YURI COSTA DA MATA

PROPOSTA DE INCORPORAÇÃO DE GERENCIAMENTO DE RISCO AO PROCESSO DE DESENVOLVIMENTO ÁGIL SCRUM

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo – como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. M.e Marcel Luiz Garcia de Miranda

São Paulo
2015

YURI COSTA DA MATA

PROPOSTA DE INCORPORAÇÃO DE GERENCIAMENTO DE RISCO AO PROCESSO DE DESENVOLVIMENTO ÁGIL SCRUM

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo – como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. M.e Marcel Luiz Garcia de Miranda

São Paulo
2015

Catalogação-na-publicação

da Mata, Yuri
PROPOSTA DE INCORPORAÇÃO DE GERENCIAMENTO DE RISCO AO PROCESSO DE DESENVOLVIMENTO ÁGIL SCRUM / Y. da Mata -- São Paulo, 2015.
68 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1.SCRUM 2.Métodos ágeis 3.gerenciamento de risco 4.Métodos ágeis 5.risco I.Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t.

DEDICATÓRIA

Aos meus irmãos
Janaina, Ariadne, Cauê (sempre presente), Larissa, Iago e Gabriela.

AGRADECIMENTOS

Primeiramente, a todos que participaram desse processo e que estiveram comigo desde o começo até a conclusão dessa etapa, família, amigos e a Sociedade do Azeite.

Aos meus primeiros gerentes Gustavo Guello e Luís Fernando Kaefer que me apresentaram ao PECE em 2010, quando me mudei para a cidade de São Paulo.

Ao meu orientador, Marcel Miranda, por toda a atenção e orientação que me proveu durante o desenvolvimento desse trabalho.

Aos meus colegas e amigos de turma, pela troca de experiências, conversas e trabalhos.

À Universidade de São Paulo – USP, à Escola Politécnica da Universidade de São Paulo – EPUSP e ao PECE – Programa de Educação Continuada em Engenharia que me abriam as portas dessa especialização de altíssimo nível.

RESUMO

O objetivo desse trabalho é propor a utilização do gerenciamento de risco no processo de desenvolvimento ágil *SCRUM*. O ambiente analisado utiliza o modelo cascata de desenvolvimento de software. Esse modelo enfrenta algumas fragilidades devido às constantes alterações de escopo e requisitos de seus sistemas. Dessa forma, o ambiente estudado poderia se beneficiar da utilização de metodologias ágeis de desenvolvimento de software, combinando técnicas de gerenciamento de risco a fim de mitigar e mapear os possíveis impactos dessas alterações no decorrer do projeto.

O desenvolvimento linear e sequencial do modelo cascata, com as alterações como as descritas anteriormente, geraria replanejamento de atividades. Caso essas alterações surjam tardiamente, será maior seu impacto sobre o projeto, podendo atrasar o cronograma e aumentar o custo, chegando ao ponto de que o desenvolvimento do projeto tenha de ser descontinuado. Em contrapartida, a utilização de técnicas ágeis de desenvolvimento pode auxiliar na correção desse cenário, visto que todo o desenvolvimento e entregas são planejados de maneira incremental e, ao final de cada ciclo de desenvolvimento, espera-se que exista um software em funcionamento. Entretanto, as metodologias ágeis não prevêm uma atividade descrita com o intuito de gerenciar os riscos do projeto que, segundo a literatura, é uma das fases mais importantes do projeto, pois diminui as chances de insucesso.

Como intuito de preencher esta lacuna, o trabalho se baseia em artigos cujos autores propõem a utilização de técnicas de gerenciamento de risco como etapas de um projeto *SCRUM*, além de utilizar literaturas que detalham o método ágil *SCRUM* e a gestão de risco. A partir da análise e comparação dos artigos utilizados é proposta uma adaptação a um dos modelos descritos e, em seguida, a aplicação desde modelo em um projeto de software desenvolvido em um Instituto de pesquisa. Dessa forma, esse trabalho e suas conclusões podem auxiliar outras empresas que possuam ambientes próximos ao descrito, indicando como utilizar o modelo proposto pelo trabalho ou, no caso de empresas que já utilizem *SCRUM* como modelo de desenvolvimento, como incrementá-lo com técnicas de gestão de risco.

Palavras-chave: Métodos ágeis, metodologia ágil, *SCRUM*; gestão de risco, gerenciamento de risco, risco, *PMBOK*.

ABSTRACT

The objective of this study is to propose the use of risk management to agile development process *SCRUM*. The analysis environment uses the waterfall model of software development, but faces some difficulty due to constant scope changes and requirements of their systems, thus studying the environment could benefit from the use of agile methodologies combined software development management techniques risk in order to mitigate and map the potential impacts of these changes on the project.

Because of the nature of the linear development model and the sequential waterfall model changes as described above generate activities of redesign if those changes arise later the greater the impact on the project could delay the schedule, cost increase to the point that the project development can be discontinued. In contrast, the use of agile development techniques can assist in correcting this scenario, since all development and deliveries are planned incrementally at the end of each development cycle, it is expected that there is a software running. However, agile methodologies do not provide an activity described in order to manage project risks which according to literature is currently one of the most important phases of the project because it reduces the chances of project failure.

The work uses of articles whose authors propose the use of risk management techniques as steps in a *SCRUM* project, and use of references detailing the agile method *SCRUM* and risk management. From the analysis comparison of used items, it proposed an adjustment to one of the models described and then the application model from a software project developed in a research institute. Thus, the findings of this study may help other companies that have rooms next to that described the use of the proposed model for the work and for companies already using *SCRUM* as a development model, increase it with risk management techniques.

Key - words: Agile methods, Agile, *SCRUM*; Risk management, risk management, risk, *PMBOK*.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. Introdução

Em um ambiente de uma empresa de médio porte (no decorrer do texto, será tratada por Instituto, por motivos legais), cujo foco não é exclusivamente de tecnologia da informação, entretanto fornece produtos de software tais como: projetos de geoprocessamento, sistemas de inteligência de negócio e projetos de otimização de processos tanto internos quanto para clientes externos a organização. Os produtos oferecidos foram desenvolvidos utilizando o modelo cascata de desenvolvimento. Dessa forma, durante a fase de desenvolvimento, a equipe encontrou diversas dificuldades à medida que alterações de requisitos e escopo foram solicitadas pelo cliente, chegando a casos de projetos serem descartados ou replanejados desde as fases iniciais.

Por se tratar de uma empresa que trabalha em conjunto com a prefeitura é necessário possuir um processo que gerencie, controle e monitore os projetos que são desenvolvidos internamente pela equipe de desenvolvimento, pois é necessário apresentar relatórios para o governo da cidade. Um dos fatores a serem analisados durante o processo é o gerenciamento de risco que autores encontrados na literatura classificam como etapa fundamental para reduzir os riscos de insucesso do projeto de software.

O risco, segundo o *PMBOK* (2013), é um evento ou uma condição incerta que pode ocorrer e afetar o objetivo do projeto de maneira positiva ou negativa. Gerenciar esses riscos é uma atividade complexa, pois é necessário predizer possíveis eventos que possam impactar o projeto, o que não é uma tarefa fácil (ARNUPHAPTRAIRONG, 2011). O aparecimento de riscos está relacionado ao dinamismo das empresas, visto que o ambiente organizacional é dinâmico e as empresas possuem diferentes diretrizes e a constante busca por novas oportunidades fazem com que haja repriorização de atividades e objetivos.

Para adaptar ao dinamismo organizacional, a utilização de metodologias ágeis para projetos de desenvolvimento de software vem ganhando espaço dentro das equipes de desenvolvimento nas últimas décadas. Os métodos de desenvolvimento que possuem os conceitos de ágil, possuem como características principais o desenvolvimento e entregas incrementais, e possuem a premissa de que os requisitos podem mudar constantemente durante o desenvolvimento. Além disso, seu objetivo é entregar o software com qualidade e em funcionamento aos clientes o mais rápido possível. Porém, as propostas de métodos ágeis não definem atividades próprias para o gerenciamento de risco. Como, então, usufruir das vantagens das técnicas de desenvolvimento ágil e, ao mesmo tempo, garantir uma gestão de risco apropriada?

1.1. Motivação

Pressman (2011) enumerou as seguintes desvantagens de se utilizar o modelo de desenvolvimento de software cascata:

- Projetos de software raramente seguem o fluxo sequencial que o ciclo cascata propõe;
- O cliente dificilmente consegue estabelecer explicitamente todas as necessidades;
- O cliente deve ser paciente, já que uma versão do sistema só estará disponível ao final do projeto;
- Erros não detectados nas fases iniciais podem causar impactos desastrosos.

Analisando o ambiente descrito, em que projetos sofriam modificações de requisito e escopo, chegou-se ao ponto de replanejamento total das atividades do projeto. A utilização de modelos de desenvolvimento ágil surge como opção para minimizar os impactos causados sobre o projeto. Porém, como citado anteriormente, os métodos ágeis não possuem atividades específicas de gerenciamento de risco em suas definições. Dessa forma, é possível utilizar modelos de gerenciamento de risco difundidos na literatura e em ambientes organizacionais associados às atividades descritas nos métodos ágeis e assim agregar valor o modelo de desenvolvimento aplicado no Instituto.

1.2. Objetivo

Com o cenário de alterações constantes de requisitos e as dificuldades de adaptação do modelo cascata a essas mudanças, o objetivo desse trabalho é propor a incorporação de um modelo de gerenciamento de risco ao processo *SCRUM* de desenvolvimento ágil de software.

1.3. Justificativa

Os projetos de softwares desenvolvidos pelo Instituto analisado utilizam o modelo de desenvolvimento cascata e a gestão do projeto utiliza atividades propostas pelo PMI, tais como: a elaboração de documentos iniciais de projeto, cronogramas, matriz de designação e responsabilidades do projeto e matriz de riscos potenciais ao projeto.

Durante conversas com os gestores, percebeu-se que, durante o processo de desenvolvimento dos projetos, havia muitas alterações de requisitos e do escopo do projeto, por motivos diferentes, por exemplo: os usuários envolvidos não possuíam a compreensão completa de todas a regras do produto ou porque os usuários finais não estavam engajados no projeto durante as fases iniciais.

Exposto esse cenário, sugere-se a utilização de um processo de desenvolvimento de software com características de entregas segmentadas e incrementais. Dessa forma, o Instituto optou por utilizar o modelo *SCRUM* como método para o processo de desenvolvimento de software, visto que seu modelo propõe atividades que garantem que ao final de uma *SPRINT* o software possua uma versão funcionando de acordo com os requisitos especificados, além de atender o princípio das metodologias ágeis, que é priorizar as interações entre os indivíduos, as ferramentas e processos (BECK et al, 2001).

Entretanto, o modelo *SCRUM*, por si só, não prevê uma atividade pré-definida de gerenciamento de risco, que, segundo Arnuphaptrairong (2011), é uma atividade crucial

para projetos de desenvolvimento de software, pois essa atividade auxilia no planejamento, controle das atividades e na garantia de sucesso do projeto.

Partindo dessa premissa, justifica-se o estudo e análise de modelos de riscos que, combinados com as atividades propostas pelo modelo *SCRUM*, devem agregar valor o processo de desenvolvimento adotado pelo Instituto.

1.4. Metodologia

O método escolhido para o desenvolvimento desse trabalho foi o estudo de caso. Realizou-se uma revisão bibliográfica sobre métodos de desenvolvimento ágil de software, sobre conceitos de gestão de risco e, também, sobre modelos que propuseram a utilização de técnicas de gerenciamento de risco incorporadas aos métodos ágeis de desenvolvimento *SCRUM*.

O passo seguinte foi a realização de análise, comparação e discussão dos modelos de desenvolvimento ágil de software que definem etapas de gerenciamento de risco. A partir desta discussão, foi feito o detalhamento dos passos de gerenciamento de risco de um dos modelos apresentados, aplicando-o num contexto real de projeto de desenvolvimento de software em uma empresa de médio porte. Colheu-se dados para serem analisados posteriormente. A aplicação desse modelo, tornou possível discutir os resultados e realizar as considerações finais do trabalho.

A técnica de pesquisa de estudo de caso é recomendada quando se as seguintes situações existem: a principais questões na pesquisa envolvem as perguntas “porque” e “como”; quando o pesquisador um certo controle da situação a ser analisada; e, por último, quando o estudo de caso é voltado para um fenômeno contemporâneo (Yin, 2014).

Além das situações recomendadas para utilizadação de estudo de caso, Yin (2014) também define o estudo de caso em três tipos: exploratório, explicativo e descritivo. O estudo de caso exploratório é aquele em que se formulam perguntas e hipóteses antes da pesquisa ser realizada. Por outro lado, o tipo explicativo é o estudo de caso da

pesquisa já realizada. Por último, Yin (2014) propõe o estudo de caso do tipo descritivo, que visa descrever as diferentes caracteríscas de um fenômeno baseado em uma ou mais teorias para recolher os dados a serem analisados.

O presente trabalho propõe um modelo de gerenciamento de risco combinado à técnica de desenvolvimento ágil de software *SCRUM*, sendo classificdo como um estudo de caso descritivo.

1.5. Estrutura do trabalho

O trabalho está estruturado em 4 capítulos, o capítulo 1 – INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas, a metodologia e a estrutura do trabalho.

