão obtendo sucesso em fazê-lo [1]. O SAFe é um *framework* para permitir escalar práticas ágeis em empresas ou grande projetos [2]. O objetivo é propor através do SAFe, um modo de escalar projetos baseados em metodologias ágeis e nos princípios do *Lean.*

1.4 – Metodologia

O trabalho consiste na realização de uma pesquisa bibliográfica onde são realizados levantamentos e estudos de publicações como livros e artigos acerca do tema.

1.5 – Estrutura do Trabalho

Este trabalho é estruturado da seguinte forma:

- **Capítulo 1 – Introdução**: Capítulo traz uma breve descrição sobre o trabalho, e mostram quais são os objetivos, motivações, justificativas, metodologia e a estrutura do trabalho.

- **Capítulo 2 – Princípios *Lean***: Capítulo descreve a história da *Lean Manufacturing*, descrevendo seus valores e princípios.

- **Capítulo 3 – Metodologias Ágeis**: Capítulo descreve a história das metodologias ágeis, premissas, princípios e aplicações.

- Capítulo 4 – *Scaled Agile Framework* (SAFe): Descrição do SAFe, princípios e métodos.

- Capítulo 5 – Conclusões: Considerações finais e as contribuições deste trabalho realizado, finalizando com proposta de trabalhos futuros.

- Capítulo 6 – Referências Bibliográficas – Contém uma lista, em ordem numérica dos autores, dos trabalhos pesquisados e referenciados ao longo deste trabalho.

Capítulo 2

Lean Manufacturing (Produção enxuta)

2.1 Origem

A produção enxuta teve seu início no Japão com a família Toyoda e a empresa Toyota Motor Company. Após ter passado por períodos difíceis no final da década de 1930, quando o governo japonês obrigou a empresa a produzir caminhões militares, com métodos bastante artesanais, a Toyota resolveu ingressar na fabricação em larga escala, porém encontrou uma série de dificuldades. Com o objetivo de formar a base de uma nova forma de pensar sobre a fabricação, logística e desenvolvimento de produtos a Toyota criou o Sistema Toyota de Produção (TPS – *Toyota Production System*).

Atualmente o paradigma da manufatura enxuta é discutido amplamente na literatura. Considera-se aplicar em quase todos os processos de produção, trazendo benefícios de melhora da produtividade, maior valor agregado aos produtos, redução de desperdícios e maior satisfação dos clientes.

A Toyota pioneira no sistema se concentrou em um princípio fundamental do *Lean* que é a eliminação de perdas ao longo do processo de produção. Para ela, se não gerava valor para o cliente foi definido como perda. Dentro desse contexto, destacam-se como perda: atividades desnecessárias para o momento, movimentação, transporte, espera, processamento extra... a indústria automobilística foi palco, mais uma vez, para profundas transformações na produção industrial no ultimo quarto do século XX. Esse novos conceitos de produção referem-se a um conjunto de inovações organizacionais que a Toyota vinha desenvolvendo desde a metade da década de 1950 [6]. Esta nova abordagem criada na Toyota foi mais tarde nomeada de *Lean Production* (Produção enxuta) [7].

A essência da produção enxuta busca identificar e eliminar todos os desperdícios existentes na cadeia de produção, concentrando esforços nas atividades que criam valor para o cliente. De acordo com Womack, Jones (1998), desperdício é “qualquer atividade humana que absorve recursos, mas não cria valor”. Para Ohno (1997), “desperdício se refere a todos os elementos de produção que só aumentam os custos sem agregar valor – por exemplo, excesso de pessoas, de estoques e de equipamentos”.

Alguns dos desperdícios mais comuns encontrados em um sistema de manufatura são:

- Excesso de estoque de matéria prima;

- Excesso de produção, tanto de produto semielaborado como também do produto final;

- Movimentação excessiva, tanto de materiais como também de operadores;

- Fila de espera, aguardando liberação das máquinas para processamento, tanto de materiais como também de operadores;

- Tempo de transporte de matérias primas, produto semielaborado e produto final dos processos;

- Perdas dentro do processo, como exemplo demora no aquecimento das máquinas ou mesmo de processos não necessários;

- Correção de falhas de produção devido à má qualidade ou ao não atendimento aos requisitos do cliente.

A eliminação de todo desperdício existente nos processos das organizações leva a uma maior eficiência. Analisando todas as fontes de desperdícios identificadas anteriormente, não existem dúvidas sobre os resultados que podem ser alcançados com a implementação da manufatura enxuta. O desenvolvimento da compreensão dos gerentes e supervisores em relação ao que é desperdício e as suas causas, é fundamental para o sucesso de uma iniciativa *Lean.*

2.2 Princípios Lean

Os principais princípios do *Lean* são:

a) Valor:

Segundo Jones e Womack (1998), o ponto de partida para o pensamento *Lean* é o valor. O Valor só pode ser definido pelo cliente em termos de produto especifico (um bem ou um serviço ou ambos simultaneamente) que atenda às necessidades do cliente a um preço especifico em um momento especifico;

b) Cadeia de valor:

Todo produto ou serviço possui uma cadeia de valor e sua analise deve mostrar três tipos de ações existentes: atividades com adição de valor,

atividades sem adição de valor, mas necessárias e atividades sem adição de valor;

c) Fluxo:

O uso da equalização da produção da sincronização e fluxo de peças unitárias para acabar com as esperas interprocessos representa um avanço formidável. Assim, esse princípio relata a importância do fluxo continuo, onde as etapas de produção estão organizadas em uma determinada sequência, de maneira que o produto passe para as etapas seguintes sem estoques intermediários ou itens semiacabados;

d) Produção Puxada:

Conforme Jones e Womack (1998), uma produção puxada em termos simples, significa que um processo inicial não deve produzir um bem ou serviço sem que o cliente de um processo posterior o solicite. Dessa forma, a empresa deve puxar o pedido através do cliente ao invés de produzir conforme sua capacidade (empurrar o pedido);

e) Perfeição:

O foco deste princípio é que a eliminação dos desperdícios deve ser uma rotina nas organizações. Dessa maneira, a empresa não deve nunca interromper esforços para realizar melhorias nos processos.

Lean Thinking (ou mentalidade enxuta) é uma filosofia e estratégia de negócios para aumentar a satisfação dos clientes através da melhor utilização dos recursos. A gestão *Lean* procura fornecer, de forma consistente, valor ao cliente com custos mais baixos (PROPÓSITO), identificando e sustentando melhorias nos fluxos de valor primários e secundários (PROCESSOS), por meio de envolvimento das pessoas qualificadas, motivadas e com iniciativa (PESSOAS). O foco da implementação deve estar nas reais necessidades dos negócios e não na simples aplicação da ferramenta *Lean*. [9]

2.3 Técnicas *Lean*

Ao observar os processos de produção, com o foco nas operações que criam valor para o cliente, vários pontos críticos nos processos se tornam aparentes. Para diminuir ou ate mesmo eliminar esses pontos críticos, técnicas são desenvolvidas. Pode ser que não seja necessária uma revolução na empresa, mas é o conhecimento destas e a aplicação delas que podem tornar o processo mais produtivo.

***Just in Time* (JIT):** é o primeiro pilar do sistema Toyota de Produção. Em um processo de fluxo, as partes corretas necessárias a montagem alcançam a linha de produção no momento em que são necessários na quantidade necessária e isenta de defeitos. Segundo Vollman et al. (1997), JIT reúne não só os conceitos e técnicas do planejamento e controle da produção, mas também é uma base filosófica para administração da produção, trazendo benefícios como redução da complexidade do planejamento de materiais, da necessidade de rastreamento no chão de fabrica, dos estoques em processos e da logística de suprimentos.

Autonomação (Jidoka): Segundo pilar do Sistema Toyota de Produção, a autonomação significa automação com um toque humano. São dispositivos instalados nas máquinas que irão impedir a produção de produtos com defeitos. O sistema de conferencia desenvolvido por Sakichi Toyoda, fundador da Toyota, no seu tear auto ativado foi a base do desenvolvimento dos sistemas de autonomação como, por exemplo, os dispositivos de parada automática, dispositivos de segurança e dispositivos a prova de erros. Esses dispositivos permitem que a máquina trabalhe sem a supervisão direta do operador, que passa a supervisionar a produção de diversos equipamentos ao mesmo tempo.

Nivelamento da produção: busca distribuir a produção de forma homogênea ao longo da jornada de trabalho. Em uma linha de montagem de automóveis, o número total de carros a serem produzidos por dia de trabalho. Podemos observar em alguns sistemas nivelados, a produção mais apertada em alguns períodos do mês, implicando em trabalho em horário extraordinário na proximidade da data de entrega do pedido e uma ociosidade após a entrega. “A fim de evitar flutuações na produção... precisamos tentar manter a flutuação na linha de montagem final em zero” (Ohno, 1997).

Sistema puxado: no sistema puxado, o processo final retira as quantidades necessárias do processo precedente num determinado momento e este procedimento é repetido na ordem inversa passando por todos os processos anteriores. Já no sistema "empurrado", o planejamento da produção é feito com base na previsão de demanda e nos estoques disponíveis, produzindo o volume necessário para atender aquela demanda total. O TPS é um sistema puxado. "O Sistema de produção puxado, onde o processo subsequente busca produtos no processo precedente, não sendo este dedicado a apenas um tipo de produto, requererá trocas de ferramentas com maior frequência" (Ohno, 1997).

Kanban: são instruções colocadas num pedaço de papel para comunicar de forma clara e direta informações necessárias ao operador da estação de trabalho, como por exemplo, quantidade a produzir, quantidade retirada, quantidade recebida, código do item, estoque mínimo, estoque atual, local de armazenagem. O Kanban funciona como um pedido de produção para os processos anteriores e apresenta como vantagens: organiza a produção de forma a se produzir apenas o necessário; reduz estoques intermediários e de matéria prima; reduz o tamanho dos depósitos e das equipes dedicadas a controle e movimentação de estoques; facilita o controle gerencial e de supervisão de processos.

Controle visual: são sistemas de informação visual criados para manter baixa ou prevenir a superprodução. Como exemplo tem-se os quadros sinalizadores de parada das linhas de produção, colocados em locais visíveis, que "quando as operações estão normais, a luz verde está ligada; quando um operário deseja ajustar alguma coisa na linha e solicita ajuda, ele acende a luz amarela. Se uma parada na linha for necessária para corrigir um problema, a luz vermelha é acesa" (Ohno, 1997).

Cinco por quês: base cientifica do sistema Toyota, é uma técnica que ajuda a descobrir a raiz do problema e corrigi-lo. Segundo Ohno (1997), os cinco por quês é um método adaptado do hábito de observação de Sakichi Toyoda. Ele passava horas e mais horas observando os teares que as mulheres mais velhas tinham em casa para tecer a mão, buscando melhorias que posteriormente foram introduzidas em seus teares. Ohno (1997) recomenda: "fique na área de produção durante todo dia e observe – eventualmente você

irá descobrir o que deve ser feito... entendendo o que é desperdício... descobrimos maneiras de transformar movimento em trabalho... perguntando-se cinco vezes por quê".

Troca rápida de ferramentas (setup): quando se trata de trabalhar com pequenos lotes ou mesmo sob encomenda, o tempo para troca de produto na linha deve ser extremamente curto, para não comprometer a disponibilidade das máquinas para produção. Segundo Womack e Jones (1998), Taiichi Ohno e seus colaboradores desenvolveram a troca rápida de ferramentas de um produto para o próximo, dimensionando corretamente as máquinas para que as etapas de processamento pudessem ser realizadas imediatamente adjacentes umas das outras, enquanto o objeto em produção era mantido um fluxo continuo.

Balanceamento das linhas de produção: Ohno (1997) destaca como condição básica para o funcionamento do TPS, o estabelecimento de uma sincronia nos processos de produção, processo por processo atuando de forma balanceada, cadenciada, fazendo fluir toda a produção. O desbalanceamento das linhas de produção pode ser observado através do calculo dos tempos de ciclo (tempo necessário para realizar um processo de produção) das operações de produção. O desenvolvimento de um sistema de produção com tempos de ciclo nivelados poderá exigir um novo *lay out*, a redistribuição da carga de trabalho entre os operários da linha de produção e eliminação da movimentação excessiva de materiais.

Manufatura Celular: consiste em agrupar os processos de produção que compõem uma determinada família de produtos, numa célula de produção, com distâncias mínimas entre os postos de trabalho e balanceando-se os tempos de ciclo dos processos, buscando assim, um fluxo de produção próximo do contínuo. Com família de produtos bem definidas, a criação dessas células de produção se mostra mais produtiva que o *lay out* tradicional. Vollmann et al. (1997) destacam:

"*Células tipicamente no formato de U para aumentar a interação entre os trabalhadores e reduzir o manuseio de materiais. Trabalhadores treinados podem operar diversas máquinas. A manufatura celular pode tornar a capacidade de produção mais flexível, permitindo mudanças mais ágeis no mix de produção".*

Manutenção produtiva total: as máquinas devem estar permanentemente em condições operacionais; isto é conseguido através de um conjunto de técnicas de manutenção, onde todos participam, não só a equipe de manutenção, mas também operadores e demais envolvidos no processo de produção. Vollmann et al. (1997) destaca que o foco é aplicar os mesmos cuidados abordados na qualidade do produto para a qualidade dos processos e equipamentos. Ohno (1997) destaca que mesmo uma máquina mais antiga pode dar condições de atender às necessidades da produção. Por isto a manutenção é parte integrante do TPS, evitando problemas nas máquinas, nos processos e na qualidade.

Métodos a prova de falhas: são ferramentas e sistemas de controle desenvolvidos para garantir a qualidade de cada parte produzida, em cada um dos processos, ou seja, a qualidade é avaliada durante sua criação. Com isto se reduz drasticamente o custo da não qualidade. É também conhecido como *poka-yoke* (à prova de defeitos) ou *baka-yoke* (à prova de bobeiras). Ohno (1997) dá como exemplo de dispositivos à prova de erros, a instalação de instrumentos do tipo passa-não-passa, sensores de defeitos que não permitem que a máquina opere e dispositivos para barrar erros de produção em processos anteriores.

