GESTÃO ÁGIL DE PROJETOS DE SOFTWARE

Trabalho de Conclusão de Curso

Engenharia da Computação

Nome do Aluno: Gilson Pereira da Silva
Orientador: Prof. Genésio Gomes Cruz Neto

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

GILSON PEREIRA DA SILVA

GESTÃO ÁGIL DE PROJETOS DE SOFTWARE

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Dezembro de 2015.

De acordo

Recife

____/_____/_____

_____
Orientador da Monografia

(Na versão final esta página deve ser substituída pela folha de aprovação digitalizada.)

Dedico à minha mãe, que aguarda ansiosamente a minha volta por cima.

Agradecimentos

Agradeço à Deus, em primeiro lugar, por possibilitar chegar até aqui, agradeço também a minha mãe ao meu pai que está no andar de cima, a minha família, aos meus professores da Poli, principalmente aos da área de Engenharia de Software que, de alguma forma colaboraram na minha formação e aos meus amigos.

Resumo

O trabalho a seguir versa sobre o gerenciamento ágil de projetos de software, que prioriza tanto à questão das mudanças de requisitos quanto à otimização dos processos de produção. Inicialmente são apresentadas as metodologias tradicionais de software, o conceito de agilidade e uma visão geral do tema, ou seja, o gerenciamento ágil de projetos de software. Em seqüência o documento traz uma explicação do Scrum, a metodologia ágil mais consagrada para o uso na gestão ágil de projetos de software, e outras metodologias usadas na gestão ágil de projetos de software. Como contribuições na área é detalhado um estudo qualitativo com gerentes de projetos que usam métodos ágeis, bem como são apresentadas sugestões para melhorias na aplicação do Scrum integrando-o ao método Kanban.

Abstract

The following work deals with the agile management of software projects, which prioritizes both the changing requirements and the optimization of process production. Initially are presented the traditional software methods, the concept of agility and an overview about agile management of software projects. In the following the document provides an explanation of Scrum, the most devoted agile methodology for use in agile management of software projects. As a contribution in the area we detail a qualitative study with project managers who use agile methods as well as suggestions for improvement in the implementation of Scrum integrating Kanban method.

Sumário

Conteúdo

Índice de Figuras

Figura 23. Exemplo de itens do quadro *Kanban*

Índice de Tabelas

Tabela de Símbolos e Siglas

RUP	Rational Unified Process
XP	Extreme Programming
PMI	Project Management Institute
PMBoK	Project Management Book of Knowledgement
WIP	Work In Progress

Capítulo 1

Introdução

Este capítulo apresenta uma visão geral do trabalho e seus objetivos.

1.1 Caracterização do Problema

O desenvolvimento de projetos de engenharia de software envolve muito cuidado e atenção. Software é intangível, não se pode tocar, não se pode pegar. Engenharia de software, segundo SOMMERVILLE [23], é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação até a manutenção do sistema, depois que ele entrou em operação. Um dos problemas mais recorrentes e mais comuns na produção de software é a grande quantidade de mudanças de requisitos que são solicitadas pelos clientes e usuários. A má gestão destas mudanças mais tarde pode custar muito caro para o projeto de software, de acordo com as pesquisas recentes realizadas pelo GROUP [6].

As metodologias de desenvolvimento de projetos de software ou até mesmo de gerenciamento de projetos envolvendo a produção de software mais tradicional, tais como o RUP [19] (Rational Unified Process) e o PMBoK (Project Management Book of Knowledge) do PMI [17] (Project Management Institute) acabam sendo o empecilho para a inclusão de mudanças de requisitos de software, pois elas são orientadas ao planejamento. Embora permitam mudanças de requisitos, o principal foco ou objetivo delas é seguir sempre um “plano” de projeto de desenvolvimento de software. O modelo cascata, também chamado de preditivo, é um exemplo de modelo de ciclo de vida muito referenciado que não permite mudanças de requisitos, devido a sua orientação ao planejamento.

Já os modelos de gestão ágil de projetos de software, como Scrum são uma evolução no que diz respeito a possibilitar mudanças no processo, e precisam estar sempre em evolução, pois o Scrum também possui limitações. Portanto, há a necessidade cada vez mais recorrente de estudos voltados à evolução nos projetos de software, de utilizar novos modelos de processo, em que se foque na gestão dos requisitos, da otimização do trabalho e do seu fluxo dentro do projeto para que a entrega do produto de software atenda às necessidades da organização, em termos de custo, prazo e principalmente de qualidade.

1.2 Objetivos e Metas

O objetivo do projeto é apresentar as metodologias ágeis de projeto de software, suas aplicações e seus diferenciais em relação às metodologias tradicionais, como o RUP [19] e o preditivo (ou cascata).

Especificamente o projeto pretende:

- Apresentar as metodologias Scrum e Kanban com seus princípios básicos e fundamentais.
- Apresentar visões comparativas entre as metodologias Scrum e Kanban para a gestão ágil de projetos de software.
- Realizar um estudo qualitativo com empresas/entidades locais em Pernambuco que utilizam métodos ágeis de gestão de projetos, identificando seus benefícios e desafios em relação às metodologias tradicionais.
- A partir do estudo da literatura, do estudo comparativo e da pesquisa realizada com empresas, propor um novo processo que integra as metodologias Scrum e Kanban.

1.3 Metodologia e Estratégia de ação

Para apresentar o que é proposto neste trabalho, pretende-se utilizar as seguintes metodologias ou estratégias de ação:

1. Fazer um levantamento bibliográfico (ou referências) sobre as metodologias ágeis de projeto de software, sobretudo Scrum e Kanban.
2. Fazer um estudo qualitativo com base em entrevistas com desenvolvedores e gerentes de projetos sobre o uso de métodos ágeis de gestão de projetos. Serão feitas entrevistas com 2 empresas entidades: uma universidade e uma Startup.
3. A partir das informações coletadas, propor um novo processo que melhor integre os métodos Scrum e Kanban. Esta integração é dada pelo aproveitamento das vantagens de cada um destes dois modelos de métodos ágeis, Scrum e Kanban.

1.4 Estrutura do Documento

Segue a estrutura dos demais capítulos:

- Capítulo 2: Gestão Ágil de Projetos de Software – mostra um dos problemas do gerenciamento tradicional de projetos, que é a questão de permitir mudanças no plano de projetos, incluindo mudança de requisitos, mostra o gerenciamento ágil de projetos de software, que inclui os conceitos de agilidade de projetos e o de gerenciamento ágil de projetos e suas limitações.

- Capítulo 3: Metodologias para a Gestão Ágil de Projetos – mostra as metodologias utilizadas que são: Scrum, Lean e Kanban. É falado sobre cada uma delas com seus conceitos e fundamentos.

- Capítulo 4: Sugestão de Melhorias na Gestão Ágil de Projetos – exibe os estudos de campo realizados e a melhoria do Scrum combinando com o Kanban.

- Capítulo 5: Conclusão – apresenta as considerações finais sobre a monografia, bem como os estudos futuros pretendidos pelo autor.

- Bibliografia – Referências utilizadas para fazer este trabalho.

Capítulo 2
Gestão Ágil de Projetos de Software

Uma das áreas da computação mais apaixonantes é a engenharia de *software,* pois exige dedicação e esforço para entregar ao cliente o que ele precisa. Isso não envolve apenas entregar o programa em operação. Segundo SOMMERVILLE [23], *software* não é apenas o programa, mas também toda a documentação associada e os dados de configuração necessários para fazer com que esses programas operem corretamente. Este capítulo apresenta como a gestão ágil de projetos tem se tornado uma alternativa consistente aos modelos tradicionais de engenharia de software.

1.1 Projetos de Software e o Problema da Mudança de Requisitos

Engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Além disso, segundo SOMMERVILLE [23], a Engenharia de *Software* procura sempre trabalhar e produzir os produtos de software de acordo com as restrições organizacionais e financeiras, buscando soluções que estejam dentro destas restrições.

Uma das atividades da Engenharia de *Software* é o gerenciamento de projetos de software, que envolvem algumas ou todas estas atividades:

- Elaboração de propostas.

- Planejamento e programação de projetos.
- Custo do projeto.
- Monitoramento e revisões de projetos.
- Seleção e avaliação de pessoal.
- Elaboração de relatórios e apresentações.

Outra atividade a ser executada é a gestão de riscos, pois envolve prevê-los para evitar os riscos que possam afetar a produção e a Engenharia do *Software*.

Segundo o RUP [19] (Rational Unified Process) um projeto de *software* é dividido em quatro fases: concepção, elaboração, construção e transição. Além disso, RUP [19] é um processo de desenvolvimento de software que obtém as melhores práticas, de forma que possam ser adaptadas para uma grande variedade de projetos e de organizações. A Figura 1 mostra o gráfico do RUP, ilustrando as fases (as colunas) e as disciplinas nela contidas (linha).

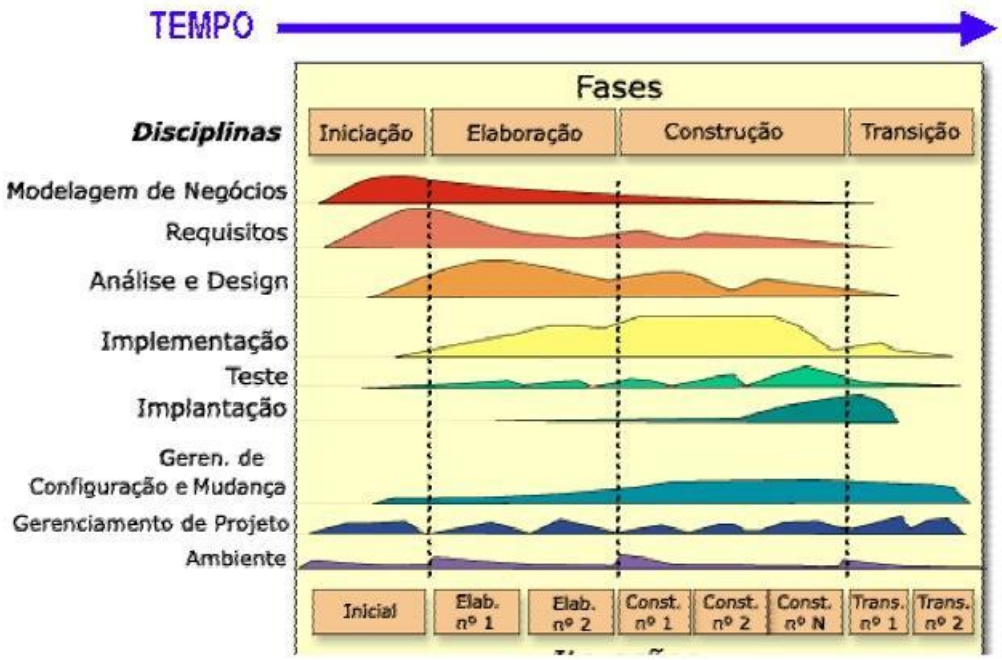


Figura 1. O gráfico do RUP

Fonte: Site do Wthrex

De acordo com o PMI [17], projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. O *software* é um desses resultados a ser entregue num projeto de software. Segundo o *Software* Extension to The PMBoK Guide Fifth Edition [18], projetos de software devem ser obtidos para satisfazer requisitos de serviços, ou oferecer apoio para as operações. Além disso, para obter os produtos de *software* faz-se necessário ter um gerenciamento de projeto de software adequado às restrições da organização que está sendo produzido.

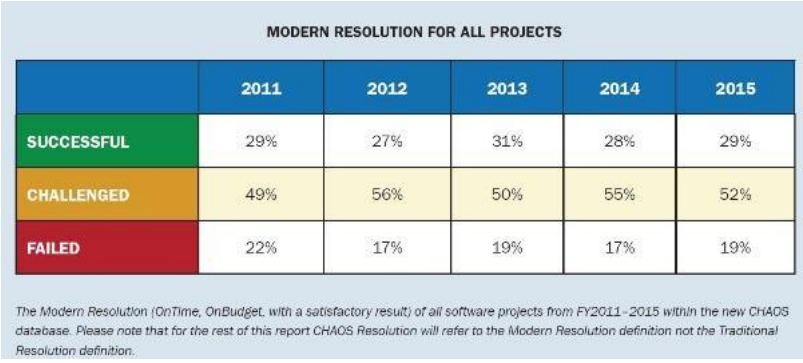
1.1.1 Problema do Gerenciamento Tradicional de Projetos

Durante muito tempo a indústria usou métodos que davam uma maior previsibilidade para o desenvolvimento de *software*, buscando o entendimento antecipado, geralmente fixando as restrições de escopo, de custo e do prazo de

desenvolvimento. Porém, com o passar dos anos, percebeu-se que a previsibilidade não se provou verdadeira, principalmente para projetos nos quais se busca o desenvolvimento científico de tecnologias que ainda não existem. Até mesmo nos projetos em que a tecnologia é de certa forma trivial, a complexidade que envolvia os negócios acabava tornando o processo imprevisível. Afinal de contas, o cliente não sabe o que quer até que veja partes de um *software* funcionando e o mercado mudar as suas necessidades, o seu foco. Além disto, quando o conhecimento sobre a tecnologia e sobre os negócios é baixo, a gestão de projetos torna-se complexa, fazendo que até atrapalhe o desenvolvimento do produto.

Por isso, percebeu-se que os métodos mais tradicionais de desenvolvimento, como o método cascata e o RUP, eram extremamente “pesados”, com relação a documentação e hierarquia entre os indivíduos. Eles não permitiam o grau de aprendizado necessário para até simplificar o desenvolvimento de *software*. Assim, no final dos anos 1990, iniciaram os experimentos em métodos mais leves, que eram focados em maior interação entre as partes interessadas e na entrega incremental de software funcionando, que pudesse ser usado pelo cliente mesmo antes do produto completo estar pronto. No ano de 2001, em um encontro de 17 especialistas na área de Engenharia de Software na cidade de Salt Lake City, nos Estados Unidos foi protocolado um Manifesto Ágil para o desenvolvimento de software.

De acordo com o GROUP [6], uma pesquisa realizada entre os anos de 2011 e 2015 mostrou que apenas 35% dos projetos terminaram com sucesso, inclusive, os projetos de *software*. Ainda sobre esta pesquisa, 52% dos projetos apresentaram mudanças de requisitos e de outros componentes no projeto, incluindo aqueles da área de *software* no último ano. Os dados desta pesquisa estão na Figura 2, Figura 3 e Figura 4.



