Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção
Curso de Engenharia de Produção

Aplicabilidade de conceitos do Sistema Toyota de Produção junto à metodologia Scrum na fábrica de *software*.

Casimiro Beleze

TCC-EP-14-2009

Maringá - Paraná
Brasil

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção
Curso de Engenharia de Produção

Aplicabilidade de conceitos do Sistema Toyota de Produção junto à metodologia Scrum na fábrica de *software*.

Casimiro Beleze

TCC-EP-14-2009

Trabalho de conclusão de curso apresentado como requisito de avaliação no curso de graduação em Engenharia de Produção na Universidade Estadual de Maringá – UEM.
Orientador: Prof.(º): M. Sc. Lafaiete H. R. Leme

Maringá - Paraná
2009

DEDICATÓRIA

Dedico este trabalho aos meus pais Reginaldo e Adriana, meu irmão Pedro e minha noiva Fabiane pelo apoio, compreensão e dedicação durante esta etapa da minha vida que julgo ser o término do começo.

AGRADECIMENTOS

À Deus, por tudo aquilo que eu fui e tive, tudo aquilo que sou e tenho e por tudo aquilo que ainda serei e terei.

Aos meus grandes exemplos de vida que são meu pai, minha mãe, meu irmão e minha noiva e a todos os meus familiares que sempre me apoiaram. Especialmente ao meu pai, pois sem ele ter acreditado em mim, nada seria possível e também a minha noiva por ter agüentado nos dias de mau humor e as noites em claro.

Ao meu orientador Lafaiete, que fez mais do que orientar, me ajudou a realizar um sonho que era publicar um artigo no Simpósio Nacional de Engenharia de Produção – SIMPEP.

Ao professor Gonçalo que aceitou fazer parte da banca e também pelo ótimo professor que é devido à grande experiência de vida.

Aos meus grandes amigos que fiz na empresa que trabalhei. Sem a oportunidade de aprendizado, tanto profissional como pessoal, que eles me deram, esse trabalho teria sido muito mais penoso. Foi lá que aprendi o verdadeiro espírito de equipe.

À Marcos Pereira, consultor da SWQuality e acima de tudo um amigo, pelos excelentes treinamentos sobre Scrum e pela ajuda neste trabalho.

Aos meus amigos e colegas da Engenharia de Produção e da UEM. Em especial ao William Fabris, vulgo Madruga, e ao Paulo Henrique Soares, pela ajuda, mais que além da conta.

À equipe Kanbanlizando pelo 36º lugar no Desafio Sebrae no estado do Paraná em 2009.

Valeu a pena!!!!!!

“O homem nunca sabe do que é capaz,
até que seja obrigado a tentar”.
Charles Dickens

RESUMO

Este trabalho apresenta um enfoque diferenciado para o gerenciamento da fábrica de software, baseado na produção em larga escala, na automação dos processos de desenvolvimento e gerenciamento, e na utilização de conceitos de Engenharia da Produção, representados pelo Sistema Toyota de Produção junto à metodologia Scrum. Para isto, foi feito uma análise abrangente destas duas filosofias de trabalho, em seguida elencando os pontos de união entre elas de acordo com um estudo de caso feito em uma fábrica de software de Maringá, Paraná.Os resultados apresentados foram muito satisfatórios, pois houve uma redução significativa no tempo de desenvolvimento dos projetos, como pode-se constatar no final deste trabalho.

Palavras-chave: Scrum, *Kanban*, Sistema Toyota de Produção, Desenvolvimento Ágil, Gestão de Projeto de *Software.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

STP	Sistema Toyota de Produção
PO	*Product owner* ou dono do produto
SM	Scrum *master* ou mestre Scrum
PB	*Product backlog*
SB	*Sprint backlog*
SP1	*Sprint planning* I
SP2	*Sprint planning* II
PDCA	Planejar, fazer, checar e agir
3W	*Who, What, Why*
5W2H	*What, Who, When, Where, Why, How e How much*
WIP	*Work in process*
PMI	*Project Management Institute*
PMBOK	*Project Management Body of Knowledge*
UML	Unified Modeling Language
SIG	Grupo de interesses específicos

1 INTRODUÇÃO

Segundo Taiichi Ohno (1997), no sistema automatizado, o controle visual, ou a “gestão pela visão”, pode ajudar a trazer fraquezas da produção (ou seja, em cada jogador) à superfície. Isto nos permitirá então adotar medidas que fortaleçam os jogadores envolvidos. Um time de campeonato combina bom trabalho de equipe com habilidade individual. Da mesma forma, uma linha de produção onde o *just-in-time* e a automação com um toque humano funcionam juntos será mais forte do que outras linhas. Sua força está na sinergia destes dois fatores.

Segundo Ken Schwaber e Beedle (Schwaber; Beedle, 2002), Scrum simplesmente oferece um *framework* e um conjunto de práticas que mantém tudo visível. Isto permite aos praticantes do Scrum saber exatamente o que está acontecendo e fazer no local os ajustes para manter o projeto na direção dos objetivos desejados.

Esses dois pontos sem dúvida definem um ponto de intersecção entre o Sistema Toyota de Produção e o Scrum. Ambos pregam o controle visual da gestão das suas operações, mas por trás disso tudo há uma essência maior: o trabalho em equipe. Todos em busca de um mesmo objetivo, como um verdadeiro time. Seja na produção de uma peça manufaturada ou de uma nova funcionalidade de um *software*.

1.1 Justificativa

Estes dois temas foram trazidos a tona pelo autor devido à solidez de cada um e principalmente pelos resultados positivos que ambos desempenham individualmente nas empresas que os aplicam, além de terem um enfoque atual e serem quebradores de paradigmas.

São tratados neste trabalho de forma associada devido à conjuntura existente entre as duas metodologias, fato este lançado baseado na experiência e vivência do autor com o método Scrum na fábrica de *software* e seus estudos sobre o Sistema Toyota de Produção na Engenharia de Produção.

1.2 Definição e delimitação do problema

Este estudo tece uma linha tênue entre alguns conceitos como *sprint* e ciclo PDCA, *story board* e *Kanban*, 3W e 5W2H, entre outros aspectos, respectivamente do Scrum e STP. Salienta-se como o uso em conjunto desses conceitos pode afetar positivamente o desempenho dos membros da equipe envolvida e o andamento das atividades relacionadas ao bom desenvolvimento do produto almejado.

Sabe-se que tanto a adoção do Scrum quanto do STP trazem vantagens as organizações que implementaram alguma das duas filosofias de forma independente. Contudo o uso em conjunto de Scrum e STP para gestão da produção de uma fábrica de *software*, como por exemplo, na empresa que serviu de estudo de caso a este trabalho, também trouxe resultado favorável.

1.3 Objetivos

O objetivo deste trabalho não foi inventar uma nova metodologia, e sim somar duas formas de trabalho para se obter um melhor aproveitamento e entendimento daquilo que está proposto a ser feito.

1.3.1 Objetivo geral

Scrum não é um método fim de gestão, mas sim um meio, quase uma filosofia, que incentiva a implementação de novas ferramentas de trabalho, aperfeiçoando e maximizando os resultados positivos e minimizando os erros e retrabalhos.

Portanto o objetivo geral foi levantar pontos em comum entre os conceitos do Sistema Toyota de Produção e o Scrum, analisando a viabilidade e implementando ferramentas deste sistema japonês neste método ágil de desenvolvimento de *software*.

1.3.2 Objetivos específicos

Os objetivos específicos deste trabalho foram:

- Verificar os pontos comuns entre a metodologia Scrum e o Sistema Toyota de Produção, estudando como eles podem coexistir, certificando-se que não são mutuamente exclusivos;
- Elaborar uma proposta para unir as duas metodologias;
- Implantar esta proposta de união de forma cooperativa na Organização;
- Analisar e apresentar os resultados positivos desta implementação.

REVISÃO DA LITERATURA

2.1 Desenvolvimento de Software

2.1.1 UML

A *Unified Modeling Language* (UML) é uma linguagem visual para modelar sistemas. Isso quer dizer que ela é constituída de elementos gráficos que permitem representar os conceitos do software a ser desenvolvido. Através dos elementos gráficos definidos nessa linguagem, podem-se construir diagramas que representam diversas perspectivas de um sistema (Bezerra, 2002).

Segundo Bezerra (2002), cada elemento gráfico possui uma *sintaxe* (isto é, uma forma predeterminada de se desenhar um elemento) e uma *semântica* que definem o que significa o elemento e para que ele deva ser utilizado.

Um processo de desenvolvimento que utilize a UML como linguagem de suporte a modelagem envolve a criação de diversos documentos. Na terminologia da UML, esses documentos são denominados artefatos de *software*. São os artefatos que compõe as visões do sistema.

Esses artefatos gráficos produzidos durante o desenvolvimento de um sistema de *software* são definidos através da utilização dos diagramas da UML. Os diagramas da UML são listados na Figura 1.

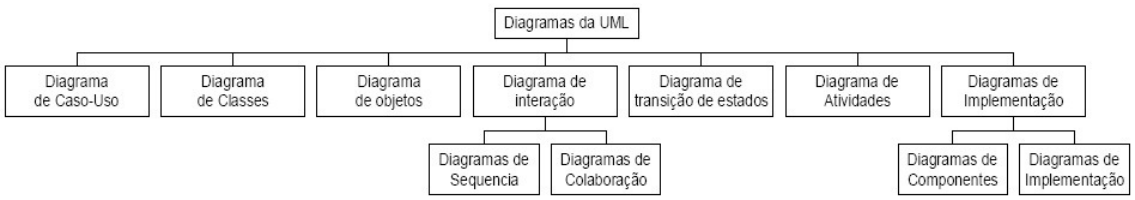


Figura 1 – Os diagramas UML.

2.1.2 PMI e PMBOK

O PMI (*Project Management Institute*) é uma entidade sem fins lucrativos que congrega profissionais que atuam no campo relacionado à Gerência de Projetos. Foi fundado em 1969 nos Estados Unidos e está presente hoje em todos os continentes através de Capítulos, Seções Regionais e de Grupos de Interesse Específicos (SIGs). É um fórum de excelência na área de gerência de projetos promovendo seu crescimento, divulgação, educação e valor nas organizações e praticantes, além de manter um programa de Certificação profissional – Project Management Professional – PMP. Com o lançamento do PMBOK (*Project Management Body of Knowledge*), o PMI dá continuidade ao seu esforço de desenvolvimento da profissão de Gerenciamento de Projetos.

Em 1987 o PMI produziu a primeira versão do PMBOK, o qual fornece uma referência básica em nível de conhecimentos e práticas do gerenciamento de projetos, constituindo-se um padrão mundial, aceito inclusive pela ANSI (*America National Standart Institute*). Com o objetivo de organizar o conhecimento sobre gerenciamento de projetos o PMI definiu nove áreas de conhecimento em gerência de projetos como mostra a Tabela 1 na próxima página.

O objetivo principal do guia PMBOK é identificar o subconjunto do conjunto de conhecimentos em gerenciamento de projetos que é amplamente reconhecido como boa prática. Uma boa prática não significa que o conhecimento descrito deverá ser sempre aplicado uniformemente em todos os projetos. A equipe de gerenciamento de projetos é responsável por determinar o que é adequado para um projeto específico.

O guia PMBOK está organizado em três seções:

- Seção I – A estrutura do Gerenciamento de Projetos – que fornece uma estrutura básica para o entendimento do gerenciamento de projetos. Capítulo 1 - Introdução - define termos-chave e fornece uma visão geral do restante do Guia PMBOK, Capítulo 2 - Ciclo de vida e organização do projeto - Descreve o ambiente no qual os projetos operam. A equipe de gerenciamento de projetos deve entender este conceito mais amplo. Gerenciar as atividades do dia-a-dia do projeto é necessário, mas não suficiente para garantir o sucesso.

Tabela 1 – Áreas de conhecimento segundo o PMI.

Área de Conhecimento	Processo
Gerência de Integração	Desenvolvimento do Plano de Projeto
	Execução do Plano de Projeto
	Controle de Alterações Integrado
Gerência de Escopo	Inicialização
	Planejamento de Escopo
	Definição de Escopo
	Verificação de Escopo
	Controle de Alteração de Escopo
Gerência de Tempo	Definição de Atividades
	Sequenciamento de Atividades
	Estimativa de Duração de Atividades
	Desenvolvimento do Cronograma
	Controle do Cronograma
Gerência de Custo	Planejamento de Recursos
	Estimativa de Custo
	Provisionamento dos Custos
	Controle de Custos
Gerência de Qualidade	Planejamento da Qualidade
	Garantia da Qualidade
	Controle da qualidade
Gerência de Recursos Humanos	Planejamento Organizacional
	Designação de Pessoal
	Desenvolvimento do Time
Gerência de Comunicação	Planejamento de Comunicação
	Distribuição da Informação
	Informações de Desempenho
	Fechamento Administrativo
Gerência de Risco	Plano de Gerência de Risco
	Identificação de Riscos
	Análise Qualitativa de Riscos
	Análise Quantitativa de Riscos
	Planejamento de Resposta de Risco
	Monitoramento e Controle de Risco
Gerência de Aquisição	Plano de Aquisição
	Plano de Solicitação
	Solicitação
	Seleção da Procedência
	Manutenção do Contrato
	Fechamento do Contrato

- Seção II – A norma de Gerenciamento de Projetos de um projeto – especifica todos os processos de gerenciamento de projetos usados pela equipe para gerenciar um projeto. Capítulo 3 – Processos de Gerenciamento de Projetos de um projeto - Descreve os cinco grupos de processos de gerenciamento de projetos necessários para qualquer projeto e os processos de gerenciamento de projetos que os compõem. Este capítulo descreve a natureza multidimensional do gerenciamento de projetos.
- Seção III – As áreas de Conhecimento em Gerenciamento de Projetos – Organiza os 44 processos de gerenciamento de projetos dos grupos de processos do capítulo 3 em

nove áreas de conhecimento, conforme descrito a seguir. Uma introdução à Seção III descreve as legendas dos fluxogramas de processo usados em cada capítulo de área de conhecimento e um texto introdutório aplicável a todas as áreas de conhecimento.