Trabalhadores com múltiplas habilidades: a aplicação de técnicas de manufatura enxuta nos processos produtivos, normalmente acarreta mudanças de lay out, do fluxo de produção, na cadência do ritmo de trabalho, bem como também do número de máquinas por operador. Tudo isso poderá exigir que o operário passe a supervisionar mais de uma máquina, e não apenas trabalhar em um único posto de trabalho, em uma determinada máquina. Deverá então ser treinado nas diversas operações que forem necessárias, desenvolvendo assim habilidades e mudando a organização do trabalho de monofuncional para multifuncional.

Trabalho padronizado: Womak e Jones (1998) definem como trabalho padrão: "cada aspecto da tarefa é analisado, otimizados e então executado sempre exatamente da mesma forma, de acordo com um padrão de trabalho". Para a criação de um procedimento padronizado, é importante a vivência de seu criador no chão de fábrica, como observador participante. O procedimento deverá ser de fácil entendimento, devendo ser testado e revisado várias vezes ates da emissão final e do treinamento dos operadores. Ohno (1997)

destaca que a padronização do trabalho não deve ser estabelecida de cima para baixo, mas pelos próprios operários da produção, reduzindo riscos de se padronizar operações erradas.

Mapeamento fluxo de valor: é fundamental para o sucesso da iniciativa lean que gerentes, supervisores e operadores de chão de fábrica enxerguem e compreendam o fluxo de valor, ou seja, aquelas operações que realmente agregam valor aos produtos. Rother e Shook (2003) desenvolveram uma metodologia para facilitar o mapeamento do fluxo de valor, de forma a possibilitar a eliminação dos desperdícios, com consequente criação de valor. Consiste em "seguir a trilha de produção de um produto, desde o consumidor até o fornecedor, e cuidadosamente desenhar uma representação visual de cada processo no fluxo de valor de material e informação", identificando-se os desperdícios que deverão ser tratados.

5S: a organização das áreas de produção é fundamental e o programa 5S se mostra eficiente no atendimento a este quesito. Sujeiras e itens desnecessários são eliminados e todas as ferramentas e componentes tem um local para armazenamento identificado e organizado. O programa se baseia em 5 termos japoneses, que começam com a letra S e relacionados pelo *Lean* Institute Brasil (2003) como: *Seiri* (senso de utilização) descarta os itens desnecessários; *Seiton* (senso de organização) organiza os itens necessários; *Seiso* (senso de limpeza); *Seiketsu* (senso de padronização): padronização dos três primeiros Ss, *Shitsuke* (sendo de autodisciplina): disciplina para manter os outros 4Ss.

Kaizen: termo japonês que significa "melhoria incremental contínua" (Womak e Jones, 1998). São atividades normalmente realizadas por grupos de funcionários da organização, que atuam no sentido de eliminar os desperdícios ou problemas de produção identificados nos processos. A utilização da técnica de mapeamento de fluxo de valor normalmente expõe problemas que necessitam de intervenções, ou seja, os pontos onde um evento *Kaizen* seria apropriado. Neste caso, as equipes vão para as áreas de produção e buscam identificar melhores formas de trabalho, através do desenvolvimento de um lay out, equipamentos de transporte, sistema de estocagem, porém de forma criativa, tentando aproveitar ao máximo os recursos existentes na organização.

Linha de produção flexível: o mercado se apresenta cada vez mais diversificado, com clientes exigindo muitas opções de compra. O desenvolvimento de linhas de produção flexíveis permite o atendimento dos pedidos dos clientes em menor prazo e é fundamental para uma maior eficiência dos sistemas de produção (Ohno, 1997). A flexibilização exige sistemas de troca rápida de ferramentas, manufatura celular, manutenção produtiva total, bem como outras técnicas comentadas anteriormente. Empresas de ponta estão começando a entender os requerimentos necessários para a flexibilidade, em relação ao tipo e volume de produto a produzir (Vollmann et al., 1997).

2.4 Benefícios do *Lean*

As organizações que têm se baseado na filosofia *Lean* na busca por melhorar seus processos, têm tido como resultados principais:

- Redução no prazo de desenvolvimento de produtos;
- Redução do número de pedidos processados em regime de urgência;
- Melhoria da qualidade do produto final;
- Redução do tempo de entrega;
- Aumento na satisfação dos clientes;
- Melhora no relacionamento da área comercial com a área de produção;
- Redução do estoque de produto semielaborado entre os processos e de produto final nos depósitos;
- Redução nos espaços requeridos para estocagem de produtos;
- Redução no tempo de troca de ferramentas para mudança do mix dos produtos;
- Visão clara das capacidades do sistema de produção;
- Redução dos custos operacionais de produção;
- Aumento da segurança do trabalho e da saúde ocupacional;
- Melhora no nível de moral dos trabalhadores.

2.5 Obstáculos para implantação

O Sistema Toyota de produção está sendo amplamente implementado em várias empresas ao redor do mundo, para se atingir um alto grau de desempenho, competitividade, acelerar processos, reduzir perdas e melhorar a qualidade.

O conhecimento das dificuldades para implantação pode determinar um melhor planejamento de implementação e pode ser fator de sucesso ou fracasso e também de resultados temporários ou consistentes.

A maioria das empresas julga não ter as mesmas necessidades que a Toyota, costuma afirmar que o sistema só é aplicável em indústrias automobilísticas. Este tipo de pensamento acaba por criar a cultura da empresa e cultura é difícil de mudar. Qualquer mudança de filosofia em uma empresa parte, necessariamente da alta diretoria. Quando ela não acredita na necessidade de mudança, pior será praticá-la ou implantá-la no chão de fábrica.

É importante avaliar que todo processo de inovação possui fatores inibidores que impedem, dificultam e até mesmo bloqueiam qualquer tentativa de mudança.

A falta de objetivo de uma organização que não tem um planejamento estratégico de médio e longo prazo são as situações comuns, especialmente naquelas que necessitam de faturamento diário para fecharem seu balanço. Produzem de forma contínua, sem se importarem com os desperdícios. Exige-se dos colaboradores mais do que os próprios são capazes de oferecer. O resultado disso é que a organização tem um futuro incerto a cada mês, sempre dependendo do faturamento e todos atuam em função dele. Porém, a necessidade do faturamento leva outro problema: a produção sem qualidade, onde os produtos ruins são despachados a qualquer custo, interessando somente o lucro imediato. Para aqueles que colaboram para que a organização alcance suas metas dentro de um determinado período, algumas organizações oferecem prêmios que estimulam a competição interna e o individualismo, eliminando o espírito de equipe. [26]

O *Lean* é utilizado para criar e definir um produto comercializável. No próximo capitulo veremos como alcançar este objetivo em projetos e desenvolvimento de produtos.

Capítulo 3

Metodologias Ágeis

"Agilidade é a habilidade de criar e responder a mudanças como forma de manter a lucratividade num ambiente turbulento de negócios." [3]

3.1 História

Highsmith[3], cita que em 2001, em um encontro entre 17 Líderes que trabalhavam no contra fluxo dos padrões da indústria de software, foi discutido formas de trabalho objetivando chegar a uma nova metodologia de produção de *software*, que pudesse ser usada por todos eles e em outras empresas, substituindo os métodos tradicionais de desenvolvimento. O grupo chegou ao consenso de que alguns princípios eram determinantes para obtenção de bons resultados. O resultado deste encontro foi a identificação de 12 princípios e a publicação do Manifesto Ágil [4] que é representado com quatro premissas:

- Indivíduos e iterações são mais importantes do que processos e ferramentas;
- Software funcionando é mais importante do que documentação completa;
- Colaboração com cliente é mais importante do que negociação de contratos
- Adaptação a mudanças é mais importante do que seguir o plano inicial.

O Manifesto Ágil, não rejeita processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, simplesmente mostra que eles têm importância secundaria quando comparado com indivíduos e interações, com o software funcionando, com a colaboração com o cliente e as respostas rápidas a mudanças e alterações. [13]

A importância do entendimento dos conceitos do manifesto ágil é necessária para definição de preferencias e não de alternativas para o desenvolvimento de software, fortalecendo o foco na atenção de certos conceitos sem eliminar outros. Sendo assim, para seguir os conceitos ágeis deve-se valorizar mais a certas coisas do que a outras.

Para as premissas citadas acima é necessário o seguinte entendimento:

Indivíduos e interações são mais importantes do que processos e ferramentas, os softwares não são construídos por uma única pessoa, eles são construídos por uma equipe, então elas precisam trabalhar em conjunto (incluindo programadores, testers, projetistas e também o cliente). Processos e ferramentas são importantes, mas não é tão importante quanto trabalhar juntos.

Software funcionando é mais importante que documentação completa, a documentação deve existir para ajudar as pessoas a entender como o sistema foi construído, mas é muito mais fácil entender como funciona vendo o sistema funcionar do que através de diagramas que descrevem o funcionamento ou abstraem o uso.

Colaboração com o cliente é mais importante do que a negociação de contratos, somente o cliente pode dizer o que ele espera do software e normalmente eles não sabem explicar exatamente o que eles esperam e ainda, eles mudam de ideia ao longo do tempo e conforme eles vêm o software funcionando. Ter um contrato é importante para definir responsabilidades e direitos, mas não deve substituir a comunicação. Trabalhos desenvolvidos com sucesso tem constante comunicação com o cliente para entender suas necessidades e ajuda-los a descobrir a melhor forma de expressa-las.

Adaptação à mudança é mais importante que seguir o plano inicial, mudanças é uma realidade no ambiente de negócios e elas acontecem por inúmeras razões: as regras e leis sofrem alterações, as pessoas mudam de ideia e a tecnologia evolui. O software precisa refletir essas mudanças. Um projeto de software certamente deve ter um plano, mas ele deve ser flexível o suficiente para comportar as mudanças quando elas aparecem, senão ele se torna irrelevante.

É importante notar que as metodologias vieram antes do Manifesto. Algumas são citadas nos anos 80, como *Extreme Programming* (XP). A reunião foi feita para proporcionar a troca de ideias, que acabaram chegando em alguns pontos comuns para todos esses métodos.

Estes líderes de diversas áreas de formação, com ponto de vista diferentes sobre modelos e métodos de desenvolvimento de software, criaram o manifesto para encorajar melhores meios de desenvolver software (AMBLER, 2002). As principais ideias foram compiladas para expressar os princípios que guiam as metodologias ágeis.

O Manifesto é apoiado por doze princípios além das premissas.

3.2 Princípios

Os 12 princípios são:

1 – Prioridade é satisfazer o cliente através da entrega continua e adiantada de software com valor agregado;

2 – As mudanças nos requisitos são bem vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagens das mudanças visando vantagem competitiva para o cliente;

3 – Frequentes entregas de software funcionando, de poucas semanas a poucos meses, com preferencia à menor escala de tempo;

4 – As pessoas de negocio e desenvolvedores devem trabalhar diariamente em conjunto por todo projeto;

5 – Os projetos devem ser construídos em torno de indivíduos motivados. Dando o ambiente e o suporte necessário e confiança para fazer o trabalho;

6 – O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face;

7 – Software funcionando é a medida primaria do progresso;

8 – Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;

9 – Continua atenção técnica e bons designs aumentam a agilidade;

10 – Simplicidade para maximizar, a quantidade de trabalho não realizado é essencial;

11 – As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis;

12 – Em intervalos regulares, a equipe deve refletir sobre como tornar-se mais efetiva, e então, ajustar-se de acordo com seu comportamento.

As ideias referentes ao movimento ágil têm sido rapidamente disseminadas pela comunidade de desenvolvimento. Porém, mesmo que os desenvolvedores avaliem de forma positiva, as técnicas adotadas pela metodologia, a decisão final ainda tem que ser tomada pela empresa na qual estão inseridos.

O conjunto de práticas que é o coração das metodologias ágeis pode e está sendo aplicado em empresas e projetos que não se dizem ágil. Um bom exemplo é o método de trabalho da empresa *37signals*, criadora do *Ruby on Rails*, descrito pelos seus fundadores Jason Fried e David H. Hansson no livro Rework (2010). Apesar da *37signals* não adotar nenhum método em especifico, alguns de seus capítulos descrevem práticas adotadas que estão em concordância tanto com Manifesto Ágil como com as práticas de *Scrum* e *Extreme Programming.*

Como o domínio de metodologias ágeis é muito amplo, neste trabalho são tratados os termos de mais relevância e abrangência, ou com foco especifico em Scrum e Extreme Programming, por serem metodologias mais amplamente utilizadas e muitas vezes conjuntamente.

3.3 *SCRUM*

Scrum trabalha com a complexidade de desenvolvimento de softwares através do controle de inspeção, adaptação e visibilidade de requisitos de um processo empírico fazendo uso de uma série de regras e práticas, onde controle não significa controle para criar o que foi previsto e sim controlar o processo para orientar o trabalho em direção a um produto com maior valor agregado possível. [15]

No jogo de *Rugby*, o "*Scrum*" é a forma de reiniciar o jogo após uma falta. Todos os jogadores se posicionam em um bolo humano para competir pela bola. É uma jogada onde um time de oito integrantes trabalha em conjunto para levar a bola adiante no campo do adversário e concretizar seu único objetivo. O *Goal*.

A primeira referência na literatura ao termo "*Scrum*" aplicado ao desenvolvimento de produtos vem do artigo escrito por Takeuchi e Nonaka (1986), onde foram sumarizadas as melhores práticas comuns utilizadas por empresas japonesas inovadoras. A ideia foi posteriormente aprimorada e refinada, por estes autores em outro trabalho (TAKEUCHI; NONAKA, 1995).

