Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proposta de Melhoria para o Processo de Desenvolvimento de Software do Exército Brasileiro com base no Modelo de Gestão de Risco e na Metodologia Ágil

Ricardo Alves Moraes

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof.ª Dr.ª Simone Borges Simão Monteiro

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Pós-graduação em Computação Aplicada

Coordenador: Prof. Dr. Marcelo Ladeira

Banca examinadora composta por:

Prof.ª Dr.ª Simone Borges Simão Monteiro (Orientadora) — EPR/UnB
Prof. Dr. Rodrigo Bonifácio de Almeida — CIC/UnB
Prof. Dr. Carlo Kleber da Silva Rodrigues — Exército Brasileiro

CIP — Catalogação Internacional na Publicação

Moraes, Ricardo Alves.

Proposta de Melhoria para o Processo de Desenvolvimento de Software do Exército Brasileiro com base no Modelo de Gestão de Risco e na Metodologia Ágil / Ricardo Alves Moraes. Brasília : UnB, 2015.
199 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2015.

1. Desenvolvimento de *Software*, 2. Método Ágil, 3. Gestão de Risco

CDU M827p

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Proposta de Melhoria para o Processo de Desenvolvimento de *Software* do Exército Brasileiro com base no Modelo de Gestão de Risco e na Metodologia Ágil

Ricardo Alves Moraes

Dissertação apresentada como requisito para conclusão do
Mestrado Profissional em Computação Aplicada

Prof.ª Dr.ª Simone Borges Simão Monteiro (Orientador)
EPR/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
CIC/UnB

Prof. Dr. Carlo Kleber da Silva Rodrigues
Exército Brasileiro

Prof. Dr. Marcelo Ladeira
Coordenador do Programa Pós-graduação em Computação Aplicada

Brasília, 11 de junho de 2015

Dedicatória

À Deus, por sua onipresença,
Aos meus pais José Evilásio (in memory) e Maria de Lourdes,
à minha esposa Shélida, e em especial a minha Filha Laura que é o anjo que Deus colocou em minha vida, razão da minha existência e de toda a minha felicidade,
a todos os meus irmãos, em especial, Juliana.

Agradecimentos

Em primeiro lugar inicio meus agradecimentos a Deus, por sempre ter orientado meus passos para que chegassem a este momento.

De forma especial, gostaria de citar algumas pessoas que, de alguma forma, me apoiaram, me compreenderam e me motivaram para desenvolver e concluir este projeto.

A minha orientadora Profª Drª Simone Borges Simão Monteiro, pelo incentivo e orientação, acreditando no meu potencial como profissional e lembrar-me que só temos liberdade de escolha quando temos mais de uma alternativa.

Aos colegas do MPCA2012, pela convivência neste período, cada qual buscando o seu melhor em seus trabalhos e com esperança no futuro. E em especial aos grandes amigos Misael Araujo, Antonio Ferreira, Arthur Alves, Jackson Freitas, Rogério Conceição e Rubens Santos pelas conversas que contribuíram com certeza na conquista deste projeto.

Ao Departamento de Ciência da Computação por aceitar a proposta de pesquisa e aos Professores do Programa, por terem colaborado no aprimoramento de meus conhecimentos, em especial o Prof. Marcelo Ladeira que contribuiu sobremaneira para à conclusão do meu projeto.

As Administrações Públicas que permitiram fazer a troca de informações, possibilitando uma integração dos conhecimentos, em especial o Exército Brasileiro representado pela Chefia do Centro de Desenvolvimento de Sistemas, Gen. Brig. Bráulio de Paula Machado, unidade militar que proporcionou a aplicação dos conhecimento adquiridos ao longo da jornada percorrida, sempre com foco na melhoria dos serviços prestados pelo Centro.

A todos que de uma forma ou outra contribuíram com a realização deste trabalho e a convivência neste período, destacando os amigos do Curso de Engenharia de Produção representados pela Roberta Ramos, Iana Giesbrecht, Vitor Brixi, Luís Henrique, Leandro Batista e Renan Holdorf e em especial os grandes amigos Coronel Gladistone e Cauê Zaghetto.

Resumo

Devido a crescente demanda de desenvolvimento de *software* na Administração Pública Federal (APF), a utilização da metodologia ágil com a abordagem de gestão de riscos passou a ser uma alternativa estratégica para a organização, uma vez que possibilita identificar e mitigar os possíveis riscos e traz uma maior celeridade ao processo de desenvolvimento. O sucesso do desenvolvimento ágil está relacionado à seleção apropriada dos métodos e técnicas disponíveis. Segundo a literatura e as boas práticas, não há uma única técnica ou método para uso em desenvolvimento ágil, bem como na gestão de riscos. Neste sentido, o objetivo desta pesquisa é propor um processo de desenvolvimento de *software* e gestão dos riscos associados do Exército Brasileiro fundamentado na metodologia ágil a luz das práticas adotadas pela Administração Pública Federal. O estudo inicia-se com uma revisão bibliográfica onde são tratados os conceitos envolvidos no desenvolvimento de *software*, com o estudo das metodologias, a seguir parte da observação de órgãos federais que utilizam o método ágil como fundamento de desenvolvimento de *software*, e de acordo com a adequação das metodologias ágeis utilizadas por diversos autores, apresenta-se uma análise qualitativa das boas práticas do uso da metodologia ágil adotada pelos órgãos federais. A pesquisa finaliza com uma proposta de processo de desenvolvimento de *software* ágil e os métodos para mitigação de riscos para o EB.

Palavras-chave: Desenvolvimento de *Software*, Método Ágil, Gestão de Risco

Abstract

Because of software development demand in the Federal Public Administration (APF), the use of agile methodology with the risk management approach has become a strategic alternative for the organization as it helps identify and mitigate possible risks and brings greater speed the development process. The success of agile development is related to the selection of appropriate methods and techniques available. According to the literature and best practice, there is no single technique or method for use in agile development and risk management. In this sense, the objective of this research is to propose one software development process and management of risks associated with the Brazilian Army (EB) based on the agile methodology in light of the practices adopted by the Federal Public Administration. The study starts with a literature review where the concepts regarding the software development are presented first, followed by the study of methodologies, then by part of the observation of federal agencies processes that use agile software development method, lately according to the adequacy of agile methodologies used by different authors. It is presented a qualitative analysis of best practices of using agile methodology adopted by these federal agencies. The research concludes with a proposal for Agile Software Development process and methods to mitigate risks to the EB.

Keywords: Software Development, Agile Method, Risk Management

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo descreve a contextualização do desenvolvimento de *software* na Administração Pública Federal (APF), e apresenta os objetivos pretendidos, bem como as justificativas que fundamentam a relevância da pesquisa.

1.1 Contextualização

Com o crescimento e a complexidade das organizações, o uso de serviços de Tecnologia da Informação (TI) vem se tornando uma estratégia cada vez mais utilizada para garantir a continuidade da missão organizacional dos diversos órgãos da Administração Pública Federal (APF).

O mercado Mundial apresenta um crescimento de cerca de 4,8% no consumo de *software* e serviços de TI em relação a 2012. Nesse cenário, o Brasil passou a ocupar a sétima posição no *ranking* mundial, com crescimento da ordem de 15,4%, segundo a pesquisa realizada no ano de 2013, dos quais o setor de *software* e serviço de TI obtiveram crescimento de 13,5%, de um cenário de consumo de 61,6 bilhões de dólares, sendo 10,7 bilhões de dólares em *software* e 14,4 bilhões de dólares em serviços relacionados a TI (ABES, 2014) [1].

No cenário brasileiro o consumo de *software* e serviços é explorado por cerca de 11.230 empresas dedicadas ao desenvolvimento, produção, distribuição de *software* e prestação de serviços de TI. O consumo do setor público é cerca de 8,7%, totalizando um volume de 1.061 serviços de TI, com um reflexo de crescimento de mais de 1,1% do consumido em 2012 (ABES, 2014 [1].

No atual ambiente de desenvolvimento de *software*, os requisitos são alterados frequentemente durante o ciclo de desenvolvimento do produto a fim de atender às alterações da demanda (RISING e JANOFF, 2000) [53]. Considerando ainda, seus recursos restritos o desenvolvimento de *software* se torna um desafio.

Os serviços de TI se tornou um dos principais eixos de proteção dos interesses das instituições públicas (BRASIL, 2013b) [18], e com a sua adequada elaboração, negociação e administração, pode contribuir para a preservação de ativos estratégicos, a segurança da operação, a neutralização de responsabilidades legais da pessoa jurídica contratante ou contratada e de seus gestores, o atendimento de normas regulatórias e de normas técnicas, a implantação de *frameworks* de Gestão e Governança de TI baseados em modelos consagrados e para a realização de auditorias.

A crescente demanda de TI tem levado as instituições a repensarem seus processos de gestão de produtos e serviços de TI, uma vez que as necessidades extrapolam sua capacidade em prover meios de TI com os Recursos Humanos disponíveis em seus quadros. Desta forma, a busca por serviços terceirizados, ou por contratação externa, se torna uma alternativa frequentemente adotada pelas organizações.

Associado a crescente demanda de desenvolvimento de *software* para Saavedra *et al* (2014) [54] os *software* são desenhados para atender as necessidades do cliente, porém essas necessidades evoluem com o tempo devido a ocorrência de mudanças no contexto operacional do mercado ou da tecnologia. Outrossim, os software que são projetados de forma adaptável conseguem perpetuar o atendimento dessas necessidades ao logo do ciclo de desenvolvimento e com isso maximizar o valor da vida útil do produto.

Nessa perspectiva os métodos ágeis de desenvolvimento de *software* respondem ao ambiente altamente volátil e imprevisível e supostamente aumentam dentre outras coisas a satisfação da equipe por intermédio da melhora na comunicação entre os membros e entre a equipe e o cliente, promovendo um processo de melhoria continua a partir do *feedback* do cliente (SAAVEDRA, SCHRIEVEROOFF e LINDEMANN, 2014) [54].

Gradativamente, a TI tem deixado de ser interpretada como área de suporte para servir como base às estratégias de negócio das instituições públicas, consequentemente exigindo um maior monitoramento e controle da área de TI, com a aplicação de métodos consagrados para melhorar seus projetos e investimentos no intuito de assegurar as demandas de serviços (BRASIL, 2012a) [16].

O uso da TI alinhada à estratégia do negócio sem métodos definidos eleva os riscos de se alcançar os objetivos esperados e a necessidade de investimento. A ausência de métodos e procedimentos dificulta ainda mais o gerenciamento dos serviços de TI, tornando uma atividade complexa para as instituições públicas.

Paralelamente, existe a preocupação com a aplicação de melhores práticas no processo de prestação de serviços, diante da infinidade de padrões e *frameworks* disponíveis, que proporcionam o apoio aos objetivos estratégicos das organizações (ALFARAJ e QIN, 2011) [7].

As especificações técnicas e operacionais oferecidas por esses padrões e *frameworks*

dificultam às organizações realizarem uma escolha entre eles, devido à falta de diretrizes práticas orientadas e ferramentas que apoiem a decisão (ALFARAJ e QIN, 2011) [7].

Nesse contexto, em 2012 o Tribunal de Contas da União (TCU), por intermédio do acórdão n. 2.585/2012 publicou o último relatório de levantamento de avaliação da Governança de TI na Administração Pública Federal (APF), realizado em 350 instituições, no qual foi concatenado com os dois últimos relatórios realizados nos anos de 2007 e 2010. No cenário apresentado pelo TCU, os dados demonstram que houve uma evolução da estrutura da governança de TI das APF, com a compreensão do uso da TI alinhada ao negócio. Entretanto, apenas 54% das instituições que participaram do levantamento de 2012 declararam possuir políticas coorporativas de TI com o estabelecimento de objetivos, indicadores e metas de TI, bem como mecanismos para avaliação de desempenho. Contudo, o TCU ressalta que existe uma grande preocupação, em face dos riscos decorrentes da inexistência de 46% das instituições não possuírem uma política definida (BRASIL, 2012a) [16].

Uma das dificuldades que pode ocorrer na aplicação de ferramentas está na identificação adequada de padrões que atendam à organização, a fim de fornecer a garantia de um pleno atendimento de seus requisitos organizacionais (BERNROIDER e IVANOV, 2011) [11]. A eficiência das atividades organizacionais se manifesta pelo domínio dos métodos, controle e gestão, o que requer o conhecimento dos seus processos operacionais, tanto internos quanto externos, que conduz a organização na prática do controle e no ganho de padrões de maturidade (TONINI, CARVALHO e SPINOLA, 2008) [59].

Associado a esse entendimento, o TCU por intermédio do Acórdão n. 1.233/2012, diante do resultado de sua auditoria, orienta as diversas administrações públicas analisadas a estabelecerem obrigatoriedade interna e as entidades sob sua jurisdição que executem um processo formal de implantação em sua estrutura de controle interno mediante definição de atividades de controle em todos os níveis da organização para mitigar os riscos de suas atividades, segundo o acórdão, pelo menos nos seguintes processos:

1. planejamento estratégico institucional;
2. planejamento estratégico de TI;
3. funcionamento dos comitês de TI;
4. processo orçamentário de TI;
5. processo de *software*;
6. gerenciamento de projetos;
7. gerenciamento de serviços de TI;

8. segurança da informação;
9. gestão de pessoal de TI;
10. contratação e gestão de soluções de TI; e
11. monitoração do desempenho da TI organizacional.

Destaca-se que n. 43, inciso I da Lei n. 8443/1992, bem como a orientação do art 250, inciso II do Relatório do Tribunal de Contas (RTCU), à Comissão Interministerial de Governança Coorporativa e de Administração de Participações Societárias da União (CGPAR), e o previsto no Decreto n. 6021/2007, art 3, I, b, ressaltam que os entes sob sua jurisdição devem vincular seus processos de contratação de serviços de desenvolvimento e ou manutenção de *software* a um processo de *software*, pois, sem essa vinculação, o objeto do contrato não estará precisamente definido, em desconformidade com o disposto na Lei n. 8.666/1993. O art 6, inciso IX, de tal Lei prevê (BRASIL, 1993) [14]:

> "IX- Projeto Básico - conjunto de elementos necessários e suficientes, com nível de precisão adequado, para caracterizar a obra ou serviço, ou complexo de obras ou serviços objeto da licitação, elaborado com base nas indicações dos estudos técnicos preliminares, que assegurem a viabilidade técnica e o adequado tratamento do impacto ambiental do empreendimento, e que possibilite a avaliação do custo da obra e a definição dos métodos e do prazo de execução, devendo conter os seguintes elementos:
>
> 1. desenvolvimento da solução escolhida de forma a fornecer visão global da obra e identificar todos os seus elementos constitutivos com clareza;
> 2. soluções técnicas globais e localizadas, suficientemente detalhadas, de forma a minimizar a necessidade de reformulação ou de variantes durante as fases de elaboração do projeto executivo e de realização das obras e montagem;
> 3. identificação dos tipos de serviços a executar e de materiais e equipamentos a incorporar à obra, bem como suas especificações que assegurem os melhores resultados para o empreendimento, sem frustrar o caráter competitivo para a sua execução;
> 4. informações que possibilitem o estudo e a dedução de métodos construtivos, instalações provisórias e condições organizacionais para a obra, sem frustrar o caráter competitivo para a sua execução;
> 5. subsídios para montagem do plano de licitação e gestão da obra, compreendendo a sua programação, a estratégia de suprimentos, as normas de fiscalização e outros dados necessários em cada caso;
> 6. orçamento detalhado do custo global da obra, fundamentado em quantitativos de serviços e fornecimentos propriamente avaliados;"

Os órgãos de controle devem orientar e ou determinar a forma de gestão para mitigar os riscos ligados aos serviços de desenvolvimento de *software*, destacando a obrigatoriedade

do uso de processos fundamentados em metodologias reconhecidas. Apresentado a obrigatoriedade da definição do processo de desenvolvimento de *software* por parte da APF, o Exército Brasileiro (EB) tem estabelecido por intermédio da Portaria n. EB80-MT-78.001, o uso do Manual Técnico para a Metodologia de Desenvolvimento de *Software* do Exército (MDS-EB), fundamentado nos princípios apregoados pelo RUP, o qual é conhecido como método de desenvolvimento tradicional.

Neste contexto, este estudo se propõe a apresentar uma proposta de processo de desenvolvimento de *software* para o EB, tomando como base as melhores práticas de desenvolvimento ágil adotadas pela APF, possibilitando a tomada de decisão estratégica na seleção adequada do método ao projeto a ser desenvolvido, bem como a proposta de mitigação dos riscos associados ao processo proposto, segundo o apresentado pelo TCU por intermédio do Acórdão n. 2314-33/2013

1.2 Problema

Os serviços de TI são feitos com base em análise apropriada e contínua, com tomada de decisão clara e transparente, com equilíbrio apropriado entre benefícios, oportunidades, custos e riscos, de curto e longo prazo (ABNT, 2009) [2].

Para Cruz, Andrade e Figueiredo (2011) [29], as maiores complexidades que envolvem os serviços de TI dizem respeito à identificação dos requisitos necessários à garantia da qualidade dos resultados esperados, aos critérios de aceitação, à gestão de mudanças, às transferências de conhecimentos e à legislação pertinente. Envolve também questões de relacionamento entre o executor e o cliente, o que implica em competências administrativas e jurídicas, complexidades essas que apresentam riscos para as partes envolvidas e, como consequência, é comum à ocorrência de sérios prejuízos à administração.

Entretanto, para Serrador e Pinto (2014) [57], o uso de métodos alternativos para a implementação de projeto, são mais eficientes do que os métodos tradicionais, reconhecendo que nos últimos anos os desafios das técnicas de gerenciamento de projetos com o Ágil foram ganhando popularidade. Destaca-se que as metodologias ágeis em contraste com as abordagens de gerenciamento de projetos tradicionais, enfatiza o projeto contínuo, o escopo flexível, possibilitando que mais tarde seja possível, entender e absorver de forma ágil as modificações solicitadas pelo cliente.

Além disso, o Ágil é descrito como iterativo e incremental, buscando evitar as abordagens padrão que enfatizam o processo inicial do projeto e o congelamento da especificação, um escopo do projeto fixo e a baixa interação do cliente (SERRADOR e PINTO, 2014) [57].

Para as instituições públicas, o processo de desenvolvimento de *software* representa um fator crítico para o negócio. O setor de desenvolvimento se apresenta como uma área complexa, que além de ser requerida uma excelente habilidade no planejamento e no gerenciamento, como garantia da continuidade dos negócios, suportam a estratégia da instituição e atendem aos requisitos do cliente. Nesse sentido, dado a ausência de especificidade da legislação federal com relação à gestão dos riscos dos ativos de TI adotado pela APF, a questão desta pesquisa é: Como propor uma forma inovadora do atendimento das expectativas estratégicas do Exército Brasileiro (EB) com o desenvolvimento de *software* coorporativo fundamentado nas metodologias ágeis, erradicando os riscos associados ao método ágil adotado pela Administração Pública Federal (APF)?

Dessa questão decorre a hipótese de que existe uma inovação no conhecimento originado da prática dos métodos ágeis no desenvolvimento de *software* que deve ser integrado ao processo de desenvolvimento interno na APF, haja vista que esta inovação está associada a várias dimensões: processos, relacionamento com o cliente, agregação de serviços, sistema de crédito ou cobrança, relacionamento com a comunidade etc. (BARROSO, DUARTE, *et al.*, 2007) [9]. Assim, um modelo de desenvolvimento interativo pode tanto contribuir para melhorar a qualidade, eficiência e eficácia na execução das tarefas, quanto contribuir na padronização das ações de atendimento ao cliente e gestão dos riscos associados, tornando institucional a criação e difusão do método interativo e incremental (ágil).

1.3 Justificativa

Com a contínua expansão da demanda bem como a reestruturação da Tecnologia da Informação no Exército Brasileiro, o Centro de Desenvolvimento de Sistemas (CDS) passou a ser o Órgão responsável pela assessoria técnica no desenvolvimento de *software* que envolve a Organização.

Segundo Porter (1999) [49] a estratégia organizacional torna as escolhas sobre o que não fazer tão importantes quanto às escolhas sobre o que fazer. As decisões a respeito dos clientes alvo, das variedades e necessidades a serem desenvolvida pela organização são fundamentais para o desenvolvimento da estratégia. Porém, se tornam igualmente importantes às decisões de não atender a outros clientes ou a outras necessidades e de não oferecer determinadas características ou serviços. Por fim, uma das funções mais importantes de uma estratégia é orientar as escolhas relacionadas às opções excludentes, às atividades individuais e às decisões cotidianas.

Porter (1999) [49] enfatiza ainda, que as organizações devem ser flexíveis no sentido de reagirem rapidamente às mudanças de mercado, desta forma, é importante que pratiquem

de modo constante o *benchmarking* no intuito de buscarem as melhores práticas.

A luz do que foi apresentado pelo Porter (1999)[49] a proposta de um desenvolvimento inovador de *software* vem no sentido de possibilitar que a escolha do método de desenvolvimento seja baseado na estratégia organizacional, a qual apoia o cumprimento da missão do EB, alinhando as atividades de assessoria técnica de desenvolvimento de *software* coorporativo e orgânico por parte do CDS à missão da organização.

Por outro lado, o crescente aumento na demanda em TI, particularmente no que tange ao desenvolvimento de *software*, não diferente à realidade do Exército, obrigou o CDS a repensar seus macro-processos de desenvolvimento uma vez que as necessidades extrapolariam rapidamente sua capacidade em prover meios de TI apenas com os Recursos Humanos disponíveis em seus quadros. Desta forma a busca por métodos que respondam com maior agilidade, de forma interativa e incremental, oferecendo uma maior satisfação tanto do ponto de vista do cliente quanto da equipe se tornou uma alternativa estratégica a ser adotada pelo EB.

Paralelamente a isso, a preocupação com a gestão de risco, custo e tempo transformaram-se em uma conduta necessária na elaboração das especificações técnicas da execução dos serviços de forma qualitativa, eficiente e ágil, especificamente no que tange ao Desenvolvimento e Manutenção dos *Software* Coorporativos da Organização.

Boehm e Turner *apud* Buresh (2008) [25] acreditam que as práticas de engenharia de *software* são por vezes enigmáticas e não só se prestam facilmente para a imposição de práticas de outras indústrias. Não é que o desenvolvimento de *software* e engenharia de *software* são processos incontroláveis; mas sim, os métodos conduzidos corretamente é que permitem que um gerente de projeto se concentre em responder e conseguir um valor rapidamente. O valor do ágil no método de desenvolvimento de *software* é conduzido na compreensão do que as variáveis podem ser rastreadas, e que as mesmas podem ser administradas com segurança, para que a informação adequada e oportuna esteja disponível para os clientes e a gerência do projeto em todo o processo de desenvolvimento.

Complementar as considerações expostas, a pesquisa realizada por Melnik e Maurer (2006) [45] sobre a dimensão da engenharia de *software* enfatiza que uma das questões que as equipes de *software* enfrentam é a rotatividade voluntária que tem um grave impacto na gestão. A pesquisa revela uma forte iniciativa das pessoas para o sucesso de projetos de desenvolvimento de *software*. Foi analisado se o processo de desenvolvimento usado tem um impacto sobre a satisfação no trabalho. Como resultado, houve um grande indicativo de satisfação no desempenho e produtividade na avaliação da experiência das pessoas em participar nos grupos de trabalho que utilizavam o método ágil em comparação com as experiências anteriores de trabalhar em uma equipe não-ágil (tradicional). Dentre as 384 pessoas entrevistadas, nas diversas funções desempenhadas como gerente, funcionário

e consultor, é possível observar que somando os indicadores dos que se sentiram muito melhor e melhor totalizam 78% de satisfação no uso do método ágil no desenvolvimento de *software*, demonstrado na Tabela 1.1.

Tabela 1.1: Comparação da satisfação das metodologias ágil e não-ágil por entrevistado que pratica ágil, agrupados por função, Fonte: (MELNIK; MAURER, 2006) [45]

Função	Muito melhor	Melhor	Sem mudanças	Pior	Muito pior	Total de entrevistados
Gerente	49%	35%	10%	5%		79
Funcionário	39%	37%	16%	5%	2%	292
Consultor	62%	23%	8%		8%	13
Total	42%	36%	15%	5%	2%	384

O valor da entrega desses tipos de serviços deixa de estar pautada apenas no atendimento do escopo inicial e passa a ser subjetivo baseado na percepção do cliente dependente da fronteira de interação entre a equipe desenvolvedora e o cliente BROWNING e HONOUR apud SAAVEDRA *et al* (2014) [54].

Em última análise, o trabalho desenvolvido por Carvalho e Mello (2012) [26], pontua os seguintes benefícios na utilização do método ágil: aumento da satisfação de clientes, por meio da diminuição das reclamações; melhoria na comunicação e aumento da colaboração entre envolvidos nos projetos; aumento da motivação da equipe de desenvolvimento de produtos; melhoria da qualidade do produto produzido; aumento de produtividade da equipe de desenvolvimento; diminuição no tempo gasto para terminar projetos de desenvolvimento de novos produtos; e diminuição do risco em projetos de desenvolvimento de novos produtos.

Dessa forma, se coloca a seguinte questão: Como seria o desenvolvimento ágil de *software* no EB a fim de acompanhar as tendências de mercado citadas pelos autores Carvalho e Mello (2012), assim como Melnik e Maurer (2006) e Saavedra (2014), que podem conduzir a atualização da Metodologia utilizada pelo EB que possibilite obter resultados de redução no tempo de desenvolvimento, maior interação com o cliente, maior satisfação do cliente de forma agregadora de valor ao EB.

Face ao exposto, a próxima seção apresenta o objetivo da pesquisa.

1.4 Objetivo

1.4.1 Geral

O objetivo desta pesquisa é propor um processo de desenvolvimento de *software* para o Exército Brasileiro (EB) fundamentado na metodologia ágil à luz das práticas adotadas pela Administração Pública Federal (APF), e nos processos de mitigação dos riscos apontados pelo Tribunal de Contas da União (TCU) por intermédio do Acórdão n. 2314-33/2013.