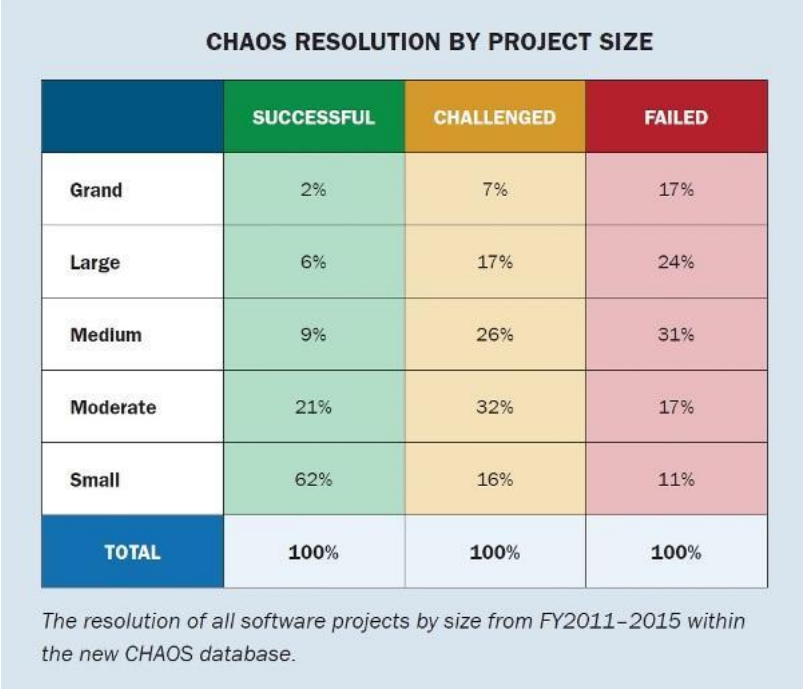
MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Figura 2. Pesquisa do Standish Group entre os anos de 2011 até 2015

Fonte: Site do Standish Group



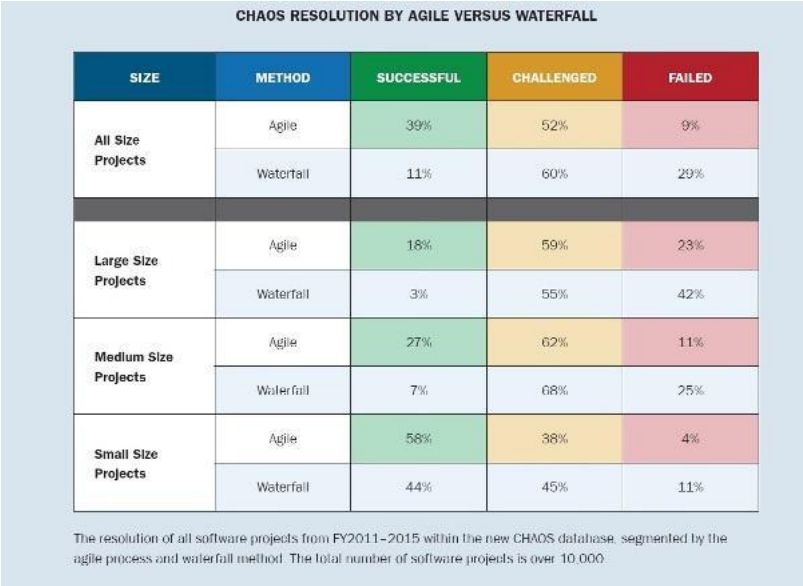
CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

Figura 3. Pesquisa do Standish Group com relação ao tamanho do projeto

Fonte: Site do Standish Group



CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

Figura 4. Pesquisa do Standish Group com relação entre o modelo ágil e o cascata

Fonte: Site do Standish Group

Então baseado nesta pesquisa e segundo relato dos próprios profissionais da área de software, um dos problemas mais recorrentes e mais comuns na produção, na Engenharia de Software e principalmente no gerenciamento de projetos de *software* são as muitas mudanças de requisitos na produção do *software* que é solicitada pelo cliente ou pelos usuários. Essas mudanças de requisitos são muito mais custosas, se deixarem essas mudanças para mais tarde no seu projeto de desenvolvimento, segundo o relato da experiência dos próprios profissionais de engenharia de software.

As metodologias de desenvolvimento de projetos de *software* ou até mesmo de gerenciamento de projetos envolvendo a produção de software mais tradicionais, que são o RUP [19] (Rational Unified Process) e o PMBoK (Project Management Book of Knowledge) do PMI [17] (Project Management Institute) acabam sendo o empecilho para a inclusão de mudanças de requisitos de *software*, pois elas são focadas no planejamento, ou seja, embora elas permitam mudanças de requisitos, o principal foco ou objetivo delas é seguir um plano de projeto de desenvolvimento de

software. Além disso, o modelo cascata, de acordo com SOMMERVILLE [23], ou preditivo, segundo o PMBoK do PMI [17] (um dos modelos de ciclo de vida) também é um empecilho, pois ele não permite mudanças de requisitos, pois também é orientado ao planejamento. O modelo cascata e o preditivo são ilustrados pela figura 5.

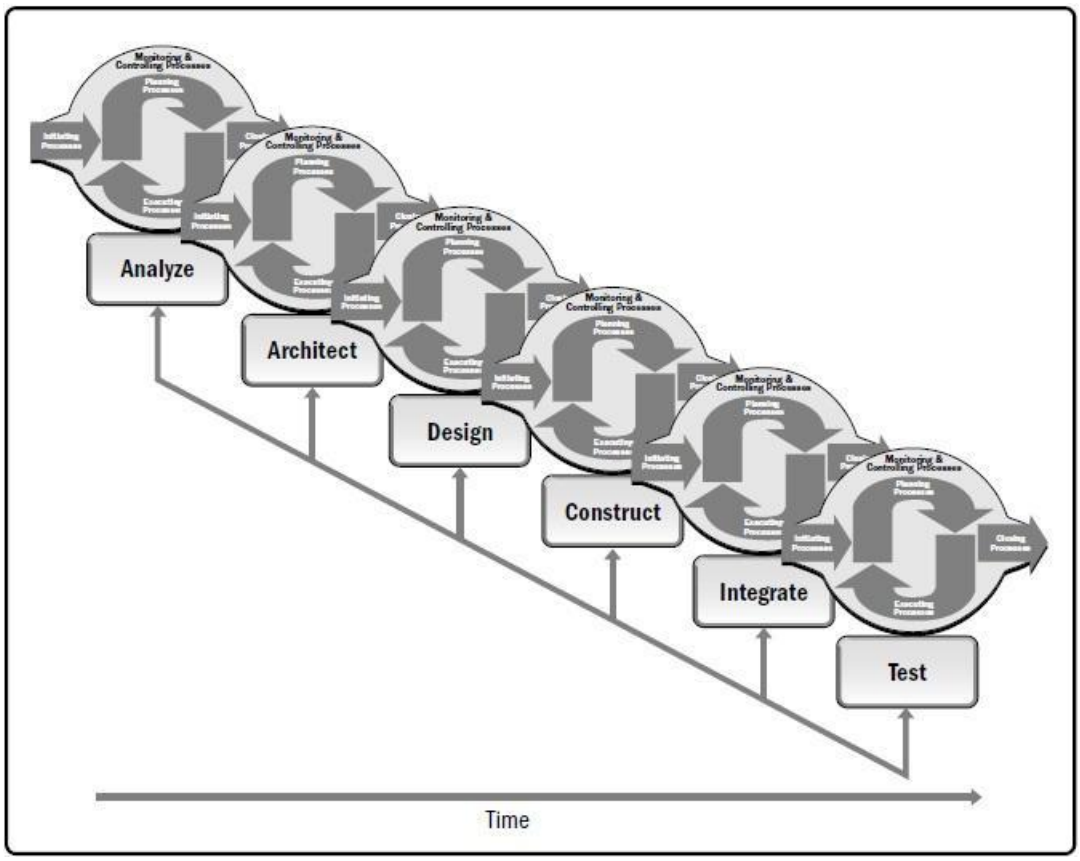


Figura 5. Ilustração do modelo preditivo ou cascata

Fonte: PMBoK Quinta Edição

Portanto, há a necessidade, cada vez mais recorrente dos projetos de *software* de utilizarem um modelo alternativo à gestão mais tradicional de projetos de software. Um modelo em que se possa focar na gestão dos requisitos, tratando da sua mudança, da otimização do trabalho e do seu fluxo dentro do projeto para que a entrega do produto de software atenda às necessidades da organização, em termos de custo, prazo e principalmente de qualidade.

A gestão ágil de projetos de *software* é uma alternativa que responde as necessidades que já foram relatadas, quanto a mudanças de requisitos, da otimização dos processos, para assim atender às necessidades dos clientes e dos usuários.

1.2 **Agilidade**

Desenvolvimento ágil de *software* é uma metodologia de desenvolvimento em que eles buscam minimizar os riscos no projeto pelo desenvolvimento de *software* em curtos intervalos de tempo, chamados de iterações, que, geralmente vão de duas a quatro semanas. Cada iteração dessas é como se fosse um mini-projeto de desenvolvimento de *software* e para cada uma delas exige todo o esquema necessário de desenvolvimento para implantação da nova funcionalidade: planejamento, análise de requisitos, análise e projeto, codificação, teste, implantação e documentação.

Enquanto que num projeto tradicional de *software* não se é obrigatório entregar alguma funcionalidade nova em cada iteração, no projeto ágil de desenvolvimento de *software* fica essa obrigação: a de entregar alguma funcionalidade nova e relevante de sistema de *software* a cada final das iterações, onde a equipe de projeto avalia as prioridades do projeto todo.

Métodos ágeis enfatizam comunicações em tempo real, preferencialmente face-a-face a documentos escritos, já que a equipe de projeto atua e se comporta de forma mais unida e, conseqüentemente, mais consistente, com essa equipe disponível da seguinte forma: os programadores e seus clientes (para eles, clientes são as pessoas que definem o produto de software, podendo ser o gestor, o analista de negócio ou mesmo o cliente), os testadores, os projetistas das iterações e gerentes ou líderes de projetos. Também enfatizam o trabalho de software como medida de progresso que, combinado com a comunicação presencial, geram uma

quantidade menor de documentações (gerando apenas uma quantidade útil de artefatos).

Segundo BECK [2], o Manifesto Ágil prega os seguintes valores:

- Indivíduos e Iterações mais do que Processos e Ferramentas;
- Software funcionando mais do que uma documentação detalhada;
- Colaboração com o cliente mais do que a negociação de contratos;
- Responder a mudanças mais do que seguir um plano.

Ainda segundo o manifesto, os princípios da agilidade valorizam:

- A mais alta prioridade é a satisfação do cliente, por meio da liberação mais rápida e contínua de *software* de valor;
- Receba bem as mudanças de requisitos, mesmo em estágios mais tardios do desenvolvimento. Processos ágeis devem admitir mudanças que trazem vantagens competitivas para o cliente;
- Libere *software* freqüentemente (em intervalos de 2 semanas ou 4 semanas), dando preferência para uma escala de tempo mais curta;
- Mantenha pessoas ligadas ao negócio (clientes) e desenvolvedores trabalhando juntos na maior parte do tempo do projeto;
- Construa projetos com indivíduos motivados, dê a eles o ambiente e suporte que precisam e confie neles para ter o trabalho realizado;
- O método mais eficiente e efetivo para repassar informação entre uma equipe de desenvolvimento é pela comunicação face-a-face;
- *Software* funcionando é a principal medida de progresso de um projeto de software;
- Processos ágeis promovem desenvolvimento sustentado. Assim, patrocinadores, desenvolvedores e usuários devem ser capazes de manter conversação pacífica indefinidamente;

- A atenção contínua para a excelência técnica e um bom projeto (design) aprimora a agilidade;
- Simplicidade;
- As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas;
- Em intervalos regulares, as equipes devem refletir sobre como se tornarem mais efetivas, e então refinarem e ajustarem seu comportamento de acordo.

As definições modernas de desenvolvimento de *software* ágil evoluíram a partir de meados da década de 1990, sendo uma reação ao que se chama de métodos pesados, caracterizada por uma pesada regulamentação, regimentação e micro gerenciamento usando o modelo de desenvolvimento em cascata. O processo originou-se da visão de que o modelo em cascata era pesado, burocrático, lento e, por incrível que pareça contraditório a forma com que os engenheiros de *software* sempre realizaram o trabalho com eficiência.

Uma visão que levou ao desenvolvimento de métodos ágeis e iterativos era retorno à prática de desenvolvimento vistas nos primórdios da engenharia de *software*.

No princípio, métodos ágeis eram conhecidos como métodos leves. Em 2001, membros proeminentes da comunidade se reuniram em Snowbird e adotaram o nome de métodos ágeis, tendo assim publicado o manifesto ágil, documento que reúne os princípios e práticas desta forma de desenvolvimento. Depois algumas destas pessoas fundaram a Agile Alliance, uma organização não lucrativa que promove o desenvolvimento ágil. Os métodos ágeis iniciais, criados a priori em 2000, foram:

- Scrum (1996);
- Kanban (2006);
- XP (1996);

- FDD e DSDM (1995);

1.3 Gestão Ágil de Projetos

Gerenciamento Ágil de Projetos ou do inglês Agile Project Management (APM) é a alternativa à maneira tradicional de gerenciamento de projetos. Conjunto de princípios, valores e práticas que visam às equipes de projetos entregarem produtos, serviços ou resultados de valor, cujos ambientes são considerados instáveis, complexos e desafiadores. A gestão ágil de projetos foi criada a partir dos valores e dos princípios dos Métodos Ágeis de Desenvolvimento de *Software*, a partir do Manifesto Ágil. HIGHSMITH [7] disse: Conectado a iniciativas correlatas das indústrias da construção civil e aeroespacial (uma vez que o cenário de incertezas era comum a ambos), o movimento para o desenvolvimento ágil de *software* ampliou sua abrangência, sendo estendido ao gerenciamento de projetos. Para esse contexto, CHIN [3] propôs a criação de um novo modelo e não uma simples expansão do Gerenciamento Clássico de Projetos.

O Gerenciamento Ágil de Projetos, de acordo com HIGHSMITH [7] e CHIN [3], se desfaz da postura antecipatória, fortemente calcada no planejamento prévio de ações e atividades, típica do gerenciamento tradicional de projetos, e busca o desenvolvimento da visão de futuro e da capacidade de exploração.

HIGHSMITH [7] afirma que são cinco os objetivos principais para um bom processo exploração, que acabam por constituir a base para o Gerenciamento Ágil de Projetos:

1. Inovação Contínua: A entrega de produtos que atendam aos requisitos dos clientes e que agreguem valor ao negócio. Inovação aqui é associada ao ambiente organizacional cuja cultura estimule o autogerenciamento e a autodisciplina.

2. Adaptabilidade do Produto: Os produtos desenvolvidos devem não apenas valor ao cliente no estado presente, mas também tem que ser adaptáveis às novas necessidades futuras.
3. Tempos de Entrega Reduzidos: Foco, direcionamento preciso e capacidade técnica da equipe é necessária para a redução dos prazos de desenvolvimento de produtos com o intuito de melhor aproveitamento das oportunidades de mercado e de melhor retorno sobre o investimento.
4. Capacidade de Adaptação do Processo e das Pessoas: Formação de equipes em que os integrantes estejam confortáveis com as mudanças e que não as enxerguem como empecilhos e sim como desafios de um ambiente dinâmico; estabelecimento de processos que respondam rapidamente às alterações do negócio.
5. Resultados Confiáveis: Entrega de produtos de valor ao cliente, garantindo a operação, o crescimento e aumento da lucratividade da empresa como um todo.