O coração do *Scrum* é a iteração. [15] A cada iteração a equipe analisa os requisitos, a tecnologia e suas habilidades e então se dividem para construir e entregar o melhor software possível adaptando-se diariamente conforme surjam as complexidades, dificuldades e surpresas.

3.3.1 *Sprint* – Ciclo de desenvolvimento *Scrum*

Segundo Koscianski (2006), o ciclo de vida do *Scrum* é dividido em três fases:

- **Pré Planejamento (*Pre-game Phase*):** os requisitos são descritos em um documento chamado *Backlog*. A seguir os requisitos são classificados por prioridade, onde é estimado "o esforço" para seu desenvolvimento. Nesta fase inclui a definição dos integrantes da equipe, identificação da necessidade de treinamento, as ferramentas a serem utilizadas, como também uma lista com os prováveis riscos de projeto. A fase é concluída com uma proposta de arquitetura de software. As alterações futuras devem ser descritas no *Backlog*.

- **Desenvolvimento (*Game Phase*):** os riscos previamente identificados devem ser mapeados e acompanhados ao longo do projeto para avaliar o seu impacto. Nesta fase o software é desenvolvido em ciclo interativo (*Sprints*), onde são adicionadas novas funcionalidades. Cada um desses *Sprints* com duração de 2 a 4 semanas são desenvolvidos de forma tradicional (análise, projeto, implementação e testes).

- **Pós-planejamento (*Post-game Phase*):** Nesta fase acontece a integração do software, os testes finais e a documentação do usuário. A equipe se reúne para analisar o estado do projeto e o software atual é apresentado ao cliente.

Para representar o ciclo de vida de um requisito são utilizadas derivações do *Product Backlog*, como *Release Backlog* e *Selected Backlog*. (TEAMSYSTEM; ARAUJO et al, 2005,2006)

A figura abaixo representa uma visão geral do ciclo de vida do *Scrum* com seus artefatos.

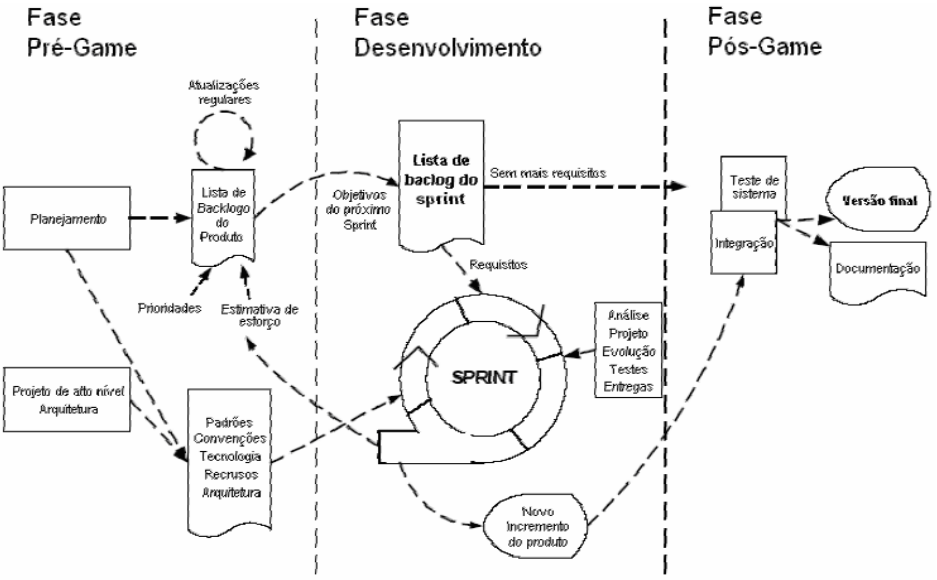

FIGURA 1 – Visão geral do ciclo de vida do Scrum

Fonte: SCHWAUBER, 1995.

A FIG. 1 ilustra todo o ciclo de vida da metodologia *Scrum*, apresentando suas fases, práticas, ferramentas e as iterações entre elas (com entradas e saídas).

O trabalho inicia-se com o levantamento dos requisitos e a elaboração de uma lista de *backlog* do produto, são estabelecidos padrões e convenções referentes a tecnologia, recursos e arquitetura que são utilizadas ao longo do projeto.

Em seguida são estabelecidos objetivos, de acordo com as características do produto, para o próximo Sprint. Após definido o objetivo, são selecionados itens do *backlog* do produto que constituirão o *backlog* do *Sprint*, estes itens são trabalhados no período definido inicialmente (que pode variar de sete a trinta dias) e ao final da iteração, será entregue ao cliente um incremento funcional do produto. Este ciclo se repete até serem concluídos todos os itens que compõe o *backlog* do produto.

3.3.2 Papéis *SCRUM*

Scrum implementa sua estrutura iterativa e incremental através de três papéis: o *Product Owner*, o *Team* e o *Scrum Master*. Toda responsabilidade no projeto é dividida entre esses três papéis. [15]

- Scrum Master

O *Scrum Master* é responsável pelo processo *Scrum*, por ensiná-los a todas as pessoas envolvidas no projeto, por implementá-lo, fazendo dele uma cultura na organização e ainda por garantir que toda a equipe siga as regras e as práticas do *Scrum*. [15]

Ele também protege a equipe assegurando que ela não se comprometa excessivamente com relação aquilo que é capaz de realizar durante um *Sprint*.

O *Scrum Master* atua como facilitador na *Daily Scrum Meeting* e torna-se responsável por remover quaisquer obstáculos que sejam levantados pela equipe durante essas reuniões. [19]

O papel do *Scrum Master* pode ser exercido por qualquer pessoa da equipe, entretanto é tipicamente exercido por um gerente de projeto ou um líder técnico.

- *Scrum Team*

Scrum Team é a equipe de desenvolvimento.

Um *Scrum team* típico tem de seis a dez pessoas auto-organizáveis, auto-gerenciáveis e multifuncionais. [19] Nela, não existe necessariamente uma divisão funcional através de papéis tradicionais, como programador, designer, analista de testes ou arquiteto. Todos no projeto trabalham juntos e são responsáveis por completar o conjunto de trabalho com o qual se comprometem a cada iteração. A equipe não pode ser alterada durante o Sprint.

Quando necessário uma equipe maior no *Scrum* é usando o *Scrum of Scrums*. Cada *Scrum Team* trabalha normalmente, mas cada equipe também contribui com uma pessoa que deverá frequentar o *Scrum of Scrums Meeting* para coordenar o trabalho de múltiplas equipes *Scrum*. Esses encontros são análogos ao *Daily Scrum*, mas não acontecem necessariamente todos os dias. Fazer essa reunião duas ou três vezes por semana tende a ser suficiente na maioria das organizações. [19]

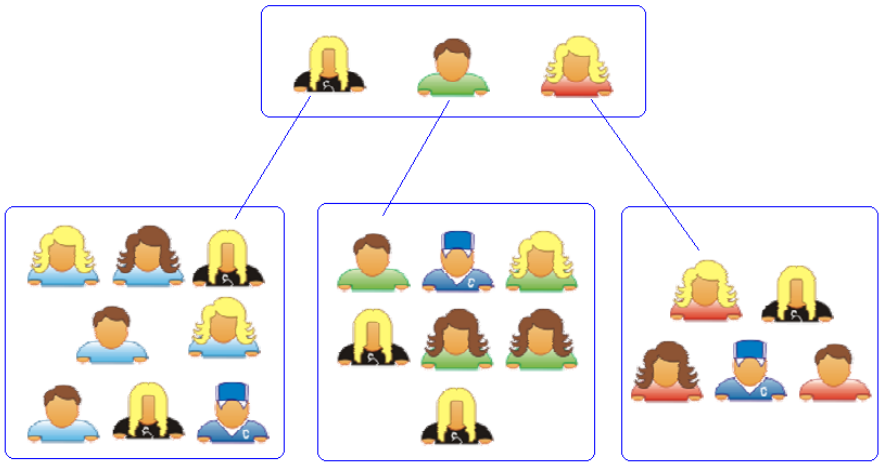

FIGURA 2 – *Scrum of Scrums*

Fonte: www.scrumalliance.org

- Product Owner

O *Product Owner* é responsável por representar os interesses dos *stakeholders* no projeto. O *Product Owner* é a pessoa que define todos os itens de requisito do projeto numa lista chamada *Product Backlog*. Utilizando uma lista ele é responsável por garantir que as funcionalidades que agreguem maior valor ao produto sejam desenvolvidas primeiro. Isto é feito através da frequente priorização do *Product Backlog* para que os itens de maior valor agregado sejam entregues a cada iteração. [15]

3.3.3 Artefatos

- Product Backlog

O *Product Backlog* é uma lista contendo todas as funcionalidades desejadas para um produto. O conteúdo desta lista é definido e priorizado pelo *Product Owner*. O *Product Backlog* é totalmente dinâmico, ele é modificado toda vez que se identifica algo que o produto precisa para ser mais apropriado, competitivo ou proveitoso [15], por isso ele nunca está completo, ele sempre contém os requisitos mais conhecidos e melhor entendidos.

- Sprint Backlog

O Sprint *Backlog* é uma lista de tarefas que o *Scrum Team* se compromete a fazer em um Sprint como potencial incremento do produto entregável [15]. Os itens do *Sprint Backlog* são extraídos do *Product Backlog*, pela equipe, com base nas prioridades definidas pelo *Product Owner* e a percepção da equipe sobre o tempo que será necessário para completar as várias funcionalidades. A quantidade de itens do *Product Backlog* que serão trazidos para o *Sprint Backlog* é definida pelo *Scrum Team* que se compromete a implementá-lo durante o Sprint.

Os itens do *Sprint Backlog* são divididos em tarefas com detalhes suficientes para que possam ser executadas em até 16 horas. [15]

Durante o Sprint, o *Scrum Master* mantem o *Sprint Backlog* atualizando-o para refletir que tarefas são completadas e quanto tempo a equipe acredita que será necessário para completar aquelas que ainda não estão prontas. A estimativa do trabalho que ainda resta a ser feito no Sprint é calculada diariamente e colocada em um gráfico, resultando em um *Sprint Burn Down Chart*. [19]

- Burn Down Chart

O monitoramento do progresso do projeto é realizado através de dois gráficos principais: *Product Burndown Chart* e *Sprint Burndown Chart*. Estes gráficos mostram ao longo do tempo a quantidade de trabalho que ainda resta ser feito, sendo um excelente mecanismo para visualizar a correlação entre a quantidade de trabalho que falta ser feita (em qualquer ponto) e o progresso do time do projeto em reduzir este trabalho. [20]

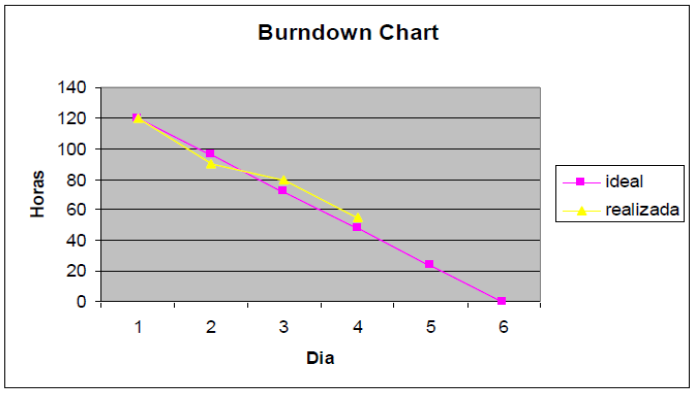

FIGURA 3 – Gráfico que demonstra Burndown Chart

Fonte: www.infoq.com

3.3.4 Utilização

O *Scrum* é uma metodologia destinada a pequenas equipes com menos de dez pessoas. Schwaber e Beedle (2001) sugerem que a equipe seja composta de cinco a nove integrantes, se mais pessoas estiverem envolvidas no projeto, devem-se formar múltiplas Equipes *Scrum*. Assunto que será abordado no capítulo 4. Bem como a utilização de *Framework* para escalar as práticas Ágil.

3.4 *EXTREME PROGRAMMING (XP)*

Segundo Franco [21], a metodologia *Exterme Programming* (XP) surgiu como uma tentativa para solucionar os problemas causados pelos ciclos de desenvolvimento longos dos modelos de desenvolvimento tradicionais. O XP é composto por práticas que se mostraram eficientes nos processos de desenvolvimento de software nas ultimas décadas. Depois de aplicado e obtido sucesso num caso real, o XP foi formalizado através de princípios chaves e doze práticas.

O *Extreme Programming* enfatiza o trabalho em equipe. Os gerentes, clientes e desenvolvedores são todos parceiros iguais em uma equipe colaborativa. Ele implementa um ambiente simples, mas eficaz, que permite as equipes se tornar altamente produtivas. A equipe se auto organiza em torno do problema para resolvê-lo mais eficientemente possível.

3.4.1 Princípios

Os princípios chaves são: comunicação, simplicidade, *feedback* e respeito. Esses são os pilares sobre os quais a metodologia XP é sustentada.

- **Comunicação**: A maioria dos problemas que ocorrem nos projetos invariavelmente tem sua causa associada ao fato de alguém não ter informado alguma coisa muito importante para uma pessoa que precisava saber. Dessa forma, a comunicação é do valor de maior importância no XP.

- **Simplicidade**: A simplicidade não é fácil. Uma das coisas mais difíceis de fazer é não olhar para as coisas que serão necessárias implementar no dia seguinte, na semana seguinte e no mês seguinte. Deve ser implementado apenas aquilo que é necessário e realmente importa ser construído. Isso significa dizer que as pessoas só devem se

preocupar em solucionar hoje os problemas de hoje. A complexidade custa muito caro e tentar prever o futuro é muito difícil. É necessário aguardar o futuro para ver se está certo ou não.

