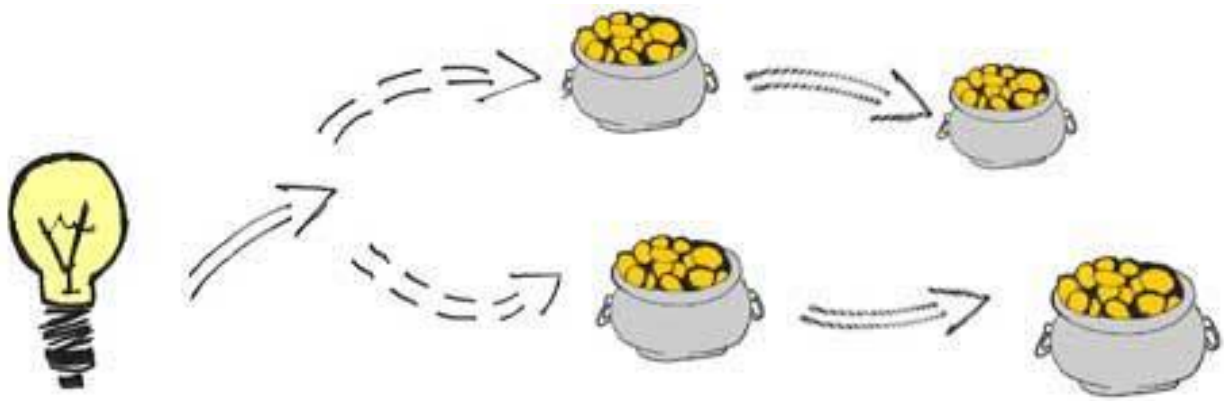
Rafael Dias Ribeiro,MSc,CSM,CSPO,ACP,PMP.
Horácio da Cunha e Sousa Ribeiro,MSc.

Métodos Ágeis

em Gerenciamento de Projetos



1ª Edição

Rio de Janeiro
Horácio da Cunha e Sousa Ribeiro
2015

Gerenciamento de Projetos com Métodos Ágeis

contato@cursospin.com.br
Av. Djalma Batista, nº. 946, sala 08 (Centro Empresarial Santo Remédio) - Vieiralves
Nossa Sra. das Graças – Manaus
Telefone (92) 3584-1966
Site: WWW.cursospin.com.br
contato@cursospin.com.br

Rafael Dias Ribeiro
Horácio da Cunha e Sousa Ribeiro

Gerenciamento de Projetos com Métodos Ágeis

1ª Edição

Rio de Janeiro
Horácio da Cunha e Sousa Ribeiro
2015

Direito Editorial
Horácio da Cunha e Sousa Ribeiro

Catalogação na fonte por: Edirlane Carvalho de Souza Freitas – CRB7-5463

R484g Ribeiro, Rafael Dias; Ribeiro, Horácio da Cunha e Sousa Ribeiro.

Gerenciamento de projetos com métodos ágeis / Rafael Dias Ribeiro, Horácio da Cunha e Sousa Ribeiro. Rio de Janeiro: [s.n.], 2015.
115 p.; il.; 23 cm.

Contém Referências
ISBN: 978-85-919102-1-2

1. Projetos. 2. Gerência de projetos. 3. Métodos ágeis. 4. SCRUM. 5. XP. 6. Engenharia. 7. Metodologia. I. Título.

CDD 658.404

Dedicatória:

Este livro é dedicado a Emanuel Bodstein Ribeiro o projeto mais complexo da minha vida.

Rafael Dias Ribeiro

Dedico este livro a Vera Lúcia Dias Ribeiro que foi inspiração para todo o meu trabalho.

Horacio da Cunha e Sousa Ribeiro

Conteúdo

CAPÍTULO 1

1.1 - Introdução

Pela necessidade de um aumento contínuo de competitividade, com o dinamismo e a velocidade com que a informação e o conhecimento circulam, o ambiente corporativo de várias empresas necessita utilizar a gerência através de projetos para se tornar mais eficaz, ágil e competitivo.

Há uma crescente demanda de profissionais detentores de habilidades para gerenciar projetos, que utilizem boas práticas e ferramentas para alcançar os objetivos do projeto e não apenas intuição ou bom senso. Para responder a esse desafio, colaboradores devem reunir atributos de conhecimentos técnicos sobre gerenciamento de projetos, usando de modo eficiente esse conhecimento, aumentando as chances de sucesso do projeto. Neste capítulo, iremos compreender o que realmente é um projeto e vamos identificar as características de projetos orientados a planos e de projetos orientados a valor.

É muito comum no meio corporativo, ouvirmos:

“Próximo mês, iniciamos o Projeto XYZ(...)”
“Estão todos dedicados ao Projeto XYZ(...)”

Mas será que o conceito de projeto é usado corretamente? Vamos entender o que é um projeto e suas características.

A definição do PMBoK®-*Project Management Body of Knowledge*, sobre projeto, é:

“Um PROJETO é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo.”

Observe que os projetos possuem características bem significativas,

como:

- Temporários, possuem um início e um fim definidos;
- Planejados, executados e controlados;
- Entregam produtos, serviços ou resultados exclusivos;
- Desenvolvidos em etapas, implementam uma elaboração progressiva; realizados por pessoas; sofrem restrições.

A empresa, assim como a execução de determinados projetos, é algo “vivo”, que muda seu comportamento durante a execução e, em momentos diversos, poderá exigir abordagens diferentes do gestor.

1.2 - Análises de Ambientes

No livro *Management* 3.0, de *Jurgen Appelo*, o autor procura explicar as diferenças entre os possíveis ambientes organizacionais, nos quais projetos ocorrem(neste trabalho, chamaremos de cenários) utilizando duas dimensões distintas.

A primeira dimensão diz respeito à estrutura do sistema:

Simples: Fácil de entender.
Complicado: Muito difícil de entender.

A segunda dimensão diz respeito ao comportamento do sistema:

Ordenado: Totalmente previsível.
Complexo: Parcialmente previsível, mas com muitas “surpresas”.
Caótico: Imprevisível.

1.2.1 -Teoria de *Cynefin*

Outra abordagem sobre os ambientes organizacionais é o *Cynefin Framework*, de *Dave Snowden*, que descreve uma perspectiva sobre a natureza evolutiva de sistemas complexos, inclui sua incerteza inerente.

O nome serve como um lembrete de que todas as interações humanas são fortemente influenciadas e, muitas vezes, determinadas por nossas experiências, tanto através da interferência direta da experiência pessoal, bem como através da experiência coletiva, tais como histórias ou música.

O *Cynefin Framework* tem cinco domínios. Os quatro primeiros são:

Simples: você faz X e você sempre terá Y, e não importa quantas vezes você faz X, você obterá o mesmo resultado Y. Você pode prever com confiança o resultado final da atividade. Nesses casos, a coordenação pode ser usada com grande efeito.

A relação entre causa e efeito é óbvia para todos. A abordagem é: Sentir - Categorizar - Responder e, assim, podemos aplicar as melhores práticas(*Best Practice*).

Por exemplo, em uma loja do McDonald's, só existe uma melhor forma de preparar um hambúrguer(sempre a mesma temperatura e sempre a mesma duração), o procedimento já é predefinido e treinado.

Complicado: existe uma relação entre causa e efeito, porém você tem que investir tempo e energia em trabalhar fora dessa relação e, muitas vezes, há uma série de possíveis respostas. Esse é o reino de especialistas que concentram tempo e energia em trabalhar tais relações de causa e efeito. Nesse caso, a cooperação é eficaz nesse domínio, porque muitas vezes há um objetivo final claro em mente, mas você precisa das forças combinadas de uma gama de pessoas para alcançá-lo.

A relação entre causa e efeito requer uma análise ou alguma outra forma de investigação e / ou a aplicação de conhecimento especializado. A abordagem é Sentir - Analisar - Responder e, nesse caso, podemos aplicar boas práticas (*Good Practice*).

Por exemplo, existem diversas formas de se criar a mistura do concreto em uma obra. Dependendo da temperatura, umidade, ferramenta e outros fatores, se pode utilizar uma técnica em detrimento de outra. Por exemplo, na obra da Usina de Itaipu, na mistura do concreto, foi utilizado escamas de gelo, em vez de água no estado líquido, para evitar micro fissuras. Em uma construção, não é comum se utilizar gelo no processo de concretagem.

Complexo: esse domínio é caracterizado por causas e efeitos que são tão entrelaçados e intrincados que as coisas só fazem sentido em retrospecto. Você ouve as pessoas dizerem: "*Ah, este resultado aconteceu porque(...)*", mas se você voltar um pouco atrás e verificar o que aconteceu, de fato, vai obter um resultado diferente.

Retroceder e voltar a jogar, e ainda ter outro resultado. Esse fenômeno ocorre porque, em situações complexas, tudo é tão interligado que uma pequena mudança em uma parte do sistema pode ter um impacto enorme em outro lugar e vice-versa.

O sistema é imprevisível em detalhes, mais ainda podemos discernir padrões. São nessas situações complexas que a colaboração vem à tona. A colaboração funciona bem para cenários complexos, pois o estilo de trabalhar de forma colaborativa corresponde à natureza das questões que representam situações complexas.

Complexidade é imprevisível, é colaborativa, é adaptável; complexidade é confuso – é difícil trabalhar a questão, e muito menos a resposta – é colaborativa, pois envolve reunir uma diversidade de pessoas e talentos para experimentar, criar e testar possíveis abordagens para a solução de um determinado problema.

Complexidade é imprevisível, isto é, dependendo dos fatores(ex: clima organizacional, relacionamento interpessoal, autoconhecimento da equipe, conhecimento prévio do trabalho que será realizado e outros) não temos a certeza do resultado produzido. Assim, no ambiente complexo, temos um grande esforço para desenvolver o aspecto de confiança entre os membros da equipe para que proporcione maior criatividade e, dessa forma, inovações. A relação entre causa e efeito só pode ser percebida em retrospecto, mas não com antecedência. A abordagem é Probabilidade - Sentir - Responder e, assim, podemos aplicar a prática emergente(emergente).

Por exemplo, ao entrar em um consultório médico e relatar que está com dor no estômago(efeito), além de tratar dos efeitos, o médico irá iniciar uma série de testes até descobrir a causa.

Caótico: este é o lugar no qual é impossível discernir a relação entre causa e efeito. A melhor abordagem nesse domínio é simplesmente agir. Não há nenhuma relação entre causa e efeito no nível de sistemas, a abordagem é Agir - Sentir - Responder e, assim, podemos descobrir novas práticas.

Desordem: é o quinto domínio. É o estado de não saber que tipo de causalidade existe. Em pleno uso, a estrutura do Cynefin tem subdomínios, e o limite entre simples e caótico é visto como algo catastrófico.

Um exemplo bem interessante sobre a mudança do ambiente de simples para caótico é ilustrado no filme, *Um Dia de Fúria(Falling Down, 1993*), no qual o ator principal se irrita com uma regra(normal em sistemas simples) e provoca algo catastrófico.

Observe que existe um grande esforço das empresas em transformarem ações complexas em complicadas e, posteriormente, em simples(o que não é possível para todas as atividades), pois economicamente é mais interessante e facilita a expansão do modelo.

1.3 - Tolerância ao Erro

Através da atividade anterior, fomos capazes de identificar cenários simples, complexos e complicados. Iremos analisar agora a tolerância ao erro(ou melhor, à possibilidade de aprender com o erro) e o tempo de resposta do sistema.

Nos cenários caóticos e complexos, o erro é melhor entendido do que nos cenários simples e complicados, já que a relação causa-efeito não existe ou é desconhecida. Essa aceitação é conhecida com "*Fail Fast*" ou "*Learning Fast*".

Nos cenários simples e caóticos, o tempo de resposta precisa ser mais rápido do que nos cenários complexos e complicados, pois a demora pode provocar perdas significativas. Por exemplo, caso você entre em uma lanchonete do tipo "*fast food*"(cenário simples) e o atendimento demore por volta de 20 minutos, você provavelmente irá procurar outro estabelecimento para fazer sua refeição ou, em um prédio incendiando(cenário caótico), você não poderá demorar muito para agir, concorda?

Cenários simples e complicados são mais comuns em projetos de construção como aviões, pontes, prédios, máquinas de café, etc. Pois seguem um fluxo de construção de peça por peça, montagem, testes e estão prontos para uso, seu processo pode ser decomposto em um plano bem detalhado por etapas(ou entregas parciais) até o todo estar pronto. Projetos complexos como softwares, desenhos e redesenhos de processos, campanhas publicitárias e outros são construídos dia a dia, etapa após etapa, assim é muito arriscado definir um plano muito detalhado, já que necessidades e prioridades mudam de acordo com o andamento do projeto(e pela própria dinâmica das organizações).

Diferentes tipos de projetos requerem diferentes métodos. Alguns projetos, especialmente projetos de profissionais do conhecimento, em ambientes de rápida transformação, têm características complexas e necessitam de técnicas emergentes(técnicas de agilidade).

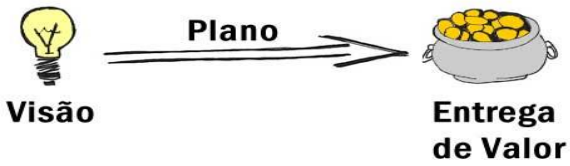
Características de Projeto de Construção(Trabalho Industrial)	**Características de Profissionais do Conhecimento**
O trabalho é visível.	O trabalho é invisível.
O trabalho é estável.	O trabalho é instável.
Ênfase na execução das etapas.	Ênfase na mudança.
Mais estruturação e menos decisão.	Menos estruturação e mais decisão.
Foco nas respostas certas.	Foco nas perguntas certas.
Definição de tarefas.	Entendimento das tarefas.
Comando e controle.	Autonomia.
Padronização.	Inovação contínua.
Foco na qualidade.	Foco na qualidade.
Medição de desempenho padronizada.	Ensinamento e aprendizado contínuo.
Minimização do custo pago ao trabalhador por tarefas.	Trabalhador é visto como um ativo e não como custo.

Quadro de comparação do trabalho industrial com profissionais do conhecimento, adaptado de PMI-ACP ExamPrep, MakiGiffiths, 2012.

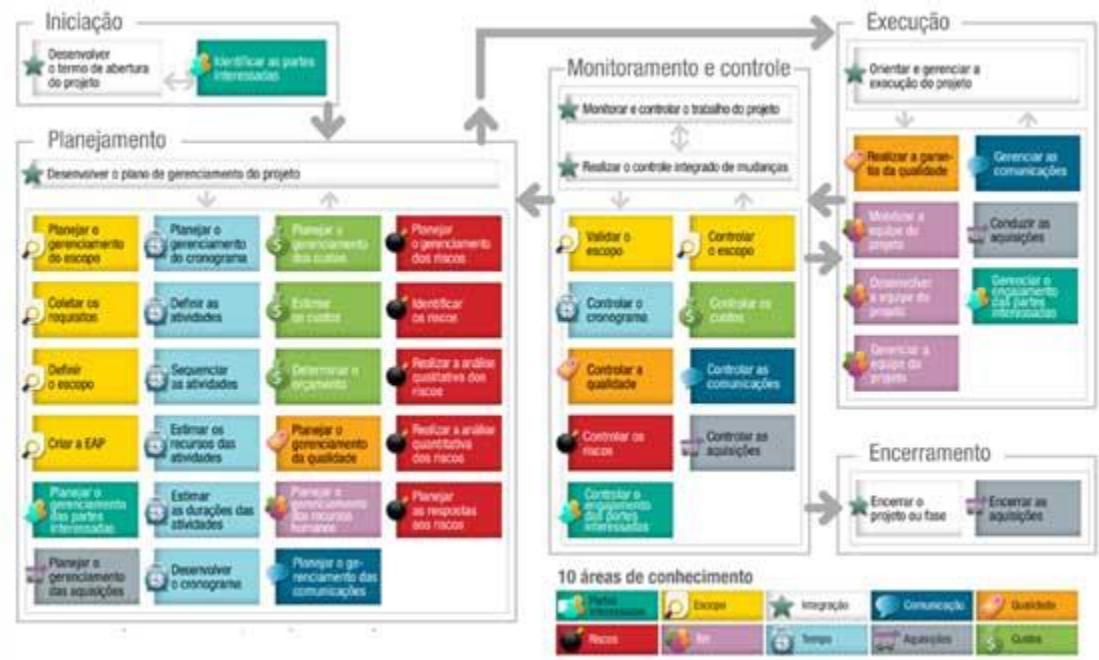
1.4 - Projetos Direcionados à Valor e Projetos Direcionados à Planos

Cenários de projetos de construção, normalmente, seguem uma abordagem orientada a plano, já projeto complexo deve ter uma abordagem orientada a valor, visto que a entrega final tem grande probabilidade de não ser a ideia inicial do projeto.

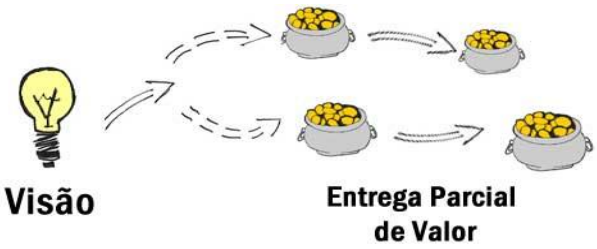
1.4.1 - Projeto Direcionado á Plano:



Em projetos direcionados a plano, como o próprio nome sugere, temos um grande esforço no planejamento, pois um bom plano aumenta as chances de uma boa execução. O guia de boas práticas PMBok® define 47 processos, em 10 áreas de conhecimento, apresentados na figura abaixo, que aumentam as chances de sucesso de um projeto.



1.4.2 - Projeto Direcionado à Valor:



Em projetos orientados a valor, quando se investe muito no planejamento, é gerado um risco muito grande ao projeto, já em cenários de grande incerteza, o planejamento precisará ser baseado em muitas premissas(eventos incertos que definimos como verdade para fins de planejamento) que serão falsas, o que pode gerar tantas mudanças no plano original que o esforço de adaptação não compensaria a energia gasta no desenvolvimento do plano original.

Observe que nenhum projeto é totalmente simples, complicado ou complexo. Pacotes de trabalho de um mesmo projeto podem ter características diferentes, sendo assim, é importante observar o momento do projeto para definir a abordagem adequada de gerenciamento e, nisso, o planejamento em ondas sucessivas ajuda muito.

Outro ponto importante é que técnicas de gerenciamento de projetos complicados, como as boas práticas do PMBoK® (ex: Plano de comunicações, estimativas de custo, declarações de trabalho de aquisição, etc.), podem ser aplicadas em projetos complexos, assim como técnicas de agilidade (ex: Reunião de Retrospectiva, Reunião de Planejamento da Interação, Definição de Visão do Produto, planejamento de release, etc.), de projetos complexos, podem ser aplicadas em cenários complicados e simples, tudo depende da análise do gerente do projeto.

Perguntar qual a melhor forma de gerenciar um projeto sem conhecer o projeto em si, ambiente organizacional e o produto do projeto equivale a perguntar “o que é melhor, o martelo ou o alicate? A resposta

depende se você deseja pendurar um quadro ou fazer uma instalação elétrica". Podemos concluir que NÃO existe uma Bala de Prata em Gerenciamento de Projetos. Assim, o entendimento dos possíveis cenários e o conhecimento das diversas técnicas e ferramentas de gerenciamento são essenciais para o gestor de projetos.

CAPÍTULO 2

2.1 – Introdução:

Diferentes tipos de projetos necessitam de diferentes métodos de gerenciamento. A abordagem ágil é muito utilizada em projetos orientados a valor. Como vimos, projetos orientados a valor geralmente são realizados por profissionais do conhecimento e possuem elevado grau de incerteza, por grande indefinição do escopo e elevado número de mudanças.

A maior parte dos conceitos e princípios ágeis surgiu com foco em projetos de desenvolvimento de software e atualmente são utilizados em diversos tipos de projetos que possuem grandes incertezas, como campanhas publicitárias, novos produtos, planejamento de orçamento e muitas outras áreas. Neste livro iremos manter o foco em projetos de software, porém estes conceitos NÃO são específicos para estes tipos de projetos.

Existem diversos tipos de abordagens ágeis, como veremos neste livro e estas abordagens ágeis buscam estar alinhadas com diversos princípios definidos no documento "*Manifesto for Agile Software Development*", criado em Fevereiro de 2001, por diversos especialistas em projetos de software e disponível na URL http://agilemanisfesto.org

Manifesto para Desenvolvimento Ágil de Software

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas.
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo **valor nos itens à direita, valorizamos mais os itens à esquerda**.

2.2 - Indivíduos e interações mais que processos e ferramentas

Observe que o primeiro valor do manifesto deixa claro uma importante mensagem, processo e as ferramentas provavelmente serão necessário no projeto, porém, devemos tentar concentrar a atenção da equipe sobre os indivíduos e interações envolvidos no projeto. Lembre-se que projetos são realizados por pessoas, e não por ferramentas, assim como os problemas são resolvidos por pessoas, e não processos.

Focando primariamente no desenvolvimento dos indivíduos envolvidos no projeto e enfatizando as interações produtivas e eficazes, melhoramos as chances de sucesso do projeto.

Lembre-se que isso não é dizer que processos e ferramentas não podem ajudar na conclusão com êxito de um projeto. Processos e ferramentas bem desenhados e adequados são ativos de grande importância.

2.3 - Software em funcionamento mais que documentação abrangente

O segundo valor do manifesto destaca a necessidade entregar o software. Projetos de software são normalmente iniciados com os objetivos de criação de valor para a empresa por meio de um produto de software de alta qualidade, mas muitas vezes entregas em partes intermediárias(incrementos), raramente a documentação é completamente atualizada e reflete a realidade do software, porém é essencial se documentar o que precisa ser documentado em um software, mas sem exagero. Lembre-se sempre que software sem documentação é certamente problemático e dificulta o suporte e a manutenção. Mas uma documentação completa sem software não agrega absolutamente nada a nenhuma organização.

