FACULDADE DE TECNOLOGIA DE SÃO PAULO

GERÊNCIA DE RISCOS APLICADA A METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO

SÃO PAULO – SP

2013

FACULDADE DE TECNOLOGIA DE SÃO PAULO

GERÊNCIA DE RISCOS APLICADA A METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO

Taiguara Maran Vieira da Silva

Monografia apresentada à Faculdade de Tecnologia de São Paulo para a obtenção do Grau de Tecnólogo em Processamento de Dados

Orientadora: Professora Sandra Harumi Tanaka

SÃO PAULO – SP

2013

Dedico este trabalho a todos que contribuíram, mesmo que indiretamente em minha formação acadêmica.

Agradecimentos

À minha orientadora, Profª Sandra Harumi Tanaka, fico extremamente grato pela sabedoria, atenção e paciência destinadas a mim nesse período tão intenso da minha vida.

A todos meus amigos e familiares que sempre estiveram ao meu lado.

Ao meu grande amigo Ualace pelas aulas de cálculo.

Por fim, gostaria de agradecer a minha namorada por ter me dado todo apoio que precisei durante minha longa jornada na FATEC.

Sumário.

Lista de figuras

Curriculum Vitae

Taiguara Maran Vieira da Silva, nascido em 21 de Setembro de 1988, em Caieiras, estado de São Paulo, está cursando Processamento de Dados pela Faculdade de Tecnologia de São Paulo. Trabalha há três anos em um dos maiores bancos mundiais, tendo atuado a maior parte do tempo como analista de risco de crédito, porém atuou grande parte deste período junto aos projetos de I.T. do banco.

Resumo

A utilização das metodologias ágeis está, cada vez mais, ganhando espaço no âmbito de desenvolvimento de sistemas. Em um segmento de mercado onde a inovação e o pioneirismo são artefatos estratégicos fundamentais, as metodologias tradicionais de gestão de projetos podem dificultar a competitividade por se preocupar mais nos processos e menos nos resultados em curto prazo.

O objetivo deste trabalho é apresentar estas Metodologias Ágeis de Desenvolvimento presentes no mercado e suas respectivas gestões de riscos. De modo a descrever o processo de utilização de tais metodologias e suas vantagens e desvantagens.

Diante do estudo de caso apresentado, evidenciou-se que as Metdologias Ágeis de Desenvolvimento tem restrições no que se refere a aplicações em ambientes muito voláteis.

Assim, conclui-se que tais metodologias devem ser muito bem dominadas antes de serem empregadas em um novo ambiente.

Palavras-chave: Metodologias Ágeis, Riscos.

Abstract

The use of agile methodologies is increasingly gaining space in the context of systems development. In a market segment where innovation and pioneer artifacts are key strategic, traditional methodologies of project management can hinder competitiveness through focus more on process and less on short-term results.

The aim of this paper is present these Agile Development Methods on the market and their respective risks management. It describes the process of using these methodologies and their advantages and disadvantages.

Before the case study, it became clear that Agile Software Development has restrictions with applications in highly volatile environment.

Thus, we conclude that methods should be thoroughly evaluated before being used in a new environment.

Keywords: Agile Methods, Risks

1 INTRODUÇÃO

A presente monografia trata de um trabalho de conclusão do curso de Tecnologia de Processamento de Dados.

O enfoque do estudo será a análise de metodologias ágeis para o desenvolvimento de softwares. Essa análise objetiva demonstrar os benefícios da aplicação dessas metodologias, e demonstrar sua grande preocupação com a gestão de riscos.

O mercado está cada vez mais dependente da indústria de desenvolvimento de software. No entanto, as empresas fornecedoras desses softwares não conseguem atender de forma efetiva a essas demandas, devido à complexidade, altos custos e prazos extensivos para implementação e manutenção de seus produtos e serviços.

Segundo SOMMERVILLE (2003) a melhoria no processo de gerenciamento e execução do desenvolvimento de software leva ao aperfeiçoamento da qualidade do resultado final. E com o objetivo de aprimorar o modo como os projetos de software são executados, foram desenvolvidas novas metodologias para agilizar e flexibilizar o desenvolvimento de softwares.

As empresas demandantes de softwares buscam obter esses produtos em prazos e custos cada vez mais enxutos, que podem não ser alcançados mesmo com a crescente evolução da tecnologia. Um dos motivos para a falha no cumprimento desses requisitos é a excessiva formalidade nos modelos de processos utilizados nos últimos 30 anos (LARMAN, 2004).

Segundo organizações como Gartner Group e Standish Group, que estudam o desempenho das empresas desenvolvedoras de software, somente 75% dos projetos de software alcançam os resultados esperados. As principais justificativas para o fracasso na entrega dos produtos solicitados são características de modelos rigorosos e tradicionais de desenvolvimento de software, como processos com formalização excessiva e lentidão na entrega de produtos.

O novo paradigma do desenvolvimento de software é buscar metodologias que consigam suprir as necessidades dos clientes de forma rápida e com qualidade, possibilitando as parametrizações e customizações que surgem em um ambiente de constante mudança.

E para suprir essas necessidades, surgiu a partir da década de 90 as primeiras metodologias de desenvolvimento ágil. Essas metodologias vieram em reação às metodologias

tradicionais, com o intuito de gerar resultados mais eficientes no desenvolvimento de software. As metodologias ágeis foram corroboradas com a publicação de o “Manifesto para Desenvolvimento Ágil de Software” que contém o conjunto de princípios que definem critérios para os processos de desenvolvimento ágil de sistemas.

A partir da década de 90 surgiram diversas metodologias de desenvolvimento ágil. No entanto, esse estudo buscou selecionar as principais metodologias utilizadas no mercado de desenvolvimento de software que buscam evitar, ou mitigar grande parte dos riscos de insucesso em um projeto.

- SCRUM
- KANBAN

2 RELEVÂNCIA E PROBLEMÁTICA

É de relevante importância a realização deste trabalho com o propósito de demonstrar os benefícios e a aplicabilidade das metodologias ágeis de desenvolvimento de software. Por ser um conceito mais recente, é necessário ainda ressaltar os pontos fortes que essa forma de tratativa de projetos tem a oferecer, e o grande avanço que essas tecnologias trouxeram em relação à mitigação dos riscos de um projeto.

Sua importância recai também sobre o Estudo de Caso da tentativa de um grande banco mundial aplicar a Metodologia de Desenvolvimento Ágil de software SCRUM.

3 OBJETIVOS

3.1 OBJETIVO GERAL.

Desenvolver um estudo a respeito de metodologias ágeis de desenvolvimento de software e destacar sua preocupação com o risco de insucesso de um projeto, além de, apresentar um estudo de caso.

