Rafael Martins

Desenvolvimento de projetos ágeis alinhados às práticas do PMBOK

São Paulo – SP

Dezembro / 2012

Rafael Martins

Desenvolvimento de projetos ágeis alinhados às práticas do PMBOK

Trabalho apresentado ao curso
de graduação em Processamento
de dados da Faculdade de Tecnologia
de São Paulo como requisito
a conclusão do mesmo.

Orientador:

Prof. Shigueo Tomomitsu

Faculdade de Tecnologia de São Paulo – FATECSP

São Paulo – SP

Dezembro / 2012

RESUMO

Neste trabalho foi realizada uma pesquisa bibliográfica visando retratar uma possível sinergia entre o modelo tradicional de gerência de projetos e metodologias ágeis de desenvolvimento de software. Através da comparação e sobreposição dos conceitos trabalhados pelas práticas sugeridas para gerência de projetos pelo *Project Management Institute (PMI)*, em sua obra *PMBOK* e valores defendidos pelos principais autores e defensores das mais utilizadas metodologias ágeis *SCRUM* e *XP*.

Palavras-Chave: Metodologias de Desenvolvimento. PMBOK. Extreme Programming. SCRUM. Sinergia.

ABSTRACT

In this work it was performed a literature research in order to portray a possible synergy between the traditional model of project management and agile software development. By comparing and overlapping concepts suggested for project management by the Project Management Institute (PMI), in its guide PMBOK and values espoused by leading authors and defenders of the most used agile methodologies SCRUM and XP.

Keywords: Development Methodologies. PMBOK. Extreme Programming. SCRUM. Synergy.

SUMÁRIO

Introdução

Qualquer profissional com alguma experiência em gerência de projetos, ou mesmo aqueles que já trabalharam em uma equipe de desenvolvimento de algum produto, seja ela nacional ou internacional, é capaz de citar inúmeras ocasiões onde os projetos, apesar de grandes esforços de seus colaboradores, terminaram muito longe de suas estimativas inicias de escopo, tempo e custo. Parece que, apesar de todos serem cuidadosos e muitas vezes experientes na área, sempre há algo que lhes foge à vista.

Mais especificamente, quando se trata de engenharia de software e metodologias de desenvolvimento de software, a bibliografia capaz de apontar esses erros recorrentes é vasta e antiga.

Em 1975, o livro *The Mythical Man-Month* (BROOKS,1995) já retratava a realidade dura que iria ser vivida pelas gerações futuras. Frederick P. Brooks, dentre outras, é o autor da seguinte frase:

"*Adding manpower to a late software project makes it later*".[1] (BROOKS, 1995)

Esse pensamento define bem um aspecto da gerência de projetos, nos mostrando que não é por falta de vontade ou motivação da força de trabalho que a maioria dos projetos falha, e sim por falta de planejamento e controle.

Apesar de ser uma obra com mais de 30 anos de idade, é considerada muito atual, pois a situação da gerência de projetos de TI, como reportado em estudos mais recentes não tornou-se mais eficaz (THE STANDISH GROUP, 1995).

O *Chaos Report, ou "Relatório do Caos"*, como é conhecido em português, é um estudo realizado periodicamente pelo *The Standish Group* que pretende apontar a situação do desenvolvimento de projetos de software.

Alguns dos pontos básicos do estudo são:

- Identificar em quais situações (cenários) os projetos de software falham;
- Principais justificativas às falhas;
- Quais controles e fatores reduzem a probabilidade de falha de um projeto de software;

[1] Tradução: Adicionar força de trabalho a um projeto de software atrasado, o atrasa ainda mais

Segundo o *Guia PMBOK*[2] (PMI, 2008), um projeto de sucesso deve respeitar e atender três limitações: escopo, tempo e custo, necessariamente nessa ordem de precedência e importância.

Nesse estudo, com a colaboração de 365 empresas de desenvolvimento de software das mais variadas áreas de negócio, é possível verificar mais a fundo o cenário onde os projetos eram desenvolvidos.

Em 1995:

- Os Estados Unidos despendia mais do que 250 bilhões de dólares por ano em desenvolvimento de software em aproximadamente 175.000 projetos;
- O custo médio de desenvolvimento para uma empresa de grande porte era de US$2.322.000; empresa média US$1.331.000; e para uma empresa pequena era de US$434.000;

O que mostra que as empresas, independente de seus portes, investem quantias consideráveis de seus faturamentos em projetos. Tal fato é, na verdade, algo muito positivo para os profissionais que atuam nessa área, pois denota grande procura e confiança em suas profissões.

Porém, ainda segundo o relatório, os dados mais alarmantes, e desapontadores são os seguintes:

- Absurdos 31,1% dos projetos iniciados eram cancelados (não concluídos de forma alguma);
- 52,7% custaram cerca de 190% da sua estimativa inicial (orçamento);

Os projetos avaliados foram também classificados de acordo com o grau de desenvolvimento atingido e qualidade do produto ou serviço final, como sugerido a seguir.

Tipo 1 - Projeto bem sucedido: o projeto foi terminado no prazo e no custo estimado, com todas as características e funções especificadas inicialmente (escopo, tempo e custo foram atendidos);

[2] O Guia PMBOK (Project Management Body of Knowledge Guide) é um compilado de práticas tidas, por muitos profissionais, como as melhores do mercado, e seu objetivo é maximizar a probabilidade de sucesso dos projetos através de processos bem definidos.

Tipo 2 - Projeto posto em risco: o projeto foi terminado e seu produto é operacional, porém apresentou seu prazo e/ou seu custo acima do esperado, e oferece menos características e funções do que fora inicialmente especificado e contratado;

Tipo 3 - Projeto cancelado: o projeto foi cancelado em algum ponto durante o seu desenvolvimento, pois não atende às especificações de escopo, tempo e custo;

Ao contrário do que seria o mais intuitivo, as falhas em projetos seguindo a classificação anterior, não era igualmente distribuídas de acordo com o tamanho da organização, tão pouco as maiores empresas levavam vantagem na porcentagem de sucesso. Apenas 9% dos projetos em grandes empresas eram do tipo 1, enquanto que as pequenas e médias tinham uma taxa de sucesso de 28% e 16,2% respectivamente.

A fim de descobrir as causas de tantas falhas, mesmo nas grandes empresas, onde o grau de formalidade do negócio e a conseqüente presença de sólidos processos é costumeiramente superior, foram entrevistados executivos de TI. As três maiores razões apontadas foram o envolvimento do usuário (ou a falta), suporte do gerenciamento executivo ("*buy-in*" de gerentes do negócio) e requisitos claros e objetivos.

O resumo dos apontamentos pode ser visto a seguir na tabela 1.

Fatores de sucesso do projeto	**% de respostas**
1. Envolvimento do usuário	15,9%
2. Suporte do gerenciamento executivo	13,9%
3. Requisitos claros	13,0%
4. Planejamento apropriado	13,0%
5. Expectativas realistas	8,2%
6. Marcos menores	7,7%
7. Equipe competente	7,2%
8. Senso de propriedade	5,3%
9. Visão e objetivos claros	2,9%
10. Trabalho duro, equipe focada	2,4%
11. Outros	13,9%
Total	**100%**

Tabela 1: Fonte: *Chaos Report*, THE STANDISH GROUP, 1995: Fatores de sucesso do projeto.

Conforme apontado pela Tabela 1, é notável, e ao mesmo tempo curioso, que os principais fatores de sucesso não dizem respeito à equipe de desenvolvimento diretamente, e sim a sua relação com o cliente e o entendimento preciso da situação problema.

Tal resultado nos induz a crer que uma empresa que possua profissionais altamente técnicos e qualificados não garante o sucesso dos seus projetos. Aparentemente, a combinação de uma equipe competente de desenvolvimento ao envolvimento da área de negócio em geral parece representar o cenário ideal.

Das empresas pesquisadas, foram escolhidos quatro casos para estudo, representativos entre empresas pequenas, médias e grandes, e entre diversas indústrias. Conforme o critério apresentado anteriormente, duas empresas com projetos do tipo 3 e duas com projetos do tipo 1. Os projetos escolhidos foram o DMV, o CONFIRM, o HYATT e o ITAMARATI.

O projeto DMV foi iniciado pelo departamento de trânsito da Califórnia, em 1987, para revitalizar os sistemas responsáveis pelo registro de licenças. Em 1993, depois de terem sidos gastos US$ 45 milhões, o projeto foi cancelado.

O projeto CONFIRM foi um projeto iniciado em 1994 pelas empresas American Airlines, Budget Rent-A-Car, Marriot Corp. e Hilton Hotels, que, após investirem cerca de US$ 165 milhões, cancelaram seu projeto de reserva de viagens, hotéis e aluguel de veículos.

Enquanto o projeto CONFIRM falhava, o projeto de reserva HYATT do Hyatt Hotels foi muito bem sucedido. Hoje, pode-se reservar um quarto através do celular, pedir que o ônibus de cortesia passe no aeroporto e ter suas chaves esperando você no balcão, tudo isso com um prazo e custos menores que o estimado, e com funcionalidade extras.

O projeto ITAMARATI, do Banco Itamarati, também foi um grande sucesso. O seu projeto para melhoria do atendimento ao cliente teve muitos dos ingredientes chaves que contribuíram para seu fim no custo e no prazo estimados.

Levando em conta os pontos levantados pelos executivos de TI, a tabela 2 estabelece a relação ponderada entre critérios citados e sua pontuação, onde os fatores com maior frequência têm maior peso na pontuação de escala 0 a 100.

Critério de Sucesso	**Pontos**	**DMV**	**CONFIRM**	**HYATT**	**ITAMARATI**
1. Envolvimento do usuário	19	Não	Não	Sim	Sim
2. Suporte do Gerenciamento Executivo	16	Não	Sim	Sim	Sim
3. Requisitos claros	15	Não	Não	Sim	Não
4. Planejamento apropriado	11	Não	Não	Sim	Sim
5. Expectativas realistas	10	Sim	Sim	Sim	Sim
6. Marcos menores	9	Não	Não	Sim	Sim
7. Equipe competente	8	Não	Não	Sim	Sim
8. Senso de propriedade	6	Não	Não	Sim	Sim
9. Visão e objetivos claros	3	Não	Não	Sim	Sim
10. Trabalho duro, equipe focada	3	Não	Sim	Sim	Sim
Total	**100**	**10**	**29**	**100**	**85**

Tabela 2: Fonte: *Chaos Report*, THE STANDISH GROUP, 1995: Estudo de casos, Fatores de sucesso do projeto.

Conforme apontado pelo estudo de caso resumido na tabela 2, fica evidente que os projetos DMV e CONFIRM tinham pequenas chances de sucesso, enquanto que, por outro lado, o HYATT e ITAMARATI possuíam um ótimo cenário de desenvolvimento.

O principal resultado apontado pelo relatório, e daí vem o seu nome, foi que os projetos de desenvolvimento de software estavam num estado caótico e que muitos aspectos deveriam ser revisados. Após essa constatação, uma lista de fatores de sucesso foi elaborada e publicada, visando apontar um caminho que levasse a um cenário mais favorável no futuro.

Apesar dos apontamentos feitos em 1995, e o crescente interesse e consequente investimento de empresas em busca de melhores soluções, a taxa de sucesso para projetos de software em grandes empresas mostrou um pequeno e linear crescimento nos anos subsequentes. Nesse período, os projetos classificados como "bem sucedidos" apresentaram um crescimento de cerca de 1,7% ao ano, e, estimativas sugerem que atingiria a marca de 50% somente no ano de 2014 (MARASCO, 2006).