1.4.2 Especifico

1. Analisar o processo de desenvolvimento de *software* atual do EB;
2. Analisar casos de desenvolvimento de *software* nas APF's que utilizam metodologias ágeis; e
3. Propor um método de mitigação dos riscos indicados pelo TCU por meio do Acórdão n. 2314-33/2013 para o EB que envolva o uso da metodologia ágil pela APF, à luz das práticas (Normas e cases) estudadas.

Com base nos objetivos específicos, pretende-se obter o fundamento suficiente para propor as melhores práticas das metodologias ágeis estudadas em um processo de desenvolvimento de *software* do EB. Para facilitar o entendimento do leitor, apresenta-se uma breve descrição da estrutura no presente trabalho.

No Capítulo 2 será apresentada uma revisão da literatura sobre o desenvolvimento de *software* e a gestão de risco, seus conceitos e principais características e ferramentas de aplicação. O Capítulo 3 ilustra a metodologia aplicada na presente pesquisa, e o Capítulo 4 retrata o Estudo de Caso realizado com a análise do processo atual de desenvolvimento de *software* do EB e os *benchmarks* realizados nas APFs. Já o Capítulo 5 mostra a proposta do processo de desenvolvimento de *software* ágil para o EB, seguido do Plano de Gestão de Risco. Por fim, o Capítulo 6 apresenta a conclusão do trabalho e as sugestões para trabalhos futuros.

Capítulo 2

Revisão da Literatura

Este capítulo descreve os conceitos relacionados à gestão e ferramentas de riscos de serviços de TI, os principais processos e métodos de desenvolvimento de *software* existentes.

2.1 Gestão de Risco aplicado ao processo de serviços de Tecnologia da Informação

De acordo com Rezende e Abreu (2001) [52], o Planejamento Estratégico da Tecnologia da Informação (PETI) fornece uma visão ampla de conceitos, modelos e ferramentas necessárias para suportar a tomada de decisão e facilitar a estratégia de negócio. Alinhado com essa visão Albertin (2005) [6] considera a TI como:

> "... um papel fundamental na transformação organizacional, não se limitando a mudanças nos processos e na produtividade dos indivíduos. O desafio permanece em determinar exatamente qual é a sua participação, sua contribuição final e o limite para a transformação organizacional (TURNER *apud* (ALBERTIN, 2005)) [6]".

Conforme o *Information Systems Audit and Control Association* (ISACA) (2013) [37], a governança de TI completa e institucionaliza boas práticas garantindo que a área de TI da organização, fundamente os objetivos do negócio, habilitando a organização a obter vantagens de suas informações, maximizando os benefícios, capitalizando as oportunidades e ganhando em poderes competitivos.

A Organização para a Cooperação e Desempenho Econômico (OCDE) *apud* Weill e Ross (2006) [61] definiu dos lados da governança: o comportamento e o normativo. O comportamental da governança de TI define os relacionamentos formais e informais, conferindo direitos decisórios a indivíduos ou grupos. Já o normativo define mecanismos, formalizando os relacionamentos e estabelecendo regras para assegurar que os objetivos sejam atingidos.

A governança de TI considera esta área não apenas como um suporte à organização, mas uma parte importante para que mantenha a gestão administrativa e estratégica da organização, sendo o principal objetivo manter os processos e práticas relacionados à infraestrutura de sistemas, redes e dispositivos utilizados na empresa.

Para Fernandes e Abreu (2008) [30], a governança é essencial para garantir melhorias eficientes e eficazes nos processos da organização, no qual se fornece uma estrutura que se interliga entre si: processos de TI, os recursos de TI e as informações às estratégias e objetivos da organização. Além disso, assegura que as informações da organização e a tecnologia aplicada suportem aos objetivos do negócio.

Segundo Weill e Ross (2006) [61], existem cinco grupos de decisões de TI, que devem ser considerados de forma conjunta para que haja eficácia na Governança:

1. Princípios de TI: esclarecem o papel de negócio da TI constituindo regras gerais a serem seguidas pela organização;
2. Arquitetura de TI: definem os requisitos de integração e padronização;
3. Infraestrutura de TI: determinam os serviços compartilhados e de suporte;
4. Necessidades de aplicações do negócio: especificam a necessidade comercial das aplicações de TI a serem adquiridas ou desenvolvidas internamente; e
5. Investimentos priorizados de TI: Definem quais iniciativas serão financiadas e quanto deverá ser gasto.

Fernandes e Abreu (2008) [30] consideram que a governança de TI alinha a tecnologia da informação aos requisitos do negócio fundamentada na continuidade e o atendimento às estratégias do negócio, dividida em seis objetivos:

1. Permitir a TI um posicionamento mais claro e consistente em relação às outras áreas de negócio da organização. Isto significa que a TI deve entender os objetivos, as diretrizes e as estratégias de negócio para que estes objetivos possam ser traduzidos em iniciativas tecnológicas;
2. Alinhar e priorizar as iniciativas de TI com a estratégia do negócio;
3. Alinhar a arquitetura e a infraestrutura de TI às necessidades do negócio, em termos de planejamento;
4. Prover a TI com processos operacionais e de gestão, necessários para atender os serviços em conformidade com os padrões exigidos pelo negócio;
5. Garantir a TI uma estrutura de processos que possibilitam a gestão do seu risco para continuidade operacional da organização; e

6. Prover regras claras para as responsabilidades sobre as ações e decisões relacionadas a TI.

Alinhado aos objetivos da Governança está à terceirização de TI como processo de relacionamento entre o cliente e o fornecedor do serviço na identificação consistente das informações, relacionamentos, controle e trocas como mecanismo de assegurar a continuidade dos serviços em um nível aceitável, agregando valor de sustentabilidade para ambos (FERNANDES e ABREU, 2008) [30].

Associado a estes objetivos está a necessidade da mensuração do desempenho organizacional e processual com monitoramento das saídas, bem como de sua capacidade. Segundo Fernandes e Abreu (2008) [30], o a Governança de TI é composta por quarto grandes etapas, como é representado na Figura 2.1.

Figura 2.1: Ciclo da Governança de TI, Fonte: (BRASIL, 2013b) [18], adaptado pelo autor

O ciclo da governança de TI compreende as etapas de alinhamento estratégico, decisão, Compromisso, Priorização e Alocação de recursos, Estrutura, Processos, Operações e Gestão e Medição de desempenho que são detalhadas a seguir:

1. Alinhamento estratégico refere-se ao planejamento estratégico da TI que leva em consideração as estratégias da organização para seus produtos e segmentos de atuação;

2. Etapa de decisão, compromisso, priorização e alocação de recursos referentes às responsabilidades pelas decisões relativas a TI, em termos de: arquitetura, serviços de infraestrutura, investimentos e necessidades de aplicações;

3. Etapa de estrutura, processo, operação e gestão referem-se à estrutura organizacional e funcional da TI, aos processos de gestão e operação dos produtos e serviços, alinhados com as necessidades estratégicas e operacionais da organização; e

4. Etapa de medição de desempenho refere-se à determinação, coleta e geração de indicadores de resultados dos processos, produtos e serviços de TI, conforme demonstrado na Figura 2.2.

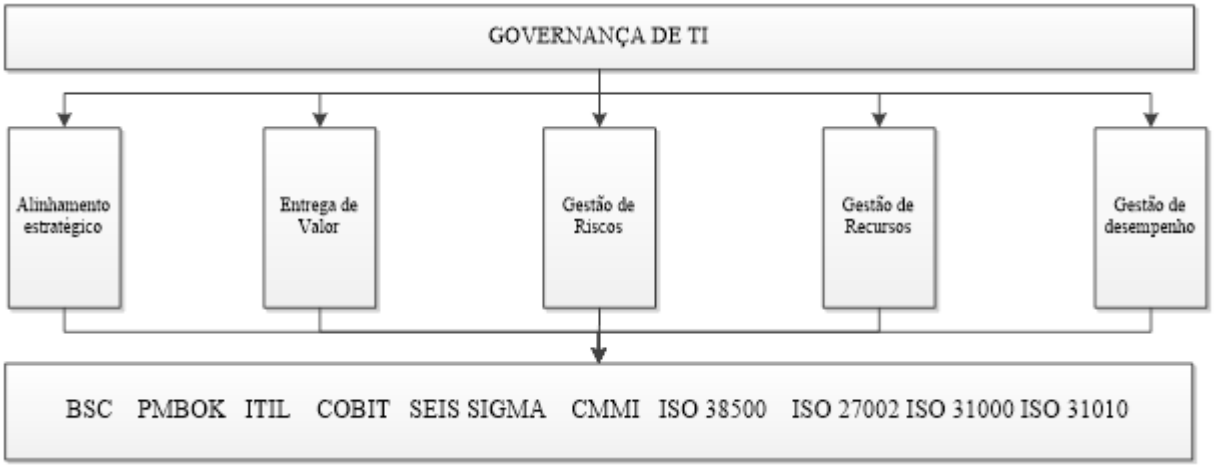

Figura 2.2: Governança de TI, Fonte: (BRASIL, 2013b)[18], adaptado pelo autor

Analisando a Figura 2.2, uma vez constituída a base consultiva é possível definir as atribuições referentes à mensuração dos recursos e métodos necessários a TI, com objetivo de conseguir suprir as demandas internas, alinhadas com os objetivos da organização.

A capacidade de avaliar o desempenho da organização e melhorar o conhecimento dos processos possibilita desenhar eventuais métodos necessários para potencializar a gestão dos riscos e consequentemente minimizar suas consequências.

Na seção a seguir são apresentados conceitos concernentes à Gestão de Risco alinhada a Governança de TI, fundamentada na Associação Brasileira de Normas Técnicas (ABNT NBR) ISO 31000:2009 e 31010:2012 de Gestão de Risco e Técnicas para o processo de avaliação de riscos, respectivamente.

2.1.1 ABNT NBR ISO 31000:2009 - Gestão de Risco

A Norma ISO 31000:2009 foi elaborada pela Comissão de Estudo Especial de Gestão de Risco da ABNT, com a finalidade de estabelecer um número de princípios que precisam ser atendidos para tornar a gestão de riscos eficaz, com perspectiva de integrar o processo de gerenciamento de riscos na governança, estratégia e planejamento, gestão, processos de reportar dados e resultados, políticas, valores e cultura em toda a organização, com uma melhoria continua do *framework* (ABNT, 2009) [2].

Segundo a ISO 31000:2009, uma de suas características chave é a inclusão do estabelecimento do contexto como uma atividade no inicio deste processo genérico de gestão de risco, com estabelecimento de captura dos objetivos da organização, do ambiente em que ela persegue esses objetivos, suas partes interessadas e a diversidade de critérios de risco, auxiliando o processo de identificação e avaliação natural e a complexidade de seus riscos.

Para a ISO 31000:2009, a implantação e manutenção da Norma na gestão dos riscos possibilita a organização:

1. Aumentar a probabilidade de atingir os objetivos;
2. Encorajar uma gestão proativa;
3. Estar atento para a necessidade de identificar e tratar os riscos através de toda a organização;
4. Melhorar sua governança;
5. Melhorar os controles;
6. Minimizar perdas; e
7. Aumentar a resiliência da organização.

A citada ISO 31000:2009 é destinada a atender às necessidades de uma ampla gama de partes interessadas, incluindo:

1. Os responsáveis pelo desenvolvimento da política de gestão de risco no âmbito de suas organizações;
2. Os que precisam avaliar a eficácia de uma organização em gerenciar riscos; e
3. Desenvolvedores de normas, guias, procedimentos e códigos de práticas que, no todo ou em parte, estabelecem como risco deve ser gerenciado dentro do contexto especifico desses documentos.

Fundamentado na ISO 31000:2009, na presente seção quando usada às expressões "gestão de riscos" e "gerenciamento de riscos" em termos gerais referem-se à arquitetura (princípios, estrutura e processo) para gerenciar riscos eficazmente, e à aplicação dessa arquitetura para riscos específicos respectivamente.

Como estrutura relacional a Figura 2.3 demonstra como ocorrem os relacionamentos entre os princípios da gestão de riscos, estrutura e processo da Norma.

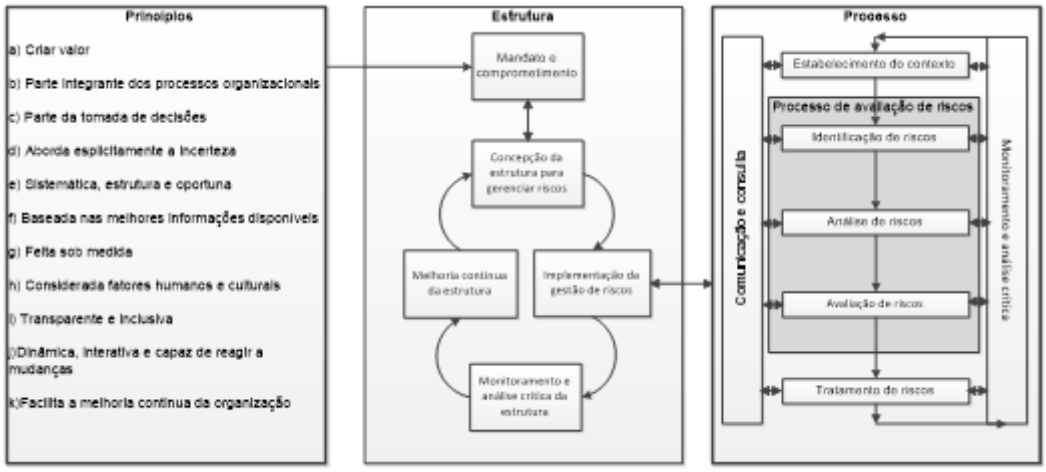

Figura 2.3: Relacionamentos entre os princípios da gestão de riscos, estrutura e processo. Fonte: (ABNT, 2009) [2]

Na composição dos princípios e diretrizes, a ISO 31000:2009 estabelece genericamente gestão de riscos, podendo ser aplicada em uma gama de atividades da organização incluindo estratégias, decisões, operações, processos, funções, projetos, produtos, serviços e ativos organizacionais, bem como associação a normas vigentes. Como definições de risco a ISO 31000:2009 [2] aplica:

1. Efeito da incerteza nos objetivos;
2. Um efeito é um desvio em relação ao esperado, positivo e/ou negativo;
3. O risco é muitas vezes caracterizado pela referência aos eventos potenciais e às consequências, ou uma combinação destes;
4. O risco é muitas vezes expresso em termos de uma combinação de consequências de um evento (incluindo mudanças nas circunstâncias) e a probabilidade de ocorrência associada; e
5. A incerteza é o estado, mesmo que parcial, da deficiência das informações relacionadas a um evento, sua compreensão, seu conhecimento, sua consequência ou sua probabilidade.

A Norma apresenta uma estrutura de formas de auxiliar a gerência dos riscos por intermédio da aplicação do processo de gestão de riscos em diferentes níveis e dentro de contextos específicos, assegurando que a informação sobre os riscos provenientes desse processo seja adequadamente reportada e utilizada como base para a tomada de decisões e a responsabilização em todos os níveis aplicáveis, demonstrado na Figura 2.4.

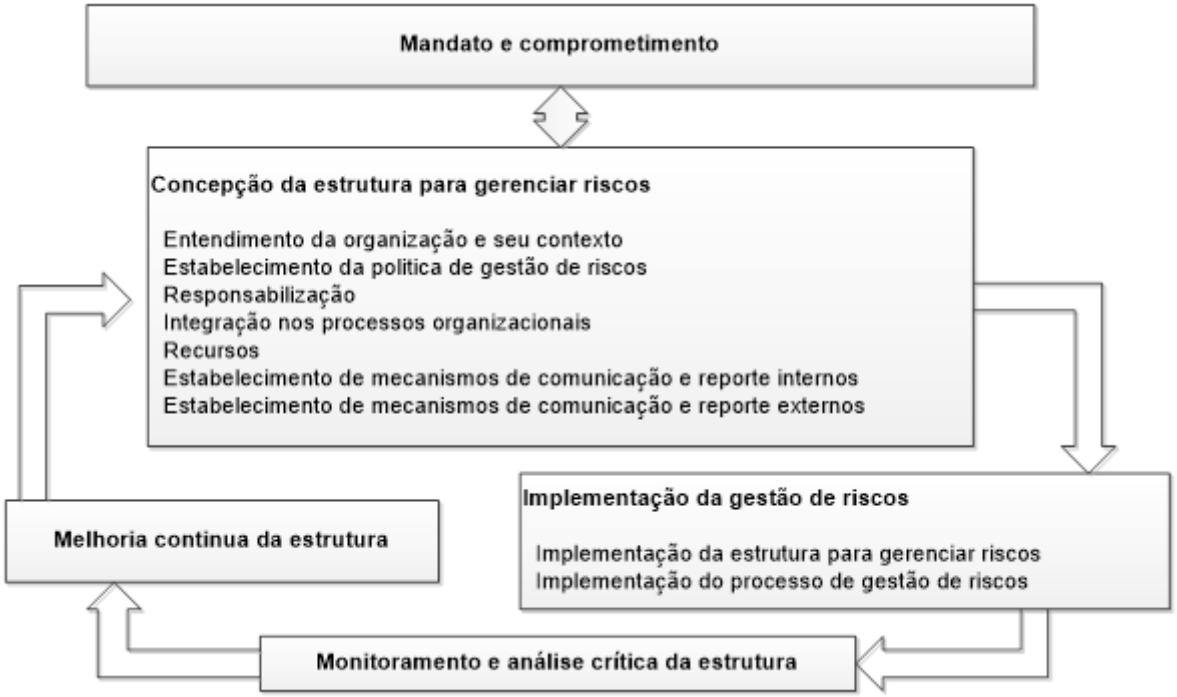

Figura 2.4: Relacionamento entre os componentes da estrutura para gerenciar riscos. Fonte: (ABNT, 2009) [2]

Na abordagem da estrutura da Norma, existem 5 componentes necessários para gerenciar riscos e a forma como eles se interelacionam de maneira interativa, a saber: a) Mandato e comprometimento; b) Concepção da estrutura para gerenciar riscos; c) Implementação da gestão de riscos; d) Monitoramento e análise critica da estrutura; e e) Melhoria continua da estrutura. A seguir serão expostos com maior detalhamento:

1. Mandato e comprometimento: O uso da gestão de risco e a garantia de sua contínua eficácia requerem comprometimento da organização, bem como planejamento rigoroso e estratégico com definição e aprovação de políticas de gestão de riscos, alinhamento da cultura da organização e a política de gestão de riscos, como também com os objetivos e estratégias da organização e as conformidades legais e regulatórias.

2. Concepção da estrutura para gerenciar riscos:

   (a) Entendimento da organização e seu contexto: antes de iniciar a concepção e a implementação da estrutura para gerenciar riscos, é importante avaliar e compreender o contexto externo e interno da organização, uma vez que estes podem influenciar significativamente a concepção da estrutura.

(b) Estabelecimento da política de gestão de riscos: convém que a política de gestão de riscos estabeleça claramente os objetivos e o comprometimento da organização em relação à gestão de riscos.

(c) Responsabilização: convém que a organização assegure que haja responsabilização, autoridade e competência apropriadas para gerenciar riscos, incluindo implementar e manter o processo de gestão de riscos, e assegurar a suficiência, a eficácia e a eficiência de quaisquer controles.

(d) Integração nos processos organizacionais: convém que a gestão de riscos seja incorporada em todas as práticas e processos da organização, de forma que seja pertinente, eficaz e eficiente.

(e) Recursos: convém que se aloque recursos apropriados para a gestão de riscos, com pessoal habilitado e experiente, bem como processos, métodos e ferramentas.

(f) Estabelecimento de mecanismos de comunicação e reporte interno: convém que estabeleça mecanismos de comunicação interna e reporte a fim de apoiar e incentivar a responsabilização e a propriedade dos riscos.

(g) Estabelecimento de mecanismo de comunicação e reporte externo: convém que se desenvolva e implemente um plano sobre como se comunicará com partes interessadas externas.

3. Implementação da gestão de riscos

   (a) Implementação da estrutura para gerenciar riscos: convém que se definida a estratégia e o momento apropriado para implementação da estrutura, que aplique a política e o processo de gestão de riscos aos processos organizacionais, e que atenda aos requisitos legais e regulatórios.

   (b) Implementação do processo de gestão de riscos: convém que a gestão de riscos seja implementada por intermédio de um plano aplicado em todos os níveis e funções pertinentes da organização, como parte de suas práticas e processos.

4. Monitoramento e análise critica da estrutura: convém que seja realizado medição do desempenho da gestão de riscos utilizando indicadores, os quais devem ser analisados criticamente, de forma periódica, para garantir sua adequação.

5. Melhoria continua da estrutura: com base nos resultados do monitoramento e das análises criticas, convém que decisões sejam tomadas sobre a política, o plano e a estrutura da gestão de riscos podem ser melhorados.

O ciclo da gestão de riscos é concluído com o processo incorporado na cultura e nas práticas e adaptado aos processos de negócio da organização (ABNT, 2009) [2], conforme demonstrado na Figura 2.5.

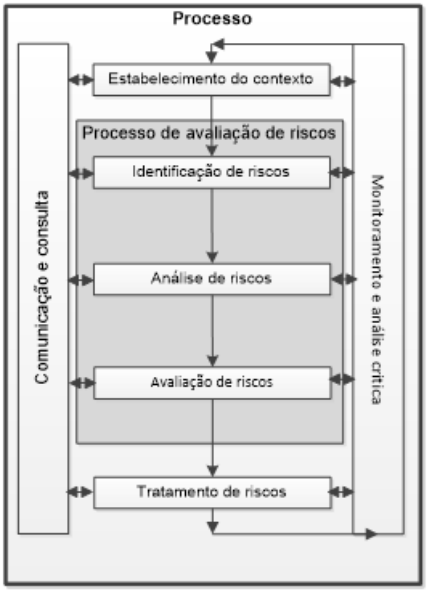

Figura 2.5: Processo de gestão de riscos, Fonte: (ABNT, 2009) [2]

Conforme demonstrado na figura 2.5 o processo de gestão de riscos é desenvolvido em 5 macro atividades, a saber:

1. Monitoramento e consulta: a comunicação e a consulta às partes interessadas internas e externas acontecem durante todas as fases do processo de gestão de riscos.
2. Estabelecimento do contexto:
   (a) Generalidades: ao se estabelecer o contexto é definido os parâmetros externos e internos a serem levados em consideração ao gerenciar riscos, e estabelecido o escopo e os critérios de risco para o restante do processo.
   (b) Estabelecimento do contexto externo: entender o contexto externo é importante para assegurar que os objetivos e as preocupações das partes interessadas externas sejam consideradas no desenvolvimento dos critérios de risco.
   (c) Estabelecimento do contexto interno: é o ambiente interno no qual se busca atingir seus objetivos.
   (d) Estabelecimento do contexto do processo de gestão de riscos: são estabelecidos os objetivos, as estratégias, o escopo e os parâmetros das atividades em que o processo de gestão de riscos está sendo aplicado.

(e) Definição dos critérios de risco: é a definição dos critérios a serem utilizados para avaliar a significância do risco, refletindo os valores, objetivos e recursos.

3. Processo de avaliação de riscos:

   (a) Generalidade: é o processo global de identificação de riscos, análise de riscos e avaliação de riscos.

   (b) Identificação de riscos: é identificar as fontes de risco, áreas de impacto, eventos (incluindo mudanças nas circunstâncias) e suas causas e consequências potenciais.

   (c) Análise de riscos: envolve desenvolvimento de compreensão dos riscos com decisões sobre as necessidades dos riscos serem tratados, e sobre as estratégias e métodos mais adequados de tratamento de riscos.

   (d) Avaliação de riscos: é o auxilio na tomada de decisão com base nos resultados de um evento ou conjunto de eventos, ou por extrapolação a partir de estudos experimentais ou a partir dos dados disponíveis.

4. Tratamento de riscos:

   (a) Generalidade: envolve a seleção de uma ou mais opções para modificar os riscos e a implementação dessas opções, fornecendo possíveis novos controles ou modificar os existentes.

   (b) Seleção das opções de tratamento de riscos: seleciona a opção mais adequada de tratamento de riscos envolve equilibrar, de um lado, os custos e os esforços de implementação e, de outro, os benefícios decorrentes, relativos a requisitos legais, regulatórios ou quaisquer outros, tais como o da responsabilidade social e o da proteção do ambiente natural.

   (c) Preparando e implementando planos de tratamento de riscos: a finalidade é documentar como as opções de tratamento escolhidas serão implementadas.

5. Monitoramento e análise critica: é o planejamento como parte do processo de gestão de riscos envolvendo a checagem periódica ou vigilância regulares em resposta a um fato especifico.

Por fim, de acordo com a ISO 31000:2009 [2] a análise de risco é o processo de compreender a natureza do risco, como forma de base para a avaliação de riscos e para as decisões sobre o tratamento de risco fundamentado em estimativa.

O tópico a seguir apresenta as ferramentas e técnicas de gestão de risco fundamentado na Norma ABNT NBR ISO 31010:2012.

2.1.2 ABNT NBR ISO 31010:2012 - Ferramentas e Técnicas de Gestão de Risco

A Norma ISO 31010:2012 tem por objetivo apoiar a Norma ISO 31000:2009, como fornecimento de diretrizes detalhadas sobre critérios de seleção e aplicação de técnicas qualitativas e quantitativas para avaliação de riscos, elaborada pela Comissão de Estudos Especial de Gestão de Riscos da ABNT fundamentada em conteúdo técnico e estrutura idêntica do *Tecnical Committe Dependability* (IEC/TC 56) em conjunto com ISO TMB *"Risk management"*, conforme ISO/IEC Guide 21-1:2005 (ABNT, 2012) [3].

Diferente da ISO 31000:2009 que fornece um *framework* com passos a serem seguidos para a gestão de riscos, a Norma ISO 31010:2012 dispõe de ferramentas que poderão ser fundamentadas nas melhores práticas. Estas incluem técnicas especificas de análise de risco, detalhando métodos de aplicabilidade, a exemplo do uso da técnica *Brainstorming* na fase de identificação do risco, bem como da técnica Delphi com aplicação de questionários.