2.4 - Colaboração com o cliente mais que negociação de contratos

O terceiro valor reforça a necessidade de ser flexível e eficiente, ao invés de rígidos e não cooperativos. É semelhante a diferença entre "estar certo" e fazer a coisa certa". Poderíamos construir o produto exatamente como originalmente especificado, mas se o cliente mudar de ideia ou de prioridade, você não concorda que devemos ser flexíveis e trabalhar para a nova meta ? É claro que sim, as mudanças e ajustes deverão ser refletidos em aditivos contratuais ou ajustes, mas não deve ser um impeditivo para a continuidade do desenvolvimento e entrega do software. Atualmente existem diversas novas formas de contratos para acolher projetos com características ágeis, como pagamento de preço fixo por interação(ou Sprint), pagamento por pontos, estórias, ou outras que permitem a flexibilidade necessária para projetos orientados à valor.

2.5 - Responder a mudanças mais que seguir um plano

Em projetos com grande número de incertezas, é quase certo que os planos iniciais serão alterados. Em vez de investir esforços na tentativa de trazer o projeto de volta aos planos originais, nós deveríamos gastar esforço e energia responder às inevitáveis mudanças no projeto.

Observe que este valor não está sugerindo abandonar o planejamento e apenas reagir às mudanças. Nós ainda precisamos planejar, mas temos de reconhecer que os planos iniciais foram criados quando conhecíamos menos sobre o Projeto(no início), e com o desenvolvimento do trabalho, vamos precisar atualizar o plano. Muitos dos métodos ágeis focam em macro planos superficiais(criação de estórias, *Product Release*, casos de uso, etc) , e um planejamento mais específico para iterações(ou *Sprints*).

2.6 - Os 12 Princípios por trás do Manifesto:

(Disponível em http://agilemanifesto.org/iso/ptbr/principles.html)

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
7. Software funcionando é a medida primária de progresso. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
8. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
9. Simplicidade - a arte de maximizar a quantidade de trabalho não realizado é essencial.
10. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis
11. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

2.7 - Declaração de Interdependência

Em 2005, *The Agile Leader ship Network(ALN)*, criou a Declaração de Interdependência para Gerenciamento de Projetos Ágeis e está disponível em http://pmdoi.org/

A Declaração de Interdependência promove abordagens ágeis e adaptáveis para unir as pessoas , projetos e valor. Para alcançar estes resultados :

- Nós aumentamos o retorno sobre o investimento, fazendo fluxo contínuo de valor, o nosso foco.
- Nós entregamos resultados confiáveis por envolver os clientes em interações frequentes e compartilhamos a propriedade.
- Esperamos a incerteza e a gerenciamos através de iterações , antecipações, e adaptações.
- Nós liberamos a criatividade e inovação, reconhecendo que os indivíduos são a melhor fonte de valor, e criando um ambiente onde eles podem fazer a diferença.
- Nós melhoramos o desempenho através de prestação de contas do grupo para resultados e responsabilidade compartilhada para a eficácia do time.
- Nós melhoramos a eficácia e confiabilidade por meio de estratégias específicas, processos e práticas.

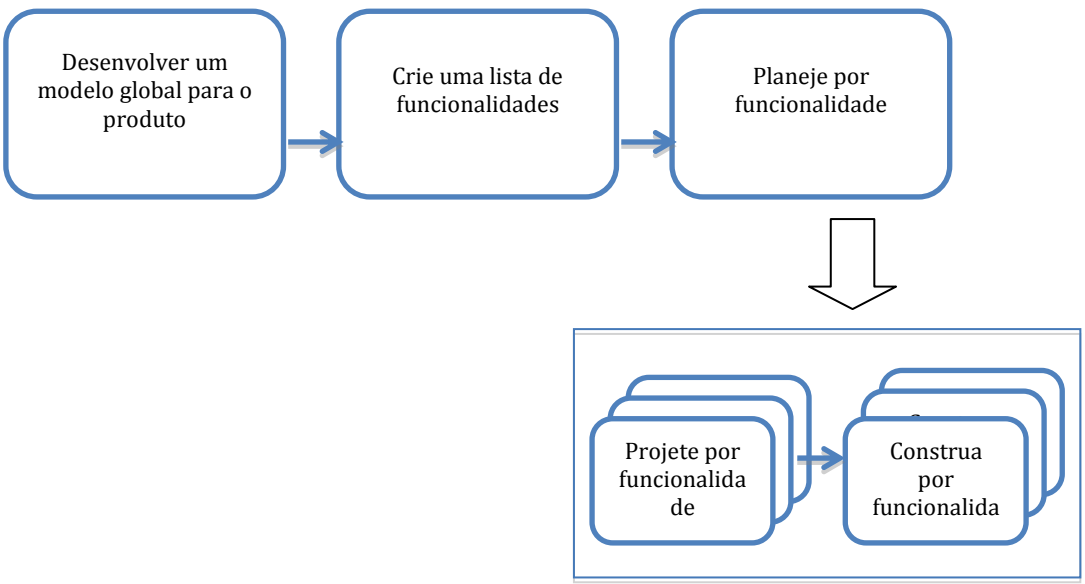
Existem diversas metodologias e frameworks ágeis, os mais comuns e utilizados são o *Scrum, Extreme Programming(XP), Feature Driven Development(FDD), Dynamic Systems Development Method(DSDM), Crystal, Lean, KanBan*, e outros.

O *Scrum* e o *XP* serão mais detalhados nos próximos capítulos, pois são os de maior aceitação e uso no Brasil, porém todos possuem características muito valiosas para o gerenciamento de projetos orientados a valor.

CAPÍTULO 3

3.1 – Feature Driven Development(FDD) - Desenvolvimento Dirigido a Funcionalidades

É uma abordagem simples de entender e poderosa para o desenvolvimento de produtos. Uma equipe de projeto seguindo o método FDD irá primeiro desenvolver um modelo global para o produto, construir lista de recursos e planejar o trabalho. A equipe então se move através da concepção e construção de iterações para desenvolver cada recurso. O FDD busca apresentar resultados frequentes, tangíveis e funcionais.



O FDD recomenda uma série de boas práticas oriundas da Engenharia de Software, como:

3.1.1 - Modelagem de Domínio do Objeto:

As equipes de explorar e explicar o domínio (ou ambiente de negócios) do problema a ser resolvido.

3.1.2 - Desenvolvimento por Funcionalidade:

Esta prática envolve decompor as necessidades em funcionalidades e definir períodos de desenvolvimento de uma ou mais funcionalidades em intervalos de duas semanas ou mais curtos.

3.1.3 - Propriedade Individual(código):

As áreas de código devem ter um único proprietário para garantir consistência, desempenho e integridade conceitual.

(Nota: que é um bem diferente da ideia de Propriedade Código Coletiva do XP que visa difundir o conhecimento para outros membros da equipe) .

3.1.4 - Times Dinâmicos:

Estes são pequenas equipes, formadas dinamicamente de acordo com características de cada projeto. Os times ajudam a mitigar o risco associado à propriedade individual.

3.1.5 - Inspeções:

São revisões que ajudam a garantir boa qualidade e design de código.

3.1.6 - Gerenciamento de Configuração:

Essa prática envolve rotulagem de código, controle de alterações e gerenciamento do código fonte.

3.1.7 - Construções Regulares:

Através de entregas pequenas e constantes, o time incrementa o produto de software com a nova funcionalidade desenvolvida. Esta prática também permite criar facilmente uma versão demo.

3.1.8 - Visibilidade:

Controle e acompanhamento do progresso e dos resultados baseado em funcionalidades desenvolvidas.

Você poderá aprofundar seus conhecimentos sobre o FDD, acessando o web site oficial da metodologia, na URL: http://www.featuredrivendevelopment.com/

3.2 - Dynamic Systems Development Method(DSDM) - Metodologia de Desenvolvimento de Sistemas Dinâmicos

DSDM foi um dos pioneiros dos métodos ágeis. É uma metodologia de desenvolvimento bastante prescritiva, baseada em *Rapid Application Development* (RAD)- Desenvolvimento Rápido de Aplicações, o DSDM enfatiza o envolvimento constante do usuário durante todo o projeto. Cria um amplo ciclo de vida de projeto, abrangendo aspectos de um projeto ágil analisando sempre a viabilidade e necessidade do negócio para a implementação.

3.2.1 - - Ciclo de Vida DSDM

O ciclo de vida do DSDM é tanto iterativo e incremental. Portanto, a solução não pode ser entregue à empresa de uma só vez, mas de uma série de incrementos que incrementam a solução com cada entrega. Desta forma, as necessidades de negócios urgentes podem ser priorizadas e abordadas cedo, enquanto características menos importantes são implementadas e entregues mais tarde.

A natureza iterativa permite que os representantes de negócios vejam e se envolvam no trabalho em construção, analisando e já sugerindo os ajustes e alterações necessárias durante o desenvolvimento de um incremento da solução. Faz parte do ciclo de vida DSDM o ciclo de vida de um projeto de gestão e um ciclo de vida de desenvolvimento de produtos em um único processo. O DSDM pode ser utilizado sozinho como metodologia ágil, porém o uso do DSDM com outros métodos e boas práticas de gerenciamento de projetos ou técnicas de desenvolvimento detalhados podem contribuir para a melhora nos processos e resultados.

CICLO DE VIDA DSDM

Fonte: http://www.dsdm.org/content/6-lifecycle

O DSDM é centrado em 8 princípios(definidos antes da criação do Manifesto Ágil, porém bem alinhados com os valores defendidos no Manifesto Ágil). São eles:

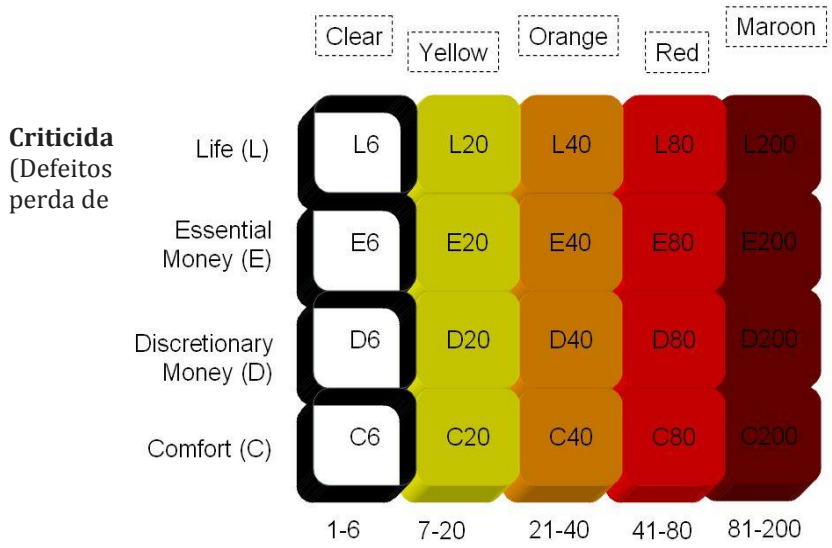
- Foco na necessidade de negócio
- Entregas no prazo
- Colaboração
- NUNCA comprometa a qualidade
- Desenvolvimento incremental com bases sólidas
- Desenvolva iterativamente
- Comunicação contínua e clara
- Demonstre Controle

Você poderá aprofundar seus conhecimentos sobre o DSDM, acessando o web site oficial da metodologia, na URL http://www.dsdm.org/

3.3 - Crystal

Crystal é uma família de metodologias desenvolvidas por Alistair Cockburn, em meados da década de 1990, destinadas para projetos que vão desde aqueles executados por pequenas equipes de desenvolvimento com baixa criticidade e prove abordagens até com grandes equipes que implementam sistemas de alta criticidade.

A família de metodologias Crystal utiliza cores diferentes para indicar o "peso" do projeto e qual a metodologia aplicar.



A Família *Crystal* promove uma série de princípios ágeis, como:

- **Entregas Frequentes**
- **Melhoria Reflexiva** (Verificação constante e busca contínua de promoção de melhoria e implementação de novos métodos)
- **Comunicação Osmótica** (Membros são alocados próximos uns dos outros para melhorar a comunicação, este conceito também é conhecido como *War Room* - Sala de Guerra. Veremos mais sobre comunicação mais adiante)

- **Segurança Pessoal** (O *Crystal* defende um ambiente seguro para que todos possam apresentar suas dúvidas e questionamentos)
- **Foco** (Cada membro do time deve saber o que precisa fazer e ter liberdade suficiente para trabalhar no que é necessário)
- **Fácil acesso ao usuário** (Fácil acesso aos usuários chaves e rápido feedback)
- **Ambiente automatizado** (Testes automatizados, controle de configurações, integração contínua,...)

Você poderá aprofundar seus conhecimentos sobre o *Crystal*, acessando o web site do criador destas metodologias, na URL: http://alistair.cockburn.us/

3.4 - Lean Software Development – Desenvolvimento de Software Enxuto

O *Lean* não é especificamente uma metodologia ágil, porém muitos de seus valores estão presentes em valores ágeis e sua adoção pode agregar muitos valores em diversas outras metodologias e boas práticas. Baseado na metodologia desenvolvida pela Toyota conhecida como *Lean Manufacturing.*

O *Lean* está diretamente ligado a redução do desperdício, para o *Lean*, desperdício é tudo que não é feito para o cliente, no caso dos softwares podemos ter: Espera(tempo de desenvolvimento parado por falta de informações), documentação excessiva, funcionalidades e rotinas não solicitadas pelo cliente.

O *Lean Software Development* identifica **sete** conceitos principais, são eles:

Eliminar o Desperdício: Para maximizar o valor, precisamos minimizar o desperdício.

O Lean classifica o desperdício em 3 categorias:

MURA: Desperdício gerado por antecipar possíveis necessidades do cliente (desenvolver o que não foi solicitado pelo cliente acreditando que um dia poderá ser útil. Este conceito está muito próximo ao de *Gold Plating*, do gerenciamento de projetos). Podemos diminuir a MURA evitando desenvolver o que não foi solicitado e evitando paradas em trabalhos ainda não terminados.

MURI: Desperdício gerado pela falta de planejamento (ou planejamento mal feito). Na MURI também classificamos o desperdício causado pelo excesso de burocracia.

MUDA: Desperdício gerado pela cultura de trabalho, desperdícios do dia a dia, como: espera, super processamento, defeitos, locomoção, inventário, transporte desnecessário e super produção.

"Empoderar o Time": Em vez de da abordagem de micro gestão, devemos respeitar o conhecimento superior dos membros do time e deixar que eles sejam responsáveis pelas decisões técnicas(locais) necessárias, para tornarem-se mais produtivos e bem sucedidos, aumentando as chances de sucesso do projeto.

Entregar Rapidamente: Podemos aumentar o Retorno sobre Investimento (*return on investment*) – ROI com entregas de valor rápidas. Com entregas rápidas e frequentes o cliente já pode iniciar o uso do software (das funcionalidades já desenvolvidas) gerando valor mais rapidamente para

o investimento realizado no desenvolvimento. Como o cliente prioriza o que deve ser feito primeiro, funcionalidades mais importantes (e as de maior risco, assunto que será tratado mais adiante) geralmente são produzidas mais cedo, e consequentemente, entram em uso mais cedo, gerando maior retorno.

Otimizar o todo: Para o *Lean* o sistema não é apenas a soma de suas partes, devemos observar como além de cada parte e sim como todas estão alinhadas aos interesses de negócio da empresa e como otimizá-las da melhor forma possível.

Construir com Qualidade: O *Lean* não testa a qualidade no final do processo. A qualidade do produto final deve ser assegurada com a qualidade de cada etapa/ parte do processo utilizando técnicas como refatoração, integração contínua e muitas outras que veremos mais adiante.

Adiar Decisões: Parece estranho este conceito mas a mensagem aqui é equilibramos o planejamento antecipado com a tomada de decisões e decidirmos o mais tarde possível. Veja que não estamos defendendo a procrastinação, mas sim esperarmos o máximo possível antes de decidir.

Em muitos projetos na fase inicial estamos decidindo baseado em várias premissas (lembrando, *Premissa - Eventos incertos que para fins de planejamento consideramos como verdadeiros o que na execução do projeto não necessariamente será verdadeira*), o que sempre trás riscos ao projeto, como estar restrito a uma solução limitada por uma tecnologia disponível até o momento.

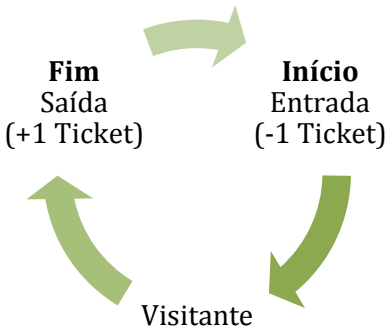
Amplificar Aprendizagem: Este conceito envolve facilitar a comunicação cedo e sempre, promovendo o feedback o mais rápido possível, e aprendizado contínuo com base no que aprendemos sobre projetos, softwares e negócios.

Você poderá aprofundar seus conhecimentos sobre o *Lean*, acessando o web site oficial do *Lean*, na URL http://www.thetoyotasystem.com/

3.5 - KanBan

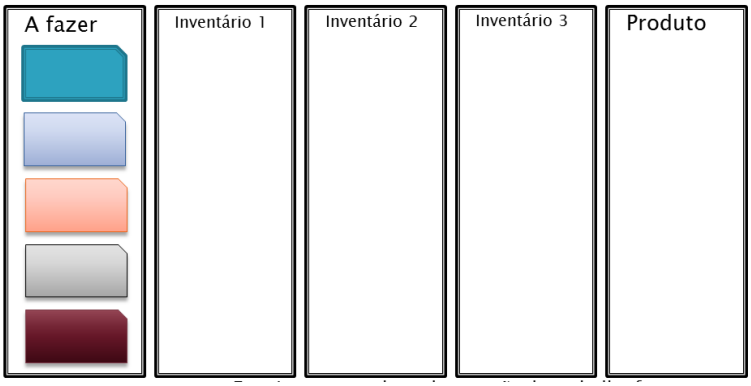
Antes de iniciarmos nossa conversa sobre o KanBan, deixem contar uma história sobre os jardins do palácio imperial Japonês. Em Abril, centenas de visitantes vão aos jardins do palácio para apreciarem as flores de Sakura(flor de cerejeira). Como os jardins, apesar de grandes, são limitados fisicamente, existe um controle de cartões de entrada e saída.

Um visitante para entrar nos jardins precisa de um cartão, este cartão é retirado na entrada e devolvido na saída. Um novo visitante só entra se tiver cartões disponíveis. O que este cartão controla? O número de atividades(pessoas) que os Jardins são capazes de atender. Um controle simples e eficiente garante o atendimento sem prejudicar a qualidade do serviço.

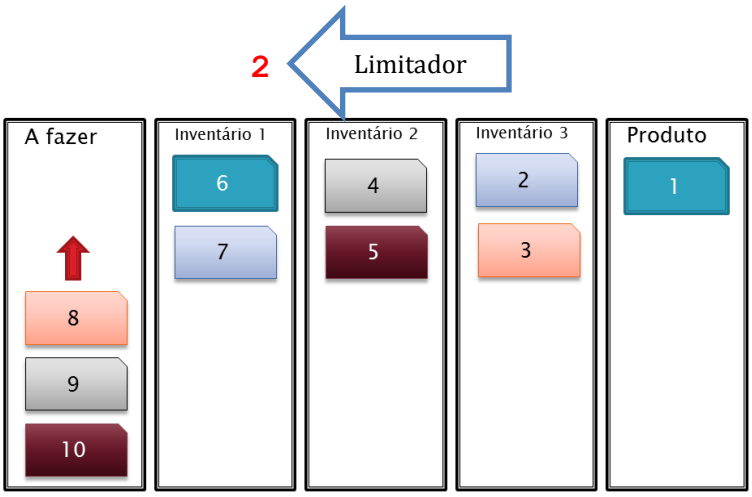


O KanBan possui a mesma filosofia dos cartões, é uma ferramenta visual que auxilia o acompanhamento do fluxo de trabalho e controle do **WIP**(***W****ork* ***i****n* ***P****rogress*," Trabalho em Progresso").

O Quadro Kanban permite visualizar o trabalho que está em andamento e limitar o WIP. Tradicionalmente (mas NÃO é uma REGRA), o quadro Kanban é dividido em quadros ou status como DO(À Fazer) – DOING(Fazendo) – DONE(Feito), as tarefas que precisão ser realizadas são listadas(ou "coladas") na parte **À Fazer**, quando elas começam a ser feitas, elas mudam para o status **Fazendo** e quando estão(totalmente) prontas, vão para o status **Feito**, bem os status podem ser completamente diferentes, por exemplo, no caso de desenvolvimento de software poderíamos ter **Modelado, Em Desenvolvimento, Desenvolvido, Em implantação, Pronto**. Ou qualquer outro estado que faça sentido para o trabalho e para a equipe.

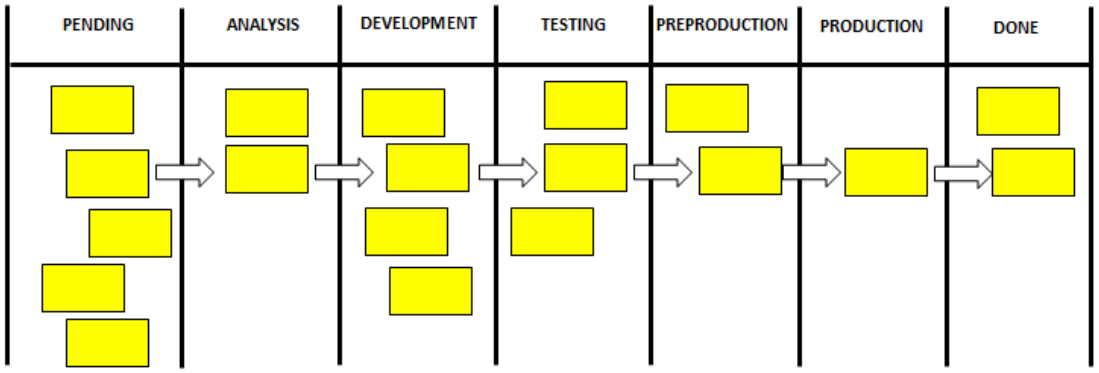


Exemplo que quadro KanBan com tarefas não iniciadas



Exemplo de quadro KanBan com tarefas em andamento. Observe que cada inventário pode determinar um limitador de tarefas que podem ser feitas paralelamente, no exemplo o número máximo de tarefas que podem

estar no “inventário 1” , são duas paralelamente. Este limitador, geralmente está associado a capacidade de atendimento da equipe.