Como sugerem os dados do último relatório Chaos, estamos muito longe de atingir os previstos 50% de sucesso, e, na verdade, uma pequena regressão dessa taxa foi apresentada nos últimos anos (THE STANDISH GROUP, 2010), como ilustrado a seguir pela tabela 3.

Ano	**Sucesso**
1995	16%
2001	28%
2003	31%
2006	35%
2010	32%

Tabela 3: Fonte: *Chaos Report, THE STANDISH GROUP:* Evolução da taxa de sucesso.

Comparadas as taxas de sucesso dos anos 1995 e 2006, nota-se uma grande melhoria, uma vez que o dobro de projetos foram finalizados com sucesso.

Porém, o relatório mais atual, divulgado em 2010, aponta um grande retrocesso na evolução da gerência de projetos. Apresentando taxas de sucesso aproximadamente igual a de 2003, ou seja, com 7 anos de retrocesso, os projetos do tipo 1 (bem sucedido) representam apenas 32% do total. A grande maioria dos projetos estão classificados como tipo 2 (posto em risco) e representam 44% do total, e, os 24% restantes, são do tipo 3 (cancelados).

Tais resultados, representam uma piora considerável, pois além da porcentagem de sucesso ter diminuído, a taxa de insucesso aumentou, colocando o relatório no pior cenário geral dos últimos cinco anos e, também, com a maior taxa de insucesso da década (CREAR[3], 2010).

Os dados negativos divulgados no período compreendido entre 2006 e 2010, discutivelmente, podem ter sido distorcidos negativamente devido a crise financeira mundial que iniciou-se nos EUA na segunda metade de 2008. Porém, em contrapartida, enormes pacotes de estímulos e financiamentos a projetos foram anunciados no mundo todo por seus governantes, em uma sequência de reuniões e acordos com os mais importantes blocos econômicos para reversão de seus efeitos.

Mais especificamente, nos EUA, onde o estudo publicado pelo *The Standish Group* foi realizado, o governo iniciou um massivo programa de recuperação, com investimentos da

[3] Jim Crear, Standish Group CIO em 2010

ordem de 831[4] bilhões de dollares americanos, chamado *American Recovery and Reinvestment Act of 2009* em resposta a crise. Segundo dados oficiais do governo norte americano, os estados que receberam os maiores incentivos foram California, onde está localizado o *Silicon Valley*[5], e New York, maior centro financeiro do mundo, com a quantia de, aproximadamente, $ 35 e $17 bilhões de dollares respectivamente. (U.S. GOVERNMENT, 2012)

Outra forte contra-argumentação que pode ser citada, é a constatação do fato que as principais empresas de tecnologia americanas apresentaram uma enorme força de recuperação em 2009 no período imediatamente após a crise (NASDAQ, 2012). Tal fato, novamente, denota grande favorecimento e importância econômica do setor de Tecnologia da Informação perante o mercado tradicional. A evolução do índice *NASDAQ*[6], sumariza esse movimento pós crise financeira, sendo, inclusive, o único índice de grande repercussão mundial a já ter superado os níveis anteriores ao do início da recessão.

Segundo a Gartner, maior companhia de pesquisa e aconselhamento em TI do mundo, durante a crise financeira, os investimentos em projetos inovadores como *Cloud Computing* e *Software Virtualisation* não enfrentaram retração alguma, na verdade, cresceram cerca de 45% no ano. Áreas mais tradicionais da TI, como prestação de serviços, também reportaram bom crescimento, com cerca de 10%. E, apenas, alguns setores apresentaram retração de fato, especialmente os produtores de infraestrutura e semicondutores com cerca de 15%. Os resultados publicados, no geral, descrevem um bom cenário e ótima performance perante grande retração global. (GARTNER INC, 2009 & 2010)

O cenário atual demonstra através dos grandes investimentos e crescente demanda, a valorização dos projetos de TI, e sobretudo os bem sucedidos. Porém, os dados apresentados por este trabalho de conclusão, sugerem que talvez as metodologias de desenvolvimento atuais não estejam atingindo esse objetivo, ou, ao menos, evoluindo e adaptando-se ao mesmo

[4] Valor inicial do pacote era de $ 787 bilhões e foi revisado para $ 831 bilhões

[5] Vale do Silício se refere à parte sul da Baía de São Francisco no Norte da Califórnia, nos Estados Unidos. A região é o lar de muitas das empresas maiores do mundo da tecnologia. Responsável por um terço (1/3) de todo o investimento de capital de risco nos Estados Unidos.

[6] O NASDAQ (National Association of Securities Dealers Automated Quotations, em Português Associação Nacional Corretora de Valores e Cotações Automatizadas) é uma Bolsa de valores eletrônica norte americana, constituída por um conjunto de corretores conectados por um sistema informático. Esta bolsa lista mais de 2800 ações de diferentes empresas. Dentre essas estão as mais importantes empresas de TI do mundo como: IBM, Microsoft, Google, Apple e Oracle. Porém, em sua maioria, são empresas de pequena e média capitalização. Caracteriza-se por compreender as empresas de alta tecnologia em eletrônica, informática, telecomunicações e biotecnologia.

passo que os problemas. Ainda, podemos dizer que vivenciamos uma época em que o fracasso é mais comum que o sucesso.

Objetivos

Este trabalho tem como objetivo traçar um paralelo entre metodologias de desenvolvimento de software e gerenciamento de projetos que apontem melhorias consideráveis na situação atual do caos. Mostrando sinergia entre a formalização do trabalho e planejamento aqui demonstrada pelo PMBOK, que é extremamente respeitado e conceituado no mercado, e a produtividade apresentada pelas metodologias de desenvolvimento ágil.

Metodologia

Este trabalho de conclusão de curso utiliza basicamente a modalidade de pesquisa bibliográfica, onde são consultadas obras e artigos amplamente difundidos na Engenharia de Software, Gestão de Projetos e metodologias de desenvolvimento. Adicionalmente, foi utilizado trabalho de pesquisa de campo, consultando profissionais experientes da área para levantamento de dados e opiniões, e curso de extensão universitária especializado no assunto.

Apresentação das metodologias

PMBOK - Project Management Body Of Knowlegde

O Project Management Institute (PMI), visa desenvolver e definir padrões para atingir a excelência em gerenciamento de projetos, independentemente da área de negócio a qual será aplicada (PMI, 2012).

Fundado em 1969, nos EUA, Pensilvânia, com apenas 5 membros voluntários, hoje, atinge mais de 350.000 membros em todos os continentes. Desde então, o PMI vem estabelecendo as bases para a profissão de gerente de projetos, promovendo eventos, capacitação e certificação de profissionais do mundo todo.

Sua principal publicação, o Project Management Body Of Knowledge, também conhecido como guia PMBOK, é o resultado de extensa pesquisa de campo e experimentação, onde são apontadas e sugeridas ao leitor as ditas melhores práticas do mercado. Tais práticas, são definidas através de modelos de processos, artifícios e métricas que são, na verdade, um grande compilado de técnicas amplamente utilizadas no mercado, e que, comprovadamente, possuem alto grau de sucesso.

A gerencia de projetos, segundo o PMI, é:

> *"A aplicação de conhecimento, habilidades, ferramentas, e técnicas para planejar atividades para atingir os requisitos do projeto".*

O objetivo do PMBOK, como está explícito em seu próprio texto é:

> "*Identificar o subconjunto do conjunto de conhecimento em gerência de projetos que é amplamente reconhecido como boa prática. Identificar significa fornecer uma visão geral, e não uma descrição completa. Amplamente reconhecido significa que o conhecimento aqui exposto e as práticas descritas são aplicáveis à maioria dos projetos na maior parte do tempo, e que existe um consenso geral em relação ao seu valor e sua utilidade. Boa prática significa que existe um consenso geral de que a aplicação correta dessas habilidades, ferramentas e técnicas podem aumentar as chances de sucesso em uma ampla gama de projetos diferentes. Ser uma boa prática não implica que o conhecimento descrito deverá sempre ser aplicado uniformemente em todos os projetos; a equipe de gerenciamento de projetos é responsável por determinar o que é adequado para um projeto específico." (PMI, 2004)*

Para tanto, ainda segundo o guia, é preciso seguir e aplicar de forma adequada os grupos de processos sugeridos nele, que totalizam 42 processos e são divididos em: Iniciação, Planejamento, Execução, Monitoramento e controle, e Encerramento.

A figura 1 mostra a interação entre os 5 grupos de processos ao longo do projeto, onde a curva mais acentuada denota maior importância do grupo para aquele período.

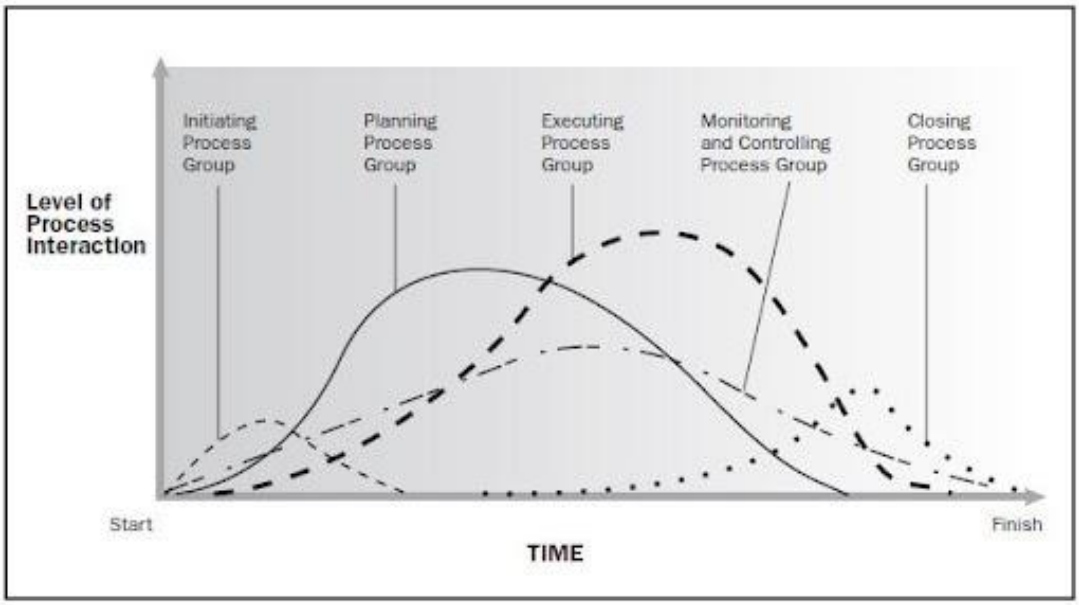

Figura 1: *PMBOK 4th edition, PMI, 2004:* Iteração entre os grupos de processos no ciclo de vida do projeto

Grupo de processos de iniciação: Define e autoriza o projeto ou fases do projeto.

Grupo de processos de planejamento: Define e refina objetivos e curso de ação para atender os requisitos do projeto.

Grupo de processos de execução: Integra pessoas e recursos para efetivar o plano do projeto.

Grupo de processos de monitoramento e controle: Processos que regularmente medem e monitoram o progresso e identificam variações em relação ao plano, de forma que ações corretivas podem ser tomadas.