3.2 OBJETIVOS ESPECÍFICOS.

1. Explanar sobre gerenciamento de riscos e metodologias de desenvolvimento de software tradicionais e ágeis;
2. Apresentar as principais características das metodologias de desenvolvimento ágil SCRUM, KANBAN;
3. Apresentar riscos e algumas formas de mitigá-los.
4. Apresentar um Estudo de Caso.

4 PLANO OU DELINEAMENTO DA PESQUISA

O trabalho terá início com uma pesquisa bibliográfica sobre o tema “metodologias ágeis de desenvolvimento de software”, e após isso este trabalho será realizado por meio de Estudo de Caso.

4.1 LIMITAÇÕES DO MÉTODO

O método de Estudo de Caso permite tirar apenas conclusões específicas para o Projeto de automatização da Tesouraria do banco. O conhecimento adquirido com a análise do Estudo de Caso não pode ser generalizado para outros projetos sem que sejam feitas as adaptações necessárias.

5 REVISÃO DE LITERATURA

Estamos inseridos em um mercado que preza cada vez mais pela agilidade na entrega de seus projetos, principalmente quando o produto é um software que irá trazer melhorias para os processos de negócio das Organizações, porém, sempre houve uma preocupação com os riscos de insucesso de um projeto. No entanto, antes do surgimento de metodologias ágeis de desenvolvimento de software, foram criadas diversas metodologias de gerenciamento de riscos que será o marco inicial de nosso trabalho.

5.1 GERENCIAMENTO DE RISCOS

5.1.1 DEFINIÇÃO DE RISCO

Para começar a discorrer sobre o assunto, primeiramente é preciso definir o que é risco. O padrão ISO31000 define risco com apenas cinco palavras:

Risco é “o efeito da incerteza nos objetivos”

Ela contém todas as três palavras essenciais que qualquer definição de risco deve incluir.

1. Risco trata de incerteza e pode nunca acontecer.
2. Risco importa e deve ser gerenciado pois tem um efeito.
3. Medimos seu efeito contra objetivos definidos.

5.1.2 O QUE É GERENCIAMENTO DE RISCOS?

Segundo a Universidade de Surrey, o gerenciamento de riscos garante que:

- Os objetivos têm mais probabilidades de serem alcançados;
- Situações causadoras de danos não irão ocorrer, ou têm menor probabilidade de ocorrer;
- Situações benéficas serão alcançadas ou têm maior probabilidade de serem alcançadas;
- Não é um processo de evitar os riscos.

O objetivo do Gerenciamento dos Riscos não é eliminar os riscos, mas Gerenciar os Riscos envolvidos em todas as atividades, para maximizar as oportunidades e minimizar os efeitos adversos.

Mais especificamente, a Gestão de Risco é um processo formal (de negócios) usado para identificar os riscos e oportunidades, em uma organização, estimar o impacto potencial

desses eventos e fornecer um método para tratar esses impactos, para reduzir as ameaças até um nível aceitável ou para alcançar as oportunidades.

Em sua forma básica, o processo de Gestão de Risco envolve:

- A identificação dos riscos e oportunidades;
- A medição e avaliação desses riscos, a partir de uma perspectiva da exposição atual;
- A determinação de um nível alvo (ou desejado) de exposição (apetite do risco);
- Um plano de gerenciamento (envolvendo controles, ações e retrocessos) para evoluir do estado atual para o estado alvo.

5.2 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

As Metodologias Ágeis de Desenvolvimento de Software surgiram na década de 90, para suprir as necessidades que o mercado demandava por inovação e redução de prazos para a execução de projetos. As metodologias tradicionais começaram a ser questionadas, devido ao seu alto grau de documentação.

O termo Metodologia Ágil emergiu com a publicação de um manifesto desenvolvido por dezessete especialistas da área de software. Em 2001, eles se reuniram nos Estados Unidos para convergir suas metodologias. Após este estudo, eles criaram o Manifesto for Agile Software Development, ou Manifesto Ágil que contém um conjunto de princípios que definem critérios para os processos de desenvolvimento ágil de sistemas (AMBLER, 2004).

Os dezessete participantes desse manifesto foram: 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.

Segundo BECK et al (2001), os doze princípios dos métodos ágeis são:

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”.
8. “Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente”.
9. “Contínua atenção à excelência técnica e bom design aumenta a agilidade”.
10. “Simplicidade, a arte de maximizar a quantidade de trabalho não realizado, é essencial”.
11. “As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis”.
12. “Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo”.

Com o objetivo de desenvolver melhores formas de desenvolvimento de software, o manifesto Ágil começa a valorizar BECK et al (2001):

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

Pressman (2005) define Engenharia de Software Ágil como a combinação entre filosofia e um conjunto de recomendações. A filosofia busca a satisfação plena do cliente e a rápida e incremental entrega de versões funcionais do software. Propõe ainda a formação de equipes pequenas e motivadas para a realização do trabalho, métodos informais e mínima formalização de documentação. Como complemento, as recomendações orientam para a importância da rápida entrega do software para o cliente a comunicação ativa entre os desenvolvedores e os demandantes do software.

Muitos processos ágeis de desenvolvimento de software estão surgindo no mercado e sendo utilizados. Para apresentação e discussão neste trabalho escolhemos duas metodologias, devido ao seu grande sucesso e aceitação no mercado. As metodologias ágeis escolhidas são:

- SCRUM
- KANBAN

5.2.1 SCRUM

O nome vem de mecanismos do rugby para colocar uma bola de volta no jogo: um círculo denso de pessoas e normalmente é separado pelos membros do time de rugby que brigam pela posse da bola. Esse termo foi utilizado pela primeira vez por Nonaka e Takeuchi (TAKEUSHI e NONAKA, 1986). Scrum é um framework ágil para a realização de projetos complexos. Scrum originalmente foi formalizado para projetos de desenvolvimento de software, mas funciona bem para qualquer escopo, complexo e inovador de trabalho. Um dos motivos para a abrangência de campos é a simplicidade do framework Scrum.

O Scrum destaca-se dos demais métodos ágeis pela maior ênfase dada ao gerenciamento do projeto. Reúne atividades de monitoramento e feedback, em geral, reuniões rápidas e diárias com toda a equipe, visando a identificação e correção de quaisquer deficiências e/ou impedimentos no processo de desenvolvimento (SCHWABER, 2004). O método baseia-se ainda em princípios como: equipes pequenas de, no máximo, sete pessoas; requisitos que são pouco estáveis ou desconhecidos; e iterações curtas. Divide o desenvolvimento em intervalos de tempos de no máximo, trinta dias, também chamados de Sprints.

O Scrum implementa um esqueleto iterativo e incremental por meio de três papéis principais (SCHWABER, 2004):

Product Owner: representa os interesses de todos no projeto;

Time: desenvolve as funcionalidades do produto;

ScrumMaster: garante que todos sigam as regras e práticas do Scrum, além de ser o responsável por remover os impedimentos do projeto.