Exemplo que quadro KanBan no desenvolvimento de Software.

Pois bem, como um sistema de controle visual pode ser tão útil ? O Quadro Kanban permite:

Visualizar o Fluxo de Trabalho: O quadro KanBan é uma excelente ferramenta “*Low-Tech, High Touch*”.Quando inserimos a tarefa(ou funcionalidade) no quadro, tornamos o trabalho tangível, o que é um desafio para gerentes de projetos de softwares e serviços. A visualização nos permite identificar onde estão ocorrendo os gargalos no fluxo de trabalho e assim melhorar os processos redesenhando-os ou redimensionando a equipe em cada tarefa.

Limitar o WIP (Trabalho em Progresso): O Trabalho até estar pronto para o uso é custo, por exemplo, uma funcionalidade em desenvolvimento que ainda não é utilizada agrega algum valor para a empresa ? Um serviço que está sendo desenhado, agrega ? **NÃO !** Só temos valor quando a funcionalidade ou serviço podem ser utilizados pela empresa. Uma das regras mais importantes para qualquer metodologia ou framework ágil é “**ENTREGAR MAIS QUE INICIAR**“.

No Quadro KanBan pode-se definir limites para não iniciar novas atividades até que as que estão no status “**Fazendo**” migrem para “**Feito**”.

Tornar Regras e Processos Explícitos: Com o quadro é mais fácil discutirmos os trabalhos que estão em andamento, que precisarão ser feitos e como um trabalho em andamento poderá impactar em outro.

Colaboração: A percepção visual do trabalho facilita a o debate aberto do time sobre como resolver atividades que estão ”engarrafando” o fluxo, além de dar a percepção do todo(do conjunto de atividades feitas, fazendo e à fazer).

Você poderá aprofundar seus conhecimentos sobre o KanBan, acessando o web site oficial do KanBan, na URL http://www.thetoyotasystem.com/

CAPÍTULO 4

4.1 - eXtremingProgramming

Os Valores em XP são conceitos não tangíveis que se acredita fazer uma grande diferença na qualidade final do produto e na motivação dos times. Eles são:

Valores:

Comunicação

O valor da comunicação é visto dentro do time, entre seus membros, e entre o time e o cliente. Ambos têm igual importância.

A comunicação deve ser:

- DIRETA
- EFICAZ
- ESCLARECEDORA

Feedback

É um valor que engloba as relações interpessoais, mas também se refere ao feedback que o próprio código do projeto devolve aos membros do time.

Para que o feedback do código funcione bem, são necessários testes automatizados de unidade e um servidor de integração contínua para que os testes mais longos sejam rodados com freqüência e, se quebrarem, sinalizem uma inconsistência. Com o feedback o cliente pode:

- Identificar erros rapidamente
- Definir prioridades

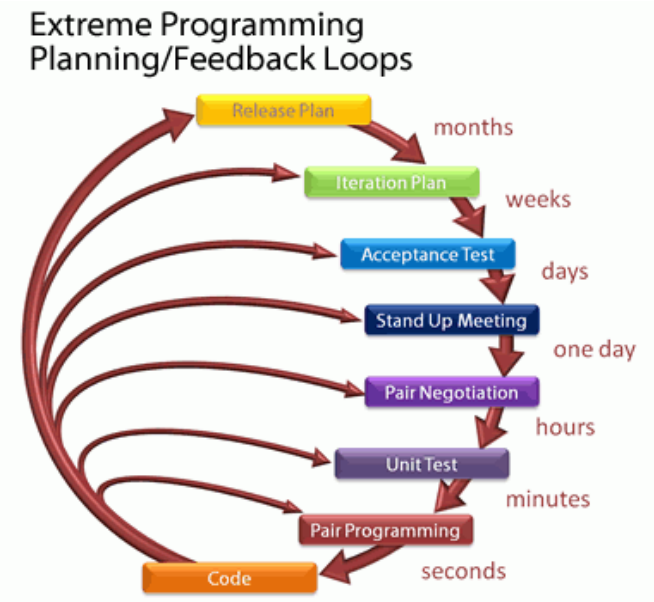


Imagem de ciclos de Feedback do XP

Coragem

O XP prega que os desenvolvedores precisam ter coragem para refatorar o código em prol de melhorias em clareza e design – e nada melhor para dar coragem do que testes automatizados.

Coragem é também apagar o código, mesmo funcionalidades inteiras, não importa o trabalho que tenha sido empregado para desenvolvê-la.

Coragem para não tentar prever o futuro, mas sim focar no que é realmente necessário no momento. XP associa a essa ideia a sigla YAGNI(*you ain't gonna need it*- você não vai precisar disso)

Simplicidade

Considere que, na média, o tempo de construção de um software é cerca de 30% do tempo investido nele. Os outros 70% são dedicados a manutenção do sistema. Neste tipo de cenário, a simplicidade é essencial para tornar esse período maior muito agradável.

O desenvolvedor deve:

- Implementar apenas o básico
- Não antecipar funcionalidades (Não gerar *Gold Plating*)

Propriedade Coletiva do Código

É comum que desenvolvedores trabalhem em partes independentes do código. A consequência desta abordagem é que cada desenvolvedor se sente responsável apenas por sua parte.

O ideal é o sentimento de time onde todos são responsáveis pelo código. Assim, um desenvolvedor é livre para interferir em qualquer parte do código sem irritar o “dono” do código.

Programação Pareada(*Pair Programming*)

A programação em par é uma forma eficaz de reduzir a incidência de bugs em um sistema. Quem trabalha continuamente com programação em par se habitua a corrigir e ter seu trabalho corrigido dezenas de vezes ao dia.

A incidência de erros identificados pelo colega costuma ser tão elevada que surpreende quem não está acostumado ao uso da técnica e as equipes que trabalham em par conseguem reduzir drasticamente a inserção de defeitos em seus códigos.

A programação em par ajuda os desenvolvedores a criarem soluções mais simples, mais rápidas de implementar e mais fáceis de manter. A programação em par também é uma forma de fazer com que o desenvolvedor tenha mais confiança no código que produz. Observe que todas estas característica fazem com que a programação em par acelere o desenvolvimento significativamente, embora à primeira vista pareça o contrário.

Programar em par exige que as pessoas envolvidas sejam receptivas, compreensivas umas com as outras, engajadas e, sobretudo, humildes. É necessário aceitar que somos falíveis para que possamos programar em par. Jerry Weinberg, no livro The Psychologyof Computer Programming apresenta o termo "ego less programming", ou seja, programação sem ego.

Descubra mais sobre "ego less programming" no site:http://blog.codinghorror.com/the-ten-commandments-of-egoless-programming/

Testes Automatizados eRefatoração

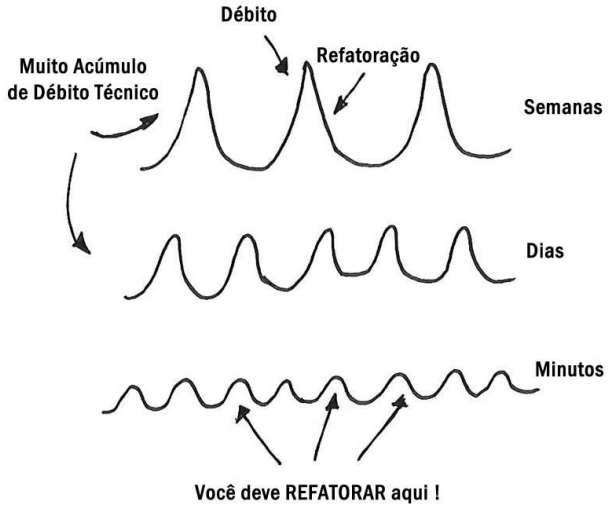
Um dos grandes desafios é trabalhar em códigos antigos.... O XP prega a refatoração constante! Mas, quanto maior o projeto, maior é a quantidade de código não escrita por nós ou antigo, o que aumenta a insegurança de refatorar o código impedindo a evolução do projeto.

Com o aumento do projeto é comum que pequenas partes de código mal escrito se acumulem e, quando menos se esperar, compromete todo o projeto. Este conceito foi nomeado por Joe Yoder como "*Big Ball of Mud*".

A melhor forma de evitar este problema é através de pequenas refatorações constantes.

O autor Martin Fowler define refatoração como:

" ***uma técnica controlada para reestruturar um trecho de código existente, alterando sua estrutura interna sem modificar seu comportamento externo."***



Exemplo de acúmulo de débito técnico ao passar do tempo. Representa-se na figura acima que quando refatoramos constantemente, representado na figura com ondas com a legenda Minutos, nosso débito é perceptivelmente menor do que quando refatoramos com intervalos de dias ou semanas.

Este comportamento também é refletido na economia gerada com o hábito da refatoração constante. Na imagem abaixo, o eixo vertical representa o débito técnico no código, quanto mais cedo os ajustes necessários forem realizados, menor será o custo da mudança para ajustes técnicos.

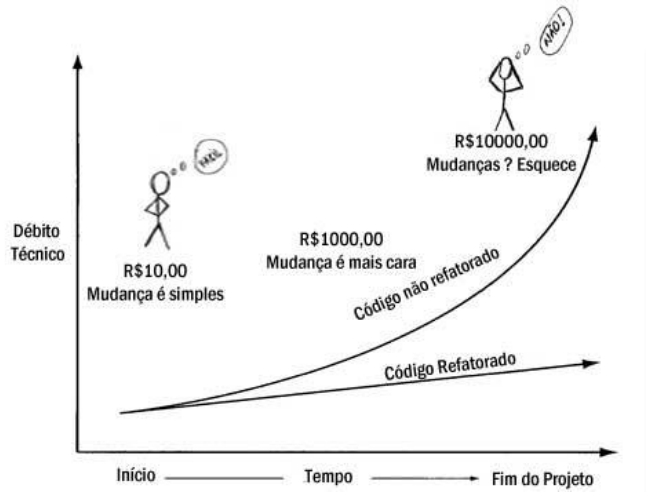


Imagem que representa o custo dos ajustes técnicos no código durante o tempo do projeto.

O XP prega o uso extensivo de testes automatizados que descrevem o comportamento de uma funcionalidade, preferencialmente escritos antes mesmo do código que eles testam, prática que recebe o nome de desenvolvimento dirigido por testes (*Test Driven Development* - TDD).

Os testes automatizados tem 2 funções importantes:

- Permitir refatoração: Podemos refatorar o código com mais segurança. Isto é, podemos alterar o código e verificar automaticamente se o software continua funcionando.

- Documentar: Os testes devem ter nomes que explicam quais funcionalidades eles testam, assim ao executar o código, o desenvolvedor sabe quais funcionalidades foram implementadas e qual o comportamento esperado pelo código.

4.2 - Como realizar o TDD ?

A filosofia do TDD é que os testes guiem o próprio design das classes do sistema.

O processo TDD funciona da seguinte maneira:

1º Faz-se o teste automatizado para o caso mais simples
2º Roda-se o teste (que não deverá passar pois a funcionalidade ainda não foi implementada)
3º Implementa-se através da mudança mais simples possível que faça o teste passar
4º Se o código não estiver o melhor possível:
- Refatora
- Certifique-se que os testes continuem passando

5º Se o código estiver bom
- Volte para o primeiro item com o próximo teste mais simples

Vejamos como os conceitos do TDD podem ser aplicados utilizando um exemplo para desenvolvimento de um simulador do jogo de cartas *Black Jack*

O *Black Jack* é jogado com um ou mais baralhos de 52 cartas, para cujos valores serão designados um total de pontos.

As cartas de 2 a 10 terão o valor dos números. Reis, damas e valetes valem 10 pontos cada e ases poderão ser usados tanto como 1 ou 11.

O objetivo para o jogador será comprar cartas até um total de 21 pontos(sem exceder), vencendo o total das cartas do distribuidor.

Passo 1: Desenho do deque de cartas(*Design a deck of cards*)

```
public class Deck
{
privatereadonlyIlist<Card> cards = new List<Card>( );
public Deck( )
{
cards.Add(Card.Dois_de_Copas);
cards.Add(Card.Tres_de_Copas);
//..restante de copas
cards.Add(Card.Dois_de_Ouros);
cards.Add(Card.Tres_de_Ouros);
//..restante de ouros
cards.Add(Card.Dois_de_Espadas);
cards.Add(Card.Tres_de_Espadas);
//..restante de Espadas
cards.Add(Card.Dois_de_Paus);
cards.Add(Card.Tres_de_Paus);
//..restante de paus
//Coringa
cards.Add(Card.Coringa);
}
```

Defeito identificado ! Não existe **Coringa** no *Black Jack*. Como testar automaticamente em cada novo incremento para que problemas assim não ocorram ?

Devemos criar os Critérios de Teste, para o *Black Jack* vamos testar se existem:

- 52 cartas no Baralho.
- 13 Cartas de Cada Naipe.
- Não pode existir carta Coringa.
- Uma carta de Cada Tipo

```
public class DeckTest
{

public void Verify_Deck_contains_52_cards( )
 {
var deck = new Deck( );
Assert.AreEqual(52,deck.Count( ) );
 }
}
```

Classe que testa se existem 52 cartas no baralho.

Classes para Testes

- 52 cartas no Baralho. --- testado.
- 13 Cartas de Cada Naipe.
- Não pode existir carta Coringa.
- Uma carta de Cada Tipo

```
public class DeckTest2
{

public void Verify_Deck_contains_52_cards( )
 {
var deck = new Deck( );
Assert.AreEqual(52,deck.Count( ) );
 }
public void Verify_Deck_contains_13_cards_for_each_suit( )
 {
var deck = new Deck( );
Assert.AreEqual(13,deck.NumberofCopas( ) );
Assert.AreEqual(13,deck.NumberofEspadas( ) );
Assert.AreEqual(13,deck.NumberofOuros( ) );
Assert.AreEqual(13,deck.NumberofPaus( ) );
 }
}
```

Classe que testa se existem 52 cartas no baralho e se são 13 de cada naipe.

Classes para Testes

- 52 cartas no Baralho --- testado.
- 13 Cartas de Cada Naipe --- testado.
- Não pode existir carta Coringa.
- Uma carta de Cada Tipo

```
public class DeckTest2
{

//..Continuação
public void Verify_Deck_contains_no_joker( )
 {
var deck = new Deck( );
Assert.IsFalse(deck.Contains(Card.Coringa ) );
 }
}
```

Classe que testa se coringa nas cartas geradas

Classes para Testes

- 52 cartas no Baralho --- testado.
- 13 Cartas de Cada Naipe --- testado.
- Não pode existir carta Coringa --- testado.
- Uma carta de Cada Tipo

```
public class DeckTest2
{

//..Continuação
public void Verify_Every_Card_in_the_Deck( )
 {
var deck = new Deck( );
Assert.IsTrue(deck.Contains(Card. Dois_de_Copas ) );
Assert.IsTrue(deck.Contains(Card. Tres_de_Copas ) );

  //...Testar todas as cartas válidas para cada naipe
}
}
```

Classe que testa se existe apenas uma carta de cada tipo

Classes para Testes

- 52 cartas no Baralho --- testado.
- 13 Cartas de Cada Naipe --- testado.
- Não pode existir carta Coringa --- testado.
- Uma carta de Cada Tipo --- testado.

Este pequeno exemplo ilustra como gerar classes em função dos testes necessários. Uma vez implementado qualquer novo incremento no código irá passar por estes testes, já que eles fazem parte do código.

Você poderá aprofundar seus conhecimentos sobre o *eXtreme Programming*, acessando o web site oficial, na URL: http://www.extremeprogramming.org/

CAPÍTULO 5

5.1 - SCRUM

O que é SCRUM ?

Scrum é um método ágil **empírico**, **iterativo** com **entregas incrementais**.

Empírico porque se apoia no empirismo que afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido. O Scrum emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos.

5.1.1 - História

Scrum é baseado em um artigo de 1986 escrito por *HirotakaTakeuchi* e *Ikujiro Nonaka* para a *Harvard Business Review*, o "*The Game Development New Product*". Neste trabalho, os autores utilizaram o esporte do *rugby* como uma metáfora para descrever os benefícios da auto organização de equipes de desenvolvimento de produtos inovadores e de entrega. *Jeff Sutherland*, *Ken Schwaber* e *Mike Beedle* aderiram as ideias deste artigo, incluindo a metáfora, e as aplicou na área de desenvolvimento de software. Eles chamaram seu novo método de Scrum*. *Schwaber* e *Beedle*, escreveu sobre suas experiências em seu livro Desenvolvimento de Software Ágil com Scrum em 2002.

> *"O Scrum ou formação ordenada é uma situação frequente no rugby, geralmente é usado após uma jogada irregular ou em alguma penalização. Os 8 Avançados das duas equipes formam uns contra os outros. O Scrum-half(Médio-Formação) da equipe que não cometeu a infração insere a bola no meio do "túnel" formado pelas duas primeiras linhas de cada equipe com a finalidade de que os jogadores da sua equipe consigam ganhar(talonar) a bola."*

5.2 - Pilares SCRUM

Como o Scrum é baseado no empirismo, os pilares do Scrum baseiam-se em:

Transparência:
Todo o processo deve estar visível a todos os envolvidos. Esta transparência deve ser refletida, no ambiente, nas pessoas e nos processos.

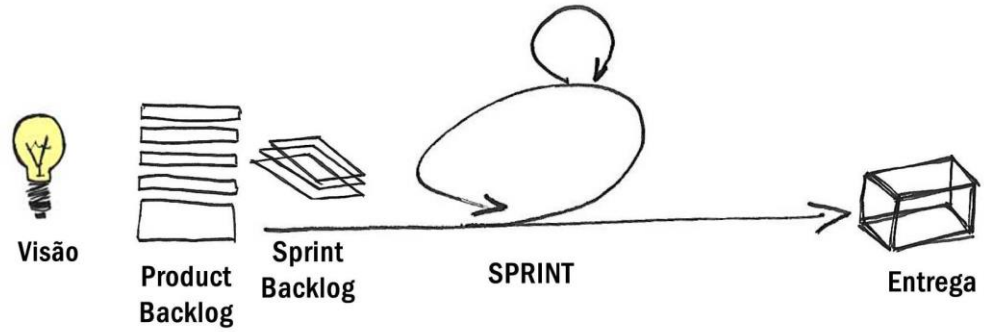
Inspeção:
O processo em si deve ser inspecionado regularmente na busca por anomalias e/ou oportunidades de melhorias.

Adaptação:
Caso a inspeção detecte algum processo que precise ser ajustado ou melhorado, as adaptações deverão ser feitas o mais rápido possível. O quanto antes as mudanças sejam feitas, antes o novo processo proposto é testado e validado.

No Scrum temos oportunidades constantes de verificar os 3 pilares (Transparência – Inspeção e Adaptação), e estas oportunidades são nos eventos da Reunião de Planejamento, Scrum Diário, Reunião de Revisão da Sprint e Reunião de Retrospectiva da Sprint

5.3 - O Framework SCRUM

Scrum é um framework para suportar o desenvolvimento e manutenção de projetos/produtos complexos. Na verdade ele simplesmente fornece uma estrutura para entrega, mas não diz como fazer práticas específicas, deixando isso para a equipe de determinar.



O projeto começa com uma visão clara oferecido pelo negócio, e um conjunto de características do produto em ordem de importância. Esses recursos fazem parte da carteira de produtos, que é mantido pelo cliente ou representante do cliente referido como o *Product Owner*. Uma caixa de tempo comumente referido como uma iteração ou Sprint, é a quantidade de tempo que a equipe tem para concluir as características selecionadas.

Sprints são geralmente de uma a quatro semanas de duração, e esta duração é mantida durante toda a vida do projeto. O *Product Owner* (negociando com o time) seleciona itens do *Product Backlog* que acredita que pode ser concluída no Sprint, e cria um *Sprint Backlog* composto pelos recursos e tarefas, como parte da Reunião de Planejamento (*Planning Meeting*) do Sprint.

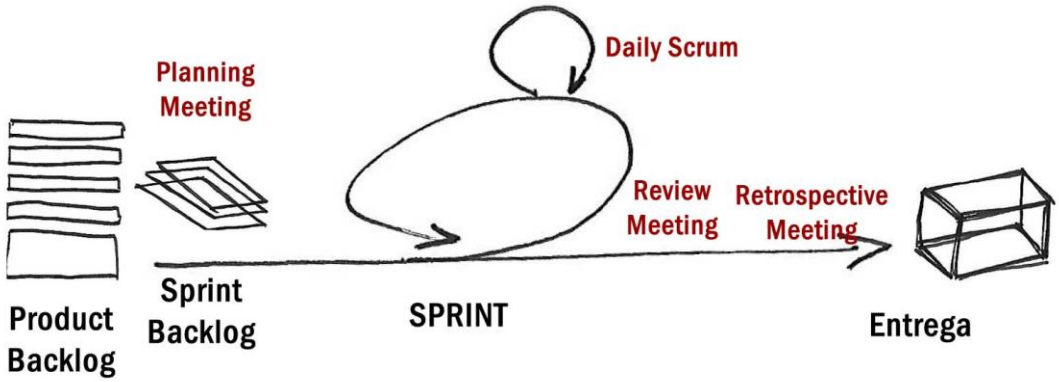
O tamanho da Sprint é influenciado (ou pode ser alterado) dependendo do escopo, tamanho do time, disponibilidade do cliente, conhecimento do time sobre agilidade, conhecimento do time sobre a tecnologia, mudanças na formação do time. Porém sempre que um novo tamanho é definido, as métricas de produtividade deverão ser

Uma vez que o time se comprometeu com um *Sprint Backlog*, o trabalho começa tarefa. Durante este tempo no *Sprint*, a equipe está protegida de interrupções e permitiu a concentrar-se em atender o objetivo do

Sprint. Nenhuma alteração para o *Sprint Backlog* são permitidos, no entanto, o *Product Backlog* pode ser alterado, em preparação para o próximo *Sprint*.