O Capítulo 4, **Gerenciamento de integração do projeto**, descreve os processos e as atividades que integram os diversos elementos do gerenciamento de projetos, que são identificados, definidos, combinados, unificados e coordenados dentro dos grupos de processos de gerenciamento de projetos. Ele consiste nos processos de gerenciamento de projetos: Desenvolver o termo de abertura do projeto, desenvolver a declaração do escopo preliminar do projeto, desenvolver o plano de gerenciamento do projeto, orientar e gerenciar a execução do projeto, monitorar e controlar o trabalho do projeto, controle integrado de mudanças e encerrar o projeto.

O Capítulo 5, **Gerenciamento do escopo do projeto**, descreve os processos envolvidos na verificação de que o projeto inclui todo o trabalho necessário, e apenas o trabalho necessário, para que seja concluído com sucesso. Ele consiste nos processos de gerenciamento de projetos: Planejamento do escopo, definição do escopo, criação de EAP (Estrutura Analítica do Projeto), verificação do escopo e controle do escopo.

O Capítulo 6, **Gerenciamento de tempo do projeto**, descreve os processos relativos ao término do projeto no prazo correto. Ele consiste nos processos de gerenciamento de projetos: Definição da atividade, seqüenciamento de atividades, estimativa de recursos da atividade, estimativa de duração da atividade, desenvolvimento do cronograma e controle do cronograma.

O Capítulo 7, **Gerenciamento de custos do projeto**, descreve os processos envolvidos em planejamento, estimativa, orçamentação e controle de custos, de modo que o projeto termine dentro do orçamento aprovado. Ele consiste nos processos de gerenciamento de projetos: Estimativa de custos, orçamentação e controle de custos.

O Capítulo 8, **Gerenciamento da qualidade do projeto**, descreve os processos desenvolvidos na garantia de que o projeto irá satisfazer os objetivos para os quais foi

realizado. Ele consiste nos processos de gerenciamento de projetos: Planejamento da qualidade, realizar a garantia da qualidade e realizar o controle da qualidade.

O Capítulo 9, **Gerenciamento de recursos humanos do projeto**, descreve os processos que organizam e gerenciam a equipe do projeto. Ele consiste nos processos de gerenciamento de projetos: Planejamento de recursos humanos, contratar ou mobilizar a equipe do projeto, desenvolver a equipe do projeto e gerenciar a equipe do projeto.

O Capítulo 10, **Gerenciamento das comunicações do projeto**, descreve os processos relativos à geração, coleta, disseminação, armazenamento e destinação final das informações do projeto de forma oportuna e adequada. Ele consiste nos processos de gerenciamento de projetos: Planejamento das comunicações, distribuição das informações, relatório de desempenho e gerenciar as partes interessadas.

O Capítulo 11, **Gerenciamento de riscos do projeto**, descreve os processos relativos à realização do gerenciamento de riscos em um projeto. Ele consiste nos processos de gerenciamento de projetos: Planejamento do gerenciamento de riscos, identificação de riscos, análise qualitativa de riscos, análise quantitativa de riscos, planejamento de respostas a riscos e monitoramento e controle de riscos.

O Capítulo 12, **Gerenciamento de aquisições do projeto**, descreve os processos que compram ou adquirem produtos, serviços ou resultados, além dos processos de gerenciamento de contratos. Ele consiste nos processos de gerenciamento de projetos: Planejar compras e aquisições, planejar contratações, solicitar respostas de fornecedores, selecionar fornecedores, administração de contrato e encerramento do contrato.

2.1.3 Desenvolvimento Ágil de Software

De acordo com este autor o desenvolvimento ágil de *software* procura trabalhar de forma não burocrática, eliminando gargalos gerados por muitas discussões e indecisões durante o desenvolvimento de um *software*.

Kent Beck (Beck, 2001) e seus colaboradores ao criar o manifesto ágil, passaram a valorizar:

- *Indivíduos e interações* em vez de processos e ferramentas;
- *Softwares funcionando* em vez de documentação abrangente;
- *Colaboração do cliente* em vez de negociação de contratos;
- *Resposta a modificações* em vez de seguir um plano.

Embora as idéias subjacentes que guiam o desenvolvimento ágil tenham estado conosco há muitos anos, somente durante a década de 1990 é que foram cristalizados em um "movimento". Em essência os métodos ágeis foram desenvolvidos em um esforço para vencer as fraquezas percebidas e reais da engenharia de *software* convencional. O desenvolvimento ágil pode fornecer importantes benefícios, mas ele não é aplicável a todos os projetos, produtos, pessoas e situações. Ele também não é o contrário à sólida prática de engenharia de *software* e pode ser aplicado como uma filosofia prevalecente.

Na economia moderna é frequentemente difícil ou impossível prever como um sistema baseado em computador evoluirá com o passar do tempo. Condições de mercado mudam rapidamente, necessidades dos usuários finais evoluem e novas ameaças de competição emergem sem alerta. Em muitas situações, não podemos mais definir completamente os requisitos antes do início do projeto. Os engenheiros de *software* devem ser ágeis o suficiente para responder a um ambiente de negócio mutante.

Isso significa que o reconhecimento dessas causas realísticas modernas nos obriga a descartar princípios, conceitos, métodos e ferramentas, todos importantes na engenharia de *software*? Certamente que não, como todas as disciplinas da engenharia, a engenharia de *software* continua a evoluir. Ela pode ser adequada facilmente para encarar os desafios colocados pela demanda por agilidade.

2.1.3.1 O que é Agilidade?

O que é agilidade no contexto do trabalho de engenharia de *software*? Ivar Jacobson (Jacobson, 2002) fornece uma discussão útil:

"Agilidade tornou-se atualmente uma palavra mágica quando se descreve um processo moderno de *software*. Tudo é ágil. Uma equipe ágil é uma equipe esperta, capaz de responder adequadamente a modificações. Modificações (*software*, membros da equipe, tecnologias, etc.) são o foco principal do projeto e tem impacto no produto final. O apoio para modificações deveria ser incorporado em tudo que fazemos em *software*, algo que se adota porque está no coração e na alma do *software*. Uma equipe ágil reconhece que o *software* é desenvolvido por indivíduos trabalhando em equipes e que as especialidades dessas pessoas e sua capacidade de colaborar estão no âmago do sucesso do projeto."

Na visão de Jacobson, o acolhimento de modificações é o principal guia para a agilidade. Os engenheiros de *software* devem reagir rapidamente se tiverem de acomodar as rápidas modificações que Jacobson descreve. Mas a agilidade é mais do que uma resposta efetiva à modificação. Ela encoraja estruturas e atitudes de equipe que tornam a comunicação mais fácil, entre todos os envolvidos no processo. Enfatiza a rápida entrega do *software* e dá menos importância para produtos intermediários (nem sempre isso é bom). Adota os clientes como parte da equipe de desenvolvimento. Admite que o planejamento em um mundo incerto tem seus limites e que um plano de projeto deve ser flexível.

A Aliança Ágil (Beck, 2001) define 12 princípios para aqueles que querem alcançar agilidade:

1. Nossa maior prioridade é satisfazer o cliente desde o início por meio de entrega contínua de *software* valioso.
2. Modificações de requisitos são bem vindas, mesmo que tardias no desenvolvimento. Os processos ágeis aproveitam as modificações como vantagens para a competitividade do cliente.
3. Entrega de *softwares* funcionando frequentemente, a cada duas semanas até dois meses, de preferência no menor espaço de tempo.

4. O pessoal do negócio e os desenvolvedores devem trabalhar juntos diariamente durante todo o projeto.
5. Construção de projetos em torno de indivíduos motivados. Forneça-lhes o ambiente e apoio que precisam e confie que eles farão o trabalho.
6. O método mais eficiente e efetivo de levar informações para dentro da equipe de desenvolvimento é a conversa face a face.
7. *Software* funcionando é a principal medida de progresso.
8. Processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante, indefinidamente.
9. Atenção contínua a excelência técnica e ao bom projeto facilitam a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho que não precisa ser realizado – é essencial.
11. As melhores arquiteturas, requisitos e projetos surgem de equipes auto-organizadas.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva, então sintoniza e ajusta adequadamente seu comportamento.

A agilidade pode ser aplicada a qualquer processo de *software*. Entretanto para conseguir isso é essencial que o processo seja de modo que permita à equipe de projeto adaptar tarefas e aperfeiçoá-las. Conduzir o planejamento para que se entenda a fluidez de uma abordagem de desenvolvimento ágil, eliminar tudo, menos os produtos de trabalho mais essenciais e mantê-los simples. Enfatizar uma estratégia de entrega incremental que forneça o *software* funcionando ao cliente o mais rápido possível (Pressman, 2006).

2.1.3.2 O que é um Processo Ágil?

Qualquer processo ágil de *software* é caracterizado de modo que atenda a três suposições-chave sobre a maioria dos projetos de *software* (Fowler, 2002):

1. É difícil prever antecipadamente quais requisitos de *software* vão persistir e quais serão modificados. É igualmente difícil prever como as prioridades do cliente serão modificadas à medida que o projeto prossegue.

2. Para muitos tipos de *software*, o projeto e a construção são intercalados, isto é, as duas atividades devem ser realizadas juntas de modo que os modelos de projeto sejam comprovados à medida que são criados. É difícil prever o quanto de projeto é necessário antes que a construção seja usada para comprovar o projeto.
3. Análise, projeto, construção e testes não são tão previsíveis (do ponto de vista do planejamento) como gostaríamos.

Dadas essas três suposições surge uma questão importante, como criar um processo que possa gerenciar a imprevisibilidade? A resposta está na adaptabilidade do processo (as modificações rápidas do projeto e das condições técnicas). Um processo ágil, portanto, deve ser adaptável.

Mas a adaptação contínua sem progresso realiza pouco. Assim, um processo ágil de *software* deve ser adaptado incrementalmente. Para realizar a adaptação incremental, uma equipe ágil requer o *feedback* do cliente (de modo que adaptações apropriadas possam ser feitas), isso pode ser feito através de protótipos operacionais. Assim, uma estratégia de desenvolvimento incremental deve ser instituída. Incrementos de *software* devem ser entregues em curtos períodos de tempo de modo que a adaptação acerte o passo com as modificações (imprevisibilidade). Essa abordagem iterativa habilita o cliente a avaliar o incremento de *software* regularmente, fornece o *feedback* necessário à equipe de *software* e influencia as adaptações do processo feitas para acomodar o *feedback*.

2.1.3.2.1 A Política de Desenvolvimento Ágil

Há um considerável debate sobre os benefícios e a aplicabilidade do desenvolvimento ágil de *software* em contraposição aos processos mais convencionais de engenharia de *software*. Jim Highsmith (Highsmith, 2002) descreve, de forma jocosa, os extremos ao caracterizar o sentimento do campo pró-agilidade (“agilistas”): “Os metodologistas tradicionais são um punhado de bitolados que preferem produzir documentação perfeita a um sistema funcionando que satisfaça às necessidades do negócio”. Em contrapartida, ele descreve, também de forma jocosa, a posição do campo de engenharia de *software* tradicional: “Os metodologistas levianos, quer dizer, “ágeis” são um punhado de gloriosos hackers que terão uma grande surpresa quando tiverem de ampliar seus brinquedos para chegar a um *software* que abranja toda a empresa”.

Como toda argumentação sobre tecnologia de *software*, esse debate metodológico arrisca-se a degenerar em uma “guerra religiosa”. Se a guerra tiver início, o pensamento racional desaparecerá e crenças, em vez de fatos, orientarão a tomada de decisão. Ninguém é contra a agilidade. A verdadeira questão é: qual é o melhor modo de alcançá-la? Outra questão tão importante como essa é: como construir *softwares* que satisfaçam às necessidades do cliente atual e exiba característica de qualidade que lhe permitam ser estendido e ampliado para satisfazer às necessidades do cliente no longo prazo? (Pressman, 2006).

Indubitavelmente pode-se dizer que não existem respostas absolutas para essas questões. Mesmo dentro da escola ágil, há muitos modelos de processo propostos, cada qual com uma abordagem sutilmente diferente para o problema da agilidade. Em cada modelo há um conjunto de “idéias” que representam um afastamento significativo da engenharia de *software* convencional. No entanto, muitos conceitos ágeis são simples adaptações, de bons conceitos da engenharia de *software*. Concluí-se que há muito a ser ganho considerando o melhor de ambas as escolas, e quase nada a ser ganho denegrindo qualquer uma dessas abordagens.

2.1.3.2.2 Fatores Humanos

Aqueles que propõem o desenvolvimento ágil de *software* sofrem muito para enfatizar a importância dos “fatores pessoais” no desenvolvimento ágil bem sucedido. Como Cockburn e Highsmith (2001) declaram, “O desenvolvimento ágil enfoca os talentos e habilidades dos indivíduos moldando o processo a pessoas e equipes específicas”. O ponto-chave dessa declaração é que o processo se molda às necessidades das pessoas e da equipe, e não o contrário.

“O que é considerado meramente suficiente por uma equipe é ou suficiente demais ou insuficiente para outra”. (Alistair Cockburn)

Se os membros de uma equipe de *software* tiverem de estabelecer as características do processo que é aplicado para construir *software*, uma certa quantidade de características chave deve existir entre as pessoas de uma equipe ágil e na equipe em si. Segundo Pressman (Pressman, 2006) essas características são as seguintes:

- **Competência**: Em um contexto de desenvolvimento ágil, “competência” inclui talento inato, habilidades específicas relacionadas a *software* e conhecimento global do processo que a equipe decidiu aplicar. Habilidade e conhecimento do processo podem e devem ser ensinados a todas as pessoas que servem como membros de equipes ágeis.

- **Foco comum**: Embora os membros da equipe ágil possam realizar diferentes tarefas e trazer diferentes habilidades ao projeto, todos deveriam estar focados em uma meta – entregar um incremento de *software* em funcionamento ao cliente dentro do prazo prometido. Para atingir essa meta, a equipe também vai concentrar-se em adaptações contínuas (pequenas e grandes) que farão o processo satisfazer às necessidades da equipe.