Um projeto no Scrum se inicia com uma visão do produto que será desenvolvido (SCHWABER, 2004). A visão contém a lista das características do produto estabelecidas pelo cliente, além de algumas premissas e restrições. Em seguida, o Product Backlog é criado contendo a lista de todos os requisitos conhecidos. O Product Backlog é então priorizado e dividido em releases.

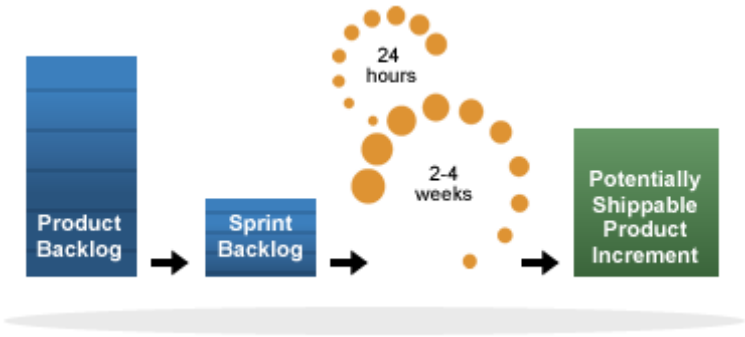

Figura 01: Esqueleto Scrum – Scrum Alliance (DUNCAN et al., 2005)

Projetos Scrum progridem em uma série de Sprints, que são iterações não maiores do que um mês. No início de um Sprint, os membros da equipe se comprometem a entregar um número de características que foram listadas no Product Backlog. No final do Sprint, esses recursos estão feitos - estão codificados, analisados e integrados no produto em desenvolvimento ou sistema. No final do Sprint há uma revisão durante a qual a equipe demonstra a nova funcionalidade para o Product Owner e outras partes interessadas que fornecem feedback que possa influenciar o próximo Sprint (COHN, 2002).

Cada Sprint inicia-se com uma reunião de planejamento (Sprint Planning Meeting), na qual o Product Owner e o Time decidem em conjunto o que deverá ser implementado (Selected Product Backlog). A reunião é dividida em duas partes. Na primeira parte (Sprint Planning 1) o Product Owner apresenta os requisitos de maior valor e prioriza aqueles que devem ser implementados. O Time então define, colaborativamente, o que poderá entrar no desenvolvimento da próxima Sprint, considerando sua capacidade de produção. Na segunda parte (Sprint Planning 2), o time planeja seu trabalho, definindo o Sprint Backlog, que são as tarefas necessárias para implementar as funcionalidades selecionadas no Product Backlog. Nas primeiras Sprints, é realizada a maioria dos trabalhos de arquitetura e de infraestrutura. A lista de tarefas pode ser modificada pelo Time ao longo da Sprint, e as tarefas podem variar entre quatro e dezesseis horas para a sua conclusão.

No final da Sprint é realizada a reunião de revisão (Sprint Review Meeting) para que o Time apresente o resultado alcançado na iteração ao Product Owner. Nesse momento, as funcionalidades são inspecionadas e adaptações do projeto podem ser realizadas. Em seguida, o ScrumMaster conduz a reunião de retrospectiva (Sprint Retrospective Meeting), com o objetivo de melhorar o processo/time e/ou produto para a próxima Sprint.

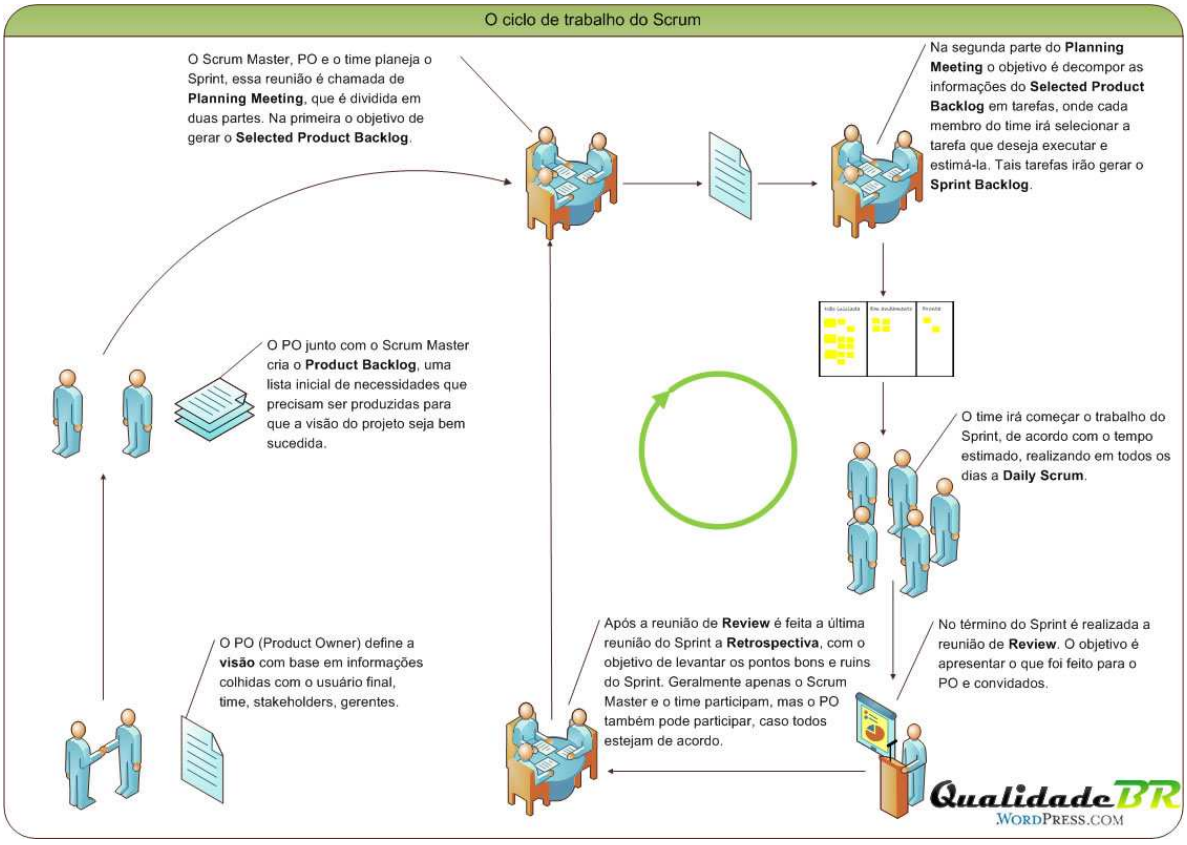

Figura 02 – Ciclo do Scrum (QUALIDADE-BR, 2010)

Scrum é permitir manter o foco na entrega do maior valor de negócio, no menor tempo possível. Isto permite a rápida e contínua inspeção do software em produção. As necessidades do negócio é que determinam as prioridades do desenvolvimento de um sistema. Todos podem ver o software real em produção, decidindo se o mesmo deve ser liberado ou continuar a ser aprimorado por mais um Sprint.