A regra do SCRUM não permite que a Sprint Backlog seja alterada, porém se o item que está em desenvolvimento perde o objetivo de negócio, isto é, não agrega mais valor, não faz sentido continuar a ser desenvolvido, e deve ser interrompido. Geralmente o tempo restante da Sprint pode ser dedicado a refatoração de códigos existentes ou pode se iniciar uma nova Sprint, mas são casos de exceção.

Durante o *Sprint*, a equipa de verifica no diário com o outro sob a forma de uma reunião, geralmente de 10 a 15 minutos conhecido como *Daily Scrum* (Scrum Diário).



No final do *Sprint*, o time (e o cliente, quando possível) reúnem-se para a reunião conhecida como Revisão (*Review Meeting*) . Eles também possuem uma Reunião de Retrospectiva (*Retrospective Meeting*) com o objetivo de aprender como melhorar. Esta reunião é fundamental, pois seu foco é sobre os três pilares do *Scrum*: transparência, inspeção e adaptação.

5.4 - Artefatos, Eventos e Papéis

Artefatos

O Scrum prevê alguns artefatos que nos permite ter uma visão sobre o andamento do projeto e da Sprint. Estes artefatos são conhecidos como Backlog.

5.4.1 - Backlog de Produto(Product Backlog)

É uma lista ordenada criada pelo Time.

O formato da lista pode variar de acordo com cada time, pode ser formadas por casos de uso, Funcionalidades (do FDD), porém os mais comuns são *user stories*, que veremos mais adiante quando tratarmos de *Backlog* e priorizações.

Durante a evolução do projeto podem ocorrer alterações nesta lista, como a inclusão de novos itens, mudanças na prioridade de desenvolvimento e entrega, alguns itens podem ser ajustados de acordo com importância para o negócio ou até mesmo eliminados. Mas o **ÚNICO** que pode inserir, alterar ou remover é o Dono do Produto (*Product Owner*).

5.4.2 - Backlog da Sprint (*Sprint Backlog*)

É um conjunto de itens selecionados para serem implementados (e entregues como incremento ao produto) durante a *Sprint*.

O objetivo do *Backlog* da Sprint é tornar o trabalho visível e tangível para que o Time atinja a **META** da *Sprint*.

Meta: A meta da Sprint são os itens(casos de uso, funcionalidades ou estórias do usuário) que precisam ser adicionadas ao produto(incremento gerado pela Sprint) com sucesso. Geralmente na promoçãodos itens do backlog de produto para o backlog da Sprint, definimos itens a mais do que a meta, pois como as estimativas possuem certo grau de imprecisão, é possível que a meta seja atingida antes do término da Sprint, assim o time ainda tem itens para serem implementados.

***Definição de Pronto**: Esta definição precisa ser acordada e respeitada por todos do time, geralmente a definição de pronto significa dizer que o item já está codificado (de acordo com patterns definidas), testado (teste unitário, integração,...), documentado (manual atualizado), e implemetável. Em quanto um item não atingiu todas as características determinadas na definição de pronto, ele é considerado WiP (Work In Progress – Trabalho em Progresso).*

5.5 - Os papéis e responsabilidades no Scrum:

A organização dos recursos humanos envolvidos no projeto utilizando o Framework *Scrum* é separada em três papéis: *Product Owner*, *Scrum Master*, *Dev. Team* (Desenvolvedores).

O ***Product Owner*** ou dono do produto, basicamente o representante do cliente dentro do time *Scrum* , ele é responsável por entender o que o cliente precisa e transportar este conhecimento para os desenvolvedores. Algumas de suas responsabilidades são:

- Participar do *Daily Scrum*
- Responder dúvidas dos desenvolvedores sobre o que está sendo desenvolvido ou indicar que poderia respondê-las
- Prover uma meta clara para cada *Sprint*
- Obter *feedback* e expectativas dos clientes e extrair deles as necessidades
- Manter o *Product Backlog* atualizado

*O **Product Owner é uma pessoa** e não um comitê. O Product Owner pode representar o desejo de um comitê no Backlog do Produto, mas aqueles que quiserem uma alteração nas prioridades dos itens de Backlog devem convencer o Product Owner. Para que o Product Owner tenha sucesso, toda a organização deve respeitar as suas decisões. As decisões do Product Owner são visíveis no conteúdo e na priorização do Backlog do Produto. Ninguém tem permissão para falar com a Equipe de Desenvolvimento sobre diferentes configurações de prioridade, e o Time de Desenvolvimento não tem permissão para agir sobre o que outras pessoas disserem.*

As responsabilidades do *Product Owner*, são listadas abaixo para cada fase do projeto, mas não se limitam a elas.

Antes do início do trabalho, fase conhecida como *Pre-Game:*

- Identificar as necessidades Estratégicas do Projeto (Patrocinador, Time, Infraestrutura, Áreas Envolvidas, etc)
- Realizar *Kick-Off*
- Descobrir a visão do produto e elaborar artefatos necessários
- Descobrir os requisitos para o *Product Backlog*
- Organizar e priorizar o *Product Backlog*

Durante o Trabalho, fase conhecida como *Game*:

- Participar de todas as Reuniões de Planejamento e de Revisão. Quando necessário visitar a reunião diária e participar da retrospectiva (geralmente, quando convidado pelo Time)
- Estar disponível para o Time (ou garantir que alguém designado por ele esteja)
- Elaborar Plano de *Release* (Plano de versões, veremos mais adiante)
- Manter o *Product Backlog*
- Atualizar o Plano de *Release*

Após a conclusão das entregas do projeto, na finalização do projeto, fase conhecida como *Post-Game:*

- Promover e participar da Retrospectiva do Projeto
- Tornar resultados visíveis para outros (e futuros) projetos *Scrum* na empresa

É muito comum empresas que iniciam nos métodos ágeis definir Product Owner sem poder de decisão; Com baixa disponibilidade; Com interesse em apenas uma parte do projeto; ou com acúmulo de outras funções no time(ou desenvolvedor ou Scrum Master). Todas estas características enfraquecem e podem comprometer a adoção da agilidade e do projeto.

- P.O. sem poder de decisão
- P.O. com baixa disponibilidade
- O “metade” P.O.
- P.O. distante
- P.O. “proxy”
- P.O. “da sua parte”

Segundo o *Scrum Guide*, o ***Scrum Master*** é responsável por garantir que o *Scrum* seja entendido e aplicado. O *Scrum Master* faz isso para garantir que o *Time Scrum* adere à teoria, práticas e regras do *Scrum*. O *Scrum Master* é um servo-líder para o *Time Scrum*.

O *Scrum Master* ajuda aqueles que estão fora do *Time Scrum* a entender quais as suas interações com o *Time Scrum* são úteis e quais não são.

O *Scrum* Master ajuda todos a mudarem estas interações para maximizar o valor criado pelo *Time Scrum*. Algumas de suas responsabilidades são:

- Educar o Time e *stakeholders* sobre o processo
- Assegurar que a equipe faz o *Daily Scrum* no horário certo e de modo produtivo
- Resolver os impedimentos da melhor maneira possível
- Manter o foco das reuniões
- Indicar pontos de melhoria no processo e no ferramental

Segundo o *Scrum Guide*, a **Equipe de Desenvolvimento** (*Dev. Team*) consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto "Pronto" ao final de cada *Sprint*. Somente integrantes da Equipe de Desenvolvimento criam incrementos.

As Equipes de Desenvolvimento são estruturadas e autorizadas pela organização para organizar e gerenciar seu próprio trabalho. A sinergia resultante aperfeiçoa a eficiência e a eficácia da Equipe de Desenvolvimento como um todo.

Os Times Scrum são **auto organizáveis** e **multifuncionais**. Equipes auto organizáveis escolhem qual a melhor forma para completarem seu trabalho, em vez de serem dirigidos por outros de fora da equipe. Equipes multifuncionais possuem todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe.

O que é auto-organização?

A auto organização é o processo onde uma estrutura ou padrão aparece em um sistema sem uma autoridade central ou elemento externo que define um planejamento.

Segundo a físico-química, a auto organização é a característica de que sistemas abertos podem se organizar espontaneamente quando sujeitos a um dado gradiente. O termo auto, nesse caso, reflete o fato de que não há nenhuma instrução do ambiente sobre como a organização deve ocorrer ou como o sistema deve se adequar em resposta ao gradiente. Em outras palavras, o gradiente imposto é completamente neutro em termos de informações e a organização emerge de dentro

do sistema. Processos de auto organização são ubíquos na natureza: de células a órgãos e organismos, de indivíduos a organizações sociais, de casas a bairros, cidades, etc.
(Fonte:http://cienciaecultura.bvs.br/scielo.php?pid=S0009-67252011000100010&script=sci_arttext)

Para a química, a capacidade de auto organização (*self-assembly*) consiste na formação (espontânea) de estruturas complexas a partir de componentes simples. Por exemplo, as espécies químicas denominadas anfifílicas -que incluem os detergentes e alguns lipídios - contêm uma parte hidrofílica (que "gosta" de água), e uma hidrofóbica (que não "gosta" de água). Por este motivo, estas espécies agregam-se na presença de água, formando, por exemplo, micelas ou bicamadas, isto é, se auto organizam.
(Fonte: http://dererummundi.blogspot.com.br/2007/11/auto-organizao-de-sistemas-qumicos.html)

É muito comum auto organização ser chamada de anarquia, mas vamos a alguns esclarecimentos:

Anarquia vem do grego *anarchia* que significa "sem regras". Encontramos outras variações sobre o significado da palavra como: "falta de ordem" ou "negação ou ausência de autoridades ou ordem estabelecida". Estes significados são tratados como sinônimos para caos(sem ordem) ou complexidade(ordem, mas não imposta por uma autoridade).

Na mente da maioria das pessoas, anarquia é igual ao caos. Este equívoco é provavelmente a principal razão pela qual algumas pessoas especialistas não gostam de associar a auto organização com anarquia. Vejam, as galáxias se comportam de uma maneira anárquica, e ainda assim elas não são caóticas, os países(não estamos falando dos governos) as suas relações de comércio são anarquias e também não são necessariamente são caóticas.

A auto organização pode ser considerada uma variação complexa da anarquia. Tanto a química, física e a biologia possuem suas próprias definições de auto-organização, mas nenhuma delas aborda a liderança, gerência ou autoridade. Assim, não faz muito sentido alterarmos os conceitos de auto organização apenas para aplicar no contexto social.

Por exemplo, quando todos as netas e netos de minha mãe(crianças entre 2 e 7 anos) se reúnem na casa dela com seus muitos amiguinhos, correndo e gritando para todos os lados, nós adultos podemos acreditar que é um sistema anárquico quando na verdade eles estão auto organizados. Isto ocorre porque a forma como eles se auto organizam não necessariamente agrada(ou é aprovada) por nós(suas principais partes interessadas).

Este comportamento é observado em empresas/departamentos de software onde os desenvolvedores disputam em horário de trabalho campeonatos de vídeo game e ainda sim entregam seus trabalhos. Isso provoca um choque cultural na empresa que pode atingir até a diretoria executiva com essa nova forma de relacionamento.

É importante observar que a auto organização não faz distinção entre bem e mal, virtudes ou vícios, atitude que agregam valores ou não.

Outra confusão muito comum é entre a auto organização e as Propriedades (Técnicas) Emergentes. Quando uma propriedade não pode ser rastreada à qualquer uma das partes individuais do sistema, Esta propriedade é chamada de uma propriedade emergente. Por exemplo, sua personalidade é uma propriedade emergente do seu cérebro. Ela não pode ser atribuída a neurônios individuais.

As propriedades emergentes possuem as seguintes características:

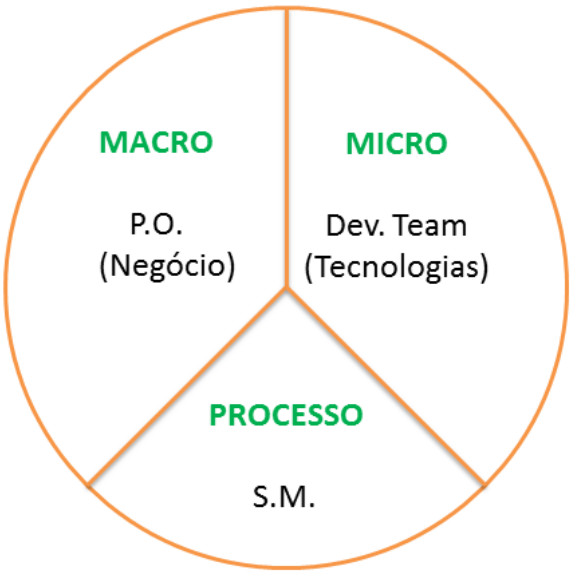
Superveniência é a observação de que a propriedade deixará de existir se você tirar as peças individuais do sistema. Por exemplo, a sua personalidade desaparece se remover seus neurônios.

A propriedade não é um agregado, ou seja, esta propriedade não é simplesmente o resultado da soma das propriedades das partes individuais. Por exemplo, uma única molécula não tem fluidez. Então você não pode simplesmente somar a fluidez de um bilhão de moléculas individuais para determinar a fluidez da água.

Deve haver **causalidade para baixo** (*downward causality*), o que significa que a propriedade emergente deve influenciar o comportamento das peças individuais. Por exemplo, a cultura de um grupo de pessoas influencia o comportamento dos seus membros.

O comportamento emergente em times é o resultado da interação entre os seus membros. Assim, o time é responsável por sua própria cultura, processos, imagem perante a organização e muitas vezes até pelo seu “próprio nome”.

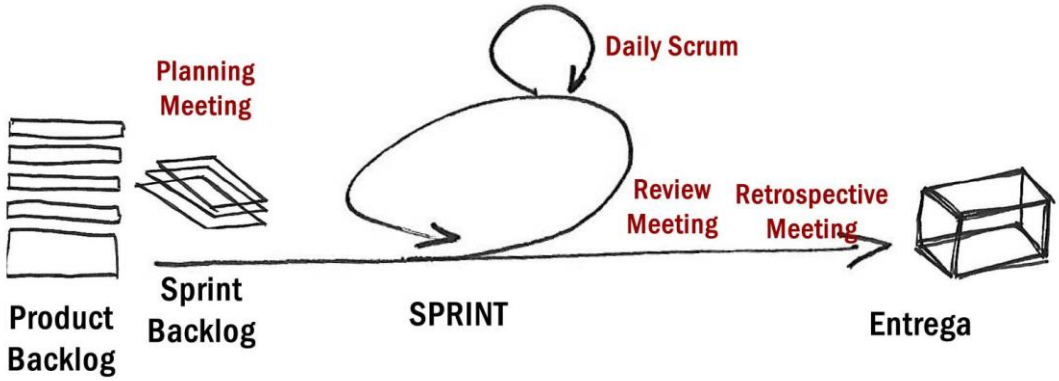
Como regra geral, podemos dividir os papéis e responsabilidades como:



Observe que dentro do Scrum as responsabilidades de Gerenciamento de Projetos estão distribuídas em papeis distintos.

Assuntos relacionados ao macro negócio, priorização, definição do que será feito, orçamento, ou qualquer outro assunto associado ao negócio é de responsabilidade do *Product Owner* (P.O.). Assuntos referentes a técnica de desenvolvimento, arquitetura, e assuntos mais técnicos são de responsabilidade do time de desenvolvimento(Dev. Team) e por fim, assuntos associados aos processos de trabalho são de responsabilidade do *Scrum Master*. Estes personagens formam o *TIME SCRUM* e são, em conjunto, responsáveis pelo gerenciamento do projeto.

5.6 - Eventos Scrum



5.6.1 - Sprint

Segundo o Scrum Guide, a *Sprint* é um *time-box* de um mês ou menos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado. *Sprints* tem durações coerentes em todo o esforço de desenvolvimento. Uma nova *Sprint* inicia imediatamente após a conclusão da *Sprint* anterior.

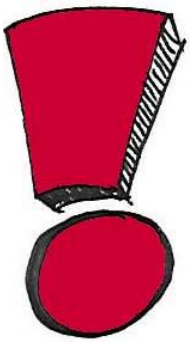
As *Sprints* são compostas por uma reunião de *Planning meeting* (Planejamento da Sprint), *Daily Scrum* (Reuniões Diárias), o trabalho de desenvolvimento, uma *Review Meeting* (Revisão da *Sprint*) e a *Retrospective Meeting* (Restrospectiva da *Sprint*).

5.6.2 - Reunião de Planejamento - Planning Meeting

A reunião de planejamento de Sprint, é realizada no primeiro dia de cada Sprint. O *Scrum Master*, *Product Owner*, e o *Dev. Team* estão todos os presentes. Esta reunião não deve ocupar mais de 5% da duração da *Sprint* (por exemplo, em uma *Sprint* de 2 semanas, não deve durar mais de 4 horas) e é dividida em duas partes e deve chegar responder as seguintes perguntas:

- O que será entregue no Incremento resultante nesta Sprint ?
- Como faremos para entregar o Incremento nesta Sprint ?

Na primeira parte, o *Product Owner* apresenta o conjunto de características que ele gostaria de ver concluída no *Sprint* ("o quê"), os itens do topo do *backlog* do produto. O Time avalia o esforço necessário para a conclusão de cada item e negocia o que é possível entregar na *Sprint*.



A quantidade de itens selecionados para a Sprint é prerrogativa do Time de Desenvolvimento. Ele é o único capaz de avaliar o esforço para cada item já que será ele que irá desenvolvê-los. Em times com baixa maturidade é importante que o Scrum Master relembre da importância de não subestimar ou sobrecarregar a Sprint.

Após selecionar os itens que serão realizados na *Sprint*, o Time com o P.O. definem a Meta da *Sprint*.

Na segunda parte da Reunião de Planejamento, o Time define como irá transformar os itens do *backlog* da *Sprint* em Incrementos do produto. Nesta etapa o time determina as tarefas necessárias para implementar esses recursos ("o como"). Estimativas de trabalho são revistas para ver se a equipe tem o tempo para concluir todas as características solicitadas na *Sprint*. Se assim for, a equipe se compromete com a *Sprint*. Se não, os recursos de menor prioridade voltam para o *Product Backlog*, até que a carga de trabalho para a *Sprint* seja de tamanho adequado para obter o compromisso da equipe

5.6.4 - Scrum Diário - Daily Scrum

O *Daily Scrum* é uma reunião rápida e informal , com duração máxima de 15 minutos onde participam apenas o time *Scrum*. Geralmente cada membro do time responde as seguintes perguntas:

O que fiz desde o último *Daily Scrum*;

O que irei fazer até o próximo e;

Quais impedimentos estão me atrapalhando.

O objetivo desta reunião é melhorar a comunicação na equipe e dar para todos uma visão mais clara do andamento do time na *Sprint*, além de facilitar a identificação e resolução de problemas e impedimentos.

5.6.5 - Reunião de Revisão da Sprint - Review Meeting

No final da *Sprint*, o time convida os interessados (clientes) para uma reunião de revisão da *Sprint*, onde as tarefas que foram concluídas na Sprint são demostradas e o feedback é solicitado.(**APENAS AS QUE ATENDAM A DEFINIÇÃO DE PRONTO**). Esta reunião não deve ocupar mais de 2.5% do tamanho da *Sprint* (por exemplo, em uma Sprint de 2 semanas, não deve durar mais de 2 horas).

Esta reunião é importante para demostrar para demostrar ao cliente as novidades que estarão disponíveis para o uso, quanto para que o time de desenvolvimento receba o *feedback* real dos usuários do sistema e entenda melhor suas necessidades.

Caso o cliente tenha dúvidas, estas deverão ser respondidas e qualquer problema(ou possível problema) encontrado deve ser anotado. Nesta reunião o *Product Owner* aprova ou rejeita os itens implementados. Caso rejeite, o item volta para o *Backlog* do Produto (e pode ou não entrar na próxima Sprint, dependerá da avaliação do *Product Owner*).

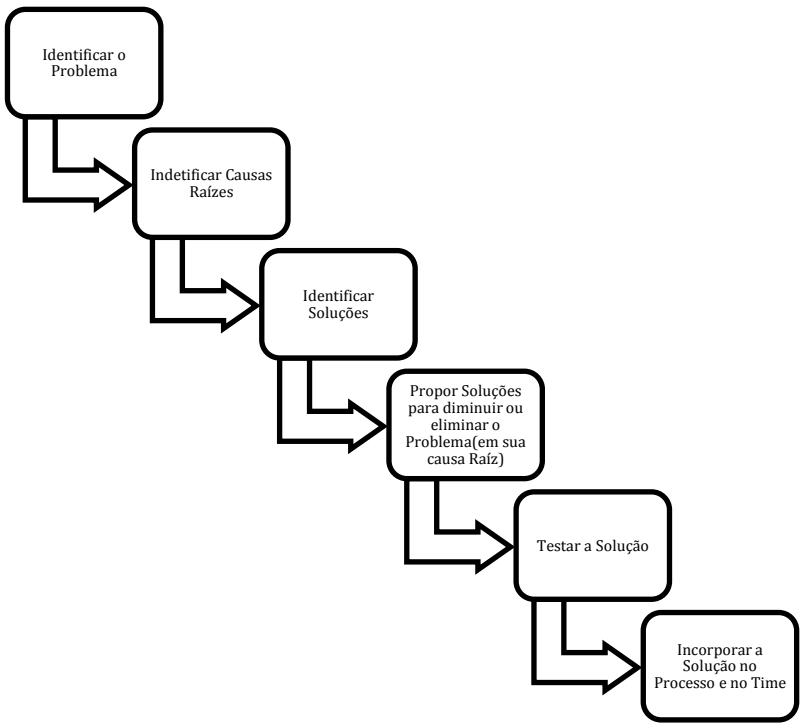
5.6.6 - Reunião de Retrospectiva da Sprint - Retrospective Meeting

A Reunião de Retrospectiva é um recurso de melhoria contínua, uma ferramenta de comunicação e evolução do time. Times imaturos costumam dar menos valor para este evento, porém a prática de Retrospectiva vem sendo utilizada em diversos ambientes de equipes e projetos (mesmo em

ambientes não ágeis), pois permite intensificar as características fortes de cada time e eliminar pontos de fraqueza.