- **Colaboração**: Engenharia de Software (independentemente do processo) diz respeito a avaliar, analisar e usar as informações que são comunicadas à equipe de *software*. Criar informações que ajudarão o cliente e outros a entender o trabalho da equipe, e construir informações (*software* de computador e banco de dados relevantes) que forneçam valor de negócio para o cliente. Para realizar essas tarefas os membros da equipe precisam colaborar – uns com os outros, com o cliente e com os gerentes de negócios.

- **Capacidade de tomada de decisão**: Qualquer boa equipe de *software* (inclusive equipes ágeis) deve ter a liberdade de controlar o seu próprio destino. Isso implica que seja dada autonomia à equipe – autoridade para tomada de decisões sobre tópicos técnicos e de projetos.

- **Habilidade de resolver problemas vagos (ambigüidades)**: Os gerentes de *software* devem reconhecer que uma equipe ágil terá que lidar continuamente com ambigüidades e será continuamente confrontada por modificações. Em alguns casos a equipe precisa aceitar o fato de que o problema que está sendo resolvido hoje pode não ser o problema que precisará ser resolvido amanhã. No entanto, as lições aprendidas de qualquer atividade de solução de problema (inclusive aquelas que resolvem o problema errado) podem ser benéficas para a equipe mais adiante no projeto.

- **Respeito e confiança mútua**: A equipe ágil deve tornar-se o que DeMarco e Lister (1998) chamam de a equipe “consolidada”. Uma equipe consolidada exibe a confiança e o respeito

necessários para torná-la “tão fortemente aglutinada que o todo é maior que a soma das partes”.

- **Auto-organização**: No contexto do desenvolvimento ágil, a auto-organização implica três coisas, (1) a equipe ágil organiza-se para o trabalho a ser feito, (2) a equipe organiza o processo para melhor acomodar seu ambiente local, (3) a equipe organiza o cronograma de trabalho para conseguir melhor entrega do incremento de *software*. A auto-organização tem um certo número de benefícios técnicos, porém, mais importante que isso, ela serve para aperfeiçoar a colaboração e aumentar a moral da equipe. Em essência, a equipe serve como sua própria gerência. Ken Schwaber (Schwaber, 2002) trata desses tópicos quando descreve: “A equipe seleciona quanto trabalho acredita que pode realizar dentro da iteração e a equipe se compromete com o trabalho. Nada desmotiva tanto uma equipe quanto alguém de fora assumir compromissos por ela. Nada motiva tanto uma equipe quanto a aceitação da responsabilidade de cumprir os compromissos que ela própria estabeleceu”.

2.2 Modelo Ágil de Processo

2.2.1 Scrum

Scrum é muito mais que uma ferramenta e, definitivamente, ele não é apenas uma ferramenta. Scrum é uma mentalidade, no mesmo sentido de que o Sistema Toyota de Produção também foi sempre uma mentalidade, uma filosofia e não apenas uma ferramenta. (Gloger, 2009).

A primeira experiência com o Scrum ocorreu em uma fábrica de automóveis, em que se constatou que a utilização de equipes pequenas e multidisciplinares produzia melhores resultados. Em analogia a essas equipes, associou-se a formação do Scrum a de um jogo de *Rugby*. A partir de 1995, Ken Schwaber formalizou a definição de Scrum e ajudou a implantá-lo em desenvolvimento de *software* em todo o mundo.

Dentre as técnicas de utilização do Scrum, há a entrega de produtos em períodos de tempo pré-estabelecidos, nunca inferiores há uma semana ou superiores há trinta dias. Para estimular o contato entre empresa e cliente, as necessidades do cliente são revistas em períodos regulares de tempo, a essas ações dá-se o nome de *sprint*, que nada mais é do que um

intervalo de tempo em que se roda um ciclo PDCA completo. Ao término de cada *sprint*, o cliente recebe um conjunto de funcionalidades desenvolvidas e prontas para serem utilizadas. A melhor maneira de comprovar se o *software* atende às necessidades é fazer com que o cliente o utilize, apontando as qualidades e o que falta ser aperfeiçoado (Schwaber, 2002).

Para controlar a gestão da produção das novas funcionalidades é utilizado um *story board* em que os membros do time visualizam o andamento do trabalho de todos na equipe. Desta forma é mais fácil notar onde estão os gargalos e as áreas que geram mais retrabalho, não com o intuito de apontar culpados, mas sim dicas de melhorias.

Scrum não é um processo previsível, ele não define o que fazer em toda circunstância. Ele é usado em trabalhos complexos nos quais não é possível prever tudo o que irá ocorrer e oferece um *framework* e um conjunto de práticas que torna tudo visível. Isso permite aos praticantes do Scrum saber exatamente o que está acontecendo ao longo do projeto e fazer os devidos ajustes para manter o projeto se movendo ao longo do tempo visando alcançar os seus objetivos.

- Scrum é um processo ágil para gerenciar e controlar trabalho de desenvolvimento.
- Scrum é uma abordagem baseada em equipe para iterativamente, incrementalmente desenvolver sistemas e produtos, quando requisitos estão mudando rapidamente.
- Scrum é um processo que controla o caos de conflito de interesses e necessidades.
- Scrum é uma forma de melhorar a comunicação e maximizar a cooperação.
- Scrum é uma forma de detectar e causar a eliminação de qualquer coisa que fica no caminho do desenvolvimento e entrega de produtos.
- Scrum é uma forma de maximizar a produtividade.
- Scrum tem controlado e organizado o desenvolvimento e implementação de múltiplos produtos e projetos com mais de mil desenvolvedores e implantadores.
- Scrum é uma forma para que todos se sintam bem sobre o seu trabalho, suas contribuições, e que tenham feito muito melhor que possivelmente poderiam.

Comparando com o modelo tradicional o Scrum possui como grande diferencial: focar em realizar pequenas entregas, ou seja, antes de finalizar toda a análise no modelo tradicional, no

Scrum permite entregar algo funcionando. Na Figura 2, vemos uma ilustração de como é o fluxo de processo da metodologia Scrum.

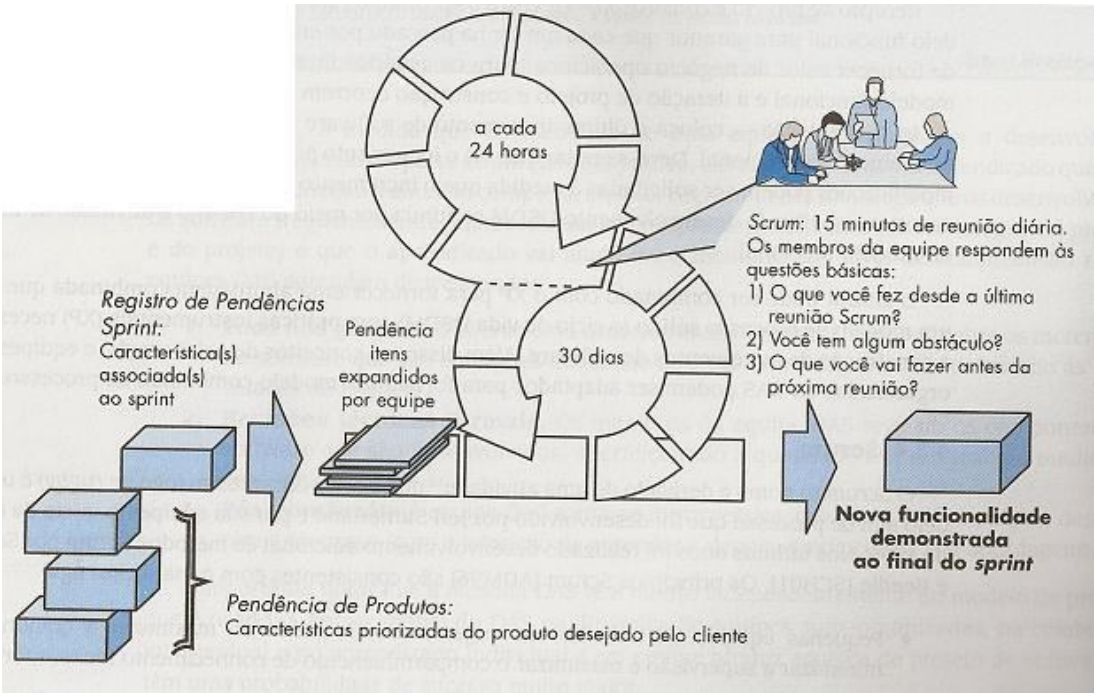


Figura 2 – Fluxo de processo Scrum (Pressman, 2006, pg. 70).

2.2.1.1 Papéis e Responsabilidades de um Time Scrum

2.2.1.1.1 *Product Owner*

O *product owner* (PO) representa os clientes, interessados ou financiadores do produto. O PO é uma parte da equipe de entrega do produto. Pode ser um analista de negócios da empresa ou até mesmo o próprio cliente.

O PO é responsável por:

- Definir a visão do produto.
- Priorizar cada item por valor de negócio.
- Gestão do retorno sobre o investimento (ROI).
- Gerir o *product backlog*.

- Aceitação da nova funcionalidade.
- Planejamento dos *releases* (Liberações de versão)
- Atuar como mediador, quando existirem vários clientes.

2.2.1.1.2 Scrum *Master*

O gestor do produto (*Product manager* – Scrum *master*) tem um papel diferenciado do tradicional comando e métodos de controle. O Scrum *master* trabalha junto e para a equipe.

O Scrum *master* é responsável por:

- Controlar o processo do Scrum, fazer com que ele aconteça e seja executado perfeitamente.
- Remover qualquer *parking lot* (Impedimento) que a equipe encontrar.
- Proteger o time de interferências externas para garantir que a sua produtividade não seja afetada.
- Visualizar estratégias e guiar o time para atingir as metas.
- Criar vitórias de curto prazo.
- Permitir que a equipe auto-organize-se para concluir o trabalho.
- Assegurar que a comunicação tenha caminho aberto e eficiente.
- Garantir e ajudar a equipe a seguir os processos.
- Comandar a *daily* Scrum (Reunião diária).

2.2.1.1.3 Team

Team pode ser traduzido ao pé da letra como time mesmo, ou equipe. O número ideal de pessoas no time é entre 5 e 9 pessoas, porém não é uma regra. Um membro da equipe é um indivíduo que se compromete com a conclusão do trabalho. Seu lema é “nós vamos entregar!”. Os membros da equipe são multifuncionais. Entendem-se equipes multifuncionais como um conjunto de pessoas de diferentes especialidades que atuam e desenvolvem atividades de diferentes naturezas e agem como um time, visando a excelência do processo.

As habilidades e competências individuais são extremamente importantes para a formação dessas equipes. Elas devem ser complementares de forma que possibilitem uma integração real.

A formação de cada profissional que atua em equipes multifuncionais deverá fornecer competências e habilidades que, somadas, possibilitarão a eficiência e a eficácia do trabalho, como por exemplo, um programador pode atuar na análise e concepção do produto e ajudar nos testes caso seja necessário. O cenário ideal aponta para equipes multifuncionais que atuam de forma coordenada e articulada.

Nesta metodologia a equipe não é obrigada a entregar o que o *product owner* propôs que eles fizessem, mas sim o que eles se propuseram a entregar, isso desde que seja na seqüência de prioridades que o PO elencou. Se eles se comprometeram que vão entregar apenas uma estória, terão que entregar esta estória.

Cada membro da equipe é responsável por:

- Entregar o que foi comprometido (focar na meta do *sprint*).
- Qualidade do que for entregue, se empenhar para o trabalho e fazê-lo com a mais alta qualidade, sem *bugs.*
- Estimar as necessidades e garantir que o esforço adicional seja realizado caso necessário.
- Compromisso (Eu quero e vou conseguir).
- Definir os itens do *sprint backlog*.
- Colaborar com os membros da equipe e ajudá-los no que for preciso.
- Participar da *daily* Scrum, todos os dias.
- Participar das reuniões de *sprint planning*
- Participar da *sprint retrospective*.
- Levantar os impedimentos.

2.2.1.1.4 Comparando com o PMBOK

Comparando as responsabilidades do time Scrum com o gerente de projetos no PMBOK, como se vê na Figura 3, o Scrum divide mais as responsabilidades, que eram somente do gerente de projetos, com o time.

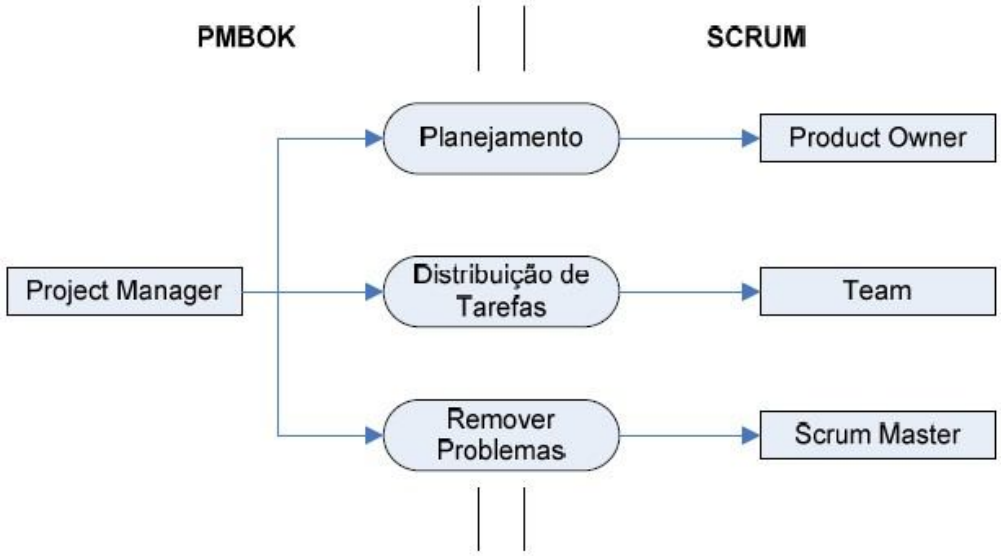


FIGURA 3 – Comparativo entre as responsabilidades no PMBOK e no Scrum.