A equipe Scrum é auto-organizável, e não há nenhum líder geral da equipe que vai decidir qual pessoa vai fazer qual tarefa ou como um problema será resolvido. Essas são questões que são decididas pela equipe como um todo. A equipe é multifuncional e apoiada por dois indivíduos específicos: um Scrum Master e um Product Owner. O ScrumMaster pode ser visto como um treinador para a equipe, ajudando os membros da equipe a utilizar o framework Scrum. O Product Owner é o proprietário do produto e representa a empresa, clientes ou usuários, orientando a equipe de desenvolvimento na construção do produto correto.

O ambiente de trabalho de Scrum é extremamente importante. Ele deve ser aberto, para facilitar o diálogo da equipe e a auto-organização.

5.2.2 DEFINIÇÕES IMPORTANTES DO SCRUM

Scrum Master: o Scrum Master auxilia o time e a organização a adotarem o Scrum, educa o time, treinando-o e levando-o a ser mais produtivo e a desenvolver produtos de maior qualidade. Também ajuda o time a fazer o seu melhor em um ambiente organizacional que pode ainda não ser otimizado para o desenvolvimento de produtos complexos.

O Scrum Master deve conhecer os processos, práticas, artefatos e terminologia Scrum e saber como aplicá-los corretamente.

Sprint: é um espaço curto de tempo em que o objetivo definido deve ser alcançado de forma rápida e precisa.

Scrum Team: é a equipe de um projeto Scrum formada pelos desenvolvedores de software, é uma equipe auto-gerenciada e auto-organizada.

Product Owner: pessoa responsável pelo Product Backlog e por garantir que o projeto esteja sendo realizado de acordo com o escopo.

Daily Scrum ou Daily Meeting: é uma reunião de curta duração realizada diariamente pelo time, na qual cada um dos membros sincroniza seu trabalho e progresso e reporta quaisquer impedimentos a fim de que o Scrum Master os remova.

A Daily Scrum é sempre feita no mesmo horário e no mesmo local durante as Sprints e tem duração máxima de 15 minutos (SCHWABER e SUTHERLAND, 2004). Durante a reunião, cada membro explica o que ele realizou desde a última reunião diária, o que ele vai fazer antes da próxima reunião diária e quais obstáculos estão em seu caminho.

Sprint Planning: o Sprint Planning Meeting: é uma reunião na qual estão presentes o Product Owner, o Scrum Master e todo o time, bem como qualquer pessoa interessada que esteja representando a gerência ou o cliente. O Sprint Planning é uma reunião de um dia, com duração de 8 horas, que inicia um Sprint. A reunião é dividida em dois segmentos de 4 horas. Esses segmentos têm duração fixa, não podendo se estender (SCHWABER, 2004).

Durante o Sprint Planning, o Product Owner descreve as funcionalidades de maior prioridade para a equipe. A equipe faz perguntas durante a reunião de modo que seja capaz de quebrar as funcionalidades em tarefas técnicas, após a reunião. Essas tarefas irão dar origem ao Sprint Backlog. O Time e o Product Owner definem um objetivo para o Sprint, que é uma breve descrição daquilo que se tentará alcançar no Sprint.

Incremento de produto potencialmente entregável: é o incremento de um produto potencialmente entregável é um conjunto de realizações que poderia ser liberado para cliente.

O Product Owner toma a decisão sobre quando realmente liberar o lançamento de qualquer funcionalidade ou produto.

Product Backlog: lista de pendências do projeto organizadas por prioridades. O Product Backlog é gerenciado pelo Product Owner.

	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Figura 03: Exemplo de Product Backlog (COHN, 2002)

Sprint Backlog: é uma lista de tarefas que define o trabalho do Time por um Sprint. A Sprint Backlog é composta por itens, que são tarefas que o Time ou um membro do Time definiu como necessárias para transformar um item em uma funcionalidade do sistema.
Cada tarefa possui uma identificação daqueles responsáveis por realizá-la e o trabalho restante estimado na tarefa em qualquer dia durante o Sprint.

Retrospectiva: é uma reunião, com duração fixa de três horas, onde Scrum Master encoraja o Time a revisar, dentro do modelo de trabalho e das práticas do processo do Scrum, seu processo de desenvolvimento, de forma a torná-lo mais eficaz na próxima Sprint.

A finalidade da Retrospectiva é inspecionar como correu a última Sprint, identificar e priorizar os principais, itens que correram bem e aqueles que poderiam ser melhorados. No final da Retrospectiva da Sprint, o Time deve ter identificado medidas de melhorias factíveis que ele implementará na próxima Sprint.

User Stories: descreve uma funcionalidade que terá valor para um usuário do software. User Stories não são pertencentes ao Scrum em sua essência, mas já são tão comumente usadas com essa metodologia.

Story Points: é um método de estimativa ágil que ajuda o Time a visualizar quando uma história estará terminada. Cada Time define o tamanho do seu story point.

Estimativas: são formas de mensurar o tamanho de uma história, medida em Story Points, horas ou outra forma definida pelo Time. São realizadas pelo Scrum Team, com base em comparação entre as histórias e experiências prévias, portanto, com o passar de tempo, e com equipes mais experientes, as estimativas se tornam mais precisas.

Acceptance Criteria: em tradução literal, critérios de aceitação. São os requisitos mínimos para uma história ser considerada completa.

Velocidade: em Scrum, a velocidade é a quantidade de trabalho do Product Backlog que uma equipe pode realizar em um Sprint. Isso pode ser estimado pela visualização das Sprints anteriores, assumindo que a composição da equipe e duração Sprint são mantidas constantes.

Burndown Chart: gráfico que monitora quanto trabalho ainda falta ser executado durante um Sprint. É o principal instrumento para controle dos Sprints. Ele mostra o status das tarefas e o número de horas que faltam para terminá-las. O eixo Y do gráfico representa o número de horas que faltam para o Sprint e o eixo X o número de dias. Exemplo de Burndown Chart:

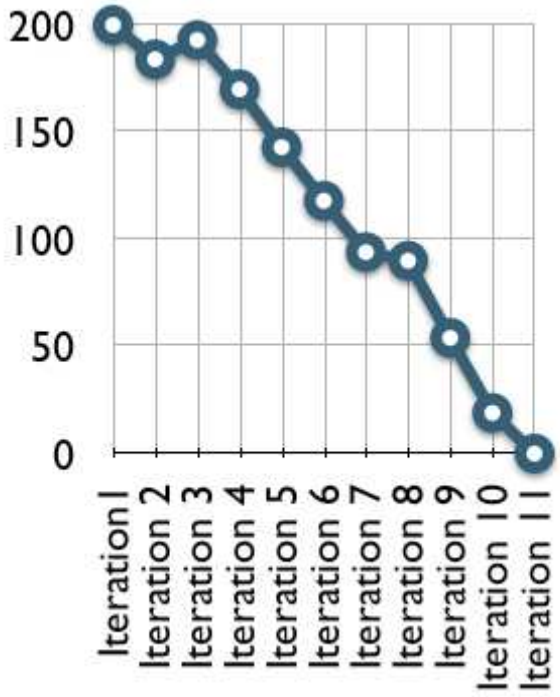

Figura 04: Exemplo de Burndown Chart (COHN, 2002)

Release: em tradução literal, release significa lançamento.