Grupo de processos de encerramento: Formaliza a aceitação do produto, serviço ou resultado de uma fase e, consequentemente, faz com que o projeto chegue ao seu fim.

Todos projetos que adotem as práticas sugeridas pelo PMBOK, devem necessariamente seguir essa mesma sequencia de iniciação, desenvolvimento e finalização de cada grupo de processos, independentemente do tipo de projeto sendo desenvolvido e da área de negócio a qual os mesmo estão sendo aplicados. Porém, mais especificamente, quais processos dentro de cada grupo será desenvolvido, e quais artefatos do processo serão de fato confeccionados, podem variar de acordo com o tamanho, grau de complexidade e necessidades especiais de cada projeto. Fica ao critério do gerente e sua equipe refiná-los e definir, com base em sua experiência, quais serão necessários (T. DE SOUZA, 2010).

O guia PMBOK, apesar de advogar um grande grau de formalidade apresenta certa flexibilidade em suas especificidades, e, portanto, nem todo projeto necessariamente terá de passar pelos 42 processos propostos.

Divisão do conhecimento

Todo o conhecimento apresentado no guia, incluindo cada um dos processos, está dividido em 9 áreas (PMI, 2004):

- **Gerenciamento da integração do projeto:**
  Núcleo do gerenciamento do projeto, pois descreve processos requeridos para certificar-se que os vários elementos do projeto estão propriamente coordenados e alinhados. Este grupo de processos é diferenciado dos demais, pois, é o único que tem

duração igual ao do projeto, marcando o início oficial do projeto e também seu término.

- **Gerenciamento do escopo do projeto:**
  Descreve os processos necessários para definir, especificar, controlar, e documentar os requisitos do produto ou serviço que será entregue ao final do projeto junto às partes envolvidas e suas mudanças ao longo do seu desenvolvimento. Deve definir um produto ou serviço que atenda aos critérios de aceitação definidos pelo grupo anterior. Seu entregável mais notável é a Estrutura Analítica de Projetos (EAP).

- **Gerenciamento de tempo do projeto:**
  Descreve os processos requeridos para garantir que o projeto seja completado dentro do prazo. Define quais atividades, sua sequência e duração, relacionamentos e responsáveis que irão realizá-las para produzir os entregáveis definidos anteriormente nos processos do escopo. Seu entregável mais notável é o Cronograma.

- **Gerenciamento dos custos do projeto:**
  Descreve os processos requeridos para que o projeto seja completado dentro do orçamento aprovado. Estima, determina e controla os custos do desenvolvimento de cada atividade envolvida no projeto, fluxo de caixa, incluindo condições de mercado e organizacional, recursos humanos e tempo disponível. Seu entregável mais notável é o Orçamento.

- **Gerenciamento da qualidade do projeto:**
  Descreve os processos requeridos para garantir que o produto ou serviço final vá satisfazer as necessidades pelas quais foi requerido.
  Estabelece requisitos e padrões de aceitabilidade dos entregáveis, visando atender às necessidades e expectativas dos *stakeholders* e documenta como o projeto planeja alcançá-los (Plano de qualidade).
  Conceitos como prevenção de qualidade (invés de inspeção) e o melhoramento contínuo (*plan-do-check-act*) são fundamentais para esse grupo de processos.

- **Gerenciamento dos recursos humanos do projeto**

  Descreve os processos requeridos para fazer o uso mais efetivo das pessoas envolvidas no projeto, organizando, gerenciando e liderando-as.

  Esse grupo de processos define quais os perfis de profissionais e qualidades interpessoais são desejáveis na equipe, também, deve ser definida quais as responsabilidades de cada membro da equipe.

- **Gerenciamento das comunicações do projeto**

  Descreve os processos requeridos para garantir rápida e adequada geração, coleção, disseminação, armazenamento e disposição final das informações do projeto.

  A comunicação é um meio fundamental desde o primeiro contato com os *stakeholders* até a conclusão e encerramento do projeto de sucesso. Para que um projeto consiga atingir as expectativas e necessidades das partes envolvidas, é necessário bem entendê-las e distribuí-las à equipe.

- **Gerenciamento dos riscos do projeto**

  Descreve os processos relacionados a identificar, analisar e responder aos riscos do projeto. O objetivo desse grupo de processos é aumentar a probabilidade e impacto de eventos positivos (oportunidades), aproveitando-os de maneira a maximizar os resultados do projeto, e, simultaneamente, diminuir a probabilidade e impacto dos eventos negativos.

- **Gerenciamento das aquisições do projeto**

  Descreve os processos requeridos para adquirir bens e serviços de fora da organização contratante do projeto. Esse grupo de processos envolve contratos legais e documentos entre compradores e fornecedores de produtos e serviços, incluindo a confecção com suas condições e restrições, administração do contrato, conduta das partes e o encerramento dos mesmos.

Deste modo, o PMBOK sugere forte foco em processos, e visa, em linhas gerais, estabelecer um padrão de conduta do projeto para garantir um mínimo grau de formalidade, consequente controle, e qualidade de entregas. Para tanto, cada processo

possui entradas e saídas definidas que são organizadas de forma estritamente racional e sequencial.

A tabela 4 ilustra a divisão dos processos pelas áreas de conhecimento:

Área do Conhecimento	**Grupos de Processos da Gerência de Projetos**				
	Iniciação	**Planejamento**	**Execução**	**Monitoração e Controle**	**Encerramento**
Gestão da Integração	Termo de Abertura	Plano de gerenciamento	Orientar e gerenciar a execução	Monitorar e controlar o trabalho	Encerrar o projeto ou a fase
Gestão de Escopo		Coletar os requisitos; Definir o escopo; Criar a EAP		Verificar escopo; Controlar escopo	
Gestão de Tempo		Definir atividades; Sequenciar atividades; Estimar os recursos das atividades; Estimar as durações das atividades; Desenvolver o cronograma		Controlar o cronograma	
Gestão de Custo		Estimar custos; Determinar orçamento		Controlar custos	
Gestão de Qualidade		Planejar a qualidade	Realizar a garantia da qualidade	Realizar o controle da qualidade	
Gestão de Recursos Humanos		Desenvolver o plano de recursos humanos	Mobilizar a equipe; Desenvolver a equipe;		

			Gerenciar a equipe		
Gestão da Comunicação	Identificar as partes interessadas	Planejar as comunicações	Distribuir a informação; Gerenciar as expectativas das partes interessadas	Reportar o desempenho	
Gestão de Risco		Planejar a gerência de risco; Identificar riscos; Analisar riscos qualitativamente; Analisar riscos quantitativamente; Planejar respostas a riscos		Monitorar e controlar os riscos	
Gestão de Aquisições		Planejar aquisições	Realizar aquisições	Administrar aquisições	Encerrar aquisições

Tabela 4: *PMBOK 4th edition, PMI, 2004:* Distribuição de processos pelas áreas de conhecimento.

A figura 2 apresenta os possíveis fluxos de desenvolvimento dentro do ciclo de vida do projeto, passando por todos os 5 grupos de processos e 9 áreas de conhecimento, ainda, citando seus principais artefatos e entregáveis .

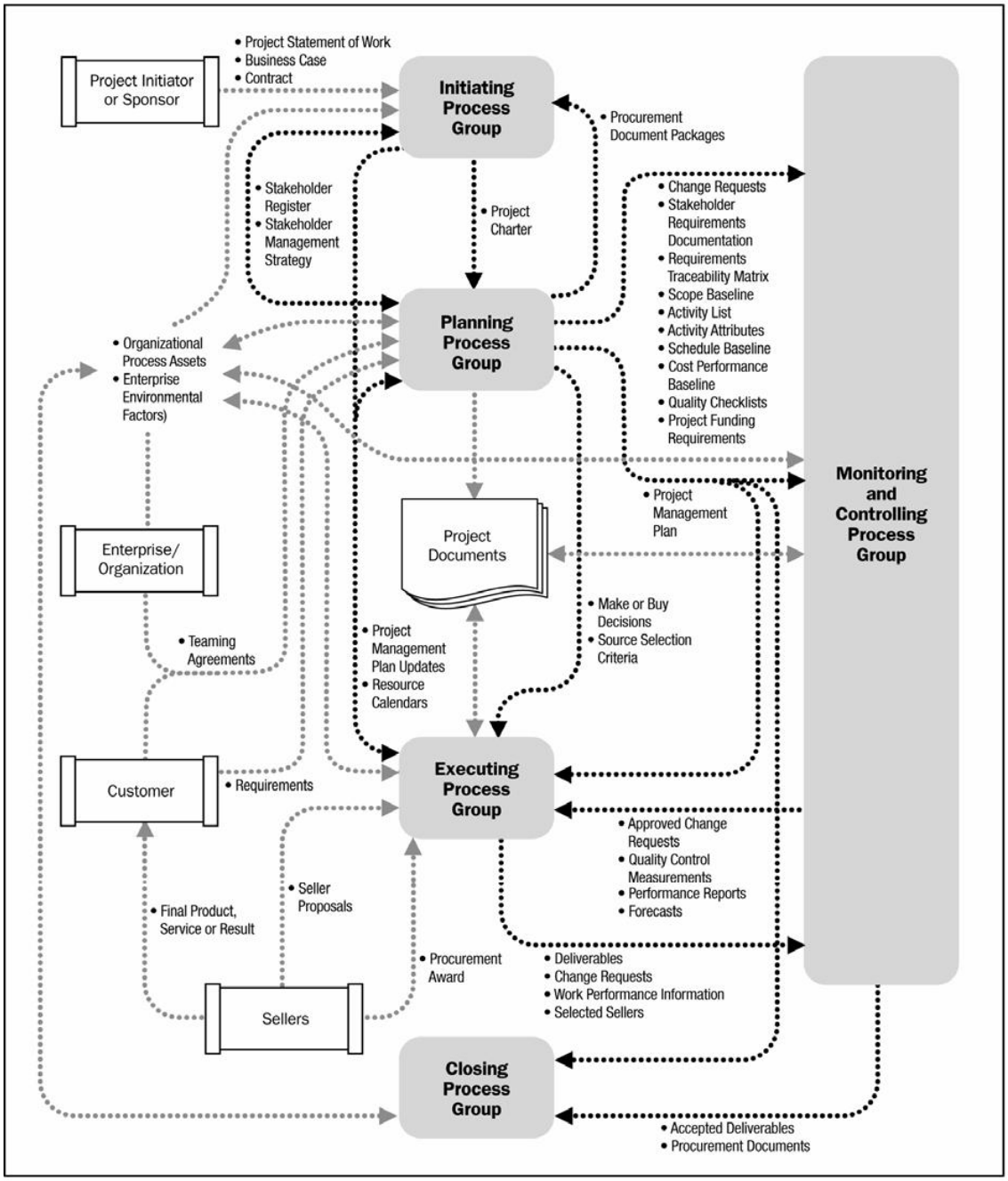

Figura 2: Fluxos de desenvolvimento dentro do ciclo de vida de um projeto (PMI, 2004)

Metodologias Ágeis

Com a falta de processos simples porém bem definidos para o desenvolvimento de projetos de software que, além de atender as necessidades dos clientes, também produzam um bem durável e de fácil manutenção. Não são raras as situações onde os profissionais envolvidos em seu desenvolvimento, acabam por se desgastarem em intermináveis horas extras e mutirões que, no final, não trazem qualidade ao produto final, e ainda, resultam em grande frustração das partes interessadas (MARTIN, 2006).