Os valores e princípios descrevem as razões do Gerenciamento Ágil de Projetos, já as práticas descrevem como realizá-lo. HIGHSMITH [7] menciona que: os valores principais deste novo enfoque de gerenciamento de projetos endereçam tanto a necessidade de criação e entrega de produtos ágeis, adaptáveis e de valor, como a necessidade de desenvolvimento de equipes de projeto com as mesmas características. Os quatro valores centrais do Gerenciamento Ágil de Projetos são:

1. As respostas às mudanças são mais importantes do que o seguimento de um plano.
2. A entrega de produtos está acima da entrega da documentação.
3. Priorização da colaboração do cliente sobre a negociação de um contrato.
4. Os indivíduos e suas interações são mais importantes do que os processos e as ferramentas.

HIGHSMITH [7] diz, em relação ao primeiro valor, que os projetos exploratórios, alvo do Gerenciamento Ágil de Projetos, são caracterizados por processos que enfatizam a visão do futuro e a exploração, ao invés do planejamento detalhado e a respectiva execução. Ainda segundo ele e CHIN [3], não se trata apenas de absorver pequenas alterações de escopo, prazo ou custo, mas sim de dar abertura à mudança completa de planos, requisitos, tecnologia e arquitetura no decorrer do projeto. HIGHSMITH [7] embasa sua opinião, ao apresentar o estudo de caso de uma companhia que, erroneamente, recusou-se a realizar mudanças no plano traçado inicialmente para um projeto de desenvolvimento de *software*, que foi orçado aproximadamente em 125 milhões de dólares, levando a um grande e custoso desastre.

Abordando o segundo valor, HIGHSMITH [7] argumenta que não se trata de menosprezar a importância da documentação, mas sim de assumir que os resultados só podem ser avaliados pela equipe de projeto e pelo cliente quando algo concreto é apresentado, ou seja, quando se tem um produto (ou software) operante. Ele ainda defende a manutenção de um patamar mínimo e suficiente de documentação para auxiliar o processo de comunicação e colaboração, propiciar a transferência de conhecimento, preservar a informação histórica, servir de base para a melhoria de produtos e processos e, algumas vezes, atender aos requisitos legais.

O terceiro valor, de acordo com HIGHSMITH [7], tem o pressuposto de estabelecer uma parceria entre o cliente e a equipe de projeto, na qual cada um possui papéis e responsabilidades específicos e bem definidos, sendo cobrados para isso. Esta parceria deve ser marcada por uma forte colaboração e não por disputas contratuais. Apesar de reconhecer a existência de diferentes partes interessadas no projeto, ele diz que o cliente tem que “reinar soberano” e apresenta a seguinte definição de cliente: “[...] um indivíduo ou grupo de indivíduos que usa os produtos ou serviços para gerar valor ao negócio”.

De acordo com HIGHSMITH [7], o quarto valor diz que os processos são utilizados como guias ou trilhas e as ferramentas são utilizadas para melhorar a eficiência, mas sem pessoas com qualificações técnicas e comportamentais adequadas, nenhum processo ou ferramenta produz resultado algum. O movimento ágil deposita valor no indivíduo e reconhece sua capacidade de auto-organização, autodisciplina e sua competência. Enfocando o sucesso na visão do Gerenciamento Ágil de Projetos, ele ainda menciona que o sucesso é direcionado pelas pessoas e suas interações entre pessoas ou equipes tecnicamente qualificadas e a capacidade da equipe de aprender e aplicar os conhecimentos adquiridos são determinantes para o sucesso ou fracasso de um projeto, estando estritamente ligados ao atendimento das expectativas do cliente. Como as pessoas são normalmente guiadas por um conjunto interno de valores, desenvolver a agilidade depende do perfeito alinhamento entre o ambiente e este sistema de valor de cada indivíduo. Esta é a razão pela qual o Gerenciamento Ágil de Projetos é fortemente calcado em valores e também o motivo por que muitas vezes é impossível a aplicação do Gerenciamento Ágil de Projetos em determinadas organizações ou equipes (tal como ocorre com a adoção dos Métodos Ágeis de Desenvolvimento de *Software*).

Além dos valores já descritos, o Gerenciamento Ágil de Projetos possui um conjunto de princípios chaves, que norteiam sua aplicação.

Na gestão ágil de projetos, a ênfase do gerenciamento de projetos se dá na fase de execução do projeto, ao invés da fase de planejamento, mesmo nas duas abordagens serem gastas mais tempo na fase de execução. No mundo ágil, as decisões cruciais para o sucesso ou falha do projeto devem ser tomadas na execução do projeto, pois na fase de planejar, no geral, não se sabe o que o cliente deseja realmente. Isso não significa que o planejamento e a definição do projeto devam ser ignorados, porém os focos nas tomadas de decisões são mais tomados na execução do projeto.

No projeto que envolve agilidade é levado em consideração um fator relevante: a incerteza. Incerteza é tudo aquilo que vai causar alguma

mudança de projeto. Segundo CHIN [3] há dois tipos de incertezas: Incerteza Interna e Incerteza Externa. A incerteza interna envolve variação que está dentro dos aspectos do projeto e que podem ser controlados pelo gerente de projetos. Fatores que são o escopo, tempo, custo, tomada de decisões, mudanças de cliente, mudanças no ambiente de negócio específico da indústria e obstáculos técnicos. Já as incertezas externas lidam com fator que não está no alcance do projeto e que o gerente de projetos não tem como ter o controle. Esses fatores são, por exemplo, decisões estratégicas de negócios.

Outro fator a ser levado em consideração no mundo ágil é a questão da velocidade. Confunde-se muito achar que na gestão ágil de projetos o trabalho no projeto deve ser apressado ou que se tenha pressa. Ou seja, entregar o mais rápido possível nos projetos que são características da gestão ágil de projetos não é entregar o produto de qualquer jeito, sem estar dentro do critério de aceitação do cliente.

O uso da gestão ágil de projetos se dá em ambientes de projetos de desenvolvimento de tecnologias ou plataformas e em ambiente organizacional único. No primeiro caso, a natureza dos projetos é única, com grau de incerteza alto, exigindo da equipe comprometimento, criatividade e determinação. No segundo caso, que não há múltiplas organizações participando do projeto que sejam o cliente do projeto.

A tabela 1 mostra os aspectos entre o gerenciamento clássico e o gerenciamento ágil em seus diversos atributos.

Tabela 1: Comparação entre o Gerenciamento Tradicional e o Gerenciamento Ágil.

Aspecto	Tradicional	Ágil
Ambiente de Projeto	Operacional	Desenvolvimento de Tecnologia

Partes Interessadas Organizacionais	Uma ou mais organizações	Uma organização
Foco do Gerente de Projeto	Gerenciar cronograma, escopo e recursos	Alcançar os resultados de negócio
Fronteiras do Projeto	Estático	Dinâmico
Ambientes Internos e Externos ao Projeto	Separado	Integrado
Papéis da equipe de projeto	Definido por título ou função	Definido por expertise e desejos para o sucesso da equipe
Papéis e responsabilidades	Desenvolver barreiras que não devem ser rompidas	Encorajar a superação das barreiras
Tomada de decisões	Centralizada	Descentralizada
Reuniões	Menos freqüentes	Mais freqüentes
Orientação do Gerente de Projeto	Interno ao projeto	Interno e externo ao projeto
Perfil do	Gestor ou líder	Líder

Gerente de Projeto		

1.4 Limitações para a Gestão Ágil de Projetos

A gestão ágil de projetos possui limitações na sua adoção e uso. Apresentamos algumas delas:

1. Projetos Grandes: A gestão ágil é mais usada na construção de pequenos projetos, pois eles têm duração mais curta. Os projetos grandes possuem a limitação de em pelo menos uma das fases do projeto não possibilitar nem a divisão dessas partes quanto à de entregar a parte obrigatória do produto para a dada iteração. Porém, há estudos voltados para a gestão ágil de projetos que são aplicados para projetos grandes.

2. Equipes Distribuídas Geograficamente: Por exigir que a comunicação na gestão ágil de projetos seja preferencialmente presencial, projetos que utilizem equipes distribuídas geograficamente passam a não ser bem aplicados, pois perde a característica de passar maior união da equipe com a comunicação face-a-face. Porém já há estudos utilizando a gestão ágil de projetos que são distribuídos geograficamente. Um desses exemplos é o que vai ser apresentado em um dos estudos de campo, no Capítulo 4.

1.5 Considerações finais

Este capítulo apresentou os problemas relacionados a gestão de projetos de software, utilizando os resultados da pesquisa do GROUP [6], o RUP [19] e o

modelo cascata. Um destes problemas é a quantidade grande de mudanças de requisitos e da imprevisibilidade deles. A solução proposta, gestão ágil de projetos de software, foi colocada para solucionar esse problema e também do problema da otimização do trabalho nos projetos. Foi discutido o conceito de agilidade de projetos, além do gerenciamento ágil de projetos em si e suas limitações de uso.

Capítulo 3

Metodologias para a Gestão Ágil de Projetos

Neste capítulo apresentaremos as principais metodologias utilizadas para realizar a gestão ágil de projetos de *software*.

1.1 Scrum

O Scrum responde ao uso para a gestão ágil de projetos. De acordo com o SCRUM GUIDE [21], é um *framework* para desenvolver e manter produtos complexos. Esta definição do Scrum consiste em papéis, eventos, artefatos e as regras do Scrum que unem os demais os mantendo integrados. É um *framework* dentro do qual as pessoas podem tratar e resolver problemas tanto complexos quanto adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Scrum é:

- Leve.
- Simples de entendimento.
- Difícil de dominar.

Scrum não é processo ou técnica de desenvolvimento de produtos, mas é um *framework* que pode ser empregado vários processos ou técnicas, contendo a eficácia das práticas de gerenciamento e desenvolvimento de produtos, de modo que possa ser melhorado. Na figura 6 é exibido o esquema geral no Scrum.

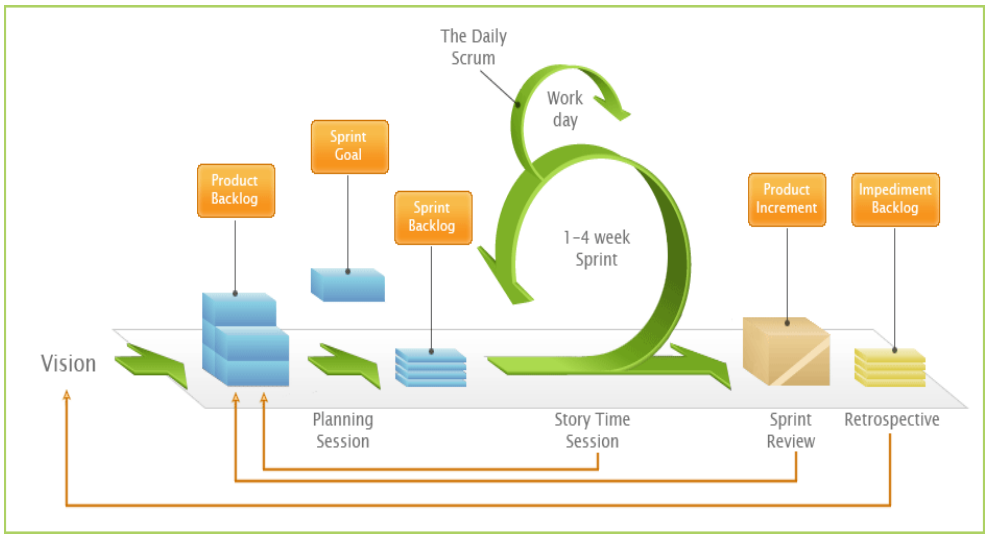


Figura 6. Esquema geral do Scrum

Fonte:http://andreykurka.wordpress.com/2013/10/24/scrum-no-burndown

Este *framework* consiste nos times do Scrum relacionados a papéis, eventos, artefatos e regras, onde cada componente serve a um objetivo específico e é essencial para o uso e o sucesso do Scrum. As regras deste *framework*, segundo o SCRUM GUIDE [21], integram os eventos, papéis e artefatos, administrando as relações e interações entre eles.

Scrum é fundamentado numa abordagem iterativo-incremental em busca de aperfeiçoar a previsibilidade e o controle de riscos. Ele é sustentado em três pilares para o apoio a implementação de controle de processo empírico: transparência, inspeção e adaptação. O primeiro exige que os aspectos do processo tenham que estar visíveis aos responsáveis pelos resultados, em que sejam definidos por um padrão comum para os observadores compartilharem um entendimento comum. O segundo refere-se às inspeções dos artefatos e do progresso do projeto pelos usuários do Scrum, visando detectar variações. Observe que esta inspeção não pode ocorrer freqüentemente, pois ao invés de ajudar, atrapalha a própria execução das tarefas. Já o último se refere a ajustes que devem ser realizados quando um ou

mais aspectos de um processo desviou para fora dos limites aceitáveis, devendo estas medidas sejam tomadas o mais breve possível.

Papéis

Time *Scrum* é composto pelo *Product Owner*, o Time de Desenvolvimento e o *Scrum Master*. Esta equipe é auto-organizável e multifuncional, escolhendo a melhor maneira para completar o seu trabalho, ao invés de ser dirigidos por alguém de fora do time. Ela possui todas as competências necessárias para completar o trabalho sem a dependência de outras pessoas de fora da equipe. Times Scrum entregam produtos de forma iterativa e incremental, maximizando as oportunidades de realimentação. Entregas incrementais de produtos garantem que esteja disponível uma versão potencialmente funcional do produto.

O Product Owner

Responsável por maximizar o valor do produto e do trabalho do Time de Desenvolvimento, ele também é o único responsável por gerenciar o *Backlog* do produto. Este papel ainda é responsável por 1) expressar claramente o *Backlog* do Produto, 2) ordenar os itens desse *Backlog* com maior valor de negócio para que sejam desenvolvidos primeiro, 3) dar a garantia do trabalho realizado e 4) dar garantia que o *Backlog* do produto contenha visibilidade, transparência, clareza e o entendimento dos itens desse *Backlog* pelo time de desenvolvimento no nível necessário.

O Time de Desenvolvimento

São profissionais que realizam o trabalho de entregar uma versão potencialmente executável que incrementa o produto ao final de cada *Sprint*. Este time é estruturado e autorizado pela organização tanto para organizar quanto gerenciar seu próprio trabalho. O time de desenvolvimento é auto-organizado, é multifuncional, os seus integrantes são somente desenvolvedores, podendo ter habilidades específicas e áreas de especialização, porém a responsabilidade é do time, não havendo sub-times dedicados a domínios específicos. É preferenciável que este time de desenvolvimento contenha entre 3 a 9 integrantes.