O capítulo 2 – REVISÃO BIBLIOGRÁFICA apresenta a revisão bibliográfica sobre métodos ágeis, detalha os princípios do modelo *SCRUM*. Também apresenta os conceitos sobre gerenciamento de risco, definição de risco e gerenciamento de risco e detalha o modelo de gerenciamento de risco previsto pelo *PMBOK*. Além das revisões bibliográficas sobre métodos ágeis e gerenciamento de risco, a seção apresenta e discute duas propostas de modelo que relacionam os conceitos de gerenciamento de risco com a metodologia ágil *SCRUM*.

O capítulo 3 – PROPOSTA DE DETALHAMENTO PARA O MODELO APRESENTADO PELA UNIVERSIDADE DO PORTO apresenta um estudo de caso que propõe um detalhamento do modelo criado pela Universidade do Porto. Também contém a discussão sobre a aplicação desse modelo a um projeto desenvolvido por uma equipe de desenvolvimento de software de um Instituto de Pesquisa que presta serviço para uma cidade do interior paulista.

Por fim, o capítulo 4 – CONSIDERAÇÕES FINAIS, descreve as conclusões obtidas no desenvolvimento deste trabalho, as contribuições do trabalho e propões alguns trabalhos para continuação deste trabalho.

2. Revisão bibliográfica

2.1. Metodologia Ágil

O mercado muda rapidamente e, para acompanhar essas mudanças, o ciclo de desenvolvimento de software precisa ser flexível o suficiente para impedir que as organizações percam novas oportunidades para suas concorrentes. Para alcançar essa habilidade, o processo de desenvolvimento de software precisa ter foco no que é realmente importante para a organização, isto é, faz-se necessário ajustar o processo de modo que passe a priorizar tudo que dê valor para atingir a meta final e diminuir as atividades que não agregam valor ao negócio. Os valores ágeis destacam os pontos que agregam valor ao processo de desenvolvimento de software (SMITH; SIDKY, 2009).

Com o objetivo de identificar os valores que seriam mais benéficos para o processo de desenvolvimento de software, reuniu-se, em 2001, um grupo composto pelos seguintes autores: 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. Eles escreveram um documento chamado “O Manifesto Ágil para Desenvolvimento de Software” que possui os seguintes itens:

- “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” (BECK et al*,* 2001)

O manifesto ágil de desenvolvimento de software reconhece que existam processos e ferramentas, documentação abrangente, negociação de contratos e planos. Entretanto, propõem que os itens: indivíduos e interações, software em funcionamento, colaboração com o cliente e a resposta para mudanças sejam mais valorizados em relação aos primeiros itens. Ou seja, defendem que os últimos itens agregam mais valor ao processo de desenvolvimento de software.

Além dos itens citados acima, o “Manifesto Ágil para o Desenvolvimento de Software” define doze princípios que representam as características ou traços inerentes ao processo ágil:

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 aumentam 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” (BECK et al, 2001).

Segundo Pressman (2011), os métodos ágeis mais utilizados e difundidos são “Extreme Programming (XP) ” e “*SCRUM*”. Porém, muitos outros têm sido propostos, como por

exemplo: Crystal, Desenvolvimento de Software Adaptativo (*Adaptive* Software *Development - ASD*), Método de Desenvolvimento de Sistemas Dinâmicos (*Dynamic Systems Development Method - DSDM*), Desenvolvimento Dirigido à Funcionalidades (*Feature Driven Development - FDD*), Desenvolvimento de Software Enxuto (*Lean* Software *Development - LSD*). Embora os métodos ágeis sejam todos baseados em desenvolvimento e entrega incrementais, propõem diferentes formas para atingir estes objetivos. Nem todo processo ágil aplica os doze princípios do manifesto ágil atribuindo-lhes o mesmo peso, e alguns modelos relevam a importância de um ou mais princípios. Entretanto, estes princípios definem a essência da agilidade mantida nos processos ágeis (PRESSMAN, 2011).

Segundo Smith e Sidky (2009, p. 9)**: "**Não há metodologia melhor que outra; tudo depende como se adaptará ao ambiente que será aplicado, considerando suas limitações".

2.2. *SCRUM*

SCRUM, cujo nome é derivado de uma atividade que ocorre durante uma partida de *rugby*, é um método de desenvolvimento de software ágil que foi concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 90. Atualmente, o método *SCRUM* é mantido em desenvolvimento por Schwaber e Beedle (PRESSMAN, 2011).

SCRUM é um meio ágil de gerenciar projetos, geralmente projeto de desenvolvimento de software. O desenvolvimento ágil de software com *SCRUM*, normalmente é visto como uma metodologia; entretanto, é necessário enxergá-lo como um modelo para gerenciar processos (MICK COHN, 2014), modelo este que, segundo seus criadores, Ken Schwaber e Jeff Sutherland, "pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente, entregam produtos com o mais alto valor possível" (SCHWABER; SUTHERLAND, 2013, p. 3).

O modelo do *SCRUM* consiste em: a equipe *SCRUM* e seus papéis, eventos, artefatos e regras. Cada componente possui um papel importante para o sucesso do *SCRUM* (SCHWABER; SUTHERLAND, 2013).

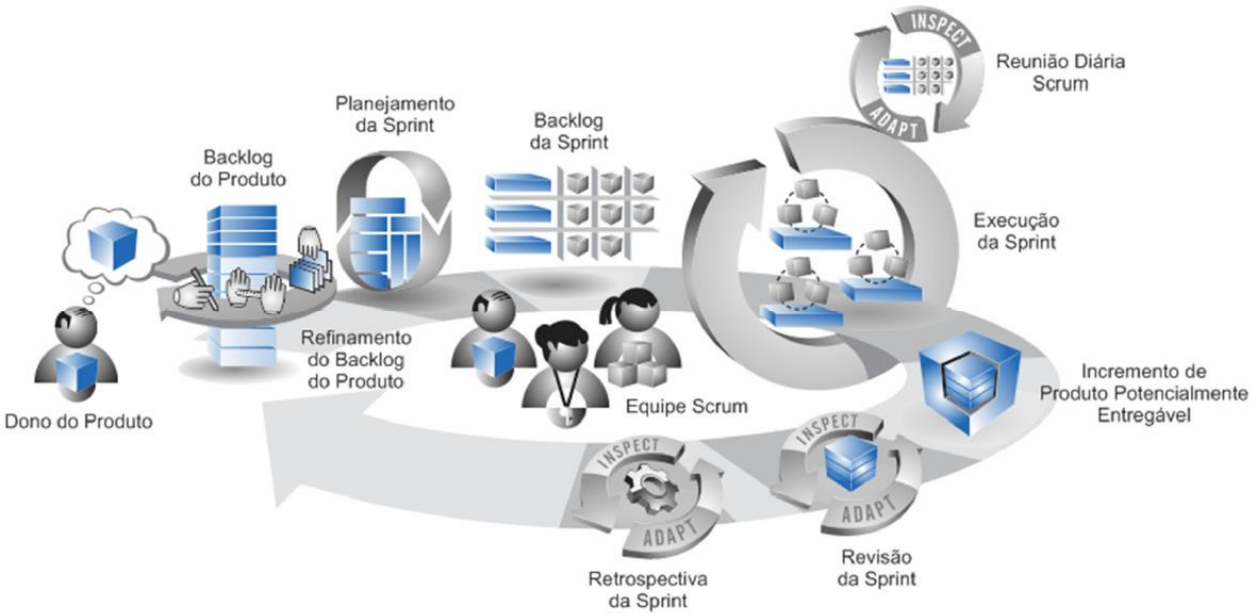

Figura 1 - Modelo SCRUM. Fonte (RUBIN,2012)

2.2.1. Papéis

Muitos modelos de processo de desenvolvimento ágil (por exemplo, *SCRUM*) permitem que sua equipe possua autonomia para tomada de decisões e definição de como serão desenvolvidas as atividades técnicas do projeto. É permitido que a equipe selecione sua própria abordagem (por exemplo, processos, métodos, ferramentas), limitados apenas pelas exigências do negócio e normas da organização. Conforme o projeto avança, a equipe se auto-organiza, concentrando competências individuais de maneira que seja mais benéfica para o projeto. Para atingir esse ponto, uma equipe ágil pode conduzir reuniões de equipe diárias para coordenar e sincronizar o trabalho que deve ser realizado para esse dia (PRESSMAN, 2011).

A equipe do *SCRUM* é composta pelo dono do produto, pela equipe de desenvolvimento e o *SCRUM MASTER*. As equipes de *SCRUM* são auto-organizáveis e multi-disciplinares. Ao invés de serem coordenadas por outras pessoas, as equipes se gerenciam e decidem a melhor forma de trabalhar. Uma equipe multidisciplinar possui todas as competências necessárias para realizar as atividades propostas sem que seja necessário interveção de pessoas de fora da equipe. Esse modelo de equipe proposto visa aperfeiçoar a flexibilidade, criatividade e produtividade (SCHWABER; SUTHERLAND, 2013).

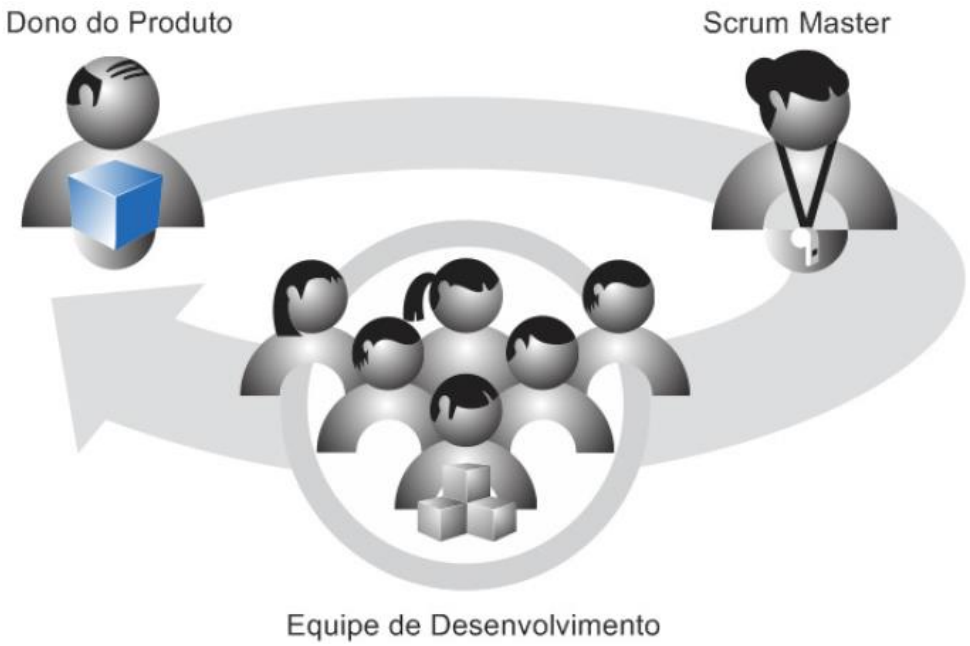

Figura 2 - Equipe SCRUM. Fonte (RUBIN,2012)

Dono do produto

O dono do produto (*product owner*) é responsável por representar os interesses de todos os *stakeholders*[1] no projeto e do sistema resultante. O dono do produto consegue o financiamento inicial e permanece por todo o projeto atuando na criação de requisitos

[1] Termo em inglês que é utilizado para descrever uma pessoa ou um grupo de pessoas que sejam relevantes para o projeto. Em tradução livre, são as partes interessadas no projeto.

gerais iniciais, monitora o retorno sobre o investimento (ROI) e objetivos (SCHWABER, 2004).

Segundo Ken Schwaber e Jeff Sutherland (2013), o dono do produto também é o responsável por gerir o *BACKLOG do* produto. As seguintes atividades estão associadas a gestão *BACKLOG do* produto:

- Compreender os itens do *BACKLOG do* produto;
- Priorizar os itens do *BACKLOG do* produto a fim atingir as metas e missões propostas no projeto;
- Garantir que o trabalho realizado pela equipe de desenvolvimento esteja dentro dos valores definidos no projeto;
- Garantir que todos os membros da equipe *SCRUM* compreendam com clareza a prioridade dos itens do *BACKLOG* do produto para que saibam quais são os itens que serão executadas e em que fase serão realizados; e,
- Garantir que a equipe de desenvolvimento compreenda os itens do *BACKLOG* do produto com clareza, a fim de reduzir riscos de falta de compreensão dos requisitos do sistema.

SCRUM MASTER:

É um facilitador que organiza reuniões diárias, juntamente com a equipe de desenvolvimento, acompanha o *BACKLOG* de trabalho a ser realizado durante as *SPRINTS*, registra decisões, avalia a evolução do projeto contra o atraso e se comunica com clientes e gestores de fora da equipe (SOMMERVILLE, 2011).

Segundo Schwaber (2004), as responsabilidades do *SCRUM MASTER* podem ser resumidas da seguinte forma:

- Remover as possíveis barreiras entre a equipe de desenvolvimento e o dono do produto para facilitar a comunição entre eles;
- Ensinar ao dono do produto como maximizar o ROI e atender os seus objetivos do projeto através de *SCRUM*;

- Proporcionar um ambiente para a equipe de desenvolvimento que estimule a criatividade e dê a autonomia necessária para o desenvolvimento das atividades;
- Auxiliar a equipe de desenvolvimento a maximizar sua produtividade;
- Criar melhorias no processo de criação de novos incrementos do produto utilizando melhores práticas e ferramentas;
- Manter todos os envolvidos no projeto cientes do progresso das atividades.