A transição de um incremento do produto potencialmente entregável da equipe de desenvolvimento em uso rotineiro pelos clientes. Releases geralmente acontecem quando um ou mais Sprints resultaram em um produto com valor suficiente para compensar o custo para implantá-lo.

Review Meeting: no final de cada Sprint uma reunião de revisão da mesma é realizada. Durante essa reunião, o Scrum Team mostra o que eles realizaram durante o Sprint. Normalmente isto toma a forma de demonstração das novas funcionalidades.

5.2.3 GERÊNCIA DE RISCOS NO SCRUM

A gerência de riscos é uma atividade importante a ser realizada no decorrer de um projeto. Ela tem o objetivo de manter o controle sobre possíveis entraves que possam ameaçar a boa execução deste. Em projetos tradicionais, a Gerência de Riscos segue um processo de identificação, análise, desenvolvimento de respostas e monitoramento dos riscos. Estas atividades são realizadas antes do início do desenvolvimento e revisadas, posteriormente, em marcos pré-definidos do projeto. Todo o processo é documentado formalmente e comunicado às partes interessadas no gerenciamento dos riscos.

O SCRUM não fala explicitamente sobre um processo formal de Gerência de Riscos, mas é possível realizá-lo sem sacrificar a agilidade desta metodologia. Sendo assim, ele pode ser realizado de maneira orgânica ou evidente. No primeiro caso, os riscos são identificados naturalmente, frutos de reuniões diárias, plannings ou revisões. No segundo caso, basta incluir a realização desta atividade, explicitamente, na pauta de alguma cerimônia do SCRUM. Seja qual for o caso, após a identificação, é necessário realizar as demais atividades do processo.

Uma boa prática é a manutenção de um quadro de riscos. A idéia por trás deste é de funcionar como o quadro de tarefas do SCRUM, porém dividido em três partes: mitigar, aceitar e evitar, que são as categorias básicas de respostas aos riscos.

Evitar um risco é a ação de eliminar a causa deste inviabilizando a sua ocorrência (ex: mudança de estratégia de projeto).

Mitigação de um risco é a suavização das consequências, ou a diminuição da probabilidade de um risco por meio de um plano de mitigação (ex: compra de um seguro).

Aceitação de um risco é a ação de simplesmente aceitar suas consequências, podendo haver ou não um plano de contingência para o caso deste ocorrer.

A medida que os riscos vão sendo identificados, eles vão sendo adicionados ao quadro, de acordo com o modo de gerenciamento escolhido para os mesmos. O mais importante é que o quadro esteja acessível para as partes interessadas e responsáveis pelo gerenciamento dos riscos.

Figura 05: Gerência de riscos (G4F 2010)

5.3 KANBAN

A história do Kanban para desenvolvimento de software, assim como a história da grande maioria das metodologias, modelos de maturidade, processos de desenvolvimento e processos em geral, começa com um grande desejo, muitas idéias, testes (na prática) e diversos ajustes (o método “científico”) até atingir os primeiros casos de sucesso.

A principal diferença do Kanban para as demais metodologias de desenvolvimento de software atuais é que ele foi um modelo adaptado de outra indústria, a de manufatura, mais especificamente da Toyota. David Anderson foi o grande responsável por essa adaptação.
A história começa em 2002, quando Anderson, cansado de ver equipes de desenvolvimento e departamentos inteiros de TI à mercê de outros departamentos, decide voltar seus esforços para responder a duas perguntas:

1. Como proteger a minha equipe da demanda incessante de negócio e alcançar o que a comunidade ágil chama de ritmo sustentável?
2. Como adotar uma abordagem ágil em toda a empresa e superar inevitáveis resistências à mudança?

Anderson tinha um grande desejo: encontrar no setor de TI uma relação “ganha-ganha” entre o departamento de negócio e as equipes de desenvolvimento de software e TI. Na tentativa de atingir esses objetivos, David idealizou, testou e falhou muitas vezes nas diversas organizações em que trabalhou. Ele notou que implantar um processo de desenvolvimento de software totalmente prescritivo, na maioria das vezes, não funcionava. Assim, chegou à conclusão que um processo precisava ser adaptado para cada situação, e que para fazer isso, era necessária uma liderança ativa em cada equipe. Porém, esta era muitas vezes inexistente. E, mesmo com uma liderança certa, ele duvidava que mudanças significativas acontecessem sem uma metodologia ou, no mínimo, sem orientações de como adaptar o processo para atender a diferentes situações.

Para David, sem um conjunto mínimo de orientações para guiar o líder, treinador ou engenheiro de processo, qualquer adaptação no processo estava susceptível a ser aplicada subjetivamente. Novos Times sempre irão resistir a mudanças se você empurrar para eles processos que foram feitos para outras realidades e que tiveram bons resultados lá. A conclusão que David chegou é que é preciso ter uma evolução com o novo Time de uma forma incremental, partindo do processo que é atualmente seguido. Isso porque: “Cada time é diferente: diferentes conjuntos de habilidades técnicas, capacidades e experiência. Cada

projeto é diferente: orçamento, cronograma, escopo e riscos diferentes. E, cada organização é diferente: o processo de produção de software é diferente em cada área de negócio."
Esse é o principal motivo pelo qual o Kanban é um framework para melhorias. Isto é, ele orienta que o processo de trabalho deve ser customizado em cada Time de cada projeto de cada organização. Ou seja, um processo não deve ter suas práticas seguidas à risca da mesma forma em todos os times, de todos os projetos, de todas as organizações do mundo, como a grande maioria das metodologias do mercado prescreve.

Em 2005, o método de trabalho de David Anderson era baseado principalmente na Teoria das Restrições (TOC – Theory of Constraints) e na FDD (Feature Driven Development). A Teoria das Restrições foi apresentada pela primeira vez em 1984 por Eliyahu M. Goldratt, no famoso livro A Meta. Segundo David Anderson , "a habilidade de identificar gargalos em um sistema é o primeiro passo para entender a Teoria das Restrições". O efeito dos sistemas puxados, ou, processo de produção puxado são tópicos que também ajudam a compreender melhor a teoria. Nele, a saída de produtos acabados, tal como o software pronto para ser usado, ao final do processo de desenvolvimento, dita o ritmo da introdução de novos requisitos no sistema. Isso evita acúmulos de produtos inacabados ao longo da linha de montagem, diminuindo a quantidade de trabalho em progresso. Já a FDD é uma famosa metodologia ágil que o próprio David ajudou a criar. Mais tarde, em 2007, após fazer algumas customizações na sua forma de trabalho inspiradas em práticas do Sistema Toyota de Produção, David apresentou nas conferências "Lean New Product Development" e "Agile 2007" os resultados preliminares do uso de Kanban na Corbis, uma empresa fundada por Bill Gates, da Microsoft.