O Scrum Master

Um servo-líder, que é responsável pela garantia do entendimento do *Scrum* ao time *Scrum*, aderindo à teoria, práticas e regras do Scrum. O *Scrum Master*:

- Para o *Product Owner*:
    - Encontra maneiras paga gerenciar o *Backlog*;
    - Comunica a visão, objetivo e itens do *Backlog* do Produto de forma clara;
    - Ensina ao Time *Scrum* a criação de itens do *Backlog* de forma clara e concisa;
    - Compreensão em longo prazo do planejamento do Produto, da agilidade;
    - Facilitar os eventos conforme necessários.
- Para o time de desenvolvimento:
    - Oferece treinamento em autogerenciamento e interdisciplinaridade e em ambientes organizacionais onde o *Scrum* não é totalmente compreendido e adotado;

    - Remove obstáculos para o progresso do Time de Desenvolvimento;
    - Facilita os eventos conforme necessários;
    - Ensinar e liderar o time a criar produtos de alto valor.
- Para a organização:
    - Lidera e treina a organização na adoção do *Scrum*;
    - Planeja a implementação do *Scrum* na organização;
    - Auxilia as partes interessadas a compreender e aplicar o *Scrum*;
    - Causa mudanças com o intuito de aumentar a produtividade do Time *Scrum*;
    - Trabalha com outros *Scrum Masters* para o aumento da eficácia da aplicação do *Scrum* nas organizações.

Sprint

É um *time-boxed* de duas a quatro semanas, podendo ser considerado um projeto com um horizonte menor que um mês, em que uma versão incremental potencialmente utilizável do produto, é criada. *Sprints* possuem durações consistentes com o esforço de desenvolvimento. Uma nova *Sprint* inicia logo depois do término da *Sprint* anterior. Cada Sprint possui uma reunião de planejamento da *Sprint*, reuniões diárias, o trabalho de desenvolvimento, uma revisão da *Sprint* e a retrospectiva da *Sprint*. Durante a *Sprint*:

- Não pode ter mudanças que ponham em risco o objetivo de cada *Sprint*;
- As metas de qualidade não devem diminuir;

- O escopo pode ser negociado e clarificado entre o Time de Desenvolvimento e o *Product Owner* à medida que for mais aprendido.

Somente o *Product Owner* pode cancelar uma *Sprint*, mesmo sob influência das outras partes interessadas do projeto e esse cancelamento ocorre quando uma dada Sprint perdeu o sentido de sua existência.

Eventos Scrum

Segundo o SCRUM GUIDE [21], o *Scrum* prescreve quatro Eventos formais, contidos dentro dos limites da *Sprint*, para inspeção e adaptação, que são:

- Reunião de planejamento da *Sprint*, onde é verificado o trabalho a ser realizado, gerando um plano criado de forma colaborativa pelo Time *Scrum*, com duração máxima de oito horas.
- Reunião diária, que é um evento de 15 minutos para ajustar o que o Time de Desenvolvimento fez, fará e quais obstáculos encontrados do trabalho realizado depois da Reunião Diária anterior. Esta é uma reunião chave para inspeção e adaptação.
- Reunião de revisão da *Sprint*, que é feita no final de cada *Sprint*. Ela serve para a colaboração do Time *Scrum* e das partes interessadas sobre o que foi feito e fazer adaptações do Backlog do Produto, caso necessário. Com base nisso e nas mudanças durante a *Sprint*, os participantes colaboram nas próximas coisas que podem ser feitas para agregar valor. Esta é uma reunião de duração de 4 horas.
- Retrospectiva da *Sprint*, que ocorre depois da revisão da *Sprint* e antes da reunião de planejamento da próxima *Sprint*. Ela serve para o Time *Scrum* inspecionar a si próprio e fazer um plano de melhorias para a próxima *Sprint*. O *Scrum Master* participa dela mais como um membro auxiliar.

Artefatos do Scrum

Segundo o SCRUM GUIDE [21], os artefatos do *Scrum* representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação. Os artefatos definidos para o *Scrum* são especificamente projetados para maximizar a transparência das informações chave de modo que todos tenham o mesmo entendimento dos artefatos. Estes artefatos são:

- *Backlog* do Produto, que é uma lista ordenada e única de tudo que é necessário no produto. O seu responsável é o *Product Owner*. O *Backlog* evolui à medida que o projeto for desenvolvido. Ele lista todas as características funções, requisitos, medições e correções que sejam necessárias ao produto nas futuras versões.
- *Backlog* da *Sprint*, que é um subconjunto de itens do *Backlog* do Produto devidamente selecionados para a *Sprint*, junto com o plano para entregar o incremento do produto e atingir o objetivo da *Sprint*. Na *Sprint*, todo o trabalho é mostrado em uma *Task Board*, segundo a figura 7.

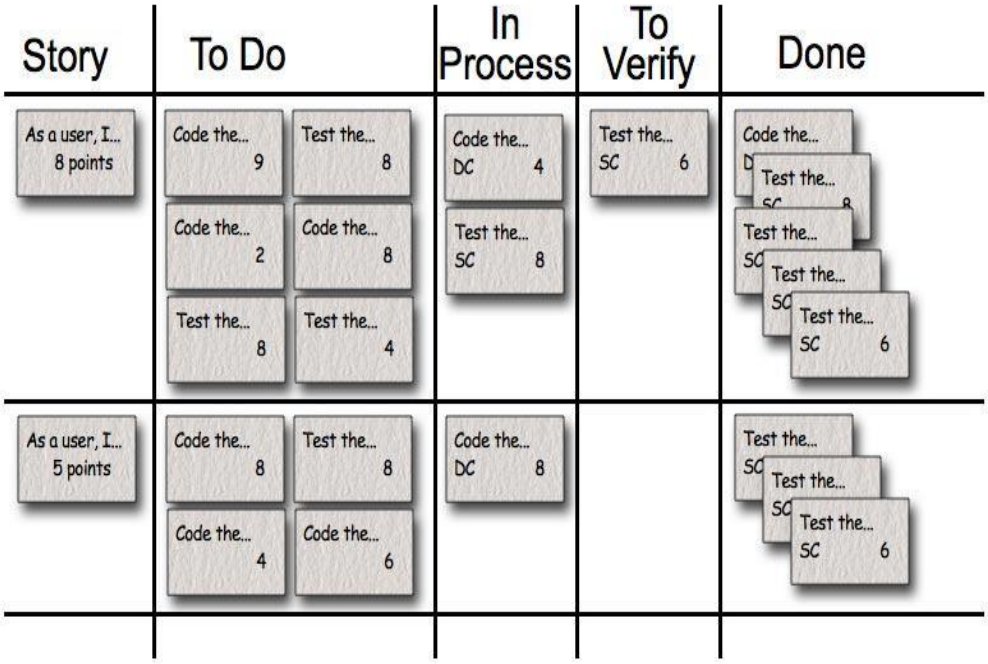


Figura 7. Exemplo de Task Board

Fonte: https://www.mountaingoatsoftware.com/agile/scrum/task-boards

- O incremento, de acordo com o Scrum Guide [21], é a soma de todos os itens do *Backlog* do Produto completados durante a *Sprint* e o valor dos incrementos de todas as *Sprints* anteriores. Ao final da *Sprint* um novo incremento deve estar “Pronto”, o que significa que deve estar na condição utilizável, atendendo à definição de “Pronto” do Time *Scrum*.

1.2 **Lean**

Nos primeiros anos após à Segunda Guerra Mundial, um engenheiro da Toyota, chamado Taichi Ohno, implantou o sistema puxado, inspirado num sistema de estoque de supermercado.

O *Lean Thinking* ou pensamento enxuto prega que seja produzido o que o cliente deseja, na quantidade solicitada, no prazo acordado e a um preço justo. As restrições a serem usadas são basicamente tempo, custo e valor.

A figura 8 ilustra as restrições básicas sob a luz do pensamento *Lean*.

Figura 8. Restrições do pensamento *Lean*

Além desses princípios, não se pode deixar de frisar alguns princípios do *Lean* definidos por POPPENDICK e POPPENDICK [16], que são:

• Construir com integridade;

• Respeitar as pessoas;

• Entregar rápido;

• Ver o todo.

Já WOMACK e JONES [24] proporam cinco princípios enxutos propostos originalmente pensados sob a ótica da manufatura, facilmente se adaptam para toda a organização. Esses princípios, segundo eles, são:

- Especificar e aumentar o **valor** dos produtos sob a ótica do cliente.
- Identificar a **cadeia de valor** para cada produto e remover os **desperdícios.**
- Fazer o valor **fluir** pela cadeia.
- De modo que o cliente possa **puxar** a produção.

- Gerenciando rumo à **perfeição.**

Desperdício é tudo aquilo que não agrega valor ao produto ou ao serviço, adicionando tempo e custo. Na manufatura existem sete, segundo OHNO [11]: superprodução, espera, transporte, processos desnecessários, movimentação, defeitos e estoque. Há mais três fontes de desperdícios no desenvolvimento de produtos: reinvenção, falta de disciplina e falta de integração da tecnologia da informação. Esses desperdícios são um sintoma e não a causa raiz do problema, que evidenciam os pontos de problema do sistema nos processos e nos níveis da cadeia. Então, os desperdícios precisam ser eliminados ou, no mínimo, reduzidos.

O *Lean* usa o princípio do *Just in Time*, que significa que cada processo tem que ser suprido com os itens necessários, no momento certo, na quantidade necessária e no local certo e por isso nada deve ser decidido, iniciado ou produzido. A viabilidade do *Just in Time* depende de três elementos relacionados: fluxo contínuo, *Takt time* e produção puxada:

- O fluxo contínuo é a resposta à necessidade de redução do lead time de produção, no limite, um fluxo unitário, onde não há estoques em processo.
- O *Takt Time* é o tempo necessário para produzir um componente ou um produto completo, baseado na demanda do cliente. Em outras palavras, o *Takt Time* associa e condiciona o ritmo de produção ao ritmo das histórias de usuário. Também conhecido como cadência ou ritmo, essa métrica do *Lean* é calculado segundo mostra a figura 9:

Figura 9. Cálculo do *Takt Time*

- Na lógica da “produção puxada”, o fornecedor produzirá somente quando houver demanda de seu cliente, podendo ser viabilizada através do *Kanban*, um sistema de sinalização entre cliente e fornecedor que informa ao processo fornecedor exatamente o que, quanto e quando produzir.

Outras Métricas do *Lean* são:

- *Lead Time*: Tempo gasto para se entregar uma funcionalidade/história de usuário desde que sua solicitação é feita pelo cliente. É calculado, como mostra a figura 10:

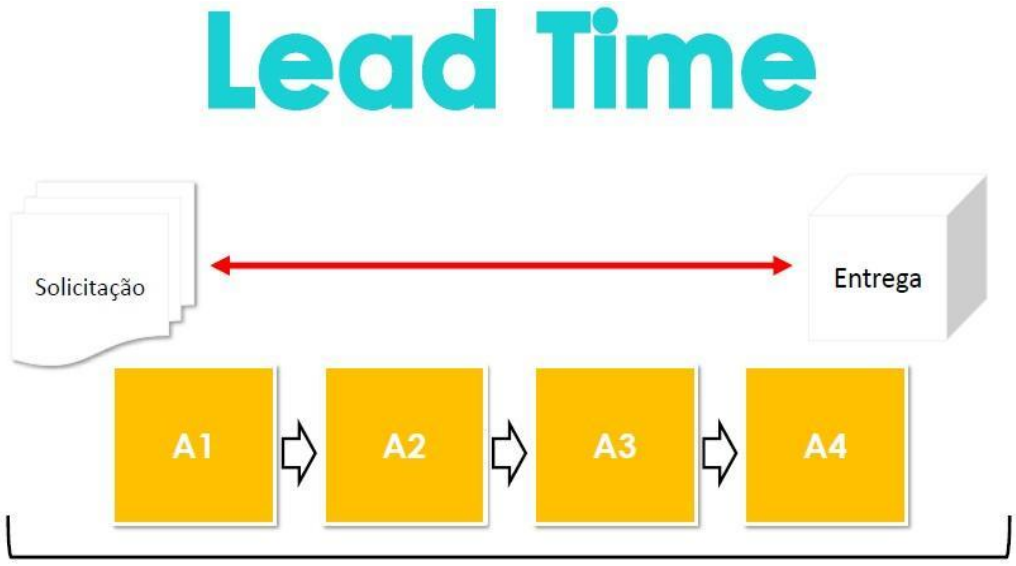


Figura 10. Ilustração do *Lead Time*

- Eficiência do Processo: O quanto, em termos de atividades que agregam valor num determinado *Lead Time* total. É calculado, segundo a figura 11:

Figura 11. Cálculo da Eficiência do Processo

- *Througput*: Ou vazão, é o cálculo da quantidade de histórias ou funcionalidades que são desenvolvidas em um determinado período de tempo. Calcula-se assim, como mostra a figura 12:

Figura 12. Cálculo do *Throughput*

O termo *Jidoka* se refere à interrupção do processo ao identificar alguma anormalidade, e a idéia é a de evitar a propagação de erros e desencadear um esforço em conjunto para resolver o problema.

Outro termo de *Lean* é o *Poka Yoke*, que se baseia em dispositivos à prova de erros e quando os erros são identificados eles não se transformam em defeitos, e as suas causas são eliminadas. Objetiva-se evitar a execução de inspeções, checagens sucessivas e auto-inspeções.

Outro termo do *Lean* é o *Kaizen*, em japonês, significa “mudança para melhor”, utilizado pelas empresas com o significado de “melhoria contínua”. Existe o *Kaizen* sistêmico, que visa melhorar o fluxo de valor do produto como um todo e o *Kaizen* de processo ou pontual, que se busca reduzir os desperdícios em áreas específicas do fluxo. O kaizen sistêmico também pode ser chamado de *Kaikaku* que por sua vez significa “mudança radical”. A forma como é aplicada a melhoria contínua depende das experiências de cada empresa, incluindo aquelas de Tecnologia da Informação, já que podem iniciar com um *Kaizen* Sistêmico ou Pontual. Para ambos os casos é necessário utilizar uma ferramenta: Mapeamento do

Fluxo de Valor (MFV). Esta ferramenta vai direcionar as melhorias para que elas não sejam perdidas. Por exemplo, imagina-se que o lead time, o tempo que uma empresa de TI leva para entregar uma funcionalidade crucial de um sistema de software, passando por todos os processos necessários, seja de duas horas e que a organização gaste semanas de trabalho para reduzir um minuto desse tempo com o uso do *Kaizen* Pontual. Depois se percebe, após a análise do fluxo de valor, que esse processo não seja mais necessário para a entrega da funcionalidade. Neste caso, todo o trabalho de otimização envolvendo pessoas, programas de computador, etc., seria perdido. Por este motivo, antes de iniciar uma melhoria pontual, tem que conhecer toda cadeia de processos de transformação do produto, ou seja, todo o fluxo de valor, de ponta a ponta e certificar-se de que eles são obrigatórios e não podem ser modificados, unificados ou eliminados. Empresas que nunca usaram o *Kaizen* sugerem-se iniciar com o *Kaizen* Pontual em um único produto para que, por meio do aprendizado, expandir para todos os fluxos da organização. Usando o MFV, a companhia tem a visão da transformação do produto como um todo e reduz-se a possibilidade de algumas melhorias, que são realizadas em alguma parte do processo, sejam perdidas.