Uma vez que a revisão está concluída, o time(sem as partes interessadas, a não ser que o time de desenvolvimento e o *Scrum Master* decidam convidar...) realizam uma retrospectiva para determinar o que eles fizeram bem que eles querem continuar fazendo, o que foi ruim na *Sprint*, e quais as recomendações de melhoria daqui para frente. Um plano de ação é criado e esses itens são implementadas ao longo da próxima *Sprint*, e revisado para a eficácia na retrospectiva da *Sprint* seguinte. Geralmente a reunião de retrospectiva não deve ocupar mais de 3,75% da Sprint (por exemplo, em uma *Sprint* de 2 semanas, não deve durar mais de 3 horas).

Existem diversas técnicas de Retrospectiva, porém todas elas possuem etapas:



Independente do método/framework ágil que for utilizado existe elementos comuns que quando bem utilizados agregam valor ao projeto.

Nesta seção vamos apresentar algumas dinâmicas que podem ser utilizadas nas reuniões de retrospectiva.

5.6 - Dinâmica Bons e Ruins

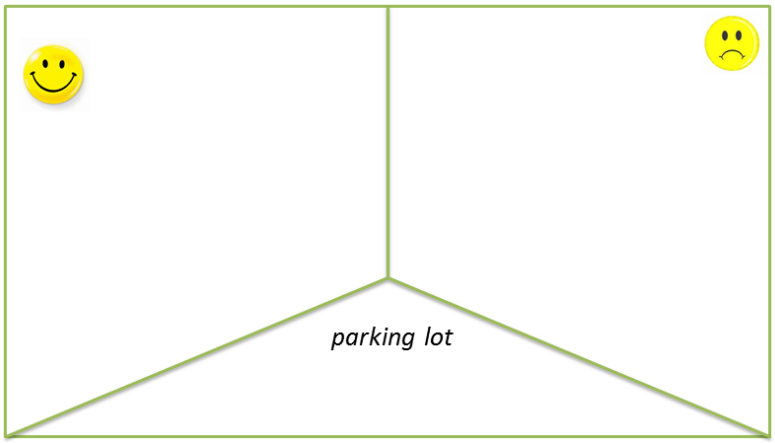

Esta dinâmica é muito interessante em retrospectivas de equipes com alguma experiência em agilidade ou para times que já se conhecem a algum tempo:

- Permite a exposição dos problemas relevantes no momento em que o time se encontra
- Permite que o time apresente, entenda e proponha soluções para os problemas(autogerenciamento)
- Integra o time.
- Gestão do conhecimento do time.

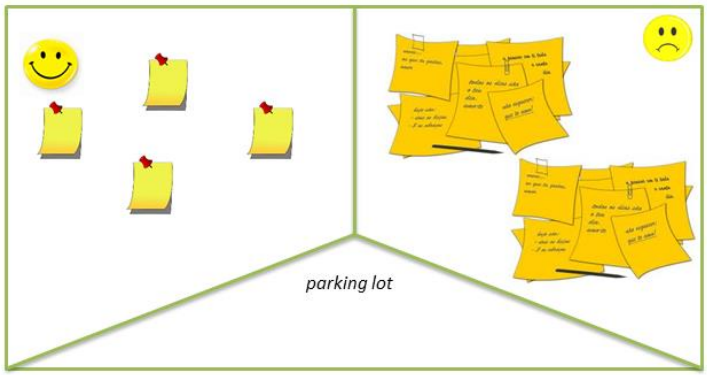
Nunca é tarde para lembrar ao time que a Retrospectiva não é caça as bruxas e sim uma ferramenta de melhoria, logo, não é o momento de acusações e sim de soluções. O time deve ter maturidade e não levar nada para o lado pessoal e o facilitador deve lembrar o time estes pontos sempre que necessário!

1º) Cada membro do time ganha *post it* e canetas(de preferência todos da mesma cor) e escreve os pontos bons e ruins da S*print*

2º) Quando todos finalizarem, colam-se os *post its* em um lugar visível(veja um exemplo de quadro sugerido abaixo)



3º) É feita a leitura dos tópicos ruins, observe que os problemas mais sérios terão mais post its



4º) O time discute os problemas e PROPÕE AÇÕES DE SOLUÇÃO

OBS: Dependendo do número de problemas é interessante concentra esforços para resolver 2 ou 3 mais sérios.

5º) Se faz a leitura dos pontos bons !(Práticas que devem ser mantidas)

5.7 - Considerações:

- Problemas que não geram ações e soluções podem ser adiados para discussão posterior, estes problemas devem ficar na área de *parking lot* do quadro
- **Ações devem ser atitudes concretas que permitam a execução sem dupla interpretação**. Por exemplo: ”Quem desenvolveu deve realizar testes unitários!”
- Sugiro que **ações sejam endereçadas**, por exemplo, fulano de tal é o responsável por lembrar-se da importância da ação XYZ, durante o próximo *sprint*
- Ao final da retrospectiva, deve-se jogar no lixo os *post its* e deixar fixadas as ações combinadas visíveis para todos, sugiro que elas fiquem no quadro *kanban* para que todos se lembrem !

5.8 - Dinâmica Mercado de Habilidades (*Market ofSkills*)

Esta dinâmica é muito interessante em retrospectivas de equipes que estão no processo de transformação para times! Alguns benefícios são:

- Acelera o autoconhecimento como time.
- Permite que o time inicie seu auto gerenciamento, onde cada um assume como e quando ajudar.
- Integra o time.
- Gestão do conhecimento do time.

1º) Liste as habilidades que você tem e julgue ser útil para o time e para o produto

2º) O que você tem vontade de aprender ? Como isso será útil para o time ? Como isso será útil para o produto ?

Nesta etapa, os membros do time identificam interesses comuns, o que facilita o processo de comunicação e aproximação.

3º) Informações adicionais. Apresente o que você tiver vontade para o time(extra time, extra produto, ...) que você sinta vontade de compartilhar

4º) Venda-se para o time !

Durante a venda, todos podem acrescentar habilidades esquecidas pelo vendedor(passo 1), formas de ajudar nas vontades.

5º) Defina ações !

Após a primeira aplicação desta dinâmica, o time deve repeti-lá (mais adiante no projeto) e verificar a evolução do time. Por exemplo, na primeira dinâmica, o time tinha identificado o seguinte quadro de conhecimento:

Conhecimento/Habilidade	Número de Membros com Domínio
Java	2
SQL Server	1
CSS	1
Teste Automatizado	1

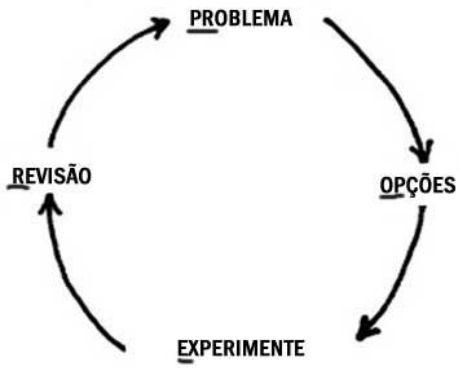
Após ações como programação pareada, a tendência é que o time compartilhe conhecimento e melhore suas habilidades. A próxima dinâmica poderia gerar um resultado melhor, como o ilustrado abaixo:

Conhecimento/Habilidade	Número de Membros com Domínio
Java	3
SQL Server	2
CSS	2
Teste Automatizado	2

5.8 - Técnica PrOpER

A abordagem mais simples é escolher uma coisa e pular dentro. Se não é óbvio qual é o problema para trabalhar em primeiro lugar, então você pode ter uma abordagem ágil. Faça uma tempestade de ideias, gere uma lista de áreas problemáticas para trabalhar no que poderia melhorar a vida da sua equipe no projeto. Então priorize esta lista com base em sua missão de treinamento - agora você tem um ponto de partida !

Você pode aplicar o ciclo PrOpER adequando a cada episódio de *coaching* ou na retrospectiva.



5.8.1 - Problema:

Escolha o problema para trabalhar. Veja como a equipe trabalha. O que precisa ser melhorado?

5.8.2 - Opções:

Considere as suas opções. O que você poderia tentar que poderão influenciar a situação para melhor? Relacione pelo menos três opções.

5.8.3 - Experimente:

Escolha uma opção e a execute.

5.8.4 - Revisão:

Analise o resultado. Você melhorou as coisas? Mesmo se as coisas não melhoraram, você já aprendeu alguma coisa ?

Vamos praticar o PrOpER através de um exemplo.

5.8.5 - EXEMPLO:

Problema*:*

Jack chegou tarde para a reunião de Daily Scrum hoje. Aconteceu na semana passada também. Você está preocupado, porque ele está trabalhando na construção de um ambiente de teste novo. Ele está perdendo informações importantes sobre os problemas da equipe com o ambiente de teste atual.

Opções: *Aqui estão algumas opções que você pode considerar:*

- *Pegue o touro pelos chifres: Quando Jack chega, peça um tempo para informá-lo sobre o que ele perdeu até o momento no Daily Scrum. Enquanto você está revendo os pontos, fale com ele sobre a importância de participar do Daily Scrum todo dia.*

- *Educar a equipe: Executar uma sessão de treinamento para toda a equipe para aprender a melhorar o Daily Scrum, o que pode ajudar o Jack a entender porque é importante para todos na equipe a participação na reunião.*

- *Deixe-os segurando o bebê: Você precisa de alguém para cobrir a você: pergunte se Jack pode ajudá-lo, executando o amanhã o Daily Scrum.*

- *Espere e veja: Não fazer nada, e esperar para ver se a equipe faça que Jack perceba que o seu atraso é um problema por si só.*

Experimente:

Você escolhe a opção para primeiro falar com Jack sobre isso. Inicie a conversa, mencionando que você percebeu que ele perdeu o Daily Scrum algumas vezes. Ele parece genuinamente surpreso que isso era importante, já que a partir de sua perspectiva, seu trabalho não está diretamente ligado a nenhuma em estórias de qualquer cliente, por isso certamente que ele não precisaria estar lá. Explique o motivo que sua preocupação é que ele está perdendo informações de seus companheiros de equipe que precisarão ser consideradas na construção do ambiente de teste novo. Também explique que o Daily Scrum é para a equipe, e não para o cliente. Sugerira que ele convoque uma reunião com o testador(e time) para verificar as questões que provavelmente não foram atendidas. Ele acena que concorda ele chegará no horário da próxima Daily Scrum.

Revisão: *Revisão do resultado. No dia seguinte, Jack chegou a tempo? A sua conversa fez a diferença? Se ainda há problemas, então que outras opções você pode tentar?*

Ao tentar chegar com as opções, estão aqui algumas ideias a serem considerados:

- *Traze à tona o problema: Torne o problema visível para a equipe*
- *Socializar o problema: Converse com a equipe sobre o problema*
- *Espere e Veja: Deixe este problema: Se piorar, a equipe provavelmente notará*

◦ *Tire o corpo fora: Transfira o problema para outra pessoa dentro ou fora da equipe*

◦ *Análise de causa raiz: Procure a causa raiz do problema*

◦ *Educar a equipe: Forneça a equipe mais informações para que eles vejam uma solução*

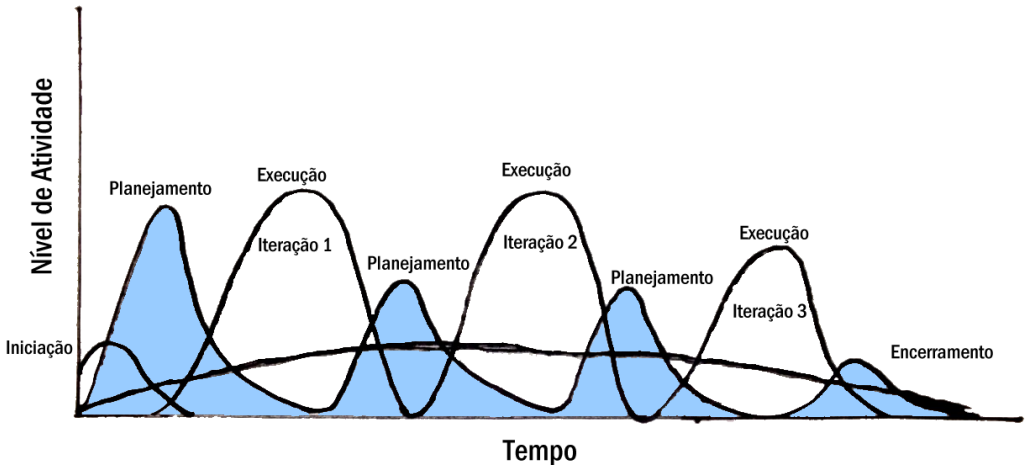
◦ *Coloque-os no comando: entregue a responsabilidade para a equipe ou a um membro da equipe.*

CAPÍTULO 6

6.1 - Planejamento Orientado à Valor

Em projetos ágeis, devido a sua própria característica de incerteza, o planejamento de versões e a priorização do que será desenvolvido (*Backlog*) é estratégica para melhorar o ROI (*Return Over Investiment*).

As técnicas de planejamento de ágil tem forte afinidade com o planejamento em ondas dos métodos de gerenciamento de projetos orientados à planos, porém com algumas diferenças conforme apresentadas na figura abaixo. Veja que não é interessante antecipar todo o planejamento já que como temos muitas incertezas, teremos grandes probabilidades de mudanças caso a antecipação seja feita. Ao invés disso, fazemos um planejamento de alto nível para identificar e planejar os releases, e posteriormente planejamos mais detalhadamente cada iteração.



Como vimos no fluxo do framework Scrum, tudo se inicia na definição da visão do projeto.

6.2 - Visão em um projeto ágil

Visão é uma clara imagem que gera uma atração emocional entre pessoas e produto (quando se fala a visão quem escuta deve ser capaz de imaginar como será o produto). A visão deve responder as seguintes perguntas:

- Quem irá comprar este produto? Quem é o cliente alvo? Quem irá usar o produto? Quem são os usuários alvo?

- Quais problemas do cliente(ou usuários) o produto pretende resolver? Qual valor o produto adicionará?

- Quais atributos o produto deve possuir para resolver estes problemas e quais garantirão o sucesso do produto?

- Como o produto pode ser comparado a produtos ou alternativas existentes ? Quais são os pontos diferenciais deste produto?

- Qual o preço alvo do produto? Como a empresa pretende ganhar dinheiro com este produto? Quais serão as fontes de faturamento e qual o seu modelo de negócio?(quando aplicável)

Lembre-se que uma boa visão de produto permanece relativamente constante ao passo que o caminho para implementação da visão é frequentemente adaptado.

Uma das técnicas bem interessantes de se iniciar a identificação e criação da Visão é conhecida como *Elevator Statement*, esta técnica prevê que você deve apresentar o produto que será criado em poucos segundos, como em uma viagem no elevador. Como sugestão, existe um modelo proposto apresentado abaixo.

Para <***cliente/público-alvo***> que <***necessidade do cliente/público-alvo ou oportunidade***>, o <***nome do produto***> é um <***categoria do produto***> que <***principal benefício ou razão para comprar o produto***>. Diferentemente do <***principal competidor ou alternativa***> nosso produto <***principal diferenciação***> .

Se após apresentar seu *ElevatorStatement,* não fique claro o que é seu produto, qual o objetivo e para quem se destina, ele NÃO está bom!

Você pode consultar mais sobre esta técnica na URL: http://www.flyingsolo.com.au/marketing/business-marketing/preparing-your-elevator-statement

O próximo passo é materializar seu produto, mesmo sendo um software!

A técnica de *Product Vision Box* propõe a criação da caixa do seu software com informações como:

- Nome do Produto
- Gráficos
- Três ou quatro pontos chave(benefícios) para “vender” o produto
- Principais funcionalidades no verso
- Principais requisitos operacionais

Você pode fazer esta dinâmica com o time envolvido no projeto usando caixa de sapato, cola e canetas coloridas lembrem-se que na agilidade temos diversas técnicas *Low-Tech, High Touch*.

Observe que neste passo, conseguimos identificar os principais diferenciais e as funcionalidades essenciais do futuro software, além de alinhar as expectativas dos os participantes.

Você pode consultar mais sobre esta técnica na URL: http://www.agile-ux.com/2011/03/04/a-day-in-life-of-an-agileux-practitioner-vision/

Após a criação da caixa do produto, podemos iniciar o *Product Road Map*, que apresenta as funcionalidades e características do produto(e em que versão a funcionalidade será inserida)**.** Para esta etapa podemos usar a técnica *Rememberthe Future,* esta técnica tem como objetivo *d*escobrir o entendimento do sucesso do cliente e iniciar a visualização do *Road Map* do produto/projeto.

Nesta técnica ao invés de olhar o passo a passo, você deve se posicionar no momento final desejado e “relembrar” o que foi feito para chegarmos neste ponto.

A técnica "Remember the Future" não é um jogo de adivinhação do futuro e sim uma ferramenta que nos auxilia no entendimento do sucesso do projeto/produto e como podemos atingir o objetivo final. Logo, NÃO é plano, NÃO é cronograma, NÃO é determinístico!

Você pode consultar mais sobre esta técnica na URL: http://innovationgames.com/remember-the-future/

Após a aplicação das etapas anteriores, conseguimos ter um melhor entendimento da visão do projeto e criar o *Project Data Sheet,* a planilha de dados do projeto.

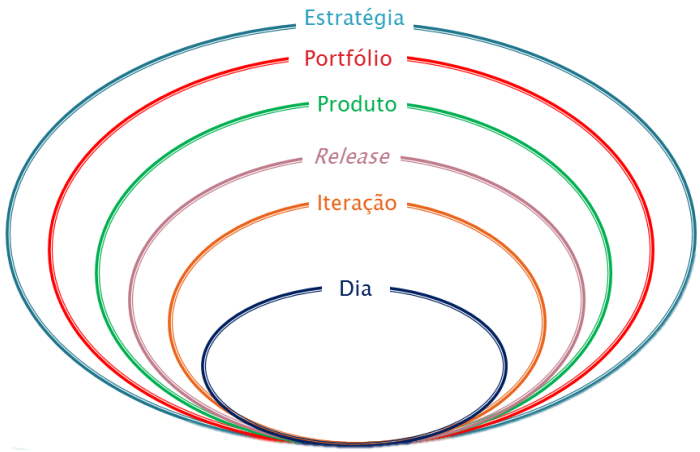
Project Name:				
Start Date:				
Clients				
Project Objective Statement:				
Trade Off Matrix				
	Fixed	Flexible	Accept	Target
Escope			X	1.250 pts
schedu		X		± 6 weeks
Budget			X	± U$5.000
Quality			X	
...				
Strategic Points				
Client Benefits				
Performance Attributes				
Product Architecture				

Exemplo de Planilha de Dados do Projeto

6.3 - Planos Adaptativos

As estratégias, portfólio e produto podem ser planejados com técnicas de gestão de Portfólio, Planejamento Estratégico e outras conhecidas e utilizadas no mundo dos negócios, porém algumas técnicas ágeis de definição de visão (como vimos anteriormente) auxiliarão a todos a

entender o que realmente é necessário para o sucesso do projeto. A figura abaixo apresenta a cebola do planejamento estratégico em projetos orientados á valor.

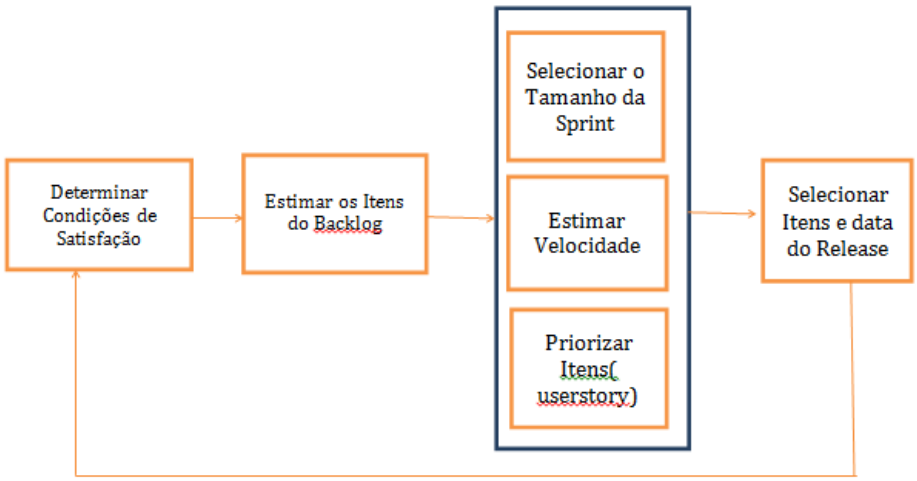


Cebola de Planejamento

6.4 - Planejamento de Release

Este planejamento é uma reunião de planejamento de alto nível que trata de iterações das futuras *Sprints*. O objetivo é planejar quais itens serão desenvolvidos, quando deverão estar prontos e quais recursos serão necessários para isso. As entregas irão ocorrer ao longo do projeto e a performance do time, assim como as ações de monitoramento e controle.

O fluxo abaixo ilustra uma sugestão de planejamento de *Release*.



Observe que serão necessários que o *Product Backlog* já esteja definido e priorizado, as datas das *Sprints* definidas, e (se for um time já existente) a velocidade média do time.