> *"Software development fails to deliver, and fails to deliver value. This failure has huge economic and human impact. We need to find a new way to develop software.*[7]*"*
>
> (BECK, 1999).

Segundo Robert C. Martin[8], na virada do milênio, esse cenário gerou uma crescente inflação de processos, cada vez mais rígidos e complexos. Essa forte e crescente tendência em revisar processos, visando cercar e evitar falhas já vivenciadas, contribuiu para grande inflexibilidade na condução de projetos de software. Algumas das grandes empresas nessa época, chegaram a de fato, iniciarem projetos com abordagens experimentais à procura de melhores metodologias para desenvolvimento de seus softwares.

Nesse cenário, um grupo de experientes desenvolvedores e consultores de software nos Estados Unidos, engajados nessa busca por processos simples e sobretudo flexíveis, reuniram-se a fim de consolidar as diferentes idéias e experiências obtidas nessas empresas em um conjunto de princípios e regras. Proclamaram-se *The Agile Alliance*[9], e o resultado dessa discussão produziu o chamado *The Manifesto of the Agile Alliance*[10].

O Manifesto da Aliança Ágil, é um conjunto de princípios e valores que definem como um projeto de software deve ser planejado, encarado, e desenvolvido a fim de ser considerado Ágil (BECK, 1999). Aborda desde os aspectos mais técnicos, como a produção de código puro, a temas mais sociais como formação e comunicação da equipe.

---

[7] Tradução: O desenvolvimento de software deixa de entregar, e deixa de entregar valor. Esta falha tem impacto econômico e humano enorme. Precisamos encontrar uma nova maneira de desenvolver software.

[8] Robert Cecil Martin, amplamente conhecido na comunidade de desenvolvimento como "Uncle Bob" (tio Bob), é um renomado consultor de software e autor Americano. Teve papel fundamental no grupo que criou a Aliança Ágil.

[9] Tradução: A Aliança Ágil.

[10] Tradução: O Manifesto da Aliança Ágil.

A base desses valores e princípios, que segundo os signatários do Manifesto Ágil, seria a solução para um excelente desenvolvimento de software, vem, na verdade, de uma combinação de ideias antigas, já existentes e consagradas, com novas, baseadas na experiência profissional de cada membro..

> *(...) Cada um tinha uma combinação diferente de velhas ideias, novas ideias, e modificações de ideias antigas. Mas todos enfatizaram uma estreita colaboração entre a equipe de programadores e especialistas em negócios, e comunicação face a face (como mais eficiente do que a documentação escrita); entregas frequentes e de valor ao negócio; rígidas equipes auto-organizadas e formas para elaborar o código de tal modo que os requisitos mutáveis não eram seriam uma crise."* (*The Agile Alliance, 2012)*

O Manifesto Ágil

> *"Manifesto para o desenvolvimento ágil de software*
>
> *Estamos descobrindo formas superiores de desenvolvimento de software através de métodos colaborativos.*
>
> *Através desse esforço chegamos aos princípios:*
>
> ***Indivíduos e interações** invés de processos e ferramentas*
>
> ***Software funcional** invés de documentação detalhada*
>
> ***Aproximação dos clientes** invés de discussões contratuais*
>
> ***Resposta à mudança** invés de seguir o planejado*
>
> *Com isso concluímos que, enquanto há valor nos termos a direita, damos muito mais valor aos termos a esquerda."*

Assinaturas:

Kent Beck; Mike Beedle; Arie van Bennekum, Alistair Cockburn; Ward Cunningham; Martin Fowler; James Grenning; Jim Highsmith; Andrew Hunt; Ron Jeffries; Jon Kern; Brian Marick; Robert C. Martin; Steve Mellor; Ken Schwaber; Jeff Sutherland e Dave Thomas.

Os princípios e valores definidos na declaração acima estão detalhados, ainda segundo os autores signatários, a seguir.

Valores

Indivíduos e interações invés de processos e ferramentas

Tom DeMarco, em sua obra *Peopleware*, 1987, ressalta a importância da relação humana no desenvolvimento de software, e, através de sua pesquisa, constatou que as pessoas são o mais importante componente do sucesso (ou fracasso).

> *"For the overwhelming majority of the bankrupt projects we studied, there was not a single technological issue to explain the failure.*[11]*"*
>
> (DEMARCO, 1999).

Resumidamente, segundo Kent Beck[12], um dos mais influentes membros signatários, a importância do indivíduo descrita pelo manifesto está na habilidade de compartilhar o conhecimento entre todos da equipe, através de comunicação e interação direta com os colegas. Esse talento, o de trabalhar bem em equipe, é mais estimado que a própria habilidade de gerar puro código, por exemplo. Ainda, é dito que, uma equipe composta somente de programadores medianos, porém que sigam essa filosofia de trabalho, pode ser muito superior a uma equipe de excelentes programadores (muito técnicos) que não a siga (BECK, 1999).

Ferramentas e processos são encarados como meros utensílios que devem ser utilizados para facilitar e viabilizar algumas tarefas. Sua importância é reconhecida, porém, nunca supervalorizada. Desta forma, nenhuma ferramenta possui a capacidade determinante de sucesso em um projeto (COCKBURN, 2000).

Software funcional invés de documentação detalhada

Robert C. Martin, em sua obra "*Agile Principles Patterns and Practices in C#*", 2006, sustenta que escrever software sem documentação é temerário. Pois, o código não é o meio ideal de comunicação para transmitir a lógica tão pouco a estrutura de um sistema.

---

[11] Tradução: Para a esmagadora maioria dos projetos fracassados que foram estudados, não havia uma única questão tecnológica para justificar o fiasco.

[12] Kent Beck é um engenheiro de software norte americano. Mundialmente reconhecido pela criação da *eXtreme Programming* (XP) e *Test Driven Development*. Beck foi um dos 17 signatários originais do Manifesto Ágil em 2001.

Ainda segundo o autor, preferencialmente, a equipe precisa de texto legível que descreva o sistema e seu desenho de forma clara e objetiva. O que é defendido por este princípio não é o extermínio de toda documentação, pelo contrário, este, valoriza a documentação a ponto de não incluir aquilo que é desnecessário, resultando, assim, na produção de documentação mínima, com alto nível de abstração, legível e clara.

Para fazê-lo Martin indica a confecção de artifícios que delimitem o sistema e, de forma geral, mostrem sua estrutura e principais módulos, sem a necessidade de entrar em especificidades. De modo que, ao longo do ciclo de vida de um projeto, quando mudanças forem implementadas devido a volatilidade ou mutabilidade de requisitos, não será necessário refazer incontáveis páginas de documentação, apenas, para mantê-las atualizadas e consistentes.

O produto desse princípio é um mapa do sistema que pode ser utilizado para apresentação do software, por exemplo, para um novo membro da equipe, e, rapidamente, introduzi-lo ao contexto trabalhado ou mesmo para um interessado da área de negócio.

Toda especificidade e detalhamento deve ser transmitido diretamente de um membro da equipe para o outro, através do trabalho em conjunto e interações cotidianas. Sendo essa, a maneira mais eficiente e precisa de imergir alguém no contexto do sistema, segundo os autores.

As duas formas de contextualização atuam em conjunto e de forma complementar, pois, possuem diferentes níveis de detalhamento.

O primeiro mandamento da documentação de software bem sucedida, segundo Robert C. Martin, e que resume a ideia por de traz desse princípio é:

> "*Produce no document unless its need is immediate and significant*[13]".
>
> (MARTIN, 2006)

Aproximação dos clientes invés de discussões contratuais

Projetos de sucesso, estatisticamente, como já citado antes nesse trabalho de graduação, são aqueles que têm o envolvimento direto dos clientes (THE STANDISH GROUP, 1995).

[13] Tradução: Não produza documento algum, a não ser que sua necessidade seja imediata e valiosa.

Kent, em sua obra *Extreme Programming Explained,* 1999, esclarece que o software não é uma *commodity*[14] e seu desenvolvimento não pode ser destacado de seu ambiente natural. Segundo Kent, a área de negócio deve estar próxima fisicamente e, ainda, com frequência participar da confecção do produto, reportando suas necessidades e expectativas além de prover feedbacks a cada entrega.

Por natureza, os requisitos de softwares são voláteis[15] (SOMMERVILLE, 2007), e por isso, caso não haja essa interação com a parte interessada, dificilmente será possível entregar um produto funcional e que atenda todas as necessidades do contratante com precisão.

O contrato é um documento legal que representa a formalização da prestação de um serviço. Deve registrar o modelo que foi acordado para a interação entre a equipe de desenvolvimento e o cliente, identificando os responsáveis e meios, nada mais. Não deve ser um registro de funcionalidades tão pouco deve especificar requisitos, custos e cronograma de entregas do projeto (BECK, 1999).

Resposta à mudança invés de seguir o planejado

A habilidade em responder a mudanças costumeiramente define as chances de fracasso e sucesso de um projeto de software. Não basta que se apresente um bom desenho e, também, profunda e precisa documentação de requisitos com colaboração do cliente, se não há flexibilidade e prontidão para adaptá-los as mudanças quando necessário.

Mesmo que os requisitos para um determinado projeto não mudem com tanta frequência, o ser humano é limitado e, por isso, tem grandes dificuldades em enxergar muito a frente e prever todos os caminhos possíveis.

A estratégia defendida por este princípio é usar diferentes níveis de detalhamento para diferentes fases do projeto. Para tanto, deve-se detalhar com precisão as tarefas imediatas (até 3 semanas), estimar de forma geral as seguintes (meses) e ter-se apenas uma ideia do que será desenvolvido após isso (ano).

Dessa forma, é possível saber exatamente o que está sendo desenvolvido e ter certeza de que aquilo é atual, afinal, é um requisito que foi detalhado há poucos dias. Essa prática

[14] *Commodity* é um termo utilizado para definir algo que possui padrão de qualidade igualitário. Ou seja, se um produto é uma commodity, sua qualidade é reconhecida, e, portanto, não há diferenciação (de qualidade) da fonte ou empresa que o produz.
[15] Sommerville, classifica os requisitos em *Enduring*(duradouro) e *Volatile* (volátil). Porém, complementa que, mesmo o primeiro tipo, é mutável e que a diferença entre eles é, na verdade, apenas o nível de mudança.

diminui muito a chance do requisito já ter sido modificado pelo cliente ou mesmo perdido o sentido no contexto do projeto (MARTIN, 2006).

Princípios

Os valores discutidos anteriormente inspiraram os seguintes 12 princípios que regem o desenvolvimento Ágil. Esses princípios, segundo os signatários do Manifesto Ágil, são os grandes diferenciais em comparação as práticas tradicionais de desenvolvimento de software.

- *Nossa maior prioridade é satisfazer o consumidor através de entregas rápidas e constantes de software funcional.*

Estudo publicado pelo MIT, *Sloan Management Review*[16], em 2001, mostra que essa é uma das práticas que mais ajudaram as grandes empresas a desenvolver software de alta qualidade. Segundo o estudo, existe uma forte correlação entre a primeira entrega e a última, onde quanto mais cedo for entregue algum protótipo, mesmo que pouco funcional, e, a partir de sua aprovação pelo cliente, for acrescido gradativamente de funcionalidades através de entregas frequentes, maior será a chance de ter-se um software de excelente qualidade ao final do desenvolvimento.