- ***Feedback***: A veracidade dos relatórios do estado atual das atividades é extremamente importante em XP. *Feedback* significa perguntar e aprender com as respostas. A única forma de saber a necessidade do cliente é perguntando a ele. O único modo de saber se um código faz o que ele se destina a fazer é testando-o. Quanto mais cedo se obter o *feedback*, mais tempo se terá disponível para reagir. A metodologia XP fornece um *feedback* rápido e frequente por parte dos seus seguidores.

- **Coragem**: Depois de falar nos três valores, comunicação, simplicidade e *feedback*, é a hora de se esforçar como nunca antes. Se o trabalho não for desempenhado na sua velocidade mais rápido possível, alguém mais o irá fazer, e ele vai lucrar no lugar. A coragem significa tomar as decisões na hora que elas precisam ser tomadas. Se uma funcionalidade não esta funcionando, ela deve ser consertada. Se algum código parece não estar bem escrito, ele deve ser refatorado. Se não será possível entregar tudo o que havia prometido dentro do prazo acordado, o cliente deve ser informado. A coragem é uma virtude difícil de aplicar. Ninguém gosta de estar errado ou quebrar um acordo. O único modo de consertar um engano é admitir que houve o engano e consertá-lo.

Programadores de um time que implementa *Extreme Programming* constantemente se comunicam com seus clientes e colegas programadores [21].

3.4.2 Práticas

O núcleo principal de XP compõe-se de doze práticas, que vão de acordo com os valores e princípios do processo. Percebe-se que algumas dessas práticas também são empregadas em outros processos de software, contudo, em XP as mesmas possuem uma abordagem mais coletiva. As práticas são reladas a seguir [22]:

- **Jogo do planejamento:** o planejamento utiliza-se de casos de uso simplificados (estórias) levantados com a participação da equipe de desenvolvimento (técnico) e do cliente (negócio), estimulados pela figura do gerente, que utiliza como métrica a razão entre o tempo estimado para desenvolver e o tempo realmente gasto (que deve ser menor).

- ***Releases* pequenos:** os releases devem ser de curta duração, estimulando sua frequência e, em decorrência disto, a constante comunicação entre os envolvidos e o melhoramento do sistema.

- **Metáfora:** utilização de uma metáfora (exemplo modelo) que auxilie na compreensão do sistema pelos envolvidos (programadores e clientes), oferecendo uma visão geral sobre o sistema.

- **Projeto Simples:** o projeto deve satisfazer os problemas atuais, implementando as funcionalidades já definidas e dispensando o investimento na resolução prévia de problemas que ainda estão por vir.

- **Testes constantes:** inclui a aplicação de testes para verificação de cada parte produzida pelos programadores (teste de unidade) e testes para verificação do sistema como um todo, junto ao cliente (teste funcional).

- **Refatoramento**: melhoria do projeto do código já definido, por meio da aplicação de uma série de passos, visando uma melhor adaptação do projeto a mudanças.

- **Programação em pares:** o código é produzido por um par de programadores em um mesmo computador, os quais se alternam nos papéis de codificador, que elabora os algoritmos e a lógica de programação, e de observador, que pensa em melhorar, simplificar e corrigir o código produzido.

- **Propriedade coletiva do software:** equipe trabalha de forma unida, prevalecendo a busca pela qualidade e melhoria do código e do sistema como um todo.

- **Integração continua:** integrações podem ser realizadas a qualquer tempo, desde que não existam erros nas novas funcionalidades.

- **Semana de quarenta horas**: existe a possibilidade de que a carga horária da equipe exceda quarenta semanais, observando, porém, a carga tolerável que não gere insatisfação entre seus membros.

- **Cliente no local:** requer a inclusão na equipe de uma pessoa da parte do cliente, que domine ou, ao menos, detenha conceitos principais sobre o negócio ao qual o sistema se destina, para auxiliar na definição das funcionalidades e na utilização do software.

- **Padrões de codificação:** a utilização de padrões é imprescindível para garantir o desenvolvimento em equipe, já que todos possuem acesso e poder de alteração sobre o código.

As práticas do XP se relacionam entre si. A combinação dessas práticas possibilita a equipe poupar esforços com a concepção (projeto) do sistema, mantendo simultaneamente a flexibilidade diante de mudanças de requisitos.

Segundo BECK [22], a gerencia de projetos com XP envolve basicamente dois papéis, o treinador (*coach*), que se preocupa principalmente com a execução técnica e a evolução do processo; e o rastreador (*tracker*), que coleta métricas sobre o que está sendo desenvolvido e verifica possíveis divergências com métricas estabelecidas. A equipe ainda é composta pelos seguintes papéis: programador, que ocupa o papel principal, analisando, projetando, testando, codificando e integrando o sistema, a fim de produzir rapidamente código de alta qualidade; cliente, que escolhe o que irá agregar valor ao seu negócio, o que deve ter prioridade de desenvolvimento e participa na definição dos testes funcionais; testador, que ajuda na definição e escrita dos testes funcionais; consultor, que possui elevado nível de conhecimento em determinada tecnologia ou assunto que não é de compreensão dos envolvidos no projeto.

3.4.3 Ciclo do XP

No clico de vida da metodologia XP, definido por BECK [22], o desenvolvimento de software, encontra-se organizado nas seguintes fases: Exportação, Planejamento, Iterações do Release, Produção, Manutenção e Morte.

Um projeto com XP tem início na fase de Exportação. Nesta fase o cliente escreve as primeiras estórias sobre o sistema, enquanto a equipe prepara uma arquitetura prototipal a partir da qual o sistema será desenvolvido e busca estabelecer uma metáfora que represente o funcionamento do sistema, fornecendo uma possível para o problema.

Os requisitos extraídos das estórias escritas pelo cliente e a metáfora desenvolvida pela equipe serão utilizados como entrada para a fase de Planejamento. Nela a equipe estabelece estimativas de desenvolvimento para estórias, com base na experiência de sistemas anteriores e na arquitetura prototipal considerada, o cliente estabelece prioridades, considerando as características de seu negocio. São definidas as estórias que serão implementadas em cada release.

Tendo como ponto de partida do release, a fase de Iterações do Release será iniciada. As estórias pertencentes ao release são implementadas por meio de iterações, com duração de média de uma a duas semanas. Realiza-se o projeto, a codificação e a refatoração em cada iteração. Durante a codificação das funcionalidades, são realizados testes de unidade para cada componente produzido. Ao final de cada iteração são realizados testes de aceitação (ou testes funcionais), considerando cenários de testes

produzidos pelos clientes, com auxílio da equipe, a partir das estórias. Parte-se então para próxima iteração, ocasião em que também serão corrigidos possíveis problemas detectados nos testes de aceitação.

Durante o desenvolvimento das iterações as estórias poderão ser modificadas, desmembradas, excluídas ou surgirem novas estórias, sendo aplicados os procedimentos de planejamento do release (estimativa e prioridade), acrescentando as alterações ao plano do release. As alterações nas estórias terão reflexos na velocidade do projeto.

Na fase de produção, as funcionalidades desenvolvidas durante as iterações passam pela aprovação do cliente e o release é colocado em operação. Parte-se então para o desenvolvimento de um novo release.

A fase de Manutenção envolve as fases de planejamento, iterações do release e produção. Procura-se produzir novas funcionalidades, com o sistema existente funcionando, incorporando novas pessoas a equipe e melhorando o código. A Morte consiste no término do projeto pela satisfação total do cliente ou pela inviabilidade técnica ou econômica de se continuar o desenvolvimento. A Figura 4 apresenta elementos gerais e as fases presentes no ciclo de vida do XP.

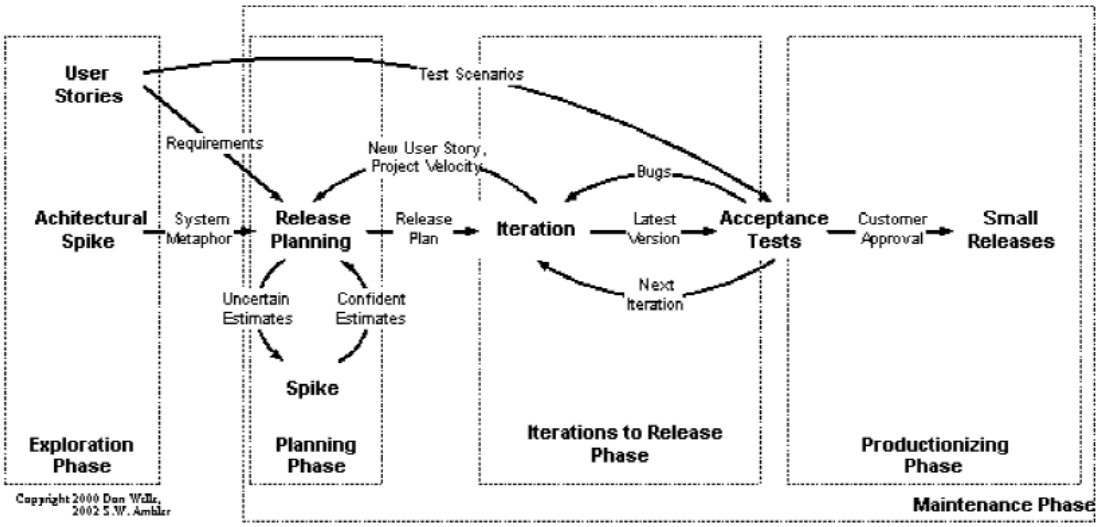

FIGURA 4 Ciclo de vida do XP

Fonte: http://www.semeru.com.br/blog/tag/frameworks-ageis/

A aposta do XP é que é melhor fazer algo simples hoje e pagar um pouco mais amanhã para fazer modificações necessárias do que implementar algo complicado hoje que talvez não venha a ser usado sempre considerando que requisitos são mutáveis.

Uma característica que podemos destacar nesta metodologia é a programação em pares.

3.5 Desafios

O desafio é como usar essas metodologias ágeis em grandes empresas e equipes, uma vez que normalmente essas metodologias são baseadas em equipes pequenas. Neste caso, pelo menos é necessário resolver os problemas de comunicação internos na equipe, uma vez que é comum em grandes empresas os funcionários estarem separados geograficamente.

Apesar do interesse crescente no uso das metodologias ágeis, ainda faltam casos de sucesso de seu uso em projetos grandes e críticos. Quanto mais organizações usem as metodologias ágeis, melhores serão os resultados empíricos em termos e vantagens, desvantagens, riscos e procedimentos para sua adoção nas organizações. Mesmo assim, os resultados iniciais em termos de qualidade, confiança, datas de entregas e custo são promissores. No próximo capitulo, será tratada uma ferramenta de apoio para escalar as metodologias ágeis.

Capítulo 4

Scaled Agile Framework (SAFe)

"O tamanho da empresa é claro que importa. Você provavelmente não poderia executar um projeto XP (*Extreme Programming*) ou *Scrum*, com uma centena de programadores, nem com 50. 20, provavelmente. 10 é definitivamente factível..." – Ken Beck. "*Extreme Programming Explained: Embrance Change,"1st Edition, [Publication date: 2000]*

4.1 Origem

O SAFe criado por Dean Leffingwell em 2009 [5], apoia as grandes organizações a se mover em direção a uma forma mais ágil de se trabalhar. Por grandes organizações, entende-se mais de mil pessoas em TI, e mais de duzentas e cinquenta no desenvolvimento de software e ainda sim ser eficaz.

Com base nos princípios imutáveis do desenvolvimento ágil, o pensamento sistêmico *Lean* e no fluxo de desenvolvimento do produto, o SAFE vem sendo adotado por centena das maiores empresas do mundo, em todos os segmentos da indústria. A adoção do SAFE oferece benefícios substanciais ao negócio em tempo de chegada ao mercado mais rápido e de maior qualidade.

4.2 Princípios

O SAFe é baseado em nove princípios *Lean* e Ágil,[11] são eles:

1 – Ter Visão econômica;

2 – Aplicar o pensamento sistêmico;

3 – Supor variações e preservar opções;

4 – Incrementar com rapidez, ciclos de aprendizagem integrados;

5 – Marcos de base na avaliação objetiva dos sistemas de trabalho;

6 – Visualizar e limitar os processos de trabalho, reduzir o tamanho dos lotes e gerenciar o tamanho das filas;

7 – Aplicar cadência, sincronia com o planejamento de domínio cruzado;

8 – Destravar a motivação intrínseca dos trabalhadores do conhecimento

9 – Descentralizar a tomada de decisão.

Estes princípios representam lições aprendidas com aplicações ágeis, práticas *Lean* e SAFe a mais de dez anos em centenas de contextos.

4.2.1 Ter visão econômica

Com o objetivo de alcançar as metas *Lean* de forma sustentável no menor tempo, com a melhor qualidade e valor para as pessoas e sociedade, requer uma compreensão fundamental da economia na construção de um sistema.

Sem este entendimento, mesmo um sistema tecnicamente competente pode custar muito para desenvolver, levar muito tempo para entregar, ou ter os custos de fabricação ou de funcionamento que não possuem um valor economicamente viável.

Com está finalidade, os sistemas construídos são apresentados em toda cadeia de liderança, gerência e operações. Todo o entendimento dos impactos econômicos serão decididos e definidos por eles. Tradicionalmente, as restrições econômicas nas atividades de construir um sistema são conhecidas por poucos, aqueles que tomam a decisão de mercado, autoridades que tem o entendimento do negócio, mercado e planejamento do cliente. Entretanto, centralizando tal conhecimento nas operações, significa que as decisões cotidianas são: a) Tomadas sem entendimentos, ou b) escalado para aqueles que têm o conhecimento.

A primeira escolha otimiza pouco os resultados econômicos. A segunda aumenta atrasos no valor de entrega, o que terá o mesmo efeito no final.

Os métodos *Lean* tratam o impacto econômico, e constantemente se esforçam para ter certeza que a cada dia as decisões econômicas são tomadas no contexto adequado.