- *As datas de início/fim das futuras Sprints já devem considerar os feriados, férias de membros do time e quaisquer interrupções já planejadas.*
- *A formação do time impactará em todo o andamento, em projetos que temos uma restrição de tempo mais severa, a montagem de um time sênior mitiga o risco de atraso, porém impactará no custo.*

Com os itens pontuados e priorizados e com a estimativa média de entregas do Time, podemos definir quantas *Sprints* serão necessárias. Por exemplo, um projeto de 60 pontos com um time de 12 pontos de produtividade por *Sprint* durará 5 *Sprints*.(60 / 12 = 5).

6.5 - User Story

No levantamento tradicional de requisitos, o partimos do princípio que o cliente, em linguagem de negócios identifica suas necessidades, estas necessidades identificadas são passadas ao analista que traduz em linguagem técnica que posteriormente são entregues aos desenvolvedores que as transforma em códigos.

Observe que neste processo temos pontos fracos como:

- Premissas falsas podem ser geradas pelo cliente ou pelo analista
- Requisitos detalhados desnecessariamente(desperdício)

Na técnica de *User Story*, o dono do produto descreve o que precisa ser feito, identifica quem usará a funcionalidade e porquê (ajuda a não gerar itens que não agregam valor ao negócio).

Esta técnica utiliza os 3 C's:

- **Cartão**:
  Cada estória é escrita em um cartão com um objetivo específico, o que permite mais clareza no que é necessário que seja desenvolvido.

- **Conversa**:
  Como o cartão é uma descrição simples, ele leva a conversas com o time e com o cliente sobre a funcionalidade, o que permite um melhor entendimento sobre a percepção de valor, identifica riscos e prioridades.

- **Confirmação**:
  Através das conversas com o time e clientes poderemos entender como validar o cartão e confirmar se o que o temos definido é realmente o necessário o sucesso da demanda.

As estórias também utilizam os conceitos conhecidos com **INVEST**:

Independente: A estórias são mais fáceis de se trabalhar quando são independentes, isto é, não dependem de outras estórias para acontecerem.

Negociável – A estória não é um contrato com definição de funcionalidades, ela é negociável para melhor atender as necessidades do negócio.

Valiosa para usuários e clientes - A estória precisa estar associada a um valor para o usuário ou clientes, sem isso, não existe razão para ela existir.

Estimável – A estória precisa ser estimável, mesmo que com alguma imprecisão, precisamos dimensionar o esforço para implementá-la.

Small(pequena) – Estórias representam situações simples, com poucos personagens.

Testável – Toda estória precisa ser testável. O cliente deve identificar quais seriam as condições de testes da estória escrita. As condições de teste definidas pelo cliente ajudam o time a entender se a meta da estória foi bem sucedida.

Uma boa estória deve responder: **QUEM? COMO? POR QUE?**

Como um <**PERFIL**> eu posso /gostaria/devo <**FUNÇÃO**> para <**VALOR AO NEGÓCIO**>

Ou **POR QUE? QUEM? COMO?**

Com o propósito de**<VALOR AO NEGÓCIO**>, como um**<PERFIL>**, eu posso/gostaria/devo<**FUNÇÃO**>

Exemplo 1:

Uma estória para criar uma panela específica:

Como um *Cozinheiro(* **Usuário)**

Quero *panela de inox, com fundo oval e antiaderente*

*(***Funcionalidade)**

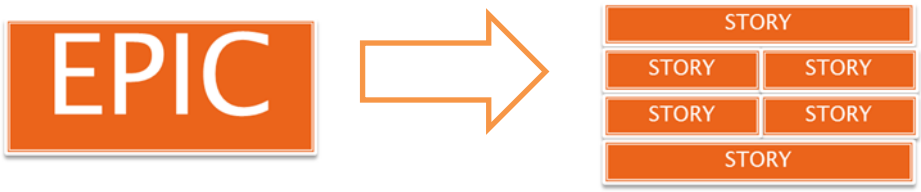
Para *cozinhar um salmão(***Valor de Negócio)**

Exemplo 2:

Como ***comprador*** que ***não tem cartão de crédito***
Quero que o sistema de ***suporte a pagamento em boleto bancário***

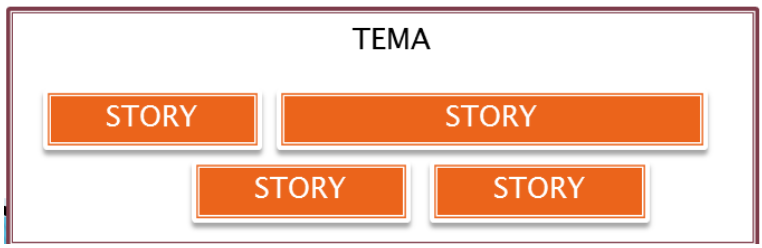
6.6 - Stories, Temas e EPICS

Algumas estórias podem ser mais complexas e com uma previsão maior do que a capacidade de entrega da Sprint, estas estórias são conhecidas com **EPIC**. Estas estórias precisam ser analisadas e revistas com o objetivo de identificar melhor seus objetivos e a sua possível decomposição em estórias menores.(Lembre-se que as estórias devem respeitar as premissas do INVEST).



Geralmente os EPICs aparecem na base do *backlog* do produto, pois ainda não possuem granularidade para serem implementadas.

Existem casos em que um grupo de estórias está associado a um determinado tema específico, nestes casos, no momento de priorização do *backlog* do produto, estas estórias tendem a serem produzidas na mesma *Sprint* ou em *Sprints* próximas.

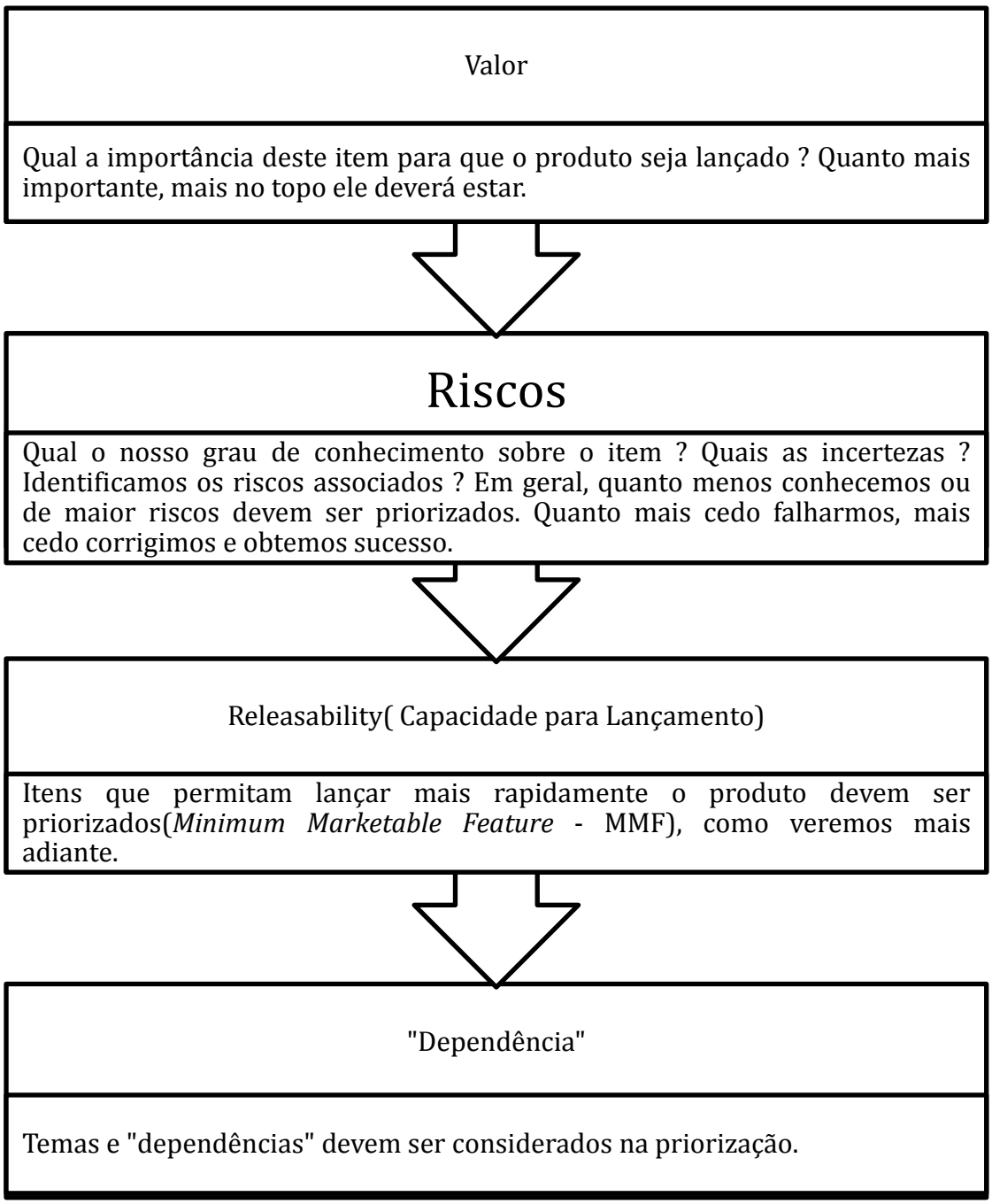


As estórias que estão agrupadas no tema NÃO estão associadas a nenhuma regra de precedência, lembre-se que as estórias devem ser Independentes.

6.7 - Priorização de Backlog

A priorização do *Backlog* é de responsabilidade do *Product Owner* e está diretamente associada a melhoria do retorno sob investimento (ROI), pois quanto antes as funcionalidades mais importantes forem entregues, antes gerarão retorno para o negócio.

Em geral, a priorização do *backlog* do produto segue uma sequência lógica, conforme apresentado abaixo:



Técnicas que podem ser utilizadas na priorização do *Backlog* de Produto

6.8 - Priority markets

Nesta técnica de priorização, simulamos um mercado, onde para cada avaliador é dado algum dinheiro virtual(Reais de desenvolvimento), que deverão ser usados para licitar itens do backlog.

O método é democrático, em que todos os interessados possuem voz no processo de priorização. Também permite transparência para que todos possam ver como ocorreu a priorização de cada item.

Esta ferramenta é simples de administrar e se adapta bem a um grande número de interessados e listas longas de itens do backlog, mitigando os debates sem fim em busca de um consenso para a priorização.

Passo 1: Defina a lista de itens ou temas.

Passo 2: Identifique as partes interessadas e o peso para cada parte de acordo com sua importância para o projeto.

Para nosso exemplo, vamos definir 4 atores e seus pesos.

Rafael (2.0)	Emanuel (1.0)	Helga (1.0)	Letícia (1.0)

Passo 3: Defina a quantidade de *Reais de desenvolvimento* no jogo e distribua-os entre os participantes. Estes *Reais* serão utilizados pelas partes interessadas para licitar e comprar as mudanças no sistema.

Dica:

O número exato do total de Reais de desenvolvimento não é critico. Na prática valores pequenos como R$10,00 por participante, já podem apresentar uma distribuição que representa bem a priorização. No nosso exemplo utilizaremos um valor de R$20,00 por peso/participante, como temos 5, o total será de R$100,00 de desenvolvimento

Por exemplo:

Total de ***Reais de Desenvolvimento*** **não alocado: R$100,00**

	Rafael (2.0)	Emanuel (1.0)	Helga (1.0)	Letícia (1.0)
Carrinho de Compras				
Alerta de Estoque				
Lista de Mais Vendidos				
Geração de Boleto de pagamento				
TOTAL NÃO GASTO				

Divida o total de Reais de desenvolvimento de acordo com o peso de cada participante.

	Rafael (2.0)	Emanuel (1.0)	Helga (1.0)	Letícia (1.0)
Carrinho de Compras				
Alerta de Estoque				
Lista de Mais Vendidos				
Geração de Boleto de pagamento				

TOTAL NÃO GASTO	R$40,00	R$20,00	R$20,00	R$20,00

Passo 4: Os interessados distribuem seus *Reais de desenvolvimento* sobre os itens do backlog de acordo com suas preferências. *Reais de desenvolvimento* que foram licitadas são deduzidos seus *Reais* não gastos. Imagine as partes interessadas alocar seus *Reais* da seguinte forma:

	Rafael (2.0)	Emanuel (1.0)	Helga (1.0)	Letícia (1.0)
Carrinho de Compras	R$6,00	R$8,00	R$5,00	R$5,00
Alerta de Estoque	R$5,00	R$4,00	R$3,00	R$4,00
Lista de Mais Vendidos	R$3,00	R$3,00	R$4,00	R$5,00
Geração de Boleto de pagamento	R$6,00	R$5,00	R$8,00	R$6,00
TOTAL NÃO GASTO	R$20,00	R$0,00	R$0,00	R$0,00

Passo 5: O time estima o “custo” do desenvolvimento de cada item.

	Rafael (2.0)	Emanuel (1.0)	Helga (1.0)	Letícia (1.0)	TOTAL PAGO	CUSTO ESTIMADO(DEV. TIME)
Carrinho de Compras	R$6,00	R$8,00	R$5,00	R$5,00	R$14,00	7
Alerta de Estoque	R$5,00	R$4,00	R$3,00	R$4,00	R$16,00	5
Lista de Mais Vendidos	R$3,00	R$3,00	R$4,00	R$5,00	R$15,00	5
Geração de Boleto de	R$6,00	R$5,00	R$8,00	R$6,00	R$24,00	11

pagamento						
TOTAL NÃO GASTO	R$20,00	R$0,00	R$0,00	R$0,00		

Passo 6:O ROI é calculado e a proposta de priorização é determinada.

Vamos calcular retorno sobre o investimento (ROI) de cada item backlog dividindo a oferta total (Total Pago) pelo custo de desenvolvimento (CUSTO ESTIMADO). Após o cálculo, vamos ordenar o backlog do maior para o menor ROI. No nosso exemplo, temos:

	Rafael (2.0)	Emanuel (1.0)	Helga (1.0)	Letícia (1.0)	TOTAL PAGO	CUSTO ESTIMADO(DEV. TIME)	TOTAL PAGO/ CUSTO ESTIMADO
Carrinho de Compras	R$6,00	R$8,00	R$5,00	R$5,00	R$14,00	7	14/7 = 2
Alerta de Estoque	R$5,00	R$4,00	R$3,00	R$4,00	R$16,00	5	16/5 = 3,2
Lista de Mais Vendidos	R$3,00	R$3,00	R$4,00	R$5,00	R$15,00	5	15/5 = 3
Geração de Boleto de pagamento	R$6,00	R$5,00	R$8,00	R$6,00	R$24,00	11	24/11 = 2,18
TOTAL NÃO GASTO	R$20,00	R$0,00	R$0,00	R$0,00			

Lista não priorizada	Lista priorizada

	TOTAL PAGO/ CUSTO ESTIMADO
Carrinho de Compras	14/7 = 2
Alerta de Estoque	16/5 = 3,2
Lista de Mais Vendidos	15/5 = 3
Geração de Boleto de pagamento	24/11 = 2,18

	TOTAL PAGO/ CUSTO ESTIMADO
Alerta de Estoque	3,2
Lista de Mais Vendidos	3
Geração de Boleto de pagamento	2,18
Carrinho de Compras	2

Dica:
Ordenando por ROI nos permite identificar a priorizar as melhores oportunidades. Veja que o item ***Geração de Boleto de pagamento*** *tem a maior oferta global(R$24,00), ele também tem um custo de desenvolvimento alto(R$11,00) e por isso está mais abaixo na lista de prioridades.*

Em um projeto típico, haverá um grande número de itens que não têm *Reais de desenvolvimento* pagos por eles. Esta é realmente uma vantagem do método, como as partes interessadas podem concentrar a sua atenção em um número limitado de itens críticos no *Product Backlog*.

Passo 7: Realocação de Reais de Desenvolvimento

Após a entrega do item **Carrinho de Compras,** por exemplo, os reais gastos nele estão poderão ser realocados. Este montante é primeiro colocado nos itens sem Reais pagos por eles. Os itens completos deverão ser removidos da tabela de licitação.

Nota: Reais de desenvolvimento não alocados poderão ser redistribuídos para as partes interessadas para a próxima rodada de priorização(geralmente na próxima iteração). Redistribuindo Reais de desenvolvimento e permitindo que as partes interessadas possam alterar lances anteriores, somos ágeis na resposta a mudanças nas prioridades dos *stakeholders*.

Geoff Watts e Jason Haines, em seu trabalho sobre *Priority Markets*, (disponível em https://www.scrumalliance.org/community/articles/2009/february/priority-markets) cita alguns comportamentos que surgem desta dinâmica, o qual os autores chama de "**A psicologia do mercado**". São eles:

- **Negociação**: Os interessados podem ver os lances uns dos outros e ajustar o seu lance de acordo com o mercado.
- **Negociação:** Uma das partes interessadas possa "vender" seus *Reais* de desenvolvimento para outro dos interessados para as prestações em espécie.
- **Votação tática:** Com base em outros lances, umas das partes interessadas podem alocar mais ou menos *Reais* para determinados itens.
- **Intervenção governamental:** Quando o mercado livre gera prioridades que não estão nos interesses estratégicos do projeto, o *Product Owner* pode intervir e alocar *Reais* adicionais para um determinado item. Como em qualquer mercado livre, a intervenção governamental pode ter uma série de consequências negativas: extras Reais injetados no sistema reduzem o valor relativo de outros

Reais, resultando em inflação; os interessados podem também se sentem marginalizados por terem suas prioridades substituídas.

6.9 - Theme screening

Outra técnica bem interessante de priorização é a *Theme Screening*. Para esta técnica, geralmente selecionamos de 5 a 9 critérios para avaliar o que é mais importante para o próximo *Sprint*. Estes critérios devem representar aspectos do nosso produto que consideramos importantes para a priorização de requisitos.

DICA:
Esses critérios podem ser definidos a partir da nossa visão de produto, ou de requisitos de negócio a que desejamos atender. Exemplos comuns podem incluir itens como "Colabora para atingir a meta", "Impacto nos processos organizacionais", "Elimina problemas antigos dos usuários", "Facilidade de desenvolvimento", "Posicionamento do produto", etc.

Após definir os critérios, selecione um tema que servirá como base para os fatores de comparação. Lembre-se que temas são grupos de requisitos funcionalmente ligados ou que tenham objetivos de negócio complementares. Este tema deve ser algo que deveria entrar no próximo Sprint.

Um fator importantíssimo a ser considerado na escolha dos critérios de análise e dos temas é que todos devem ser capazes de entender o significado de cada critério de análise e serem capazes de avaliar cada tema relativo a esses critérios.

Por exemplo, após a escolha podemos ter um quadro conforme apresentado abaixo.

CRITÉRIOS	Temas				
	Tema A	Tema B	Tema C (Base)	Tema D	Tema E
Colabora para atingir a meta	+	+	0	-	0
Elimina problemas antigos dos usuários	+	-	0	0	-
Facilidade de desenvolvimento	+	-	0	+	-
Impacto nos processos organizacionais	0	0	0	+	0
Score	3	-1	0	1	-2

Legenda:

+ = Mais que

- = Menos que

0 = mesmo que

Devemos agora escolher um tema como base line(no nosso exemplo, o Tema C), isto é, um tema que servirá de base de comparação para todos os demais temas. Procure buscar sempre um tema que tenha uma prioridade intermediária entre os temas escolhidos.

Agora, comparamos cada critério de análise com o nosso base line, ou seja, perguntamos pros nossos envolvidos se determinado tema é melhor ou pior que o base line em cada critério de análise. No nosso exemplo usamos " + " quando o tema comparativamente é melhor que o tema base para o critério, - se o tema contribui menos ou 0 se tem a mesma contribuição(Lembre-se que esta percepção é subjetiva).

Somamos o total para cada temas, e teremos uma pontuação total para o tema. Basta então ordenar nossos temas, veja como ficou a priorização de nosso exemplo.

1°) Tema A

2°) Tema D

3°) Tema C

4°) Tema B

5°) Tema E

ATENÇÃO

Existem 2 grandes riscos nesta técnica. O primeiro é a escolha de critérios equivocados, geralmente quando não foi bem desenvolvida a visão do produto. O segundo risco está associado a escolha do Base line, um com prioridade muito alta fará com que muitos temas pontuem negativo. Um base line muito baixo fará com que muitos temas pontuem muito alto. Assim, a escolha de um base line intermediário na escala de prioridade é bem importante. Se errarmos na escolha da base line, poderemos criar distorções nos resultados.

6.10 - Kano model

A técnica Kano é baseada em entrevistas com os usuários e experts. Ela é bem interessante quando a opinião de todos tem o mesmo valor (diferentemente da técnica de *Priority Markets*, onde podemos atribuir pesos maiores para os *Decision Makers*). Esta técnica foi desenvolvida por um professor de gerenciamento de qualidade chamado *Noriaki Kano* que estudou e classificou as preferências dos clientes em 3 categorias:

1. **Mandatório** (*Basic expectations*): são características que são consideradas básicas que devem estar presentes no produto. A ausência delas irá frustrar o cliente, mas a sua presença não irá aumentar a satisfação dele, já que o cliente espera que aquela característica esteja presente.

2. **Linear** (*Satisfiers*): são características que quanto mais, melhor e que podem aumentar ou diminuir a satisfação do cliente. Geralmente estão ligadas à performance.

3. **Desejadas** (*Delighters*): são características que satisfazem o cliente quando estão presentes no produto, porém, caso não estejam não irão causar insatisfação.

Após a identificação das funcionalidades, realizamos questionários com grupos de 10 a 30 usuários com perfis variados, com uma pergunta funcional e outra disfuncional.

Pergunta Funcional: Como você se sentirá se esta funcionalidade estiver presente?

Por exemplo:

Se o próximo release incluir o carrinho de compras, como você se sentirá ?

( ) Eu **gostaria** que fosse desta forma
(X) Eu ficarei **neutro**
( ) Eu **posso viver** desta forma
( ) Eu **não gostaria** que fosse desta forma

Pergunta Disfuncional: Como você se sentirá se esta funcionalidade estiver ausente ?

Por exemplo:

Se o próximo release NÃO incluir o carrinho de compras, como você se sentirá ?