2.2.1.2 Product Backlog

O *product backlog* é de propriedade e responsabilidade de gerenciamento do PO. É uma dinâmica de priorização para todo o trabalho que precisa ser feito para entregar o produto.

O *product backlog* nada mais é do que uma lista do escopo do projeto. Cada item desta lista é um produto a ser entregue. Estes itens ou produtos alto nível são descrições do trabalho também conhecidos como estórias e são priorizados de acordo com o valor que têm para o negócio. Se uma equipe é responsável por vários projetos, as estórias de todos os projetos disputam a prioridade de serem executadas na mesma lista.
As estórias sempre devem compreender os 3W:

1. *Who*? (Quem?)

2. *What*? (O que?)
3. *Why*? (Por quê?)

Estórias sem atender os 3W, não são estórias. Exemplo, não existe estórias sem personagem. Atender essa regra é fundamental para uma boa descrição da necessidade do cliente e principalmente para compreender o que realmente ele deseja. Caso seja possível quebrar, cada estória pode ser subdivida em tarefas menores que corresponderiam a capítulos, isto é para que um membro da equipe não passe mais de um dia sem finalizar algo, ou seja, escrever um capítulo da estória por dia. Isto favorece a motivação do time, atingindo conquistas diárias.

2.2.1.3 *Sprint*

2.2.1.3.1 O que é um *sprint*?

Este nome, s*print*, é dado ao espaço de tempo em que uma ou mais estórias tem para serem desenvolvidas. O número de estórias por *sprint* irá variar de acordo com a capacidade de produção do time. De acordo com a metodologia Scrum, por convenção, este período não pode exceder quatro semanas. Recomenda-se que um *sprint* seja fixado em quatro semanas (20 dias úteis.) ou de preferência duas semanas (2 *sprints* de 10 dias cada). Na Figura 4, vemos como é feita a iteração que formam vários *sprints* e como os resultados tanto positivos como negativos ajudam a realimentar o *sprint* seguinte, buscando uma melhoria contínua.

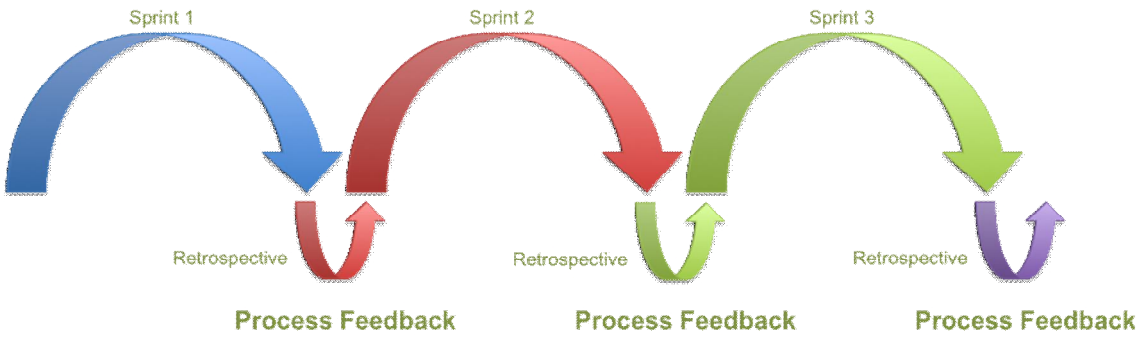


Figura 4 – Iteração formada pelos *sprints*

2.2.1.3.2 *Sprint Planning* I

O planejamento do *sprint* é uma breve reunião de cerca de meio a um dia inteiro de duração. Esta reunião é realizada antes da execução de um *sprint*. O *product owner*, Scrum *master* e a equipe participam desta reunião. O PO deve atender quaisquer perguntas que a equipe tenha e caso não consiga responder alguma na hora, deve procurar dar uma resposta o mais breve possível. Caso a equipe ache que a resposta para a pergunta é muito relevante, o time pode vetar que a estória entre no *sprint* até que a pergunta seja respondida.

Considerações importantes:

- Calcular horários disponíveis para o *sprint* antes da reunião.
- O Scrum *master* e a equipe definem o tempo de disponibilidade dos recursos para o *sprint* antes da realização desta reunião de *sprint planning* I (SP1). Isto é, simplesmente o número de horas de trabalho do pessoal disponível para os dias do *sprint*. Um dia inteiro de produtividade das pessoas deve ser fixado. Ninguém é produtivo para um dia inteiro, devemos levar em consideração cada pessoa, chamadas de telefone, tempo médio de checagem de e-mails, reuniões, parada para o café e outras interrupções.
- O PO deve estar presente na reunião de *sprint* SP1 e na reunião de *sprint review* junto com o time.
- Durante a reunião de SP1 é importante que o PO apresente os requisitos para o time.
- Todos os membros do time devem anotar todas as considerações feitas para cada estória para que todos prestem atenção, pois nesta reunião ainda não se sabe quem irá fazer cada atividade.
- Se os requisitos não estiverem claros para o time, o PO deve esclarecer na hora. Caso não fique claro, a equipe deve rejeitar a estória e o PO deve documentar os requisitos de maneira mais clara.
- Se o PO não estiver presente durante o SP1, correrá o risco de suas estórias não entrarem no próximo Sprint, dependendo da sua prioridade e da sua complexidade.

O time deve avaliar os requisitos seguindo os seguintes critérios:

a) Os requisitos estão claros? A maneira como os requisitos estão escritos está compreensível?
b) A estória possui um tamanho adequado?
c) E possível testar todos os requisitos?
d) Os requisitos necessitam de protótipo? (Criar novas telas ou fazer reuso alterando vários campos em uma tela já existente).
e) Os requisitos impactam em outras partes do produto?

2.2.1.3.3 *Sprint Planning* II e *Sprint Backlog*

Na reunião de *sprint planning* II (SP2) a equipe agora está pronta para criar o *sprint backlog*. A equipe seleciona os itens de alta prioridade do *product backlog* que eles calculam o que podem finalizar no *sprint* e alcançar a meta. Esses itens que serão desenvolvidos no *sprint* corrente formam o *sprint backlog*.

A equipe trabalha através dos itens selecionados, um de cada vez. Para cada item a equipe analisa e determina todas as tarefas que são exigidas para que o item a seja considerado *done*. Esta tarefa é conhecida como expansão.

É importante compreender o que significa *done*. Desenvolvido totalmente e o trabalho testado, isso significa que o *software* pode ser potencialmente liberado. Não é desejável acabar em uma situação em que itens foram feitos parcialmente, isso pode ocorrer se não completar todos os itens do *sprint backlog*. Um item que está 80% completo, não é *done*, ou seja, não é feito. Procurar ao máximo trabalhar com uma única estória por time, fazendo a divisão da estória em atividades durante a execução e não na reunião de SP2. Evitar que haja paralelismo entre as estórias, pois correrá o risco de não ter nada pronto ao final do *sprint*.

Deve-se manter a tarefa estimada dentro de um intervalo de 4 a 8 horas. Se a estimativa começar a ser menor que o limite, é sinal de que a tarefa está muito pequena e pode ser unida a outra. Caso comece a ser maior que o limites então é um sinal de que a tarefa deve ser dividida em pequenas tarefas. Como no exemplo a seguir:

PROCESSO DE CONTRATOS, CONTROLE DE GARANTIA E RELATÓRIO PASSIVO, que era uma grande atividade para desenvolver uma única tela, foi dividida em outras cinco atividades:

- FINANC_01 - PROCESSO DE CONTRATOS
- FINANC_01 - PROCESSO DE CONTRATOS - CADASTRO DE GRUPO DE GARANTIA
- FINANC_01 - PROCESSO DE CONTRATOS - TELA DE CONTRATO
- FINANC_01 - PROCESSO DE CONTRATOS - CONTROLE DE GARANTIAS
- FINANC_01 - PROCESSO DE CONTRATOS – RELATÓRIO PASSIVO

2.2.1.4 Atribuindo Tarefas

Em tradicionais gestões de projetos, como no PMBOK, o gestor atribui as tarefas aos recursos disponíveis. Isto é normalmente feito adiantado. Em um ambiente ágil, a equipe é deixada sozinha para se auto-organizar, isto não implica anarquia contra a gestão, mas simplesmente o fato de que eles são as pessoas que farão o trabalho e são os melhores para decidir sobre a forma de resolver, quando resolver e quem deve resolver, para que eles possam alcançar a sua meta ao final do *sprint.* Claro que eles podem e devem contar com a ajuda da liderança da equipe que é o Scrum *master*, caso ocorram impasses e conflitos durante esta auto-organização.

No início do *sprint* a equipe se reúne, entra em um consenso de acordo com as habilidades de cada um e decide quem deve fazer cada tarefa inicialmente, porém não atribuem todas as tarefas no começo, mas colocam todas no quadro *story board*. À medida que as tarefas vão sendo cumpridas, os membros da equipe vão atualizando os cartões (Cada cartão representa uma tarefa) ao longo do quadro. A tendência é que a próxima tarefa que um membro pegue é aquela que está em seguida na prioridade do *sprint backlog*, mesmo que ele não tenha certa habilidade para desempenhá-la. Isto irá fazer com que a equipe fique cada vez mais multifuncional.

Os *bugs* ou erros provenientes de implementações realizadas pela equipe em *sprints* anteriores devem ser sinalizados de forma diferenciada no quadro e entram como prioridade máxima, ou seja, acima de todas as outras que foram planejadas.

2.2.1.4.1 *Story Board*

O quadro *story board* é uma representação visual de uma parte do *product backlog* e consequentemente do *sprint backlog* como um todo. Nele está visualmente amostra tudo aquilo que a equipe se propôs a desenvolver no *sprint* corrente. A Figura 5 mostra um exemplo ilustrado de um quadro *story board* genuinamente nos moldes da metodologia Scrum e na sua forma mais simplificada. Nele temos as seguintes divisões:

- *Stories*: cada cartão desta coluna representa uma estória. São os únicos cartões que não são movimentados.
- *Tasks to do* ou simplesmente *to do*: cada cartão desta coluna representa uma tarefa a ser feita e que ainda não foi iniciada. Ao iniciar uma das tarefas, o membro da equipe deve escrever seu nome no cartão e movimentá-lo para a coluna seguinte.
- *Work in progress* (WIP) ou simplesmente *doing*: cada cartão desta coluna representa uma tarefa que está em andamento. Ao terminar de executar uma tarefa, o membro da equipe deve movimentar o seu respectivo cartão para a coluna seguinte.
- *Done:* cada cartão desta coluna representa uma tarefa que já está concluída.

Quando todas as tarefas de uma estória estão na última coluna (*done*) pode-se dizer que esta estória está concluída e pronta para passar pela validação do PO ou do cliente. Vale lembrar que a equipe deve evitar ao máximo começar a desenvolver as estórias seguintes caso ainda não tenham concluído a estória anterior. Essa atitude visa evitar o paralelismo entre as tarefas.

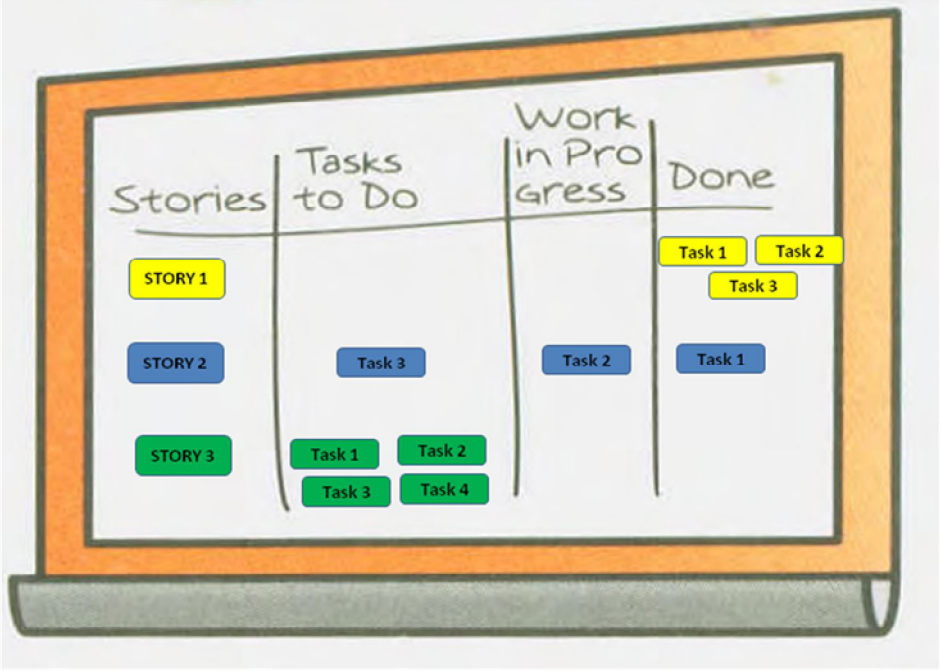


Figura 5 – Ilustração de um quadro *story board*.

2.2.1.4.2 *Sprint Burn-down*

Sprint burn-down é um gráfico que nos fornece um parecer diário sobre as equipes, ou seja, a sua produtividade. Permite o acompanhamento do progresso do comprometimento das equipes semanalmente e se a equipe vai cumprir os seus objetivos que foram traçados.

O gráfico ajuda na tomada de decisões, tais como:

- A equipe deve acelerar o trabalho ou continuar no mesmo ritmo.
- Se a equipe está indo rápido, pode-se puxar tarefas do próximo *sprint*.
- Se a equipe está indo devagar, deve-se tomar alguma ação para que ela consiga atingir a meta do *sprint* e cumprir todas as tarefas.

Na Tabela 2, tem-se um exemplo de um *sprint* de 10 dias, onde consta o *burn-down* que foi planejado de acordo com as horas previstas para cada tarefa. Quando as tarefas previstas são concluídas ao final de cada dia, subtrai-se o valor referente a elas do total previsto em horas. Caso contrário o valor continua como está, como se nota no dia 1 na linha do *burn-down* cumprido.