Equipe de desenvolvimento

A equipe de desenvolvimento é responsável pelo desenvolvimento das funcionalidades selecionadas para a *SPRINT*. As equipes são auto-gerenciáveis, auto-organizadas e multidisciplinares; a equipe de desenvolvimento é responsável por descobrir como transformar os itens do *BACKLOG* do produto em funcionalidades que serão desenvolvidas de maneira incremental dentro de uma iteração do projeto, além de gerir o seu próprio trabalho enquanto desenvolve as atividades. Os membros da equipe são responsáveis pelo sucesso de cada iteração e do projeto como um todo (SCHWABER, 2004).

Segundo Ken Schwaber e Jeff Sutherland (2013), a equipe de desenvolvimento possui as seguintes características:

- A equipe deve ser autoorganizada. Após definidos os itens que constituirão o *BACKLOG do* produto, cabe à equipe de desenvolvimento definir como serão os incrementos das funcionalidades a serem entregues ao final das *SPRINT*S;
- As equipes de desenvolvimento são multidisciplinares, isto é, possuem todas as habilidades necessárias para o desenvolvimento do incremento;
- Não há hierarquia numa equipe *SCRUM*, todos os membros são tratados como desenvolvedores;
- A equipe é formada por membros que podem possuir especialidades específicas, porém a responsabilidade e comprometimento de entrega do incremento cabe a todos os membros da equipe *SCRUM*;

- A equipe de desenvolvimento não possui subequipes especializadas em uma determinada fase do projeto, por exemplo, teste ou análise de negócio.

2.2.2. Eventos

O *SCRUM* utiliza eventos para estruturar fluxos de trabalho no processo de desenvolvimento de software incremental. Todos os eventos possuem um período de tempo fixo para execução das atividades o que garante que não haja desperdício de tempo no processo de planejamento (HUNDHAUSEN, 2012).

Os eventos propostos pelo *SCRUM* são projetados para permitir que exista transparência e inspeção das fases do processo de desenvolvimento de software, caso o evento seja ocultado, resultará na redução da transparência e na diminuição de percepção de encontrar pontos de melhorias e adaptação para o processo (SCHWABER; SUTHERLAND, 2013).

SPRINT:

SCRUM organiza o trabalho em iterações ou ciclos de iteração de até um mês, chamados *SPRINTS*.

A *SPRINT* é um planejamento de todo trabalho a ser feito: os itens do *BACKLOG do* produto são avaliados e selecionados para o desenvolvimento, e o software é implementado (SOMMERVILLE, 2011). No final de cada *SPRINT* deve ser criado algo com valor tangível para o usuário ou para o cliente, quando uma *SPRINT* termina outra *SPRINT* deve iniciar imediatamente (RUBIN, 2012).

Segundo Ken Schwaber e Jeff Sutherland (2013, p. 8), durante a *SPRINT*:

- Não são feitas mudanças que possam pôr em perigo o objetivo da *SPRINT*;
- As metas de qualidade não diminuem;

- À medida que os itens do *BACKLOG da SPRINT* são compreendidos pela equipe de desenvolvimento, o escopo da *SPRINT* pode ser renegociado com o dono do produto.

Reuniões do *SCRUM*

Segundo Ken Schwaber e Jeff Sutherland (2013), as reuniões do *SCRUM* são diárias e rápidas, tipicamente possuem 15 minutos de duração, sendo que toda a equipe *SCRUM* participa. Durante a reunião três perguntas chaves são respondidas por todos os membros da equipe:

- O que você fez desde a última reunião?
- Quais foram os obstáculos que você encontrou?
- O que você precisa realizar até a próxima reunião?

O *SCRUM MASTER*, lidera a reunião e avalia cada resposta de cada membro da equipe. A reunião diária ajuda a equipe a encontrar possíveis problemas o mais cedo possível. Além disso, essas reuniões diárias levam à “socialização do conhecimento” e, assim, promovem uma estrutura de equipe de auto-organização (PRESSMAN, 2011), isto é, significa que todos os membros da equipe sabem o que está acontecendo e, se surgirem problemas, podem replanejar o trabalho de curto prazo para lidar com os problemas. Caso um problema seja encontrado, todos os membros da equipe *SCRUM* participam do planejamento para solucioná-lo, a decisão não é tomada de maneira hierárquica pelo *SCRUM MASTER* (SOMMERVILLE, 2011).

Planejamento da *SPRINT*

O planejamento da *SPRINT* é realizado na reunião de inicial da *SPRINT* e todos os membros da equipe *SCRUM* devem participar, é o momento em que o trabalho a ser realizado é planejado.

A reunião de planejamento da *SPRINT* responde as seguintes questões:

- O que pode ser entregue como resultado do incremento da próxima *SPRINT*?
- Como o trabalho necessário para entregar o incremento será realizado? (SCHWABER; SUTHERLAND, 2013; p.9).

Ao final da reunião de planejamento do *SPRINT*, a equipe de desenvolvimento deve estar apta para explicar ao dono do produto e ao *SCRUM MASTER* como pretende trabalhar como uma equipe autoorganizada para atingir os objetivos da *SPRINT*e criar o incremento previsto para o produto (RUBIN, 2012).

Revisão da *SPRINT*

A reunião de revisão do *SPRINT* é uma reunião de quatro horas de duração, em que a equipe apresenta o que foi desenvolvido durante a *SPRINT* para o dono do produto e para os *stakeholders* que queiram participar. É uma reunião informal, cujo objetivo é reunir as pessoas com intuito de ajudá-los, de forma colaborativa, a determinar o que será realizado no próximo *SPRINT* (SCHWABER, 2004).

Segundo Ken Schwaber e Jeff Sutherland (2013) as atividades da reunião de revisão do *SPRINT* são as seguintes:

- Todos os membros da equipe *SCRUM*, incluindo *SCRUM MASTER* e dono do produto devem participar e os *stakeholders* também são convidados;
- O dono do produto mostra os itens do *BACKLOG* do produto que foram atendidos durante a *SPRINT* e quais não foram;
- Cabe à equipe de desenvolvimento discutir os pontos positivos e negativos da *SPRINT* e comentar os problemas encontrados durante o desenvolvimento da *SPRINT*, além de mostrar as ações tomadas para contornar esses problemas;
- A equipe desenvolvimento demonstra o trabalho que está finalizado e responde as questões sobre o incremento;

- O dono do produto discute o andamento do *BACKLOG* do produto. Com o resultado da análise do *BACKLOG* do produto, o dono do produto projeta a possível data de conclusão das atividades;
- Durante a reunião, o grupo discute quais seriam os itens mais relevantes para definição de quais itens do *BACKLOG* serão desenvolvidos na próxima *SPRINT*;
- Uma nova análise do uso potencial do produto que está sendo desenvolvido é realizada;
- É realizada uma análise sobre orçamento, potenciais capacidades do produto e do mercado para a próxima versão esperada do produto.

Retrospectiva da *SPRINT*

O *SCRUM MASTER* lidera a reunião de retrospectiva junto com a equipe de desenvolvimento. São três horas de reunião em que o *SCRUM MASTER* encoraja a equipe a revisar a *SPRINT* com o propósito de avaliar o processo do modelo do *SCRUM* e suas práticas. Com isso, haverá uma evolução do processo para torná-lo mais eficaz e agradável para a próxima *SPRINT* (SCHWABER, 2004).

Ao final da reunião de retrospectiva da *SPRINT*, a equipe *SCRUM* terá enumerado os pontos de melhoria que serão implantados na *SPRINT* seguinte. A fase de implementação desses pontos de melhoria mostra a capacidade de adaptação que a equipe possui com relação às inspeções realizadas pela própria equipe. Apesar da reunião de retrospectiva ser um evento previsto pelo modelo do *SCRUM*, pontos de melhoria que são identificados no decorrer da *SPRINT* podem ser implementos a qualquer momento (SCHWABER; SUTHERLAND, 2013).

2.2.3. Artefatos

Os artefatos previstos no *SCRUM* tem o objetivo de apresentar os itens que serão desenvolvidos durante o projeto e fornecem informações que auxiliam a inspeção das atividades e a adaptação do projeto. São compostos pelo *BACKLOG do* produto,

BACKLOG da SPRINT e pontos de incremento do produto utilizáveis. (SCHWABER; SUTHERLAND, 2013).

***BACKLOG* do produto**

Segundo definição de Schwaber (2004), o *BACKLOG do* produto é a lista de requisitos funcionais e não funcionais que se tornarão funcionalidades quando o produto for entregue. O *BACKLOG do* produto centraliza e compartilha o conhecimento do que será construído e a ordem de como será desenvolvido. Essa informação é acessível para todos os participantes do projeto, é o coração do modelo do *SCRUM* (RUBIN, 2012).

Os itens que constituem o *BACKLOG do* produto são priorizados considerando-se os itens que agregaram maior valor ao negócio; o dono do projeto é o responsável para priorizar os itens (SCHWABER, 2004).

Mudanças nos itens que constituem o *BACKLOG do* produto refletem alterações nos requisitos do negócio e medem o quão rápido ou lentamente a equipe de desenvolvimento consegue transformar os requisitos em funcionalidades (SCHWABER, 2004).

Refinamento do *BACKLOG* do produto

O processo de refinamento do produto do *BACKLOG* – no inglês o termo utilizado é *groomin* – é constituído por três atividades: criação e refinamento (incluindo detalhes para os itens do *BACKLOG do* produto), estimativas e priorização dos itens do *BACKLOG do* produto (RUBIN, 2012).

Este é um processo contínuo, em que o dono do produto e a equipe de desenvolvimento colaboram para detalhar os itens do *BACKLOG do* produto. É durante a etapa de refinamento que os itens que formam o *BACKLOG do* produto e, por consequência, os do *BACKLOG da SPRINT* são analisados, revisados e detalhados. Cabe à equipe de

desenvolvimento definir quando o refinamento de cada item está finalizado. Normalmente, essa etapa do processo consome 10% da capacidade da equipe de desenvolvimento. Entretanto, os itens que constituem o *BACKLOG do* produto podem sofrer alterações a qualquer momento, exigindo um esforço maior da equipe de desenvolvimento e do dono do produto. Além da definição dos itens do *BACKLOG do* produto o dono do produto soluciona os pontos de conflito entre os itens, auxiliando na compreesão total dos itens. Porém, é a equipe de desenvolvimento que define qual é a estimativa final para o desenvolvimento de cada item. (SCHWABER; SUTHERLAND, 2013).

O refinamento do *BACKLOG do* produto deve garantir que os itens que estão no topo estejam prontos para serem movidos para a próxima *SPRINT* e que a equipe de desenvolvimento garanta que estejam finalizadas ao final da *SPRINT* (RUBIN, 2012).

BACKLOG da SPRINT

É constituido pelos itens com maior prioridade do *BACKLOG do* produto. O *BACKLOG da SPRINT* é definido na reunião de planejamento da *SPRINT*. (SCHWABER; SUTHERLAND, 2013).

Incremento de Produto Potencialmente Entregável

O *SCRUM* prevê que a cada *SPRINT* finalizada uma nova versão do produto seja entregue e em funcionamento, visto que o dono do produto pode optar por entregar essa funcionalidade imediamente para os *stakeholders* do projeto. Para isso é necessário que o código esteja bem escrito, que as funcionalidades estejam testadas e que os requisitos estejam documentados (SCHWABER,2004).

Portanto, o incremento do produto é composto por todos os itens desenvolvidos na *SPRINT* atual junto com os itens desenvolvidos nas *SPRINT*S anteriores (SCHWABER; SUTHERLAND, 2013).

2.2.4. Monitoramento

A qualquer momento, o dono do produto pode avaliar o andamento do projeto a fim de medir o quanto de trabalho ainda é neessário para atigir os objetivos traçados. As práticas mais comuns são os gráficos *burndown*[2] e *burnup*[3], também são conhecidos como gráficos de queima, em português (SCHWABER; SUTHERLAND, 2013).

Gráfico burn-down

É um gráfico que mostra, no eixo vertical, a quantidade de trabalho necessária para finalizar o trabalho ao longo do tempo, representado no eixo horizontal. A quantidade de trabalho pode ser medida em horas ou em itens do *BACKLOG do* produto. O gráfico de *burndown* é uma curva decrescente, pois entende-se que a tendência é que o trabalho diminua ao longo do tempo. É possível projetar os resultados calculando a tendência de quando o trabalho será finalizado (RUBIN, 2012).

[2] É um gráfico que ilustra a quantidade trabalho nacessária para atingir o final da *SPRINT*.
3 É um gráfico que ilustra o progresso do trabalho para atingir o final da *SPRINT*.