4.2.2 Aplicar o pensamento sistêmico

Segundo Deming (1994), um dos mais importantes estudiosos de sistemas, constantemente focado na visão ampliada de problemas e desafios enfrentados pelas pessoas que produzem sistemas de todos os tipos, sistemas de manufatura, sistemas sociais, sistemas de gerenciamento e governamentais. Umas de suas conclusões foi o entendimento de que os problemas enfrentados no local de trabalho foram impulsionados

por uma série de interações complexas que ocorrem dentro dos sistemas que a operação costuma fazer durante seu trabalho.

Ampliar a visão é a chave para desvendar os mistérios do sistema, para a melhoria do processo de desenvolvimento, da qualidade dos produtos e serviços que são o objetivo do sistema.

4.2.3 Supor variações e preservar opções

Criadores de sistemas tendem a ter inclinações naturais de tentar reduzir as mudanças. Embora seja verdade que a mudança pode conduzir a maus resultados, o caso contrário, também pode ser verdadeiro.

A mudança não é propriamente boa ou ruim. Pelo contrário, é a economia associada com o tempo e o tipo de mudança que vai determinar os resultados. Um foco em eliminar a mudança muito cedo, perpetua uma cultura de prevenção de riscos, em que as pessoas não podem errar e ganhar experiência aprendendo que não funciona. Diferente do entendimento geral das intenções do sistema, no *Lean,* se reconhece que muito pouco se sabe realmente no inicio do projeto. Entretanto, nos projetos praticados atualmente tendem a conduzir os desenvolvedores a uma e única opção – um ponto de solução potencial – e em seguida modificar o projeto até que, eventualmente encontra-se a finalidade do sistema. Essa abordagem pode ser realizada, desde que não se escolha o ponto de partida errado; então iterações subsequentes para refinar essa solução pode ser muito demorado e levar a não otimização do projeto.

Neste processo, o projeto fica com o tempo maior para as opções, podendo convergir quando necessário, e produzir resultados técnicos e econômicos melhores.

4.2.4 Incrementar com rapidez, ciclos de aprendizagem integrados

No modelo tradicional, com ciclos fechados, o custo do investimento começa imediatamente e se acumula até que a solução seja entregue. Muitas vezes, há pouco ou nenhum valor real entregue até todos os recursos da solução estejam disponíveis [11]. Durante o ciclo de desenvolvimento é difícil obter qualquer *feedback* significativo porque o processo não permite. A solução não é implementada de tal forma que o cliente consiga avaliar os incrementos a cada ciclo.

O risco permanece no projeto até a data limite de entrega e até mesmo na utilização inicial. O processo fica sujeito a erros e problemas e normalmente resultam em perda de confiança entre as partes, cliente e empresa. Em uma tentativa de ajuste a empresa e o cliente ainda se esforçam para definir os requisitos e selecionar o que ficou de melhor na entrega.

As soluções *Lean*, abordam o problema diferente. Ao invés de escolher um único requisito padrão assumido no início do projeto que é menos oneroso e vai proporcionar uma adequação no final do projeto, elas utilizam uma gama de requisitos e opções, (Principio 3 - Supor variações e preservar opções), e produzem uma solução de forma incremental em uma série de curto espaço de tempo chamado de Time-boxes. Cada Time-box resulta um incremento de um "pedaço" da solução que pode ser avaliado pela empresa e o cliente. Time-boxes subsequentes tem como base os incrementos anteriores e a solução evolui até que seja liberada.

Os conhecimento adquiridos a partir dos pontos de integração não é unicamente com a finalidade de estabelecer a viabilidade técnica, muitos pontos de integração podem servir como soluções mínimas viáveis ou protótipos para serem avaliados pelo mercado, estabelecendo usabilidade e ganhando *feedbacks* objetivos dos clientes. Sempre que necessário esses pontos de feedbacks rápidos permitem que as empresas possam mudar a direção para um curso alternativo, que atenderá melhor as necessidades dos clientes.

Pontos de integração servem como mecanismo primário para controlar as mudanças no desenvolvimento da solução. Quanto maior a frequência desses pontos, mais rápido será o aprendizado. No desenvolvimento de sistemas complexos, pontos de integração são usados para garantir que cada elemento está cumprindo sua responsabilidade, contribuindo para a intenção da solução. Estes pontos locais devem ser mais integrados no próximo nível superior do sistema. Quanto maior for o sistema, maior será a quantidade de níveis de integração. Todas as partes interessadas precisam entender que quando esses pontos de integração demorar para acontecer, o projeto pode ser comprometido, mas mesmo assim o conhecimento em tempo útil ajuda a facilitar os ajustes necessários ao escopo, abordagem técnica, custo ou tempo de entrega necessário para obter o controle de projetos para expectativas revisadas.

4.2.5 Marcos de base na avaliação objetiva dos sistemas de trabalho

O desenvolvimento de grandes sistemas atualmente exige um investimento substancial que pode chegar a milhões. Juntos, empresa e clientes possuem uma responsabilidade fiduciária para garantir que o investimento em novas soluções vai entregar o benefício econômico necessário. Caso contrário não há nenhuma razão para se fazer o investimento [11].

Claramente, as partes interessadas devem colaborar de modo a ajudar o benefício econômico durante todo o processo de desenvolvimento e não apenas achar que no tudo ficará bem no final. Para enfrentar esse desafio, a indústria evoluiu a um processo (cachoeira) fechado em fases de desenvolvimento, em que o progresso é medido e controlado através de uma série de metas especificas.

O controle é feito através de marcos que seguem o processo aparentemente lógico e sequencial de descoberta, de requisitos, projeto, implementação, teste e entrega. O que nem sempre funciona quando se têm muitos marcos.

As causas deste problema é a falta de reconhecimento de pontos críticos como: centralização de exigências e decisões, escolhas precipitadas, mudanças, demora na tomada de decisões...

Ao longo do desenvolvimento, o sistema é construído em incrementos, cada um dos quais é um ponto de integração que demonstra alguns indícios quanto à viabilidade da corrente solução em processo. Além disso, isto é feito por rotina, em uma cadência, que fornece a disciplina necessária para assegurar a disponibilidade periódica e de avaliação, bem como os limites de tempo predeterminados que podem ser utilizados para entrar em colapso o campo de opções desejáveis.

O que realmente é medido nesses pontos de integração está sujeito à natureza e ao tipo de sistema que está sendo construído, mas o sistema pode ser medido e avaliado pelas partes interessadas relevantes com frequência, e durante todo o ciclo de desenvolvimento da solução. Isso fornece a governança financeira, técnica e de objetivos de diminuir o desperdício para assegurar que o investimento contínuo vai produzir um retorno proporcional.

4.2.6 Visualizar e limitar os processos de trabalho, reduzir o tamanho dos lotes e gerenciar o tamanho das filas

Sobrecarga das equipes e soluções com mais tempo de desenvolvimento é um problema comum. Isso gera uma redução do foco em qualquer tarefa, reduz a produtividade e rendimento e aumenta o tempo de espera para novas funcionalidades.

O primeiro passo é fazer o processo de trabalho atual visível para todos os interessados. Esta visualização mostra a quantidade total de trabalho em que cada etapa e também serve como um processo inicial de diagnóstico, mostrando os pontos de estrangulamento atuais.

Próximo passo é começar a equilibrar a quantidade de trabalho em processo contra a capacidade de desenvolvimento disponível. Quando qualquer passo atinge seu limite de processo de trabalho, nenhum novo trabalho deverá ser levado adiante. No entanto, o que limita os processos de trabalho requer conhecimento, disciplina e empenho, pois pode até parecer contra intuitivo para aqueles que acreditam que quanto mais trabalho, mais se entrega. Até certo ponto isso é verdade, a partir desse ponto torna-se turbulento e o fluxo diminui. Não há nenhum substituto para gestão eficaz de processos de trabalho. [11]

Outra forma de reduzir o processo de trabalho é melhorar o fluxo e diminuir os tamanhos dos lotes, sejam requisitos, projetos, códigos, testes e outros itens de trabalho que se movem através da solução. Simplesmente, pequenos lotes passam pela solução mais rápido e com menos mudanças, o que promove a aprendizagem mais rápida. O tamanho do lote depende tanto do custo de retenção (o custo de retorno e valor atrasado) e o custo de transição (custo de implementação e teste do lote). Para economizar na manipulação de lotes menores, aumenta-se o rendimento no foco principal e coloca a redução de custo de transição associado a qualquer lote. Isto é essencial para o pensamento sistêmico (Principio 2), e um elemento critico na otimização a longo prazo.

O último elemento na realização do fluxo através da gestão e geralmente reduzir as filas. Para desenvolvimento da solução, isso significa que quanto mais tempo na fila de trabalho, aguardando para ser implementado pela equipe, quanto maior tempo de espera, não importa o quanto eficiente é a equipe, se for necessário um serviço mais rápido, a fila deve ser reduzida e aumentada a velocidade de processamento. Enquanto aumentar a velocidade de processamento é um objetivo consistente e comum, a maneira

mais rápida para reduzir o tempo de espera é reduzir a fila. Isto é realizado, mantendo *backlogs* curtos.

Reduzir o comprimento das filas, diminui atrasos, reduz o desperdício e aumenta a previsibilidade dos resultados.

4.2.7 Aplicar cadência sincronia com o planejamento de domínio cruzado

Cadencia fornece um padrão rítmico, o batimento cardíaco confiável do processo, transforma eventos imprevisíveis em previsíveis e tem muitos benefícios: tempo de espera previsível, facilita o planejamento e fornece uma função de força e reduz custos de transição de eventos chave, incluindo o planejamento, integração, demonstrações, *feedback* e retrospectivas.

De todos os eventos que ocorrem o planejamento de domínio cruzado é o mais crítico. Periodicamente todas as partes interessadas se reúnem para o planejamento de domínio cruzado e sincronização. Este evento serve como ponto de apoio em torno do qual todos os outros eventos operam. Ele também serve como a exposição plenária do verdadeiro conhecimento atual. Serve para três propósitos principais:

- Avaliar do estado atual da solução
- Realinhar todas as partes interessadas a uma visão técnica e comercial comum
- Planejar e se comprometer com próximo incremento da solução.

O desenvolvimento de sistemas em larga escala é, fundamentalmente, uma atividade social e este evento de planejamento fornece uma oportunidade contínua para construir e melhorar o *networking.*

Não existe uma cura para incerteza inerente de desenvolvimento de soluções, se houvesse, certamente seria pior que a doença. No entanto, a aplicação de cadência, sincronização e planejamento de domínio cruzado, fornece sistemas *Lean*, com as ferramentas que precisam para operar na zona de segurança.

4.2.8 Destravar a motivação intrínseca dos trabalhadores do conhecimento

Os líderes *Lean*-ágil operam dentro do relativamente novo, mas básico. A verdadeira gestão dos trabalhadores do conhecimento é um oximoro. Segundo Drucker [28], "trabalhadores do conhecimento são indivíduos que sabem mais sobre o trabalho que eles realizam que os seus patrões". Nesse contexto, como pode qualquer gestor tentar gerenciar, ou até mesmo coordenar as atividades técnicas daqueles que são infinitamente mais capazes de definir as atividades necessárias para cumprir a missão do que elas?

Na verdade, eles não podem. Em vez disso, o que eles podem fazer é desbloquear a motivação intrínseca dos trabalhadores do conhecimento.

4.2.9 Descentralizar a tomada de decisão

No *Lean*, o objetivo é simples: Para fornecer valor sustentável no menor tempo, rápido, a tomada de decisão deve ser descentralizada. Qualquer decisão que deve ser escalada para níveis mais elevados de autoridade introduz um atraso e pode diminuir a fidelidade da decisão, devido à falta de contexto local, além de mudanças padrões de fatos que ocorrem durante o tempo de espera. A tomada de decisão descentralizada reduz atrasos, melhora o fluxo de desenvolvimento do produto, permite um *feedback* mais rápido além de mais soluções inovadoras.

No entanto, isso não quer dizer que todas as decisões precisam ser descentralizadas. Algumas decisões são estratégicas, críticas e de grande alcance para serem centralizadas, estas devem ser feitas pela administração.

Se para a solução tenham de ser tomadas os dois tipo de decisão, a criação de um quadro de tomada de decisão é um passo fundamental para garantir o fluxo de valor para o cliente.

4.3 Níveis

O SAFe é estruturado em três níveis: *Team, Program e Portfólio*. Na base da estrutura está o *Team*, e aqui o SAFe será utilizado para organizar os times, de acordo com seus talentos e capacidades. No nível seguinte, *Program*, há a integração de todo o trabalho realizado por diferentes times, que deve ser sincronizado com as outras áreas.

No geral isso acontece por meio de eventos, ou cerimonias, que podem ser de planejamento, de acompanhamento periódico ou de revisão. Por fim, o nível de Portfólio, onde o framework permite fazer a gestão das demandas organizacionais. Aqui, isso é feito de maneira ágil, buscando descobrir gargalos operacionais da empresa e, principalmente orientando as iniciativas para o que é mais importante, para gerar mais valor econômico para a operação. Abaixo a figura 5 mostra todas as fases.

Chamada de "*Big Picture*", ela representa todo o SAFE em uma única imagem. Neste trabalho, serão detalhados os níveis, cada um dos papéis, as equipes, atividades e artefatos necessários para dimensionar essas práticas para a empresa.