Num ambiente *Lean* TI, para que o *Kaizen* possa ser utilizado, é necessário saber se esta empresa utiliza a Tecnologia da Informação como apoio ou suporte ao negócio ou como produto. No primeiro caso, o MFV deve ser feito a partir de um produto da organização que é suportado pela TI. Agora, para o segundo caso, por exemplo, numa fábrica de software, o MFV pode ser utilizado para identificar quais tipos de informações passam por cada departamento, organizando as pessoas fisicamente de acordo com o processo. Neste caso, a meta do *Kaizen* Sistêmico é aproximar os colaboradores a criar fluxo na informação, fazendo com que ela chegue mais rápido ao cliente. Além do *Kaikaku*, o caso de *Kaizen* de processo aplicado em TI pode ser entendido como melhorias diárias para atender aos objetivos da empresa, sejam nos sistemas dela, sejam nos sistemas desenvolvidos pelos clientes, no processo de trabalho individual ou da equipe.

A aplicação do *Kaizen* em uma empresa exige esforço e habilidade em saber o que está sendo feito é o correto. O responsável pelo produto também deve alinhar

essas melhorias com a estratégia da empresa. Até o *Kaizen* Pontual deve ser aplicado cuidadosamente, pois deve ser feito apenas depois de um processo ser estável e padronizado. Sob o ponto de vista dos negócios da empresa, deve-se saber: se o uso do *Kaizen* está alinhado à estratégia da companhia, se o fluxo de valor é enxuto e se o trabalho padronizado foi estabelecido para cada área do processo, caso contrário, a melhoria pode ser desnecessária, transformando-se em desperdício. Quando se fala em melhoria, é levantada a questão do uso do *Kaizen*, mas tem que entender que a melhoria contínua não deve ser feita apenas em momentos de crise, mas sempre. Grandes empresas como a Toyota usam *Kaizen* no seu dia a dia, tentando encontrar problemas para melhorar e mais, mostrando o verdadeiro espírito da cultura enxuta. Ou seja, a mudança não se faz de fora para dentro, mas de dentro para fora.

COSTA e JARDIM [5] disse: o *Lean Thinking*, tradução para pensamento enxuto, é uma maneira de você pensar a melhoria e a reorganização de um ambiente produtivo. A aposta-chave é que entendendo o que é valor para o cliente você será capaz de identificar e eliminar os desperdícios via o melhoramento contínuo dos processos de produção, e assim alavancar a sua posição competitiva, em particular no que se refere a fatores como a *velocidade* no atendimento aos clientes, a flexibilidade para se ajustar aos seus desejos específicos, a qualidade e o preço do produto ou serviço ofertado.

Mas o que é valor e o que é desperdício? De acordo com OHNO [11], valor é o que os clientes querem, na quantidade necessária, com grau de qualidade desejada a um preço justo.

Exemplo de agregar valor ao produto: se a funcionalidade de um *software* a ser produzida faz parte do negócio ao sistema, ela agrega valor ao cliente e, portanto, deve ser otimizada. Mas se determinada funcionalidade ou recurso do sistema o cliente não pagaria por ela, mas se faz necessário ao negócio, ela vai agregar valor ao negócio e deve ser minimizada. Caso contrário, se trata de desperdício, ou seja, o cliente não requeira a funcionalidade e nem o recurso é necessário pelo negócio devendo, assim, ser eliminado.

E desperdício? Segundo OHNO [11], desperdício é qualquer coisa além do mínimo de equipamento, materiais, peças, espaço e tempo do operador que sejam absolutamente essenciais para agregar valor ao produto. A seguir relataremos sete principais casos de desperdícios, que podem ocorrer em empresas de TI:

1. Defeito: O quanto se perde recursos e tempo em busca de consertar erros ou descartando produtos?
2. Produção Excessiva: As atividades devem ser realizadas para atender aos anseios do cliente, pois ele paga, ele deseja e qualquer atividade excessiva vai acarretar em acréscimo de recursos e de tempo.
3. Transporte ou Troca de Tarefas: Tempo de troca ou de adaptação quando alguém sai de um processo e entra no ritmo de outro causa desperdício de tempo.
4. Espera: É qualquer atraso desde o final da atividade de um processo e o início da próxima atividade.
5. Estoque: É o trabalho aguardando, em progresso, adiantado ou inacabado. Para este item, é melhor ter uma atividade concluída do que dez inacabadas.
6. Processamento Extra: Realizar ou executar mais tarefas ou atividades do que o necessário para chegar ao resultado. Aqui pode ser feita a seguinte pergunta: Quantas atividades fazemos que é realmente necessário e que agrega valor ao cliente ou ao negócio?
7. Locomoção: É o tempo gasto na movimentação de pessoas pela busca de informações. Ou seja, quanto tempo é utilizado para conseguir encontrar uma informação crucial para o seu trabalho?

Exemplos de desperdícios, listados nos principais casos, são dados a seguir:

1. Tarefa inapropriada e desnecessária.
2. Reuniões e assinaturas.

3. Processos que satisfazem apenas objetivos de curto prazo, mas não agregam valor ao cliente.
4. Esforço para corrigir resultados imprevisíveis devido a causas desconhecidas.
5. Inspeções e retrabalhos.
6. Esforço necessário para transferir informações ou materiais por falta de integração entre os processos e até mesmo dentro dos processos.
7. Esforço utilizado para criar informações incorretas e para lidar com suas conseqüências.
8. Recursos despendidos em processos secundários que ainda não podem ser utilizados.
9. Falta de foco, toda vez que a atenção não está voltada para atingir os resultados.
10. Áreas com objetivos desalinhados aos objetivos estratégicos da organização.
11. Áreas de trabalho desorganizadas, com excesso de material e de difícil acesso.
12. Layout de salas inadequadas.
13. Ruídos e outros fatores que causam a dispersão do pessoal.

Outro conceito utilizado no *Lean* é o Fluxo de Valor, que é um caminho que envolve todas as etapas de fornecimento, mesmo as que não agregam valor, necessárias para gerar um resultado, desde a solicitação até a entrega do produto.

Já o Mapeamento do Fluxo de Valor (MFV) é a elaboração de um mapa que exibe o fluxo de informações ou materiais. Por meio da análise do mapa de fluxo de valor é possível observar quais as etapas que agregam e retiram valor do produto, identificando gargalos e atrasos nos processos produtivos, propor melhorias de processos e visualizar onde é possível aplicar ferramentas para reduzir os

desperdícios, aumentando a eficiência produtiva. Faz-se, enfim, o estado presente do processo ou da empresa, o As-Is, e onde se pretende chegar, o To-Be.

No MFV, os processos são divididos:

- Aqueles que geram valor;
- Os que não geram valor, mas são importantes para a manutenção da qualidade;
- Os que não geram valor e que devem ser evitados ou eliminados.

Geralmente há cinco passos para criar os Mapas de Fluxo de Valor:

1. Identifique a família de produto e o processo a ser mapeado: Qual a família dos produtos é realmente importante mapear? Isso deve ser identificado neste passo e é fundamental focar nos esforços primeiramente nas áreas mais críticas.
2. Desenhe o processo atual: É importante o envolvimento dos funcionários envolvidos no processo para desenhá-lo. Aqui tem que ser identificado as etapas, pontos de início e de fim do processo, informações de fornecedores, tempos envolvidos, etc. Identificar também as etapas que agregam valor e as que não agregam valor. Esta fase retrata o atual momento do processo e não o processo ideal.
3. Avaliar o fluxo de valor atual: Aqui, algumas perguntas têm que ser respondidas ao cliente. Elas são:
    a. Esta etapa do processo agrega valor ao cliente?
    b. Quanto e quais os recursos que estão sendo utilizados?
    c. Quais desperdícios estão sendo gerados?
    d. Dizer quais processos de fabricação, medidas e tratamentos que agregam valor?
    e. Qual é o tempo de espera entre os processos de fabricação?

f. Como reduzir ou eliminar os gargalos no processo?

g. Como implementar mudanças?

4. Criar o estado futuro do Mapa de Fluxo de Valor: Aqui a participação de toda a equipe é novamente fundamental, pois é feito o esforço para criar o fluxo ideal para o processo, o fazendo fluir melhor reduzindo os desperdícios e a espera entre as etapas.

5. Criar o plano de ação: Observando o funcionamento dos processos hoje e como gostariam que ele fosse feito, é hora de criar um plano de ação. Há um sem número de templates voltados para isto. Pode-se utilizar as ferramentas *Kaizen*, Pensamento A3, 5Ws, DMAIC, etc. O objetivo aqui é fazer com que a equipe entenda quais ações são importantes e o momento que elas devam ocorrer para levar o estado atual para o estado futuro.

Com estes passos do MFV, objetiva-se chegar à perfeição no fluxo de um processo que tem os seguintes requisitos: Lead Times curtos, maior vazão, menor takt time, processo enxuto e alto valor agregado.

Ganhos reais tem sido relatados na literatura e também empiricamente relacionados ao uso da filosofia *Lean* e a conseqüente diminuição dos tempos totais de atendimento, redução dos índices de falhas, redução dos custos de produção, dentre outros indicadores de produtividade e qualidade.

Normalmente, o *Lean Thinking* é identificado e mostrado segundo 5 passos, que são:

1. Identificar o que é valor para o cliente

Segundo OHNO [11], valor é o que os clientes querem, na quantidade necessária, com grau de qualidade desejada a um preço justo e desperdício é qualquer coisa além do mínimo de equipamento, materiais, peças, espaço e tempo do operador que sejam absolutamente essenciais para agregar valor ao produto.

Segundo a visão *Lean*, a melhor maneira de identificar os desperdícios é você se colocar na posição do seu cliente e fazer uma reflexão crítica dos processos de produção, na maneira como são feitos hoje. Verificar quais as tarefas que são úteis ao cliente e se elas são realizadas, em favor dele. Ou as atividades são feitas em função de hábitos históricos, ou até mesmo pelo simples conforto em realizá-las de uma maneira aparentemente mais fácil.

2. Mapear o fluxo de produção e identificar os desperdícios

Aqui se deve pensar em reformar o sistema de produção para agregar valor ao cliente e utilizando o MFV, identificar nos processos quais desses processos ou recursos agrega valou ao cliente ou ao negócio. Desenhe ou utilize alguma ferramenta para desenhar um fluxo representando as atividades atuais ou o fluxo que seja mais relevante para o resultado do sistema. Em seguida, analise com a informação ou material percorrem este fluxo, ficando atento à duração neste percurso. Quais atividades ou recursos são realmente necessários ou produtivos? Sob a perspectiva do cliente, quais atividades deveriam ser eliminadas?

Uma coisa a ser observada é que os tempos gastos em filas, retrabalhos, inspeções, controles, etc., podem ser necessidades atuais do sistema, mas não interessa em nada ao cliente, ou seja, não agrega valor a ele. E que poderia não haver tais atividades, caso lhe fosse entregue de acordo com as especificações desejadas de qualidade e de preço.

Com o seu mapa com a situação atual faça agora a projeção do fluxo desse mapa na situação ideal. Não inicie seu pensamento pelas escolhas do passado. Com o mapa atual e o mapa ideal, vamos definir agora o mapa de fluxo possível, dentro das condições existentes, procurando envolver a equipe a desenvolver estas atividades e fazendo com que a sua participação gere um sentimento de co-autoria, colaboração, consolidando as mudanças.

Se o raciocínio de ideal começar pelas restrições de pessoas, dinheiro, tempo e de equipamentos, essa visão tende a ser conservadora. Por causa disso, não deixe de pensar livremente o sistema ideal, ainda que momentaneamente inalcançável, sem perder a chance de denominar de desperdícios os tempos, obstáculos e retenções do fluxo que de alguma forma atrasam a entrega dos produtos e serviços ao cliente, embora seja impossível removê-las atualmente. A visão deste alvo ideal é que dá norte e gás ao processo de *Kaizen*.

3. Implantar o Fluxo Contínuo

Para implantar o fluxo contínuo deve-se "enfrentar" os lotes e os departamentos, pois até eles podem ter sido implantados por nós mesmos no passado para compensar custos e restrições reais e existentes. Os lotes referem-se a quantidade de itens que vão ser processadas de uma só vez. Deve-se processar uma quantidade de itens de forma sustentável, pois se processa muitos itens, ou acaba fazendo muito de uma coisa e deixando de fazer outras exigidas pelo cliente, ou fazer o produto errado, pois se baseando em previsões, o produto foi feito errado e que agora o cliente não quer mais. E se processar poucos itens o suficiente, acaba fazendo com que seja preparado o seu recurso para cada pequeno lote e isso não agrega valor ao cliente. Já os departamentos são os setores de uma determinada organização e qualquer existência de burocracia em algum desse departamento vai acarretar uma queda da rapidez, do custo e da qualidade. Lembrando que se existir alguma atividade burocrática que seja inerente ao negócio, ela deve ter seu impacto minimizado. Para ir à direção do valor para o cliente, é preciso alterar as razões que são feitos para trabalhar com lotes grandes para lotes mais sustentáveis e é preciso usar a cabeça para minimizar o custo de troca, fazendo com que o sistema se torne mais fluente e sustentável.

4. Implantar o Sistema Puxado pelo Cliente

Não é simples transformar a cultura de gestão de um ambiente de produção nos dias atuais. Mas se conseguir transformar o sistema, eliminando os desperdícios, produzindo em lotes mais sustentáveis, num fluxo contínuo ou bem próximo disso, está preste a implantar o sistema puxado.

Sendo veloz ao atender a uma demanda, então, ao invés de buscar advinhar o que ocorrerá amanhã, você poderá se dar ao luxo de aguardar a chegada do pedido e só então disparar a produção. Essa é uma vantagem competitiva e que dá a você a leveza para atender imediatamente às necessidades dos clientes. E, conseqüentemente, gerar muito mais valor para a equipe, para o sistema e para todo o sistema e ao que está sendo entregue.

5. Buscar a perfeição

Como o sucesso na filosofia Lean depende fundamentalmente das pessoas, nesta cultura, elas são o início (a base da transformação), o meio (o instrumento) e o fim (o objetivo). Esta mudança se pauta na responsabilização, no desenvolvimento técnico e na autonomia das equipes envolvidas. Pois são as pessoas que são responsáveis pela geração de valor.