Tabela 2 – Exemplo de um *burn-down* planejado e cumprido no decorrer de um *sprint*.

	Início	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Dia 8	Dia 9	Dia 10
Burn-down planejado	100	90	80	70	50	45	40	30	20	10	0
Burn-down cumprido	100	100	90	85	70	50	40	25	10	5	0

A Figura 6, é o *sprint burn-down* resultante da Tabela 2. Esta é mais uma forma visual, além do *story board*, de poder acompanhar o andamento do sprint e o trabalho da equipe.

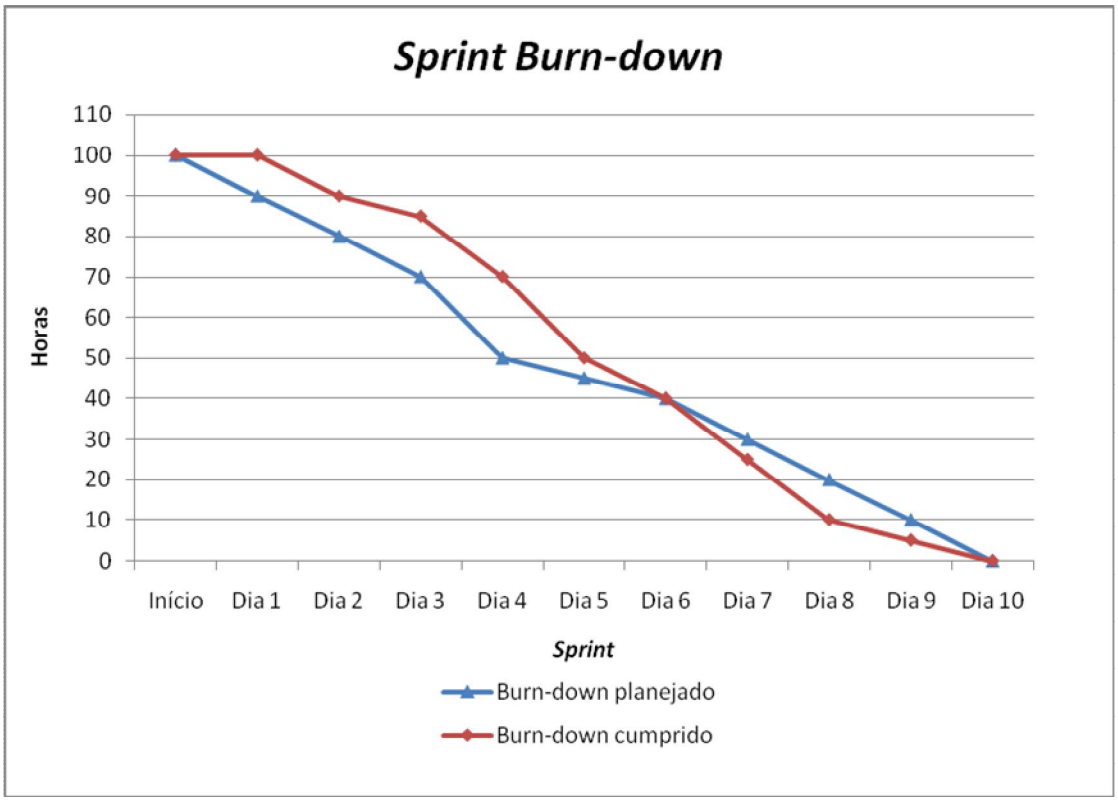


Figura 6 – Exemplo de um *burn-down* planejado e cumprido no decorrer de um *sprint*.

2.2.1.5 Reuniões de *feedback*

2.2.1.5.1 *Daily Scrum*

Todos os dias a equipe e o Scrum *master* fazem uma reunião diária. Recomenda-se que esta reunião seja realizada no mesmo local e hora todos os dias, para que vire realmente um

compromisso da equipe e também se torne uma reunião pública, em que gerentes e diretores possam participar quando tiverem interesse e puderem.
Esta reunião é curta e não deve exceder um *time-box* de 15 minutos. É importante compreender que a reunião apesar de pública é para a equipe e o Scrum *master*. Observadores são bem-vindos para participar, mas eles não devem interromper a reunião para fazer observações. A equipe é responsável por atingir a meta da sprint, e não o observador.

Esta reunião tem os seguintes benefícios:

- Visibilidade entre o grupo, toda a equipe sabe o que vai fazer.
- Permite que o Scrum *master* tome nota dos impedimentos e possa resolvê-los, garantindo que a equipe não perca produtividade. Evitando ao máximo que a equipe seja interrompida tanto por agentes externos como internos.

Durante a reunião, cada membro da equipe deve responder as três seguintes perguntas:

- O que fiz desde o nosso último encontro?
- O que pretendo fazer até o nosso próximo encontro?
- Houve ou ainda há algo que esta me impedindo de fazer o meu trabalho?

O Scrum *master* deve assegurar que a equipe não divague, qualquer assunto que está fora das três questões acima devem ser discutidas fora do cotidiano da reunião. Ele deve assegurar também que o *time-box* não seja ultrapassado.

Após cada membro responder as três perguntas, o Scrum *master* deve se atentar se algum deles não cumpriu o que foi planejado desde a última reunião. Caso algum não tenha cumprido, o Scrum *master* deve fazer as seguintes perguntas:

- O que estava o impedindo de progredir ainda persiste?
- Irá terminar hoje?
- Podemos dividir a sua tarefa?
- Precisa de ajuda de outro membro para conseguir prosseguir?

E assim o Scrum *master* tem a responsabilidade de agir sobre os impedimentos e resolvê-los para a equipe. Impedimentos não resolvidos farão com que a equipe não cumpra a sua meta do sprint. Se o Scrum *master* não puder resolver o impedimento, em seguida, deve procurar um nível superior para resolvê-lo.

2.2.1.5.1 *Sprint Review* e *Sprint Retrospective*

O *sprint review* é a reunião de entrega de tudo o que foi feito no *sprint.* O time deve fazer a entrega do *sprint* para o PO de maneira formal nesta reunião de revisão. É neste momento que o ciclo PDCA se fecha, onde os problemas levantados geram oportunidades de melhorias para o PO e para o time. Após a *sprint review* o time deverá fazer a reunião de retrospectiva, *sprint retrospective*, sem a presença do PO para que possam discutir sobre sugestões de melhoria e os problemas que ocorreram durante o *sprint*, levantando planos de ação para executar as melhorias e para que os erros e problemas do *sprint* anterior não ocorram no próximo.

2.3 Sistemas de Produção

Tubino (2000, p.27) apresenta de maneira ampla as classificações dos sistemas de produção, distinguindo da seguinte forma:

a) Pelo grau de padronização

- **Sistemas que produzem produtos padronizados:** bens ou serviços que apresentam alto grau de uniformidade e são produzidos em grande escala.

- **Sistemas que produzem produtos sob medida:** bens ou serviços desenvolvidos para um cliente específico.

b) Pelo tipo de operação

- **Processos contínuos:** envolvem a produção de bens ou serviços que não podem ser identificados individualmente.

- **Processos discretos:** envolvem a produção de bens ou serviços que podem ser isolados em lotes ou unidades e identificados em relação aos demais. Podem ser subdivididos em:

  · **Processos repetitivos em massa:** produção em grande escala de produtos altamente padronizados.

  · **Processos repetitivos em lote:** produção em lotes de um volume médio de bens ou serviços padronizados.

  · **Processos por projeto:** atendimento de uma necessidade específica dos clientes, o produto concebido em estreita ligação com o cliente tem uma data determinada para ser concluído. Uma vez concluído, o sistema de produção se volta para um novo projeto.

c) Pela natureza do produto

- **Manufatura de bens:** quando o produto fabricado é tangível.

- **Prestador de serviços**: quando o produto gerado é intangível.

Analisando estas classificações podemos enquadrar a fábrica de *software* num sistema de produção. Seu grau de padronização é baixo, onde produz produtos sob medida. Seu tipo de operação é um processo discreto e por projetos. Pela natureza do seu produto é um prestador de serviço.

2.4 Sistema Toyota de Produção

O Sistema Toyota de Produção surgiu através de um conjunto de práticas e políticas empresariais desenvolvidas pela indústria automobilística *Toyota Motor Company*. Seus fundamentos representam uma evolução do Controle da Qualidade Total e foram difundidos para o ocidente após a crise do petróleo na década de 70.

O objetivo principal do STP é aumentar a eficiência da produção, através da eliminação consistente dos custos desnecessários (Ohno, 1997), técnica que foi denominada por Shigeo Shingo, um de seus formuladores, como princípio do “não-custo” (Shingo, 2002).

Dentre os principais conceitos e técnicas que compõem o Sistema Toyota de Produção destacam-se:

a) A visão de que os sistemas de produção constituem-se numa rede funcional de processos e operações, onde os processos correspondem ao fluxo de materiais ou serviços ao longo do tempo e do espaço e as operações se referem às ações efetuadas pelos trabalhadores e máquinas;

b) A garantia da qualidade baseada no padrão zero defeitos e em sistemas à prova de erros (*Poka-yoke*);

c) O método *Kanban* utilizado para manutenção do fluxo de produção contínuo, onde as operações acontecem apenas no seu devido tempo (*Just-in-time*);

d) A redução nos tempos de *Setup* através do método da troca rápida de ferramentas;

e) O controle visual da produção, cuja finalidade é fornecer uma visão de síntese sobre a produção e, informar a ocorrência de anormalidades de maneira rápida, de forma a antecipar as medidas necessárias para a sua correção.

Resumidamente pode-se afirmar que o STP baseia-se na priorização das melhorias na função processo, via a eliminação contínua e sistemática das perdas ocorridas nos sistemas produtivos.

Sua adoção tem proporcionado maior competitividade às empresas através da maior rapidez no atendimento de necessidades que impliquem em maior flexibilidade operacional, menores custos, melhor qualidade e satisfação dos clientes.

2.4.1 *Just-in-time* e Autonomação

Os fundamentos que sustentam o Sistema Toyota de Produção na sua busca pela absoluta eliminação do desperdício são o *just-in-time* e a autonomação, a automação com toque humano.

Segundo Shingo (2002), *just-in-time* significa no momento certo, isto é, a utilização dos itens necessários, na quantidade necessária e no momento exato. Esta é a principal estratégia para atingir a produção sem estoque. O *just-in-time* trabalha com estoque zero, isto é, o mínimo estoque necessário para não prejudicar o *takt-time* e não gerar custos que não agregam valor ao produto final, onde com estoque gera-se desperdício. Outra ferramenta importante é o *kanban*, que é o meio para se obter o *just-in-time*. Monden (1984) ressalta que o *kanban* é um sistema de informação para controlar de forma harmônica as quantidades de produção em todos os processos. A Figura 7 demonstra um comparativo entre o *just-in-time* (Produção Puxada) e o *just-in-case* (Produção Empurrada), onde no *just-in-time* o estoque é eliminado completamente.

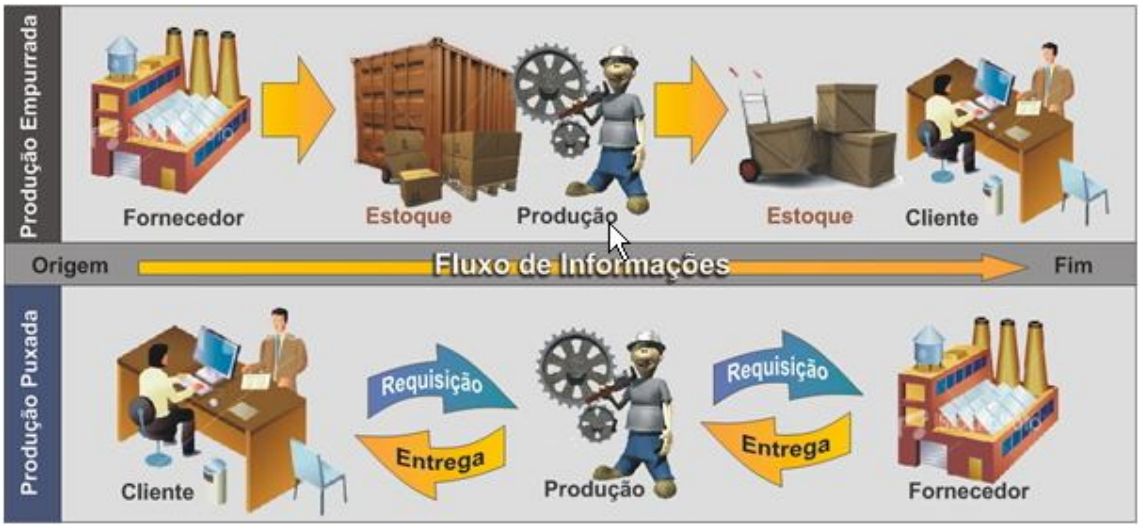


Figura 7 – Produção empurrada versus produção puxada (Freire, 2009).

Ohno (1997) destaca que, analisando pelo ponto de vista da produção, este é um estado ideal. Um problema no início do processo sempre resulta em um produto defeituoso no final do processo.

A autonomação, ou *jidoka* em japonês, consiste em dar inteligência humana à máquina e, ao mesmo tempo, adaptar o movimento simples do operador humano às máquinas autônomas. Estas máquinas estão equipadas com um dispositivo de parada automática, na presença de qualquer anormalidade. Não se produz produtos com defeitos.

Parar a máquina, quando ocorre um problema, força todos a tomar conhecimento do fato, isto é, com a devida atenção e compreensão do problema, procura-se a inexistência da recorrência dele.

2.4.2 *Kanban*

O sistema *kanban* (cartão ou registro visível em japonês) de cartões existe para programar e controlar a produção e o estoque. Os cartões, de papel ou plástico, têm a função de situar a autorização para alguma atividade de produção ou de estoque. Há vários tipos de cartões utilizados no *kanban*: cartões que autorizam produção; cartões que autorizam fornecimento; cartões que autorizam movimentação do material necessário de posição para outra. Além disso, os cartões podem ter cores diferentes para dar prioridade da operação (normal, moderada e alta).