Durante certo período, David chegou a ter dúvidas a respeito da eficiência do Sistema Toyota de Produção, mesmo com muitas pessoas falando o contrário. Porém, após conhecer um pouco mais a respeito do pensamento de Taiichi Ohno, um dos criadores de tal sistema, e a ideia por trás da cultura Kaizen, David reconheceu por meio de experiências ao longo dos cinco anos posteriores a eficiência desse sistema que originou o Kanban. "Kanban (com K maiúsculo) é um método de mudança evolutivo para monitoramento e melhoria de processos de produção, que utiliza kanban (com k minúsculo) para auxiliar na visualização do fluxo e para permitir a criação de um sistema puxado de trabalho além de outras ferramentas para catalisar a introdução de ideias Lean nas áreas de desenvolvimento de software e operações de TI. É um processo evolutivo e incremental. Kanban lhe permite atingir processos otimizados para contextos muito específicos, com resistência mínima e mantendo um ritmo sustentável para os trabalhadores envolvidos."

5.3.1 O MÉTODO KANBAN

O método kanban possui cinco propriedades centrais:

1. Visualizar o fluxo de trabalho;
2. Limitar a quantidade de trabalho em andamento;
3. Medir e otimizar o fluxo de trabalho;
4. Tornar explícitas as políticas do processo;
5. Gerenciar quantitativamente.

O Kanban não é uma metodologia, mas sim um framework para implementar mudanças de forma incremental. Esse é um conceito muito importante para que se entenda o Kanban como um todo já que, quando se fala em metodologias, fala-se em conjuntos de práticas e o Kanban não tem nenhuma prática prescrita. Há, nele, somente propriedades que devem guiar a melhoria no processo atual, não importando quais práticas estejam sendo usadas.

Ao usar o Kanban, é esperado que se consiga visualizar quais práticas estão sendo positivas e quais estão sendo negativas e isso, consequentemente, vai conduzir a mudanças, que, naturalmente adicionarão, eliminarão ou alterarão as práticas atuais de trabalho e mitigarão os risco.

A forma mais comum de se conseguir visualizar o fluxo de trabalho é pelo kanban. Esse é uma espécie de quadro, que pode ser físico, normalmente colocado em uma das paredes do local onde a equipe trabalha, ou virtual. O uso do quadro virtual tem alguns prós e contras. Os pontos fortes são basicamente a facilidade de extração de métricas e o histórico. O principal ponto fraco dessa abordagem é a dificuldade que a equipe terá para analisar e evoluir conjuntamente o processo de trabalho.

Nesse quadro, inicialmente devem ser modeladas cada uma das etapas necessárias para se produzir o software, isto é, todo o workflow de trabalho. E, em cada uma dessas etapas deve ser colocado e mantido um cartão, a fim de simbolizar um trabalho que está em andamento, como podemos ver na Figura 6. Cada um desses cartões deve conter informações detalhadas sobre a atividade, bem como quem está desenvolvendo o item, e quando o mesmo foi iniciado. No Kanban para desenvolvimento de software, o kanban deve ser mantido e evoluído por toda a equipe, o tempo todo. A proposta baseia-se em fazer a própria equipe

enxergar onde está errando e deixá-la tomar decisões seguindo um framework simples, que guiará grande parte dessas melhorias.

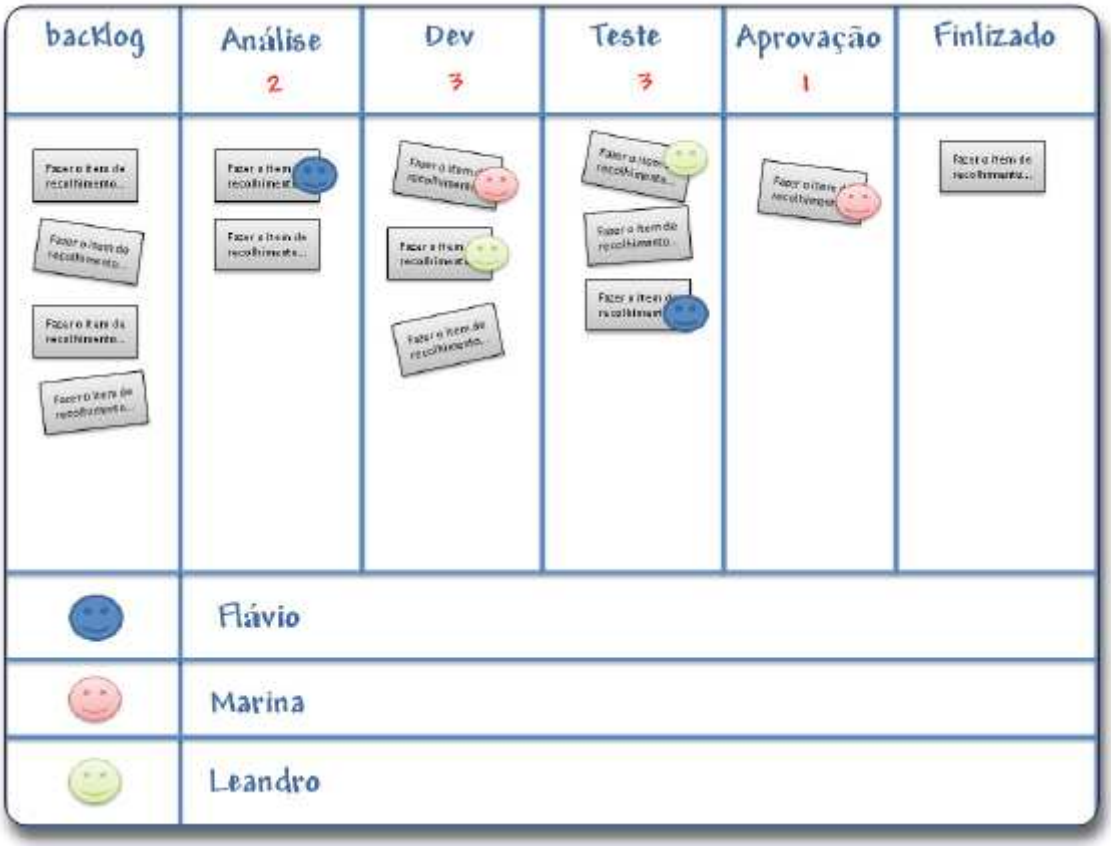

Figura 06. Ilustração de um sinalizador visual Kanban

Para Alisson Vale, especialista em metodologias ágeis de desenvolvimento, em atividades que envolvem trabalho criativo, como desenvolvimento de software, o propósito do Kanban é provocar conversações sobre o sistema de trabalho. Para limitar a quantidade de trabalho em andamento é necessário definir, monitorar e manter um limite máximo de tarefas em andamento para cada uma das etapas de trabalho mapeadas no quadro kanban, como visto na Figura 6.