O uso adequado de ferramentas na gestão de risco inclui a aplicação de métodos lógicos e sistemáticos (ABNT, 2012) [3], tais como:

1. comunicação e consulta ao longo de todo o processo;
2. estabelecimento do contexto para identificar, analisar, avaliar e tratar o risco associado a qualquer atividade, processo, função ou produto;
3. monitoramento e análise crítica de riscos; e
4. reporte e registro dos resultados de forma apropriada.

Segundo a ISO 31010:2012 as técnicas de classificação se aplicam para cada etapa do processo de identificação de riscos, análise de riscos (consequência, estimativa, semi-quantitativa ou quantitativa da probabilidade e avaliação da eficácia de quaisquer controles existentes) e avaliação de riscos. Destacam-se os mais de 35 tipos de técnicas e ferramentas, detalhadas na Tabela de aplicabilidade das ferramentas para o processo de avaliação de riscos (Tabela 2.1), e são descritas como sendo Fortemente Aplicável (FA), Aplicável (A) ou Não Aplicável (NA)

Tabela 2.1: Aplicabilidade das ferramentas utilizadas para o processo de avaliação de riscos, Fonte: (ABNT, 2012) [3], adaptada pelo autor

Ferramentas e técnicas	Processo de avaliação de riscos				
	Identificação de riscos	Análise de riscos consequência	probabilidade	nível de risco	Avaliação de riscos
Brainstorming	FA	NA	NA	NA	NA
Entrevista estruturada ou semi-estruturada	FA	NA	NA	NA	NA
Delphi	FA	NA	NA	NA	NA
Listas de verificação	FA	NA	NA	NA	NA
Análise preliminar de perigos (APP)	FA	NA	NA	NA	NA
Estudo de perigos e operabilidade (HAZOP)	FA	DA	A	A	A
Análise de perigos e pontos críticos de controle (APPCC)	FA	FA	NA	NA	FA
Avaliação de risco ambiental	FA	FA	FA	FA	FA
Técnica estruturada "E se" (SWIFT)	FA	FA	FA	FA	FA
Análise de cenários	FA	FA	A	A	A
Análise de impactos no negócio	A	FA	A	A	A
Análise de causa-raiz	NA	FA	FA	FA	FA
Análise de modos de falha e efeito	FA	FA	FA	FA	FA
Análise de árvore de falhas	A	NA	FA	A	A
Análise de causa e consequência	A	FA	FA	A	A
Análise de causa e efeito	FA	FA	NA	NA	NA
Análise de camadas de proteção (LOPA)	A	FA	A	A	NA
Árvore de decisões	NA	FA	FA	A	A
Análise da confiabilidade humana	FA	FA	FA	FA	A
Análise *Bow tié*	NA	A	FA	FA	A
Manutenção centrada em confiabilidade	FA	FA	FA	FA	FA
Sneak analusis (SA) e sneak circuit analysis (SCA)	A	NA	NA	NA	NA
Análise de Markov	A	FA	NA	NA	NA
Simulação de Monte Carlo	NA	NA	NA	NA	FA
Estatística Bayesisna e Redes de Bayes	NA	FA	NA	NA	FA
Curvas FN	A	FA	FA	A	FA
Índices de risco	A	FA	FA	A	FA
Matriz de probabilidade/consequência	FA	FA	FA	FA	A
Análise de custo/benefício	A	FA	FA	FA	A
Análise de decisão por multicritérios (MCDA)	A	FA	A	FA	A

É possível ter um entendimento que na aplicação de gestão de risco é necessário obter conhecimento dos conceitos, das ferramentas e das técnicas aplicáveis para melhor contribuir no gerenciamento dos riscos conforme apresentado pelas Normas ABNT NBR ISO 31000:2009 e 31010:2012.

Dando continuidade aos estudos de teorias e literaturas utilizadas como fundamento na pesquisa, o tópico a seguir descreve os principais conceitos de processo de engenharia de *software* e os métodos e processos existentes.

2.2 Metodologias de Desenvolvimento de *Software*

Para fins deste estudo se justifica o interesse em aprofundar o conhecimento sobre as melhores práticas no uso de métodos ágeis e de ser uma fonte possível de incentivo para a aplicação prática no processo de desenvolvimento de *software* no âmbito da APF.

Com isso, para o entendimento da aplicação deste estudo, torna-se necessário conhecer um pouco sobre o histórico da engenharia de *software*. Este tópico define alguns conceitos que servem para o esclarecimento e embasamento, partindo de conceitos mais gerais até os mais específicos, tais como: Engenharia de *Software*, Modelos de Processos de Desenvolvimento, Metodologias Tradicionais e Ágeis.

2.2.1 Engenharia de *Software*

No mundo atual, o uso de *software* se tornou algo muito presente e essencial tanto nos meios da TI quanto nas diversas áreas de atuação do ser humano e algo que há séculos ninguém imaginava que teria essa proporção (PRESSMAN, 2011) [50].

Como afirma Sommerville (2011) [58], quase todos os países são dependentes de sistemas complexos baseados em computadores, onde serviços e infraestruturas contam com os mesmos, sendo que um computador e um *software* de controle são incluídos na maioria dos produtos eletrônicos.

Em complemento, Pressman (2011) [50] também aponta que no palco mundial o *software* é, hoje, a tecnologia mais importante, além de servir como exemplo da lei das consequências não pretendidas, onde ninguém teria previsto que o *software* se tornaria uma indispensável tecnologia para a ciência, engenharia e negócio, permitindo ainda a revolução tecnológica industrial.

Além dos benefícios originários do *software*, Sommerville (2011) [58] aponta que na concepção de sistemas, a experiência inicial apenas no desenvolvimento informal de *software* não se mostrou suficiente e que, ás vezes, os projetos sofriam anos de atraso, e em consequência, ele não era confiável, o desempenho era insatisfatório, o custo superava previsões, era difícil de manter, constituindo a crise no desenvolvimento de *software*, que conceituou a engenharia de *software* em:

> "é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado. [...]".

Advindo da crise revolucionária, o software passou a ser um notável processo evolutivo de desempenho do hardware na melhoria arquitetural de computadores entre outras melhorias, resultando que a complexidade e sofisticação tem o potencial de apresentar grandes resultados quando o sistema é bem-sucedido (PRESSMAN, 2011) [50].

Pressman (2011) [50] aponta sete categorias amplas de *software* ao qual são constantes para os engenheiros de *software*, que são:

1. Aplicação da Web: abrange uma ampla diversidade de aplicações que de forma simplificada, pode ser um conjunto de arquivos juntos de hipertexto usando poucos gráficos e textos ao qual, conforme o sistema fica mais complexo, evoluem para sistemas sofisticados envolvendo até relações com o banco de dados, entre diversas funções com conteúdos fornecidas ao usuário final;

2. *Software* de aplicação: são programas únicos que procuram resolver uma necessidade especifica do negócio ao qual processam dados técnicos ou comerciais facilitando e auxiliando diversas operações relacionadas ao negócio e também serve para ser usado no controle de negócios em tempo real;

3. *Software* cientifico de Engenharia: era caracterizado antigamente por algoritmos que processam números e vão desde a dinâmica orbital do ônibus espacial a análise automotiva de tensões, de manufatura automatizada à biologia molecular, etc.. Entretanto, as aplicações modernas estão se distanciando deste cenário de algoritmos numéricos convencionais, pois aplicações interativas começaram a obter características de *software* de sistemas e de tempo real;

4. *Software* Embutido: é utilizado para controlar e implementar funções e características muito limitadas para o sistema e para o próprio sistema operacional ou fornecer para eles uma função significativa e capacidade de controle;

5. *Software* para Inteligência Artificial: utiliza-se para resolver problemas complexos, ao qual não são passiveis de análise direta ou computação, algoritmos não-numéricos;

6. *Software* para Linha de Produtos: projetado para ser usado por uma grande quantidade de clientes diferentes e fornece uma capacidade especifica;

7. *Software* de Sistemas: pode-se dizer que é um conjunto de programas escritos com a finalidade de servir outros programas (tendo como exemplo editores e utilitários para gestão de arquivos e compiladores) podendo estes processar complexas, mas determinadas, estruturas de informação, processar dados indeterminados amplamente estabelecido.

Para Sommerville (2011) [58] os produtos de *software* são classificados em: produtos genéricos, produzidos por alguma organização do tipo *stand-alone* e vendidos para qualquer cliente no mercado, como também sob encomenda, desenvolvido para um determinado cliente, por outra empresa de *software*.

Como parte do cenário da aplicação de ferramenta e dentre as várias abordagens que compõem a engenharia de *software*, é necessário ter conhecimento sobre o processo de *software*, os modelos de processos de *software*, as metodologias de desenvolvimento de *software* tradicionais e ágeis, até que se chegue ao proposto por esta pesquisa.

2.2.1.1 Processo de *Software*

O processo de Engenharia de *software* é composto por camadas de sustentação de compromisso organizacional com a qualidade, tendo como alicerce a camada de processo, demonstrado na Figura 2.6 (PRESSMAN, 2011) [50].

Apresentado o conceito de *software* no tópico anterior, mostra-se então o conceito de processo que é o ato de proceder, de ir adiante com sucessão de estados ou mudanças, realizando e executando métodos e técnicas (FERREIRA, 2011) [32]. Este de modo geral, somado ao conceito de *software*, oferece uma base aos conceitos seguintes sobre processos de *software*.

Figura 2.6: Engenharia de *Software* em Camadas, Fonte: (PRESSMAN, 2011)[50]

De acordo com Pressman (2011) [50], o processo de engenharia de *software* permite a construção oportuna e racional de *software* de computador e é o adesivo aos quais as camadas de tecnologia mantêm-se unidas. Os processos de *software* estabelecem o contexto ao qual são aplicados os métodos técnicos, formam a base para o controle gerencial, os marcos são definidos, as mudanças são geridas adequadamente, os produtos de trabalho são produzidos e a qualidade é assegurada.

Dentre as camadas da engenharia, os métodos para se construir o *software* são os que oferecem a técnica de "como fazer" enquanto as ferramentas são responsáveis pelo apoio

semi-automatizado ou automatizado para os métodos e para o processo (PRESSMAN, 2011) [50].

Reis (2002) [51] aponta também que, informalmente, o processo de *software* é um conjunto com todas as atividades necessárias que visa modificar os requisitos de usuário em *software*, sendo este formado por um conjunto de procedimentos de processos em parte ordenadas, relacionada com pessoas, recursos, artefatos, restrições e estruturas organizacionais com o objetivo de produzir e manter os projetos de *software* finais requeridos.

Para Sommerville (2011) [58] o processo de *software* é o agrupamento de atividade executadas por engenheiros de *software*, que interligadas concebem um produto de *software* ao qual existem quatro atividades fundamentais:

1. Especificação de *Software*: é a definição das restrições e a funcionalidade do *software*;
2. Desenvolvimento do *Software* (Projeto e implementação do *software*): o software dever ser construído de forma que atenda às suas especificações;
3. Validação do *Software*: garantia de que o *software* faz o que o cliente requisitou; e
4. Evolução do *Software*: busca atender as necessidades mutáveis do cliente, o *software* deve evoluir.

Essas atividades são organizadas por processo de *software* distintos e descritas em níveis de detalhes diferentes, sendo que as organizações podem usar diferentes processos para construir um mesmo produto de *software*, podendo ser mais adequado a um tipo de processo que outro, dependendo do tipo de método de aplicação, e se aplicado inadequadamente pode ocasionar a redução da utilidade ou qualidade do produto de *software* a ser concebido (SOMMERVILLE, 2011) [58].

Complementando, para Pressman (2011) [50] o processo de *software* é o alicerce para um projeto de *software* ao qual se estabelece pela identificação de uma pequena quantidade de atividades aplicáveis a todos os projetos, independente de sua complexidade ou tamanho, e podem ser citadas como:

1. Comunicação: é uma atividade que envolve a colaboração com o cliente e abrange o levantamento de requisitos e outras atividades relacionadas e alta comunicação;
2. Planejamento: é a atividade em que um plano para o trabalho de engenharia de *software* que se segue é estabelecido descrevendo as tarefas técnicas a serem conduzidos, os recursos que serão precisos, os produtos que serão produzidos, os prováveis riscos e um cronograma do trabalho;

3. Modelagem: é a atividade que para o cliente e o desenvolvedor terem um melhor entendimento dos requisitos e projeto que irá satisfazer a esses requisitos, incluindo a criação de modelos;

4. Construção: é a atividade em que se une a geração do código, seja ele automático ou manual, e os testes que são necessários para que se revelem os erros dos códigos; e

5. Implantação: é a atividade em que se entrega ao cliente o *software*, estando ele completo ou pelo menos praticamente completo, em que o produto entregue é avaliado fornecendo como base de avaliação *feedback*.

Contudo Pressman (2011) [50] ainda complementa as atividades descritas, com atividades complementares de controle, monitoramento e gestão de projeto, as quais podem ser classificadas como:

1. Acompanhamento e Controle de Projeto de *Software*: com base no plano de projeto, a equipe de *software* consegue avaliar o progresso e, para manter o cronograma, tornar a ação necessária;

2. Gestão de Risco: é a avaliação dos riscos que podem afetar a qualidade do produto ou o resultado do projeto;

3. Garantia de qualidade de *software*: para garantir a qualidade de *software*, se conduz e define as atividades necessárias;

4. Revisões técnicas formais: esforço para descobrir e remover erros antes que se alastrem para a próxima atividade ou ação, avaliam-se os produtos de trabalho de engenharia de *software*;

5. Medição: reúne e define medidas de projeto, processo e produto que auxiliam a equipe na entrega de um *software* que agrade as necessidades do usuário, podendo ser utilizadas juntamente com outras atividades;

6. Gestão de configuração de *software*: administra os efeitos das mudanças ao longo de todo o processo de *software*;

7. Gestão de Reusabilidade: indica as condições necessárias para a reutilização dos produtos de trabalho e determina mecanismos para obtenção de componentes reusáveis; e

8. Preparação e Produção do produto de trabalho: criações de produtos de trabalho são reunidas atividades necessárias como documentos, registros, modelos, listas e formulários.

Complementando a pesquisa, o próximo tópico abordará os modelos de processos de *software* e qual a relevância para a engenharia de *software*.

2.2.1.2 Modelos de processo de *software*

Os modelos de processo de *software* são responsáveis pela definição de um conjunto distinto de atividades, tarefas, marcos e produtos de trabalho que são precisos para se realizar a engenharia de *software* com alta qualidade, e que, mesmo não sendo perfeitos, estes fornecem efetivamente um útil roteiro para o trabalho de engenharia (PRESSMAN, 2011) [50].

Segundo Sommerville (2011) [58], um modelo de processo de *software* descreve simplificamente o processo ao qual, a partir de uma perspectiva específica, é apresentada. Sendo de natureza simplificada, este é uma abstração do real processo que é descrito, destacando-se dentre os modelos de processo de *software* o papel das pessoas envolvidas na engenharia de *software* e produtos de *software* e as atividades que são parte do processo de *software*, a exemplo:

1. Modelo de *Workflow*: juntamente com suas dependências, entradas e saídas, mostra a sequencia de atividades que, representam ações humanas;
2. Modelo de fluxo de dados ou de atividades: o processo é representado como um conjunto de atividades onde cada uma delas realiza alguma transformação de dados;
3. Modelo de Papel/Ação: representa as atividades pelas quais as pessoas envolvidas no processo de *software* são responsáveis e apresenta os papéis de cada uma dessas pessoas.

De acordo com Pressman (2011) [50], os modelos de processo de *software*, ao qual ele denomina como modelos prescritivos de *software*, são importantes, pois oferecem estabilidade, organização e controle para uma atividade que pode se tornar bastante caótica se deixada sem controle. A aplicação de qualquer modelo de processo de *software* deve reconhecer que, para se alcançar o sucesso é essencial à adaptação. Pressman (2011) [50] aponta também que os modelos de processo diferem fundamentalmente:

1. No fluxo geral de tarefas e atividades e interdependências entre tarefas e atividades;
2. No grau em que, dentro de cada atividade, as tarefas de trabalho são determinadas;
3. No nível em que os produtos de trabalho são reconhecidos e requeridos;
4. Na forma como são aplicadas as atividades de garantia de qualidade;

5. Na maneira como são aplicadas as atividades de controle e o monitoramento de projetos;
6. No grau geral de rigor e detalhes em que é descrito o projeto;
7. No nível em que estão envolvidos no projeto de *software*; e
8. No nível em que são prescritos os papéis e organização da equipe.

Sommerville (2011) [58] afirma que o processo é representado por um modelo que se apresenta somente informações parciais sobre o processo, não definitivo as quais são abstrações proveitosas que podem ser usadas para explanar abordagens diversas do desenvolvimento de *software* e que, naturalmente, para projetos de grande porte não existe um processo de software somente que possa ser usado, mas diferentes processos são usados para desenvolver diferentes partes do *software*. Para este projeto de pesquisa, estão sendo abordados os modelos de processo que são citados pelos autores Sommerville e Pressman que são básicos para o conhecimento sobre eles, destacando que para Pressman (2011) [50] as atividades genéricas de modelagem de processo podem ser utilizadas por todos os modelos de processo de *software*, porém, é aplicada uma ênfase diferente a essas atividades e defini-se também o fluxo de trabalho que recorre a cada atividade de forma diferente.

2.2.1.2.1 Modelo em Cascata

Para Sommerville (2011) [58] o modelo de cascata considera como fundamentais ao processo as atividades de especificação, desenvolvimento, validação e evolução, representadas em fases separadas do processo como a especificação de requisitos, o projeto de *software*, a implementação, os testes e assim por diante.

Já Pressman (2011) [50] diz que, o modelo em cascata, às vezes chamado também de ciclo de vida clássico, apresenta, para o desenvolvimento de *software*, uma abordagem sequencial e sistemática que se inicia com a especificação dos requisitos pelo cliente e avança ao longo do planejamento, modelagem, construção e implementação, procurando atingir um ponto máximo na manutenção gradual do *software* concluído.

O modelo em cascata, ou ciclo de vida de *software* descrito na Figura 2.7, é conhecido como tal por causa da sequência em cascata de uma fase para a outra sendo que os principais estágios deste retratam as atividades fundamentais de desenvolvimento, fundamentado na, vide Figura 2.7.

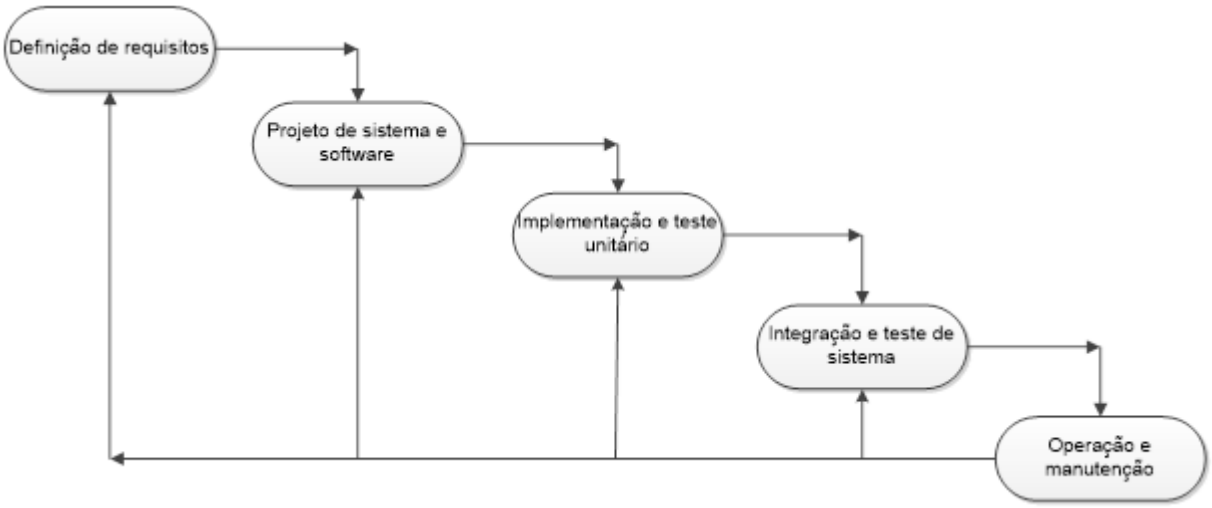

Figura 2.7: Ciclo de vida do *software*, Fonte: (PRESSMAN, 2011)[50]

Conforme a Figura 2.7, as etapas de desenvolvimento de *software* são:

1. Análise e definição de requisitos: definição de funções, restrições e os objetivos do sistema que serão especificados por intermédio da consulta aos usuários do sistema.
2. Projeto de sistema de *software*: o processo de projeto de sistema agrupa os requisitos em sistemas de hardware ou de *software*, estabelecendo a arquitetura do sistema geral, com identificação e descrição das abstrações fundamentais do sistema de software e suas relações.
3. Implementação e teste de unidades: Durante esse estágio, o projeto de *software* é compreendido como um conjunto de programas ou unidades de programa. O teste de unidades envolve verificar que cada unidade atende a suas especificações.
4. Integração e teste de sistema: as unidades de programa ou programas individuais são integrados e testados como um sistema completo a fim de garantir que os requisitos de *software* foram atendidos. Depois dos testes, o sistema de *software* é entregue ao cliente.
5. Operação e manutenção: normalmente, esta é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em operação. A manutenção envolve corrigir erros que não foram descobertos no estágio anteriores do ciclo de vida, melhorando a implementação das unidades de sistema e aumentando as funções desse sistema à medida que novos requisitos são descobertos.

Sommerville (2011) [58] ressalta que os resultados de cada fase envolvem um ou mais documentos que são aprovados, onde cada fase seguinte não deve iniciar até que

a precedente tenho sido concluída. Na prática, esses estágios se sobrepõem e trocam informações entre si. Durante o projeto são identificados problemas com os requisitos; durante a codificação, são verificados problemas de projeto, e assim por diante. O processo de *software* não é um modelo linear simples, mas envolve um sequencia de iterações das atividades de desenvolvimento.

O modelo em cascata é o paradigma mais antigo de engenharia de *software*. No entanto, nas duas últimas décadas, a crítica a esse modelo de processo tem provocado, questionamentos sobre sua eficiência (Hanna *apud* Pressman, (2011) [50]).

Dentre os diversos problemas na aplicação do modelo Cascata, Pressman (2011) [50] destaca alguns:

1. Projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Apesar de o modelo linear poder acomodar a iteração, ele o faz indiretamente. Como resultado, as modificações podem causar confusão à medida que a equipe de projetos prossegue;

2. Em geral é difícil para o cliente estabelecer todos os requisitos explicitamente. O modelo em cascata exige isso e tem dificuldade de acomodar a incerteza natural que existe no começo de muitos projetos; e

3. O cliente precisa ter paciência. Uma versão executável do programa não fica disponível até o período final do intervalo de tempo do projeto. Um erro grosseiro pode ser desastroso se não for detectado até que o programa executável seja revisto.

Para Sommerville (2011) [58], o problema com o modelo em cascata se dá pela divisão inflexível nos diferentes estágios do projeto, onde deve ser feito os acordos no estágio inicial, significando para o cliente uma resposta difícil aos requisitos, ao qual se modificam no decorrer do projeto.

Verifica-se então que o modelo em cascata é um modelo que apresenta bem definidas suas fases distintas para o desenvolvimento de *software*, mas devido a estrutura rígida ao adota-lo, apresenta certas dificuldades, como se confirma nas considerações de Sommerville (2011) [58], onde em uma análise de projetos reais, descobriu que a natureza linear do modelo em cascata leva a "estados de bloqueio" nos quais alguns membros da equipe de projeto precisam esperar que outros membros completem as tarefas dependentes. Na realidade, o tempo gasto em espera pode exceder o tempo gasto no trabalho produtivo. O estado de bloqueio tende a ocorrer mais no inicio e no fim de um processo sequencial linear.

Afirma ainda Sommerville (2011) [58], que o trabalho de *software*, atualmente, se dá em um ritmo elevado e submetido a uma torrente sem fim de modificações, sendo estas modificações de características, conteúdo da informação e funções, tornando o modelo em

cascata comumente inapropriado para esse tipo de trabalho. Mas este pode servir como um modelo de processo útil em situações onde os requisitos são fixos e o trabalho deve seguir adiante de modo linear até o fim.

2.2.1.2.2 Modelos Evolucionários

Segundo Sommerville (2011) [58], o desenvolvimento evolucionário, apresentado na figura 8, traz uma abordagem em que as atividades de especificação, desenvolvimento e validação se intercalam. A partir de especificações abstratas, um sistema inicial é rapidamente desenvolvido, em seguida são refinadas com informações do cliente para que se produza um sistema satisfatório às necessidades.

Sendo o *software* algo que evolui com o tempo, Pressman (2011) [50] diz que frequentemente os requisitos de produto sofrem mudanças à medida que o desenvolvimento do *software* avança, dificultando então, a entrega do produto final somado a pressão com os prazos de entrega reduzidos, dentre outras dificuldades. Com isso, o engenheiro de software necessita de um modelo de processo para acomodar um produto que evolui com o tempo ao qual esteja projetado explicitamente.

Em resumo, os modelos evolucionários são iterativos. Eles são caracterizados de forma a permitir aos engenheiros de *software* desenvolver versões cada vez mais completas de *software*, vide Figura 2.8 (PRESSMAN, 2011) [50].