Segundo Moura (Moura, 2003) há algumas verdades que precisamos saber sobre o sistema *Kanban*:

- *Kanban* não é inventário zero.
- *Kanban* e *Just-in-Time* não são sinônimos.
- "Você pode introduzir o *Kanban* a qualquer momento e com qualquer nível de estoque. Porém, se você não aproveitar o potencial do *Kanban* para identificar os problemas e aumentar a produtividade, você não está utilizando totalmente o sistema *Kanban*". (Taiichi Ohno - criador do sistema *Kanban*).
- O tamanho do seu inventário é o tamanho da sua incapacidade de resolver os problemas.

- É uma miopia introduzir o *Kanban* com o propósito de "não ter problemas". Os problemas de programação e controle da produção desaparecerão, mas surgirão outros tipos de problemas. Procure eliminar as causas e não culpar o *Kanban* pelos problemas.
- *Kanban* proporciona maior flexibilidade.
- *Kanban* é simplicidade, visibilidade e controle visual.
- O ponto principal do *Kanban* é por ordem na casa.
- O objetivo inicial do sistema *Kanban* é mostrar onde você tem problemas (com setup, gargalos, qualidade, manutenção, layout etc.).
- O objetivo final do sistema *Kanban* é acabar com os *kanbans.*

Da necessidade de controlar o sistema de forma a não produzir além das necessidades reais dos clientes, surgiu o *kanban*. Quando aplicado à produção o termo *kanban* ganha o significado de gestão visual da produção. Não há relação direta com outros conceitos ou ferramentas. O conceito básico e fundamental do *kanban* é: controle visual. Produção puxada, *poka-yoke*, *setup* rápido, etc. são ferramentas que colaboram para a eliminação ou redução de desperdícios e atingimento do *just-in-time*, mas não são *Kanban*.

Pode-se dizer que o *kanban* é apenas uma das ferramentas para a redução e eliminação de desperdícios que contribui para que o processo produza e entregue na exata medida das necessidades do cliente (*just-in-time*).

2.4.3 *Takt-time*

O *takt-time* é o tempo disponível para a produção dividido pela demanda do mercado. Ele organiza o modo como a matéria-prima avança pelos processos, e seu objetivo é alinhar a produção à demanda com precisão, ordenando o ritmo do sistema. Numericamente é o resultado da razão entre o tempo disponível para produção e o número de unidades a serem produzidas.

Durante os processos de produção, na ocorrência de problemas de qualidade, as rotinas de operação padrão são montadas de maneira que o avanço da produção seja de acordo com o

takt-time, ou seja, no planejamento do *takt-time* está sendo considerado que com a imediata correção do defeito, isto não influenciará no fluxo de produção. O *takt-time* funciona de forma conjunta com o conceito de autonomação, que propõe a parada da linha de produção, por qualquer operário, na detecção de algum problema. Neste caso o problema será avaliado por um supervisor e ele fará a avaliação se o problema poderá ser corrigido dentro do *takt-time*. Assim como o *kanban*, o *takt-time* está diretamente relacionado ao gerenciamento do sistema de produção tendo como foco a função do processo.

2.4.4 *Poka-yoke*

O *poka-yoke* é um mecanismo a prova de erros com a finalidade de evitar a ocorrência de defeitos em processos de fabricação e/ou na utilização de produtos e ferramentas. Originalmente foi chamado de *baka-yoke* (à prova de tolos), mas em 1963, uma trabalhadora da Arakawa Body Company, recusou-se a usar o mecanismo na sua área de trabalho devido ao termo ter uma conotação ofensiva. Assim o termo foi alterado para o atual *poka-yoke* (à prova de erros). Este dispositivo em si, não é um sistema de inspeção, segundo Shingo (2002), mas um método de detectar defeitos ou erros que pode ser utilizado para satisfazer uma determinada função de inspeção. O *poka-yoke* pode ser usado de duas maneiras para evitar os erros:

- Método de Controle: quando o *poka-yoke* é ativado, a máquina ou linha de processamento pára, de forma que o problema pode ser corrigido. O Método de Controle pode ser dividido em três subtipos: Método de Contato, Método de Conjunto e Método de Etapas. Enquanto o método de contato identifica os defeitos em função da existência ou não de contato entre o dispositivo e alguma característica ligada à forma ou dimensão do produto, o método de conjunto determina se um dado número de atividades previstas são executadas. O método das etapas determina se todos os processos operacionais estabelecidos por um procedimento são seguidos. Conforme Shingo (2002), o *poka-yoke* de controle é o mais eficiente na maioria dos casos.
- Método de Advertência: quando o *poka-yoke* é ativado, um alarme soa, ou uma luz sinaliza, visando alertar o trabalhador. Este método é normalmente recomendado quando a freqüência do defeito é baixa e quando ele pode ser corrigido. A escolha do método de *poka-yoke* deve ser feita com base no custo benefício.

2.4.5 *Kaizen* e o Ciclo PDCA de Melhoria Contínua

A melhoria contínua é baseada em um conceito japonês denominado *Kaizen*, que consiste no ponto principal da filosofia da qualidade total, qual seja, a idéia da busca contínua de melhorias em tudo o que é feito em uma organização. Significa melhoria gradual e contínua de todos os produtos e serviços, descobrindo no dia-a-dia a forma de tornar os processos cada vez mais eficientes, mais econômicos e mais confiáveis. Este conceito envolve o desenvolvimento de uma cultura de aperfeiçoamento constante em todas as atividades da empresa.

Kaizen é uma palavra japonesa que tem como origem as palavras *Kai* (Mudar) e *Zen* (Melhor), ou seja, melhoria contínua. Seu conceito baseia-se no fato de que nada está bom, apenas ficou melhor. No Sistema Toyota de Produção, qualquer proposta de melhoramento em qualquer hora é bem vinda e estudada, buscando implementá-la da maneira mais rápida possível. O conceito de *kaizen* é tão forte, que existe uma história que é uma analogia ao *kaizen*. O Tesouro de Bresa é a historia de um alfaiate pobre na busca por um tesouro descrito em um livro. Para conseguir desvendar os segredos do livro, o alfaiate teve que passar por um processo de melhoria contínua, aprendendo cada vez mais coisas que não sabia, para buscar o sonhado Tesouro de Bresa. No final da história o alfaiate que já não é mais alfaiate, e sim um assessor do rei, descobre que o tesouro não existe como imaginava. O Tesouro de Bresa é o conhecimento que ele adquiriu na busca por ele. Na teoria a principal estratégia do *kaizen* é que nenhum dia deve passar sem que algum tipo de melhoramento tenha sido feito em algum lugar da empresa. Um dos principais motivos da fama do Sistema Toyota de Produção é a aplicação do princípio do *kaizen*.

Há muitos modelos descritos na literatura para se conseguir melhoria, mas, talvez o mais conhecido e utilizado de todos seja o ciclo PDCA. Devido à sua simplicidade, o PDCA é o modelo de referência para os planos de melhoramento contínuo adotados por inúmeras organizações, proporcionando uma linguagem comum a todos na melhoria contínua da qualidade. PDCA são as iniciais das palavras inglesas *plan, do, check e act*, que significam

planejar, fazer, verificar e agir (corretivamente). A Figura 8 ilustra o ciclo PDCA, introduzido por Shewhart e popularizado por Deming e Ishikawa.

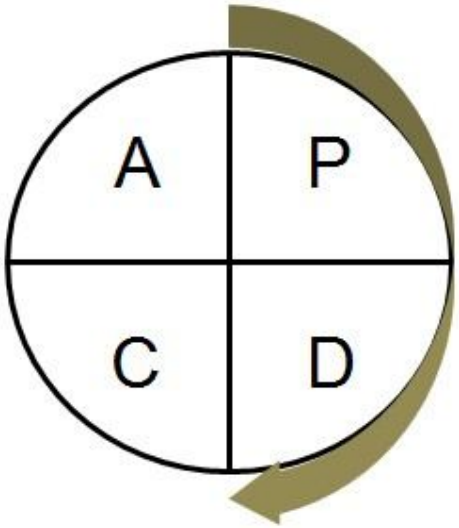


Figura 8 – Ilustração do ciclo PDCA.

2.4.6 Cinco vezes "por quê?"

Na ocorrência de um problema repetindo “por quê?” cinco vezes você pode identificar a raiz do problema. O exemplo que Ohno (1997) usa é:

Suponha que uma máquina parou de funcionar.

1. Por que a máquina parou?

Porque houve uma sobrecarga e o fusível queimou.

2. Por que houve uma sobrecarga?

Porque o mancal não estava suficientemente lubrificado.

3. Por que não estava suficientemente lubrificado?

Porque a bomba de lubrificação não estava bombeando suficientemente.

4. Por que não estava bombeando suficientemente?

Porque o eixo da bomba estava gasto e vibrando.

5. Por que o eixo estava gasto?

Porque não havia uma tela acoplada e entrava limalha.

Se não utilizasse esta ferramenta neste caso, provavelmente só trocariam o fusível, e brevemente o problema ocorreria novamente. Um dos principais motivos da evolução do Sistema Toyota de Produção foi a utilização desta ferramenta. Cada problema que surgia no desenvolvimento do Sistema, se fazia os cincos porquês, e logo encontravam a verdadeira causa do problema.

Cinco vezes “por quê?” é uma metáfora, pois as raízes dos problemas de forma geral podem ser encontradas em duas ou vinte perguntas, onde o importante é encontrar uma explicação para o evento de modo que ele não se repita.

2.4.7 5W2H

Shingo (2002) destaca a importância da cura pelas causas reais dos problemas e perdas. Aliado à ferramenta das repetições dos cinco porquês, o 5W2H, originalmente 5W1H, é uma espécie de *check-list* utilizado para garantir que a operação de busca por problemas seja eficaz. Os 5 W correspondem às seguintes palavras do inglês: *What* (o que), *Who* (quem), *When* (quando), *Where* (onde) e finalmente *Why* (por que). Os 2 H correspondem a *How* (como) e *How Much* (Quanto custa). Originalmente, diferente de hoje, para o método não interessava o segundo H (quanto custa).

2.4.8 *Andon*

Sistema de controle visual, usado no piso da fábrica para ajudar a gerência e os operários a visualizarem a produção. Existem dois tipos de *andon*: Um é o quadro da meta da produção e o outro é um sinal de trânsito, com lâmpadas verde, amarela e vermelha, as quais simbolizam o estado da qualidade, ou os problemas, em cada posto de processamento.

- **Quadros da meta de produção diária**: São geralmente quadros suspensos que são constantemente atualizados durante o turno.

- **Identificação de problemas**: Lâmpadas vermelha, amarela e verde que mostram visualmente a posição de um posto de processamento ou uma máquina. Pode até ser

uma simples lâmpada ligada a um dispositivo de inspeção, mostrando uma condição "passa-não-passa".

2.4.9 O Poder do Trabalho Individual e do Trabalho em Equipe

A implementação da autonomação esta a cargo dos gerentes e supervisores de cada área da produção. A chave está em dar inteligência humana à máquina e, ao mesmo tempo, adaptar o movimento simples do operador humano às maquinas autônomas. Qual é a relação entre *just-in-time* e automação com um toque humano, os dois pilares do Sistema Toyota de Produção? Utilizando a analogia de um time de beisebol, a autonomação correspondente à habilidade e ao talento dos jogadores individuais, ao passo que o *just-in-time* é o trabalho da equipe envolvida em atingir um objetivo preestabelecido.

Por exemplo, um jogador na área do campo externo nada terá a fazer enquanto o jogador que atira a bola ao batedor não tiver problemas. Mas um problema – o batedor oponente rebatendo a bola, por exemplo – ativa o jogador que está no campo externo, que pega a bola e a lança para o jogador que fica numa das três bases *just-in-time* para tirar o corredor da jogada.

Gerentes e supervisores numa fábrica são como o gerente da equipe e os treinadores do batedor, do base e do jogador que fica no campo externo. Um time de beisebol muito bom já dominou as jogadas, os jogadores podem enfrentar qualquer situação com a ação coordenada. Na manufatura a equipe da produção que tenha dominado o sistema *just-in-time* é exatamente como um time de beisebol que joga bem em equipe.

A autonomação, por outro lado, desempenha um duplo papel. Ela elimina a superprodução, um desperdício significativo na manufatura, e evita a produção de produtos defeituosos. Para conseguir isso, procedimentos de trabalho padronizados, correspondendo às habilidades de cada jogador, devem ser obedecidos sempre. Quando surgem anormalidades – isto é, quando a habilidade de um jogador não pode ser manifestada – instruções especiais devem ser dadas para trazer o jogador de volta ao normal. Este é um importante dever do treinador.

3 DESENVOLVIMENTO

3.1 Preparação

A preparação é semelhante ao tradicional "iniciação e planejamento" do PMBOK. Aplica-se uma abordagem leve para garantir que se produza um trabalho rapidamente. Durante a fase de preparação realiza-se o seguinte:

● Definir a visão do produto.
● Criar um *product backlog* inicial.
● Priorizar o *product backlog*.
● Estimar o tamanho do *product backlog*.

Esta fase pode levar uma questão de dias ou semanas. O objetivo é obter aprovação e iniciar rapidamente a entrega de valor do produto ao cliente.

3.2 Definição da Visão do Produto

A visão do produto é de responsabilidade do PO e é fundamental criá-la com muita competência, pois representa como o sistema deve funcionar para atender a necessidade do cliente, caso a definição da visão seja errada, haverá insatisfação do cliente e aumento do custo do produto.

A visão do produto permite que todos os envolvidos compreendam o grande objetivo do projeto.

Esta visão guia o desenvolvimento de atividades e fornece orientações para a equipe sobre o que somos e o que teremos de alcançar.