São propostos pelo Lean para as ações diárias e ofertar valor para os clientes alguns dos vários instrumentos ou meios:

- Grupos e ferramentas para a melhoria contínua.
- Gestão visual e semi-autônoma.
- Autogestão do desempenho cotidiana.
- *Feedback* freqüente.
- Resposta rápida.

WOMACK e JONES [24] registraram com sua experiência de anos junto a empresas que seguiram este caminho: “medida que as organizações

começam a especificar valor com precisão; identificam o fluxo de valor total; à medida que vão transformando o seu sistema na direção do fluxo contínuo e deixa que o cliente puxe a sua produção, algo muito estranho começa a ocorrer. Ocorre aos envolvidos que o processo de redução de esforço, tempo, espaço, custo e erros é infinito.”

Um exemplo clássico é o da troca de pneus de um carro de formula 1. As equipes devem ter demorado alguns minutos nas primeiras tentativas e como elas estudaram este processo, o tempo de troca foi diminuindo para o que é hoje, graças a uma melhoria contínua, que foram vários ciclos, por sinal. Essa melhoria contínua, obviamente, surgiu da necessidade de agilizar as trocas dos pneus e estas trocas de pneus foram necessárias, pois sem elas o carro perderia bastante em aderência.

1.3 **Kanban**

Em japonês, significa cartão, é uma ferramenta de controle visual para o acompanhamento de todo o trabalho durante seu desenvolvimento, por meio da visibilidade da fluência deste trabalho através de várias etapas do processo de desenvolvimento do produto. Esta idéia foi criada originalmente pela Toyota. O *Kanban* usa um mecanismo de controle visual para acompanhar o trabalho à medida que ele flui através das várias etapas do fluxo de valor.

A figura 13 mostra um exemplo de quadros *Kanban*.

Figura 13. Exemplos de quadros *Kanban*

O *Kanban* é baseado numa idéia em que as atividades em andamento devem ser limitadas, por meio de um mecanismo chamado de *work in progress* (WIP). Algo novo só deve ser iniciado quando uma peça de trabalho existente é liberada ou quando uma função automática inicia isso. O *Kanban* parece ser uma mudança pequena e, no entanto, muda tudo a respeito de uma empresa.

Kanban não é um processo ou ciclo de vida de gerenciamento de projetos ou de desenvolvimento de *software*. O *Kanban* é uma abordagem para introduzir mudanças em um ciclo de desenvolvimento de *software* ou metodologia de gerenciamento de projetos. O princípio do *Kanban* é que você inicia com o que estiver fazendo agora. Você entende seu processo atual ao mapear o fluxo de valor e ao concordar, em seguida, em limitar as atividades em andamento (do inglês WIP) para cada estágio desse processo. A partir daí você começa a rastrear as atividades pelo sistema para iniciá-las quando os sinais do *Kanban* aparecerem.

O *Kanban* é útil para equipes ágeis de desenvolvimento de *software*, mas também tem ganhado popularidade, em equipes que utilizam uma abordagem tradicional. Ele está sendo introduzido como parte da iniciativa *Lean* (produção enxuta) para mudar a cultura das organizações e encorajar o uso da melhoria contínua. Ele é menos prescritivo do que as outras metodologias de desenvolvimento já apresentadas. O que não impede do *Kanban* utilizar alguma ferramenta, desde que agregue valor ao negócio. O que o Kanban restringe, segundo KNIBERG e SKARIN [10], é a visualização do fluxo de trabalho, usando o quadro *Kanban* e limitar o trabalho em progresso, o WIP, exibido também no quadro.

As figuras 14, 15, 16, 17 e 18 mostram exemplos de um quadro *Kanban* sem o WIP.

Figura 14. Exemplo de quadro *Kanban* sem WIP

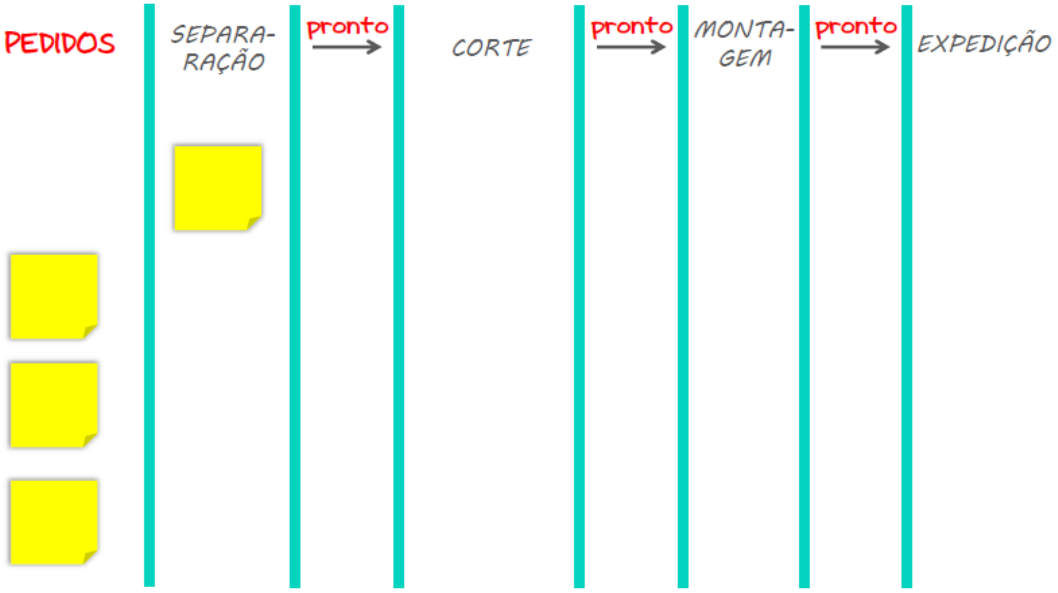


Figura 15. Exemplo de quadro *Kanban* sem WIP

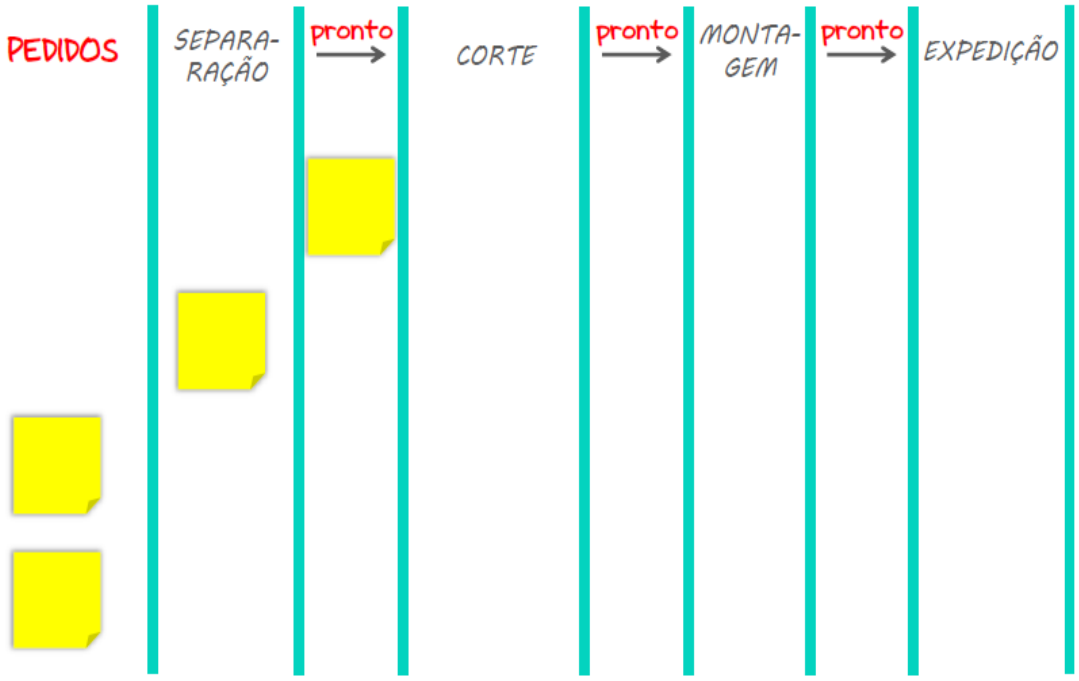


Figura 16. Quadro *Kanban* sem WIP

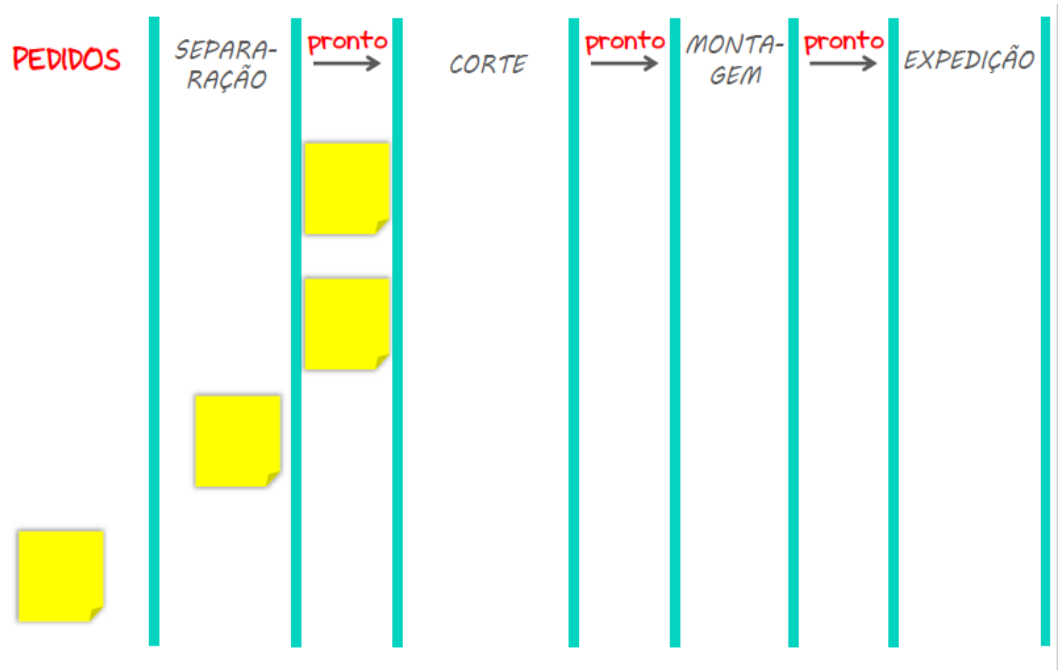


Figura 17. Quadro *Kanban* sem o WIP

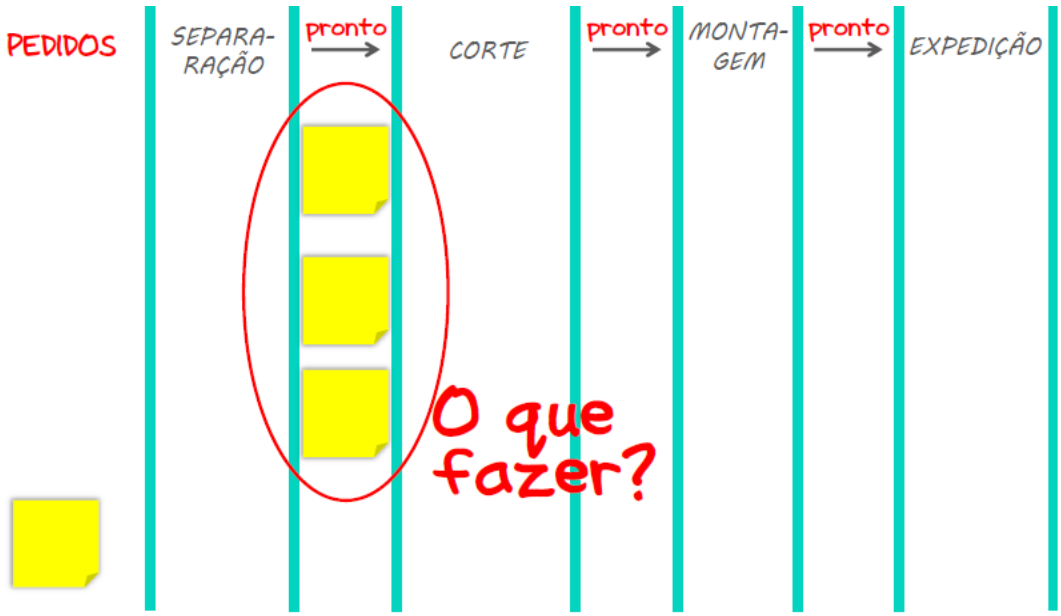


Figura 18. Quadro *Kanban* sem o WIP

Como o WIP é limitado num sistema *Kanban*, tudo que ficar bloqueado por qualquer motivo vai parar o sistema. Se certa quantidade de itens de trabalho fica bloqueada, todo o processo *Kanban* para. Isso cria a necessidade de concentrar toda a equipe e toda a empresa para solucionar o problema com o objetivo de

desbloquear o item e restaurar o fluxo de produção. Assim, a figura 18 mostra um exemplo de quadro *Kanban* agora com o WIP inserido.

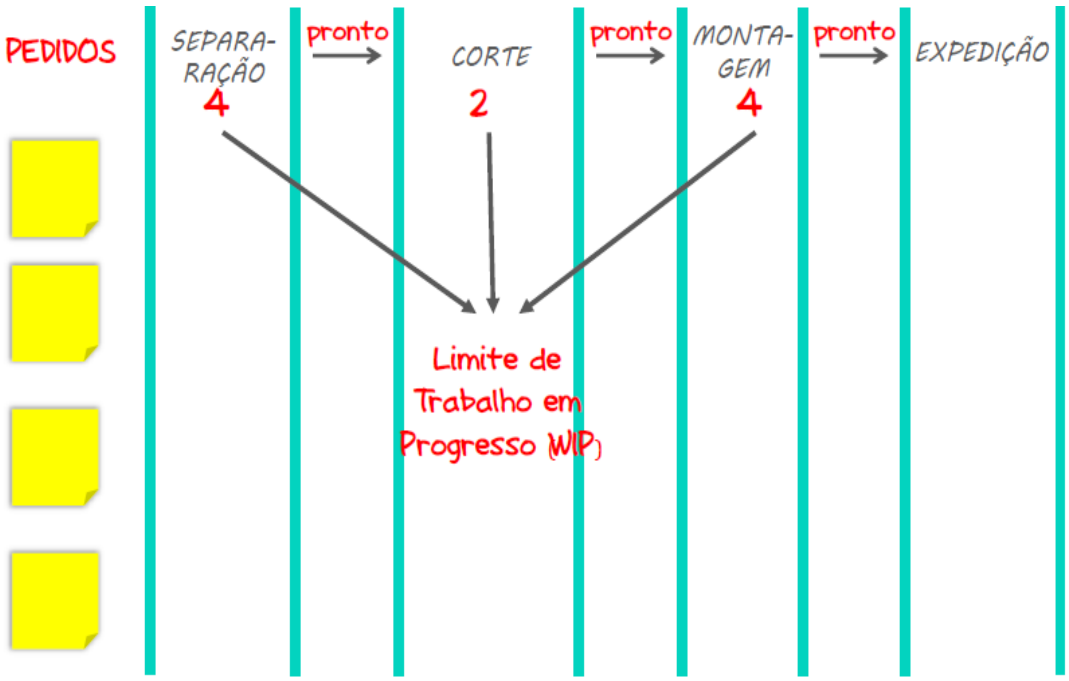


Figura 19. Quadro *Kanban* com o WIP

Para a visualização de todo o trabalho do *Kanban*, usa-se um quadro branco com post-its, ou um sistema de cartões eletrônicos. A transparência deste sistema também contribui para a mudança cultural. Métodos ágeis são bons provendo transparência sobre as atividades, tanto as concluídas como as em andamento e reportando métricas como velocidade (a quantidade de trabalho finalizado em uma iteração).