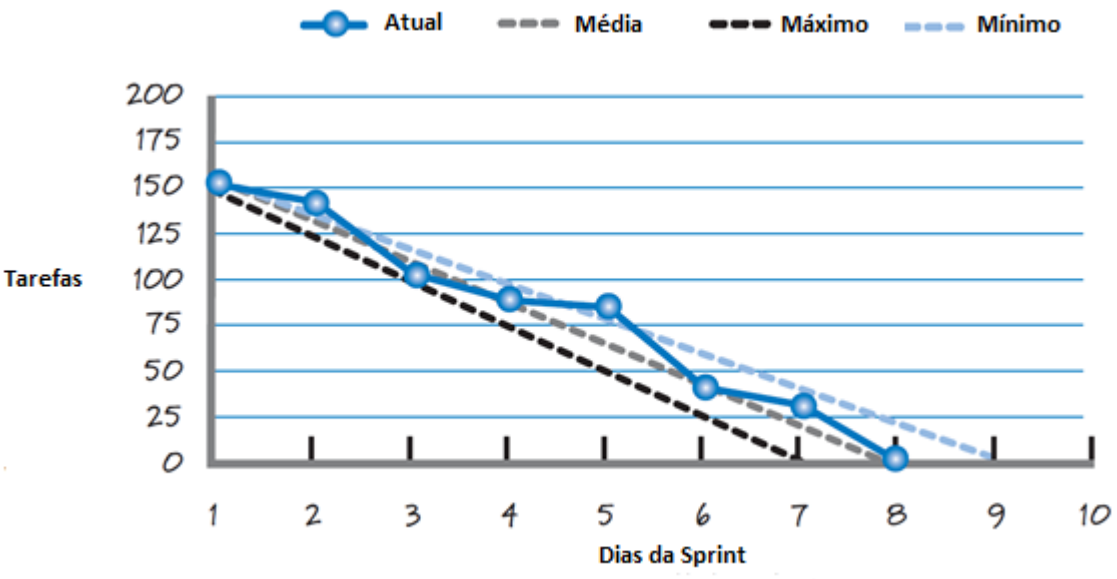

Figura 3 - Gráfico de burn-down. Fonte (RUBIN,2012)

Gráfico burn-up:

É um gráfico que mostra o progresso do trabalho para atingir a meta estabelecida no começo do projeto, no eixo vertical. O trabalho é finalizado ao longo da linha que se move para cima, aproximando-se do objetivo. É possível mostrar os resultados calculando a tendência de quando o trabalho será concluído (RUBIN, 2012).

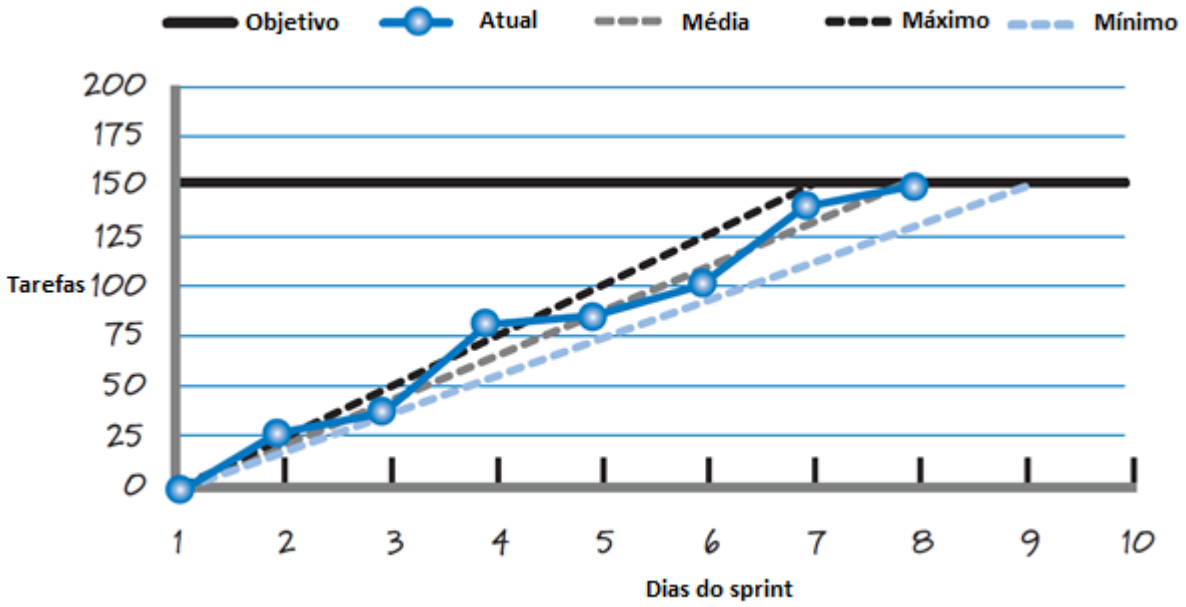

Figura 4 - Gráfico de burn-up. Fonte (RUBIN,2012)

A seção apresentou conceitos sobre metodologia ágil e apresentou as principais características do *SCRUM*.

2.3. Gerenciamento de risco

Segundo ALHawari, Thabtah e Karadsheh (2008), o gerenciamento de risco vem se tornando um fator chave dentro das organizações, uma vez que garante a execução bem-sucedida dos projetos.

O risco de um projeto tem origem na incerteza existente em todos os projetos. Os riscos conhecidos são aqueles que foram identificados e analisados, possibilitando o planejamento de respostas. Determinados riscos não podem ser gerenciados de forma proativa, o que sugere que a equipe do projeto deveria criar um plano de contingência. Riscos de um projeto que já ocorreu também podem ser considerados um problema (PMI, 2013).

Dessa forma, para projetos de desenvolvimento de software, o gerenciamento de risco tornou-se essencial, garantindo um maior controle do projeto durante sua fase de execução e reduzindo a possibilidade de insucesso do projeto (ARNUPHAPTRAIRONG, 2011).

Segundo Boehm (1991), pode - se definir risco e gerenciamento de risco como:

- Risco: Em projetos de software, é geralmente definido como o impacto ponderado pela probabilidade de um evento em um projeto.
- Gerenciamento de risco: geralmente definido como um conjunto de princípios e práticas que visam identificar, analisar e lidar com os fatores de risco para melhorar as chances de alcançar um resultado bem-sucedido de projeto e / ou evitar o fracasso do projeto.

O risco surge conforme as empresas buscam oportunidades. Durante a busca, considera-se duas dimensões: a incerteza e o custo de investimento. O maior desafio dessa busca é encontrar uma combinação apropriada dessas duas dimensões. Essa combinação é o perfil de risco da empresa, que é aceitável pelos *stakeholders* internos e externos. Para traçar o perfil de risco da empresa é necessário criar estratégias que reduzam os riscos da busca de novas oportunidades, porém a gestão de risco não pode impedir que as empresas percam seus objetivos. Dessa forma, buscar o equilíbrio entre as dimensões e gerenciá-las é a forma de reduzir os riscos para que as empresas atinjam seus objetivos. (BANNERMAN,2008; CHARETTE, 2005).

A proposta de gerenciamento de riscos em projetos de software comercial é que ela melhore os resultados do projeto. De acordo com a literatura (SIMISTER, 2004; WARD E CHAPMAN, 2004), gestão de risco pode levar a uma série de projetos e benefícios organizacionais, incluindo:

- Identificar cursos alternativos favoráveis de ação;
- Aumentar a confiança em alcançar os objetivos do projeto;
- Melhorar as chances de sucesso do projeto;

- Reduzir surpresas;
- Fornecer estimativas mais precisas (através de incerteza reduzida);
- Reduzir a duplicação de esforços (através da conscientização da equipe das ações de controle de risco).

Considerando que os projetos envolvem várias classes de participantes (clientes, desenvolvedores, usuários e equipe de manutenção), cada um com diferentes, mas altamente importantes critérios de satisfação, é claro que "resultado insatisfatório" é multidimensional:

- Para clientes e desenvolvedores, erros orçamentais e de cronograma são insatisfatórios.
- Para os usuários, os produtos com a funcionalidade errada, falhas de interface do usuário, deficiências ou insuficiências de desempenho e confiabilidade são insatisfatórios.
- Para os usuários, os produtos que não atendam às funcionalidades propostas, apresentem falhas de interface com usuário, desempenho e confiabilidade são insatisfatórios;
- Para a equipe de manutenção, um software de má qualidade é insatisfatório.

Esses conjuntos de insatisfações compõem uma lista para identificar e validar os itens de risco (BOEHM, 1991).

Portanto, o gerenciamento de riscos em projetos de software é importante para: ajudar a evitar o insucesso do projeto; evitar retrabalho; manter o foco e o esforço de equilíbrio; e estimular situações vantajosas (BOEHM, 1991). Embora nem todos os riscos se originem em práticas de software, todos eles possuem o potencial de afetar o resultado do processo de software, através do mecanismo de projeto com o qual o artefato software é normalmente entregue.

Processo de gerenciamento de risco

Os modelos de processo de gerenciamento de risco especificam as atividades necessárias para gerir os riscos de um projeto de software (por exemplo, a identificação de riscos, análise, resposta e controle) e a ordem como elas devem ser executadas para um gerenciamento de risco eficaz. Para isso, sugerem a utilização de ferramentas e técnicas a cada etapa do projeto a fim de auxiliar no processo de gestão de riscos.
Conceitualmente, a maioria dos modelos incluem um conjunto semelhante de etapas do processo, por exemplo: estratégia de risco, identificação de riscos, análise de riscos, respostas aos riscos e controle de risco (SIMISTER, 2004).

Conforme Boehm (1991), o primeiro passo do gerenciamento de risco envolve a validação dos riscos com as seguintes atividades: identificação, análise e priorização.

- Identificação do risco: cria-se uma lista enumerando os riscos que podem comprometer o sucesso do projeto;
- Análise do risco: avalia a probabilidade e o impacto de perda para cada item da lista de riscos;
- Priorização do risco: a criação de uma lista dos riscos identificados e analisados, ordenada a partir do item com maior prioridade.

O segundo passo proposto por Boehm (1991) para o gerenciamento de risco é a etapa de controle dos riscos que possui as seguintes atividades: planejamento para o gerenciamento do risco, meios para resolução dos riscos e monitoramento dos riscos.

- O planejamento de risco ajuda a endereçar medidas para cada item da lista de risco, por exemplo, transferência de risco, prevenção de risco ou redução do risco, incluindo a coordenação dos planos individuais dos riscos com todos os outros riscos do projeto;
- A resolução dos riscos gera situações em que os riscos são resolvidos ou eliminados;

- O monitoramento de risco rastreia o progresso do projeto com o intuito de resolver os itens da lista de risco e tomar medidas corretivas quando necessário.

A principal contribuição dos modelos de gerenciamento de risco é a orientação para as ações que devem ser tomadas durante o projeto e não devem ser vistas apenas como uma maneira analítica de pensar. Apesar dos modelos de gerenciamento de risco guiarem as tomadas de decisão, eles não fornecem a solução dos problemas. É necessário que a equipe possua habilidade, julgamento e persistência para aplicar o modelo, associados a ferramentas e técnicas que auxiliam durante todo o projeto (BANNERMAN,2008).

2.3.1. PMBOK

O *PMBOK* utiliza as seguintes etapas para gerenciamento de risco: processos de planejamento, identificação, análise, planejamento de respostas, monitoramento e controle de riscos de um projeto.

Planejar o gerenciamento dos riscos

Esta é a primeira etapa proposta pelo *PMBOK* (2013). É a etapa em que se define o processo de como os riscos serão conduzidos durante o gerenciamento de risco. Tal planejamento tornou-se importante, pois a partir dele será possível garantir que o grau, o tipo e a visibilidade do gerenciamento dos riscos sejam proporcionais tanto aos riscos como à importância do projeto para a organização.
Dessa forma, o processo de planejar o gerenciamento dos riscos deve começar na concepção do projeto e ser concluído nas fases iniciais do planejamento do projeto.

Identificar os riscos

Identificar os riscos é a segunda etapa definida pelo *PMBOK*. É a etapa em que se identificam os riscos que podem afetar o andamento do projeto. Além da identificação, é necessário documentar suas características.
Segundo o *PMBOK* (2013), os responsáveis pela identificação do risco são:

- Gerente do projeto;
- Membros da equipe do projeto;
- Equipe de gerenciamento dos riscos (se for designada);
- Clientes;
- Especialistas no assunto externos à equipe do projeto;
- Usuários finais;
- Outros gerentes de projetos;
- Partes interessadas e especialistas em gerenciamento de riscos.

Identificar os riscos é um processo iterativo, visto que novos riscos podem surgir ou se tornar conhecidos durante o ciclo de vida do projeto. A frequência da iteração e os participantes de cada ciclo variam de acordo com a situação.

Realizar a análise qualitativa de riscos

É a terceira etapa proposta pelo *PMBOK* (2013, p. 238) “é o processo de priorização do risco para análise ou ação adicional através da avaliação e combinação de sua probabilidade de ocorrência e impacto”. Isto é, prioriza os riscos considerando a probaliblidade do risco ocorrer durante o projeto, o impacto sobre o projeto que esse risco causaria caso ocorra e o tempo de resposta que a equipe necessita para intervir com intuito de diminuir o impacto sobre o projeto.

O resultado da análise qualitativa dos riscos gera uma lista de itens de riscos priorizados pela probablidade dos riscos ocorrerem e também um plano de como a equipe deve agir caso o risco ocorra. Essa lista de itens é utilizada como dados de entrada para a etapa

de análise quantitava de riscos e deve ser revista à medida que novos riscos são identificados durante o andamento do projeto.

Realizar a análise quantitativa de riscos

É a quarta etapa do processo de gerenciamento de risco proposto pelo *PMBOK* (2013) e consiste em analisar numericamente o efeito dos riscos identificados nos objetivos gerais do projeto. A análise quantitativa utiliza os riscos identificados e priorizados nas etapas anteriores do processo como tendo impacto potencial e substancial nas demandas concorrentes do projeto.

Essa etapa analisa os riscos e classifica numericamente os efeitos que possam ser causados pelos riscos de maneira individual ou o efeito agregado de todos os riscos do projeto. Também apresenta uma abordagem quantitativa para a tomada de decisões na presença de incertezas.

Planejar as respostas aos riscos

Esta é a quinta etapa do processo de gerenciamento de risco proposto pelo *PMBOK* (2013). Nesta etapa é que se definem as ações que serão tomadas para a redução do impacto dos riscos sobre o projeto. Para cada item da lista de riscos identificados e priorizados nas etapas anterios, define-se uma pessoa que será responsável por atuar caso o risco ocorra no decorrer do projeto.