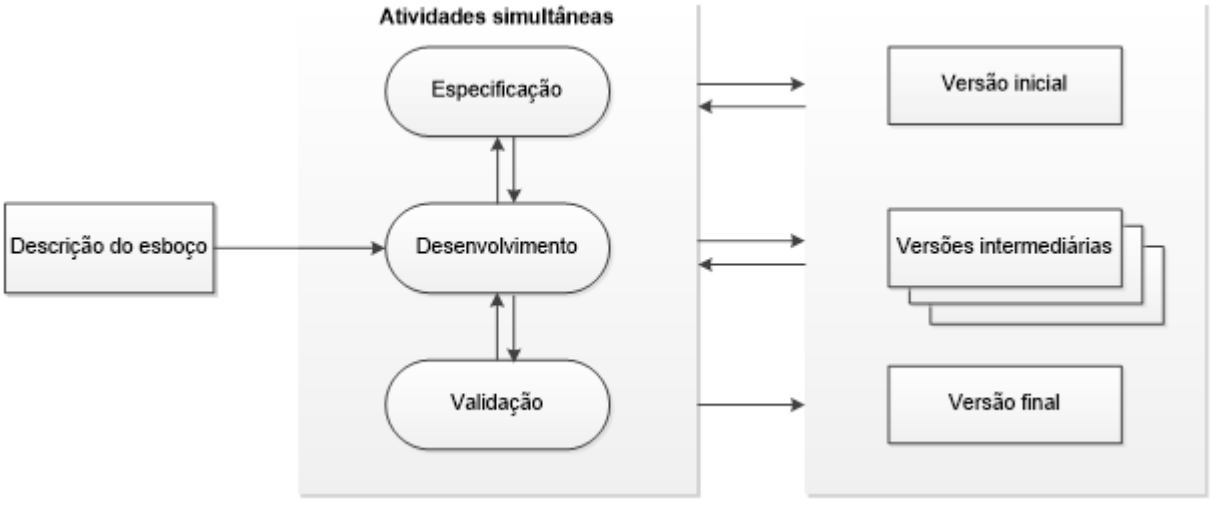

Figura 2.8: Desenvolvimento Evolucionário, Fonte: (SOMMERVILLE, 2011) [58]

De acordo com Sommerville (2011) [58], para o desenvolvimento evolucionário existem dois tipos:

1. Desenvolvimento exploratório: em que o objetivo do processo é trabalhar com o cliente, a fim de explorar seus requisitos e entregar um sistema final. O desenvolvimento se inicia com as partes do sistema que são propostas pelo cliente;

2. Confecção de protótipos descartáveis: em que o objetivo do desenvolvimento evolucionário é compreender os requisitos do cliente e, a partir disso, desenvolver uma melhor definição de requisitos para o sistema. O protótipo se concentra em fazer experimentos com partes dos requisitos que estejam mal compreendidas.

Dentro do modelo evolucionário, Pressman (2011) [50], destaca como exemplo:

1. Prototipagem: é um paradigma em que se inicia a comunicação com o cliente, onde este e os engenheiros de *software* determinam os objetivos gerais do *software*, apresenta-se as carências que são conhecidas e delimitam as áreas em que há necessidade de mais definições. A iteração é planejada com rapidez e a modelagem ocorre no modo de um "projeto rápido", concentrando-se em representar aqueles aspectos do software que para o cliente estarão visíveis, levando, então, a uma construção de um protótipo que é implantado e depois avaliado pelo cliente e utilizando como o *feedback* para se refinar os requisitos do *software*.

2. Modelo Espiral: é um modelo evolucionário de processo de *software* em que se junta os aspectos sistemáticos e controlados do modelo em cascata com a natureza iterativa da prototipagem. O *software* é desenvolvido em várias versões evolucionárias em que as primeiras iterações podem ser um protótipo ou modelo de papel e nas últimas versões são produzidas versões cada vez completas do sistema submetido à engenharia.

3. Modelo de desenvolvimento concorrente: às vezes chamado de engenharia corrente, é possível representar esquematicamente com uma série de atividades, ações e tarefas de engenharia de *software* e uma série de estados associados, como por exemplo, a atividade de modelagem determinada para o modelo espiral sendo cumprida pela invocação das seguintes ações: especificação e projeto e prototipagem e/ou modelagem de análises. Este modelo determina para cada uma das ações, atividades ou tarefas de engenharia de *software* um seguimento de eventos que dispararão transições de estado para estado, como por exemplo, a descoberta de uma inconsistência no modelo de análise durante os primeiros estágios do projeto, tendo como consequência, a geração do evento de correção do modelo de análise, onde será disparada uma passagem da atividade de análise do estado de ponto para o de aguardando modificações. O modelo concorrente de processo pode ser aplicado a todos os tipos de desenvolvimento de *software* e oferece uma imagem precisa do estado atual do projeto.

Para Sommerville (2011) [58], muitas vezes a utilização do modelo evolucionário para o desenvolvimento de *software* é mais eficaz que o modelo em cascata, pois a vantagem de um processo baseado na abordagem evolucionária é que a especificação pode ser realizada de forma gradativa. Conforme os usuários desenvolvem uma melhor compreensão de seus problemas, isso pode ser refletido no sistema de *software*. Todavia, existem três problemas partindo de uma perspectiva de engenharia de gerenciamento:

1. O processo não é visível: os gerentes necessitam que o desenvolvimento seja regular, para que possam medir o processo. Se os sistemas são desenvolvidos rapidamente, não é viável produzir documentos que reflitam cada versão do sistema;
2. Os sistemas frequentemente são mal-estruturados: a mudança constante tende a corromper a estrutura do *software*. Incorporar modificações no *software* torna-se cada vez mais difícil e oneroso; e
3. Podem ser exigidas ferramentas e técnicas especiais: elas possibilitam rápido desenvolvimento, mas podem ser incompatíveis com outras ferramentas ou técnicas, e relativamente poucos pessoas podem ter a habilidade necessária para utilizar.

2.2.1.2.3 Modelos incrementais

Segundo Pressman (2011) [50], existem várias situações em que os requisitos estão consideravelmente bem definidos, porém um processo puramente linear é eliminado pelo escopo global do esforço de desenvolvimento, além de que pode haver a necessidade de entregar rapidamente ao cliente funcionalidades do *software*, depois refina-lo e expandir para versões conseguintes do software. Nessas ocasiões é escolhido o modelo de processo que desenvolverá o *software* em incrementos.

Conforme Sommerville (2011) [58], enquanto o modelo em cascata requer dos clientes de um *software* o comprometimento com os conjuntos de requisitos específicos e que os projetistas se comprometam em particular com as estratégias de projeto, sendo que as mudanças requeriam "retrabalho" referentes ao requisito, o modelo evolucionário possibilita a postergação dos requisitos e das decisões do projeto.

O modelo incremental (Figura 2.9) surge como uma forma de minimizar o "retrabalho" no processo de desenvolvimento e de apresentar aos clientes alguma oportunidade de deixar para depois decisões referentes aos seus requisitos detalhados até o momento em que possam ter uma experiência com o sistema (SOMMERVILLE, 2011) [58].

Em resumo, segundo Pressman (2011) [50], pode-se dizer que o modelo incremental combina elementos do modelo em cascata aplicado de maneira iterativa, isto é, os requisitos básicos são satisfeitos, mas muitas características suplementares, algumas

conhecidas, outras desconhecidas, deixam de ser elaboradas. O núcleo do produto é usado pelo cliente ou passa por uma revisão detalhada. Um plano é desenvolvido para o próximo incremento como resultado do uso e/ou avaliação. O plano visa à modificação do núcleo do produto para melhor satisfazer as necessidades do cliente e a elaboração de características e funcionalidades adicionais. Esse processo é repetido após a realização de cada incremento, até que o produto completo seja produzido.

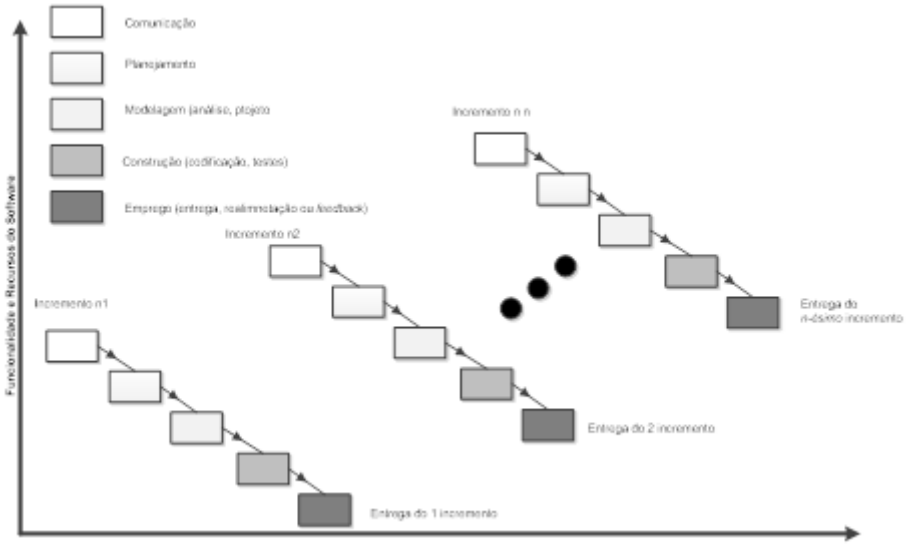

Figura 2.9: Desenvolvimento incremental, Fonte: (PRESSMAN, 2011)[50]

2.2.1.2.4 Modelo de Desenvolvimento orientado a reuso

Segundo Sommerville (2011) [58] esta abordagem de desenvolvimento orientado a reuso ocorre independentemente da utilização de algum processo genérico empregado, mas sim se utiliza de uma vasta base de componentes de *software* reusáveis podendo ser acessados e que seja oferecida alguma infraestrutura para esses componentes. Além da especificação de requisitos e do estágio de validação, todos os estágios intermediários deste processo são diferentes, sendo eles apresentados na Figura 2.10:

Figura 2.10: Desenvolvimento orientado a reuso, Fonte: (SOMMERVILLE, 2011)[58]

1. Análise de componentes: Considerando a especificação de requisitos é feita uma busca de componentes para implementar essa especificação. Em geral, não existe uma combinação exata e os componentes que podem ser utilizados fornecem somente parte da funcionalidade requerida;

2. Modificação de requisitos: durante esse estágio os requisitos são analisados, utilizando-se as informações sobre os componentes que foram encontrados. Eles são então modificados para refletir os componentes disponíveis. Quando as modificações forem impossíveis, a atividade de análise de componentes poderá ser feita a fim de procurar soluções alternativas;

3. Projeto de sistemas com reuso: durante essa fase e infraestrutura do sistema é projetada ou uma infraestrutura existente é utilizada. Os projetistas levam em conta os componentes que são reutilizáveis e organizam a infraestrutura para lidar com esse aspecto. Um novo *software* poderá ter de ser projetado, se os componentes reutilizáveis não estiverem disponíveis.

4. Desenvolvimento e integração: o *software* que não puder ser comprado será desenvolvido, e os componentes e sistemas *Commercial Off-The-Shelf* (COTS) serão integrados, a fim de criar um sistema. A integração de sistemas, nesse modelo, pode ser parte do processo de desenvolvimento, em vez de uma atividade separada.

Além dos modelos citados para a construção de *software*, existem também as metodologias de desenvolvimento que são parte integrante da engenharia de *software* apresentadas no tópico seguinte.

2.2.2 Métodos de desenvolvimento de *software*

Metodologia é o conjunto de métodos, regras e postulados usados em determinada disciplina, e sua aplicação (FERREIRA, 2011) [32]. Todavia, as metodologias de desenvolvimento de *software* nada mais são que processos que seguem um conjunto de regras e métodos que se aplicam na produção de *software*, demonstradas neste estudo como tradicionais e ágeis.

2.2.2.1 Metodologias Tradicionais

De acordo com Machado (2011) [40], as metodologias tradicionais, também chamadas de metodologias "pesadas", têm como aspecto marcante a sua divisão em fases e/ou etapas, sendo elas bem definidas e englobando atividades como análise, modelagem, desenvolvimento e testes.

As metodologias tradicionais tratam-se de métodos de processo que se apresentam no tópico que são tratados sobre os modelos de processo. Mas nesta seção, como exemplo de aplicação das metodologias tradicionais, aponta-se sobre o Processo Unificado e o *Rational Unified Process* (RUP).

2.2.2.1.1 Processo Unificado

Segundo Pressman (2011) [50], o surgimento do processo unificado se deu por volta da década de 80 em que na época, métodos e linguagens de programação orientadas a objeto ganhavam audiência entre os engenheiros de *software* e uma grande variedade de projetos e processos de análise orientada a objetos, foram propostos e, na mesma época, um modelo de processo orientado a objeto foi introduzido, sempre se buscando a melhor metodologia dentre as existentes, mas nenhum dominou o cenário de engenharia de *software*.

No inicio da década de 90, James Rumbaugh, Ivan Jacabson e Grady Booch iniciaram um trabalho em um "método unificado" que reuniu as melhores características de cada um dos seus métodos individuais e adotou características adicionais, no ramo de Orientação a Objeto (OO), por outros especialistas, surgindo então a *Unified Modeling Language* (UML). Por volta de 1997 se tornou uma norma industrial para desenvolvimento de *software* (PRESSMAN, 2011) [50].

De acordo com Scott *apud* Pressman (2011) [50] na definição geral, o processo unificado é um agrupamento de atividades que se executa para a transformação de um agrupamento de requisitos do cliente em um sistema de *software*. Todavia, este também é uma estrutura genérica de processo onde se pode customizar, adicionando-se ou removendo-se, atividades baseado nas necessidades especificas e recursos disponíveis.

Para Pressman (2011) [50] o processo unificado é de certo modo uma tentativa de se sustentar nos melhores aspectos e recursos dos modelos convencionais caracterizando-os de uma forma que implemente vários dos princípios de desenvolvimento ágil, vide Figura 2.11.

O Processo Unificado segundo Pressman (2011) [50] representado pela Figura 2.11 são vários processos de desenvolvimento de *software*, que apresentam atividades genéricas de concepção, elaboração, construção, transição e produção, as quais:

1. Concepção: abrange atividades de comunicação com o cliente e de planejamento;
2. Elaboração: inclui a comunicação com o cliente e atividades de modelagem do modelo genérico de processo;
3. Construção: é idêntica à atividade de construção definida para o processo genérico de *software*;

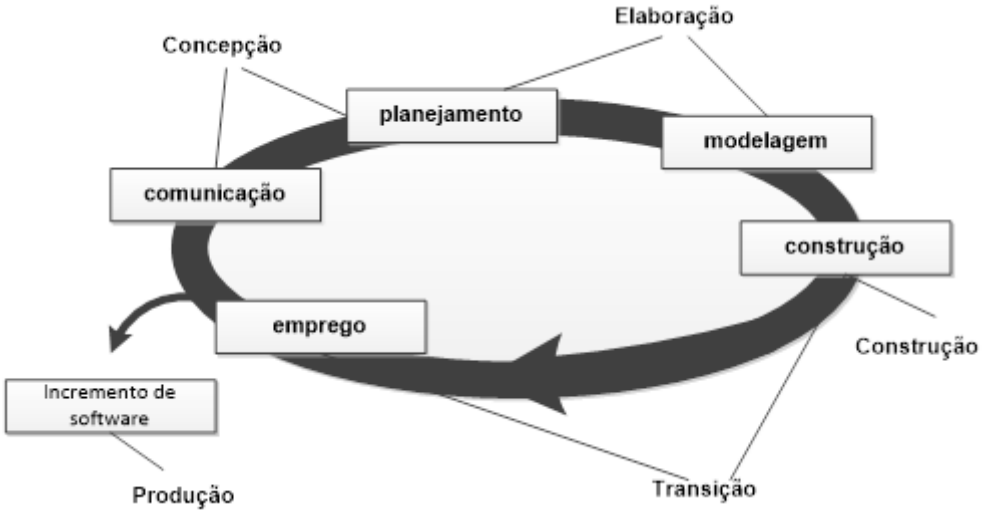

Figura 2.11: Processo Unificado, Fonte: (PRESSMAN, 2011)[50]

4. Transição: abrange os últimos estágios da atividade genérica de construção e a primeira parte da atividade genérica de implantação; e
5. Produção: coincide com a atividade de implantação do processo genérico.

Sendo o processo unificado um processo orientado por casos de uso, centrado na arquitetura e iterativo e incremental, Scott *apud* Pressman (2011) [50] definem conceitos sobre cada uma dessas abordagens que são descritas brevemente a seguir:

1. Caso de Uso: é uma sequência de ações, executadas por um ou mais atores (pessoas ou entidades não-humanas fora do sistema) e pelo próprio sistema, que produz um ou mais resultados de valor para um ou mais atores;
2. Centrado na Arquitetura: é a organização fundamental do sistema como um todo. Entre os aspectos de uma arquitetura, estão incluídos elementos estáticos, elementos dinâmicos, o modo como estes elementos trabalham juntos e o estilo arquitetônico total que guia a organização do sistema. A arquitetura também se refere a questões como desempenho, escalabilidade, reuso e restrições econômicas e tecnológicas;
3. Iterativo e Incremental: é um miniprojeto que resulta em uma versão do sistema liberada interna ou externamente. Supõe-se que essa versão ofereça um melhora incremental sobre a anterior, motivo pelo qual iteração é chamada de incremento.

2.2.2.1.1.1 Unified Process (RUP)

O *Rational Unified Process* (RUP) é um produto/processo iterativo, incremental e customizável de Engenharia de *software* que foi inicialmente desenvolvido e comercia-

lizado pela Rational, e desde 2003 pertence à *International Business Machines* (IBM) (SOMMERVILLE, 2011) [58].

Segundo a IBM o RUP é uma ampla biblioteca que oferece práticas testadas que evoluíram e continua evoluindo para colaborar com uma comunidade abrangente de parceiros, clientes e acadêmicos para reflexão de suas experiências e pensamentos mais recentes.

As práticas fornecidas através do RUP são voltadas a um conjunto de princípios e características mais sucedidas e melhores práticas das organizações de sistemas de *software* do mundo (IBM, 2013) [35], com os seguintes princípios:

1. Adaptar o processo;
2. Balancear as prioridades dos *Stakeholders*;
3. Trabalhar em equipes;
4. Demonstrar valor iterativamente;
5. Elevar o nível de abstração; e
6. Focar em qualidade.

Divide-se a arquitetura básica do RUP, de acordo com Júnior *apud* Pressman (2011) [50], em duas dimensões:

1. Horizontal: representam o tempo de vida de um projeto, os aspectos do ciclo de vida do processo de engenharia de um sistema, de acordo com o decorrer do projeto. Essa dimensão demonstra o aspecto dinâmico do processo, suas fases, iterações e milestones;
2. Vertical: representam os grupos de atividades lógicas que são realizadas durante o decorrer do tempo. Essa dimensão demonstra o aspecto estático do processo, que será composto por disciplinas, atividades, fluxos, artefatos e papéis.

De acordo com a IBM, a solução RUP possibilita que sejam liberadas versões incrementais para *feedback* continuo e inicial. Consegue-se isso por intermédio da divisão dos projetos em conjuntos de iterações e em cada uma delas, planejam-se os requisitos e o projeto, a implantação e o teste dos aplicativos, produzindo assim uma versão que pode ser entregue.

Em complemento, apontam-se resumidamente, também, as fases e disciplinas que ocorrem no RUP (Figura 2.12) que são bem descritas por Junior *apud* Pressman (2011) [50]:

1. Fases:

   (a) Iniciação: deve nesta fase conseguir dos stakeholders do projeto um consenso relacionado aos objetivos do ciclo de vida do projeto. Essa fase é focada em endereçar riscos de requisitos e negócios antes de continuar com o projeto;

   (b) Elaboração: aqui se fecha a análise da arquitetura do sistema, estabelecendo uma base sólida para o design e implementação do software;

   (c) Construção: na fase de construção, fecham-se os requisitos restantes e se completa o desenvolvimento do software, baseado na arquitetura definida; e

   (d) Transição: o foco da fase de transição é garantir que o desenvolvimento esteja disponível para seus usuários finais.

2. Disciplinas:

   (a) Modelagem de Negócios: Garantir que os objetivos e expectativas de todos os *stakeholders* do projeto estejam alinhados com os objetivos da organização, derivar desses objetivos, os requisitos de *software* que deverão ser atendidos para solucionar;

   (b) Análise e Design: transformar os requisitos em desenhos do *software* que será construído. Devem-se produzir as especificações técnicas que deverão ser seguidas na implementação de cada caso de uso do software;

   (c) Implementação: transformar os modelos lógicos e físicos na análise e design em código-fonte utilizável e testar unitariamente o código produzido;

   (d) Teste: encontrar, documentar e endereçar os defeitos encontrados na qualidade do *software* produzido. Esses defeitos surgem na comparação entre o que foi produzido com o que foi exposto nos requisitos e modelos lógicos e físicos do produto;

   (e) Implantação: Garantir que o *software* produzido fique disponível para seus usuários finais;

   (f) Gerenciamento de Projeto: Balancear objetivos que competem entre si, gerenciar riscos, monitorar o projeto e tratar regras que garantirão a entrega de um produto que irá atender às expectativas de cliente (os patrocinadores do projeto) e usuários finais; e

   (g) Ambiente: configurar o ambiente para que o processo e suas atividades possam ser executadas. Devem-se prover aqui os processos e ferramentas necessárias para que todas as atividades do projeto possam ser devidamente executadas por cada papel.

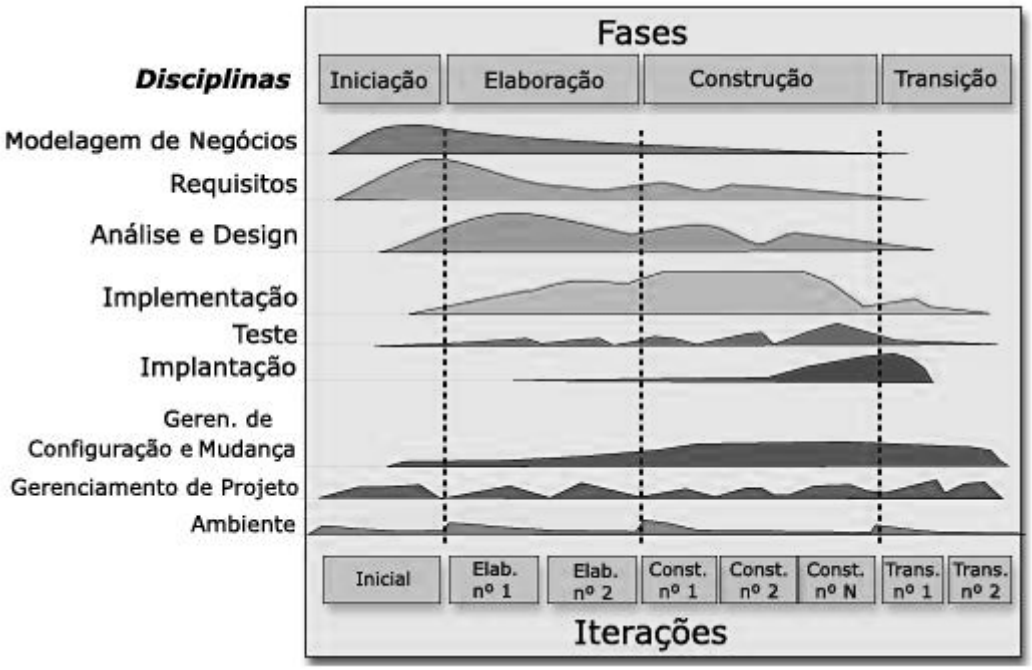

Figura 2.12: Ciclo de vida do RUP, Fonte: (PRESSMAN, 2011)[50]

Por fim a IBM (2013)[35] observa que para auxiliar o alcance das metas de iteração, a solução RUP inclui também modelos, conceitos, *cheklist* de verificação, diretrizes que ajudam no aumento da produtividade da equipe e garantia da consistência, além da melhora da qualidade.

Nesse sentido, no próximo tópico serão apresentados os conceitos de métodos ágeis.

2.2.2.2 Métodos Ágeis

Este tópico aborda sobre as metodologias ágeis de desenvolvimento de *software* que reúne várias características de processos de *software*.

De acordo com Pressman (2011) [50], a engenharia de *software* ágil combina um conjunto de diretrizes, que enfatizam a entrega em contraposição à análise e ao projeto (apesar de não desencorajar essas atividades) e a comunicação ativa e contínua entre clientes e desenvolvedores – de desenvolvimento e uma filosofia em que:

1. Encoraja a entrega incremental do software logo de início e a satisfação do cliente;
2. Equipes pequenas de projeto altamente motivadas;
3. Métodos informais;
4. Simplicidade global de desenvolvimento e produtos mínimos de engenharia de software.

A metodologia Ágil nasceu em 2001, quando um grupo de especialistas se reuniu em Utah para rascunhar aquilo que agora é conhecido como o Manifesto Ágil, focalizado no lado do desenvolvimento de *software* (PHAN, 2011)[48].

A metodologia ágil apresenta em sua abordagem fundamentos citados abaixo do Manifesto de Desenvolvimento Ágil de *Software*: *"Estamos descobrindo melhores maneiras de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo"*. Através deste trabalho, o método ágil valoriza os seguintes aspectos, vide Figura 2.13:

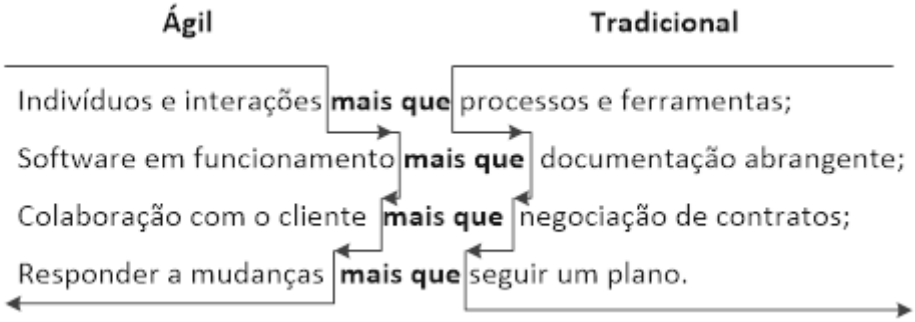

Figura 2.13: Ágil x Tradicional, Fonte: (COHN, 2011)[28], adaptado pelo autor

Os autores Beck, Beedle e Bennekum (2001)[10] consideram que mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Segundo Pressman (2011)[50], normalmente um manifesto está associado a um movimento político emergente, um movimento em que se ataca a velha guarda e apresente mudanças revolucionárias (esperando-se que seja para melhor) e que, de alguma forma, é o que o desenvolvimento ágil é. E embora as ideias que guiam o desenvolvimento ágil estivessem presentes há anos, apenas na década de 90 elas foram concretizadas em um "movimento", e que, em essência, os métodos ágeis foram elaborados num esforço que buscava vencer as fraquezas percebidas e reais da engenharia de *software* convencional.