Os seguintes itens podem contribuir para direcioná-lo a criar a visão do produto:

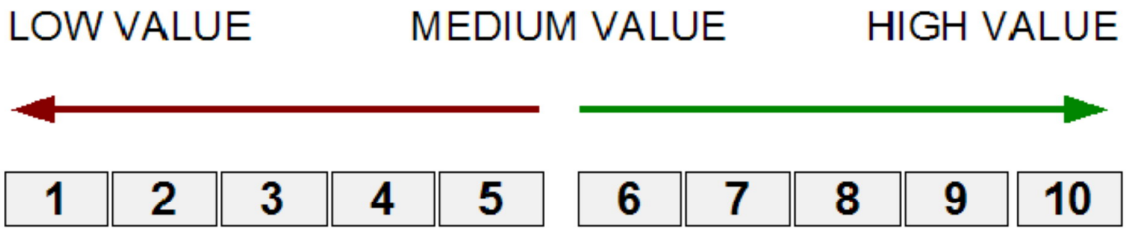
- As pessoas (por exemplo, clientes, usuários).

- A Organização.
- Qualidade.
- Inovação e criatividade (por exemplo, diferenciação de mercado).
- Ambiente (por exemplo, concorrentes, parceiros).
- Percepção.

3.3 Priorizando o *Product Backlog*

Cada item sobre o *product backlog* deve ter prioridade. Esta prioridade nos diz quais os itens que devem ser entregues primeiramente. Para priorizar os itens do *product backlog* o PO deve determinar o valor do negócio em cada item utilizando como exemplo a Tabela 3, em que *low value* é um valor baixo, *medium value* é um valor médio e *high value* é um valor alto.

Tabela 3 – Exemplo de tabela para dar valor aos itens do *product backlog*.



Aqui estão algumas diretrizes para auxiliar na determinação do valor comercial dos itens do *product backlog*:

- Qual é o benefício financeiro para introduzir esse recurso? Será que isso vai aumentar a receita ou o esforço do desenvolvimento custa mais do que realmente vale?
- Será que esse recurso aumenta a produtividade e usabilidade ou vai causar frustração e ressentimento para com o cliente e a Organização?
- Se eu introduzir esse recurso, irá resultar em grandes custos de desenvolvimento para os nossos usuários ou clientes?
- Se eu introduzir esse recurso, irá resultar em uma mudança nos processos de negócio? Se sim, estamos levando em conta os custos dessa mudança?
- Qual é o trabalho se eu não implementar este recurso?

- Se eu introduzir esse recurso, irá resultar em um aumento no risco ou vai eliminar uma série de riscos que estão atualmente evidentes?
- Será que esse recurso pode nos diferenciar no mercado?

3.4 Estimando os Tamanhos dos Itens do *Product Backlog*

Uma excelente maneira, divertida e eficaz da estimativa dos tamanhos em horas dos itens do *product backlog*, é usar um conceito chamado *planning poker*. Esta reunião deve ter um *time-box* de modo a permitir despender cerca de 2 a 3 minutos por estória. Não se deve permitir que cada situação de estimativa se torne uma discussão sobre os requisitos das estórias.

Veja como é feito a seguir:

- A cada pessoa é dado um baralho de cartas contendo a seguinte série de números que correspondem ao número de horas: 0, 0,5, 1, 2, 3, 5, 8, 13, 20, 40, 100 e uma carta com um ponto interrogação "?" no caso da pessoa não saber opinar. As cartas não seguem uma sequência linear, pois de acordo com a metodologia Scrum, quanto maior o tempo estimado, maior é a incerteza, indicando assim que a estória deve ser quebrada em outras partes.
- O facilitador lê a estimativa do usuário da estória (item do *product backlog*) ou qualquer nota adicional.
- A equipe pode pedir ao PO (ou representante) algumas perguntas. O ponto é ter certeza de que a resposta é alto nível e não entrar em detalhes.
- Quando as perguntas são concluídas ou o tempo atribuído se esgotou, cada pessoa escolhe sua estimativa (uma carta), com base no tamanho e complexidade da estória e colocam a carta de bruços sobre a mesa. Os participantes devem abster-se de indicar o que eles tenham escolhido até as cartas serem viradas para cima, todas ao mesmo tempo.
- Se existir uma grande variação entre as cartas selecionadas, a equipe discute sobre a carta mais baixa e a mais alta até entender no que os membros da equipe estavam pensando para chegar a essa estimativa. Esta é uma boa oportunidade para tomar notas e usá-las mais tarde nas reuniões de *sprint planning*.

- Todos pegam suas cartas novamente, reavaliam a estória num curto espaço de tempo, escolhendo suas cartas e em seguida voltam a colocá-las de bruços sobre a mesa.
- Este processo é repetido até que as cartas de todos os participantes sejam iguais, ou seja, não é uma votação em que a maioria vence, pois todos devem entrar em um consenso, convencendo uns aos outros do que pensam.
- E assim segue para cada estória.

3.5 *Kanban* e *Story Board*

Com este trabalho verificou-se que o maior ponto de intersecção entre o STP e o Scrum, sem dúvida alguma, é entre o quadro *kanban* e o *story board*. Portanto este estudo foi aprofundado e terá como foco principal este tema.

Na Figura 9 logo abaixo, vemos um exemplo de um quadro *kanban* numa fábrica de manufatura, em que temos as peças dividas em três cores cada. Essas cores representam a necessidade de produção ou não. A cor vermelha representa o estoque de peças em estado crítico, a cor amarela representa o estoque em estado de atenção e a cor verde representa o estado normal.

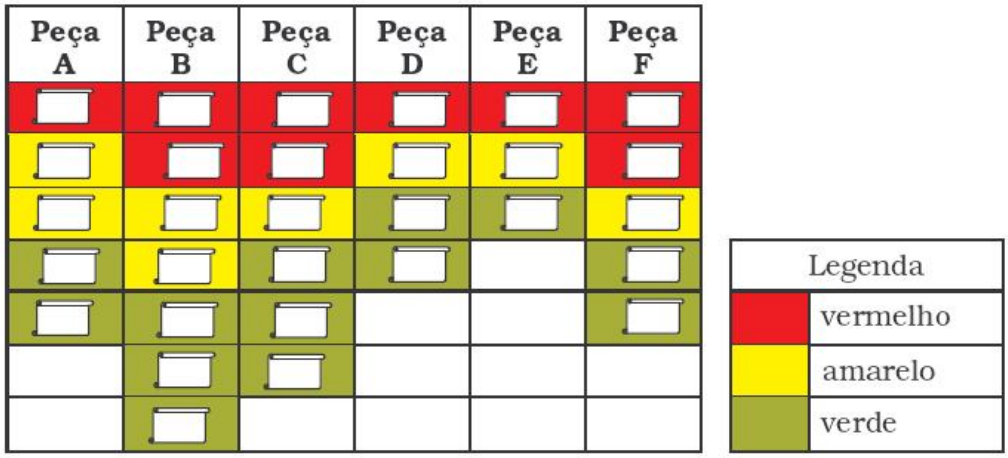


Figura 9 – Exemplo de um quadro *kanban* numa indústria (Peinado, 2007).

O simples fato de gerenciarmos a fábrica de software por cartões, como na Figura 10, já podemos chamá-lo de um sistema *Kanban*, pois o seu significado vem do japonês e quer dizer

registro ou cartão visual. Cada cartão azul representa uma tarefa e cada parte do quadro (*To do*, *doing*, *done*) representa seu status na fábrica, ou seja, à fazer, sendo feito e pronto. Nesta mesma Figura 10, para que as equipes pudessem interagir seus quadros foram implementados dois conceitos do STP. São eles:

- A forma de puxar a produção, em que a célula da equipe 2 puxa a tarefa da equipe 1, caso ela fique sem tarefas a serem feitas, gerando assim a necessidade de puxar mais trabalho.
- As cores (Vermelha, amarela e verde) indicativas nos dois quadros, para se ter a informação de "estoque" de tarefas, ou seja, as tarefas que estão prontas para serem entregues ao processo seguinte. Isto ajuda a controlar os gargalos e evitar desperdícios de recursos humanos na fábrica de *software*.

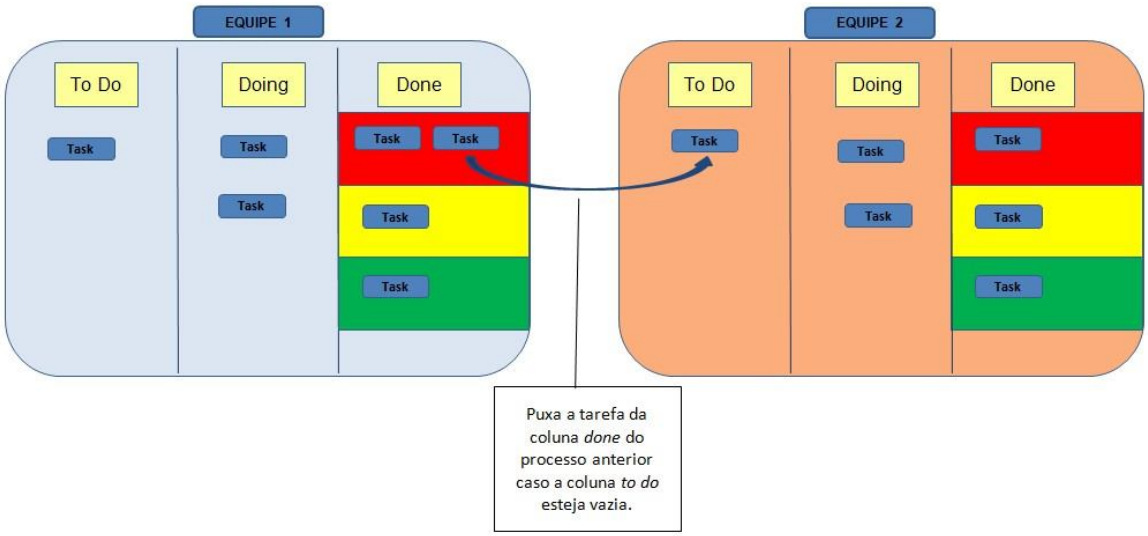


Figura 10 – Exemplo de interação entre os *story boards* utilizando conceitos do STP.

Cada quadro *kanban* ajuda o time a entender como eles estão indo, se bem ou mal, o que fazer em seguida e os torna auto direcionáveis.

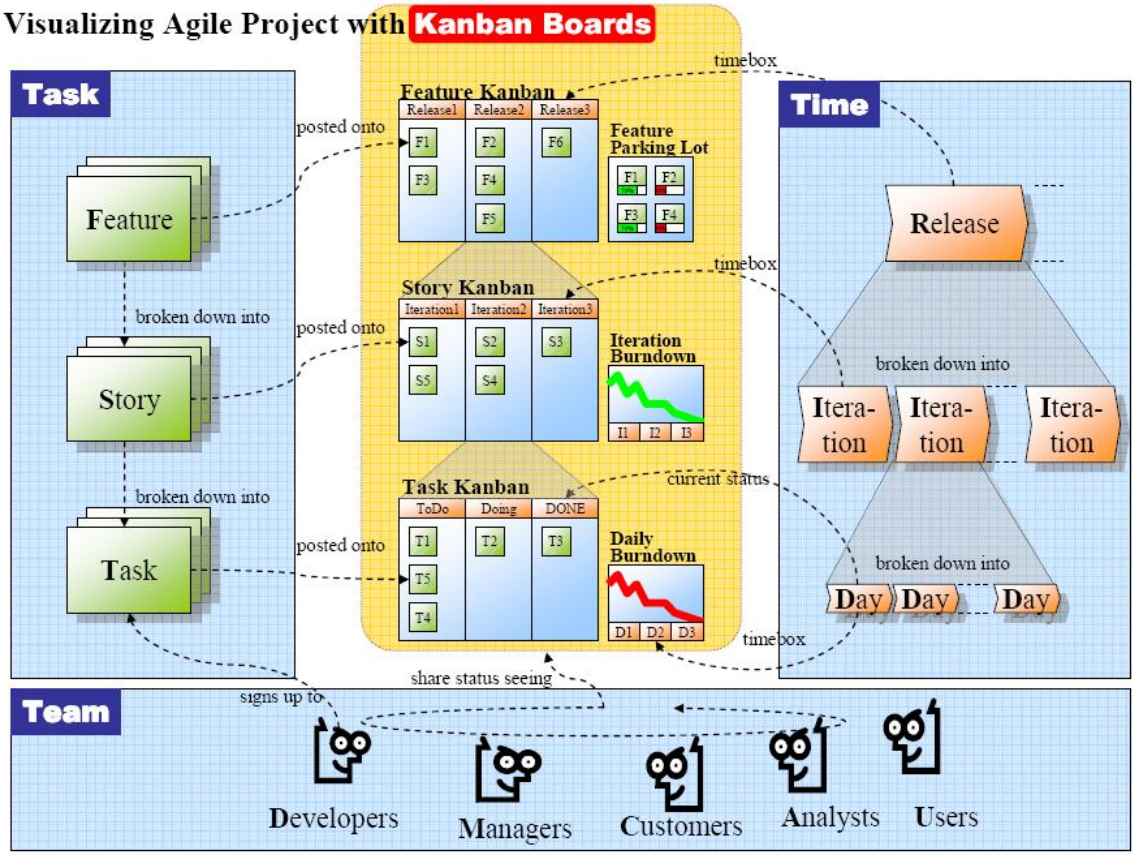


Figura 11 – Três pontos de vista e um quadro Kanban como um mapeamento de tarefas no tempo (Hiranabe, 2007).

A Figura 11 mostra como o time interage com o quadro *kanban* de acordo com as tarefas e o tempo. Na área *Task* (Tarefas), funcionalidades são quebradas em estórias, que por sua vez são quebradas novamente em tarefas. Na área *Time* (Tempo), as versões são quebradas em iterações, que por sua vez são quebradas novamente em dias, para que cada tarefa não tenha a duração maior do que um dia. A área *Team* (Time), mostra a interação dos desenvolvedores, gerentes, clientes, analista e usuários com todo este processo.

3.6 PDCA e *Sprint*

Na Figura 12 temos uma ilustração de como ficou a fusão entre as regras do *sprint* e o ciclo PDCA adotado pela fábrica de *software* em questão.