- *Mudanças de requisitos são bem-vindas, mesmo que tardias. Processos ágeis valorizam as mudanças de requisitos em prol da vantagem competitiva que fornecem aos clientes.*

A atitude e forma de encarar as mudanças são características marcantes em todo o desenvolvimento do projeto Ágil. A equipe trabalha desde o princípio para criar código que, além de funcional, é flexível, pois entende que as mudanças de requisitos virão, e, que portanto, é preciso aproveitá-las como oportunidades comerciais e não enfrentá-las como adversidades.

- *Entregar software funcional frequentemente, no intervalo de no mínimo duas semanas até o máximo de dois meses, com a preferência pelos intervalos menores.*

[16] "Product-Development Practices That Work: How Internet Companies Build Software," MIT Sloan Management Review, Winter 2001, reprint number 4226.

Como ficou evidente, os métodos ágeis têm o foco de entregas voltado a software funcional. Sendo este considerado o único entregável de verdadeiro valor ao cliente, em detrimento da produção de planos e documentações.

- *Clientes e desenvolvedores devem trabalhar juntos diariamente durante todo o projeto.*

A única maneira de desenvolver software, segundo as metodologias ditas ágeis, é manter uma constante interação entre clientes, patrocinadores e desenvolvedores, de modo a manter constante o alinhamento de interesses e expectativas.

- *Desenvolva projetos com equipe motivada. Promova o ambiente e suporte necessário, e confie que cada um fará sua parte.*

Ênfase nas pessoas, pois elas são o maior fator de sucesso. Todos os demais fatores, tais como, ambiental e gerencial, são subordinados e, caso estejam impactando negativamente as pessoas, devem ser adaptados.

- *O mais eficaz e também eficiente método de transmitir informação para os membros da equipe e entre eles é a conversa face a face.*

Em uma equipe Ágil de desenvolvimento, as pessoas conversam umas com as outras, pois a forma primária de comunicação é a interação humana. Documentos são confeccionados somente se, e, a medida que, se fazem necessários.

- *Software funcional é a forma básica para mensurar o progresso.*

Para rastrear o progresso do projeto, a equipe Ágil, considera quanto software, que foi de fato entregue, atende as necessidades do cliente. Não há o conceito de fase de projeto baseado em progresso de cronogramas, planos de entregas ou documentos produzidos para esse fim. Um projeto, por portanto, está metade terminado quando, e somente quando, metade das funcionalidades requeridas estão funcionais e entregues.

- *Processos ágeis promovem desenvolvimento sustentável de software. Os patrocinadores, desenvolvedores, e usuários deveriam manter o mesmo ritmo até o fim.*

Esse princípio prega que o desenvolvimento tenha, desde o seu início, um ritmo definido de entregas que, apesar de rápidas, são sustentáveis até o fim. Dessa forma, evitamos que grandes esforços sejam concentrados em algumas entregas, desgastando todos os envolvidos , em detrimento das demais.

- *Atenção contínua à excelência técnica e um bom desenho de software promovem agilidade.*

Alta qualidade é a chave para a agilidade. A maneira mais ágil de desenvolver é manter o código o mais claro e robusto possível, portanto, a equipe não escreve código de baixa qualidade que deverá ser revisto, mesmo que isso lhe tome um pouco mais de tempo.

- *Simplicidade, a arte de maximizar a quantidade de trabalho que não é feito, é essencial.*

Manter-se focado em resolver os problemas atuais de forma pontual, com o melhor desenho de software possível, sem tentar prever todas as mudanças e problemas futuros. Se, de fato, a solução pontual for implementada da forma correta, ou seja, com uma boa técnica, as futuras mudanças serão ágeis.

- *As melhores arquiteturas, requisitos, e desenhos de software surgem das equipes auto-organizadas.*

Uma equipe ágil é uma equipe auto-organizada. As responsabilidades e tarefas não devem ser atribuídas aos membros da equipe por pessoas de fora da equipe, e sim para a equipe. Assim, após discutir sobre essas demandas, a própria equipe irá definir quais membros serão responsável por cada uma delas segundo os seus critérios.

- *Em intervalos regulares, a equipe reflete como se tornar mais efetiva, depois refina e ajusta seu comportamento de acordo.*

Uma vez que o ambiente e os requisitos estão sempre mudando, para manter-se ágil, é necessário que a equipe esteja em constante adaptação, buscando a reciclagem de regras, convenções e relacionamentos.

Extreme Programming

Atualmente, existem várias metodologias Ágeis para desenvolvimento de software, que na verdade, são diferentes interpretações e aplicações dos princípios e valores definidos pela Aliança Ágil em 2001. É comum, ainda, as combinações de diferentes metodologias, e também adaptações das mesmas, afim de atender as necessidades específicas das empresas que as implementam (SCHWABER & BEEDLE, 2002).

A Programação Extrema, *eXtreme Programming*, ou, simplesmente, *XP*, foi concebida nos EUA no final da década de 90 por Kent Beck, engenheiro de software, durante seu trabalho no projeto C3 (acrônimo para *Chrysler Comprehensive Compensation*).

Segundo o próprio Beck, a concepção do XP ocorreu da seguinte forma:

> *"A primeira vez que fui designado para liderar uma equipe, pedi a todos que fizessem algumas coisas que acreditava serem sensatas, tais como, testes e revisões. Já na segunda vez, havia muito mais em minha mente. Pensei: se não der certo, pelo menos deve dar um bom artigo, e, pedi a equipe que elevasse a décima potência tudo aquilo que eu acreditava ser essencial, deixando de lado todo o resto."* (BECK, 1999)

Segundo Cunningham[17], grande colaborador de Beck nesse projeto, após o amadurecimento das ideias, disseminação pelo mundo e anos de aplicação, hoje, o XP ainda evolui.

É uma metodologia indicada para equipes de desenvolvimento de software de pequeno a médio porte que, com frequência, enfrentam projetos de sistemas que apresentam requerimentos voláteis ou mesmo vagos. Considerada leve, pois não possui processos massivos e complexos, eficiente, de baixo risco, flexível, previsível e científica. (WACK, 2000)

A ideia por de traz dessa metodologia é simples e pragmática. Baseada em senso comum entre os profissionais da área e princípios de programação considerados boas práticas de desenvolvimento, porém, levados ao extremo. O extremismo descrito aqui é, na verdade, segundo os defensores dessa metodologia, basicamente a busca pela excelência.

[17] Howard G. "Ward" Cunningham, norte americano, é Bacharel Interdiciplinar (Eng. elétrica e Ciências da computação) e um programador notável que desenvolveu o primeiro *wiki*. Também é conhecido por ser um dos pioneiros no D*esign Patterns* e *Extreme Programming.*

Podemos listar algumas dessas práticas, que possuem sintaxe e lógica similar a de código, extremadas segundo a ótica do XP, da seguinte forma:

- *Se rever código é bom, então faremos o tempo todo* (*pair programming).*

Todo código deve ser escrito por dois programadores em um único computador. Com isso é garantido a troca de experiências, a revisão constante do código e a transferência do modelo tácito do projeto.

- *Se testar é vantajoso, então todos testarão o tempo todo, até mesmo os usuários* (*unit testing* e *functional testing).*

Os programadores devem escrever testes unitários[18] continuamente, que precisam, necessariamente, rodar sem falhas para o desenvolvimento continuar. Clientes devem participar da elaboração desses cenários de testes que demonstrem que uma funcionalidade foi terminada.

- *Se desenhar padrões é bom, então faremos com que seja parte do trabalho diário de todos* (*refactoring).*

Os programadores devem reestruturar o sistema continuamente, removendo código duplicado, melhorando a comunicação, a simplicidade ou adicionando flexibilidade.

De fato, essa, é uma prática tão importante para o XP que motivou Martin Fowler[19] a publicar uma obra tratando somente este assunto. Escrever componentes que sejam flexíveis e reutilizáveis é a base para o desenvolvimento de um projeto de sucesso e sobretudo ágil. (FOWLER, 1999)

- *Se a simplicidade é vantajosa, então faremos com que o sistema atenda as demandas atuais com o desenho mais simples possível* (*the simplest thing that could possibly work).* (CUNNINGHAM, 2004)

O sistema deve ser desenhado da forma mais simples possível. Qualquer complexidade desnecessária deve ser removida no momento de sua descoberta.

[18] Teste isolado realizado em um componente (ou classe). Esse tipo de teste garante, se realizado com um cenário realista, que o objeto testado está funcional e íntegro. Além de contribuir para o desenho geral do sistema, uma vez que força o programador a deixar suas classes desacopladas umas das outras.

[19] Martin Fowler, autor inglês, renomado especialista em análise orientada a objetos, *UML*, *Design Patterns*, desenvolvimento Ágil e *XP*.

> *"The complexity that we despise is the complexity that leads to difficulty. It isn't the complexity that raises problems. There is a lot of complexity in the world. The world is complex. That complexity is beautiful. I love trying to understand how things work. But that's because there's something to be learned from mastering that complexity[20]"*
>
> (CUNNINGHAM, 2004)

- *Se a arquitetura é importante, todos devem trabalharam para definir e refiná-la sempre* (*metaphor).*

O desenvolvimento deve ser guiado por uma história simples (metáfora) conhecida e compartilhada por todos que retrata como o sistema funciona.

- *Se integração de testes é importante, então todos irão integrar o código e testá-lo várias vezes ao dia (continuous integration).*

O sistema deve ser integrado e testado muitas vezes num mesmo dia, a cada vez que uma tarefa é completada, utilizando-se os testes unitários escritos.

- *Se iterações curtas são vantajosas, então faremos iterações realmente e de fato curtas - com duração em segundos, minutos e horas, invés de semanas, meses e anos (the Planning Game).*

O planejamento do projeto se parece com um jogo de mesa, onde o cliente determina as prioridades e os analistas técnicos determinam as estimativas. Deve ser colocado um sistema em produção simples o mais rápido possível, e liberar novas versões em ciclos curtos.

Ao desenvolver um projeto segundo esses princípios, é esperado que se tenha um produto, e também todo o seu desenvolvimento, estável, previsível e flexível. Tais características, segundo a metodologia, são base e ferramentas estritamente lógicas para cercar e viabilizar um desenvolvimento de fato rápido.

Uma vez que evita-se complexidades desnecessárias, mantém-se o código funcional e extensível, tem-se regras claras de integração e testes, combinadas a entregas pequenas e

[20] Tradução: A complexidade que desprezamos é a complexidade que nos leva a dificuldades. Não é a complexidade em si que gera problemas. Há muita complexidade no mundo. O mundo é complexo. Essa complexidade é bela. Eu adoro tentar entender como as coisas funcionam. Mas isso é porque há algo a ser aprendido a partir do domínio dessa complexidade.

pontuais, por consequência, tem-se um grande controle sob as entregas, e, assim, evita-se também, que até mesmo os requisitos mais voláteis se percam antes de sua entrega (BECK, 1999).

Scrum

O Scrum está dentre os principais métodos ágeis de desenvolvimento de software, sendo utilizado, segundo pesquisa respondida por 2.570 participantes de 88 países, por aproximadamente 70% das empresas que adotam alguma metodologia ágil (VERSIONONE, 2009).