Ao estabelecer os limites, a equipe começa a identificar onde estão os principais gargalos do processo de produção e começa a ter que terminar todo o trabalho inacabado na linha de produção para conseguir puxar mais trabalho para o gargalo. Com isso, o trabalho tende a ser finalizado de uma forma incremental e não de uma forma cascata, tornando-se assim um sistema puxado (TOC), onde é a equipe que dita a sua capacidade de produção. O

uso do sistema puxado, juntamente com a visualização de todo o processo de desenvolvimento por toda a equipe, permite implementar mudanças no processo de modo incremental.

Consequentemente, há redução significativa da resistência, o que facilita o alcance do ritmo sustentável, teoria tão comentada em desenvolvimento ágil, mas, para a qual poucas definições existem além das 40 horas semanais de trabalho.

Outro fator importante é o ganho que se tem ao praticamente impedir que uma mesma pessoa execute várias tarefas no mesmo projeto em paralelo. Vale destacar também o processo Just in time (JIT), que evita o acúmulo de estoque ao longo do processo de desenvolvimento, como se faz no ciclo de vida em cascata. Nesse caso, software em estoque são todos os requisitos que ainda não foram liberados para o cliente usar. No desenvolvimento de um software, como em qualquer outra atividade intelectual, por mais contra intuitivo que possa parecer, executa-se com mais qualidade e mais rapidamente duas tarefas, fazendo-as uma de cada vez, do que as duas em paralelo.

5.3.2. KANBAN, RISCOS DE UMA ADOÇÃO PREMATURA.

Jim Coplien, autor e pesquisador renomado nas áreas relacionadas a Ciência da Computação, propõe em artigo recentemente publicado no blog de Jeff Sutherland, criador do Scrum, que grande parte do mercado tem adotado práticas de kanban a partir de um entendimento equivocado dos princípios da manufatura Lean. Para Coplien, a alternativa ao Kanban é o "fluxo contínuo de uma única peça".

Coplien, que é reconhecidamente uma autoridade no contexto de desenvolvimento software, sugere que o equívoco é causado por uma má interpretação do conceito Lean, que é geralmente visto pelo mercado como uma forma superficial de se aplicar as ferramentas do Sistema de Produção Toyota, sem que haja a efetiva adoção de seus princípios essenciais.

Sendo assim, a indústria de software teria se "apropriada indevidamente" da ferramenta (kanban), sem entender que os fundamentos do fluxo devem vir em primeiro lugar.

O autor contextualiza seu argumento pela citação extraída do livro Sistema de Produção Toyota:

Kanban é uma palavra japonesa que significa apenas sinal visual. O kanban é usado para controlar o trabalho que está em andamento, no contexto de uma linha de produção onde o fluxo já foi estabelecido.

Coplien, então, é mais enfático ao apontar aquilo que acredita ser o maior risco de se adotar a ferramenta kanban como metodologia:

O kanban (a metodologia) desencoraja o trabalho em equipe e aumenta o risco de não se completar o trabalho acordado dentro de um Time-box, como um Sprint. Esta interpretação do kanban é atraente aos gestores, pois sentem a necessidade de recuperar o controle que haviam perdido com o Scrum, ou seja, uma maior facilidade para consertar as coisas paliativamente, ao invés de resolver a causa raiz dos problemas. Dá uma maior sensação de sucesso imediato sem ser necessário ponderar as consequências de longo prazo geradas por decisões de curto prazo.

Coplien sugere que o estabelecimento de um "fluxo contínuo de uma única peça" é uma solução verdadeiramente Lean, e indica que o Scrum possui todos os ingredientes necessários para o estabelecimento desse fluxo:

Em vez de depender de kanban, uma solução verdadeira Lean elimina "mura", "muri", e "muda" – termos que em japonês representam inconsistência, falta de fluxo contínuo e resíduos, respectivamente – em vez de monitorar o movimento e o processamento dos materiais. Uma boa implementação Scrum dispõe as equipes ou dispositivos que elaboram os artefatos em um mesmo local. Os fundamentos do Scrum encorajam a criação de um fluxo contínuo.

Uma boa equipe Scrum se beneficia automaticamente das vantagens do kanban quando esta já pratica o fluxo contínuo. É inerente ao Scrum a limitação do trabalho em andamento, enquanto que o trabalho em equipe é encorajado. O Scrum, dessa forma, viabiliza, aos membros da equipe, uma autonomia de duração fixa para executar seus planos.

Por fim, Coplien afirma que "tal maturidade" pode servir como base para a adição posterior de técnicas como o kanban e que tanto sua própria experiência quanto a do criador da ferramenta kanban mostram que tais técnicas irão naufragar sem práticas similares ao Scrum, que primeiramente estabelecem a disciplina do fluxo.

5.4 ESTUDO DE CASO

O Projeto de automatização da tesouraria de um grande banco mundial é um projeto que visa o desenvolvimento de um sistema único para utilização dos usuários da Tesouraria. Atualmente, o banco dispõe de quatro sistemas para atender as necessidades destes usuários. Porém, é muito comum o uso de planilhas em EXCEL, pois não há outro modo de agrupar informações de mais de um sistema, e com isso, a disparidade de informações torna-se comum no ambiente de trabalho, gerando um fluxo operacional de alto risco. Com o desenvolvimento do projeto a proposta é que aconteça a integração dos sistemas SIGMO, FOS(Front Office System), TREATS e ICR(Information of Credit Risk), e com isso, minimizar o uso de planilha, e mitigar os riscos, centralizando as informações em um único sistema.
Podemos citar algumas das vantagens do desenvolvimento deste sistema:

- Processos simples, padronizados e integrados;
- Base de dados única;
- Agilidade no atendimento ao cliente;
- Agilidade na boletagem das operações;
- Consistência dos dados;
- Serviços Comuns disponíveis para todos os produtos bancários;
- Independência funcional por segmento;
- Diminuição do risco operacional;
- Diminuição do risco reputacional;
- Agilidade na confecção de relatórios;
- Redução de custo operacional;

O projeto já tem duração de três anos, nos quais foram utilizadas algumas Metodologias de Desenvolvimento de Sistemas. Até o primeiro ano de projeto, a equipe optou por utilizar a Metodologia de Desenvolvimento Ágil, SCRUM, pois ficou decidido que seria a metodologia que melhor atenderia as necessidades devido as seguintes características:

- A entrega de produtos está acima da entrega de documentação;
- As respostas às mudanças são mais importantes que o segmento de um plano;
- Priorização da colaboração do cliente sobre a negociação de contratos;
- Os indivíduos e suas interações são mais importantes que os processos e ferramentas.