Monitorar e controlar os riscos

A última etapa do processo de gerenciamento de risco proposto pelo *PMBOK* (2013, p. 254) consiste na “implementação dos planos de respostas aos riscos, acompanhamento dos riscos identificados, monitoramento dos riscos residuais, identificação de novos riscos e avaliação da eficácia do processo de riscos durante todo o projeto”.

É nesta etapa do processo de gerenciamento de risco que as ações planejadas na etapa anterior são executadas, caso o risco venha a ocorrer. Para realizar o monitoramento e controle dos riscos identificados nos projetos, podem-se utilizar técnicas de análise de variações e tendências que utilizam dados coletados durante a execução do projeto para simular o projeto real com o planejado. Durante o processo de monitorar e controlar os riscos do projeto as seguintes ações também são realizadas:

- Validar se as premissas do projeto continuam válidas;
- Atualizar ou inativar um risco;
- Garantir que as políticas e os procedimentos definidos estejam sendo seguidos;
- Modificar as reservas para contingências de custo ou cronograma à medida que os itens da lista de risco são atualizados. (PMI, 2013).

No decorrer do projeto, é possível que estratégias alternativas sejam criadas com o objetivo de diminuir o impacto do risco. Caso seja necessário criar novas estratégias para mitigar um risco, caberá ao responsável pelo risco reportar ao gerente de projeto as alterações realizadas no planejamento inicial de resposta aos riscos.

2.4. Análise de abordagens de gerenciamento de risco com *SCRUM*

Nesta seção, serão apresentados dois modelos de gestão de risco voltados para projetos de software que utilizam metodologia ágil. O primeiro foi realizado pelo Instituto Politécnico do Porto, em Portugal, sob autoria de Rita Cunha, Carla Sofia Pereira e José Ângelo Pinto (2013). Já a segunda proposta foi desenvolvida no Departamento de Computação e Ciência de Sistemas da Universidade de Estocolmo, Suécia, sob autoria de Jaana Nyfjord e Mira Kajko-Mattsson (2008).

2.4.1. Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos.

A proposta de modelo de gestão de riscos em projetos ágeis de desenvolvimento de software foi realizada pelo Instituto Politécnico do Porto, em Portugal. O estudo apresentado baseou-se na comparação de três modelos de gestão de risco: Barry Bohem, *Continuous Risk Management (CRM)* - SEI e *PMBOK*.

Barry Bohem

Conforme apresentado na seção 2.3.1, o modelo de gestão de risco proposto por Barry Bohem pode ser dividido em duas etapas: avaliação e controle.

Figura 5 - Passos para a gestão de risco por Barry Boehm (1991). Fonte (Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos, 2013)

1. Avaliação:
   1.1. Identificação: gerar a lista dos riscos que poderiam comprometer o sucesso do projeto;
   1.2. Análise: avaliar a probabilidade de perda e o impacto sobre o projeto. A análise é feita para cada item da lista dos riscos identificados;
   1.3. Priorização: ordenar os itens da lista que já foram identificados e analisados.
2. Controle:

2.1. Planejamento: planejar como os riscos serão solucionados e traçar planos para minimizar o impacto de cada risco;
2.2. Resolução: resolver e eliminar os riscos;
2.3. Monitoramento: acompanhar o progresso do projeto e resolução dos riscos.

Como pode – se ver na figura 5, o modelo proposto por Boehm (1991) é dividido em duas grandes áreas: avaliação e controle. A área de avaliação é realizada na fase inicial, em que as premissas e os objetivos do projeto são definidos, visto que consistem em atividades relacionadas à identificação dos riscos, na análise probabilística dos impactos que podem ser causados nos objetivos do projeto. Ao final dessa etapa, uma lista com os itens de risco é gerada e ordenada por prioridade. Já a fase de controle ocorre durante o planejamento do projeto e é mantida durante a execução deste, visto que apresenta atividades de planejamento de estratégias para diminuir o impacto de cada item da lista de risco a medida que um item seja resolvido e eliminado. Além disso, durante a fase de controle ocorre o monitoramento e acompanhamento do progresso do projeto e identificação de novos riscos.

Para realizar a etapa de identificação dos itens que irão compor a lista de risco é possível utilizar técnicas de *checklist* e experiências de projetos passados (BOEHM, B; 1991). Por sua vez, a etapa de monitoramento tem o objetivo de colher dados sobre os riscos que ocorreram durante o projeto e as estratégias de resolução; ao final do projeto os dados colhidos nessa etapa servirão de insumo para identificação de riscos do próximo projeto.

Apesar de o modelo de Boehm ser um modelo anterior às propostas de metodologias ágeis de desenvolvimento, é possível identificar atividades que se relacionem à aprendizagem que a equipe desenvolve entre um projeto e outro, a qual é contínua no decorrer dos projetos executados. Porém, um ponto a se destacar é que o modelo de Boehm não nomeia uma pessoa como responsável pelos itens da lista de risco.

Os autores Cunha, Pereira, Pinto (2013), quando analisaram o modelo de Boehm, identificaram alguns fatores relacionados às dimensões de risco tais como: requisitos e complexidade do projeto; planejamento e controle das atividades; e gestão e maturidade da equipe.

Continuous Risk Management (CRM)

O Instituto de Engenheira de Software da Universidade *Carnegie Mellon* criou o modelo de gerenciamento contínuo de risco. O ponto chave desse modelo é a comunicação, como se pode observar na imagem abaixo:

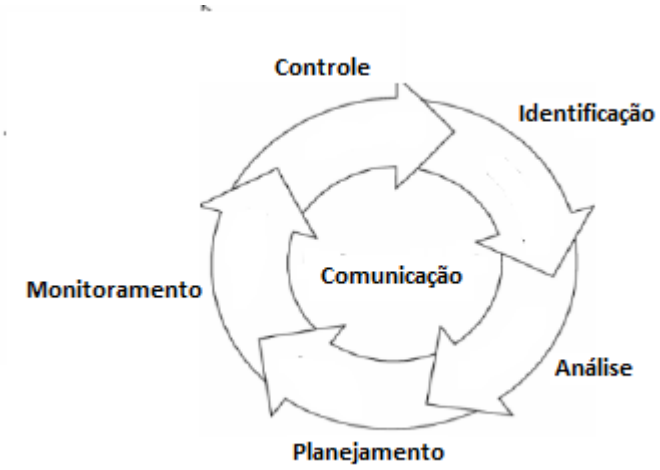

Figura 6- Continuous Risk Management (CRM), SEI. Fonte (Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos, 2013)

O modelo é composto por cinco etapas, conforme segue:

1. Identificação: identificar os riscos antes mesmo deles se tornarem um problema;
2. Análise: avaliar os impactos e a probabilidade que ocorram. A partir desse resultado os riscos são classificados e priorizados;
3. Planejamento: define as estratégias de minimização dos riscos;
4. Monitoramento: monitorar os indicadores e os planos de resolução dos riscos;
5. Controle: produz a análise para tomada de decisão de cada risco.

Da mesma forma que o modelo de gerenciamento de risco proposto por Boehm, o modelo proposto pelo Instituto de Engenharia de Software da Universidade *Carnegie Mellon* possui atividades de identificação dos riscos, análise probabilísticas e de impacto, caso o risco ocorra durante o projeto; fase de planejamento de estratégias a serem tomadas para redução do impacto do risco e rotinas de monitoramento e controle da lista de riscos. Entretanto, a atividade central dessa proposta de gereciamento de risco é a

comunicação, a qual interliga as demais etapas do modelo. A imagem desse modelo nos mostra que a proposta possui um ideal cíclico de gestão de risco.

Considerando que a premissa do modelo CRM é a comunicação e que é um modelo cíclico, é possível associá-lo aos valores propostos nas metodologias ágeis de desenvolvimento. Por exemplo, as etapas de desenvolvimento ocorrem de maneira cíclica e incremental, de modo que em cada fim de ciclo é realizada uma reunião para avaliá-lo. A partir dessas reuniões gera-se uma coleção de dados que auxiliarão no ciclo posterior, o que consiste na ideia de aprendizagem contínua. Outro valor proposto pelos métodos ágeis é a integração das equipes, a comunicação diária para entender e mapear os possíveis problemas que a equipe poderá enfrentar durante o desenvolvimento do projeto. Porém, o modelo CRM, assim como o modelo proposto por Boehm, não possui a nomeação de uma pessoa resposável pelos itens da lista de riscos.

Durante os estudos sobre o modelo CRM, Cunha, Pereira e Pinto (2013) mapearam as seguintes dimensões de risco: requisitos e complexidade do projeto; planejamento e controle dos itens da lista de risco; e gestão e maturidade da equipe de desenvolvimento.

Conforme apresentado na seção 2.3.2, o *PMBOK* consiste nas seis etapas apresentadas na figura 7: planejamento, identificação, avaliação qualitativa, avaliação quantitativa, planejamento de resposta e monitoramento e controle.

Figura 7 - Fluxo da gestão de risco – PMBOK. Fonte (Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos, 2013)

Segundo Cunha, Pereira e Pinto (2013), o *PMBOK* é um modelo geral e, mesmo que não seja voltado especificamente para o desenvolvimento de software, é bastante completo e possui características importantes quando relacionado às metodologias ágeis, como a definição de papéis (por exemplo, o dono do risco, *risk owner)*.

Outros pontos de valores das metodologias ágeis destacados pelos autores (Cunha, R; Pereira, C; Pinto; 2013) são: a comunicação entre os membros da equipe, a aprendizagem contínua da equipe e um modelo que se adapta a diversos ambientes e projetos. Eles mapearam ainda, as seguintes dimensões de risco: complexidade e requisitos do projeto, maturidade da equipe, e controle e planejamento do projeto.

A partir das análises elaboradas diante das três abordagens de gerenciamento de risco estudadas, os autores Cunha, Pereira e Pinto (2013) propuseram a tabela, ilustrada na Figura 8 a seguir. Essa tabela consolida informações sobre os valores ágeis identificados em cada modelo, as dimensões de riscos mapeadas e os pontos fortes dos modelos estudados, a partir da proposta ágil de desenvolvimento de software.

Modelos	Valores ágeis	Dimensões do risco	Pontos fortes (vertente ágil)
Barry Boehm	Aprendizagem, sentido prático, adaptação	Requisitos, complexidade do projeto, planeamento e controlo, equipa	Aprendizagem contínua.
CRM (SEI)	Colaboração, comunicação, aprendizagem, sentido prático, adaptação	Utilizador, requisitos, complexidade do projeto, planeamento e controlo, equipa	Valorização da comunicação, colaboração e trabalho em equipa.
PMBOK *Guide*	Comunicação, aprendizagem, sentido prático, adaptação	Utilizador, requisitos, complexidade do projeto, planeamento e controlo, equipa	Define papéis de utilizadores (risk owner) e adapta-se facilmente a qualquer tipo de projeto.

Figura 8 - Análise dos modelos de gestão de risco. Fonte (Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos, 2013)

Na figura 8 é possível notar a semelhança entre os três modelos estudados: Boehm, *CRM* e *PMBOK.* Todos apresentam os valores ágeis de aprendizagem contínua e adaptação e também as dimensões de risco de requistos e complexidade de projeto, e planejamento e controle de projeto e equipe.

Apesar do *PMBOK* apresentar a comunicação como valor ágil, apenas o modelo *CRM* possui a valorização da comunicação como ponto forte quando comparada a vertente ágil de desenvolvimento de software. Por outro lado, o *PMBOK* é o único dos modelos analisados que apresenta em sua proposta a nomeação de um responsável pelos itens da lista de risco (o dono do risco), ponto relevante quando se trata de metodologias ágeis em que há definição de papéis para cada etapa do desenvolvimento.

Na próxima seção, será apresentada a proposta de modelo de que integrada métodos de gerenciamento de risco e o *SCRUM* criada pelos autores Cunha, Pereira e Pinto (2013) da Universidade do Porto.