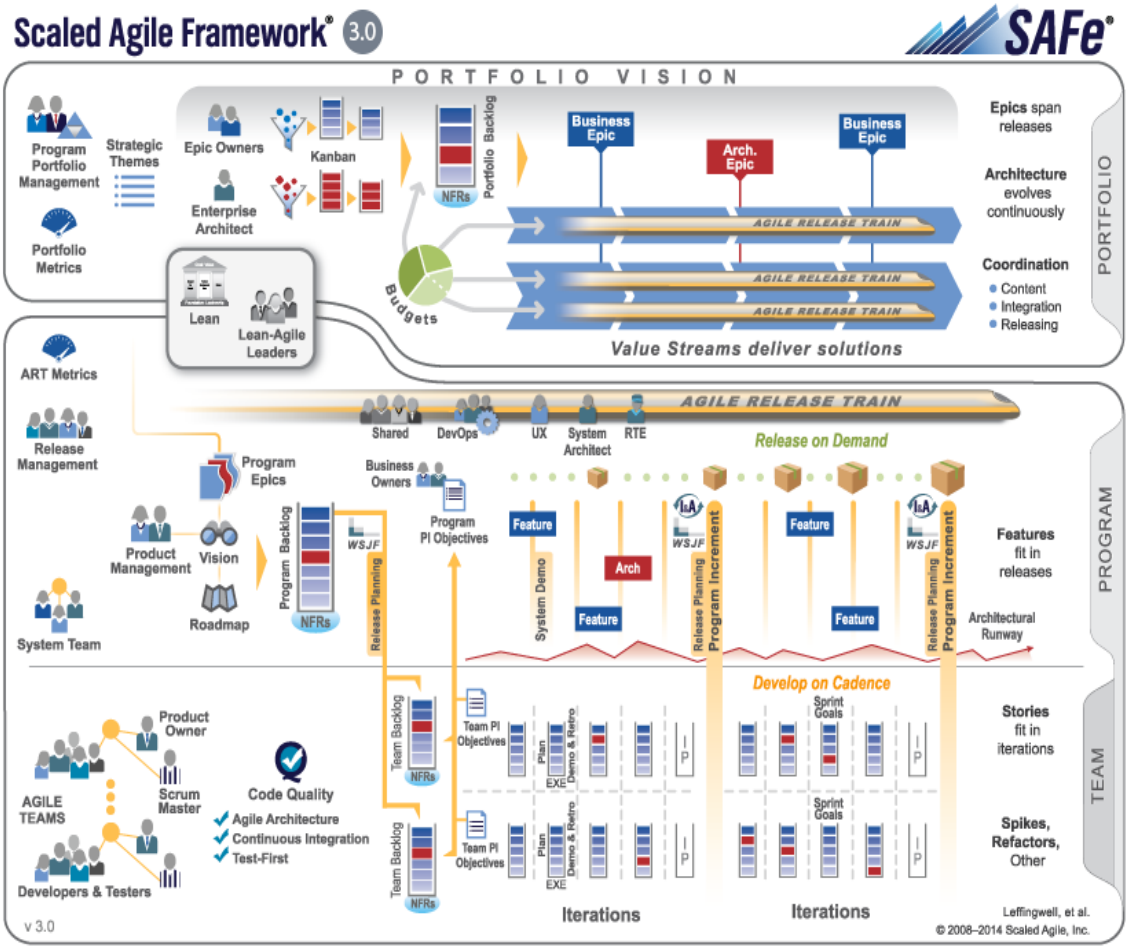

FIGURA 5 – Scaled Agile Framework – SAFe

Fonte: http://www.scaledagileframework.com/

4.3.1 Team

O níveis Team SAFe, Program e Portfolio, fornecem abstrações organizacionais com papéis e práticas destinadas a orientar a agilidade em grandes empresas. No entanto, o nível Team, que fornece uma organização, artefato, papel e modelo de processo para as atividades de equipes ágeis é realmente uma partição transparente do nível *Program*, já que não existem ARTs (*Agile Release Train* – Equipes que sincronizam as atividades entre os níveis *Program* e Portfolio através de reuniões de alinhamento).

Cada Team ágil é responsável pelas histórias dos usuários, definindo, construindo e testando através de S*prints*, interações de comprimento fixo, usando uma cadência comum e sincronizada para alinhar com outros Team e fornecendo integrações regulares e forçadas para o sistema (o sistema também tem que ser ágil assim como o *Team*). Os *Teams* usam práticas de gerenciamento de projetos *Scrum*, e *Scrumban* e, um pouco de XP, pratica técnicas estendidas para oferecer frequentemente código de alta qualidade.

FIGURA 6: Team

Fonte: http://www.scaledagileframework.com/

Equipes ágeis é o motor do desenvolvimento de software. Cada equipe tem entre 5 a 9 membros e inclui as funções necessárias para criar um incremento de qualidade. As funções incluem um *Scrum Master*, *Product Owner*, desenvolvedores e testadores dedicados. Todos precisam entregar valor. A equipe é apoiada por arquitetos de sistemas, especialistas em tecnologia, um time de sistemas, possuem recursos compartilhados e são totalmente capazes de definir, desenvolver, testar e entregar funcionando os softwares para linha de base do sistema, pelo menos, a cada duas semanas;

Iterações: as equipes desenvolvem software em curto espaço de tempo, geralmente duas semanas, sempre em conformidade com o acordado nas metas do *Sprint*.

Cada iteração representa um incremento valioso de novas funcionalidades, realizado através de um modelo padrão repetindo constantemente: Planejar a iteração, comprometer algumas funcionalidades, criar e testar histórias com foco na qualidade. Demonstrar a nova funcionalidade ás partes interessadas, realizar uma retrospectiva e voltar para iteração seguinte. No final de cada Sprint as equipes demonstram o sistema.

Uma série de iterações é usada para agregar valor. O número de iterações por programa de incremento e a decisão quando realmente liberar um incremento é deixado para equipe ART. Em muitos casos essas liberações acontecem com muito mais frequência do que nos limites de planejamento e inovação e de fato os programas podem e devem ser desenvolvidos em cadência. Liberação por demanda.

Histórias do usuário: transporta os requisitos do cliente através do fluxo de valor em código e implementação. O *backlog* da equipe consiste de histórias do usuário, próximas funcionalidades e quaisquer outros itens do *backlog* que a equipe tenha identificado para implementação. A maioria desses itens são identificados durante o planejamento da entrega, quando o gerenciamento do produto apresenta a visão, roteiro e *backlog* programado. Identificação, manutenção, prioridade, programação, elaboração, implementação, teste e aceitação das histórias do usuário e o processo de gerenciamento primário são premissas de trabalho nas empresas que trabalham com ágil.

4.3.2 *Program*

No nível *Program* é que o financiamento para as pessoas e outros recursos é aplicado em algumas importantes áreas. Nas empresas SAFe, a maioria dos programas são de um para um, seguindo os padrões de financiamento, governança e desenvolvimento incremental da Agile Release Train (ART) para entregar a sua parte no fluxo de valor. Neste caso, as ARTs são geralmente muito longas e, portanto tem uma estrutura mais persistente na missão que um programa tradicional que mais classicamente tem uma data de início e fim.

O ART é quem prevê a duração dos times, normalmente consiste de 50-125 pessoas, que servem como mecanismo de entrega de valor no nível Program. São organizados em torno de fluxos significativos de valor das empresas, períodos longos dos fluxos de produtos, sistemas, soluções ou serviços em que uma estratégia de desenvolvimento e entrega otimizados criam maiores benefícios econômicos.

No nível TEAM, regras e atividades são organizadas em torno do ART. Times Ágil proporcionam uma série continua de incrementos com valor. Desenvolvedores de software trabalham em cadência padrão, mas a equipe ART pode lançar a qualquer momento uma nova forma de se trabalhar. No que diz respeito ao desenvolvimento de cadencia, o SAFe planeja sessões de lançamentos periódicos seguido por quatro iterações de desenvolvimento e conclui com uma iteração de inovação e planejamento. O SAFe chama este período de incremento de programa.

Nas empresas que estão em transição para o SAFe, no caso de práticas de desenvolvimento de fornecedores estão fora do controle da empresa, tais programas tradicionais ainda podem existir, então no SAFe o gerenciamento de portfólio de programas fornece um modelo de governança que pode ser aplicado em gerenciar ARTs juntamente com modelos tradicionais.

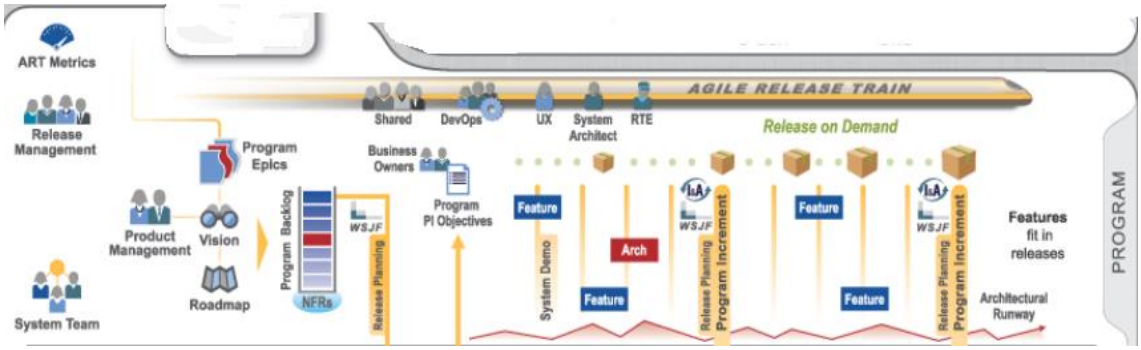

FIGURA 7 - Program

Fonte: http://www.scaledagileframework.com/

Projetos SAFe fazem várias entregas durante muito tempo, cada entrega gera valor a solução. No programa de incremento, as entregas são tipicamente 8-12 semanas, um múltiplo timebox, de iterações onde um incremento é muito importante para o sistema que está sendo desenvolvido. Cada release da solução é composto de 5 a 15 equipes ágeis e inclui os papéis e as infraestruturas necessárias para entregar totalmente testado, funcionando, software de nível de sistema.

As equipes ágeis se reúnem com as principais partes interessadas do Projeto e planejam o próximo incremento durante uma sessão do planejamento do release. Normalmente um engenheiro é o facilitador na sessão que serve como *Scrum Master* e também ajuda a manter o escopo. Todos os membros participam durante 1-2 dias, onde

eles compartilham a visão, planejam juntos, trabalham interdependentes, estabelecem esquipes com objetivos de planejamento e inovação para o próximo período.

Esses planos incluem um compromisso de entrega de um conjunto de requisitos que foram definidos como prioritários pelos gerentes do produto usando desenvolvimento de produção com prioridade econômica. Recursos aparecem impulsionados pela visão, roteiro e do programa, eles vem do *backlog* o repositório único e definitivo para o próximo trabalho previsto.

Os requisitos não funcionais fornecem a governança sobre as qualidades necessárias do sistema de desempenho, segurança, confiabilidade, disponibilidade, rendimento e outros. Os planos são feitos pelas equipes apresentando aos administradores para aprovação. Proprietários de negócios é parte integrante do projeto, e tem responsabilidades específicas para garantir que a solução recebe rápido o *feedback* do mercado.

As entregas requerem habilidades especiais para integrar, refinar e validar o código do sistema. Essas habilidades são encontradas em times de sistemas e especialistas chaves como: arquitetos de sistemas, designers, interface com usuário e recursos compartilhados de outras áreas. Os novos requisitos estão comprometidos externamente sob os auspícios da equipe de gerenciamento de liberação, que tem autoridades competentes para marketing, vendas, desenvolvimento, qualidade, operações e infra.

4.3.3 Portfolio

O nível mais alto do SAFe, onde os projetos estão alinhados a estratégia de negócios da empresa ao longo faz linhas do fluxo de valor. Em grandes organizações, pode se ter vários portfólios de projetos, um para cada instancia do SAFe, cada um com sua própria estratégia, financiamento e fiduciário. O nível de portfólio tem um numero de tópicos chave:

Portfólio de visão, que por sua vez contém:

- Temas estratégicos

-Valor Stream

- Orçamentos

- Histórias

- *Backlog* Portfolio

- Sistema *Kanban*

- Proprietários das histórias

- Arquitetura empresarial

- Gerenciamento do Projeto portfólio

- Métricas do portfólio

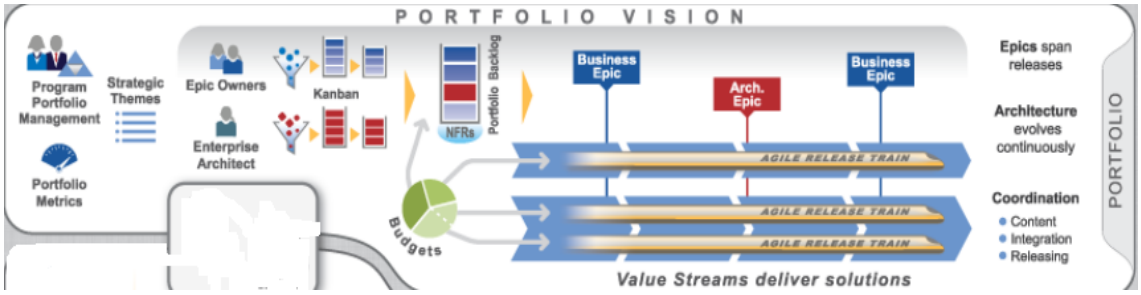

FIGURA 8 – Portfolio

Fonte: http://www.scaledagileframework.com/

A visão portfólio representa o mais alto nível de definição da solução e governança em uma instância do SAFe. As decisões tomadas aqui conduz a economia global para o portfólio.

Temas estratégicos são os objetivos de negócios, que se conectam a visão do portfólio e a estratégia de negócios em constante evolução da empresa.

O Valor *stream* é o tempo gasto para definição do sistema, desenvolvimento e implantação passos utilizado para construir e implementar sistemas que fornecem um fluxo contínuo de valor para o negócio ou cliente. O conjunto de Valor *stream* estabelece objetivos para os produtos e serviços em um portfólio. Fluxos de valor são realizados por pessoas, na construção da ART. Se ele tem um grande número de equipes (tipicamente 10-15) para construir e entregar valor em um fluxo de valor, em seguida, várias ART são formadas dentro desse fluxo. O Valor *stream* serve para a análise e melhoria sistemática pela ferramenta *Lean* de mapeamento do fluxo de valor.