Porém alguns pontos relevantes não foram levados em conta, tais como:

- **Rotatividade de mercado:** no mercado financeiro essa prática onde profissionais aprimoram seus conhecimentos em uma grande instituição por um curto período de tempo e depois partem em busca de melhores cargos e salários é muito comum.
- **Volatilidade do Mercado:** as prioridades definidas nas SPRINTS, muitas vezes se tornavam fúteis ao longo do mês, devido a grande volatilidade do mercado financeiro.
- **Documentação:** nas Metodologias Ágeis de Desenvolvimento, a documentação geralmente é deixada um pouco de lado. Porém, nas instituições financeiras, é imprescindível para o controle de fraudes e auditorias internas e externas.

Durante os primeiros meses, o desenvolvimento parecia fluir melhor. Com foco no produto final, várias funcionalidades foram entregues, aumentando significativamente a produtividade. Porém, com o decorrer do projeto, alguns papéis, dentro do SCRUM, foram se distorcendo.

O papel do SCRUM Master que deveria ser de: proteger a equipe de riscos e interferências externas, resolver problemas de logística e conhecimentos, manter o Backlog do Sprint; não estava acontecendo. Os SCRUM Master estavam exercendo um papel de gerente de projeto, apenas controlando cronogramas, gerenciando recursos e atividades. Já o papel do P.O. (Product Owner) que seria de definição da visão do Produto, elaborar e manter o Product Backlog, validar os entregáveis, também não estava acontecendo. O Product Owner estava apenas empenhado em validar os protótipos e cobrar dos SCRUM Master os entregáveis no prazo estipulado. Parte disso deve-se a grande rotatividade de mercado, pois havia grande dificuldade em definir as necessidades junto a usuários que não estavam familiarizados com os processos.

Então, o foco era de implementar o que já havia sido definido por profissionais que não estavam mais no banco. Quando a gerência do projeto se reuniu para avaliar a utilização da Metodologia Ágil, chegaram a conclusão que os resultados iniciais eram insatisfatórios, pois a equipe excedia a jornada de trabalho para atender ao prazo, e não existia documentação de todas funcionalidades entregues, e as que existiam não retratavam ao certo o que foi desenvolvido. Com isso, foram feitas diversas reuniões a fim de buscar uma solução, visto que a Metodologia Ágil de Desenvolvimento SCRUM não atendeu as necessidades do projeto. Porém, no mesmo período, o banco estava sendo auditado pelo Banco Central do Brasil, e um dos principais pontos levantados nessa auditoria foi a falta de documentação.

Com isso, o banco optou por dispensar toda equipe de I.T. envolvida no projeto, e contratar uma consultoria especializada nas metodologias SCRUM e KANBAN para dar

andamento ao projeto. As principais mudanças foram a implementação de um quadro virtual da metodologia KANBAN, adaptado as necessidades do projeto, onde três diretores, responsáveis pelas áreas de Riscos, Trading e Products, poderiam a qualquer momento modificar as SPRINTS, com isso minimizar os efeitos da grande volatilidade do mercado financeiro, pois a cada mudança significativa no mercado, havia uma reunião onde os diretores juntamente com a equipe responsável pelo projeto definiriam por modificar ou manter a SPRINT. Outra mudança importante foi a decisão de contratar três profissionais responsáveis apenas pela documentação e com isso, adequar o projeto às normas de documentação exigidas pelo BACEN.

Ao mesclar as metodologias ágeis de desenvolvimento SCRUM e KANBAN o projeto voltou a fluir, o papel do Product Owner foi adaptado aos três diretores, os quais assinaram um termo de compromisso que durante o andamento do projeto permaneceriam no banco. Porém, mesmo assim um dos envolvidos no projeto pediu demissão, acentuando a fragilidade do Product Owner em um ambiente muito volátil.

Atualmente, o projeto encontra-se em sua fase final, apesar do início conturbado e um atraso de aproximadamente seis meses na entrega, está sendo considerado um sucesso, devido à grande aceitação por parte dos usuários.

5.5 CONCLUSÃO

Cada vez mais, as metodologias ágeis de desenvolvimento de software vêm ganhando destaque, porém, muitas equipes estão adotando essas metodologias de forma prematura, o que muitas vezes pode levar ao insucesso do projeto.

Uma das principais características da aplicação do SCRUM é a necessidade de se ter papéis bem definidos, o que não ocorreu no caso do Estudo de Caso apresentado, levando o banco a substituir a equipe do projeto. Outro fator que colaborou para o insucesso do SCRUM foi a falta de documentação, pois a existente não atendia aos requisitos do BACEN.

Atualmente, com o aumento de profissionais especializados nessas novas metodologias, é possível suprir as carências de uma Metodologia Ágil de Desenvolvimento, com outra metodologia, seja ela ágil ou não, conforme apresentado no Estudo de Caso.

A adoção dessas metodologias pode parecer simples, porém, é preciso expertise, tanto técnica quanto comportamental, para deixar o time, o cliente e a empresa preparados para a mudança de paradigmas que as metodologias trazem.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON,D. Kanban: Successfull Evolutionary Change for Your Technology Business, Goodreads, 2010.

AMBLER, S. *Modelagem ágil: práticas eficazes para a Programação Extrema e o Processo Unificado*. Porto Alegre: Bookman, 2004.

BECK, K. et al. *Agile Manifesto*. 2001.

Disponível em: <http://www.agilemanifesto.org>. Acesso em: 12/04/2013.

BOEG, J. Kanban em 10 passos. 2012.

COHN, M., *Agile Estimating and Planning*, Prentice Hall, 2006.

ISO 31000:2009, Risk management – Principles and guidelines, ISO 31000:2009.

MARSHALL, L. Medindo e Gerenciando Riscos Operacionais em Instituições Financeiras, Qualitymark, 2010.

PRESSMAN, R.S. Software Engineering – A Practitioner's Approach. 6. Ed., 2005.

SCHWABER, Ken. Agile project management with Scrum, Microsoft Press. 2004.

SOMMERVILLE, Ian. *Engenharia de Software*. 7. Ed., 2004.

SUTHERLAND, J. How Scrum Manages Risk, 2012.

Disponível em: < http://scrum.jeffsutherland.com >. Acesso em: 18/06/2013.

SUTHERLAND, J. An Alternative to Kanban: One-Piece Continuous Flow, 2011.

Disponível em: < http://scrum.jeffsutherland.com >. Acesso em: 18/06/2013.

SUTHERLAND, J. Scrum Video, 2009.

Disponível em: < http://scrum.jeffsutherland.com >. Acesso em: 18/06/2013.