Modelo de gestão de risco para projetos de metodogia ágil

O modelo de gestão de risco para projetos de metodologia ágil é composto por dois passos, sendo que o primeiro é realizado apenas uma vez e o segundo é repetido a cada iteração do projeto. O primeiro passo consiste em definir a lista de riscos e o planejamento da gestão do risco. Já o segundo é composto pelas seguintes atividades: definição da lista de riscos, estratégias para minimizar o risco, monitoramento e controle. O processo está descrito na imagem abaixo:

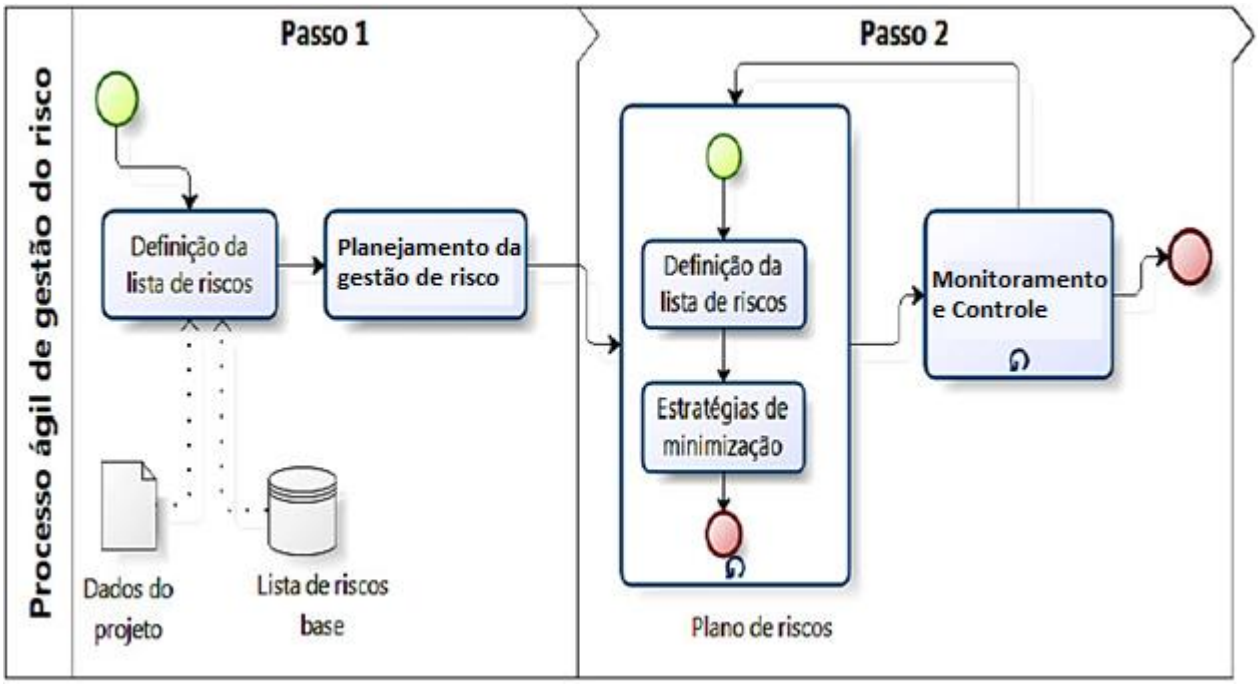

Figura 9 - Modelo proposto pela Universidade do Porto. Fonte (Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos, 2013)

Dados de entrada:

1. Dados do projeto: informações referentes ao projeto, por exemplo, lista de requisitos, orçamento e prazos.
2. Lista de riscos base: base histórica de riscos de projetos anteriores.

Atividades:

1. Definição da lista de riscos: definir a lista de riscos a partir da base histórica de riscos de projetos anteriores e nos dados do projeto;
2. Planejamento da gestão do risco: definir as estratégias de gerenciamento do risco para todo o projeto;
3. Plano de risco: determinar a lista de riscos e das estratégias para minimizar cada item da lista;
4. Monitoramento e controle: realizar o monitoramento, o controle e a aplicação das estratégias para minimizar os riscos. À medida que novos riscos surgem o plano de risco é atualizado.

O estudo concluiu que *PMBOK* é o modelo de gestão de risco que mais aproxima das metodologias ágeis, pois dos três modelos abordados é o único que faz referência a papéis de utilizadores. “É um modelo completo, adapta-se às circunstâncias do projeto e possibilita uma organização de todo o processo mais controle e eficiente, através das fases que se define. ” (CUNHA, R; PEREIRA, C; PINTO, J; 2013, p 5).

Vale acrescentar que Cunha, Pereira e Pinto (2013) incluiram no modelo proposto uma ferramenta de gestão de projetos, dessa forma o modelo poderá ser avaliado com métricas bem definidas e indicadores, que medirão a aplicabilidade do modelo e projetos reais.

2.4.2. Outlining a Model Integrating Risk Management and Agile Software Development

“Outlining a Model Integrating Risk Management and Agile Software Development”, significa, em tradução livre, “esboçando um modelo integrando gestão de risco e desenvolvimento de software ágil”; no decorrer do texto a essa proposta será referenciada como Modelo Integrado. Esse modelo foi proposto pelo Departamento de Computação e Ciência de Sistemas da Universidade de Estocolmo, Suécia, sob autoria de Jaana Nyfjord e Mira Kajko-Mattsson (2008). As autoras estruturam o modelo em conjunto com uma empresa sueca que desenvolve sistemas web para transmissão de programas de televisão via internet, *SVT - I*.

Para propor o modelo integrado, as autoras relatam que estudaram vários cenários de processo ágeis em que a gestão de risco foi aplicada na *SVT – I*. Durante esses estudos puderam mapear quatro pontos de integração entre os modelos (níveis organizacionais e fases do processo, papéis e responsabilidades, canais de comunicação e aspectos do processo) descritos a seguir:

1. Níveis organizacionais e fases do processo: identificação de onde e quando a gestão de risco se encaixa no processo de desenvolvimento;

2. Papéis e responsabilidades: identificação dos responsáveis por gerir os riscos de maneira eficaz;
3. Canais de comunicação: criação de canais de comunicação para integrar as fases do processo e todos os níveis da organização;
4. Aspectos do processo: identificação dos aspectos do processo, que são dinâmicos, e determinação da magnitude da gestão de risco dentro do processo integrado. Considerou-se os seguintes aspectos:

    4.1. Definição de risco: define o significado de risco para facilitar a comunicação na organização;
    4.2. Avaliação do risco: define o conjunto de passos para avaliar o risco com eficácia;
    4.3. Ciclo de vida do software: define as atividades de gestão de risco para cada fase do ciclo de vida do software;
    4.4. Stakeholders, papéis e responsabilidades: define como será a participação dos stakeholders. Os papéis e responsabilidades podem ser determinados por fatores como tamanho do projeto e perfil do risco;
    4.5. Ferramentas de suportes e repositórios: são as ferramentas de suporte
    4.6. Template: define os templates[4] reportar riscos, assim auxiliar na gestão efetiva dos riscos;
    4.7. Status do produto: define a quantidade a ser gasta em gestão de risco para garantir a qualidade do produto;
    4.8. Ambiente: controla como a gestão de risco passar a ser tratada no contexto cultura, social e político da organização;
    4.9. Organização: considera aspectos organizacionais, tais como as atitudes das pessoas em relação riscos, maturidade organizacional, competência e treinamento e seu impacto sobre um programa de gestão de risco.
    4.10. Medidas: toma medidas para fornecer informações sobre a integração do processo de gestão de risco aos demais processos organizacionais.

[4] É um modelo de documento.

A partir dos pontos mapeados, as autoras foram capazes de criar o Modelo Integrado que foi aplicado na empresa SVT – I, parceira no projeto, o que será detalhado na próxima seção.

Modelo Integrado

O Modelo Integrado é o modelo de gestão de risco combinado com as práticas de desenvolimento ágil *SCRUM*, proposto por Nyfjord e Kajko-Mattsson (2008), o qual gerencia todos os riscos encontrados em toda a organização. Conforme ilusta a figura 10, o processo de desenvolvimento ágil é dividido em três camadas principais: plano de visão de produto, *roadmap*[5] do produto e planejamento de versões, e fase de implementação. Todas as camadas são interligadas, mas possuem plano de gerenciamento de risco próprio e estão conectadas a uma camada única, chamada Fórum de Gerenciamento de Risco que possui a função de gerenciar todos os riscos encontrados na organização.

[5] É o mapa do projeto de desenvolvimento de software

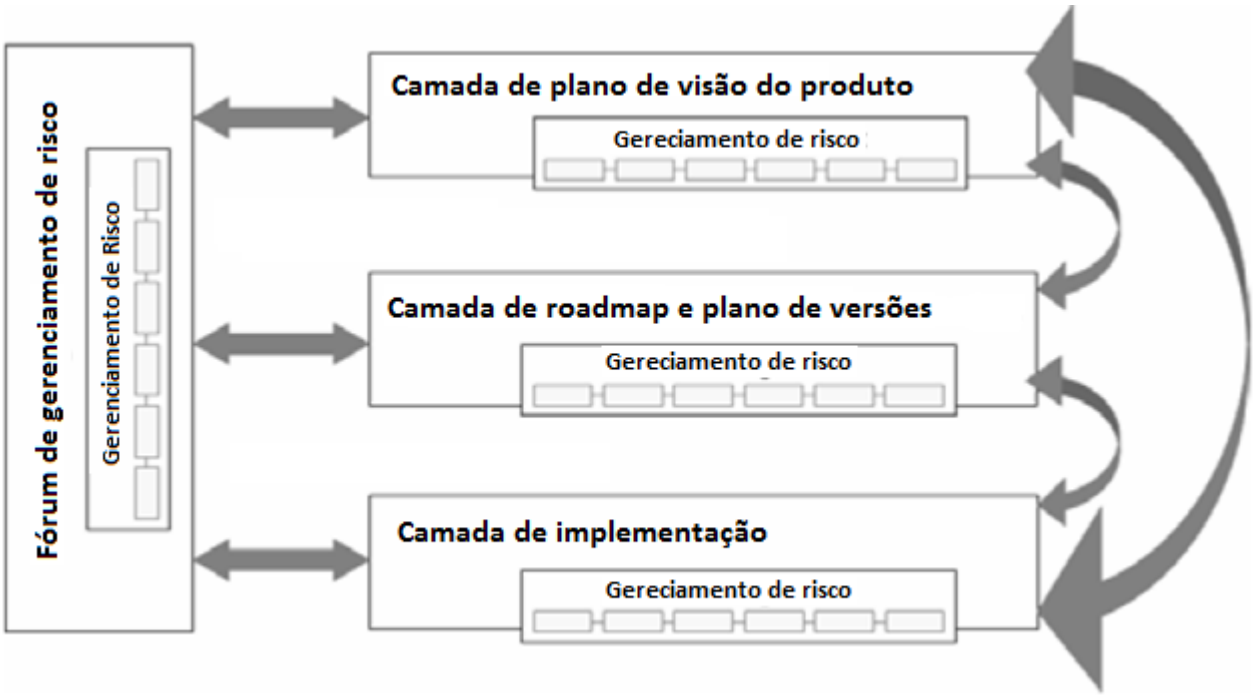

Figura 10 – Visão geral do modelo integrado. Fonte (Outlining a Model Integrating Risk Management and Agile Software Development, 2008)

O modelo integrado foi proposto a partir dos quatro pontos de integração descritos anteriormente.

1. Níveis organizacionais e fases do processo:

O modelo integrado envolve dois níveis organizacionais no processo de gerenciamento de risco: o nível de negócio e o nível de engenheria. A camada de desenvolvimento denominada de plano de visão do produto abrange o nível organizacional de negócio, enquanto o nível de engenharia está sob as camadas de *roadmap* do produto e planos de versões, ou seja, na camada de implementação.

1.1. Camada de plano de visão do produto: Fase voltada para o negócio, criação do produto, plano de trabalho e execução. Gerenciamento de risco está associado ao orçamento e recursos;

1.2. Camada de roadmap do produto e plano de versões: Quando é gerado um roadmap em alto nível do produto e um planejamento de versões, que são

revisitadas a cada início de uma nova versão. O gerenciamento de risco está associado à identificação, à análise e ao planejamento;

1.3. Camada de implementação: Fase em que é definido o plano de trabalho para cada iteração do produto, da qual participam os *stakeholders*, a equipe de desenvolvimento e o gerente de produto. O gerenciamento de risco está associado ao controle e monitoramento dos riscos identificados anteriormente; caso surjam novos riscos, também serão analisados e planejados durante essa fase.

2. Papéis e responsabilidades
   2.1. Fórum de gestão de risco: é responsável por coordenar e supervisionar todos os riscos que envolvam o projeto e participa da tomada de decisão.
   2.2. Gerente de negócio: é o responsável pelos riscos que estão na camada de plano e visão do produto;
   2.3. Gerente de produto: é o responsável pelos riscos da camada de *roadmap* do produto e planejamento de versões;
   2.4. Líder e membros time de implementação: são os responsáveis por gerenciar os riscos na fase de implementação.

3. Canais de comunicação
   3.1. As setas ilustradas na figura 10, que possui as camadas do Modelo Integrado, indicam os canais de comunicação estabelecidos no modelo entre as camadas.

4. Aspectos do processo

Para calibrar os aspectos do processo, as autoras utilizaram a experiência adquirida na SVT – I, chegando às seguintes conclusões:

   4.1. A definição de risco é o principal pré-requisito para a identificação de riscos e para comunicar de maneira eficaz;
   4.2. Os resultados da avalição de risco permitem identificar as ações pertinentes para a atuação sobre risco durante o desenvolvimento;

4.3. O conhecimento do ciclo de vida do software melhora a tomada de decisão na gestão dos riscos;
4.4. A designação de cobertura e das partes interessadas, papéis e das responsabilidades são determinadas por fatores como o perfil de risco do projeto, tipo e tamanho;
4.5. O uso de ferramentas de apoio e de repositórios são determinados por fatores como o perfil de risco do projeto, o tipo, o tamanho e a distribuição da equipe;
4.6. A criação de *templates* para formalizar a geração de riscos está associada ao tipo do projeto, ao tamanho e à distribuição da equipe;
4.7. O status do produto, que considera a expectativa de vida e valor de negócio, ajuda a determinar o montante do processo de gestão de risco necessário;
4.8. O ambiente e o contexto físico do projeto determinam a formalidade do processo de gestão de riscos, isto é, quanto maior a empresa ou o projeto, maior será a necessidade de possuir um processo de gerenciamento de risco mais robusto (a quantidade de equipes de desenvolvimento, equipes de desenvolvimento em diferentes locais geográficos, etc.);
4.9. A maturidade e treinamento das pessoas que compõem a organização auxiliam na adoção de um programa de gestão de risco.

Após aplicação do Modelo Integrado na empresa SVT – I, o estudo concluiu que, de maneira geral, o modelo proposto atingiu as expectativas. Porém identificou-se que o modelo integrado é válido em algumas circunstâncias, tais como: quando as equipes são compostas por mais de dez pessoas, para equipes distribuídas, para o desenvolvimento de módulos de segurança, sistemas embarcados, sistemas complexos, projetos inteiramente novos ou inovadores, projetos com exposição de risco elevado.