O Orçamento financia as pessoas e outros recursos para ART. Cada parte no gráfico em forma de pizza representa um orçamento para uma ART especifica. Dentro dessa forma de orçamento, os gestores estão autorizados a desenvolver características de qualquer maneira que faz mais sentido econômico e de negócios para o projeto. Com este processo a empresa exerce a sua responsabilidade fiduciária pelo investimento aos acordados para as prioridades de negócios de condução. SAFe paradigma o orçamento *Lean*-ágil, além de projetos de custos, prevê a tomada de decisão rápida e com poderes, com controle fiduciário apropriado. Um orçamento especial representa uma reserva de financiamento (ou uma atribuição de capacidade, se todo o orçamento é alocado para ART) para histórias e *backlog* portfólio. [11]

Histórias capturam as maiores iniciativas em um portfólio. Histórias de Negócios descrevem funcionalidades ou experiências de usuários; histórias arquitetônicas capturam mudanças tecnológicas que devem ser feitas para manter os sistemas fluindo.

O SAFe sugere dois sistemas *Kanban* para gerir a gestão de portfólio de pedidos em atraso; um *Kanban* para histórias de negócios, e um outro *Kanban* para histórias arquitetônicas, embora eles também possam ser combinados em um sistema. [11] Estes sistemas:

- Visualizar o fluxo de valor de nível história através da análise e em execução
- Trabalhar no limite do esforço potencial no processo para ajudar a garantir a circulação.
- Conduzir uma colaboração precoce e eficaz com as equipes de desenvolvimento
- Fornecer uma base quantitativa para a tomada de decisão econômica
- Fazer políticas relativamente ás atribuições de capacidades explicitas

Cada sistema emprega cinco etapas (funil, avaliação, análise, *backlog*, programa e implementação) para gerenciar o fluxo de trabalho e trazer visibilidade ao trabalho.

Proprietários de histórias assumem a responsabilidade por uma história como ele se move através do sistema.

O *backlog* de portfólio é o mais alto nível do quadro, ele contém o conjunto de prioridades das histórias de Negócios e Arquitetônicos que são necessárias para ajudar o portfólio a alcançar a diferenciação de mercado, eficiência operacional ou soluções mais integradas.

Métricas de portfólio: processos ágeis, práticas e produtos de trabalho são inerentemente mensuráveis e há muitas métricas que podem ser usadas para avaliar e melhorar o *Team, Program,* Portfólio e progresso da empresa. O SAFe sugere uma série de métricas de portfólio, bem como métricas para o ART.

Pensamento *Lean* é o conjunto de princípios e práticas que maximizam o valor do cliente, minimizando o desperdício e reduzindo o tempo para o mercado. O objetivo do *Lean* é ser sustentável com menos tempo, com melhor valor para as pessoas e sociedade. O SAFe incorpora cinco elementos do *Lean*: o objetivo, *Kaizen* (melhoria continua), fluxo de desenvolvimento de produto, o respeito pelas pessoas e Lideres *Lean-ágil*. O pensamento *Lean* é essencial para ampliação das práticas ágeis para o nível empresarial e é fundamental para o SAFe. [11]

A principal responsabilidade para o sucesso da empresa está nas mãos dos gerentes, líderes e executivos. No SAFe, os líderes *Lean-ágil* são aprendizes ao longo da vida e professores que ajudam as equipes a construir melhores sistemas de software através da compreensão e mostrando valores, princípios e práticas de *Lean*, pensamento sistêmico e desenvolvimento de software ágil. No SAFe, os líderes *Lean-ágil*: Passam um visão sistêmica, abraçam o manifesto ágil, implementam desenvolvimento seguindo o fluxo do produto e desbloqueiam a motivação intrínseca dos trabalhadores do conhecimento. [11]

4.4 Métricas

Qualquer discussão sobre métricas pode começar com o entendimento da intenção do que se quer medir e onde há oportunidade para que seja medido. No caso do SAFe existem várias formas de medir e várias oportunidades de se medir algum processo.

Na figura FIG. 9, é demonstrado alguns pontos para medir o ART, em nível de TEAM e PROGRAM.

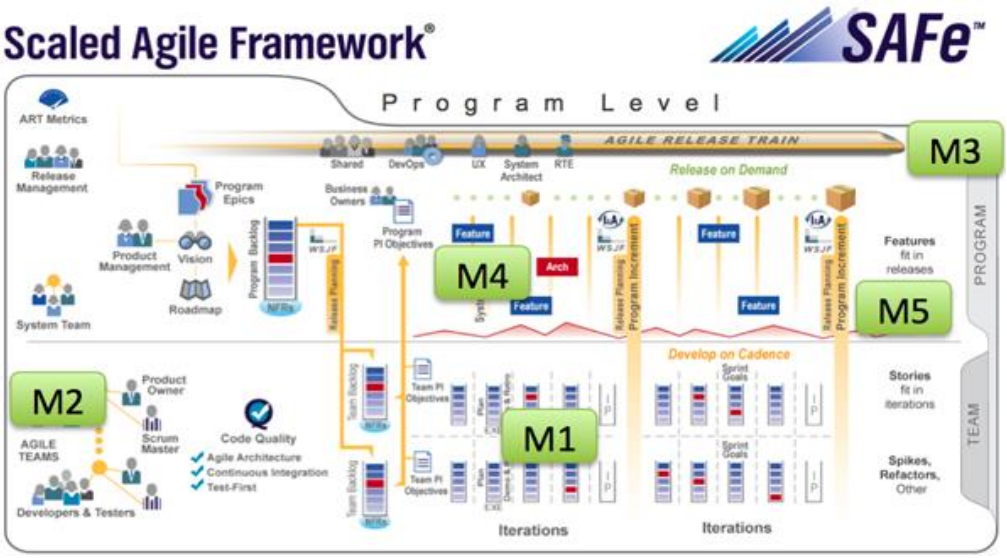

FIGURA 9 – Métricas ART

Fonte: http://www.scaledagileframework.com/

4.4.1 M1 – Métricas de iterações

Ao final de cada iteração é um momento oportuno para recolher tudo sobre as métricas que o time ágil concordou em fazer.

Isso ocorre em parte quantitativa do retrospecto da equipe. Uma dessas métricas de equipe está exemplificada na FIG. 10.

Funcionalidade	Iteração 1	Iteração 2	Iteração 3
Velocidade Planejada			
Velocidade Atual			
Histórias planejadas			
Histórias aceitas			
%Histórias aceitas			
Qualidade			
% Testes unitários			
Defeitos			
Novos casos de testes			
Novos casos de testes automatizados			
Total de testes			
Total % de testes automatizados			
Refatoração			

FIGURA 10 – Métricas de iterações

Fonte: http://www.scaledagileframework.com/

4.4.2 M2 – Auto avaliação do Team ágil

O Team Ágil deve continuamente avaliar e melhorar o seu processo. Muitas vezes isso é através de uma auto avaliação periódica e estruturada. Este dado serve para equipe debater e refletir sobre as práticas chaves que a ajudam a produzir os resultados. Uma avaliação simples de 25 pontos básicos das práticas do SAFe e Scrum, para isto basta preencher a planilha Self Assessment (anexo).

Quando o Time completa a planilha, automaticamente produz o gráfico de radar conforme FIG. 11. Com este gráfico temos os atuais itens de força e fraqueza.

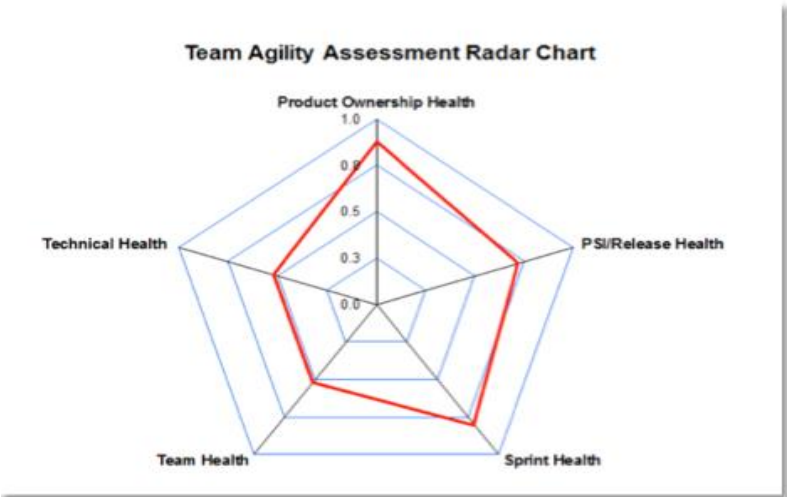

FIGURA 11 – Radar chart

Fonte: http://www.scaledagileframework.com/

4.4.3 M3 – ART auto avaliação.

Como a execução de um programa é um valor no SAFe, o ART também trabalha continuamente para melhorar o seu desempenho. Seguinte forma de auto avaliação pode ser utilizado para este fim. Isso pode ser usado em intervalos de incrementos ou a qualquer momento, a equipe faz uma pausa e avalia a sua organização e práticas. Tendendo esses dados ao longo do tempo é um indicador chave de desempenho para programa.

4.4.4 M4 – Atualização de relatórios de entrega.

Dada à natureza da criticidade de cada atualização das entregas, é muito importante avaliar o estado em tempo real.

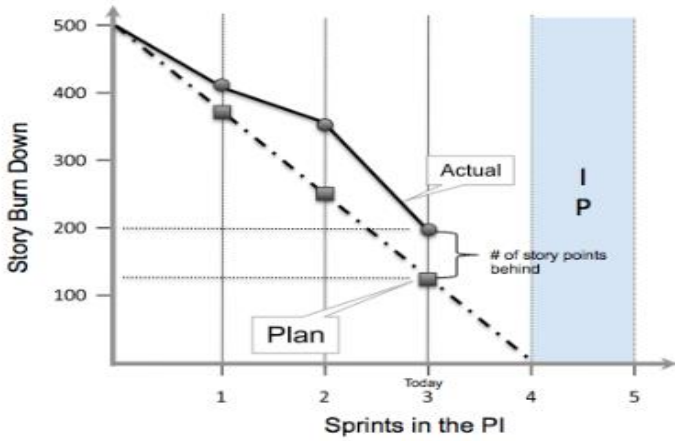

FIGURA 12 – Release Train PI burdown

Fonte: http://www.scaledagileframework.com/

4.4.5 M5 – Incremento de Programas

O Final de cada Incremento de programa é um ponto natural e significativo de medição. A FIG. 13 mostra em exemplo de métricas para o Programa.

Funcionalidade	PSI1	PSI2	PSI3
Velocidade Programa			
Previsibilidade de lançamentos			
Configurações planejadas			
Configurações aceitas			
Arquiteturas de configuração			
Histórias Planejadas			
Histórias aceitas			
Qualidade			
% Testes unitários			
Defeitos			
Total % de testes automatizados			

FIGURA 13 – PI Metrics

Fonte: http://www.scaledagileframework.com/

4.5 Aplicações

Muitas organizações começam com a metodologia ágil através de experimentação de uma ou duas equipes. Eventualmente, depois de algum sucesso moderado, o interesse em replicar este sucesso através de um departamento na empresa cresce. Ou seja, quando os desafios de adoção são muito mais profundos começam pela superfície. O SAFe pode ser usado para escalar as prática e processos projetados com uma pequena equipe. Para alinhamento com várias equipes, programas e serviços para que seja entregue algo de maior substância e valor de mercado do que apenas uma equipe.

Com SAFe é possível alinhar várias áreas de negócio e TI para uma maior colaboração, eficácia e eficiência na construção e desenvolvimento de software. O SAFe aborda cada uma destas questões e mais, identifica princípios e práticas chaves que escalam bem e devem ser mantidas em comum em suas equipes *Lean* e Ágil e programas para escalar o sucesso.

Não é incomum para grandes organizações ter algumas equipes implementando a sua própria visão sobre Ágil e fazê-lo com êxito para o trabalho em que elas são totalmente autônomas. No entanto, muitas vezes, essas organizações se esforçam ao planejar lançamentos que exijam demais de uma equipe Ágil. Algumas razões comuns para isso incluem equipes operando em diferentes cadências de iterações, usando diferentes métodos, utilizando diferentes técnicas, práticas para arquitetura e qualidade de código, e contando com ferramentas diferentes para gerenciar o ciclo de vida Ágil e relatar o progresso.

O SAFe delineia uma abordagem consistente para planejamento referido como um ART, que une um conjunto uniforme de produtos de várias equipes para entregar valor ao nível de programa. Ele faz isso em parte ao estabelecer uma consistente cadência de planejamento, execução e entrega que todas as equipes seguem dentro do programa.

Um desafio comum com o desenvolvimento de software em escala corporativa, Ágil ou não, é estabelecer e manter o alinhamento com a visão e estratégia de cima para baixo em toda organização. Muitas vezes há uma luta apenas para alinhar várias áreas de negócio com a mesma estratégia e mesmo quando há alinhamento entre os departamentos de negócios, há ainda outro desafio para garantir que a estratégia é claramente comunicada e entregue a todos o caminho para baixo nos níveis de equipe.

Certamente o SAFe não irá resolver todos os problemas na organização, ele fornece uma estrutura para implementação Ágil de forma consistente de portfólio para

programar a níveis de equipe. Em cada nível, papéis chave, responsabilidades, práticas e métricas são identificados que incorporam princípios Ágil e *Lean* e quando aplicadas de forma consistente pode ajudar a estabelecer o fluxo de alinhamento do topo de trabalho para baixo.

Não é segredo que mais e mais empresas de grande porte, antigas e novas, como John Deere, Discount Tire, Spotify, Salesforce e outras já descobriram como escalar Ágil (via abordagens do SAFe ou personalizados) para encurtar sua vantagem no tempo de desenvolvimento do produto e frequentemente entregar softwares para clientes satisfeitos. Atualmente não são apenas softwares e empresas de tecnologia, mas empresas em domínios mais tradicionais, como finanças, seguros e governamentais que buscam dimensionar o Ágil em busca dos mesmos benefícios.