A respeito da agilidade no contexto de engenharia de *software*, Pressman (2011) [50] aponta uma discussão fornecida por Ivar Jacobson que diz:

> "Agilidade tornou-se atualmente uma palavra mágica quando se descreve um processo moderno de *software*. Tudo é ágil. Uma equipe ágil é uma equipe esperta, capaz de responder adequadamente a modificações. Modificação é aquilo para o qual o desenvolvimento de *software* está principalmente focado. Modificações no software que está sendo construído, modificações nos membros da equipe, modificações por causa de novas tecnologias, modificações de todas as espécies que podem ter impacto no produto que eles constroem ou no projeto que cria o produto. O apoio para as modificações deveria ser incorporado em tudo que fazemos em software; algo que se adota porque está no coração e na alma do *software*. Uma equipe ágil reconhece que o *software* é desenvolvido por indivíduos trabalhando em equipes e que as especialidades dessas pessoas e sua capacidade de colaborar estão no âmago do sucesso do projeto". (JACOBSON *apud* PRESSMAN, (2011, p. 122)) [50]

Além dos fundamentos citados anteriormente, há também os princípios que estão por trás do Manifesto ágil (BECK, BEEDLE e BENNEKUM, 2001)[10]:

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de *software* com valor agregado;
2. Mudanças de 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 a excelência técnica e bom design aumenta a agilidade;
10. Simplicidade a arte de maximizar a quantidade de trabalho não realizado é essencial;
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis;
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

A importância da abordagem das metodologias ágeis se dá segundo Pressman (2011) [50], devido ao ambiente moderno de negócios em que se criam sistemas baseados em computador e que os produtos de *software* são apressados e sempre mutáveis. Esta representa uma razoável alternativa para a engenharia de software convencional para certos tipos de projetos de *software* e certas categorias de *software*, e tem-se demonstrado que ela entrega rapidamente sistemas bem-sucedidos.

Um ponto a se tratar dentro da abordagem da metodologia ágil é sobre os fatores humanos em que, conforme Pressman (2011) [50] no uso do método ágil inicialmente

existe uma resistência cultural da equipe, destacando a importância dos "fatores pessoais" no desenvolvimento ágil bem-sucedido, onde "o desenvolvimento ágil enfoca os talentos e habilidades dos indivíduos moldando o processo e pessoas e equipes específicas" (Cockburn *apud* Pressman, (2011)) [50], e tendo para esta declaração como ponto-chave é que o processo se molda conforme as necessidades da equipe e das pessoas, e não o contrário.

Existe um debate considerável entre a aplicabilidade e benefícios trazidos pelo desenvolvimento ágil em contraposição aos processos convencionais de engenharia de *software*. Mesmo dentro do campo ágil, há modelos de processos que são propostos onde cada qual traz uma abordagem sutilmente diferenciada para a questão da agilidade e que apresentam um afastamento significativo da convencional engenharia de *software*. Mas, muitos conceitos ágeis são simplesmente adaptações de bons conceitos de engenharia de *software* (Pressman, (2011) [50].

São apresentados a seguir, de forma resumida, alguns dos diferentes modelos de processos ágeis de desenvolvimento de *software*.

2.2.2.2.1 *eXtreme Programming* (XP)

O *eXtreme Programming* (XP) é uma das principais referências sobre metodologias ágeis atualmente e que, conforme Sommerville (2011)[58], dos métodos ágeis, ele é talvez o mais utilizado e o mais conhecido. Beck o desenvolver do método XP (SOMMERVILLE, 2011)[58] o método XP tem esse nome devido à abordagem ter sido desenvolvida para impulsionar práticas reconhecidas como boas, como, a níveis extremos, o desenvolvimento iterativo.

"Em *eXtreme Programming*, os requisitos são expressos como cenários (chamados de histórias do usuário), que são implementados diretamente como uma série de tarefas. Os programadores trabalham em pares e desenvolvem testes para cada tarefa antes de escreverem o código. Quando o novo código é integrado ao sistema, todos os testes devem ser executados com sucesso. Há um curto intervalo entre os releases do sistema" (SOMMERVILLE, 2011)[58].

A metodologia de desenvolvimento XP, segundo Pressman (2011)[50], utiliza como paradigma de desenvolvimento predileto uma abordagem orientada a objetos. Esta metodologia inclui um conjunto de práticas e regras que acontecem no contexto de quatro atividades de arcabouço que são:

1. Planejamento: que consiste na criação de um conjunto de histórias em que as funcionalidades e características requeridas para o software a ser desenvolvido são descritas, onde o cliente escreve cada história em um cartão de indexação atribuindo

um valor (uma prioridade) e os membros da equipe XP avalia atribuem um custo medido em semanas de desenvolvimento;

2. Projeto: segue o princípio de KIS (Keep It Simple, mantenha a simplicidade) rigorosamente, pois um projeto simples tem preferência em relação uma representação mais complexa, além de oferecer diretrizes de implementação para uma história da forma como ela está escrita, e o projeto de funcionalidade extra é desencorajado, pois esta é considerada pelo desenvolvedor como uma exigência futura;

3. Codificação: é a atividade em que o XP recomenda que após o desenvolvimento das histórias e que o trabalho preliminar for feito, a equipe não avance para o código, mas, que ao invés disso, se desenvolva uma série de testes unitários que exercitarão cada uma das histórias que devem ser incluídas na atual versão (incremento de software). A programação em pares é um conceito-chave utilizado nesta atividade que consiste no trabalho de duas pessoas, juntas em uma estação de trabalho, para criar o código referente a uma história;

4. Teste: esta atividade é onde se realizam os testes unitários que são criados e que devem ser implementados utilizando um arcabouço que lhes permita a automatização. Sommerville (2011)[58] apresenta também uma tabela que resume as práticas e os princípios do XP, vide Tabela 2.2:

Tabela 2.2: Práticas de Extreme Programming, Fonte: (SOMMERVILLE, 2011)[58]

Principio	Descrição
Planejamento incremental	Os requisitos são gravados em cartões de estória e as estórias que serão incluídas em um release são determinadas pelo tempo disponível e sua prioridade. Os desenvolvedores dividem essas estórias em "Tarefas".
Pequenas *releases*	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. *Releases* do sistema são frequentemente adicionam funcionalidade ao primeiro *release*.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais e nada mais
Desenvolvimento *test-first*	Um *framework* de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja manutenível
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem melhorias de código, isso mantém o código simples e manutenível.
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de *expertise*. Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo sustentável	Grandes quantidades de horas-extra não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo de *eXtreme Programming*, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

De acordo com Sommerville (2011) [58], os clientes estão envolvidos intimamente na priorização e especificação dos requisitos do sistema em um processo XP, onde o cliente faz parte da equipe de desenvolvimento e trata a respeito dos cenários com outros membros da equipe em vista que, os requisitos não se encontram especificados como uma lista de funções do sistema requerida. A equipe e o cliente desenvolvem juntos um cartão de "histórias"que englobam as necessidades do cliente e então, a equipe de desenvolvimento tenta a implementação desse cenário em uma versão futura do *software*, vide Figura 2.14.

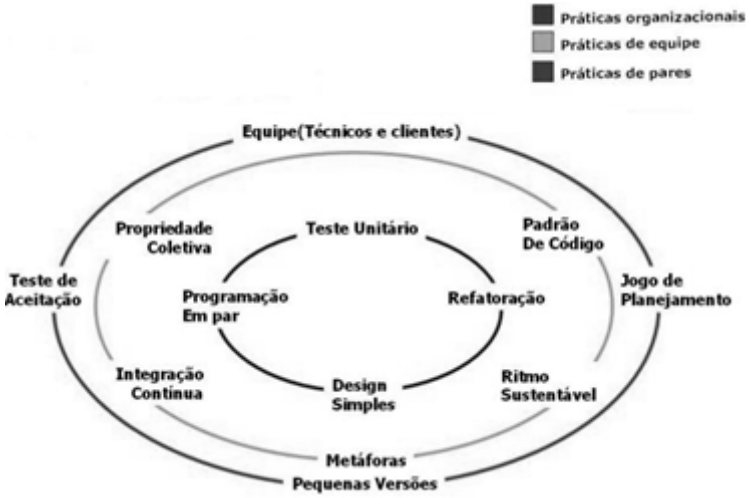

Figura 2.14: Práticas XP, Fonte: (PRESSMAN, 2011)[50]

2.2.2.2.2 Desenvolvimento Adaptativo de *Software* (DAS)

De acordo com Pressman (2011)[50], o Desenvolvimento Adaptativo de *Software* (DAS) como ilustra a Figura 2.15, é uma técnica para a construção de *software* proposta por Jim Highsmith, tendo como apoio filosófico a concentração na auto-organização da equipe e na colaboração humana.

Auto-organização é uma propriedade de sistemas adaptativos complexos semelhantes a um "Ahhh" coletivo, aquele momento de energia criativa em que a solução para algum problema impertinente surge. A auto-organização aparece quando agentes individuais independentes (células em um corpo, espécies em um ecossistema, desenvolvedores em uma equipe) cooperam (colaboram) para criar resultados emergentes. Um resultado emergente é uma propriedade além da capacidade de qualquer agente individual. Por exemplo, neurônios individuais no cérebro não possuem conscientização, mas coletivamente a propriedade de conscientização emerge. Tendemos a considerar esse fenômeno de emergência coletiva como acidental ou, pelo menos, sem regra não-confiável. O estudo de auto-organização está provando que está errada (PRESSMAN, 2011, p. 134)[50].

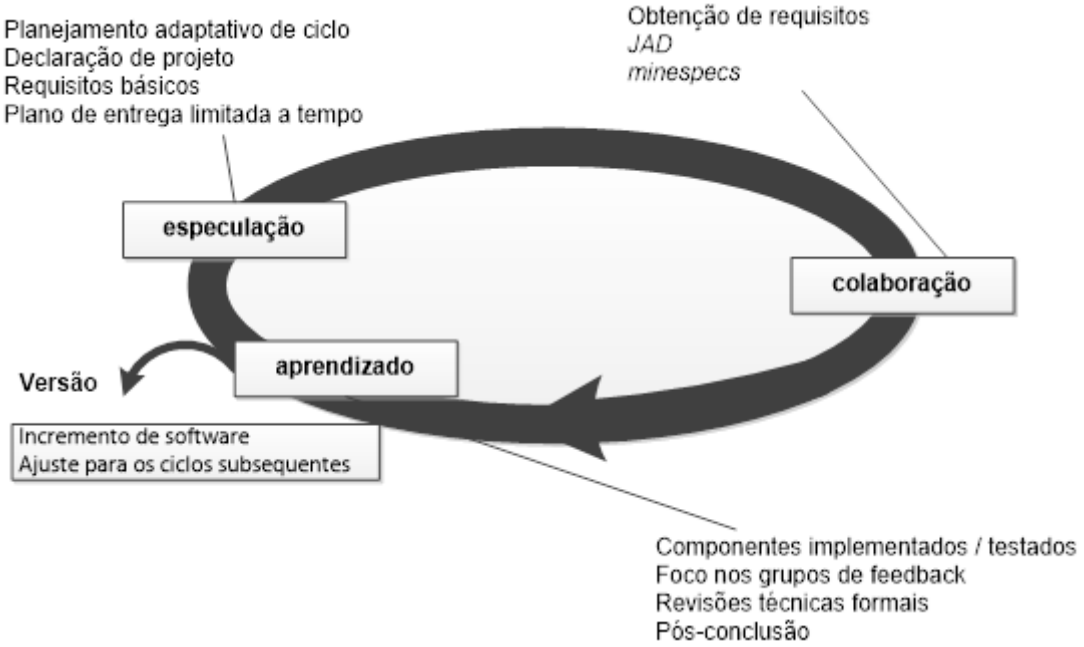

Figura 2.15: Desenvolvimento Adaptativo de *Software* (DAS), Fonte: (PRESSMAN, 2011)[50]

O ciclo de vida proposto por Highsmith *apud* Presmman (2011)[50] incorpora três fases que são:

1. Especulação: durante esta atividade se inicia o projeto e o planejamento do ciclo adaptativo – que utiliza as informações de iniciação do projeto – que define o conjunto de ciclos de versão (incrementos de software) que serão precisos para o projeto;
2. Colaboração: é atividade em que se encontra o pessoal motivado que trabalha junto de forma que se multiplicam os talentos e resultados criativos além de seu número absoluto.
3. Aprendizado: nesta atividade, conforme os membros de uma equipe DAS, iniciam o desenvolvimento dos componentes que fazem parte de um ciclo adaptativo com sua ênfase tanto no progresso rumo a um ciclo completo quanto no aprendizado. As equipes DAS têm três modos de aprendizagem:
   (a) Foco nos grupos: se fornece um *feedback* pelos clientes e/ou usuários sobre os incrementos de *software* que são entregues indicando se o produto está satisfazendo ou não às necessidades do negócio;
   (b) Revisões técnicas formais: os componentes de software que são construídos passam pela revisão dos membros da equipe DAS, vão aprendendo conforme avançam e aperfeiçoando a qualidade;

(c) Pós-Conclusão: observado o próprio processo e desempenho, a equipe DAS torna-se introspectiva.

Ainda conforme Pressman (2011)[50], o DAS produz equipes de projeto de *software* que têm uma probabilidade muito maior de sucesso enfatizando globalmente a colaboração interpessoal, o aprendizado individual e a dinâmica de equipes auto-organizáveis.

2.2.2.2.3 *SCRUM*

De acordo com Phan (2011) [48], o termo *Scrum* surgiu em 1986 na *Harvard Business Review* em um artigo entitulado *"The new product development game"* que quer dizer "O novo jogo do desenvolvimento de produtos" que foi publicado por Ikujiro Nonaka e Hirotaka Takeuchi em que eles descrevem toda uma abordagem onde equipes de projetos são formadas por pequenas equipes multifuncionais, trabalhando rumo a um objetivo comum com sucesso.

Segundo Pressman (2011)[50], o termo *Scrum* deriva-se de uma formação que se encontra no jogo chamado rugby e foi desenvolvido por Jeff Sutherland e por sua equipe no início da década de 1990.

O que ocorreu foi que, Jeff Sutherland, vice-presidente de engenharia da Easel, Inc, enquanto trabalhava em uma ferramenta de Análise de Projetos Orientados a Objeto, observou que para sua equipe era necessária uma versão mais aprimorada do Rapid Application Development (RAD) e queria um processo semelhante ao *Scrum* em que ao final de um incremento o *Chief Executive Officer* (CEO) da Easel observa-se em vez de gráficos de Gantt em papel fosse apresentado uma demonstração de código funcional (PHAN, 2011)[48]

Mais ou menos na mesma época, ainda conforme Phan (2011) [48], Ken Schwaber (2011) [56] pesquisou ativamente uma forma de auxiliar sua empresa para a melhora dos processos de *software* na busca de aumentar a produtividade das equipes. A partir de uma análise profunda sobre a forma como as independentes desenvolvedoras bem-sucedidas de *software* construíam seus *software*, ele constatou que de todas elas o processo de desenvolvimento era semelhante e que se utilizavam de processos empíricos.

E então, continuando e complementando com o Phan (2011) [48] e Schwaber (2011) [56], em 1995 Sutherland e Schwaber (2011) trabalharam juntos a pedido da Object Management Group (OMG) com o propósito de resumir o que eles tinham aprendido com o passar dos anos. Assim, eles criaram uma metodologia nova chamada *Scrum*.

2.2.2.2.3.1 Visão geral do *SCRUM*

Para se iniciar uma visão geral do *Scrum*, primeiramente, apresenta-se a seguinte definição que diz, segundo Schwaber (2011) [56], que o *Scrum* é uma estrutura processual (*framework*) para o suporte de desenvolvimento e manutenção de produtos complexos que consiste em regras, eventos, artefatos e papéis que se associam às equipes do *Scrum* onde cada componente dentro desta estrutura processual serve para um propósito específico e é indispensável ao seu uso e sucesso.

De acordo com Phan (2011) [48], a equipe de *Scrum* é formada por um *Scrum Master*, uma equipe de desenvolvimento (ou, simplesmente "equipe") com todas as habilidades necessárias, como teste, codificação, projeto e coleta de requisitos, e de um *Product Owner* (Dono do Produto) e esta equipe deve ser uma equipe multifuncional.

Em complemento ao que afirma Phan (2011) [48] e Schwaber (2011) [56], as regras do *Scrum* reúnem os artefatos, eventos e papéis, de forma a administrar as interações e relações entre eles. É um *framework* onde as pessoas podem resolver e tratar problemas adaptativos e complexos enquanto se entrega produtos com o mais alto valor possível de forma produtiva e criativa. O *Scrum* é leve, simples de entender e extremamente difícil de dominar. Para Pressman (2011) [50], o *Scrum* tem princípios que são utilizados como guia para as atividades de desenvolvimento inseridas em um processo que incorpore as seguintes atividades de arcabouço: análise, requisito, projeto, entrega e evolução. O *Scrum* tem consistentes princípios com o manifesto ágil que são:

1. Pequenas equipes de trabalho são organizadas de modo a "maximizar a comunicação, minimizar a supervisão e maximizar o compartilhamento de conhecimento tácito informal";

    (a) O processo precisa ser adaptável tanto a modificações técnicas quanto de negócios "para garantir que o melhor produto possível seja produzido";

    (b) O processo produz frequentes incrementos de *software* "que podem ser inspecionados, ajustados, testados, documentados e expandidos";

    (c) O trabalho de desenvolvimento e o pessoal que o realiza é dividido "em partições claras, de baixo acoplamento, ou em pacotes";

    (d) Testes e documentação constantes são realizados à medida que o produto é construído;

    (e) O processo *Scrum* fornece a habilidade de declarar o produto "pronto" sempre que necessário (porque a concorrência acabou de entregar, porque a empresa precisa de dinheiro, porque o usuário/cliente precisa das funções, pois foi para esta data que foi prometido...)". Propõem-se de acordo com Schwaber (2011) [56] que o *Scrum* se fundamente nas teorias empíricas de controle de processo,

onde este empirismo afirma que o conhecimento da tomada de decisões se dá no que é conhecido e na experiência. O *Scrum* utiliza uma abordagem incremental e iterativa para o controle de riscos e o aperfeiçoamento da previsibilidade. A implementação de controle de processo empírico se apoia em três pilares que são:

2. Transparência – aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados. Esta transparência requer aspectos definidos por um padrão comum para que os observadores compartilhem um mesmo entendimento do que está sendo visto. Exemplo: uma linguagem comum referindo-se ao processo que deve ser compartilhado por todos os participantes e uma definição de "Pronto" comum;

3. Inspeção: Os usuários *Scrum* devem frequentemente inspecionar os artefatos *Scrum* e o progresso em direção ao objetivo, para detectar indesejáveis variações. Esta inspeção, não deve, no entanto, ser tão frequente que atrapalhe a própria execução das tarefas. As inspeções são mais benéficas quando realizadas de forma diligente por inspetores especializados no trabalho a se verificar;

4. Adaptação: Se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o produto resultado será inaceitável, o processo ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

A Figura 2.16 mostra de modo resumido e de forma ilustrativa uma visão geral de como ocorre o processo *Scrum*.

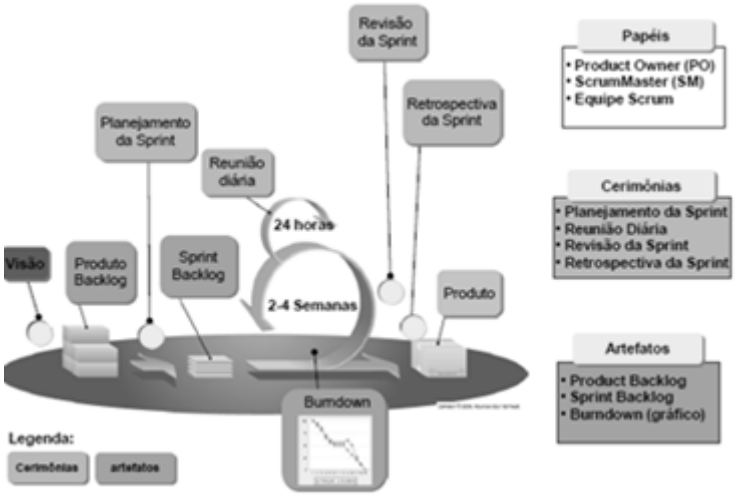

Figura 2.16: Processo *Scrum*, Fonte: (COHN, 2011) [28]

Segundo Pressman (2011) [50] o *Scrum* destaca a utilização de um conjunto de padrões de processo de *software* que tem efetividade comprovada para requisitos mutáveis,

criticalidade de negócio e projetos com prazos apertados onde se define em cada padrão de processo um conjunto de tarefas de desenvolvimento e permite que a equipe *Scrum* construa um processo que se ajuste às necessidades do projeto. Os próximos itens irão detalhar os elementos do *Scrum*.

2.2.2.2.3.2 *Time SCRUM*

Como descrito anteriormente, o *Time Scrum* é formado pelo *Scrum Master*, o *Product Owner* e a equipe de desenvolvimento e cada um exerce uma atividade proposta por cada um destes papéis previstas pelo *Scrum*.

Pressman (2011) [50] afirma que, em um mundo onde é impossível eliminar a incerteza, os padrões de processo *Scrum* possibilitam o trabalho de uma equipe de desenvolvimento de forma bem-sucedida.

> "*Times Scrum* são auto-organizáveis e multifuncionais. Equipes auto-organizáveis escolhem qual a melhor forma para completarem seu trabalho, em vez de serem dirigidos por outros de fora da equipe. Equipes multifuncionais possuem todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe. O modelo de equipe no *Scrum* é projetado para aperfeiçoar a flexibilidade, criatividade e produtividade". (SCHWABER e SUTHERLAND, 2011) [56]

Ainda segundo Schwaber (2011) [56], os produtos são entregues pelos *Times Scrum* de forma incremental e iterativa, buscando a maximização das oportunidades de realimentação. Garante-se uma versão potencialmente funcional do produto através de entregas incrementais de produto "Pronto" tornando-o um produto de trabalho sempre disponível.

2.2.2.2.3.2.1 *Scrum Master*

Dentre os papéis propostos pelo *Scrum*, o *Scrum Master* é, segundo Schwaber (2011) [56] um servo-líder para o *Time Scrum* e responsável pela garantia de que o *Scrum* seja aplicado e entendido para que o *Time Scrum* possa aderir às práticas, teoria e regras do *Scrum*.

De acordo com Pressman (2011) [50], o *Scrum Master* é aquele que lidera as reuniões e avalia de cada pessoa as suas respectivas respostas, e estas reuniões auxiliam a equipe a descobrir tão cedo quanto possível os potenciais problemas. Conforme Phan (2011)[48], o *Scrum Master* deveria possuir, antes de tudo, sete qualidades:

1. Conhecimento aprofundado, na teoria e na prática, sobre o *Scrum* na vida real;
2. Excelente habilidade de líder-servidor;

3. Fortes habilidades organizacionais;
4. Ótimas habilidades de comunicação;
5. Ótimas habilidades de apresentação;
6. Habilidades de resolução de conflitos; e
7. Habilidades excelentes de desenvolvimento humano.

Acrescenta-se também o que Schwaber (2011) [56] diz a respeito do *Scrum Master*, onde este auxilia no entendimento de quais as iterações com o *Time Scrum* que são úteis e as que não são para os que não fazem parte dele. Ele também auxilia a todos na mudança destas iterações para a maximização do valor criado pelo *Time Scrum*.

A Tabela 2.3 apresenta as várias maneiras que o *Scrum Master* serve a equipe de desenvolvimento, o *Product Owner* e a organização:

Tabela 2.3: Atividades do *Scrum Master*, Fonte: (SCHWABER e SUTHERLAND, 2011) [56]

Scrum Master	Maneiras de servir
Product Owner	– Encontrando técnicas para o gerenciamento efetivo do *Backlog* do Produto; – Claramente comunicar a visão, objetivo e itens do *Backlog* do Produto para a Equipe de Desenvolvimento; – Ensinar o *Time Scrum* a criar itens de *Backlog* do Produto de forma clara e concisa; – Compreender a longo-prazo o planejamento do Produto no ambiente empírico; – Compreender e praticar a agilidade; – Facilitar os eventos *Scrum* conforme exigidos ou necessários.
Equipe de desenvolvimento	– Treinar a Equipe de Desenvolvimento em autogerenciamento e interdisciplinaridade; – Ensinar e liderar a Equipe de Desenvolvimento na criação de produtos de alto valor; – Remover impedimentos para o progresso da Equipe de Desenvolvimento; – Facilitar os eventos *Scrum* conforme exigidos ou necessários: – Treinar a Equipe de Desenvolvimento em ambientes organizacionais nos quais o *Scrum* não é totalmente adotado e compreendido.
Organização	– Liderando e treinando a organização na adoção do *Scrum*; – Planejando implementações *Scrum* dentro da organização; – Ajudando funcionários e partes interessadas a compreender e tornar aplicável o *Scrum* e o desenvolvimento de produto empírico; – Causando mudanças que aumentam a produtividade do Time *Scrum*; – Trabalhando com outro *Scrum Master* para aumentar a eficácia da aplicação do *Scrum* nas organizações.

De acordo com Phan (2011) [48], o *Scrum Master* vem desempenhando um dos papéis mais importantes em um projeto *Scrum*, é um alguém que não auxiliará somente a remoção de impedimentos, mas ajudará também na resolução de conflitos e na transformação da equipe em uma equipe de alto desempenho, tendo pelo menos as sete qualidades.

2.2.2.2.3.2.2 *Product Owner*

Segundo Schwaber e Shutherland (2011) [56], o *Product Owner* (PO), ou dono do produto, se responsabiliza pela maximização do trabalho da equipe de desenvolvimento e do valor do produto do trabalho. O modo como isso é feito pode variar através dos *Times Scrum*, organizações e indivíduos.