Segundo as autoras Nyfjord e Kajko-Mattsson (2008), a necessidade de gestão de risco depende de vários fatores, quais sejam: o tamanho do projeto, o tipo de desenvolvimento e a complexidade do produto. Essas autoras relataram algumas preocupações quanto ao modelo, por exemplo, quanto à função do fórum de gerenciamento de risco, pois este entraria em conflito com preceitos de metodologia ágil.

2.5. Comparação entre os modelos apresentados

Comparando os modelos criados pelos autores Cunha, Pereira, Pinto (2013) da Universidade do Porto e pelas autoras Nyfjord e Kajko-Mattsson (2008) da Universidade de Estocolmo, é possível observar algumas semelhanças: ambos propõem uma etapa de identificação dos riscos, de análise dos riscos, de planejamento para minimizar o impacto, e de monitoramento e controle, da mesma forma que os modelos de gerenciamento de risco encontrados na literatura.

As duas propostas se aproximam do modelo ágil de desenvolvimento de software ao propor que as listas de riscos sejam revisitadas à medida que um novo ciclo de desenvolvimento dos projetos se inicia. Além disso, ambas preveem uma atividade cíclica com objetivo de identificar novos riscos e traçar estratégias de minimizá-los durante a fase de desenvolvimento. O modelo proposto por Cunha, Pereira e Pinto (2013) sugere que essa atividade ocorra no segundo passo de seu processo, ao passo que o modelo proposto por Nyfjord e Kajko-Mattsson (2008) prevê que seja realizada na camada de implementação.

O estudo apresentado pela Universidade do Porto, Cunha, R; Pereira, C; Pinto, J (2013), classifica o *PMBOK* como o modelo de gestão de risco que mais se aproxima das necessidades de um projeto de software utilizando metodologia ágil, ainda que o modelo do *PMBOK* não seja específico para projetos de software. Um dos argumentos para essa classificação é que o *PMBOK* define papéis de utilizadores, o dono do risco; entretanto, no modelo apresentado não há declaração de papéis e responsabilidades e nem uma associação das atividades descritas no processo aos membros da equipe *SCRUM*.

O processo de gestão de risco proposto pelas autoras Nyfjord e Kajko-Mattsson (2008) prevê os papéis e responsabilidades de todos os membros da equipe. Para cada camada do processo de desenvolvimento há um “dono do risco”; para camada de negócio, o responsável é o gerente de negócio; para a camada de produto o responsável é o gerente de produto, ao passo que para a camada de implementação os responsáveis são os líderes e os membros do time de desenvolvimento.

Como forma de comunicação, o processo esboçado por Nyfjord e Kajko-Mattsson (2008) prevê uma camada, o fórum de gerenciamento de risco, que se comunica com as demais camadas do modelo. Esse fórum é responsável por coordenar, supervisionar e tomar decisão sobre todos os riscos encontrados em todas as fases e camadas do processo de desenvolvimento de software. Essa camada fora questionada pelos membros da empresa *SVT – I*, empresa que auxiliou na calibragem e desenvolvimento do processo, pois infringia os valores ágeis; uma vez que o *SCRUM* propõe que a equipe seja auto gerenciada, logo as tomadas de decisão são feitas internamente.

Quanto aos resultados, o modelo desenvolvido por Cunha, Pereira e Pinto (2013) não fora testado em projetos reais, dessa forma, não publicaram os resultados. O processo passará por uma ferramenta de gestão de projetos que medirá os indicadores de aplicabilidade do processo de gestão de riscos.

As autoras Nyfjord e Kajko-Mattsson (2008) conseguiram atingir o objetivo proposto, ou seja, gerou um modelo de gestão de risco associado aos valores de ágeis, suprindo, dessa forma, uma carência na gestão dos projetos de software que utilizam metodologia ágil de desenvolvimento. Todavia, esse modelo adapta-se melhor em projetos que possuam algumas características específicas: uma equipe com no mínimo dez pessoas, projetos inteiramente novos e projetos que apresentem alto risco.

2.6. Considerações do capítulo

Esse capítulo apresentou conceitos de metodologia ágil, descreveu sua origem e citou os valores apresentados no Manifesto Ágil, bem como alguns processos de desenvolvimento ágil. Detalhou-se o *SCRUM*, um dos métodos ágeis mais utilizados na atualidade, descrevendo-se as atividades realizadas pelos membros da equipe, os eventos previstos e os artefatos gerados. Seguindo o desenvolvimento, o modelo de gestão de risco, *PMBOK*, foi descrito a partir das principais atividades, desde o planejamento da gestão dos riscos até a etapa de monitoramento e controle dos riscos identificados.

Posteriormente, apresentou-se dois modelos de gestão de risco baseado nos princípios da metodologia ágil: um modelo proposto por autores (CUNHA, R; PEREIRA, C; PINTO, J ,2013) da Universidade do Porto, em Portugal e outro por autoras (NYFJORD, J; KAJKO-MATTSSON, M, 2008) da Universidade de Estocolmo, na Suécia. Após apresentá-los, uma seção dedicou-se a comparar os dois modelos descritos, considerando as semelhanças e divergências entre eles.

3. Proposta de detalhamento para o modelo apresentado pelos autores Cunha, Pereira e Pinto (2013)

3.1. Apresentação

A figura 11 representa a alteração realizada no modelo proposto pela Universidade do Porto.

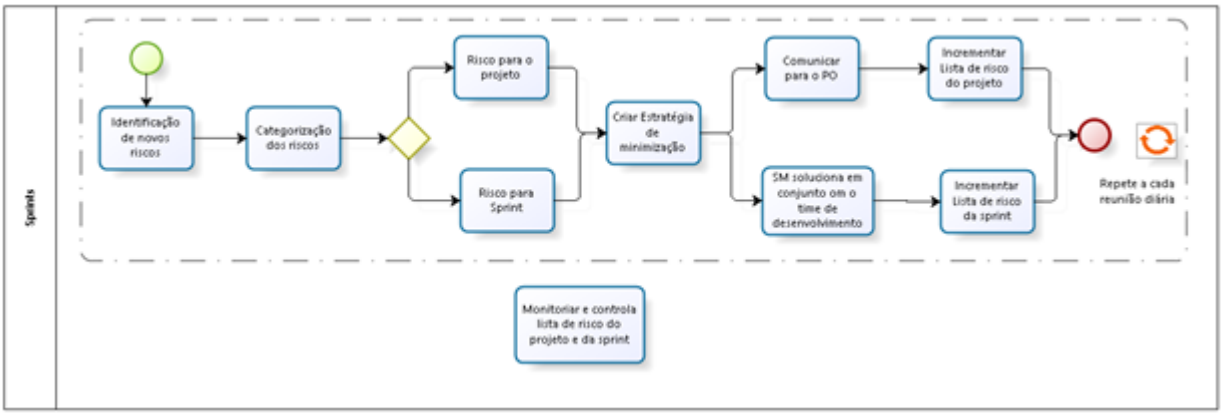

Figura 11 - Modelo proposto do detalhamento da gestão de risco nas reuniões diárias

A proposta de modelo é baseada no modelo proposto por Cunha, Pereira e Pinto (2013) da Universidade do Porto. Da mesma forma como proposto pela literatura de gerenciamento de risco, o modelo possui as seguintes etapas: planejamento da gestão de risco, identificação dos riscos, análise dos riscos, controle e monitoramento dos riscos.

As três primeiras fases são realizadas durante as fases iniciais de projeto, considerando cronograma, custo, avaliação histórica dos riscos de projetos anteriores, análise dos requisitos do projeto atual (dados que utilizados como entrada para criação da lista inicial de riscos ao projeto). Todos os membros da equipe *SCRUM* participam dessa etapa inicial de criação da lista inicial de riscos.

A partir da lista de riscos identificados, é necessário o processo de análise de impacto dos riscos e estratégicas para minimizar os riscos; esses dados resultaram o plano de gestão de riscos que, como proposto pelo *PMBOK* (2013) é necessário que seja designado um responsável para monitorar e controlar os riscos. No modelo proposto

(Figura 11) há uma tarefa de categorização do risco: os riscos são separados em riscos de projeto e risco da *SPRINT*. Os riscos de projeto devem sem geridos pelo “dono do produto”, enquanto os riscos da *SPRINT* devem ser geridos pelo *SCRUM MASTER,* que deve atualizar a lista de riscos e reportar novos riscos encontrados, assim como as estratégias para minimizá-los, para o “dono do projeto”.

As atividades de monitoramento e controle dos riscos são realizadas durante as fases de execução do projeto, isto é, durante as *SPRINTS* dos projetos de desenvolvimento de software. Na reunião diária, realizada entre os membros da equipe de desenvolvimento e o *SCRUM MASTER*, o *SCRUM MASTER* identifica se novos riscos são apontados pela equipe de desenvolvimento e acompanha o status dos riscos já reportados.

Caso novos riscos sejam apontados, as etapas de análise, categorização, criação de plano de risco devem ser realizadas. Conforme a figura 11, as etapas a serem realizadas durante a *SPRINT* são:

1. Identificação de novos riscos: a identificação de novos riscos ocorre durante as reuniões diárias. O *SCRUM MASTER* deve identificar junto com a equipe de desenvolvimento se há riscos nas atividades a serem executadas.
2. Categorização dos riscos: os riscos são categorizados segundo a tabela 1, proposta no estudo realizado pelo autor Arnuphaptrairong (2011).
   2.1. Riscos de projetos: Sãs os riscos que não estão relacionados diretamente a fase de desenvolvimento de software, porém, que impactam, ainda assim, a continuidade do projeto ou da *SPRINT*. São os riscos que possuem as dimensões: usuário, planejamento e controle e ambiente organizacional;
   2.2. Riscos da *SPRINT*: São os riscos que estão relacionados diretamente à fase de desenvolvimento de software. São os riscos que possuem as dimensões: requisitos do projeto, complexidade do projeto e equipe.
3. Criar estratégia de minimização: como previsto no *SCRUM*, quando há identificação de possíveis problemas todos os membros da equipe de desenvolvimento em conjunto com o *SCRUM MASTER* devem traçar a estratégica de prevenção em conjunto, visto que não há tomadas de decisão de maneira hierárquica. O plano de risco será atualizado com as decisões tomadas pela equipe *SCRUM*.

4. Incrementar lista de riscos do projeto: O *SCRUM MASTER* deve reportar ao “dono do produto” e atualizar a lista de riscos do projeto. Esse incremento necessariamente contém as estratégias criadas no passo anterior do plano de risco.
5. Incrementar lista de riscos da *SPRINT*: O *SCRUM MASTER* deve incrementar a lista de risco da *SPRINT*, a qual inclui o plano de risco com as estratégias geradas no passo 3. Essa lista será revisitada na reunião de retrospectiva em que os riscos serão reportados aos *stakeholders* e ao “dono de produto”. Por sua vez, os riscos apontados devem ser incorporados a lista de risco do projeto, para compor base histórica para futuras *SPRINTS* e projetos.

3.2. Apresentação do estudo de caso

O modelo proposto pelo trabalho, descrito na seção anterior, foi aplicado em um projeto de desenvolvimento de software realizado no Departamento de Tecnologia de Informação de um Instituto que atua com pesquisa, administração e planejamento junto à prefeitura de uma cidade do interior do estado de São Paulo. O projeto possuía as seguintes características:

1. Projeto de indicadores para uma cidade do interior do estado de São Paulo. O objetivo do projeto é desenvolver uma plataforma web que apresente relatórios de abstinência, compras, financeiros e da central de atendimento aos cidadãos. Os relatórios devem ser por secretaria;
2. A plataforma será utilizada pelo prefeito e seus secretários;
3. O projeto possui integração entre tecnologias: uma tecnologia utilizada para desenvolvimento web e outra ferramenta de inteligência empresarial (*business intelligence*) utilizada para realização de processamento de dados e geração de relatórios;
4. Durante a reunião de apresentação do projeto foi decidido que o padrão de desenvolvimento adotado seria em três camadas utilizando *web forms*;
5. A equipe é formada por três analistas sendo um dos analistas representa o papel do *SCRUM MASTER*, um analista de banco de dados, um analista de inteligência empresarial e o gerente de projetos o qual representa o “dono do produto”;

6. O cronograma inicial do projeto teria duração de seis meses.

3.2.1. Identificação dos riscos da reunião inicial do projeto

1. O projeto possuiu envolvimento de *stakeholders* externos ao Instituto, por exemplo, os analistas da prefeitura responsáveis pelo projeto. Esses *stakeholders* não possuíam um interesse geral pelo projeto, itens quatro e cinco da tabela 1 da dimensão de usuário;
2. Uma consultoria externa foi contratada para projetar as interfaces com usuário; porém não foi feito um teste de integridade com a ferramenta de inteligência empresarial que disponibiliza dos relatórios administrativos;
3. Três dos cinco analistas não possuíam conhecimento em desenvolvimento em três camadas utilizando *web forms*, porém a decisão de desenvolver nesse padrão foi tomada por ser uma forma de integração proposta pela ferramenta de inteligência empresarial. Item vinte e um da tabela 1;
4. Os requisitos inicias do projeto não eram bem especificados, mas à medida que as *SPRINTS* fossem iniciadas, os requisitos seriam refinados, conforme proposto pelo *SCRUM.* Itens oito e nove da tabela 1;
5. Durante a apresentação um possível risco de integração entre as plataformas de ferramenta de negócio empresarial e de desenvolvimento *web* fora apontado, pois não havia definição técnica de como seria essa integração, uma vez que a própria documentação da ferramenta de inteligência empresarial propunha mais de uma forma de realizar a integração, porém não era sabido pela equipe a melhor maneira de realizar a integração.