SAFe é uma abordagem comprovada e documentada para trazer o Ágil e seus benefícios para empresas em geral. Fornece uma visão abrangente dos princípios e práticas de negócios e técnicos necessários de cima para baixo da empresa para escalar com sucesso o Ágil. Esses princípios e práticas não são todos inteiramente novos ou exclusivos para o SAFe, mas foram puxados juntos em um pacote coeso, pronto para implantar nas organizações.

4.6 Casos de sucesso

Várias empresas estão adotando o SAFe para melhorar a qualidade de entrega dos produtos bem como o tempo para geração de valores para seus clientes. [11]

4.6.1 John Deere

Em recente anuncio o grupo de soluções da John Deere disse que eles estão no meio de uma transformação ágil em grande escala, isso afeta centenas de profissionais em todo mundo.

Problemas:

Os pontos listados abaixo fazem parte de uma série de razões para que uma transformação Ágil aconteça.

1 – Por serem grande eles tomaram uma abordagem "*all in*". Muitos desenvolvedores foram afetados.

2 – Distribuída, afetando equipes nos Estados Unidos, Europa e Índia.

3 – Os sistemas são incorporados, em tempo real e complexo. A John Deere constrói soluções que consistem em telas, receptores, sistemas de orientação e grandes hardwares, incluindo tratores, pulverizadores e colheitadeiras. Também constroem sistemas de telecomunicações, sites e portais.

4 – As datas de entregas são fixas, veículos saem das fabricas todos os anos e devem sair na hora certa. O software tem que ser de extrema qualidade e também devem estar pronto no tempo certo.

Soluções Propostas:

- Demonstrar para alta e média gerência os benefícios de se trabalhar com o SAFe, para isto foram realizados treinamentos sobre *Lean*, *Scrum* e orientações Ágil.

- Alinhar com as equipes de desenvolvimento o apoio para implantação dos processos SAFe.

- Envolvimento de todas as equipes nos processos de entrega de valor para os clientes.

- Estabelecido um plano de formação dos funcionários e do orçamento da empresa.

Resultados:

Os resultados começaram a aparecer após essas ações.

Tempo de resolução de problemas em campo: 42% menor

Despesas: 50% menos

Tempo de produção: queda de 20%

Tempo de entrega para o mercado: 20% mais rápido

Engajamento dos funcionários: até 9,8%

4.6.2 Infogain

Provedor de serviços de terceirização que adotou SAFe para uma série de programas na Índia para promover o alinhamento com as partes interessadas do cliente e equipes e proporcionar, mecanismos de entrega de software sincronizadas e confiáveis.

A Infogain passou por uma série de SAFe *Quickstarts* para seus três programas diferentes. Treinamento, consultoria e facilitação dos eventos de nível de programa.

Algumas equipes da Infogain já operam no modelo Scrum ou usam algumas práticas ágeis em diferentes aspectos do processo. Entretanto todas as equipes em todos os três programas passaram pelo mesmo treinamento SAFe como parte do *Quickstart*, para garantir uma maneira comum de operar no contexto de escala.

O resultado foi o melhor alinhamento de visão comum dos processos avançados e visibilidade do progresso efetivo, planejamento de liberação descentralizado.

4.6.3 Accenture Technology

Consultoria global de gestão, serviços de tecnologia e *outsorcing*, com mais de 330.000 pessoas atendendo a cliente em mais de 120 países. Como parte de seu esforço para acelerar a entrega de software, a Accenture adotou Ágil em larga escala e em toda sua rede global de entrega, aproveitando o SAFe com uma gama de ferramentas.

Problemas:

- Desalinhamento da solução entre as equipes
- Integração do Agil com Waterfall
- Diferentes fusos horários, costumes e atividades entre as equipes
- Diferentes ferramentas de desenvolvimento utilizadas entre as equipes

Soluções:

- Melhoria na gestão de procura e rastreabilidade de portfólio por meio de equipes de entregas ágeis.
- Gerenciamento de configuração granular e rastreabilidade
- Integração com ferramentas de ciclos de vida ágeis que permitam baseado em história, gerenciamento de configuração conduzido a partir de dados de meta
- Rastreabilidade em tempo real do status de construção e implantação

- Construção e implantação automatizada
- Eficiência no desenvolvimento como consequência da melhoria de tempos e processo de interação com a ferramenta.

Resultados:

Melhoria de 50% e merge e retrofit;
Melhoria de 63% no gerenciamento de configuração de software;
Melhoria de 59% no custo de qualidade;
Melhoria de 90% na implementação e implantação.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

O SAFe tem sido uma forma de agregar conhecimento de diversas implementações em um quadro único, um sistema integrado com praticas comprovadas e que tem demonstrado ser bastante eficiente para empresa que pretendem escalar a metodologia ágil e as práticas *Lean* não só em seu ambiente de desenvolvimento mas, também em toda a empresa. Trazendo melhorias substanciais no envolvimento dos funcionários, tempo de mercado, e na qualidade da solução final.

Embora não exista nenhuma fórmula pronta para todos os desafios que uma empresa enfrenta, mediante a sua complexidade, o SAFe pode ser utilizado sendo necessário com alguns ajustes, ou seja, ele pode ser customizado dependendo do desafio de cada empresa ou projeto. O ideal é sempre usar de maneira correta os princípios fundamentais do SAFe, dessa forma pode-se garantir o sucesso da solução. Se ele não atender a necessidade, de forma adequada pode se usar os outros métodos e princípios (Ágeis e Lean) como apoio para na solução.

Conforme já citado em capitulo anterior, Dean Leffingwell [11], "quando os princípios do SAFe não atender aos desafios de uma empresa, a utilização dos outros princípios do *Lean* ou de métodos ágeis, certificam de que o trabalho e esforço está se movendo no caminho certo e continuo para o menor tempo, para um produto sustentável, com melhor qualidade e valor para as pessoas e para a sociedade". Há um valor aqui também.

5.2 Trabalhos futuros

Os próximos passos serão aplicar os conceitos e fundamentos do SAFe para obter resultados significativos para as organizações e para seus clientes. Reservando um espaço para documentar e compartilhar os resultados com a comunidade que atualmente trabalha com a pesquisa de melhoria nos resultados e na forma de se trabalhar com o SAFe

Cada empresa tem seu modo de operar, sua cultura e suas convicções. O ideal é adaptar as métricas e os processos do SAFe de forma que se consiga trabalhar de maneira produtiva, melhorando o ambiente de trabalho e gerando valor para empresa.

As métricas devem ser definidas de acordo com os objetivos de melhorias de cada área, fazendo com que a busca da melhoria continua esteja sempre na mente dos envolvidos.

REFERÊNCIAS

[1] AMBLER, S.W. Ambysoft, **Agility at Scale Survey: Results from the Summer 2012 DDJ State of the IT Union Survey**. Disponível em: <http://www.ambysoft.com/surveys/stateOfITUnion201209.html> Acesso em 05 de agosto de 2015.

[2] MELLO, Moacyr (2013). IBM, **O mundo depende de software**. Disponível em: <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/um_caminho_seguro_para_o_processo_de_desenvolvimento_safe_scaled_agile_framework?lang=en> Acesso em 05/08/2015.

[3] HIGHSMITH, Jim. **Agile Project Management: Creation Innovative Products**. 2nd Edition. July 20, 2009.

[4] <http://www.agilemanifesto.org/iso/ptbr/manifesto.html> Acesso em 04 de agosto de 2015

[5] <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_metodologias_c3_a1geis?lang=en> Acesso em 08 de agosto de 2015

[6] WOMACK, J. P.; JONES, D. T.; ROOS, D**. The Machine that changed the world: The History of Lean Production**. Nova York. York: Harper Perennial, 1991.

[7] KRAFCIK, Jhon F. **Triumph of the Lean Production System**. Sloan Management Review, v.30, n.1, 1998.

[8] JONES, D., WOMACK, J. **A mentalidade Enxuta nas Empresas**. Editora Campus, 1998.

[9] <http://www.lean.org.br/o_que_aspx> Acesso em 09 de agosto de 2015.

[10] <http://www.culturaagil.com.br/o-que-e-safe/> Acesso em 09 de agosto de 2015.

[11] <www.scaledagileframework.com> Acesso em 09 de agosto de 2015.

[12] SOARES, Michel dos Santos. **Comparação entre metodologias ágeis e tradicionais para o desenvolvimento de Software** – Unipac – Universidade Presidente Antonio Carlos, Faculdade de Tecnologia e Ciências de Conselheiro Lafaiete.

[13] AMBLER, S. Agile Modeling: **Effective Practices for extreme Programming and the Unified Process**. 1ª edição. [S.1]: Wiley,2004.

[14] FRIED, J.; HANSON, D. Rework [S.1]: Crown Business, 2004

[15] SCHWABER, Ken. What is Scrum?
[16] TAKEUCHI, H.; NONAKA, I. **The New Product Development Game**. Harvard Business Review, p.137 – 146, Janeiro – Fevereiro 1986. Disponível em: <http://apln-richmond.pbwiki.com/f/New.New Prod Devel Game.pdf> Acesso em 14 de setembro de 2015.

[17] KOSCIANSKI, A.; SOARES, Michel dos Santos. **Qualidade de Software**. São Paulo, Novatek Editora, 2006.

[18] SCHWABER, K.; BEEDLE, M. **Agile Software Development With Scrum**. 1ª edição. Upper Saddle River: Prentice-Hall, 2001. 150p

[19] <www.improveit.com.br> Acesso em 15 de setembro de 2015.

[20] MARÇAL, Ana Sofia Cysneiros. *et al.* **Estentendo o SCRUM segundo as Áreas de Processo de Gerenciamento de Projetos do CMMI.**

[21] FRANCO, E. F. **Um modelo de gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de software e nos princípios da produção enxuta.** Escola Politécnica da Universidade de São Paulo. Dissertação de Mestrado, 2007.

[22] BECK, K. et al. **Manifesto for Agile Software Development**. Agile Manifesto, Snowbird, 2001. Disponível em: <http://agilemanifesto.org>. Acesso em 16 de setembro de 2015.

[23] OHNO, T. **O Sistema Toyota de produção**: Além da produção em larga escala. Porto Alegre, Bookman, 1997. 152 p.

[24] VOLLMANN, T.E.; BERRY, W.L. WHYBARK, D.C. **Manufacturing Planning and Control Systems.** 4ª Ed., Boston, Irwin/McGraw-Hill, 1997. 836p.

[25] ROTHER, M.; SHOOK, J. **Aprendendo a Enxergar**: Mapeando o fluxo de valor para agregar valor e eliminar o desperdício. São Paulo, Lean Institute Brasil, 2003. 112p.

[26] <https://sandrocan.wordpress.com/tag/manufatura-enxuta/> Acesso em 26 de setembro de 2015.

[27] DEMING, W. Edward. **The New Enconomics**. MIT Press, 1994.

[28] Drucker, Peter F. **O Drucker Essencial**. Harper Collins, 2001.

ANEXOS

SCALED AGILE™

SAFe ScrumXP Team Self-Assessment

Team Name: xxxxxxxxxxxxxxxx **Date**: xx/x/20xx

Scoring: 0 - Never, 1 - Rarely 2 - Occasionally, 3 - Often, 4 - Very Often, 5 - Always

Area / Question	Score		Comments
Product Ownership Health			
Product Owner facilitates user story development, prioritization and negotiation	3,0		
Product Owner collaborates proactively with Product Management and other stakeholders	3,0		
User Stories are small, estimated, functional and vertical	3,0		
Product owner facilitates development of acceptance criteria which are used in planning, review and story acceptance	3,0		
Teams refine the backlog every sprint	3,0		
Total Product Health Score	15,0	60%	
PSI/Release Health			
Team participates fully in Release Planning and Inspect and Adapt	3,0		
Product backlog for the PSI is itemized and prioritized	3,0		
Teams proactively interact with other teams on the train as necessary to resolve impediments	3,0		
Team participates in System Demo every two weeks, illustrating real progress towards objectives	3,0		
Team reliably meet 80-100% of non-stretch PSI Objecives	3,0		
Total PSI/Release Health Score	15,0	60%	
Sprint Health			
Team plans the sprint collaboratively, effectively and efficiently	3,0		
Team always has clear sprint goals, in support of PSI objectives, and commits to meeting them	3,0		
Teams apply acceptance criteria and Definition of Done to story acceptance	3,0		
Team has a predictable, normalized velocity which is used for estimating and planning	3,0		
Team regularly delivers on their sprint goals	3,0		
Total Sprint Health Score	15,0	60%	
Team Health			
Team members are self-organized, respect each other, help each other complete sprint goals, manage interdependencies and stay in-sync with each other	3,0		
Scrum Master attends Scrum of Scrums and interacts with RTE as appropriate	3,0		
Stories are iterated through the sprint with multiple define-build-test cycles (e.g. the sprint is not a waterfalled)	3,0		
Team holds collaborative, effective and efficient planning and daily meetings where all members participate, status is given clearly, issues are raised, obstacles are removed and information exchanged	3,0		
Team holds a retrospective after each sprint and makes incremental changes to continually improve its performance	3,0		
Total Team Health Score	15,0	60%	
Technical Health			
Teams actively reduce technical debt in each sprint	3,0		
Team has clear guidance and understanding of intentional architecture guidance, but is free and flexible enough to allow emergent design to support optimal implementation	3,0		
Automated acceptance tests and unit tests are part of story DoD	3,0		
Refactoring is always underway	3,0		
CI, build and test automation infrastructure is improving	3,0		
Total Technical Health Score	15,0	60%	
Total Team Score	**75,0**	**60%**	

Radar Chart Data		
Product Ownership Health	60%	
PSI/Release Health	60%	
Sprint Health	60%	
Team Health	60%	
Technical Health	60%	