O *Kanban*, no entanto, consegue ir a um passo além e fornece transparência ao processo e o seu fluxo, expondo gargalos, filas, variabilidade e desperdício e tudo que impacta no desempenho da organização em termos de quantidade de trabalho de valor entregue e o tempo de ciclo necessário para entregá-lo. Ele proporciona aos membros da equipe e às partes interessadas externas a visibilidade sobre os efeitos de suas ações ou alguma falta delas. Sendo assim, os primeiros estudos de

caso estão mostrando que o *Kanban* muda o comportamento e incentiva uma maior colaboração no trabalho.

A visibilidade dos gargalos, desperdício e variabilidade, e seus impactos, também incentivam a discussão sobre melhorias e as equipes iniciam as melhorias nos seus processos, fato visto claramente no *Kanban* e como resultado, ele encoraja a evolução incremental dos processos existentes, fenômeno que é alinhado aos valores de *Lean* e de *Agile*. O *Kanban* não exige uma revolução no que se refere na maneira como as pessoas trabalham; ele estimula uma mudança gradual, que é entendida e aceita por consenso entre as partes interessadas. O *Kanban* possui a natureza de um sistema puxado, ou seja, são os times que puxam o trabalho, de acordo com sua capacidade e não o oposto. Encoraja o comprometimento tardio de produção, tanto em priorização de trabalho novo quanto na entrega de trabalho existente.

Tipicamente, os times vão concordar em uma cadência de priorização para atender partes interessadas que estejam contra a maré e decidir a próxima coisa em que trabalhar. Estas reuniões podem ser feitas regularmente porque são normalmente muito curtas.

Kanban também possui o efeito de limitar o WIP para dar previsibilidade de tempo em ciclos e fazer as entregas mais confiáveis. O mecanismo de "parar a linha de produção" para superar os obstáculos e os erros encontrados, também encoraja níveis mais elevados de qualidade e uma queda rápida de retrabalho.

O Kanban usa também a filosofia *Just in Time*, cujo objetivo é identificar e eliminar todos os desperdícios na produção. Desperdício é quando são utilizados recursos, tempo e equipamento mais que o necessário para se produzir um produto na unidade de produção. Este sistema funciona como um quebra-cabeça, que contém várias ferramentas com o intuito de melhorar a produtividade, eliminando desperdícios. Quando o *Kanban* é aplicado sozinho, apenas limita os níveis de estoques ou *Backlog*, servindo para mostrar os problemas e a solução desses problemas deve ser alcançada com o uso ou aplicação das ferramentas do *Just in Time*.

A idéia básica transmitida pelo caminho do *Just in Time* é a redução gradativa do estoque com o qual se trabalha. As primeiras reduções podem não causar problemas, a redução continua, até que apareça algum problema. O problema exposto pela redução dos estoques é tratado e a redução dos estoques prossegue até atingir um nível no qual seja possível trabalhar somente com o estoque mínimo (indispensável à fabricação do produto). O *Just in Time* é freqüentemente confundido com estoque zero, na verdade a condição “estoque zero” é uma conseqüência de uma ação bem sucedida da filosofia do *Just in Time*.

No sistema *Kanban*, torna-se possível visualizar os problemas mais facilmente e esta é a grande vantagem apresentada por este sistema. Mas não basta localizar os problemas, mas sim é importantíssimo resolvê-los de imediato, pois qualquer falta de material é vital para o processo de fabricação. É muito mais fácil uma linha de produção parar por falta de material no sistema *Kanban*, do que no sistema tradicional de abastecimento.

Um dos itens usados no *Kanban*, o cartão *Kanban* é responsável pela comunicação e pelo funcionamento desse sistema e nele devem estar contidas as informações mínimas para o bom funcionamento da linha de produção. Se for o caso, ele poderá conter mais informações, desde que agregue valor para a área específica, onde se pretende implementar o sistema *Kanban*.

O quadro *Kanban* é a ferramenta visual e que nela ocorre toda a produção para uma área específica, para o caso de *software*, a produção são as funcionalidades do software. Este quadro deve ficar visível a todos, em uma posição estratégica onde é construído o produto. Uma coisa importante a se observar é a ordem em que estes cartões são preenchidos no quadro Kanban, devendo ser colocados na linha certa e na coluna certa. Primeiro são os cartões verdes, depois os amarelos e por último os vermelhos.

Existe o *Kanban* interno e o *Kanban* externo: o primeiro refere-se aos fornecedores que são de fora da empresa onde se trabalha e o segundo refere-se aos fornecedores que são de outros setores de uma empresa.

O número de cartões *Kanban* está diretamente relacionado com a velocidade de consumo na linha de montagem e com o tempo de reposição necessário ao ressuprimento dos lotes. O ideal é que ocorra o perfeito balanceamento entre a produção e o consumo, porque quanto maior for o equilíbrio entre o consumo e produção, menor será o tempo gasto no processo, sendo possível reduzir o número de cartões *Kanban*.

É importante observar que as peças que tenham prioridade de produção são aquelas cuja cor corresponde ao vermelho, que é considerada a faixa crítica do quadro *Kanban*. Isso significa que esses componentes deverão ser produzidos em primeiro lugar. Em seguida, as de nível intermediário são as de cores amarela e por fim, as de menor importância são representados pela cor verde.

Para o bom funcionamento da metodologia *Kanban*, são importantes algumas dicas. Se estes cuidados não forem observados o sistema pode não ser vantajoso. A seguir as recomendações:

- Quadro vazio - quando o quadro *Kanban* estiver vazio, significa que o estoque necessário está completo e que não se deve ser produzido mais nada. Qualquer produção além da determinada no quadro será considerada superprodução que é desnecessária e trará acréscimos nos custos.
- Quadro cheio - Quando mais vazio estiver o quadro, maior será o estoque de peças prontas e vice-versa. Se o estoque de peças se aproxima do ponto crítico, que é o limite WIP, sendo que esse acontecimento ocorre com freqüência, é importante a participação de todos os responsáveis, no sentido de auxiliar na identificação das causas que estão provocando esta situação. Pois, segundo o caminho do Just in time, ao trabalhar com estoques menores os problemas surgem e tem que serem solucionados ao invés de ficarem encobertos.
- Perda ou extravio do cartão – Cuide bem dos cartões, pois a perda de apenas um cartão *Kanban* pode provocar a parada das linhas de

produção e quando esse cartão perdido não é reposto e colocado no quadro, implica a existência de material no estoque, quando na verdade não existe.

O *Kanban* possui vantagens para os funcionários e para a empresa. A seguir são ilustradas as principais vantagens de uso do *Kanban*:

• O *Kanban* é um sistema autocontrolado e extremamente simples de ser implementado.

• O *Kanban* elimina a necessidade de controles por meio de documentos formais, ele contribui para a desburocratização.

• O *Kanban* valoriza o colaborador, fazendo com que ele possa contribuir com sua experiência para o sucesso do sistema.

• O *Kanban* é um processo controlado pela produção.

• O *Kanban* limita e permite reduzir os estoques.

• O *Kanban* reduz os custos de fabricação.

• O *Kanban* tem baixo custo de implantação.

1.4 **Considerações Finais**

Este capítulo falou sobre as metodologias utilizadas para a gestão ágil de projetos. O Scrum possui um foco maior na gestão do projeto, com regras, artefatos e papéis bem definidos, como: *Time Box*, *Scrum Master*, *Sprint*, Reunião Diária. O *Lean*, que possui as diretrizes para uma gestão mais eficiente e eficaz da entrega dos produtos, incluindo as de *software* ao tomar medidas seguindo seus princípios, como *Jidoka*, *Kaizen*, *Lean Thinking*. Por fim o *Kanban*, que é uma nova metodologia, usando o WIP (Work in Progress), não usa estimativas nas atividades, mas usa previsão de duração delas. O *Kanban* utiliza alguns princípios do *Lean* para uso como metodologia ágil.

Capítulo 4

Sugestão de Melhorias na Gestão Ágil de Projetos

Neste capítulo exibimos os estudos de campo de alguns profissionais da área de Engenharia de *Software* e apresentaremos a sugestão de melhoria para a gestão ágil de projetos de *software*.

1.1 Estudo de Campo para a Gestão Ágil de Projetos

Nesta seção apresentaremos um estudo de campo feito para observar a eficácia da gestão ágil de projetos de *software* utilizando o *Scrum*. Ele foi feito com dois tipos de pessoas entidades: um é o CEO de uma empresa de uma Startup e o outro é uma aluna que utilizou *Scrum* num projeto de uma disciplina. A seguir vamos observar o resultado para cada um deles.

1.1.1 **CEO de uma Startup**

Este CEO utilizou o *Scrum* em um de seus projetos, pois atendia melhor as suas necessidades quanto à gestão do projeto analisado. A seguir, o que encontramos sobre o questionário respondido por ele.

1. Que mesmo a forma de trabalho sendo remoto, o que não é recomendável sob a luz da agilidade, o *Scrum* foi utilizado, pois todos da equipe conhecem, fazendo com que não possua nenhuma curva de aprendizado para a adoção deste modelo de gestão.
2. Que para exercer alguma liderança dentro da equipe faz-se necessário possuir qualificação técnica e da metodologia em uso, se possível, com certificações de liderança.
3. O uso do *Scrum* se dá para moldar, apoiar e guiar o projeto como um todo, seguindo as regras, utilizando os artefatos e cerimônias. As Sprints têm durações de duas semanas, onde é entregue o produto ou a funcionalidade pedida ou necessária. São feitas as reuniões semanais, em que acontecem as inspeções, por todos, com o intuito de verificar o que deu certo, o que deu errado e o que pode ser melhorado. As entregas passam pela checagem com o cliente. Quando a *Sprint* falha, são feitos ajustes na próxima, seja realinhando ela, seja aumentando o esforço para realizá-la ou até mesmo criando uma nova Sprint, objetivando obter sucesso nessa nova entrega.
4. A equipe, multifuncional, é formada por 5 pessoas, em que o CEO se comporta como *Scrum Master*, *Product Owner* ou até mesmo parte do time de desenvolvimento, pois se trata de uma *Startup*. Todos têm poder de autonomia, respeitando o perfil de cada um dentro do projeto *Scrum*, visando atingir os objetivos das *Sprints* e, conseqüentemente, do projeto. Eraldo se comporta mais como gerente líder.
5. É utilizado o Trello para apoiar o uso da metodologia e do projeto como um todo, tendo um campo exclusivo para impedimentos, nas situações que algum integrante da equipe se encontra em dificuldades para realizar determinada atividade sendo devidamente relatada ao *Scrum Master*. Na ferramenta online, é registrada toda a análise dos riscos do projeto como um todo, inclusive com estratégias para lidar com eles como aceitar, mitigar, transferir, compartilhar, aproveitar.

6. Como artefatos do projeto são gerados a documentação de negócios e os diagramas na arquitetura.

7. A otimização dos projetos é feita por reuso de código, programação em par e git hub, visando reduzir esforço e falhas, com a sua devida aplicação para as próximas *Sprints* ou projetos futuros.

1.1.2 **Ex-Aluna da Disciplina de Aplicações de Engenharia de Software**

Nesta subseção, a aluna da Disciplina de Aplicações de Engenharia de Software forneceu informações sobre o uso do *Scrum*, como Desenvolvedora de Software. Tiramos as seguintes conclusões:

1. A equipe de projeto da qual a aluna participa é pequena (3 desenvolvedores 1 *Product Owner* e 1 *Scrum Master*), onde todos possuem várias habilidades técnicas, exceto o *Product Owner*. A equipe geralmente entrega alguma funcionalidade entre 15 a 21 dias. Quando alguma entrega não é aceita, ela retorna para ser novamente criada para reparar erros.
2. Semelhante ao caso do CEO da Startup visto na subseção 1.1.1, há autonomia entre os membros da equipe de efetuarem as suas atividades e o *Scrum Master* exerce o papel de líder, orientando a equipe de projeto a efetuar os trabalhos de desenvolvimento.
3. Há a interação direta com os clientes que são tratados como usuários. São eles que validam as entregas.
4. O trabalho de otimização é feito, mas sem fuso de ferramentas específicas, ficando no relato dos participantes sobre o que gostou, o que não gostou e o que pode ser melhorado.

5. Para o caso específico da aluna, não há uma reunião diária formal, requisito no *Scrum*, mas quando chega ao ambiente de

trabalho é perguntado sobre as atividades e se teve algum problema com alguma delas. As estimativas para cada atividade são discutidas pela equipe.

1.2 **Sugestões de Melhorias da Gestão Ágil de Projetos por meio da**

Construção de um Kanban Simples e Visual e Incorporando-o ao Modelo Scrum

Nesta seção vamos mostrar a construção de um simples Kanban para incorporar ao modelo do *Scrum*. As etapas são:

1. Transcrever as etapas do fluxo de valor referente ao desenvolvimento do produto/funcionalidade, respeitando a ordem cronológica das etapas. A coluna representando a fila é o To Do daquele quadro *Task Board*, visto na seção 1.4 e o Pronto representa o Done. Veja na figura 20.