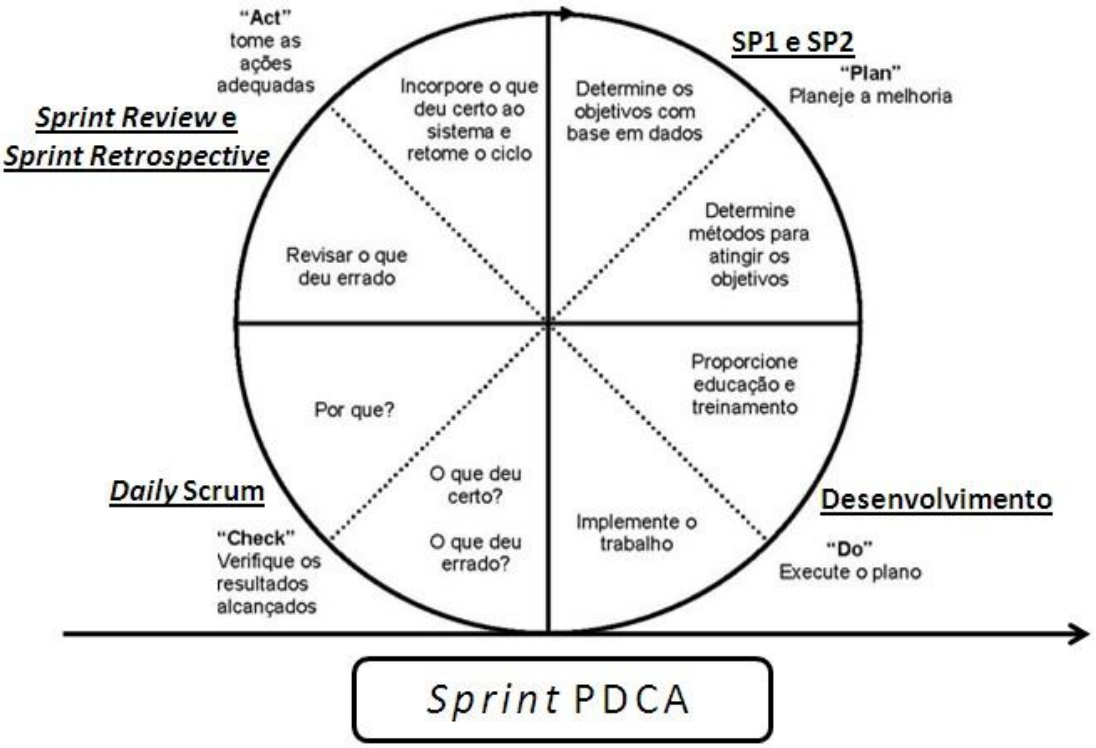


Figura 12 – Ilustração da fusão entre o as regras do *sprint* e o ciclo PDCA.

3.7 5W2H e 3W

A metodologia Scrum trabalha apenas com os 3 W, como foi visto no início do trabalho. A fábrica de software em questão resolveu adotar os 5 W e os 2 H do STP por ser mais abrangente, diminuindo assim que a probabilidade de análise de negócio não corresponda com as expectativas dos clientes. Para isso devem-se responder as seguintes perguntas.

- **1 –** ***W****hat* (o que será feito?).
- **2 –** ***W****ho* (quem fará?).
- **3 –** ***W****hy* (por que será feito?).
- **4 –** ***W****here* (onde será feito?).
- **5 –** ***W****hen* (quando será feito?).
- **1 –** ***H****ow* (como será feito?).
- **2 –** ***H****ow Much* (quanto custará?).

Na Tabela 4, temos um exemplo de uma análise de negócio feita nos moldes do 5W2H na fábrica de *software*.

Tabela 4 – Exemplo de análise de negócio com 5W2H.

ANÁLISE DE NEGÓCIO	
OBJETIVO	Criar tela de manutenção de ficha técnica
CLIENTE	Móveis XYZ
	TAREFA 1390
O QUÊ	Poder alterar as fichas técnicas de forma ampla, ou seja, que uma alteração seja possível ser replicada para várias outras.
ONDE	No departamento onde são feitas as manutenções das fichas técnicas da fábrica de móveis.
PORQUE	Para diminuir o tempo das manutenções das fichas ténicas.
COMO	Aqui são citadas as metodologias de desenvolvimento que serão utilizadas e a descrição dos requisitos.
QUEM	Equipe 1 e 2
QUANDO	No *sprint* 93
QUANTO	R$ 1.000,00

3.8 Estudo de Caso

Foi realizado pelo autor deste trabalho um estudo de caso em uma fábrica de *software* da cidade de Maringá, Paraná, que antes utilizava apenas UML (Unified Modeling Language) e PMBOK (Project Management Body of Knowledge) nos seus processos de fabricação e projetos de *softwares*. Começou-se a implantar a metologia Scrum juntamente com o Sistema Toyota de Produção, em Dezembro de 2008, e consequentemente os conceitos de *kanban*, PDCA e 5W2H na gestão da fábrica.

Ao final do estudo obtiveram resultados muito satisfatórios em relação ao intervalo de tempo entre o cadastro das tarefas (Ordens de serviço) pelos clientes até o seu término (Finalização e entrega destas ordens de serviço), que nada mais é do que o *lead time* da fábrica de *software*.

Foram filtrados nos relatórios usados para análise destes resultados todas as tarefas criadas para corrigir erros, tirar dúvidas de usuários e implementar novas funcionalidades entre julho de 2008 à julho de 2009, ou seja, o ponto em que a empresa implantou o novo sistema de trabalho está praticamente ao meio da faixa coletada.

3.8.1 Resultados

De acordo com os resultados, pôde-se concluir que o tempo médio para uma tarefa ser considerada como finalizada foi decaindo consideravelmente logo após a novas metodologias terem sido implantadas, mesmo o número total de tarefas ter se mantido em uma média relativamente constante como pode ser observado na Figura 13 e 14.

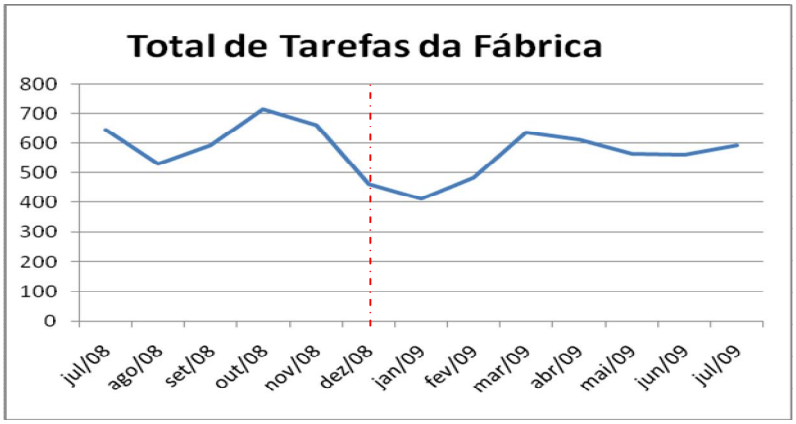


Figura 13 – Total de tarefas (Ordens de Serviço) cadastradas por mês para a empresa.

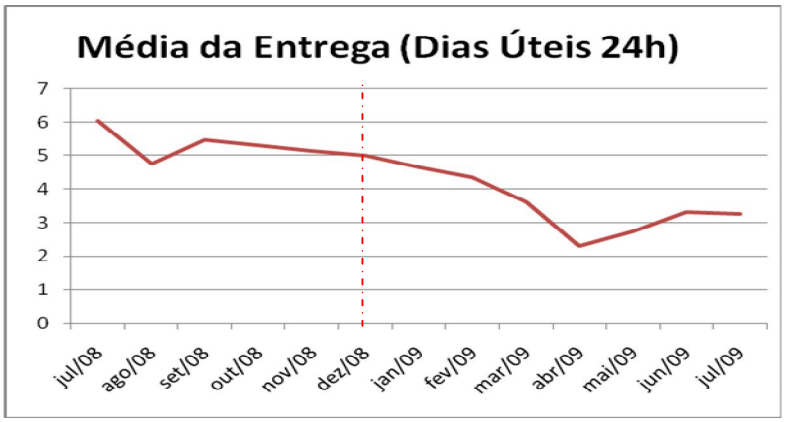


Figura 14 – Média de dias úteis que uma tarefa levou para ser considerada finalizada.

4 CONCLUSÃO

4.1 Conclusão

Scrum não é um método fim de gestão, mas sim um meio, uma filosofia que incentiva a implementação de novas ferramentas de trabalho, aperfeiçoando e maximizando os resultados positivos e minimizando os erros e retrabalhos pela gestão visual e o trabalho em equipe. É nesta concepção que a Engenharia de Produção, representada pelo Sistema Toyota de Produção e pela Manufatura Enxuta vai paulatinamente adquirindo seu espaço na Fábrica de Software, através das metodologias ágeis de desenvolvimento, dentre elas em principal o Scrum.

Este trabalho não inventou uma nova metodologia, mas apenas somou duas formas de trabalho para se obter um melhor aproveitamento e entendimento daquilo que está proposto a ser feito. Em geral levantou-se pontos em comum entre os conceitos do Sistema Toyota de Produção e o Scrum, porém sem a intenção de comparar e analisar qual deles é o melhor, mas sim mostrar que a essência dessas metodologias são exatamente as mesmas e que somadas podem surtir resultados muito positivos na redução do tempo de entrega.

4.1 Recomendações para Trabalhos Futuros

Como recomendações para trabalhos futuros, destacam-se:

a) Estudar o aumento da qualidade do produto.
b) Estudar a diminuição dos erros e de retrabalhos.
c) Estudar a eliminação de desperdício de tempo e trabalho.
d) Inserir mais ferramentas do Sistema Toyota de Produção no estudo.
e) Inserir conceitos de PMBOK junto ao STP e/ou ao Scrum.
f) Inserir os conceitos do Scrum em uma fábrica de manufatura.

REFERÊNCIAS

BECK, K., et al. ***"Manifesto for Agile Software Development"***. 2001. Http://www.agilemanifesto.org/.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistema com UML.** Rio de Janeiro: Elsevier, 2002.

COCKBURN, A.; HIGHSMITH, J. ***"Agile Software Development: The People Factor"***, IEEE Computer, vol.34, n.11, nov.2001, p.131-133.

DEMARCO, T.; LISTER, T. ***"Peopleware"***, 2a ed., Dorset House, 1998.

FOWLER, M. ***"The New Methodology"***, jun. 2002, http://martinfowler.com/articles/newMethodology.html#N8B.

FREIRE, Luiz. ***Just-in-time***. 2009. Disponível em: < http://www.luizfreire.com/lean/pullflow/index_br.php/>. Acesso em: 05/10/2009.

GLOGER, Boris. ***Scrum* vs. *Kanban***. 2009. Disponível em: <http://scrum4you.wordpress.com/2009/04/08/scrum-vs-kanban/>. Acesso em: 15/04/2009.

HIGHSMITH, J. "The Great Methodologies Debate: Part2", Cutter IT Journal, vol.15, n.1, jan. 2002.

HIRANABE, Kenji. ***Visualizing Agile Projects using Kanban Boards***. Disponível em: <http://www.infoq.com/articles/agile-kanban-boards/>. Acesso em: 15/08/2009.

JACOBSON, I. "A Resounding "Yes" to Agile Processes – But Also More", Cutter IT Journal, vol.15, n.1, jan. 2002.

MONDEN, Yasuhiro. **Sistema Toyota de Produção**. São Paulo: Instituto de Movimentação e Armazenagem de Materiais, IMAM. 1984.

MOURA, Reinaldo Aparecido. ***Kanban* - A Simplicidade do Controle da Produção.** 6. ed. São Paulo: Instituto de Movimentação e Armazenagem de Materiais, IMAM, 2003.

OHNO, Taiichi. **O Sistema Toyota de Produção:** além da produção em larga escala. Trad. Cristina Schumacher. Porto Alegre: Bookman, 1997.

PEINADO, Jurandir; GRAEML, Alexandre R. **Administração da Produção: Operações industriais e de serviços**. UnicenP, 2007.

PRESSMAN, Roger S. **Engenharia de Software**. 6. ed. São Paulo: McGraw-Hill, 2006.

SCHWABER, K. Agile **Processes and Self-Organization**, Agile Alliance, 2002, http://www.aanpo.org/articles/index.

SCHWABER, K; BEEDLE, M. **Agile Software Development with Scrum**. New Jersey: Prentice Hall, 2002.

SHINGO, Shigeo. **O Sistema Toyota de Produção do ponto de vista da Engenharia da Produção**. Bookman, 2002.

TUBINO, Dalvio Ferrari. **Manual de planejamento e controle da produção**. 2. ed. São Paulo: Atlas, 2000.

BIBLIOGRAFIA

AUGUSTO NETO, Álvaro; SANT, Nilson. **A Gestão dos Processos na Fábrica de Software.** 2003. Disponível em: <http://hermes2.dpi.inpe.br:1905/col/lac.inpe.br/worcap/2003/10.29.14.17/doc/Alvaro%20Workcap2003.pdf>. Acesso em: 01/04/2009.

BELEZE, C.; LEME, L. H. R. ***Kanban*** **na Fábrica de *Software***. In: XVI SIMPEP – Bauru, 09 a 11 de novembro. *Anais,* 2009.

BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivar. **UML:** Guia do Usuário. Rio de Janeiro: Elsevier, 2005 – 2ª reimpressão.

CORRÊA, Henrique L.; GIANESI, Irineu G. N. **Just in time, MRP II e OPT:** um enfoque estratégico. 2. ed. São Paulo: Atlas, 1993.

DINSMORE, Paul Campbell; SILVEIRA NETO, Fernando Henrique da. **Gerenciamento de Projetos:** como gerenciar seu projeto com qualidade dentro do prazo e custos previstos. Rio de Janeiro: Qualitymark, 2004.

FERNANDES, Aguinaldo A.; TEIXEIRA, Descartes S. **Fábrica de *Software*:** Implantação e Gestão de Operações. São Paulo: Atlas, 2004.

MARTINS, Petrônio G.; LAUGENI, Fernando Piero. **Administração da Produção.** 2. ed. São Paulo: Saraiva, 2005.

APÊNDICE

ANEXOS

GLOSSÁRIO