A aplicação ao desenvolvimento de produtos foi sugerida pela primeira vez em 1986 por Hirotaka Takeuchi e Ikujiro Nonaka nas obras *Harvard Business Review* e *The Knowledge Creating Company*, onde descreveram um novo método para aumentar a velocidade e flexibilidade do desenvolvimento de novos produtos. Ao fazê-lo, compararam este novo método holístico com fases que se sobrepõem a um time multifuncional de *rugby*, onde o time como um todo tenta avançar, passando a bola para trás e para frente, muito similar a um processo de realimentação ou ciclo progressivo por iterações.

Segundo os autores Schwaber e Beedle, em sua obra *Agile Software Development with SCRUM*, o objetivo do Scrum é fornecer um processo conveniente para o projeto e desenvolvimento orientados a objeto. Essa forte relação com a orientação a objetos levou os autores a apresentá-la como a definição de um *framework*[21], onde são definidas rotinas e objetos de uso comum a qualquer projeto de desenvolvimento de software, visando suportar e possibilitar aos envolvidos a elaboração de um processo ágil customizável. Esta metodologia apresenta uma abordagem empírica que aplica algumas idéias da teoria de controle de processos industriais para o desenvolvimento de softwares, reintroduzindo as idéias de flexibilidade, adaptabilidade e produtividade. O foco desta é encontrar uma forma eficiente de trabalho dos membros da equipe para produzir o software de forma flexível em um ambiente em constante mudança.

A metodologia do Scrum é baseada nos princípios e valores ágeis definidos no manifesto ágil, e, portanto, é muito semelhante ao XP. Tais semelhanças podem ser vistas na

[21] No contexto de desenvolvimento de software, um *framework* é um conjunto de bibliotecas ou classes que são essencialmente base para o desenvolvimento subjacente.

organização das equipes e na forma como os trabalhos são distribuídos nela. Usualmente, encontramos equipes pequenas, tarefas pontuais (devido aos requisitos pouco estáveis ou desconhecidos) e iterações curtas que garantem o grande volume constante e incremental de entregas.

O trabalho é todo dividido em papéis[22], sendo 3 centrais e os demais[23] auxiliares. De forma generalista podemos dizer que os papéis centrais estão de fato dedicados ao desenvolvimento ativo de todo o ciclo de vida do projeto, enquanto que, os indivíduos que desempenham papéis auxiliares podem ser encarados como meros facilitadores do mesmo, pois estão, somente, de alguma forma envolvidos e colaborando para o sucesso do projeto (SCHWABER, 2004).

Papéis centrais

O *Product Owner*[24] é a representação do cliente dentro do contexto do planejamento e desenvolvimento do projeto, portanto, é a pessoa que defenderá os interesses dos *stakeholders*[25]. É seu dever garantir que o trabalho realizado pelos demais membros da equipe sejam aderentes às necessidades da área de negócio. Para fazê-lo, este, deve formalizar os requisitos, usualmente através de *user stories*[26], priorizá-los e adicioná-los a lista de trabalhos a realizar, chamado de *product backlog*. No desenvolvimento de software e gerenciamento de produtos, um *user story* é constituído de uma ou mais sentenças simples e objetivas, em linguagem corriqueira do negócio, escrita por um usuário. O objetivo é descrever uma necessidade de forma pontual e rápida. Após descrevê-las, essa mesma pessoa, irá priorizá-las de acordo com a necessidade da área de negócio que ele representa (PICHLER, 2008). Devido a tamanha influência e importância desse papel, é necessário que este tenha grande afinidade com o negócio em questão e também com os processos específico já estabelecidos no ambiente em que são aplicados. É recomendado apenas um *Product Owner* por projeto, sendo que este pode ser membro da equipe de desenvolvimento ou um cliente do produto, porém, é altamente recomendado que não personifique também o papel de *Scrum Master* (DEAN, 2009).

[22] Do inglês *role.*
[23] O número exato de papéis auxiliares depende diretamente do ambiente e implementação em questão utilizada do SCRUM, e, é chamado originalmente, em inglês, de *Ancillary roles*.
[24] Tradução: Dono do produto
[25] *Stakeholder* é uma palavra inglesa que denota as partes interessadas no desenvolvimento de um projeto.
[26] Esta forma de descrição de requisitos é amplamente utilizada pelas metodologias ágeis.

O *Development Team* é o papel responsável pela realização do produto de software propriamente dito. Portanto, envolve toda a análise e tradução de requesitos para desenho técnicos, programação dos mesmo, testes e sua documentação. Usualmente, essa equipe é constituída de 3 a 10 profissionais polivalentes que atuam em todas as fases do desenvolvimento de maneira auto-organizada (SCHWABER, 2004).

O *Scrum Master* é um facilitador, que atua removendo impedimentos que possam prevenir ou atrasar a capacidade do time de desenvolvimento de entregar as características do sistemas previstas para o final de cada *Sprint*. Esse papel é, muitas vezes, confundido com o de um líder de equipe, coordenador ou mesmo gerente de projetos do modelo tradicional de desenvolvimento e gerência. Entretanto, o principal objetivo do *Scrum Master*, não é o de gerir pessoas, mas o de viabilizar e garantir um excelente ambiente de trabalho para que a equipe auto-organizada possa atuar de forma eficiente e aderente ao *framework* definido pelo Scrum (SCHWABER, 2004).

Papéis Auxiliares

É chamado de papel auxiliar todo aquele que não tem um papel formal definido, segundo a perspectiva do Scrum, ou regra específica de interação e frequência com os papéis centrais, mas ainda devem ser considerados pois são de grande influência ambiental.

Os *stakeholders* são um exemplo usual dessa classe de papéis, pois engloba clientes e fornecedores. Esses, são os grupos de pessoas que tornam o projeto possível, e, também, aqueles para os quais o produto de software e seus benefícios justificam sua produção. Ainda, mesmo que tenham grande importância de aspecto geral no projeto, estão diretamente envolvidos em, apenas, pequenas tarefas pontuais de cada *Sprint*. Outro exemplo bastante comum são os gerentes, uma vez que esses posam papel fundamental no controle ambiental no qual se desenvolve o projeto (SCHWABER & BEEDLE, 2002).

Artefatos

Mesmo apresentando um menor grau de formalidade, se comparado aos métodos tradicionais, o Scrum, assim como também as demais metodologias ágeis, possui um conjunto de artefatos mínimo para monitoração e controle de escopo, tempo, custo, progresso, testes e qualidade.

- *Product Backlog*, este é a lista ordenada por prioridade e maior valor de negócio de todas as características e funcionalidades entregáveis do sistema a serem ainda desenvolvidas. Sua confecção primária e manutenção, pois é um artefato dinâmico, é de responsabilidade do *Product Owner*, que conta com os demais membros da equipe, apenas, para auxiliá-lo nessas tarefas indicando estimativas de tempo e custo de desenvolvimento para cada item. O nível de detalhe na descrição dos itens é igualmente proporcional a sua importância, onde quanto maior esta for ao negócio, maior será seu detalhamento. Deste modo, segundo Schawaber, é possível manter um registro conciso, fiel às necessidades e que descreva um conjunto realista de itens que devem e, sobretudo são passíveis, da inclusão na próxima entrega. É importante que essas características do artefato sejam mantidas ao longo de todo o projeto, pois, como citado anteriormente, esse possui caráter dinâmico e, ao ser atualizado, seja por decomposição e refinamento de um item existente ou inclusão de novos, pode facilmente deixar de representar a situação atual do desenvolvimento, levando, assim, a um distanciamento do modelo à realidade. A figura 4 a seguir ilustra essas características:

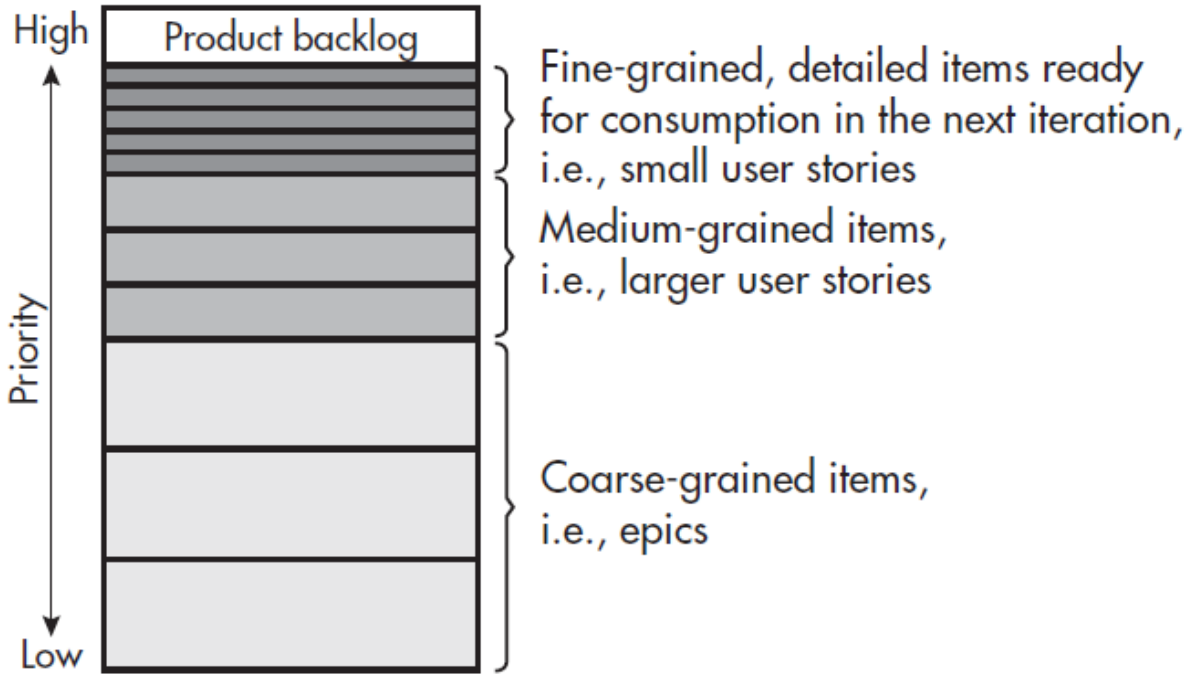

Figura 4: Esquema de organização e priorização para o *Product Backlog* como sugerido por Schawaber e Beedle (2002).

- *Sprint Backlog*, este, por sua vez, é uma seleção de itens derivada do topo do *Product Backlog*, ou ainda um subconjunto refinado deste, portanto, contém somente os requerimentos de maior valor e impacto ao negócio, que devem ser entregues o quanto antes. Possui características similares às do *Product Backlog*, pois seus objetivos primários são o delineamento de escopo e manter o controle e previsibilidade de entregas, porém difere em granularidade e detalhamento de seus itens e, também, na periodicidade de suas atualizações.

- *Burndown chart*, de forma genérica, é uma representação gráfica onde pode ser observado o progresso de um trabalho por dias decorridos, e, desta forma, obter o índice de produtividade da equipe que o realizou. Esta ferramenta é utilizada em diversas metodologias, especialmente dentre as ágeis (BURNDOWN CHART, 2012). Possui em seu eixo vertical a totalização do tempo previsto para o término de um esforço, ou um conjunto deste, usualmente em horas, e em seu eixo horizontal dias, conforme ilustrado pela figura 5.