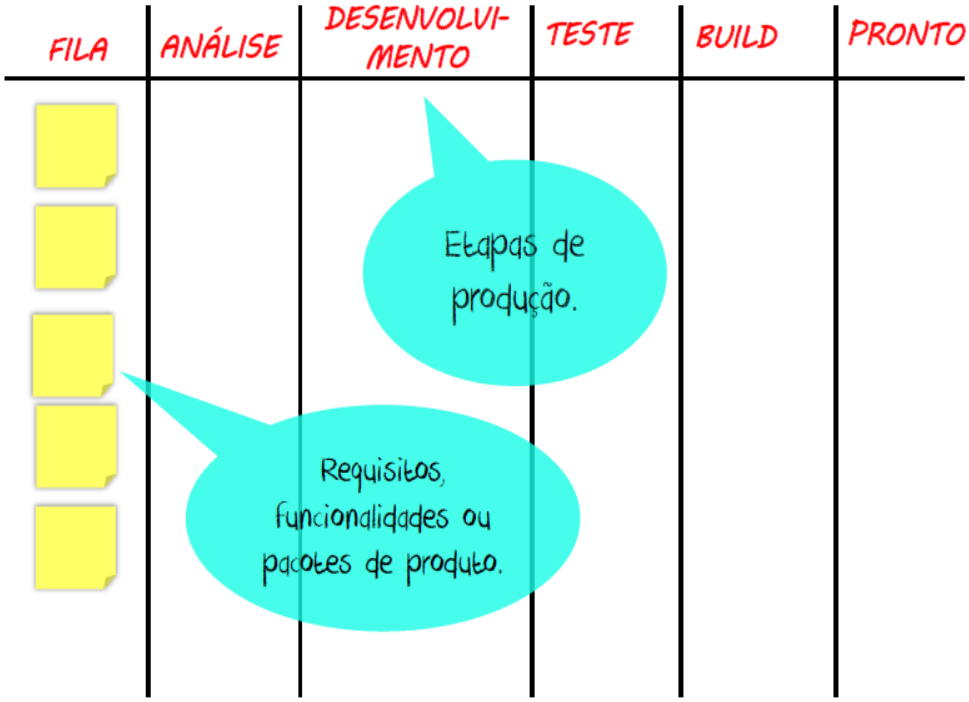


Figura 20. Montando o quadro *Kanban*

2. Estabelecer um limite, o *Work in Progress* (WIP) para cada uma das etapas descritas. A figura 21 mostra o quadro *Kanban* com o WIP.

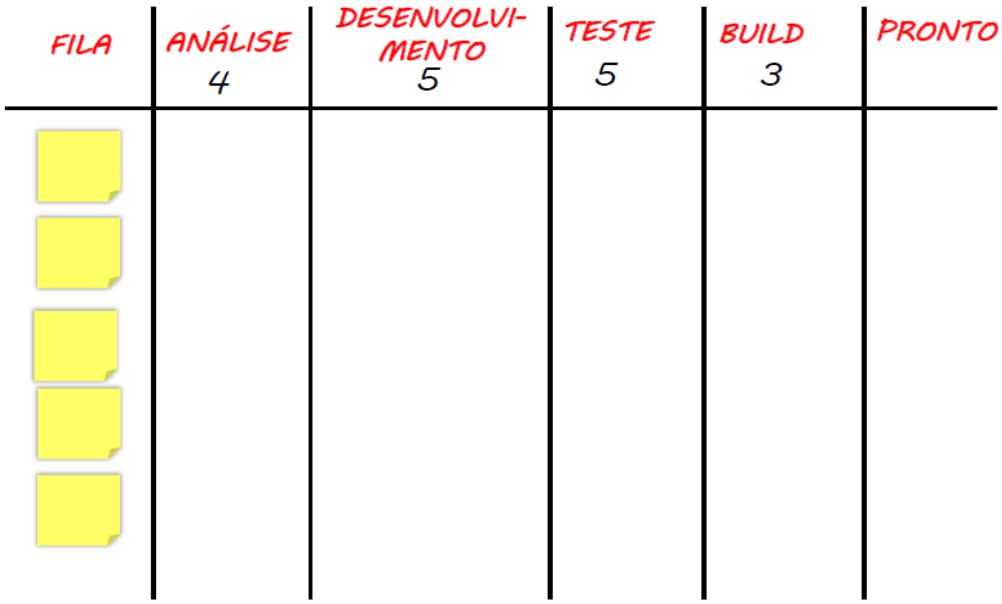


Figura 21. Quadro *Kanban* com WIP

3. Identificar os itens, o cartão *Kanban*, a serem desenvolvidos e incorporados ao seu produto. Exemplos de cartão *Kanban* nas figuras 22 e 23.

Figura 22. Exemplo de itens do quadro *Kanban*.

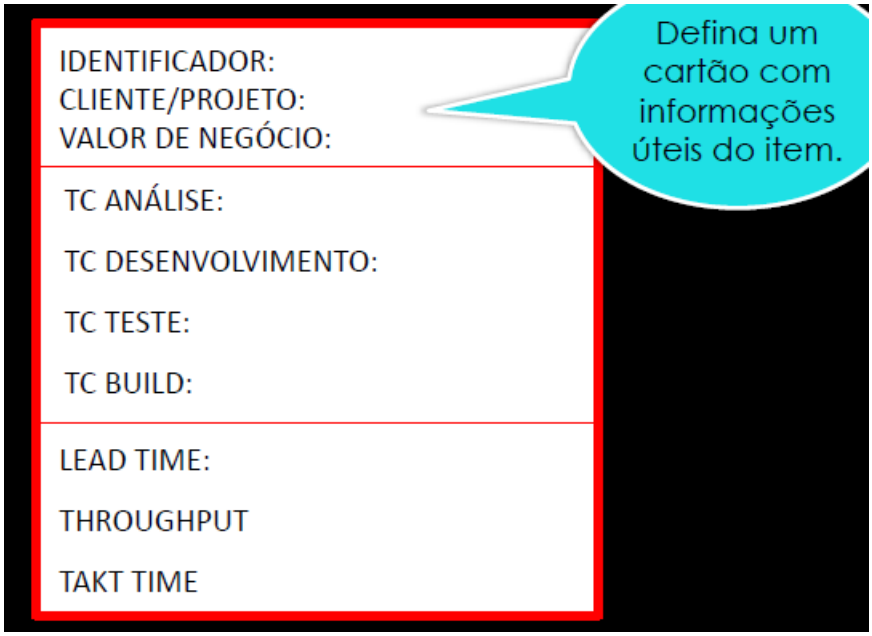


Figura 23. Exemplo de itens do quadro *Kanban*

Aqui deve ser frisado que, caso fizer sentido para o seu processo, estabelecer o valor de negócio referente a cada item ou cartão *Kanban*, junto ao cliente. Um bom caso de sucesso é usar uma prioridade de produção e pode ser usado o *fibonacci*, por exemplo.

4. Estimar o esforço relativo de cada um dos cartões *Kanban*, para a previsão da produção de cada funcionalidade/história. Novamente, um bom caso de uso é o *fibonacci*.
5. Estabelecer os limites de WIP, de acordo com o Capítulo 4, na seção 1.3, para cada coluna, inclusive as de pronto.
6. Registrar data completa (hora, dia, mês e ano) de entrada e de saída de cada item, por coluna, para cada dia de trabalho num projeto que use *Scrum*.
7. Calcular os tempos de cada ciclo (Lead Time), que foi visto no Capítulo 4 na seção 1.2.
8. Movimentar os cartões com as histórias ao longo das colunas, de acordo com o fluxo real da tarefa ou atividade.

9. Calcular e acompanhar as métricas básicas como tempos de ciclo, throughput e lead time ou tempo de processamento, de acordo com o Capítulo 4, na seção 1.2.
10. Identificar as formas de reduzir o tempo de processamento, dentro do *Daily Scrum*. Para cada atividade, são observadas pela equipe as métricas calculadas, os gargalos e problemas encontrados. A partir do que foi verificado, identificar as causas de problemas, incluindo gargalos, por meio de técnicas bem conhecidas: A3 *Report*, Diagrama de *Ishikawa*, Análise da Causa Raiz, Cinco Porquês (Idealmente com a participação de toda a equipe)
11. Ainda no *Daily Scrum*, resolver os gargalos para dada atividade, com estes passos:
    a. Definir claramente cada problema.
    b. Encontrar o maior número possível de causas geradoras do problema. Aqui pode usar *Brainstorming*.
    c. Construir o diagrama seguindo o caminho:
        i. O problema fica à direita;
        ii. Definir as categorias de causas mais apropriadas.
        iii. Aplicar o que foi encontrado no *Brainstorming*.
        iv. Para cada causa relacionar os Cinco Porquês, ligando as respostas à causa principal.
    d. Analisar o diagrama construído.
12. Aplicar as soluções para os problemas encontrados e dar seguimento ao desenvolvimento das atividades no projeto.
13. Aplicar o *Kaizen*, a Melhoria Contínua, no dia seguinte, voltando ao passo 1.

Esta medida de incorporação do *Kanban* ao Scrum visa aproveitar a questão do aperfeiçoamento do trabalho ao projeto de forma mais detalhada, visando

projetos críticos, pois a margem de erro desses projetos é muito pequena e exige as entregas de alto valor/qualidade.

1.3 Considerações finais

Este capítulo apresentou os estudos de campo de dois profissionais da área de Engenharia de *Software*: um CEO de uma *Startup* e uma aluna desenvolvedora de. Na sequência foi apresentada uma sugestão de melhoria, que foi a aplicação do *Kanban* num projeto que utiliza o *Scrum*, com um passo a passo de como fazer.

Capítulo 5
Conclusão e Trabalhos Futuros

Esta monografia versou sobre a gestão ágil de projetos de *software* que visa tratar o problema da qualidade do produto e dos requisitos de *software*. Para tanto, foi necessário mostrar um pouco as metodologias tradicionais de desenvolvimento de *software*.

Iniciamos a discussão sobre gestão ágil de projetos de *software* apresentando o conceito de agilidade, incluindo o manifesto ágil e os valores deste manifesto. Logo na sequência, foi discutido o gerenciamento ágil de projetos de software, que visa desenvolver rapidamente um produto de software com entregas incrementais e que possui o foco em práticas de gestão, segundo o Manifesto Ágil.

Apresentamos em detalhes três metodologias para a gestão ágil de projetos: *Scrum*, *Lean* e *Kanban*. Foi apresentado ainda um estudo de campo do uso do *Scrum* em duas realidades diferentes: Numa *Startup* e num projeto de uma disciplina. Finalmente foi mostrado possíveis melhorias do uso do *Scrum* em um projeto. Essa incorporação e melhoria do *Scrum* foram dadas com medidas simples, adotando práticas do *Kanban*.

Pretendo como trabalhos futuros de uma pós-graduação ou até mesmo de um mestrado, a partir deste trabalho apresentado, pesquisar e chegar a um modelo genérico de desenvolvimento e de projeto ágil de *software* e que sirva de base para utilizar o modelo ágil mais adequado para a construção do *software* (Scrum, *Kanban* ou *Lean*).

Bibliografia

[1] AQUINO, R. **Utilização do Kaizen em um Ambiente de TI.** Disponível em: <http://www.leanti.com.br/artigos/7/utilizacao-do-kaizen-em-um-ambiente-lean-ti.aspx> Acesso em outubro de 2015.

[2] BECK, K. 2001. AGILE ALLIANCE. **Manifesto for Agile Software Development**. Disponível em <http://www.agilemanifesto.org/>. Acesso em 29 agosto de 2015.

[3] CHIN, G. **Agile Project Management – How to Succeed in the Face of Changing Project Requirements.** Nova Iorque, Amacom, 2004. 229p.

[4] CRESCÊNCIO, S. **A Pirâmide Lean.** Engenharia de Software Magazine 38. Disponível em: <http://www.devmedia.com.br/a-piramide-lean-artigo-revista-engenharia-de-software-magazine-38/21660>

[5] COSTA, R. S e JARDIM, E. G. M. – **Os Cinco Passos do Pensamento Enxuto** Net, Rio de Janeiro, 2010. Disponível em: <http://www.trilhaprojetos.com.br> Acesso em novembro de 2015.

[6] GROUP, S. **Report Chaos.** Disponível em <http://www.infoq.com/articles/standish-chaos-2015> Acesso em novembro de 2015.

[7] HIGHSMITH, J. **Agile Project Management – Creating Innovative Products.** Cockburn, Highsmith. 277p.

[8] IMAI M. **Baixando os Custos e Melhorando a Qualidade.** Disponível em: <http://br.kaizen.com/artigos-e-livros/artigos/kaizen-baixando-os-custos-e-melhorando-a-qualidade.html> Acesso em outubro de 2015.

[9] LEAN WAY. **Lean Manufacturing.** Disponível em: <http://www.leanway.com.br/lean%20manufacturing#content-header> Acesso em outubro de 2015.

[10] KNIBERG, H e SKARIN, M. **Kanban e Scrum – Obtendo o melhor de ambos.** InfoQ. Disponível em: <http:www.infoq.com/BR/minibooks/kanban-scrum-minibook>

[11] MONTEIRO, E. M. e ZAMPAR, F. **Kaizen – Um aliado na Melhoria Contínua.** Disponível em: <http://www.techoje.com.br/site/techoje/categoria/detalhe_artigo/666> Acesso em outubro de 2015.

[12] OHNO, T. **O Sistema Toyota de Produção: Além da Produção de Larga Escala.** Porto Alegre. Bookman, 1997.

[13] PALLADINO, M. **Incluindo Qualidade no Processo de Desenvolvimento de Software.** Disponível em: <http://www.leanti.com.br/artigos/21/incluindo-qualidade-no-processo-de-desenvolvimento-de-software.aspx> Acesso em outubro de 2015.

[14] PICCHI, F. A. **Como Agregar Valor e Evitar Desperdícios na Cadeia de Produção.** Publicado em Julho de 2013. Disponível em: <http://www.lean.org.br/leanmail/177/-lean-it-como-agregar-valor-e-evitar-desperdicios-na-cadeia-de-informacao.aspx> Acesso em outubro de 2015.

[15] PINTO, R. S. **O que é Lean?** Disponível em: <http://www.tiespecialistas.com.be/2014/06/o-que-e-lean/> Acesso em outubro de 2015.

[16] POPPENDICK< M e POPPENDICK, T. **Entrevista ao InfoQ.** Disponível em: <http://infoq/br/articles/entrevista-Mary-Tom> Acesso em dezembro de 2015.

[17] PROJECT MANAGEMENT INSTITUTE - PMI. **Project Management Book of Knowledgement.** Newtown Square, Pennsylvania, Estados Unidos. Project Management Institute, 5º ed. 2012

[18] PROJECT MANAGEMENT INSTITUTE - PMI. **Software Extension to the PMBoK Guide Fifth Edition.** Newtown Square, Pennsylvania, Estados Unidos. Project Management Institute, 5º ed. 2012

[19] RUP - RATIONAL UNIFIED PROCESS. **Rational Unified Process,** 2002. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>

[20] SANTOS, A. G. **Scrum No Burndown**. Disponível em: <https://andreykurka.wordpress.com/2013/10/24/scrum-no-burndown/> Acesso em outubro de 2015.

[21] SCRUM GUIDE. **Scrum Guide** 2013. Scrum Alliance. 19 p.

[22] SOFTWARE, M. G. **Scrum Task Board.** Disponível em: <https://www.mountaingoatsoftware.com/agile/scrum/task-boards> Acesso em outubro de 2015.

[23] SOMMERVILLE, I. **Software Engineering.** 6º ed. 2004.

[24] WOMACK, J e JONES, D. **A Mentalidade Enxuta nas Empresas (Lean Thinking),** Editora Campus, 1º Edição, 2004.