3.2.2. Plano de gestão dos riscos

1. Para o primeiro risco encontrado, as decisões que caberiam aos *stakeholders* externos foram tomadas pelo “dono do produto”. Isto é, coube ao “dono do produto” decidir quais os relatórios seriam exibidos, a forma como estariam dispostos na tela e como seriam as navegações entre os menus de relatórios. Portanto, o “dono do produto” tornou-se o “dono do risco”;
2. Para a integração entre as plataformas de desenvolvimento e de negócio empresarial fora reservado, em cronograma, um mês para realização do estudo,

com o objetivo de realizar a integração com a melhor solução técnica. Logo, o *SCRUM MASTER* tornou-se o “dono do risco”, porém, as atividades seriam desempenhas pelo analista responsável pela ferramenta de negócio empresarial;

3. Durante o primeiro mês de projeto, fora previsto um treinamento de desenvolvimento *web* utilizando *web forms* para equipe de desenvolvimento. O *SCRUM MASTER* tornou-se o responsável pelo risco;
4. Os requisitos foram mapeados em entregas planejadas, dessa forma criaram-se as *SPRINTS*. Ao todo foram planejadas 6 entregas, sendo que última seria a versão final do software funcionando. Entre uma *SPRINT* e o outra haveria o refinamento dos requisitos. O “dono do produto” tornou-se responsável pelo risco;
5. Durante a análise da proposta de *layout* realizada pela consultoria terceirizada, encontrou-se pontos que a plataforma de desenvolvimento de negócio empresarial não suportava. Esses itens foram enumerados e retirados da proposta, com a justificativa de incompatibilidade técnica. Logo, o “dono do produto” tornou-se responsável pelo risco.

3.2.3. Monitoramento e controle

Na primeira *SPRINT*, conforme apresentado, o monitoramento das atividades de controle dos riscos foi realizado durante as reuniões diárias entre os *SCRUM MASTER* e a equipe de desenvolvimento. Durante as reuniões foram mapeados riscos de alteração de requisito, alteração da estrutura de navegação da tela. Por estar no início do projeto, as alterações foram incorporadas à *SPRINT*, porém como medida de solução para essas alterações, alguns membros da equipe de desenvolvimento necessitaram trabalhar além do previsto, por causa das alterações solicitadas. Essas alterações foram agregadas à lista de risco da *SPRINT* pelo *SCRUM MASTER* e, posteriormente na reunião de retrospectiva, incorporada à lista de risco do projeto. Houve, então, a comunicação ao *stakeholders*.

3.2.4. Resultados

Na reunião de definição da segunda *SPRINT*, houve uma reorganização estrutural da empresa, o então diretor do Instituto fora convidado para assumir o cargo de secretário de gabinete da prefeitura da cidade. Dessa forma, o diretor tornou-se o principal *stakeholder* do projeto e solicitou que o projeto fosse entregue em até um mês, prazo em que ele assumiria o cargo junto à prefeitura.

Com isso, houve uma reformulação na estrutura do projeto. Algumas atividades até então previstas no cronograma foram desconsideradas, por exemplo, treinamento da equipe de desenvolvimento, para outras atividades criou-se num novo plano de execução, como a documentação formal de requisitos – seriam realizadas após finalização do projeto. Além disso, foi esboçado um plano de contingência, necessitando que a equipe trabalhasse em regime de horas extras para suprir a demanda do projeto.

Por causa dessa movimentação estrutural do Instituto, tornou-se inviável continuar a aplicar o modelo de gestão de risco proposto, pois não era possível realizar reuniões diárias formais para discutir as atividades. Os riscos que surgiram ao longo do desenvolvimento, tais como, falta de ambiente para realizar os testes das ferramentas de negócio empresárial e da aplicação web desenvolvida foram corrigidos conforme foram surgindo, sem que houvesse a classificação e a incorporação às listas de risco tanto da *SPRINT* quanto do projeto.

Entretanto, todos os pontos mapeados durante a fase de desenvolvimento, foram atualizados a lista de risco do projeto na reunião de retrospectiva do mesmo. Esses pontos passaram, assim, a incorporar a base histórica para futuros projetos.

Apesar de não ter sido possível aplicar o modelo ao longo do projeto, o gerente de projetos do Instituto esclareceu que, ao dividir a responsabilidade de gestão do risco com o *SCRUM MASTER,* tornou suas atividades mais objetivas, pois não foi necessário que ele atuasse em todas as tomadas de decisão para solucionar um risco de *SPRINT*.

3.3. Considerações do capítulo

Este capítulo apresenta um modelo de detalhamento do processo descrito pela Universidade do Porto. No processo descrito há a nomeação dos responsáveis pelos riscos, atividade prevista pelo *PMBOK*. Caso os riscos fossem classificados como risco da *SPRINT* o *SCRUM MASTER* se tornaria o “dono do risco”, enquanto se o risco for classificado como risco do projeto, o “dono do produto” seria o responsável pelo risco.

Posteriormente, o capítulo descreve uma aplicação do modelo proposto. Modelo foi aplicado em um projeto de software de um Instituto de pesquisa de uma cidade do interior de São Paulo. Esse projeto teve o objetivo de atender os secretários e prefeito da cidade. Após a primeira *SPRINT*, houve uma alteração organizacional do Instituto, a qual inviabilizou a continuidade da aplicação do modelo. Apesar de o modelo não ter sido testado durante todo o projeto, o gerente de projetos do Instituto responsável aprovou a divisão de responsabilidade entre ele e o *SCRUM MASTER*.

4. Considerações finais

Podê-se observar que, com a inserção de atividades relacionadas à gestão de risco durante as atividades previstas pelo *SCRUM*, tornou-se possível incorporar a gestão de riscos em um modelo de desenvolvimento ágil.

No trabalho foram comparados e discutidos dois modelos de gestão de risco. O primeiro da Universidade do Porto, Portugal, intitulado de “Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos”, que apresentou um estudo comparando a proposta de gestão de risco de diferentes modelos: Barry Bohem, *Continuous Risk Manegement* e *PMBOK* e um modelo de gestão de risco baseado no *SCRUM*.

O segundo trabalho estudado, foi realizado pela Universidade de Estocolmo, Suécia, “esboçando um modelo integrando gestão de risco e desenvolvimento de software ágil” que propôs um modelo mais maduro do que o da Universidade do Porto, visto que fora aplicado junto a uma empresa de desenvolvimento de software que utilizava o *SCRUM*. Porém, enfrentou algumas dificuldades, pois os membros da equipe argumentaram que a função do fórum de gerenciamento confrontaria com preceitos do *SCRUM.*

A partir da discussão de ambos os modelos fez-se a reflexão sobre o detalhamento do processo de gestão de risco para o modelo de gestão de risco sugerido pela Universidade do Porto. A principal alteração na proposta foi a definição dos papéis, o “dono produto” tornou-se o “dono dos riscos” relevantes ao projeto, ao passo que o *SCRUM MASTER* tornou-se o responsável pelos riscos que ocorreram durante a execução da *SPRINT*. Houve também a incorporação da classificação dos riscos, seguindo as dimensões de usuário, equipe, requisitos, planejamento e controle e ambiente organizacional.

A aplicação do modelo proposto teve início em um projeto de criação de uma plataforma web que mostraria os indicadores financeiros, de administração e da central de atendimento aos cidadãos para a prefeitura de uma cidade do interior de São Paulo, porém não foi possível aplicá-lo durante todas as fases do projeto, por um motivo de reestruturação do Instituto. Apesar de não ter sido aplicado durante todas as fases do

projeto, o gerente de projeto do Instituto que o modelo foi aplicado aprovou a separação dos riscos, entre o “dono do produto” e o *SCRUM MASTER*, dividindo, assim, a responsabilidade tornando suas intervenções mais objetivas, pois atuava em questões relacionadas ao âmbito do projeto. As demais atividades previstas no modelo precisam ser testadas para análise de viabilidade do modelo.

4.1. Trabalhos futuros

Os estudos de aplicação de gestão de risco a projetos que utilizam metodologia ágil não se esgotam nesse trabalho. É necessário aplicar o modelo proposto em outros projetos, para colher informações que possam ser aplicadas ao processo e dessa forma melhorá-lo.

É possível pensar em aplicar o modelo de gestão de risco em outros modelos de desenvolvimento ágil, por exemplo, o XP, visto que o detalhamento proposto está associado à fase de execução do projeto e não apenas à fase inicial de planejamento.

A partir da atividade de classificação dos riscos prevista no modelo, é possível iniciar estudos sobre os perfis dos riscos que ocorrem nos projetos de software em empresas brasileiras que desenvolvem softwares e, dessa forma, criar uma tabela de riscos baseada no cenário nacional de desenvolvimento de software.

REFERÊNCIAS

ALHAWARI, S; THABTAH, F; KARADSHEH, L; MUSA HADI, W. **A Risk Management Model for Project Execution**, 2008.

ARNUPHAPTRAIRONG, T. **Top Ten Lists of Software Project Risks: Evidence from the Literature Survey**, 2011.

BANNERMAN, P. **Risk and risk management in software projects: A reassessment**, 2008.

BOEHM, B. **Software Risk Management: Principles and Practices**, 1991.

BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; ARICK, B.; MARTIN, R.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. **Manifesto for Agile Software Development,** 2001. Disponível em: <http://www.agilemanifesto.org>. Último acesso em: 02 mai. 2015.

COHN, M. Disponível em: http://www.mountaingoatsoftware.com/agile/scrum, acesso em: 6 dez 2014.

CHARETTE, R.N. **Why software fails?,** 2005.

CUNHA, R; PEREIRA, C; PINTO, J. **Projetos ágeis de desenvolvimento de software: proposta de modelo de gestão dos riscos**, 2013.

HUNDHAUSEN, R. **Professional Scrum Development with Microsoft Visual Studio 2012**. 1st ed. Redmond: Microsoft Press, 2012.

NYFJORD, J; KAJKO-MATTSSON, M. **Outlining a Model Integrating Risk Management and Agile Software Development**, 2008.

PMI - Project Management Institute. PMBOK® **Guide: A Guide to the Project Management Body of Knowledge**. 5th ed., Newton Square: PMI Publications, 2013.

PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional.** 7. ed. São Paulo: McGraw Hill, 2011.

RUBIN, K. **Essential Scrum: A Practical Guide to the Most Popular Agile Process.** 1st ed. Michigan: Addison-Wesley, 2012.

SCHWABER, K. **Agile Project Management with Scrum**. 1st ed. Redmond: Microsoft Press, 2004.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide: The Definitive Guide to Scrum**. 2013. 16 p. Disponível em: <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf>. Acesso em: 7 dez. 2014.

SIMISTER, S.J. **Qualitative and quantitative risk management**, 2004.

SMITH, G; SIDKY. **A. Becoming Agile, in an imperfect world**, 2009.

SOMMERVILLE, I. **Software Engineering**. 9th ed. Boston: Addison-Wesley, 2011.

WARD, S., CHAPMAN, C. Making risk management more effective, 2004.

YIN, R K. **Case Study Research, Design and Methods**. 5th ed, 2014.

APÊNDICE 1 – TABELA DE DIMENSÕES DE RISCO

Esse apêndice foi desenvolvido pelo autor Arnuphaptrairong em 2001 e está diposnível no artigo: Top Ten Lists of Software Project Risks: Evidence from the Literature Survey.

Dimensão do risco	Abreviação	Risco do software
Usuário	User1	Usuários com resistência a mudanças
	User2	Conflitos entre usuários
	User3	Usuários com atitudes negativas em relação ao projeto
	User4	Usuário não engajados ao projeto
	User5	A falta de colaboração dos usuários
Requisitos	Reqm1	Alterações contínuas de requisites
	Reqm2	Requisito do sistema não estão devidamente identificados
	Reqm3	Requisitos do sistema pouco claros
	Reqm4	Requisitos do sistema incorretos Projetos envolve nova tecnologia
Complexidade do projeto	Comp1	Projeto com alta complexidade técnica
	Comp2	Tecnologia imatura
	Comp3	Projeto envolve tecnologias que não foram utilizados em projetos anteriores

	Comp4	Falta de ferramentas de gestão de projeto eficaz
Planejamento e Controle	P&C1	Falta de ferramentas de gestão de projeto eficaz
	P&C2	O monitoramento do projeto não realizado adequadamente
	P&C3	Estimativa errada dos recursos necessários
	P&C4	Planejamento de projeto fraco
	P&C5	Indefinição dos detalhes dos projetos
	P&C6	Gerente de projeto inexperiente
	P&C7	Comunicação ineficiênte
Time	Team1	Equipe de desenvolvimento inexperiente
	Team2	Treinamento inadequado dos membros da equipe de desenvolvimento
	Team3	Equipe de desenvolvimento não possuis as habilidades técnicas exigida pelo projeto
Ambiente organizacional	Org1	Mudança na gestão da organização durante o projeto
	Org2	Políticas coorporativas que afetam o projeto negativamente
	Org3	Ambiente organizacional instável
	Org4	Reestrutução organização durante o projeto

Tabela 1 - Tabela de dimensões de risco. Fonte (Top Ten Lists of Software Project Risks: Evidence from the Literature Survey, 2011)