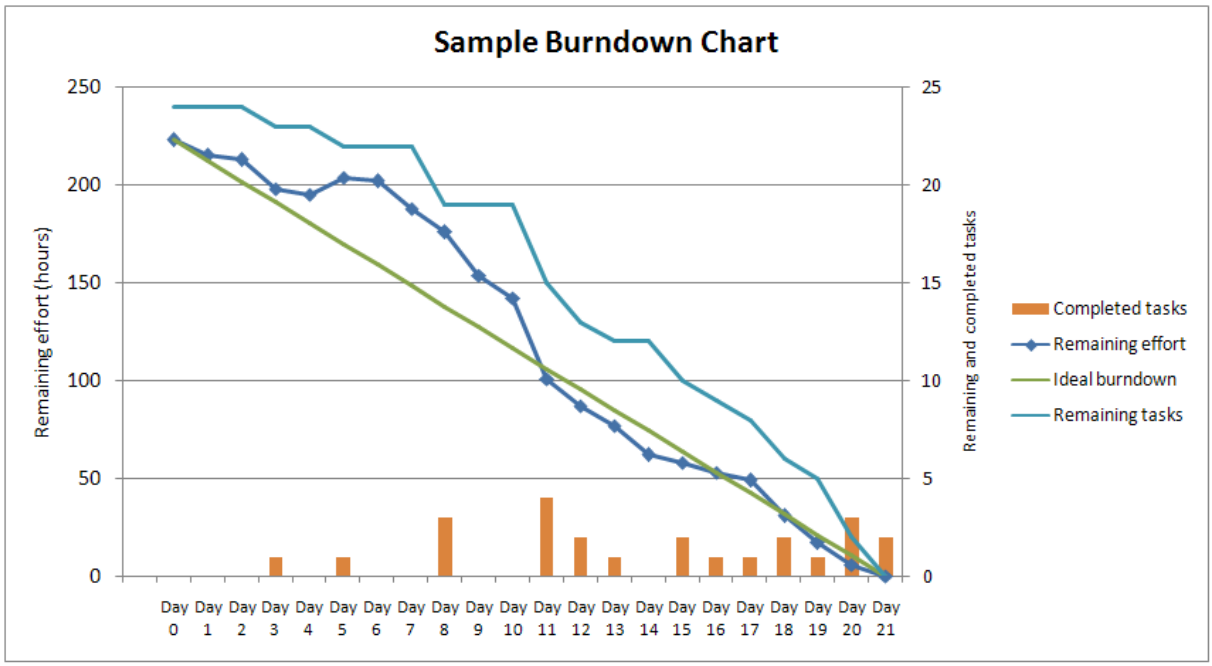

Figura 5: Exemplo de um *Burndown chart*, fonte GOOGLE DOCS, acessado em 11/11/2012.

Time-Boxes

O desenvolvimento incremental proposto pelo Scrum é dividido em estágios chamados de *Time-Boxes*[27]. Ao final de um período de aproximadamente 30 dias todos os estágios já deveram ter sido completados resultando em um, ou um conjunto de, entregável.

1. *Release Planning Meeting*[28], similar em foco ao modelo tradicional, é a fase inicial onde são estabelecidos o plano de desenvolvimento, escopo da entrega, custo e tempo estimados com forte colaboração do *Product Owner*. O processo consiste na seleção de itens de maior prioridade ou valor para o negócio do *product backlog* para confecção do *Sprint backlog* e, ao fazê-lo, a equipe procura responder a duas perguntas básicas de forma bastante prática:

    a. Como podemos tornar o vislumbrado em um produto de sucesso de maneira eficiente?

    b. Como poderíamos atender ou exceder às expectativas e retorno sob investimento aos clientes?

2. O *Sprint* é o principal estágio do Scrum e sua unidade primária de trabalho, também o mais demorado, sendo que os demais são meros suportes ou pré-requisitos para sua viabilização e validação. É uma iteração, ou ciclo de desenvolvimento, onde há de fato produção de código. Dessa forma, o que foi acordado durante a reunião de planejamento será posto em prática, ou seja, a equipe deve ao final desse *Sprint* ter todos os itens incluídos no *Sprint backlog* funcionais e entregues. A figura 6 apresenta o ciclo completo de produção de software funcional segundo o *framework* (SCHWABER & BEEDLE, 2002).

[27] Tradução: Algo que possui restrição temporal pré-definida.
[28] Tradução: Reunião de planejamento de entrega.

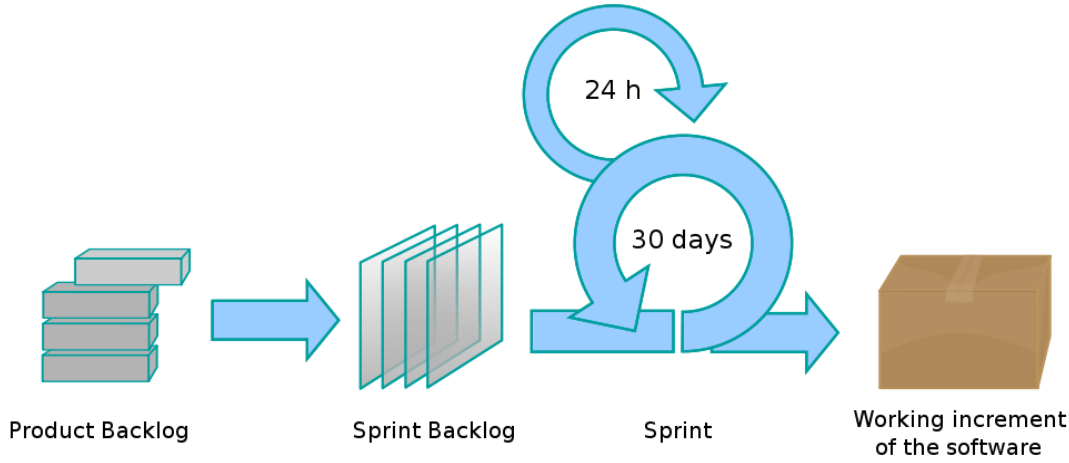

Figura 6: Processo da metodologia Scrum (SCRUM, 2008)

3. O *Sprint Review* acontece imediatamente após a conclusão do *Sprint*. É uma reunião informal com duração de até 4 horas[29] onde o progresso é revisado e possíveis lições aprendidas são registradas para refinar o processo, dessa forma, aumentando a precisão e eficiência das futuras iterações. O *Product Owner* deverá validar o trabalho produzido por esse *Sprint* e discutir com o time de desenvolvimento quais são os possíveis próximos itens do *product backlog* a serem trabalhados. Portanto, o *Sprint Review* colabora intimamente para a viabilização do próximo *Sprint* através do processo de realimentação (SCHATZ, 2009).

4. A *Sprint Retrospective* é a reunião que acontece após o *Sprint Review* e antes da próxima reunião de *Release Planning Meeting*. Nesta reunião, o *Scrum Master* encoraja o *Scrum Team* a rever, no âmbito de processos e práticas disponíveis segundo o *framework*, o seu processo de desenvolvimento para torná-lo mais eficaz e, até mesmo, agradável para o próximo *Sprint*. O objetivo primário da retrospectiva é inspecionar como correu o último Sprint no que diz respeito às pessoas, relacionamentos, processos e ferramentas utilizadas. A inspeção deve identificar e priorizar os principais itens que foram bem e aqueles itens que, se feitos de modo diferente, poderiam ter facilitado ou agilizado o processo como um todo. Estes incluem desde a composição dos *roles*, reuniões, ferramentas, definição de estados dos

[29] Sua duração é linearmente proporcional a duração do *sprint*, sendo 4 horas a duração estimada para um *sprint* de 30 dias.

entregáveis, métodos de comunicação e processos de transformação de itens do *Product Backlog* em algo tangível.

Considerações

Grandes corporações e empresas bem organizadas tendem a utilizar processos muito bem definidos e com grau de formalidade superior, pois, buscam apartar e, até mesmo, anular a dependência que os processos possam ter das pessoas que os executam, o que seria encarado como grande desvantagem estratégica. Ainda, visam melhorar a rastreabilidade, útil em auditorias e compilações de históricos, e aumentar a previsibilidade de seus futuros esforços. Por possuírem tais características, essas empresas, usualmente, optam pela abordagem tradicional de gerência e desenvolvimento de seus projetos, que apresenta um sólido conhecimento atestado por anos de estudos e colaborações dos mais experientes profissionais, para, desta forma, buscarem o sucesso. Porém, me parece que tal modelo não é tão efetivo quando aplicado a projetos de tecnologia, sobretudo ao desenvolvimento e produção de software, uma vez que, seus processos tendem a ser muito mais rígidos e até mesmo morosos. A complexidade e lentidão no desenrolar dessas práticas impacta diretamente na flexibilidade, volatilidade e dinamismo inerentes a tecnologia e, dessa forma, acaba tornando-se uma igualmente dolorosa desvantagem estratégica, pois implica em grande custo de oportunidade para os investidores e, em casos extremos, pode inviabilizar sua produção.

A tecnologia e a automação de processos vem há anos adicionando capacidades extras, agilidade e facilidades a todas as áreas de negócio e, também, as nossas vidas de forma geral. Hoje, podemos dizer que certas companhias não existiriam de forma alguma, ou mesmo faliriam instantaneamente, caso seus artefatos tecnológicos deixassem de funcionar. Por exemplo, se analisarmos um dos setores essenciais de nosso modelo econômico, o financeiro[30], é fácil perceber que todas suas

[30] O setor financeiro representa cerca de 20% do mercado nacional brasileiro, segundo o Índice IBOVESPA da bolsa de valores de São Paulo, a BVMF. Ainda, se considerarmos as 10 principais ações negociadas e as que possuem maior volume diário de negócios, o setor financeiro representa 40% desse total, sendo o holding ITAÚ UNIBANCO o segundo colocado em importância, somente atrás da PETROBRÁS (a maior empresa do Brasil). Fonte: Carteira Teórica do Ibovespa válida para 16/11/12. Disponível em: http://www.bmfbovespa.com.br/indices/ResumoCarteiraTeorica.aspx?Indice=IBOVESPA&idioma=pt-br

funcionalidades e capacidades são diretamente dependentes de sua infraestrutura tecnológica e softwares que aplicam suas regras de negócio.

Um grande reflexo dessa relação de dependência essencial é materializado em grandes centros tecnológicos com milhares de trabalhadores mantidos por estas empresas, além dos valores em seus balanços publicados trimestralmente e notas de seus representantes e setores de imprensa, onde, não é raro, observarmos investimentos de bilhões de reais em tecnologia (ITAÚ, 2012). Tal investimento representa uma fatia majoritária do lucro líquido de um ano inteiro de atividade econômica (BALANÇO ITAÚ, 2012), dedicados exclusivamente a projetos tecnológicos, tornando-se, desta maneira, virtualmente, uma empresa de tecnologia.

Desta forma, podemos concluir que, não é por falta de recursos e esforço coorporativo que a situação caótica da gerência e do desenvolvimento de projetos de tecnologia tem grande representatividade nesse setor. Talvez essas falhas sejam por estarem fazendo uso da abordagem errada, pois, seria muito natural que ao fazê-lo, mesmo que aplicada com grande empenho, resultasse em fracasso.

Em seu último estudo, publicado em 2012, o Standish Group demonstra que, projetos de software, desenvolvidos segundo metodologias ágeis têm taxa de sucesso três vezes maior se comparados às formas tradicionais.