É importante que se tenha um excelente *Prodcut Owner*, pois como afirma Phan (2011) [48], em um projeto ágil há vários elementos necessários, mas que este não pode ser bem-sucedido sem se ter um excelente *Product Owner*, o protetor do objetivo e da visão do produto, pois, um projeto *Scrum* ou Ágil, se foca na entrega de resultados, de valores e negócios.

De acordo com Schwaber e Shutherland (2011) [56], o *Product Owner* é o único responsável para o gerenciamento do *Backlog* do Produto que inclui:

1. Expressar claramente os itens de *Backlog* do Produto;
2. Ordenar os itens de *Backlog* do Produto para alcançar melhor as metas e missões;
3. Garantir o valor do trabalho realizado pelo Time de Desenvolvimento;
4. Garantir que o *Backlog* do Produto seja visível, transparente, claro para todos, e mostrar o que o *Time Scrum* vai trabalhar a seguir;
5. Garantir que a Equipe de Desenvolvimento entenda os itens do *Backlog* de Produto no nível necessário.

Ainda conforme Schwaber (2011) [56], o *Product Owner* pode delegar para alguém da Equipe de Desenvolvimento a realização do trabalho acima, mas que não deixa de ser responsabilidade do *Product Owner* a realização deles. De acordo com Phan (2011) [48], o *Product Owner* possui também sete qualidades que são listadas abaixo:

1. Saber como gerenciar as expectativas e prioridades das partes interessadas;
2. Ter uma visão clara, e conhecimento do produto;
3. Estar completamente disponível para se engajar ativamente com a equipe;
4. Ser um bom organizador;
5. Saber como transformar a visão de produto em um bom *Backlog* de Produto;
6. Ser um bom comunicador; e
7. Ser um bom lider-servidor.

Mas, além das atividades e qualidades do *Product Owner*, é importante deixar claro que o *Product Owner* não é um comitê e sim uma pessoa, que pode representar um comitê no *Backlog* de Produto, mas se quiserem alguma alteração nas prioridades dos itens de *Backlog* deverão convencer o *Product Owner*.

> "Para que o *Product Owner* tenha sucesso, toda a organização deve respeitar as suas decisões. As decisões do *Product Owner* são visíveis no conteúdo e na priorização do *Backlog* do Produto. Ninguém tem permissão para falar com a Equipe de Desenvolvimento sobre diferentes configurações de prioridade, e o Time de Desenvolvimento não tem permissão para agir sobre o que outras pessoas disseram". (SCHWABER e SUTHERLAND, 2011, p. 15) [56]

2.2.2.2.3.2.3 Equipe de Desenvolvimento

A equipe de desenvolvimento, ou simplesmente equipe, é responsável pela construção do produto de software para a liberação de uma versão a ser entregue. A Equipe de Desenvolvimento é "responsável por todas as atividades de desenvolvimento de *software* dentro de uma equipe *Scrum*, indo da redação de requisitos até o projeto, codificação e teste." (PHAN, 2011) [48]

"Um conceito de metodologia Ágil e do *Scrum*, que dita que a equipe de desenvolvimento dever ser responsável por organizar a si mesma, da maneira que os membros julgarem mais apropriado, para realizar o trabalho. Isso não significa que caso a equipe não consiga entregar, a auto-organização poderá ou deverá impedir a intervenção do *Scrum Master* ou *Product Owner*, de maneira sutil e indireta, para ajudar a equipe a seguir em frente". (PHAN, 2011, p. 35) [48]

De acordo com Kniberg (2007) [38] é preciso criar um lugar para a equipe caso não exista nenhum e que não há alternativa eficiente quando se constrói equipes *Scrum* a não ser mantê-las juntas. Neste caso, significa:

1. Audibilidade: Qualquer um da equipe pode conversar com o outro sem gritar ou sair da mesa;
2. Visibilidade: Todos da equipe podem ver todos os demais. Cada um da equipe consegue ver o quadro de tarefas. Não necessariamente perto o suficiente para lê-lo, mas pelo menos para vê-lo;
3. Isolamento: Se toda a equipe de repente levantar e se envolver em uma espontânea e animada discussão sobre implementação, não haverá ninguém de fora da equipe perto o suficiente para ser perturbado. E vice-versa.

Segundo Schwaber (2011) [56], a organização autoriza e estrutura as Equipes de Desenvolvimento para gerenciar e organizar seu próprio trabalho. A resultante sinergia aperfeiçoa a eficácia e eficiência da Equipe de Desenvolvimento como um todo. As Equipes de Desenvolvimento têm as seguintes características:

1. Elas são auto-organizadas. Ninguém (nem mesmo o *Scrum Master*) diz a equipe de Desenvolvimento como transformar o *Backlog* do Produto em incrementos de funcionalidades potencialmente utilizáveis, ou seja, a equipe se auto gerência;

2. Equipes de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto;
3. O *Scrum* não reconhece títulos para os integrantes da Equipe de Desenvolvimento que não seja o Desenvolvedor, independentemente do trabalho que está sendo realizado pela pessoa; não há exceções para esta regra;
4. Individualmente os integrantes da Equipe de Desenvolvimento podem ter habilidades especializadas na área de especialização, mas a responsabilidade pertence à Equipe de Desenvolvimento como um todo;
5. Equipes de Desenvolvimento não contém sub-equipes dedicadas a domínios específicos de conhecimento, tais como teste ou análise de negócios.

Em relação ao tamanho da equipe, Schwaber (2011) [56] apresenta que o ideal é que seja pequena o suficiente para que se mantenha e grande o suficiente para completar uma significativa parcela do trabalho e que se mantenha ágil.

> "Menos de três integrantes na Equipe de Desenvolvimento diminuem a interação e resultam em um menor ganho de produtividade. Equipes de desenvolvimento menores podem encontrar restrições de habilidades durante a *Sprint*, gerando uma Equipe de Desenvolvimento incapaz de entregar um incremento potencialmente utilizável. Havendo mais de nove integrantes é exigida muita coordenação. Equipes de Desenvolvimento grandes geram muita complexidade para um processo empírico gerenciar. Os papéis de *Product Owner* e de *Scrum Master* não são incluídos nesta contagem, ao menos que eles também executem o trabalho do *Backlog* de *Sprint*." (SCHWABER e SUTHERLAND, 2011, p. 30) [56]

Assim sendo, conforme Phan (2011) [48] uma equipe inicia-se quando um grupo de pessoas se reúne para o alcance de uma meta comum. Dependerá da capacidade da equipe trabalhar em conjunto como uma equipe auto-organizada o fato de a mesma ter desempenho baixo ou alto, além, também, da dependência da liderança do ambiente tradicional de controle e comando.

E em complemento, segundo Schwaber (2011) [56], a Equipe de Desenvolvimento consiste na realização de trabalho por profissionais em que, entrega uma versão utilizável que potencialmente incrementa um produto "Pronto" quando se finaliza uma *Sprint*, onde somente se criam incrementos os integrantes da Equipe de Desenvolvimento.

2.2.2.2.3.2.3.1 Definindo o “Pronto”

Antes de se falar sobre os eventos e artefatos propostos pelo *Scrum*, é importante para o mesmo que seja definida a situação de “Pronto” que é um fator relevante para saber quando se finaliza uma *Sprint* (que é tratada mais adiante) para se que possa apresentar um produto potencialmente entregável.

Conforme Phan (2011) [48], ainda que uma equipe trabalhe por conta própria, é importante que se defina o estado de “Pronto”, pois as atividades dependerão desta definição. Quando não há uma definição de “Pronto”, no caso de se estar trabalhando com duas equipes, por exemplo, isto se torna um problema.

> “Quando o item do *Backlog* do Produto ou um incremento é descrito como “Pronto”, todos devem entender o que o “‘Pronto” significa. Embora isso varie significativamente de um extremo ao outro para cada *Time Scrum*, os integrantes devem ter um entendimento compartilhado do que significa o trabalho estar completo, assegurando a transparência. Esta é a “Definição de Pronto” para o *Time Scrum* e é usado para assegurar quando o trabalho está completado no incremento do produto”. (SCHWABER e SUTHERLAND, 2011, p. 32) [56]

De acordo com Kniberg (2007) [38], é importante que haja uma definição clara de “Pronto” em que a equipe de desenvolvimento e o *Product Owner* concordem. Para se definir o “Pronto” no processo Scrum é geralmente melhor quando feita com o senso comum.

Para Phan (2011)[48] a definição de “Pronto” é uma concordância sobre o que deveria ser o estado finalizado ao final da *Sprint* para todas as partes da equipe *Scrum*.

2.2.2.2.3.3 Artefatos do *SCRUM*

Conforme Schwaber (2011) [56], os artefatos definidos para o *Scrum* são especificamente projetados com o intuito de maximizar a transparência das informações chaves que são necessárias para que o *Time Scrum* se assegure no sucesso para entrega do incremento “Pronto”.

Os artefatos do *Scrum* representam o trabalho ou o valor das várias maneiras que são úteis no fornecimento de transparência e oportunidades para inspeção e adaptação (SCHWABER e SUTHERLAND, 2011) [56].

Os artefatos propostos pelo *Scrum* são: o *Backlog* do Produto e o *Backlog* da *Sprint* que são tratados nos itens seguintes.

2.2.2.2.3.3.1 *Backlog* do Produto

O *Product Backlog* (ou *Backlog* do Produto) é uma origem única para qualquer mudança dos requisitos a ser feita no produto onde se lista de forma ordenada tudo que deve ser necessário no produto (SCHWABER e SUTHERLAND, 2011) [56].

Para uma definição breve sobre o que é *Produtc Backlog*, Phan (2011)[48] diz que é uma lista priorizada de requisitos que se pode incluir tudo, desde os aspectos de negócios até aspectos de tecnologia, correção de bugs e questões técnicas.

Acrescenta-se ainda, conforme Schwaber (2011) [56] que a listagem encontrada no *Product Backlog* possui os atributos da estimativa, descrição e ordem e contém todas as melhorias, correções, funções, características e requisitos que compõem as mudanças a serem realizadas no produto das versões futuras.

Kniberg (2007) [38] afirma que é no *Product Backlog* que tudo se inicia, e ele é basicamente uma lista de estórias, coisas que o cliente deseja, requisitos que são descritos utilizando a terminologia do cliente.

> "Um *Backlog* de Produto nunca está completo. Os primeiros desenvolvimentos apenas estabelecem os requisitos inicialmente conhecidos e melhor entendidos. O *Backlog* do Produto evolui tanto quanto o produto e o ambiente no qual ele será utilizado evoluem. O *Backlog* do Produto é dinâmico; mudando constantemente para identificar o que o produto necessita para ser mais apropriado, competitivo e útil. O *Backlog* do Produto existirá enquanto o produto também existir". (SCHWABER e SUTHERLAND, 2011, p. 32) [56]

Schwaber (2011) [56] afirma que as técnicas de estimativas para se prever o progresso são várias e elas têm provado ser bem úteis. Todavia, a importância do empirismo não é substituída.

Como exemplo de técnicas de acompanhamento/estimativas de progresso, apresenta-se:

1. Gráfico de Burndown (Burndown Chart): diagrama usado pela equipe de *Scrum* para mostrar quanto trabalho falta na *Sprint*;
2. kanban (iniciado em letra minúscula): denota uma técnica de cartões derivada da estrutura mais ampla do Kanban, usada para limitar o trabalho em processo, aumentando assim, o fluxo de trabalho, ver Figura 2.17 (PHAN, 2011) [48].

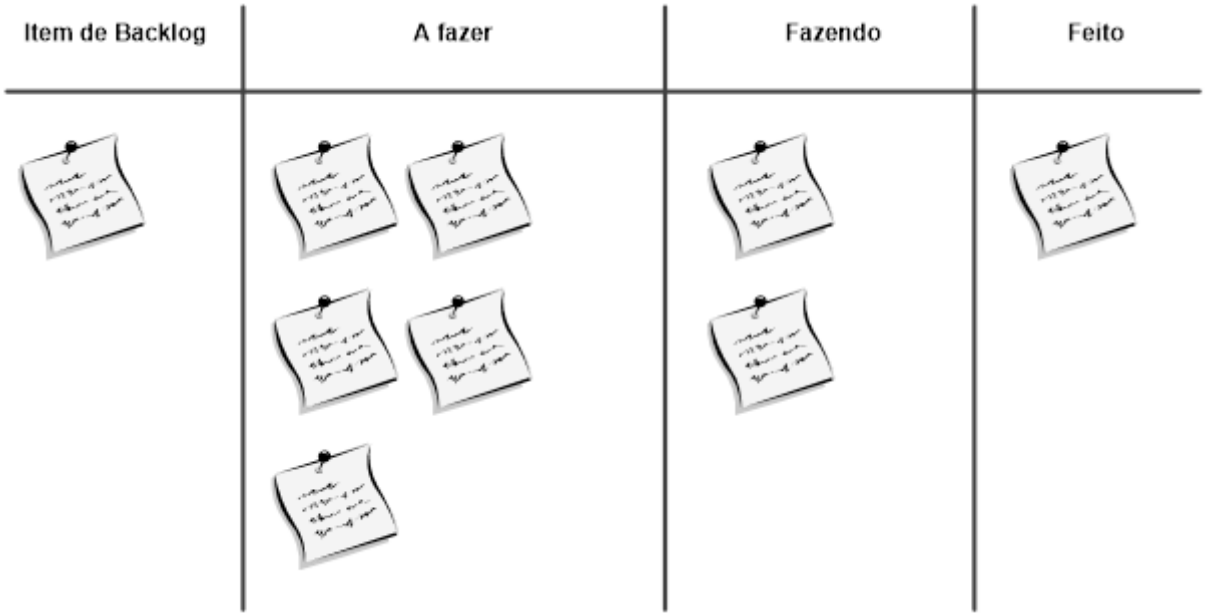

Figura 2.17: Quadro Kanban, Fonte: (COHN, 2011) [28], adaptado pelo autor

E por fim, ainda conforme Schwaber (2011) [56], geralmente, o *Product Backlog* é ordenado por risco, necessidade, valor e prioridade onde os itens que estão no topo da lista ordenada do *Product Backlog* determinam as atividades mais imediatas de desenvolvimento. Quanto maior a ordem de um item, mais ele deve ser considerado e também mais consenso existirá em relação a ele e seu valor.

2.2.2.2.3.3.2 *Backlog* da *Sprint*

Além do *Backlog* do Produto, há também o artefato do *Scrum* que é o *Backlog* da *Sprint* (ou *Sprint Backlog*) que, segundo Phan (2011) [48], é uma lista de todas as tarefas a serem realizadas durante uma *Sprint* pela equipe de desenvolvimento.

De forma mais abrangente, Schwaber (2011) [56] diz que o *Backlog* do Produto é formado pelos itens retirados do *Backlog* do Produto que foram selecionados e reunidos em conjunto para a *Sprint*, mais o plano de entrega do incremento do produto e o alcance do objetivo da *Sprint*. É a previsão do trabalho necessário para que se entregue a funcionalidade e da equipe de desenvolvimento em relação à funcionalidade que se encontrará no próximo incremento. De acordo com Kniberg (2007) [38], o *Sprint Backlog* é criado pelo *Scrum Master* após a Reunião do Planejamento de *Sprint* e antes da Reunião Diária do *Scrum*.

> "O *Backlog* da *Sprint* é um plano com detalhes suficientes que as mudanças em progresso sejam atendidas durante a Reunião Diária. A Equipe de Desenvolvimento modifica o *Backlog* da *Sprint* ao longo de toda a *Sprint*, e o *Backlog* da *Sprint* vai surgindo durante a *Sprint*. Este surgimento ocorre quando a Equipe de Desenvolvimento trabalha segundo o plano e aprende mais sobre o trabalho necessário para alcançar o objetivo da *Sprint*." (SCHWABER e SUTHERLAND, 2011, p. 35) [56]

E ainda afirma Schwaber (2011) [56] que não se considera pelo *Scrum* o tempo que se gasta no trabalho dos itens do *Backlog* da *Sprint*, mas as únicas variáveis que interessam são o trabalho restante e a data.

2.2.2.2.3.4 Eventos *SCRUM*

De acordo com Schwaber (2011) [56] no *Scrum* são utilizados os eventos prescritos para minimizar a necessidade de reuniões não definidas e para a criação de rotina. O *Scrum* utiliza-se de eventos *time-boxed*, que significa que todo evento tem uma duração máxima, garantindo uma quantidade de tempo adequada a ser gasta no planejamento, sem permitir que haja perdas no processo de planejamento.

A *Sprint* (que é para os outros eventos um container) e cada um dos outros eventos do *Scrum* se tornam uma oportunidade de adaptar e inspecionar alguma coisa e estes, para permitir uma transparência e inspeção criteriosa, são especificamente projetados (SCHWABER e SUTHERLAND, 2011) [56].

Assim sendo, os eventos compõem uma parte importante do processo *Scrum* e é composto basicamente por *Sprints*, Reunião de Planejamento da *Sprint*, Reunião Diária, Revisão de *Sprint* e Retrospectiva de *Sprint*. "A não inclusão de qualquer um dos eventos resultará na redução da transparência e da perda de oportunidade para inspecionar e adaptar." (SCHWABER e SUTHERLAND, 2011) [56]

2.2.2.2.3.4.1 *Sprint*

A *Sprint* é um dos eventos previstos pelo *Scrum* e, segundo Phan (2011) [48] consiste em uma iteração, ou um ciclo fixo, em que a equipe de desenvolvimento deve transformar os requisitos escolhidos (histórias de usuário) em um incremento do produto potencialmente a ser entregue.

Conforme Schwaber (2011) [56], a *Sprint* é o coração do *Scrum* que é um *time-boxed* de um mês, ou menos, e que é criado um "Pronto" (versão incremental potencialmente utilizável do produto). Elas têm, em todo o esforço de desenvolvimento, durações coerentes. Começa-se uma *Sprint* nova logo após o término da anterior.

> [...] "Cada Sprint tem a definição do que é para ser construído, um plano projetado e flexível que irá guiar a construção, o trabalho e o resultado do produto. *Sprints* são limitadas a um mês corrido. Quando o horizonte da Sprint é muito longo, a definição do que será construído pode mudar, a complexidade pode aumentar e o risco pode crescer. *Sprints* permitem previsibilidade que garante a inspeção e adaptação do progresso em direção a meta pelo menos a cada mês corrido. *Sprints* também limitam o risco ao custo de um mês corrido". (SCHWABER e SUTHERLAND, 2011, p. 36) [56]

De acordo com Pressman (2011) [50], as *Sprints* são unidades de trabalho necessárias para a satisfação de um requisito definido que precisa se cumprir num intervalo predefinido de tempo (normalmente de 30 dias) e que durante a *Sprint* os itens em pendência a que se destinam, as unidades de trabalho são congelados – isto significa que não se introduz nenhuma modificação durante a *Sprint* – proporcionando um ambiente de trabalho em curto prazo aos membros da equipe, mas, possibilitando estabilidade.

O propósito de cada *Sprint*, segundo Schwaber (2011) [56], é a entrega de incrementos de funcionalidades potencialmente utilizáveis e que estão de acordo com a atual definição de "Pronto".

Enquanto se estiver em uma *Sprint*, faz-se necessário atentar-se aos seguintes apontamentos:

1. Não são feitas mudanças que podem afetar o objetivo da *Sprint*;
2. A composição da Equipe de Desenvolvimento permanece constante;
3. As metas de qualidade não diminuem;
4. O escopo pode ser clarificado e renegociado entre o *Product Owner* e a Equipe de desenvolvimento quanto mais for aprendido.

A *Sprint* tem um objetivo (ou meta) a ser alcançado e Phan (2011) [48] afirma que este objetivo é oferecido pelo *Product Owner* à Equipe de Desenvolvimento para aquela *Sprint*.

O objetivo da *Sprint* deve responder a pergunta fundamental "Por que nós estamos fazendo esta *Sprint* ?[..]". De fato, uma maneira de extrair o objetivo de uma *Sprint* do *Product Owner* é literalmente fazer esta pergunta." (KNIBERG, 2007) [38]

Afirma ainda Schwaber (2011) [56], que o objetivo da *Sprint* pode ser um marco que se encontra no propósito maior do roteiro do produto e oferece alguma flexibilidade à Equipe de Desenvolvimento em relação às funcionalidades a serem implementadas na *Sprint*.

Há também a possibilidade do cancelamento de uma *Sprint* que pode ocorrer, segundo Schwaber (2011) [56], antes de terminar o *time-box* da *Sprint*, e esta só pode ser feita pelo *Product Owner*, mesmo que este esteja sob influência de partes interessadas. Os motivos

para o cancelamento pode ser quando o produto a ser entregue na *Sprint* se torne obsoleto, se a organização mudar de direção ou se as tecnologias mudarem, entre outros. Ao final do cancelamento da *Sprint* pode ser revisado qualquer item do *Product Backlog* (*Backlog* de Produto) completado e "Pronto" e todos os itens que estiverem incompletos são re-estimados e colocados de volta ao *Backlog* de Produto. Normalmente o P*roduct Owner* aceita parte do trabalho que esteja potencialmente utilizável.

Ainda afirma Schwaber (2011) [56] que o cancelamento de *Sprints* ocasiona o consumo de recursos pelo fato de que todos terão que se reagrupar para iniciar outra *Sprint* em outra reunião de planejamento de *Sprint*.

Por fim, "as *Sprints* são compostas por uma reunião de planejamento da *Sprint*, reuniões diárias, o trabalho de desenvolvimento e a retrospectiva da *Sprint* anterior" (SCHWABER e SUTHERLAND, 2011) [56] e que serão tratadas cada uma a seguir, pois são eventos previstos pelo *Scrum*, exceto pelo trabalho de desenvolvimento que é o próprio processo de construção, a codificação que fica por conta da equipe de desenvolvimento.

2.2.2.2.3.5 Incremento

O incremento, conforme Schwaber (2011) [56], são todos os itens de *Backlog* de Produto completados somados durante a *Sprint* com tudo das *Sprints* anteriores.

A abordagem que Phan (2011)[48] traz é sobre o incremento do produto potencialmente entregável que é quando durante uma *Sprint* um incremento de produto de *software* funcional foi desenvolvido e que eventualmente pode ser entregue aos usuários.

Ainda afirma Schwaber (2011) [56] que um novo incremento deve estar "Pronto" ao final da *Sprint*, o que quer dizer que para atender a definição de "Pronto" do *Time Scrum* este deve estar na condição utilizável, independente do *Product Owner* decidir em relação de liberá-la ou não.

2.2.2.2.3.6 Reunião de Planejamento da *Sprint*

Como parte do processo *Scrum*, a Reunião de Planejamento da *Sprint* é uma *time-boxed* com duração de oito horas para uma *Sprint* de um mês e que para *Sprints* menores, por exemplo, deve ser proporcionalmente menor. Se a *Sprint* for de duas semanas, então a Reunião de Planejamento da *Sprint* será de quatro horas (SCHWABER e SUTHERLAND, 2011) [56].

Em resumo, Phan (2011) [48] diz que a Reunião de Planejamento da *Sprint* se realiza em duas fases, onde na primeira o *Product Owner* apresenta os requisitos que ele acredita

fazer parte da *Sprint* à equipe e na segunda a equipe decide a forma como os requisitos selecionados irão se transformar em um incremento de produto potencialmente entregável.

Phan (2011) [48] e Schwaber (2011)[56] complementam dizendo que as partes (ou fases) da reunião de Planejamento da *Sprint* procura responder respectivamente as seguintes perguntas:

1. O que será entregue como resultado do incremento da próxima *Sprint*?
2. Como o trabalho necessário para entregar o incremento será realizado?

2.2.2.2.3.7 Primeira Parte (ou Fase)

Segundo Phan (2011) [48] nesta fase o *Product Owner* percorre os requisitos – como história de usuário - para decidir juntamente com o *feedback* da equipe quais deveriam fazer parte de quais *Sprints* e quais são os seus objetivos.

De acordo com Schwaber (2011) [56], nesta parte trabalha-se para prever as funcionalidades que serão desenvolvidas durante a *Sprint* pela Equipe de Desenvolvimento e apresenta-se a mesma, pelo *Product Owner*, os itens de *Backlog* do Produto de forma ordenada. Todo o *Time Scrum* também busca colaborar com o entendimento do trabalho da *Sprint*.

2.2.2.2.3.8 Segunda Parte (ou Fase)

Após a seleção do trabalho da *Sprint*, é decidido pela equipe como se construirá as funcionalidades durante a *Sprint* e serão transformadas em um incremento de produto "Pronto". Selecionam-se para a *Sprint* os itens do *Product Backlog* juntamente com o plano de entrega destes itens e assim dão origem justamente ao *Sprint Backlog* citado anteriormente (SCHWABER e SUTHERLAND, 2011) [56].

Conforme complementa Phan (2011) [48], nesta fase, a equipe irá tentar identificar as tarefas partindo das histórias escolhidas previamente e deduzir em quanto tempo (em horas) a equipe despenderá para a transformação dessas tarefas em incrementos de produto potencialmente entregável. Todas as tarefas de desenvolvimento que são parte da *Sprint* comumente são registradas em um Quadro de Tarefas – algum tipo de quadro branco em uma parede – para facilitar o rastreamento e a alocação por parte da equipe, a menos que seja utilizado pela equipe um *software* de planejamento.

O propósito da Reunião do Planejamento da *Sprint*, de acordo com Kniberg (2007) [38], é oferecer informação suficiente para a equipe trabalhar por algumas semanas e dar confiança ao *Product Owner* quanto ao cumprimento das atividades previstas na *Sprint*, como resultado concreto:

1. Um objetivo de *Sprint*;
2. Uma lista de membros da equipe (e seus níveis de comprometimento);
3. Um *Sprint Backlog* (ou seja, uma lista de estórias inclusas na *Sprint*);
4. Uma data definida para a apresentação da *Sprint*;
5. Data e local definidos para as reuniões Diárias.

Há também uma abordagem sobre o Planejamento de Releases em que Phan (2011) [48] diz que é um planejamento opcional nos primórdios do *Scrum* em que seu objetivo-chave é que se permita a equipe de *Scrum* à identificação de todos os lançamentos que o produto de *software* deve ter juntamente com uma agenda de entregas prováveis e com duração de quatro horas para uma *Sprint* de quatro semanas.