( ) Eu **gostaria** que fosse desta forma
( ) Eu ficarei **neutro**
( ) Eu **posso viver** desta forma
(X) Eu **não gostaria** que fosse desta forma

E então utilizando-se a seguinte categorização para cada conjunto de respostas(funcional e não funcional):

Nota: O exemplo abaixo foi baseado no trabalho de *Martin Eriksson*, chamado *MindtheProduct*, disponível na URL http://www.mindtheproduct.com/2013/07/using-the-kano-model-to-prioritize-product-development/ e adaptado pelo autor.

		Disfuncional				
		Gostaria	**Deveria**	**Indiferente**	**Suporta**	**Não Gostaria**
Funcional	**Gostaria**	**Q**	**D**	**D**	**D**	**L**
	Deveria	**R**	**I**	**I**	**I**	**M**
	Indiferente	**R**	**I**	**I**	**I**	**M**
	Suporta	**R**	**I**	**I**	**I**	**M**
	Não Gostaria	**R**	**R**	**R**	**R**	**Q**

LEGENDA

M	**Mandatório**	**I**	**Indiferente**
L	**Linear**	**R**	**Reverso**
D	**Desejado**	**Q**	**Questionável**

No caso do exemplo essa funcionalidade seria Desejado. Agora, agregando os resultados, onde as funcionalidades que estamos verificando o grau de satisfação que elas apresentam:

	Desejado	**Linear**	**Mandatório**	**Indiferente**	**Reverso**	**Questionável**
Funcionalidade A	**3**	**12**	**29**	**1**	**3**	**2**
Funcionalidade B	**22**	**6**	**21**	**6**	**1**	**0**
Funcionalidade C	**4**	**21**	**23**	**5**	**4**	**5**

Com base nesse último quadro temos que a funcionalidade A é o básico que os clientes esperam ter, veja que temos 29 entrevistados que classificaram a funcionalidade A como Mandatório. Mas é preciso lembrar que, uma vez alcançado o básico, não devemos mais adicionar esforço nessa funcionalidade pois ela não irá aumentar a satisfação do cliente. Devemos apenas mantê-la. A funcionalidade B parece ser vista por algumas pessoas como básica e por outras como “quanto mais, melhor” e a funcionalidade C é vista como um diferencial. Portanto, a prioridade seria A > B > C.

Para se manter o *Product Backlog* priorizado por esse modelo é necessário que todas as funcionalidades/características mandatórias estejam no *Road Map* (sem exceder a quantidade de esforço necessária para alcançar o básico esperado), fazer o máximo de funcionalidades/características lineares e sempre tentar deixar um espaço para as funcionalidades desejadas, pois essas podem aumentar o grau de satisfação do cliente rapidamente a seu favor.

6.11 - MMF – Minimum Marketable Feature

A Técnica de MMF pode ser utilizada para priorização de seu backlog de produto e para o planejamento de versões. Ela baseia-se em identificar as características mínimas comercializáveis de seu produto. Ela baseia-se no princípio de priorizar o essencial para a geração de valor, por exemplo, para o desenvolvimento de um telefone celular as funcionalidades essenciais são as de ligar e desligar e o envio de mensagens de texto, assim estas funcionalidades devem ser priorizadas e estarem na primeira versão do produto(ou no topo do backlog de produto), ouvir música, registrar fotografias e outras podem aparecer em versões mais avançadas(base do backlog de produto).

A regra de ouro é: **desenvolva as características de maior valor primeiro (maximização do seu retorno).**

Dica

Para simplificar o planejamento de versões, elimine dependências técnicas, lembre-se que as estórias devem respeitar o conceito de INVEST.

CAPÍTULO 7

7.1 - Estimativas ágeis

Um dos grandes desafios de profissionais da área de desenvolvimento de software é como estimar o esforço para a criação de novas funcionalidades, tarefas ou estórias. O principal objetivo de se estimar é criar uma métrica comum para definição e comparação de esforços, e consequentemente definições de prazos(e orçamentos) para projetos e/ou atividades.

Temos diversas formas de estimar software desde os antigos KLOCs(*Kilo lines of Code* ou mil linhas de código), homens – hora, pontos de função entre outras. É inegável que estimar é FUNDAMENTAL! Mas vemos que grande parte do esforço empreendido em estimar com precisão é perdida, ou devido a alguma premissa que não se tornou verdadeira ou porque a medida que trabalhamos no projeto, com mais conhecimento, somos obrigados a ajustar nossas estimativas.

As formas mais comuns em estimativas são as apresentadas pelo PMBok®, guia de Boas Práticas do Project Management Institute® e estão apresentas abaixo:

Estimativa de um ponto: Apresenta a estimativa por atividade. Pode ser baseada na opinião especializada, informações históricas ou simplesmente adivinhação.

Estimativa Análoga(“*Top Down*”): Usam opinião especializada e informações históricas para prever o futuro

Estimativa Paramétrica: Observa os relacionamentos entre as variáveis em uma atividade para calcular estimativas.

Estimativa de Três Pontos(Análise PERT): Considera a média entre a estimativa no melhor cenário(O), adicionada a quatro vezes a estimativa mais provável(M), mais a estimativa do pior cenário(P) , divido por seis.

(P + 4M +O) / 6

Atividades de desenvolvimento de software (projetos orientados à valor) geralmente são bem mais complexas do que atividades da construção civil (como uma pintura de corredor) – projetos orientados à planos, pois em um novo projeto de software**, os requisitos nunca serão completamente conhecidos até que o usuário os tenha utilizado**, assim o grande desafio nas estimativas de software está relacionado às dificuldades da clara definição do escopo do produto, da definição da estratégia de como uma atividade será realizada e pelo simples fato que em um grupo distinto, a experiência, conhecimento técnico, cultura, etc. irão influenciar na estimativa de cada um, por isso todas as estimativas tendem a apresentar imprecisão. Lembre-se que quando utilizamos “horas ou dias” como unidade de medida, estamos estimando não só o esforço necessário para a tarefa mas também a velocidade individual dos membros que trabalharão nela.

A comunidade adepta aos métodos e frameworks ágeis, geralmente, procura utilizar estimativas que levam em consideração apenas o esforço para a realização de determinada tarefa(estórias) , onde todos os envolvidos no desenvolvimento, medem as diversas tarefas e assim definem estimativas comparativas(com margens de erro) entre todo o trabalho que precisa ser realizado.

Veja a história relatada por James Surowiecki que em seu trabalho intitulado " A Sabedoria das Multidões" queapresenta uma experiência realizada em 1906, quando o cientista britânico Francis Galton ficou chocado com o resultado de um experimento realizado por ele, em uma feira municipal. O desafio era que um açougueiro profissional deveria adivinhar com precisão o peso de um boi abatido. Sua surpresa ocorreu ao descobrir que pessoas que visitavam a feira(com pouca ou nenhuma experiência em cortes de carne) eram capazes não só de adivinhar o peso final do animal, mas eram capazes de adivinhar com grande precisão(inclusive com quilogramas e os detalhes em gramas).

A expectativa de Sir Francis era que os peritos(açougueiros profissionais) estavam sempre bem e iria superar com folga uma multidão, o que não ocorreu.

Quando estamos jogando o *Planning Poker*, uma técnica de estimativa utilizada pela comunidade ágil, nós igualmente estamos aproveitando a sabedoria das multidões com relação a nossas estimativas. Estamos apostando que o público será capaz de chegar a um melhor palpite do que qualquer indivíduo único.

O *Planning Poker* é uma técnica de estimativa baseada no consenso de toda a equipe(em suas experiências, conhecimentos e aptidões), onde é utilizado um conjunto de cartas com valores específicos(pontos) relativos. O Product Owner apresenta a tarefa ou estória para o time, e, após uma breve discussão, cada um escolhe uma carta e coloca virada para baixo sobre uma mesa. Quando todas as cartas estiverem lançadas, elas são viradas e caso não haja consenso nos valores escolhidos, estas diferenças são discutidas de forma breve, geralmente, quem estimou com os maiores e menores valores explicam o motivo de sua estimativa, e uma nova rodada acontece até que haja a convergência.

Exemplo de cartas que geralmente são utilizadas no *Planning Poker*

O Planning Poker funciona porque utiliza opinião de diversos especialistas(que irão realmente realizar a tarefa), promove o diálogo que permite maior acuracidade das estimativas, principalmente em itens com maior incerteza. Além de ser uma técnica de estimar considerando a experiência de todos da equipe, proporciona um conhecimento do negócio mais homogêneo e é mais atraente (e divertida) que as demais técnicas.

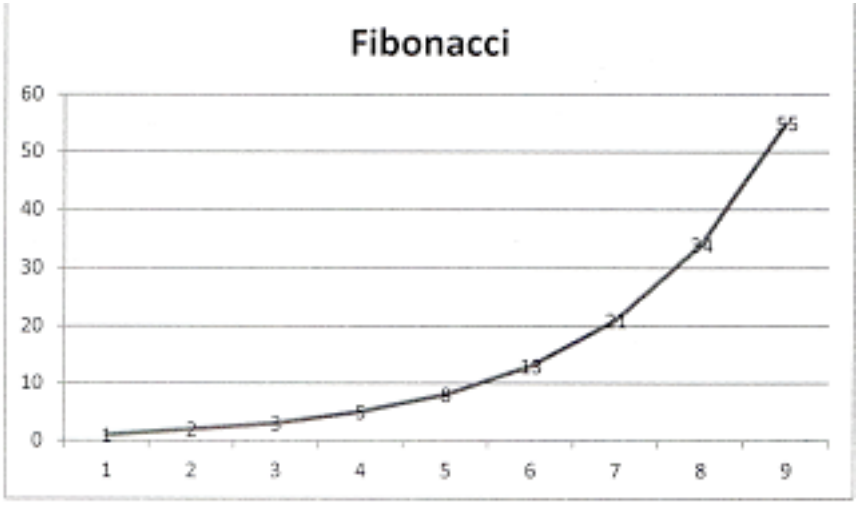


Gráfico da Sequência de Fibonacci

Geralmente o baralho do *Planning Poker* apresenta cartas com escala da sequência de Fibonacci, isso se deve ao fato da sequência de Fibonacci ser uma função quadrática, ao invés de uma função linear, assim, as diferenças entre valores são produzidas de forma muito rápida criando intervalos logo no início da sequência, veja que as estimativas apresentam erros, mas quando podemos comparar as tarefas(uma tarefa com 8 pontos não significa que ela tem exatamente este tamanho mas é algo entre 5 e 13 pontos) conseguimos dimensionar com mais precisão o esforço que deverá ser empreendido. Existe também a opção infinita, carta com a marcação "∞", geralmente quando o time identificou uma estória que considera EPIC.

Outra forma de estimativa muito utilizada no mundo ágil, é a técnica PMG, uma analogia as medidas de roupas, onde cada item é classificado como P para pequeno, M para médio e G para grande, de acordo com a percepção de esforço do time para entregá-la.

Sempre devemos lembrar que:

- **A Definição dos pontos de cada estória é de exclusividade do Time**. Lembre-se que o esforço empreendido em cada Sprint(ou interação) pode variar de acordo com cada time. 18 pontos para o "time A" e 18 pontos para o "time B", NÃO SÃO O MESMO ESFORÇO. A percepção de quanto esforço é necessário para realizar um ponto é subjetiva e diferente para cada time.

- **Os pontos definido em cada estória devem envolver TODO o esforço para entregá-la pronto para funcional no ambiente real.** Significa que devemos considerar a complexidade devido ao desconhecimento ou incertezas, o Esforço do trabalho e os Riscos associados em nossas estimativas.

- **O Tamanho do esforço deve ser relativo**. Isto é, quando dizemos que para entregar a “estória A” definimos 8 pontos e para a ”estória B” 16 pontos, significa que o esforço para entregar B é o dobro de A(mesmo que haja uma pequena variação devido a natureza imprecisa das estimativas ágeis). Observe que neste ponto devemos estar atento as estimativas por afinidade, podemos agrupar as estórias por proximidade de esforço pontuado(pelo TIME), o que facilita o dimensionamento do trabalho como um todo.

7.2 - Um pouco mais sobre times ágeis

Como nos vimos diversos métodos ágeis temos processos e eventos como as reuniões de planejamento, reuniões diárias, retrospectivas e outras, porém o primeiro valor do manifesto ágil é:

“Indivíduos e interações mais que processos e ferramentas”

Os processos e eventos são facilmente descritas (até seguidas), porém eles sempre dependerão dos indivíduos e suas interações, e estes já são mais difíceis de serem generalizados e documentados como regras. Veja que cada indivíduo possui características únicas como personalidade, conhecimento técnico, cultura, valores, momento de vida, etc. e todos estes fatores influenciarão nas interações e em como o trabalho fluirá.

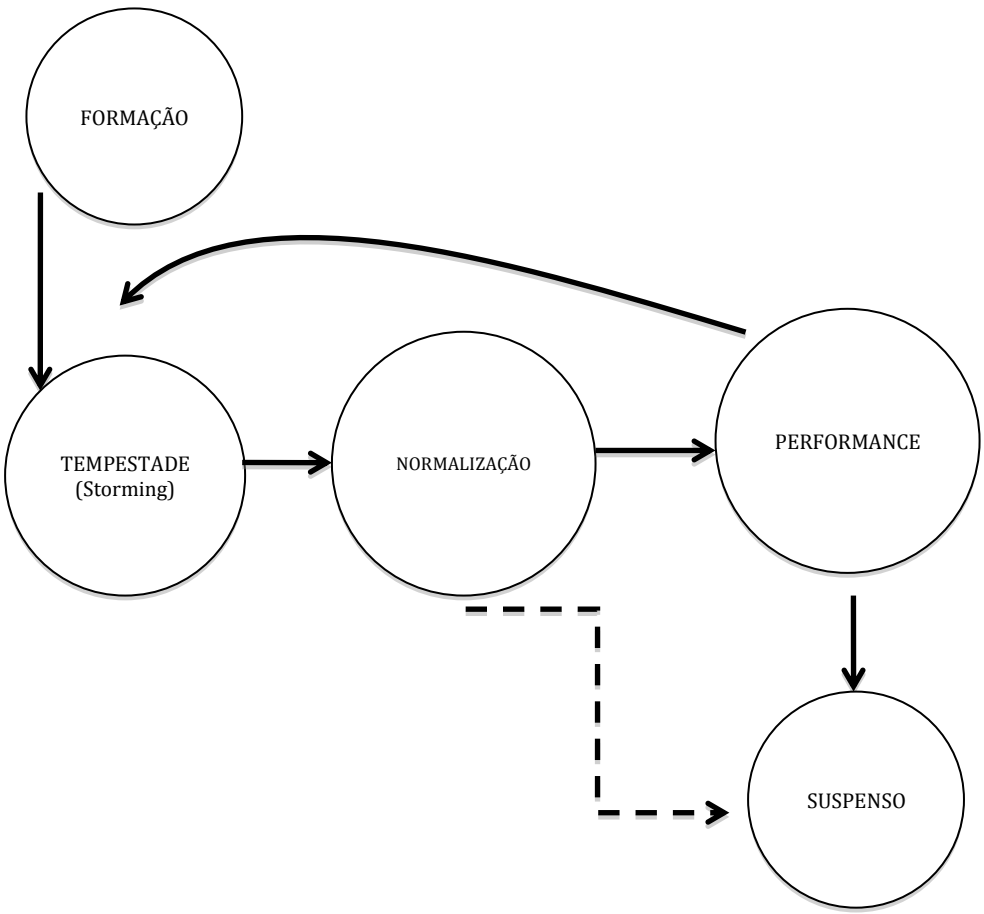
Este valor do manifesto ágil reflete a seguinte sentença mensagem :

> ***“ Bons profissionais mesmo com poucos processos(ou até mesmo sem nenhum) conseguem superar os maiores desafios. Mesmo utilizando os melhores processos, um time ruim irá falhar até mesmo em algumas tarefas simples.***

"Líderes de times ágeis devem concentrar-se nos fatores humanos(pessoas e relacionamentos) para obterem o melhor resultado".

7.3 - Liderança Adaptativa

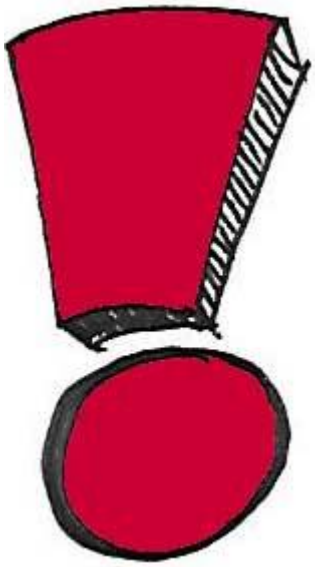
O Termo liderança adaptativa se refere a como devemos lidar com time dependendo da maturidade do time e de cada circunstância. O autor Bruce Tuckman, no trabalho "*Developmental Sequences in Small Groups"* – *Psychological Bulletins 63(1965)* foi um dos primeiros a identificar as fases em que um time pode se encontrar desde a sua formação até chegar a um estado de time de alta performance.



Na fase de **Formação** os membros (ainda não são um Time) começam a trabalhar em grupo, nesta fase eles iniciam o processo de aprendizado uns sobre os outros, neste momento inicia-se o autoconhecimento do futuro time. Nesta fase identificamos um grupo altamente comprometido, porém sem grandes conhecimentos, logo, a liderança deverá atuar bastante no direcionamento do trabalho, para isso questionamentos como: *Deixe-me ver isso? Onde está o problema? Qual o próximo passo?* Serão bem comuns.

Nota: A **Formação** ocorre sempre que um novo membro é introduzido ao time(mesmo que os demais, ou a maior parte do time permaneça). Nestes casos o tempo tende a ser menor do que a formação de um time completamente novo.

Com o andamento do trabalho diversos conflitos e divergências irão surgir, o grupo passou para a fase de **Tempestade**, agora o grupo se torna um pseudo-time, os seus membros desafiam-se entre si para a realização dos trabalhos. Durante a fase da **Tempestade** observa-se diálogos mais ásperos, conflitos abertos e até insatisfação por isso a liderança precisa atuar orientando o time nos diversos aspectos e auxiliando na solução de questões.



ATENÇÃO

Quando falamos em **CONFLITO**, devemos sempre lembrar que:

- O conflito é natural e força uma busca de alternativas
- O conflito é uma questão de equipe
- A abertura resolve conflitos
- A resolução de conflitos deve se concentrar em questões e não em personalidade
- A resolução de conflitos deve se concentrar no presente e não no passado

O *Project Management BodyofKnowledge*(PMBok)®, guia de Boas Práticas do *Project Management Institute*(PMI)®, sugere que podemos mitigar os conflitos, sempre lembrando e relembrando ao time:

- Exatamente para onde o projeto está direcionado
- As restrições e objetivo do projeto
- O conteúdo do Termo de Abertura do Projeto
- Mudanças
- Decisões Importantes
- Tornar as tarefas desafiadoras
- Seguir as boas práticas de gerenciamento de projetos(Lembro que estamos tratando de projetos orientados à valor !)

O PMBoK®, apresenta as sete origens de Conflitos em ordem de freqüência:

1º) Cronogramas
2º) Prioridades do Projeto
3º) Recursos
4º) Opiniões Técnicas
5º) Procedimentos Administrativos

6º) Custo

7º) Personalidade

Na fase de **Normalização** o time em potencial começa a se tornar um verdadeiro time, aprendendo como trabalhar uns com os outros e como time, neste momento o TIME deve criar regras para ajudar\governar o trabalho do próprio time. Neste momento a liderança é um pouco mais “tímida” e concentra-se mais no auxilio à resolução de alguns conflitos, assim como lembrar constantemente das regras criadas por eles mesmos. Nesta fase é um bom momento para desafiar o time, para auxiliá-lo a se tornar um time de alta performance.

Na fase de **Performance** os times são altamente competentes e comprometidos, autônomos, empoderados, auto organizados e auto-policiados. O time trabalha com um só, com alta performance no trabalho e nas suas entregas. Cada indivíduo conhece bem as características dos demais membros do time, a liderança assume um papel de trazer novos desafios(para que o time resolva) sempre buscando a melhoria contínua. Muitos times dificilmente chegam a fase final devido a mudanças na sua composição.

7.4 - Inteligência Emocional

Inteligência emocional é a nossa capacidade de identificar, avaliar e influenciar nossas próprias emoções, de outros indivíduos e de grupos. A imagem abaixo identifica os diferentes aspectos da inteligência emocional.

Próprio	Outros	
Auto-Gerenciamento Auto-Controle Consciência Adaptabilidade Direcionamento e Motivação	**Habilidades Sociais** Auto-Controle Liderança Inspiradora Desenvolvimento dos demais Trabalho em equipe e colaboração	Regular
Auto-Conhecimento Auto-Confiança Auto-conhecimento Emocional Precisa Auto-Avaliação	**Consciência Social** Empatia Consciência Organizacional Compreender o Ambiente	Reconhecer

A separação dos aspectos da inteligência emocional nos quadrantes significa que primeiro precisamos conhecer nossas emoções para poder controlá-las, isto é, precisamos saber o que nos irrita, nos frustra, nos faz perder o foco para depois escolhermos uma estratégia para tratar e responder a estas situações e sentimentos. Devemos desenvolver estas habilidades para decidir se vamos deixar estas emoções nos afetar ou se iremos responder de maneira diferente. Daniel Goleman, no artigo "*Primal Leadership: The Hidden Driver of Great Performance",* publicadona*Harvad Business Review*, em 2001, afirma:

"*O humor e comportamento do Líder conduz os humores e comportamentos de todos os outros. Um chefe mal-humorado e cruel cria uma organização tóxica preenchida com sentimentos negativos que ignoram as oportunidades*."

Como líderes de times devemos trabalha as habilidades que nos permitam identificar as situações que influenciam negativamente os membros do time a ajuda-los (os membros) a desenvolverem estratégicas para mitiga-las ou eliminá-las. Reconhecer que outros precisam de ajuda nos ajuda a melhorar a performance individual e do time, promove a colaboração e o trabalho em equipe.