Classificação / Abordagem	**Tradicional**	**Ágil**
Bem sucedido	14%	42%
Posto em risco	57%	49%
Cancelado	29%	9%

Tabela 5: Comparativo de desempenho entre abordagem tradicional e ágil. Fonte: The Standish Group CHAOS Report, 2012.

As metodologias de desenvolvimento ágil, segundo seus autores e renomados adeptos, como já citado nesse trabalho de graduação, não são conflitantes de forma alguma com ao modelo tradicional de desenvolvimento e gerência de projetos de software (Koch, 2011). São, de fato, apenas uma repaginação dos mesmos conceitos, porém contextualizados e aplicados especificamente ao desenvolvimento de projetos complexos e com alto grau de volatilidade, onde a constante entrega, contato direto

com o cliente e escopo de entregas extremamente reduzidos previnem esforços de desenvolvimento maiores que algumas semanas. A figura 7 ilustra essas semelhanças, onde podemos ver que ambos os modelos possuem as mesmas fases e, até mesmo, a mesma precedência e fluxo cíclico.

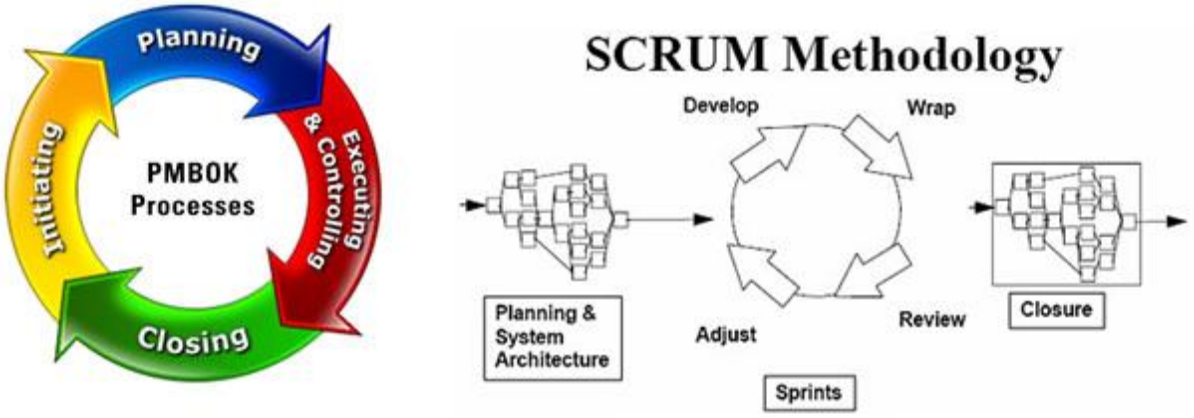

Figura 7: Comparação de fases dos modelos tradicional (PMBOK) e ágil (SCRUM). Fonte: PMI, 2012 e SCRUM, 2012.

Segundo Pressman, estima-se que caso alguma alteração tenha como custo inicial estimado de "1x" quando realizada na fase de requisitos, a mesma mudança teria um custo de "60x a 100x", se realizada durante a fase de implantação ao se valer do modelo Clássico de gerência e desenvolvimento de software (PRESSMAN, 2001). Segundo as abordagens ágeis, a mudança de requisitos é um fato corriqueiro e, ainda, não é vista como um problema (BECK, 1999). Portanto, alterações nos requisitos no modelo Clássico não são desejáveis, enquanto que no modelo ágil sim.

Conclusões

Com as evidentes necessidades das corporações em formalizar seus processos e, ainda, simultaneamente, aumentar sua velocidade e dinamismo sem comprometer a qualidade de seus produtos tecnológicos inovadores, observa-se um cenário conflitante, onde, uma possível solução, seria usufruir do alto controle provido pelo

modelo tradicional, aqui representado pelo manual PMBOK, combinado a alta produtividade e versatilidade apresentada pelos métodos ágeis.

Por características, a abordagem ágil tem o escopo de suas entregas reduzido e pontualmente focado na produção do software de fato, ou seja, em sua refinação de requisitos manifestados pelo cliente, estimativa de tempo e custo e programação dos mesmos. Portanto, não seria possível ou mesmo cabível aplicar as técnicas propostas, por exemplo, no modelo Scrum a um portfólio[31] ou mesmo a gerência de alto nível de um projeto, onde, também por natureza, o escopo é extremamente abrangente.

O modelo proposto para tentativa de aumento de sucesso no desenvolvimento de produtos de software complexos apresentado por este trabalho de conclusão utiliza-se da sobreposição de ambas as metodologias. Dos artefatos e processos sugeridos pelo PMBOK para registro e acompanhamento de requisitos funcionais e não funcionais de alto nível de abstração. Reconhecendo que se fazem necessários, sobretudo às grandes corporações. Porém arbitra a responsabilidade pelo refinamento, acompanhamento junto ao cliente e desenvolvimento factível do produto de software às práticas sugeridas pelos modelos ágeis. Desta forma, ficando com este as responsabilidades de baixo nível de abstração.

Para que haja sinergia de trabalho estre essas duas equipes dentro de um mesmo projeto, é necessário que esta divisão entre níveis de abstração seja bem definida e previamente acordada entre as mesmas.

Fica sob responsabilidade da equipe de gerência de projetos a iniciação, levantamento de requisitos de alto nível, registro de progresso global, aquisições e comunicação formal com a alta gerência da área de negócio.

Toda a responsabilidade pelo refinamento dos requisitos, consequente desenho e desenvolvimento de qualquer característica do sistema fica a cargo da equipe de abordagem ágil, que deverá atender às reuniões semanais, previstas por ambas abordagens em sua forma original, e em comum acordo com a equipe de gerência do projeto definir o escopo de suas entregas, prazos e custos previstos.

Desta forma, não se perderá de forma significativa a rastreabilidade e controles previstos e sugeridos pelo PMI sem que se adicione muita rigidez ou mesmo demora a esses processos e principalmente ao desenvolvimento de software funcional.

[31] Portfólio, segundo o PMBOK 4ª Edição, é um conjunto de projetos distintos que são independentes em sua gerência e desenvolvimento, porém possuem um fim, sob a perspectiva de seus interessados e patrocinadores, em comum.

Referências Bibliográficas

BALANÇO ITAÚ, 2012. Divulgação pública de balanços da holding Itaú Unibanco s/a. Disponível em < http://ww13.itau.com.br/PortalRI/HTML/port/infofinan/rao.shtml>. Acessado em 15 nov. 2012.

BURNDOWN CHART, 2012. Definição de Burndown Chat e seus usos. Disponível em: <http://en.wikipedia.org/wiki/Burn_down_chart>. Acesso em 08 nov. 2012

BROOKS, P. F. JR. THE MYTHICAL MAN-MONTH: ESSAYS ON SOGTWARE ENGINEERING, ANNIVERSARY EDITION. 2. ed. 1995

COCKBURN, A. AGILE SOFTWARE DEVELOPMENT: THE COORPERATIVE GAME. 2. ed. 2006.

CREAR, 2010. Jim Crear CIO of The Standish Group. 2010's Chaos Report commented. Disponível em: < http://blog.standishgroup.com/news> Acesso em: 27 jan. 2011.

CUNNINGHAM, 2004. "The Simplest Thing that Could Possibly Work : A Conversation with Ward Cunningham" Part V (19 January 2004).
Disponível em: <http://www.artima.com/intv/simplest.html>. Acessado em 10 nov. 2011.

Dean, Leffingwell. 2009. The Product Owner in the Agile Enterprise. *Agile Journal*, April 6.

DEMARCO, T., LISTER, T. PEOPEWARE: PRODUCTIVE PROJECTS AND TEAMS. 2. ed. 1999

FOWLER, M., BECK, K., BRANT, J., OPDYKE, W. ROBERTS, D., REFACTORING: IMPROVING THE DESIGN OF EXISTING CODE. 1. ed. 1999

GARTNER INC, 2009 & 2010 GARTNER Yearly Market Research. Disponível em: <http://www.gartner.com/technology/research/> Acesso em: 25 ago. 2011

KOCH, 2011. Alan S. Koch PMI, Lecture on flexible management at PMI Institute of PittsBurg. Disponível em: <http://www.pittsburghpmi.org/documents/meetings/presentations/AgileManifesto-PMBOK.pdf> Acesso em: 10 nov. 2011.

ITAÚ, 2012. Itaú Unibanco anuncia investimentos de R$ 10,4 bilhões em tecnologia. Disponível em < http://www.itau.com.br/imprensa/releases/itau-unibanco-anuncia-investimentos-de-r-104-bilhoes-em-tecnologia.html>. Acessado em 15 nov. 2012.

MANIFESTO for Agile Software Development. 2001. Disponível em: <http://agilemanifesto.org>. Acesso em: 05 nov. 2011.

MARTIN, R. C. AGILE PRINCIPLES, PATTERNS, AND PRATICES IN C#. 1. ed. 2006.
MARASCO, J. "Software development productivity and project success rates: Are we attacking the right problem?" 2006. Disponível em: <http://www.ibm.com/developerworks/rational/library/feb06/marasco/index.html> Acessado em 12 out. 2011.

NASDAQ, 2012. Composite Index progression. Disponível em: < http://www.nasdaq.com/markets/indices/major-indices.aspx> Acesso em: 30 nov. 2011

PICHLER, Roman. Agile Product Management with Scrum. Creating Products that Customers Love, 2010.

PMI, 2008. A GUIDE TO THE PROJECT MANAGEMENT BODY OF KNOWLEDGE. 4. ed. 2008

PROJECT MANAGEMENT INSTITUE, 2011 . Disponível em: <http://www.pmi.org/en/About-Us/About-Us-What-is-PMI.aspx> Acessado em 12 out. 2011

SCRUM, 2012. Disponível em: < http://scrumalliance.org/>. Acesso em: 08 nov. 2012.

SCHWABER, K. Agile Project Management with Scrum. 1. ed. [S.l.]: Microsoft Press, 2004.

SCHWABER & BEEDLE, 2002. Schwaber, Ken, and Mike Beedle. 2002. Agile Software Development with SCRUM. Prentice Hall.

SCHATZ, 2009. Schatz, Bob. 2009. The Sprint Review: Mastering the Art of Feedback. Disponível em: <www.scrumalliance.org/articles/124-the-sprintreview-mastering-the-art-of-feedback>. Acessado em 08 nov. 2012.

T. DE SOUZA, 2010. Prof. Rosenildo T. de Souza, PMP - Notas de aula do curso de Gestão de Projetos alinhados as práticas do PMBOK 4ª Edição, Escola SENAC-SP, 2010

THE STANDISH GROUP and CHAOS REPORT, 2011.
Disponível em: <http://www.projectsmart.co.uk/docs/chaos-report.pdf >
Disponível em: <http://www1.standishgroup.com/newsroom/chaos_2009.php >
Disponível em: <http://www.sdtimes.com/content/article.aspx?ArticleID=30247>
Acessado em: 22 ago. 2011

U.S Government, 2012. American Recovery and Reinvestment Act of 2009. Disponível em: < http://www.recovery.gov/Pages/default.aspx> Acesso em: 15 ago. 2011 <http://www.gpo.gov/fdsys/pkg/PLAW-111publ5/pdf/PLAW-111publ5.pdf> Acesso em: 15 ago. 2011

WAKE, W. C. Extreme Programming Explored. 1. ed. [S.l.]: Addison-Wesley Professional, 2001. SOMMERVILLE, I. SOFTWARE ENGINEERING. 9. ed. 2010