Ao final do evento de Reunião do Planejamento da *Sprint*, segundo Schwaber (2011) [56], a Equipe de Desenvolvimento terá que conseguir explicar ao *Product Owner* e ao *Scrum Master* como se pretende trabalhar como equipe auto-organizada para criar um incremento previsto e cumprir a meta da *Sprint*.

2.2.2.2.3.9 Reunião Diária

A Reunião Diária (ou Daily Scrum) é um evento do *Scrum* que começa após o inicio da *Sprint* atual. E como afirma Phan (2011) [48], as reuniões diárias começam quando se terminam os planejamentos necessários e quando se inicia o trabalho propriamente dito da *Sprint* e tem uma duração de 15 minutos.

Segundo Schwaber (2011) [56], a Reunião Diária consiste em um *time-boxed* de 15 minutos ao qual a Equipe de desenvolvimento tenha a possibilidade de sincronizar as atividades e criar um plano a ser realizado nas próximas 24 horas. Ela se realiza para que seja feita uma inspeção do trabalho desde a última Reunião Diária e para prever o trabalho que deverá se realizar antes até a próxima reunião. Ela ocorre e se mantém sempre no mesmo local e horário todos os dias para que seja reduzida a complexidade.

De acordo Phan (2011) [48], o Daily *Scrum* é uma reunião diária de progresso onde se reúnem os membros da equipe, virtualmente ou fisicamente, para a sincronização e aprendizado sobre o quanto se progrediu em direção à meta do *Sprint*.

São propostas três perguntas básicas em que, "durante a reunião cada integrante da Equipe de Desenvolvimento esclarece" e elas são: (SCHWABER e SUTHERLAND, 2011) [56]

1. O que foi completado desde a última reunião?
2. O que será feito até a próxima reunião?
3. Quais os obstáculos que estão no caminho?

A *Daily Scrum*, segundo Schwaber (2011) [56] serve também para que a Equipe de Desenvolvimento possa avaliar o progresso em direção à meta da *Sprint* e para avaliar se o progresso está tendendo a se completar, segundo as atividades previstas no *Product Backlog*.

As consequências das Reuniões Diárias são:

1. Melhoram a comunicação;
2. Eliminam outras reuniões;
3. Identificam e removem impedimentos para o desenvolvimento;
4. Destacam e promovem rápidas tomadas de decisão;
5. Melhoram o nível de conhecimento da Equipe de Desenvolvimento.

Observa-se, então, que a utilização das Reuniões Diárias proporciona muitos benefícios que auxiliam no desenvolvimento do processo *Scrum*, além de um controle constante das atividades realizadas, o que foi produzido e do que ainda falta, mantendo-se sempre uma atualização em relação ao andamento da *Sprint*.

2.2.2.2.3.10 Revisão da *Sprint*

A Revisão da *Sprint*, de acordo com Phan (2011) [48], é uma reunião em que a equipe disponibiliza uma demonstração do que ela construiu para as partes interessadas e para o *Product Owner*.

Segundo Schwaber (2011) [56], executa-se a Revisão da *Sprint* ao final da *Sprint* para que o incremento seja inspecionado e para a adaptação do *Backlog* do Produto se acaso for preciso. Durante esta reunião, as partes interessadas e o *Time Scrum* apresentam tudo o que foi feito na *Sprint* e, com base nisso e em qualquer modificação no *Product Backlog* durante a *Sprint*, os participantes auxiliam nas próximas coisas que necessitam estar prontas. É uma reunião informal e que apresenta o incremento com o intuito de motivar e obter comentários e promover a ajuda mútua.

Para a reunião de Revisão de *Sprint*, apontam-se os seguintes elementos:

1. O *Product Owner* identifica o que foi “Pronto” e o que não foi “Pronto”;
2. A Equipe de Desenvolvimento discute o que foi bem durante a *Sprint*, quais problemas ocorreram dentro da *Sprint*, e como estes problemas foram resolvidos;
3. A Equipe de Desenvolvimento demonstra o trabalho que está ”Pronto” e responde as questões sobre o incremento;
4. O *Product Owner* discute o *Backlog* do Produto tal como está. Ele (ou ela) projeta as prováveis datas de conclusão baseado no progresso até a data;
5. O grupo todo colabora sobre o que fazer a seguir, e é assim que a Reunião de Revisão da *Sprint* fornece valiosas entradas para a Reunião de Planejamento da próxima *Sprint*.

Aponta-se também um aspecto importante para a realização da Revisão da *Sprint* segundo Kniberg (2007) [38] onde ele afirma que as pessoas tendem a subestimar, mas que é uma parte importante, pois uma apresentação de *Sprint* (ou revisão da *Sprint*) bem feita causa um impacto profundo:

1. A equipe ganha crédito por suas realizações. Eles se sentem bem;
2. Outros aprendem o que sua equipe está fazendo;
3. A apresentação atrai *feedback* vital dos *stakeholder*;
4. Apresentações são (ou deveriam ser) um evento social onde equipes diferentes podem interagir umas com as outras e discutir seu trabalho. Isso tem muito valor;
5. Fazer uma apresentação força a equipe a realmente “terminar as coisas” e liberá-las (mesmo que seja apenas em um ambiente de teste) [...].

Ainda de acordo com Kniberg (2007) [38], quando a equipe não se esforça para apresentar uma boa revisão de *Sprint* (sendo os mais variados motivos) esta apresenta falhas e passam até vergonha, mas que incentiva realizar um trabalho melhor da próxima vez.

O resultado da execução da Reunião de Revisão da *Sprint*, afirma Schwaber (2011) [56], consiste em um *Product Backlog* revisado ao qual se define para a próxima *Sprint* o provável Produto de *Backlog*, podendo este ser também completamente ajustado para que sejam atendidas novas oportunidades.

2.2.2.2.3.11 Retrospectiva da *Sprint*

A Retrospectiva da *Sprint* também é um evento proposto pelo *Scrum*, e que segundo Phan (2011) [48], é uma reunião onde a equipe irá percorrer o que não funcionou e o que funcionou durante a *Sprint* que acabou de ser finalizada para que seja determinada a existência de algo que se pode aprender com a experiência vivida e que torna o processo ainda melhor na *Sprint* seguinte.

De acordo com Schwaber (2011) [56], a Retrospectiva da *Sprint* se realiza antes da Reunião de Planejamento da próxima *Sprint* e depois da Revisão de *Sprint*, e é uma reunião *time-boxed* com duração de três horas para uma *Sprint* de um mês sendo proporcionalmente com duração menor quando um *Sprint* for menor que o tempo de um mês.

A Retrospectiva de *Sprint* tem como propósito:

1. Inspecionar como foi a última *Sprint* em relação às pessoas, relações, processos e ferramentas;
2. Identificar e ordenar os principais itens que foram bem e as potenciais melhorias;
3. Criar um plano para implementar melhorias no modo que o *Time Scrum* faz em seu trabalho.

Uma abordagem sobre a Retrospectiva da *Sprint* segundo Kniberg (2007) [38] diz que a coisa mais importante da mesma é assegurar-se de que ela aconteça sempre, pois esta se mostra extremamente útil. Mas por alguma razão as equipes nem sempre parecem estar propensas a realizar as retrospectivas e com isto, no decorrer do tempo, nota-se que a equipe continua a cometer repetidamente os mesmos erros.

Ao finalizar a Retrospectiva da *Sprint*, afirma Schwaber (2011) [56] que as melhorias que serão implementadas na próxima *Sprint* devem ser identificadas pelo *Time Scrum*, sendo que, a forma de se adaptar à inspeção que o mesmo faz a si próprio é a implementação destas melhorias. A Retrospectiva da *Sprint* fornece um evento focado e dedicado na adaptação e inspeção, todavia, a qualquer momento as melhorias podem ser adotadas.

2.2.2.2.4 Outros Métodos Ágeis

Além do XP, do DAS e o *SCRUM*, existem vários outros métodos ágeis que demonstram a abordagem de desenvolvimento ágil e que são citados mais alguns a seguir:

1. Crystal: de acordo com Pressman (2011) [50], o Crystal foi criado por Jim Highsmith e Alistair Cockburn com o intuito de alcançar uma abordagem de desenvolvimento de *software* que premia a "manobrabilidade" enquanto que Cockburnd caracterizava como um jogo cooperativo de invenção e comunicação de recursos limitados, com o principal objetivo de entregar software úteis funcionando e com o objetivo secundário de preparar-se para o jogo seguinte (Cockburn *apud* Pressman, (2011)) [50]. A família Crystal na verdade é um agrupamento de processos ágeis que se apresentaram efetivos para os diversos tipos de projeto, tendo como intenção permitir que as equipes ágeis a seleção do membro da família Crystal mais adequado para o projeto e ambiente.

2. *Dynamic Systems Development Method* (DSDM): em português significa "Método de Desenvolvimento Dinâmico de Sistemas", que é uma abordagem desenvolvimento de software ágil e que "fornece um arcabouço para construir e manter sistemas que satisfazem às restrições de prazo apertadas por meio do uso de prototipagem incremental em um ambiente controlado de projeto." (CS3 *apud* Pressman, (2011)) [50]

3. *Feature Driven Development* (FDD): de acordo com Pressman (2011) [50] significa em português "Desenvolvimento Guiado por Característica" e que incialmente foi concebido como um modelo prático de processo para a engenharia de software orientada a objetos por Peter Coad e seus colegas e que depois Stephen Palmer e John Felsing melhoraram e estenderam o trabalho de Coad descrevendo um processo adaptativo e ágil que pode ser utilizado para projetos de tamanho moderado e grande e tem no contexto que uma característica "é uma função valorizada pelo cliente que pode ser implementada em duas semanas ou menos." (Coad *apud* Pressman, (2011)) [50]

Existem outras abordagens ágeis de desenvolvimento de *software* dentre as que já foram citadas anteriormente, mas a metodologia *Scrum* foi detalhada e será alvo deste projeto de pesquisa, uma vez que será proposto sua implementação na organização em estudo, a fim de propiciar melhorias no processo atual, e será relatado no Capitulo 5.

No próximo capítulo será apresentado a Metodologia da Pesquisa utilizada no projeto.

Capítulo 3

Metodologia da Pesquisa

Este capítulo visa apresentar a abordagem metodológica aplicada para o desenvolvimento desta pesquisa. Será caracterizada a forma de trabalho e definida a estratégia e as etapas utilizadas para se alcançar os objetivos propostos.

3.1 Métodos de pesquisa

Segundo Marconi e Lakatos (2007[41]), pesquisa cientifica é um procedimento formal, com método de pensamento reflexivo, que requer um tratamento científico e se constitui no caminho para conhecer a realidade ou para descobrir verdades parciais.

Neste tópico será apresentado o posicionamento epistemológico da pesquisa, seguido dos métodos escolhidos de acordo com uma interpretação pessoal e real além dos objetivos propostos. O objetivo da pesquisa é compreender o contexto a ser proposto, utilizando-se de uma metodologia de observação dos métodos de desenvolvimento de *software* já aplicados pela APF. Para tal, foi desenvolvido o problema de pesquisa associado às hipóteses estabelecidas apresentados no Capítulo 1, seguido da revisão da literatura apresentada no Capítulo 2, fornecendo subsídios para realização do Estudo de Caso descrito no Capítulo 4, possibilitando assim chegar a uma conclusão no Capítulo 6.

Na Figura 3.1 apresentada a seguir, a fundamentação metodológica da pesquisa foi dividida entre Perspectiva Filosófica, Abordagem, Estratégia e Técnica da Pesquisa. O primeiro quadrante do primeiro diamante demonstra a abertura ao entendimento da abrangência do tempo e espaço que a pesquisa se encontra, a partir da perspectiva interpretativista. Ao serem estabelecidos a metodologia e o método esse universo passa a ser reduzido convergindo para abordagem metodológica definida: qualitativa documental, baseada na revisão bibliográfica. Este conjunto, portanto, converge e dá entrada no próximo diamante permitindo a utilização da técnica de Estudo de Caso e dos métodos de coleta de dados. Estes foram estruturados na triangulação entre a entrevista, a observação

direta participativa e a análise documental. Com isso, a análise conclusiva resultou na Proposta de Melhoria.

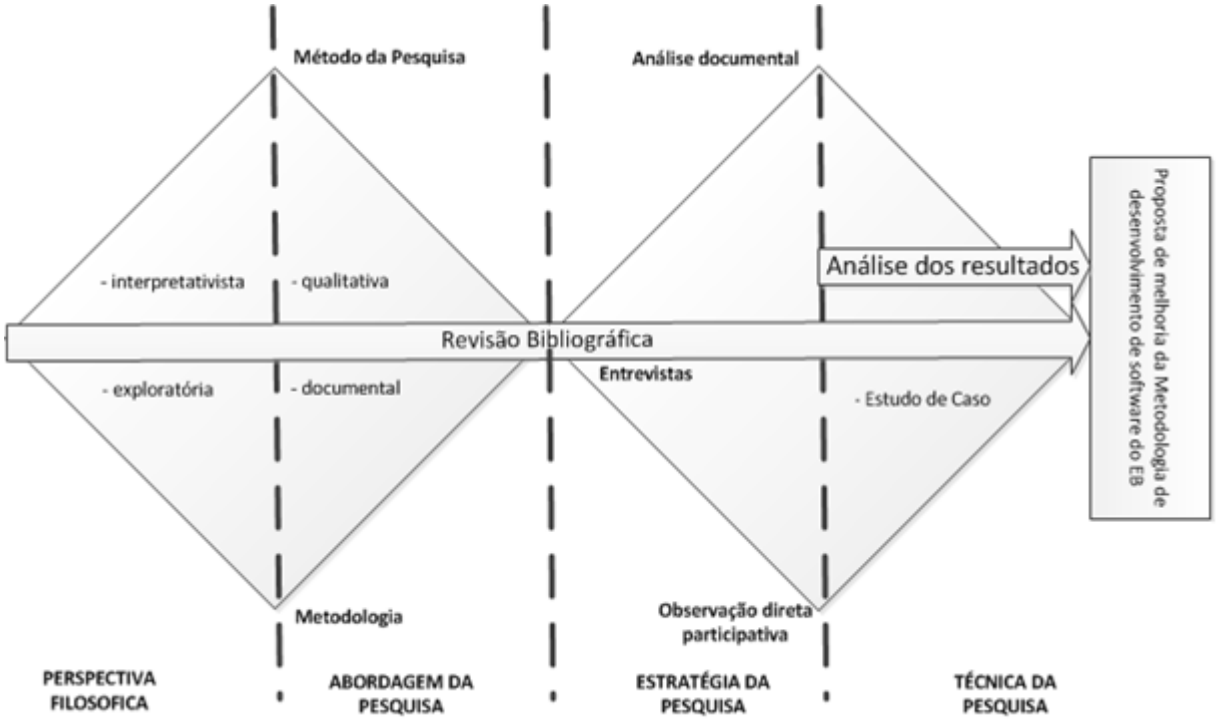

Figura 3.1: Fases da Pesquisa, Fonte: desenvolvido pelo autor

Para Girod-Séville e Perret (1999)[34] o paradigma interpretativista é um dos paradigmas epistemológicos mais identificados nas Ciências de Gestão. A escolha de cada paradigma depende dos objetivos levantados pelo pesquisador, que são fortemente influenciadas pela sua própria visão do problema, ou seja, pelo que significa a natureza da realidade para o pesquisador. Ainda na concepção das autoras, toda a pesquisa se encaixa, de fato, em determinada visão do mundo, utiliza um método e indica resultados, com o objetivo de prever, prescrever, compreender, construir ou explicar.

Esta pesquisa é fundamentada no paradigma interpretativista com o objetivo de compreender como a metodologia de desenvolvimento ágil se comporta na APF no sentido de garantir o cumprimento dos preceitos legais auditados pelos órgãos de controle. Nesse caso específico o que corresponde, aos métodos e ou processos adotados pelos órgãos visitados que possam contribuir de alguma forma na proposta de metodologia de desenvolvimento de *software* ágil s ser adotado pelo Exército Brasileiro por intermédio do Centro de Desenvolvimento de Sistemas (CDS).

Para os interpretativistas, o termo "compreender" significa fornecer interpretações aos comportamentos, e implica necessariamente na identificação de significados locais dados

atores, ou seja, significado situado (no espaço) e datados (no tempo) (GIROD-SERVILLE; PERRET, 1999, p. 24)[34].

Inicialmente para execução da pesquisa foi realizada a estruturação, conforme demonstrado na Figura 3.2, sendo executado de forma concomitante ao estudo bibliográfico de artigo, dissertações e teses, juntamente com Normas Técnicas sobre desenvolvimento de *software* e gestão risco. Em seguida ocorreram os estudos sobre as jurisprudências do Tribunal de Contas da União (TCU) especificamente nos casos relacionados aos órgãos federais que já utilizam e/ou utilizaram a metodologia ágil com fundamento dos projetos de desenvolvimento de *software*.

Entretanto, mostrou-se igualmente necessário analisar se os riscos apontados são relevantes e/ou já são submetidos de alguma forma a um processo de mitigação, sob o ponto de vista dos órgãos visitados pelo TCU. A análise dos cases estudados viabilizaram de alguma forma a compreensão metodológica mais adequada a ser aplicada em uma Administração Pública que garantisse a boa aplicação dos recursos públicos, com grau de satisfação dos usuários.

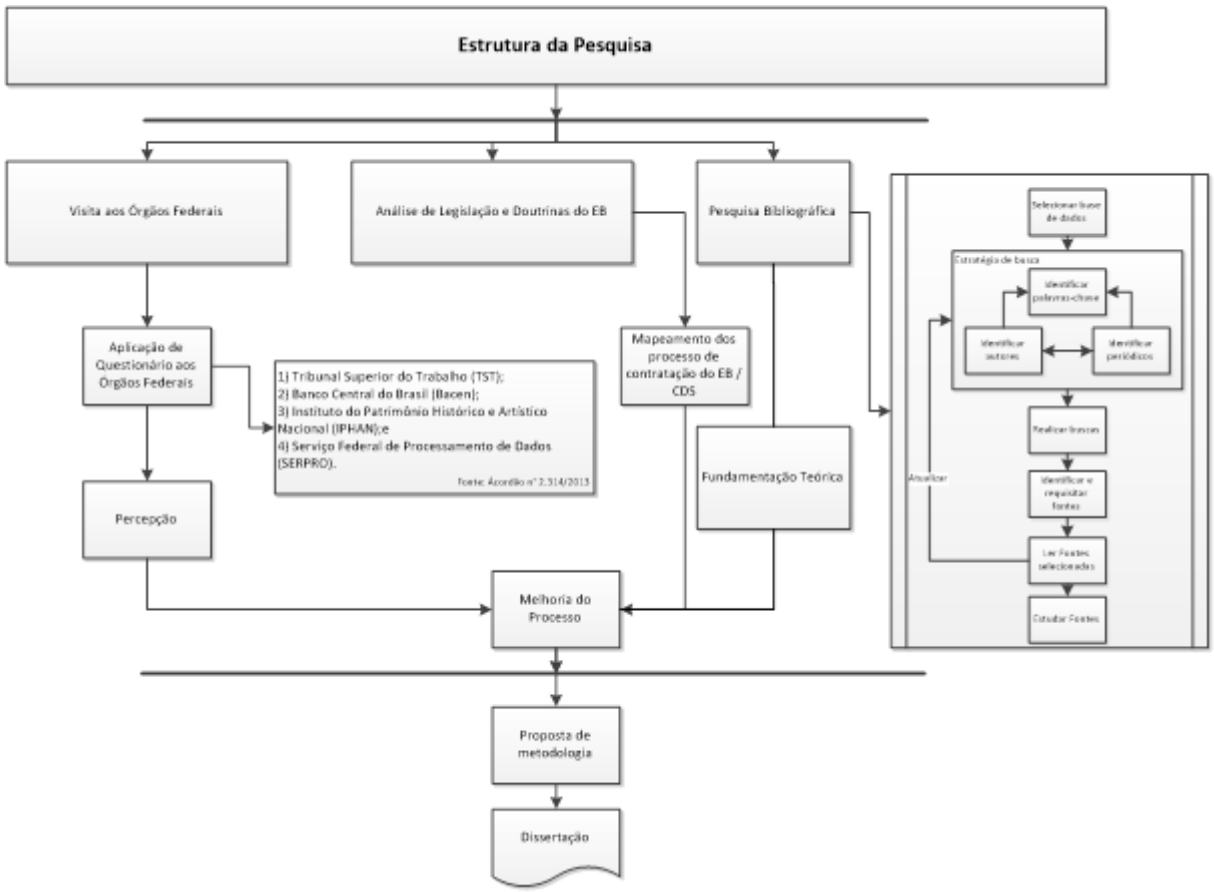

Figura 3.2: Estrutura da pesquisa, Fonte: (MEDEIROS, 2008)[44], adaptado pelo autor

3.2 Metodologia para avaliação dos *"Benchmarks"*

Para Marconi e Lakatos (2007)[41], todas as ciências caracterizam-se pela utilização de métodos científicos, destacando que nem todos os ramos de estudo que empregam métodos são ciência. Logo não há ciência sem o emprego de métodos científicos.

Como abordagem metodológica a pesquisa foi considerada, no ponto de vista do pesquisador, como qualitativa. Tendo em vista que se buscou a aproximação da teoria e dos fatos, por meio da descrição e interpretação de episódios isolados, privilegiando o conhecimento das relações entre contexto e ação. Por intermédio de análises fenomenológicas e da subjetividade do pesquisador que se chegou à formulação da proposta de metodologia de desenvolvimento de *software* ágil a ser adotado pelo EB (BERTO e NAKANO, 2000)[12].

Quanto à modalidade da pesquisa, esta é classificada como exploratória, por se tratar de uma análise das metodologias ágeis utilizadas pelas APF, bem como dos métodos de gestão de risco de TI, explorando os métodos de desenvolvimento do projeto de *software*. Desta forma, visa proporcionar maior familiaridade com as boas práticas adotadas pelas diversas APF (CERVO, BERVIAN e SILVA, 2007)[27].

Na classificação técnica existe uma conjugação das técnicas documental e revisão bibliográfica. Considerando que a pesquisa se apoia em aspectos descritos em livros, artigos, normas, dissertações e teses, ao fornecer embasamento para a análise do tema, envolvendo-se profundamente no estudo, a fim de detalhar melhor o conhecimento.

Complementar às abordagens metodológicas apresentadas a pesquisa caracteriza-se, quanto aos fins, como descritiva porque visa conhecer e analisar o processo de *software* das APF participantes do estudo de caso. Já quanto aos meios, caracteriza-se de campo, com a execução da coleta de dados de forma triangulada, aplicando entrevista, análise documental e observação direta participativa. Os dados que foram coletados e analisados e as informações que foram consolidadas resultaram em gráficos para representação visual e serviram de subsídio para a conclusão da pesquisa.

A pesquisa de campo é uma investigação realizada no local onde ocorre ou ocorreu um fenômeno ou, ainda, apresenta elementos para explicá-lo. Pode incluir entrevistas, aplicação de questionários, testes e observação participante ou não. No caso deste projeto de pesquisa, foi classificado como técnica de estudo de caso, na qual foi aplicado um questionário que gerou resultados quantitativos que auxiliaram no desenvolvimento do método a ser proposto ao EB (CERVO, BERVIAN e SILVA, 2007)[27].

Yin (2001)[62], entende que embora as pesquisas possam se sobrepor, o poder diferenciador do estudo é a sua capacidade de lidar com uma ampla variedade de evidências, documentos, artefatos, entrevistas e observações, além do que pode estar disponível no estudo histórico convencional.

Existe a possibilidade de identificar algumas situações de pesquisa em que todas as estratégias podem ser relevantes, e outras situações em que se podem considerar duas estratégias de forma igualmente atraente. Assim como, também podemos utilizar mais de uma estratégia em qualquer pesquisa a exemplo de um levantamento em um estudo de caso ou um estudo de caso em um levantamento (Yin, 2001)[62].

Em outras palavras a pesquisa é uma investigação empírica que investiga um fenômeno contemporâneo de seu contexto, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos (Yin, 2001)[62].

Depois da pesquisa bibliográfica pronta, foram, portanto, estruturados os Estudos de Caso no sentido de aprofundar o conhecimento acerca de um problema não suficientemente definido, visando estimular a compreensão, sugerir hipóteses e questões ou desenvolver a teoria (MATTAR, 1996)[43]. Corroborando o que Gil, (2007)[33]; Berto e Nakano, (2000)[12] afirmam se tratar de uma análise aprofundada de um ou mais objetos (casos), para que se permita seu amplo e detalhado conhecimento.

Os estudos de casos podem ser classificados segundo (YIN, 2001)[62]; (VOSS, 2002)[60] seu conteúdo e objetivo final ou quantidade de casos. No caso desta pesquisa, os Estudos desenvolvidos foram de cunho descritivo, quando foram analisados múltiplos casos, são eles: a metodologia de desenvolvimento software aplicada no EB e a metodologia ágil adotada nas APFs do TSE, IPHAN e BACEN.

1. Estudo de Caso aplicado ao EB

   O Estudo de Caso foi realizado no CDS, Organização Militar (OM) que possui como missão conceber, desenvolver, integrar e aperfeiçoar sistemas, programas, aplicativos e estruturas lógicas dos diversos sistemas corporativos e sistemas de informações operacionais do EB.

   As etapas empregadas para a investigação ocorreram com base na pesquisa bibliográfica citada fazendo uso de visitas técnicas e da aplicação de questionário, no intuito de recolher informações prévias sobre o campo de interesse.

   Foi utilizado como insumo desta etapa a análise de métodos (*frameworks*) de desenvolvimento tradicional e ágil de *software*, a fim de fundamentar todo o procedimento e referencial teórico para que os gestores realizem a gestão dos riscos do processo de desenvolvimento de *software* ágil no EB.

2. Estudo de Caso aplicado às APFs

   O Estudo aplicado às APFs seguiu as etapas apresentadas na figura 20, iniciando com o Planejamento do Caso, sequenciado com a estruturação do referencial teórico, seguido da realização das entrevistas de forma estruturada às APFs, com a

análise documental, observação direta participativa e tendo como objetivo realizar a comparação dos métodos ágeis analisados e utilizados pelas APF visitadas.