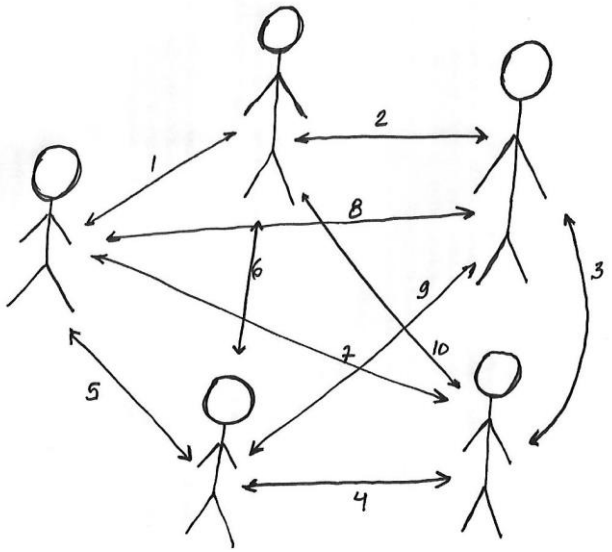
7.5 - Comunicação e o ambiente de trabalho

Segundo o Antigo Testamento (Gênesis 11,1-9), os descendentes de Noé, com a intenção de eternizar seus nomes, iniciaram o projeto da construção da torre do templo de Marduk,(nome cuja forma em hebraico é Babel ou Bavel e significa "porta de Deus") uma torre tão alta que chegaria céu. O projeto provocou a ira de Deus que para puni-los, decidiu criar os diversos idiomas para o povo da Terra, assim o processo de comunicação entre os “construtores” foi comprometido e a torre não foi terminada. Este mito é uma tentativa de se explicar as diversas origens dos idiomas e apresenta um risco real em todos os projetos (inclusive nos projetos atuais).

Segundo o *Project Management Institute*(PMI)® existe uma correlação direta entre a capacidade de comunicação e o desempenho do projeto, isto é, um bom gerenciamento de comunicações é fator determinante de sucesso ou fracasso em projetos. Segundo o *Project Management Body of Knowledge*(PMBok)® que é um conjunto de boas práticas em gerenciamento de projetos, o gerenciamento das comunicações do projeto inclui os processos necessários para assegurar que as informações do projeto sejam geradas, coletadas, distribuídas, armazenadas, recuperadas e organizadas de maneira oportuna e apropriada.

Observe que muitas vezes não dedicamos o tempo necessário neste gerenciamento, mas vamos a um simples exemplo:

Se o projeto possui 5 integrantes, existem 10 canais de comunicação, veja a imagem abaixo.



Pode-se descobrir o número de canais de comunicação de uma forma relativamente simples, veja:

[N *(N-1)] / 2
Onde: N é o número de pessoas envolvidas no projeto

Existem várias barreiras no processo de comunicação como: ambientes ruidosos, distância, codificação inadequada da mensagem, fazer declarações negativas, hostilidade, idioma, cultura e outros. Um ponto crítico das interações dentro do projeto é deixar claro que as comunicações precisam de formalidade e devem necessariamente passar pelo gerente de projetos, caso contrário, o risco de problemas de comunicação podem crescer significativamente.

O primeiro passo para um bom plano de comunicação é identificar todas as partes interessadas em um projeto - elas são: quem influencia ou é influenciado pelo projeto ou pelo resultado do projeto – já que elas podem

influenciar o fracasso ou o sucesso do projeto, é claro que algumas partes interessadas podem influenciar mais do que outras em um projeto, mas é importante que TODAS sejam identificadas. Dentre muitos fatores, a identificação das partes interessadas, possibilita ao gerente ter uma visão dos interesses e expectativas de cada parte em relação ao projeto, o que ajuda muito em negociações e no gerenciamento das expectativas.

Um conceito básico é que as comunicações devem ser eficientes(fornecendo apenas as informações necessárias) e eficazes(fornecendo informações nos formatos certo, no momento certo).

- Quem deve receber quais informações?
- Quais são as reais necessidades de informação?
- Qual informação é necessária, de que tipo?
- Em que formato e meio deve ser transmitida a informação?
- Com que frequência?
- Qual é o fluxo de informações?

Existem diversos tipos de comunicação, escrita, verbal, formal, informal, não formal, para-linguística, interativa, passiva, ativa e outras. A cultura da empresa pode ser uma barreira ou um facilitador, por isso uma analise da cultura organizacional é fundamental no papel do gerente de projetos como integrador de áreas, ou seja, inteligência conversacional. Um gerente de projeto passa aproximadamente 90% do seu tempo se comunicando e para que o projeto tenha maiores chances de sucesso, entre outras coisas, um bom plano de comunicação é fundamental.

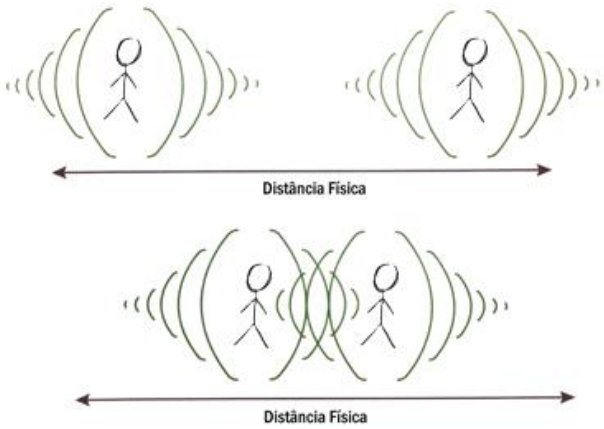
Muitas técnicas, ferramentas e recomendações ágeis preveem a interação “cara-a-cara”, assim o espaço onde o time irá trabalhar pode ser uma fator de melhora nos processos. Times ágeis preferivelmente devem trabalhar em espaços sem barreiras e de fácil acesso, agrupar os times em um mesmo ambiente facilita a interação, diminui o desperdício(provocado por

espera por uma informação ou deslocamento) além de promover a comunicação osmótica.

Comunicação Osmótica: Refere-se a toda informação útil que flui de membros da equipe como parte de conversas e perguntas diárias, quando trabalham em estreita proximidade um do outro.

Alistair Cockburn, em seu trabalho "*Agile Software Development*" define a comunicação osmótica como:

> *"Campos de energia que irradiam de pessoas. Se você está muito longe, você recebe muito pouco, mas se você estiver trabalhando em estreita proximidade, você obter todos os benefícios"*.



DICA

Para facilitar a comunicação cara-a-cara e o compartilhamento de conhecimentos, é recomendado que os times sejam de tamanho reduzido, geralmente até 12 membros ou menos(Lembre-se que o aumento da complexidade da comunicação é diretamente proporcional ao número de pessoas envolvidas). Quando os projetos são maiores o recomendado é trabalhar com diversos times, utilizando técnicas como o Scrum sob Scrum.

7.6 - Equipes virtuais

Quando trabalhamos com equipes virtuais (distantes geograficamente) demos utilizar os recursos tecnológicos(Programas de Mensagens Instantâneas, Videoconferências, VoIPs, Interfaces Remotas e Colaborativas e outros, para promover a comunicação constante.

Sempre que possível promova pelo menos uma reunião onde todos os membros do time estejam presentes, mesmo que no dia a dia eles trabalhem distante, uma atividade para que eles se conheçam(fisicamente) catalisa a fase de formação e melhora as futuras comunicações.

7.7 - Acompanhamento de produtividade

Um dos grandes desafios de projetos ágeis é o acompanhamento de produtividade. A medição é fundamental para planejarmos cronologicamente as versões e itens que serão entregues em cada Sprint.

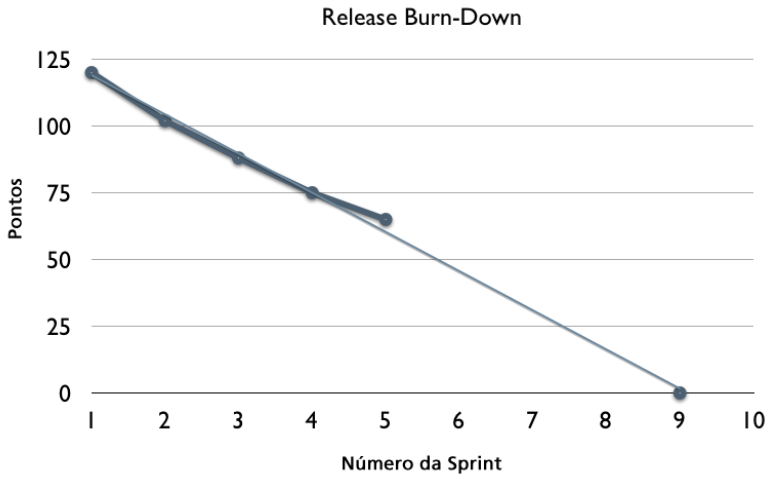
Antes de apresentarmos as formas de acompanhamento de produtividade, devemos lembrar que:

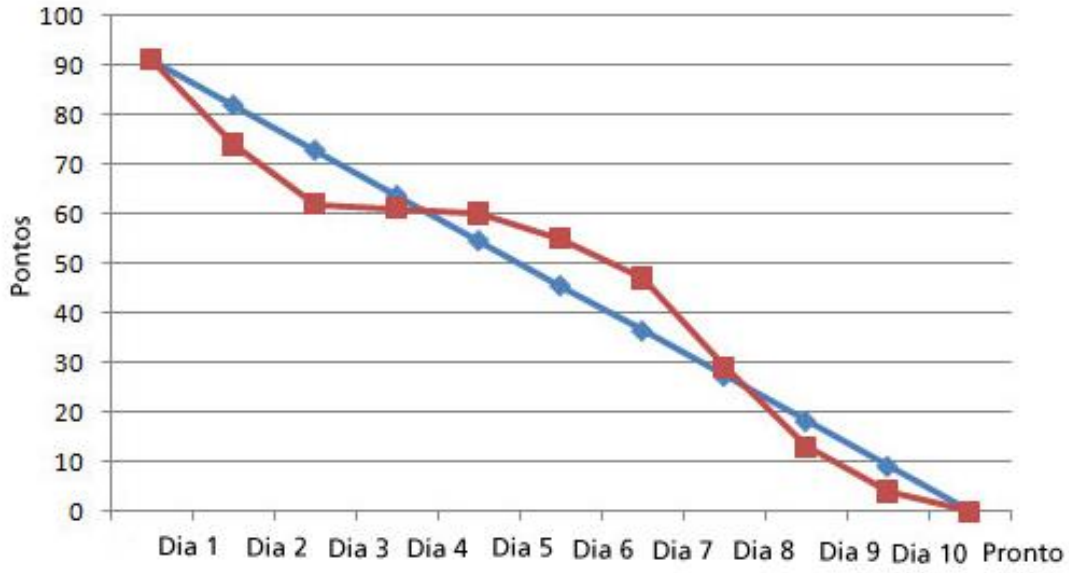
- **A mudança do tamanho da Sprint ou da composição do Time (mesmo que seja a saída ou entrada de apenas um novo membro) alterará a base histórica de entrega do time e as medições deverão ser novamente contabilizadas.**

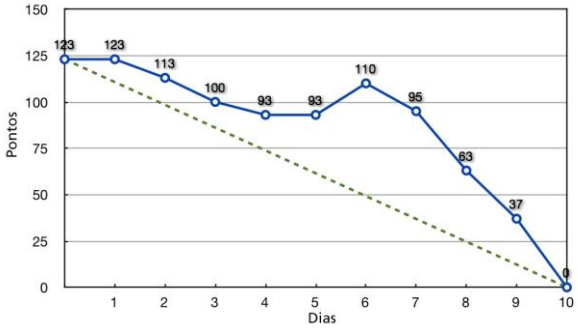
Release Burndown Chart

A ferramenta de Release *Burn-down* permite acompanhar a evolução das versões de seu produto de maneira simples e visual. O eixo vertical apresenta o esforço do release (este esforço foi estimado pelo time, em pontos, horas ou outra métrica pré-acordada), o eixo horizontal representa as

sprints. Observe que o eixo tem comportamento decrescente e inicia-se no topo do eixo vertical e termina na base(ponto O) deste mesmo eixo.





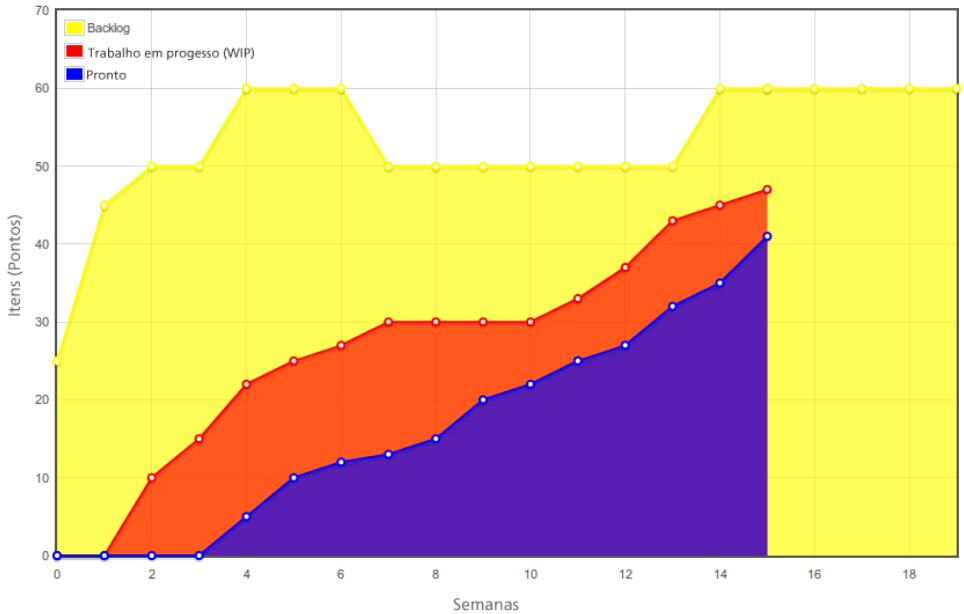


Podemos acompanhar o Previsto X Realizado em cada Sprint, assim como a velocidade do time(pontos ou horas) por Sprint, o que facilitará o planejamento para próximas versões do produto.

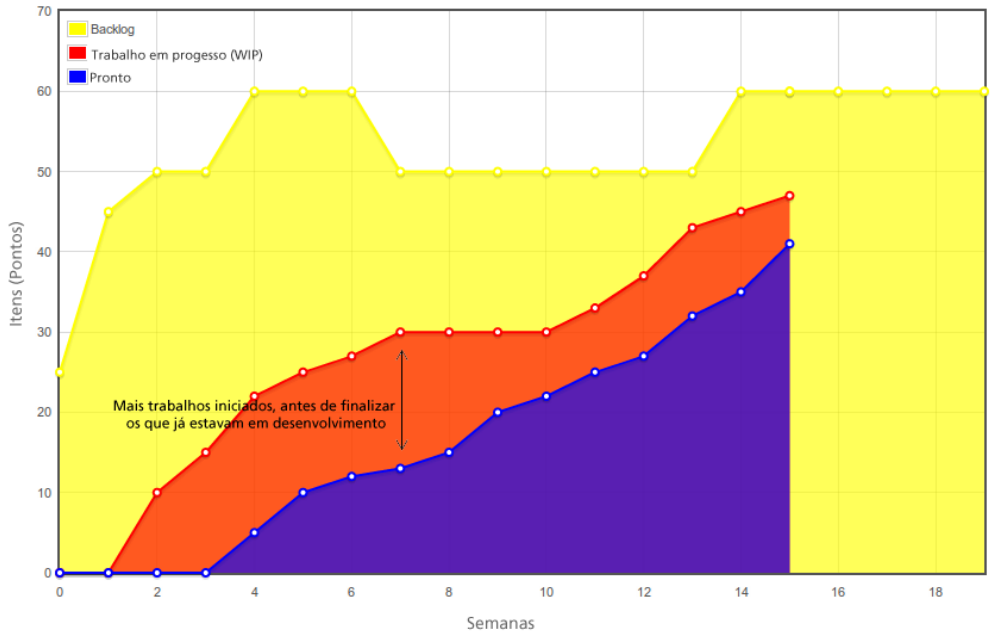
7.8 – Cumulative Flow Diagram(CFD)

O CFD é uma ótima ferramenta para monitoramento do trabalho de desenvolvimento. Com ele podemos acompanhar o trabalho que falta, o trabalho em progresso (WIP) e o trabalho pronto. Seu funcionamento é simples, no eixo vertical temos o trabalho estimado em pontos, horas ou outra métrica pré-definida pelo time, no eixo horizontal, o tempo(em dias, semanas ou sprints). Uma grande diferença entre o CFD e o Burn-down Chart é que podemos visualizar o trabalho em progresso.

Na imagem abaixo, a área em cor amarela apresenta o trabalho que



está pendente, isto é, trabalho não iniciado, a área laranja representa o trabalho em progresso e a azul o trabalho pronto.



Observe que o gráfico pode ajudar o Scrum Master e o time a analisarem o comportamento do processo e auxiliar os ajustes necessários. No exemplo da imagem temos uma área com grande amplitude na cor laranja(WIP), significa que o time iniciou mais tarefas em paralelo antes de finalizar as que já estavam em desenvolvimento, esta ação, em geral, deve ser evitada, lembre-se que é mais importante entregar tarefas do que iniciar novas(acumulo de WIP, NÃO é bom sinal!).

Dica:

Quando o gráfico de seu time apresenta grande distância do pronto para o não iniciado significa que seu time está com muitos trabalhos em progresso, é possível(e aconselhável) determinar um limitador para atividades em andamento, por exemplo, só 2 estórias em paralelo, para evitar o acúmulo do trabalho em andamento e o atraso nas entregas.

MENSAGEM FINAL

Os novos cenários organizacionais, com o uso estratégico de informações e conhecimentos como o principal diferencial competitivo demanda novas técnicas de gestão e gerenciamento diferentes das tradicionais estratégias de comando e controle, assim a adoção de técnicas emergentes para projetos orientados à valor vem ganhando grande crescimento em diversas áreas que lidam com profissionais de conhecimento. A mudança de qualquer cultura é um grande desafio, mas necessária para a adaptação e sobrevivência nas novas dinâmicas empresarias. **Evoluir é questão de sobrevivência, e para isso é preciso novos conhecimentos e CORAGEM!**

SOBRE OS AUTORES:

Rafael Dias Ribeiro

Analista de Sistemas formado pela Universidade Estácio de Sá, Mestre em Sistemas e Computação pelo Instituto Militar de Engenharia, Project Management Professional pelo PMI , Certified Scrum Master CSM e Certified Scrum Product Owner - CSPO pela ScrumAlliance e COBIT Foundation Certificate pela ISACA. Possuo experiência nas áreas de modelagem e desenvolvimento de sistemas de computação, gerenciamento de projetos, práticas emergentes de agilidade em gerenciamento de projetos complexo, mapeamento de processos de negócio, planejamento e gerência acadêmica, ensino à distância e gestão estratégica.

Horácio da Cunha e Sousa Ribeiro

Graduado em Engenharia Elétrica pela Universidade do Estado do Rio de Janeiro (1975) , Mestrado em Engenharia de Sistemas - Informática pelo Instituto Militar de Engenharia (1985), MBA em Gestão de IES pela UNESA. Diretor Geral do Instituto Superior de Tecnologia do Estado do Rio de Janeiro (FAETEC) - Coordenador de Pós-graduação dos cursos de Especialização de professores em TIC´S, e do curso de especialização em Metrologia para Software. Professor de banco de Dados, Inteligência Artificial, Sistemas de Informações, Engenharia de Software, Análise e Linguagens de Programação, Produtividade e métricas de Software. Pesquisador de processos de negócio e sua ergonomia. Pesquisador de objetos de ensino visando à otimização do aprendizado. Autor do primeiro livro sobre Inteligência Artificial do Brasil. Primeiro livro de análise orientada a objetos do Brasil. Diretor Geral da FAETERJ-Rio (antigo IST-Rio). A FAETERJ-Rio tem um curso de formação de analistas de sistemas de nível superior e duas pós-graduações (TIC aplicadas para professores - Metrologia aplicada ao software), Desenvolveu e implantou novas formas de atendimento pela secretaria, implantou a utilização de metodologias dinâmicas de ensino nas salas híbridas. Desenvolveu a biblioteca virtual em Q-RCode. Desenvolvimento de planejamento participativo e utilização

Diferentes tipos de projetos necessitam de diferentes métodos de gerenciamento. A abordagem ágil é muito utilizada em projetos orientados a valor. Como vimos projetos orientados a valor geralmente são realizados por profissionais do conhecimento e possuem elevado grau de incerteza, por grande indefinição do escopo e elevado número de mudanças.

A maior parte dos conceitos e princípios ágeis surgiu com foco em projetos de desenvolvimento de software e atualmente são utilizados em diversos tipos de projetos que possuem grandes incertezas, como campanhas publicitárias, novos produtos, planejamento de orçamento e muitas outras áreas. Este livro apresenta conceitos para todo o tipo de projeto.

Rafael Dias Ribeiro

As técnicas apresentadas neste livro são fundamentais para todo o gerente de projeto. São técnicas que devem ser obrigatoriamente do conhecimento de engenheiros, economistas, pesquisadores de todas as áreas, analistas de sistemas e profissionais interessados em formas modernas de desenvolvimento e gestão de projetos.

Modernamente os projetos devem ser desenvolvidos para acelerar os resultados. São estas técnicas que são apresentadas neste livro.

Horácio da Cunha e Sousa Ribeiro

ISBN: 978-85-919102-1-2

SPIN
Educação Profissional
contato@cursospin.com.br
Av. Djalma Batista, nº. 946, sala 08 (Centro Empresarial Santo Remédio) - Vieiralves
Nossa Sra. das Graças – Manaus
Telefone (92) 3584-1966
Site: WWW.cursospin.com.br