A aplicação dos métodos foi seguida da verificação da qualidade dos dados a serem coletados na aplicação do questionário. Em seguida foi feita a coleta e análise dos mesmos, sendo concluído com o desenho de implementações teóricas, objeto da presente pesquisa, como detalhado na Figura 3.3.

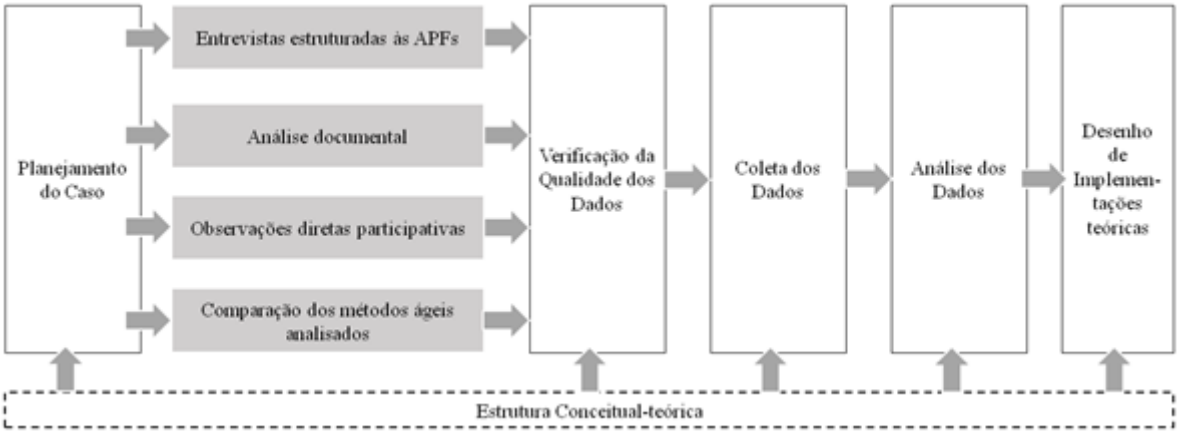

Figura 3.3: Etapas do Estudo de Caso, fonte: desenvolvido pelo autor

As visitas técnicas para a análise das metodologias ágeis utilizadas ocorreram em 3 (três) Órgãos Federais, com aplicação de questionário e observação das normas aplicadas, i) Tribunal Superior Eleitoral (TSE), ii) Instituto do Patrimônio Histórico Nacional (IPHAN) e o iii) Banco Central do Brasil (BACEN), dentre as quais formam identificadas as seguintes metodologias:

1. No TSE é adotado o Método de Desenvolvimento com Práticas Ágeis (MAgil) (BRASIL, 2014)[24], como proposta agregar ao dia a dia das equipes as melhores práticas do desenvolvimento ágil, sendo recomendado a projetos nos quais se é possível atuar com agilidade e pouco formalismo e cujo foco seja a produção e entrega de funcionalidades do sistema de forma incremental;

2. No IPHAN foi apresentado a Metodologia IPHAN de Gestão de Demandas de Desenvolvimento Ágil de *Software* (MIDAS) como um guia corporativo que estabelece o fluxo de gerenciamento de demandas, fundamentado na metodologia ágil Scrum e em boas práticas de mercado. (BRASIL, 2013)[17]; e

3. No BACEN o desenvolvimento e a manutenção de *software* é adotado o Processo Ágil de Desenvolvimento do Banco Central do Brasil (PDS-BC Ágil) como guia para a construção do produto de *software* correto e com qualidade (BRASIL, 2014)[20].

Como citado, foi utilizado um questionário durante as visitas. Estes contêm os dados obtidos nas referências e pode ser visto como uma fonte de informação, em que objetiva levantar dados para serem analisados e enriquecer o conteúdo da pesquisa. Os dados obtidos nos questionários transmitem a realidade do tema em questão. Desta forma, o método utilizado para o levantamento dos dados é classificado como entrevista sistemática estruturada.

Os dados para análise foram obtidos por meio de questionário respondido por APF que trabalham com o desenvolvimento de *software* ágil. As instituições selecionadas para responder o questionário já foram visitadas pelo Tribunal de Contras da União (TCU) descritos na pesquisa, justamente por serem instituições envolvidas com o uso do método ágil.

O questionário foi elaborado com 20 questões, com o objetivo de conhecer o perfil profissional dos envolvidos, anos de experiência na profissão e o grau de conhecimento. Além disso, teve o propósito de levantar informações sobre a situação atual das instituições pela visão dos profissionais e identificar a metodologia utilizada. Após a elaboração do questionário, foi entregue a 15 profissionais de cinco organizações. Dos quinze questionários entregues, dez foram respondidos e aproveitados para análise.

Para dar seguimento à coleta de dados foi necessário criar um formulário com o auxílio do Google Docs, disco de armazenamento na Internet que permite guardar e editar documentos, bem como compartilhar com outros usuários. Logo que o questionário foi inserido nesta ferramenta, o endereço do mesmo na Internet foi disponibilizado para as cinco organizações.

Mesmo com o auxílio da ferramenta, alguns entrevistados não responderam ao questionário. Em vista disso, o resultado foi fruto da coleta de respostas de 10 profissionais, distribuídos no Tribunal Superior do Trabalho (TST), Tribunal Superior Eleitoral (TSE), Instituto do Patrimônio Histórico Nacional (IPHAN), Banco Central do Brasil (BACEN), e o Serviço Federal de Processamento de Dados (SERPRO), mas os dados obtidos foram suficientes para análise, a qual será descrita no capítulo seguinte.

Por fim, a partir da classificação da pesquisa apresentada quanto à abordagem, técnicas e os meios, descritas anteriormente, o trabalho está sistematizado conforme apresentado na Figura 3.4. Isto é, a pesquisa está dividida em três fases distintas. A primeira corresponde à pesquisa bibliográfica, a segunda etapa diz respeito aos Estudos de Casos citados culminando na proposta do processo metodológico ágil de desenvolvimento de *software* para o EB com base no modelo de gestão de risco.

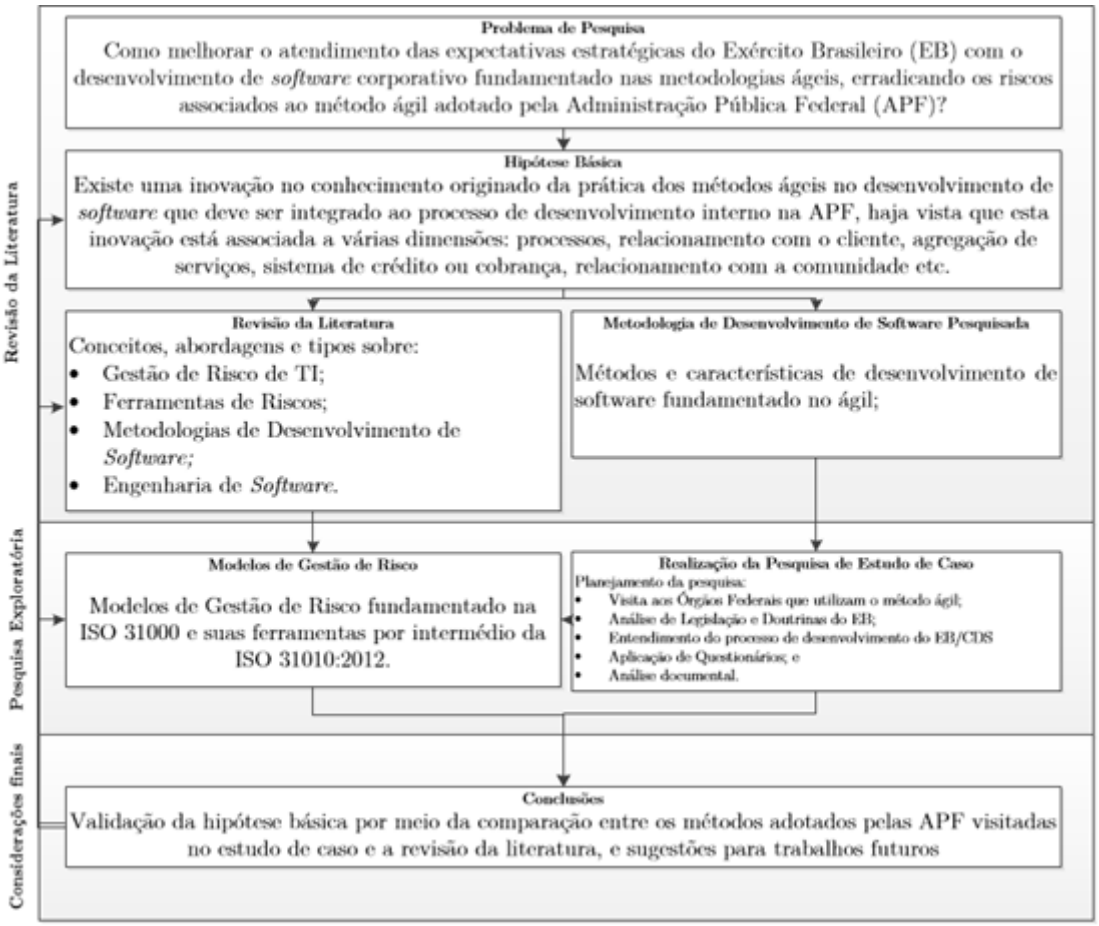

Figura 3.4: Sistematica da pesquisa, Fonte: desenvolvido pelo autor

Capítulo 4

Estudo de Caso

Neste capítulo é apresentado o estudo realizado no processo de desenvolvimento de *software* aplicado no EB, seguido do estudos dos *benchmarks* realizados nas APFs, observando as características do processo de desenvolvimento de *software* ágil adotado pelas Instituições visitadas, finalizando o capitulo com a análise da severidade dos riscos apontados pelo TCU por intermédio do Acórdão n. 2314-33/2013, no ponto de vista das Instituições visitas e pelos Gerentes de Projetos do Centro de Desenvolvimento de Sistenas do Exército Brasileiro.

4.1 Contexto do Estudo

A fim de contextualizar o cenário de onde partiram as análises da problemática e que motivou a formulação da presente pesquisa, faz-se necessária uma descrição geral do cenário da TI no Exército Brasileiro (EB) e de onde provêm as possíveis demandas a nível corporativo.

No EB, a TI está inserida nas atividades de Ciência e Tecnologia gerenciadas pelo Departamento de Ciência e Tecnologia (DCT). Além disso, demandas de serviços de TI podem chegar ao DCT de outros Departamentos e Diretorias, que não fazem parte do Sistema de Ciência e Tecnologia do Exército. Tais demandas, depois de avaliadas pelo DCT, são homologadas como sendo projetos coorporativos, que serão encaminhadas aos Órgãos sob sua responsabilidade e que efetivamente irão executar o projeto.

O CDS é o Órgão do Sistema de Ciência e Tecnologia do Exército responsável por conceber, desenvolver, integrar e aperfeiçoar sistemas, programas, aplicativos e estruturas lógicas dos diversos sistemas coorporativos e sistemas de informações operacionais do Exército.

Associada a esse cenário está à capacidade de se exercer a governança dos riscos de TI como parte integrante da Governança Corporativa, e que consiste no estabelecimento

de mecanismos para assegurar que o uso da TI agregue valor ao negócio, com riscos aceitáveis. Pretende-se com este estudo, reunir informações de governança de TI oriundas de modelos consagrados para apontar caminhos que possibilitem melhorar a gestão dos riscos do processo de desenvolvimento de *software* do EB fundamentado na metodologia ágil.

4.1.1 Centro de Desenvolvimento de Sistemas do Exército (CDS)

O universo da pesquisa se restringe a Organização Militar (OM) do Exército Brasileiro (EB). A escolha desse universo deve-se ao fato do pesquisador integrar o efetivo do Centro de Desenvolvimento de Sistemas (CDS), OM responsável pela assessoria técnica de desenvolvimento dos Sistemas de Informações do EB.

O Exército Brasileiro é uma das três Forças Armadas, cuja missão constitucional é preparar a Força Terrestre para defender a Pátria, garantir os poderes constitucionais, a lei e a ordem, participar de operações internacionais, cumprir atribuições subsidiárias e apoiar a política externa do País (BRASIL, 1988)[13].

O EB possui uma estrutura funcional hierarquizada, contudo, ao longo dos últimos anos, vem passando por transformações, no que diz respeito à modernização de sua estrutura organizacional e dos seus processos administrativos, planejados no Programa de Excelência Gerencial do Exército Brasileiro (PEG-EB), corroborado pelo Comandante, por intermédio da Portaria n. 220 Cmt. Ex., de 20 de abril de 2007 (BRASIL, 2007)[15].

Inserida no contexto da modernização citada, a visão apresentada pelo Comando da Força é chegar a uma nova doutrina com o emprego de produtos de defesa tecnologicamente avançados, profissionais altamente capacitados e motivados para que o exército enfrente, com os meios adequados, os desafios do século XXI (BRASIL, 2015b)[23].

Para o Comando do Exército, uma visão fundamentada em processos significa empenhar-se em melhorar os processos básicos da organização, com base no pressuposto de que o resultado desejado é alcançado mais eficientemente quando as atividades e os recursos relacionados são gerenciados de forma integrada.

A apresentação do Modelo de Negócio do Exercito Brasileiro tem o objetivo de descrever de forma simplificada o encadeamento dos elementos estratégicos que compõem a instituição. Para isto, a ferramenta *Business Model Canvas* (BMC) apresentada por Osterwalder e Pigneur (2009)[47], foi utilizada no intuito de promover o entendimento da integração dos processos organizacionais, da operacionalização da entrega do valor ao cliente e da construção da missão da organização considerando a participação das atividades de apoio, representada pelo processo de desenvolvimento de *software*, objeto da presente pesquisa.

Na Figura 4.1 é apresentado o CANVAS do EB (BRASIL, 2009a)[22].

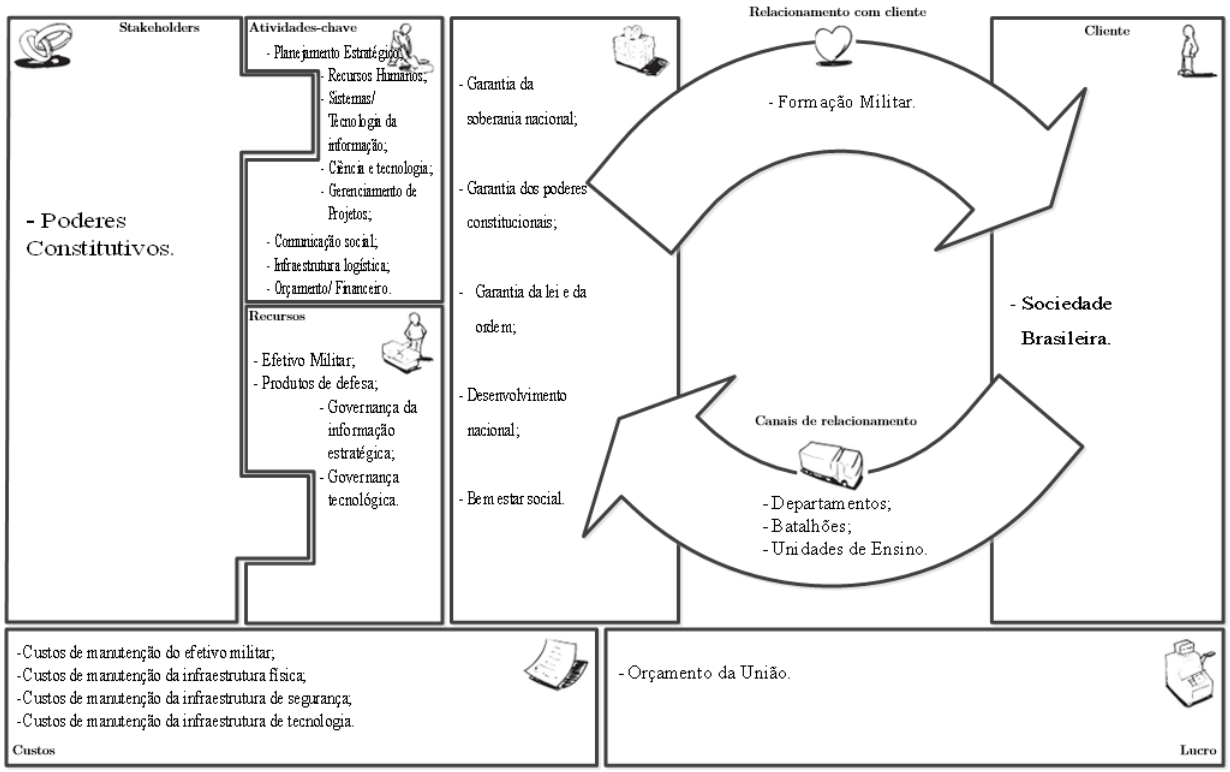

Figura 4.1: CANVAS do EB, Fonte: (OSTERWALDER; PIGNEUR, 2009), adaptado pelo autor

Com base na missão do EB apresentada, a Proposta de Valor presente na figura visa responder às necessidades dos Clientes. Com isso, é oferecida à Sociedade Brasileira a garantia da soberania nacional, bem como dos poderes constitucionais e da lei e da ordem, com o objetivo de alcançar o desenvolvimento nacional e o bem estar social. Essa entrega de valor é feita através dos Departamentos, Batalhões e das Unidades de Ensino.

O orçamento da união é fonte de recursos para as atividades do Exército Brasileiro, como definido constitucionalmente, e com isso mantém os recursos necessários para entrega do valor à sociedade brasileira, estes são listados como: os militares, os produtos de defesa, a governança da informação estratégica e a governança tecnológica. A fonte orçamentária citada também é responsável pelo cumprimento dos custos incorridos da prestação do serviço, tais como a manutenção do militar e da infraestrutura logística, de segurança e da tecnologia.

Os processos ou atividades chave que compõem a prestação dos serviços do EB são o planejamento estratégico, os gerenciamento dos recursos humanos, os sistemas e as tecnologias da informação, a ciência e a tecnologia, o gerenciamento dos projetos, a comunicação social, a infraestrutura logística e a gestão orçamentária.

Durante a elaboração da Figura 4.1, foi entendido por partes interessadas aqueles que legitimam as ações da organização e que, desta forma, têm papel direto ou indireto em seus processos. No caso do Exército Brasileiro os poderes constitutivos, (Executivo, Judiciário e Legislativo), fazem este papel. A estrutura organizacional do EB apresenta Órgãos de Assistência Direta e Imediata (OADI), Órgãos de Assessoramento Superior (OAS), Órgão de Direção Geral (ODG), Órgãos de Direção Setorial (ODS) e a Força Terrestre (FT). Durante a pesquisa, foi interagido com o Departamento de Ciência e Tecnologia (DCT) e o Centro de Desenvolvimento de Sistemas (CDS).

O DCT é o ODS responsável pelo gerenciamento do Sistema de Ciência e Tecnologia do Exército para produzir os resultados científico-tecnológicos necessários à Força Terrestre. Possui como organização diretamente subordinada, o Centro de Desenvolvimento de Sistemas (CDS), cuja missão é conceber, desenvolver, integrar e aperfeiçoar sistemas, aplicativos, programas e estruturas lógicas dos diversos sistemas coorporativos e sistemas de informações operacionais do EB.

O CDS, foco deste estudo, está inserido dentro da estrutura organizacional do Comando do Exército Brasileiro (EB), composto por Chefia, SubChefia, Divisão de Apoio, Divisão de Sistemas, Divisão de Engenharia, Divisão de Planejamento Coordenação e Controle, Divisão de Segurança, Divisão de Comando e Controle.

Na Figura 4.2, é apresentado o CANVAS do CDS que compôs o universo estudado.

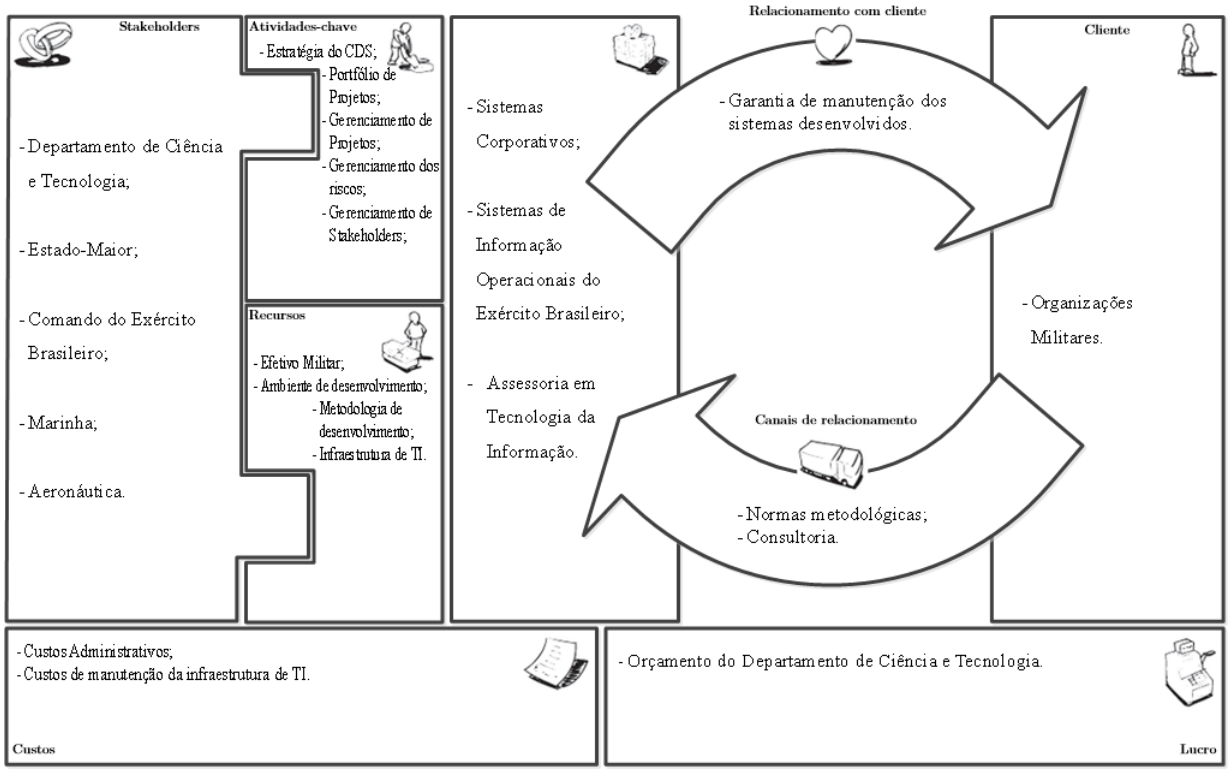

Figura 4.2: CANVAS do CDS, Fonte: (OSTERWALDER; PIGNEUR, 2009), adaptado pela autor

Na Figura 4.2, as Organizações Militares (OM) por meio da Consultoria prestada e das Normas Metodológicas estabelecidas pelo CDS recebem os sistemas corporativos, os sistemas de informação operacionais e a assessoria em tecnologia da informação produzida pelo Centro. Dessa forma, é estabelecido o relacionamento entre o prestador de serviço - CDS - e cliente - OM. A fim de manter este vínculo, o Centro ainda garante a manutenção dos sistemas que foram desenvolvidos.

Em termos das entradas financeiras necessárias para a prestação do serviço citado, toda ela é resultado do orçamento do Departamento de Ciência e Tecnologia (DCT), o qual é responsável por manter os recursos necessários para o desenvolvimento dos sistemas, são eles: os Militares, o Ambiente de Desenvolvimento, a Metodologia e a Infraestrutura de TI. Em conjunto a estes recursos, o orçamento também prevê o cumprimento dos custos incorridos, tanto os custos administrativos quanto os custos de infraestrutura de TI.

Algumas atividades são consideradas principais para a performance do Centro, no que tange à entrega do valor acordado com o cliente. Estas estão agrupadas na elaboração e manutenção da estratégia do CDS, no gerenciamento do portfólio de projetos, no gerenciamento dos riscos, no gerenciamento dos *stakeholders* e na própria gestão dos projetos de prestação do serviço.

Foram consideradas partes interessadas no serviço prestado pelo CDS, o DCT, como já apresentado, o Estado-Maior e o Comando do Exército, como dirigentes estratégicos de toda a força, bem como a Marinha e a Aeronáutica, no desenvolvimento de sistemas em parcerias, observando a missão do Exército Brasileiro de soberania e desenvolvimento nacional.

Apresentadas as duas estruturas, do Exército Brasileiro e do CDS, se faz evidente de que forma a entrega do valor do Centro compõe um pilar na agregação de valor proposta pela força. Em outras palavras, à medida que as Organizações Militares fazem uso dos serviços prestados pelo Centro em termos dos sistemas e da assessoria oferecida, estas se tornam parte da base necessária para a execução das propostas de valor do EB: a garantia da soberania nacional, dos poderes constitucionais, da lei e da ordem, o desenvolvimento nacional e o bem estar social.

A governança de TI como recurso para o modelo de negócio do EB, segundo Fernandes e Abreu (2012)[31], busca o compartilhamento das decisões de TI com os dirigentes da força, assim como estabelece as regras, a organização e os processos que norteiam o uso da TI pelos usuários, departamentos, divisões, negócios da organização, fornecedores e clientes, e também determina como a TI deve prover os serviços para a organização. É, em suma, um mecanismo de alinhamento da TI ao negócio. Esta afirmação revela a importância da proposta de valor do CDS no sentido de subsidiar as tomadas de decisão do Comando da força a fim de avaliar os rumos a serem percorridos bem como garantir o alcance dos objetivos da organização.

Para Porter (1985) a cadeia de valor é composta de atividades "primárias" e atividades de "suporte", sendo que a cadeia de valor do processo de negócio descreve as atividades (processos) que fornecem valor ao cliente, de modo que, cada uma dessas atividades tem seus próprios objetivos de desempenho vinculados a seu processo de negócio principal. As atividades primárias são aquelas envolvidas com a criação física de um produto ou serviço, referidas como agregação de valor (ABPMP, 2009)[4].

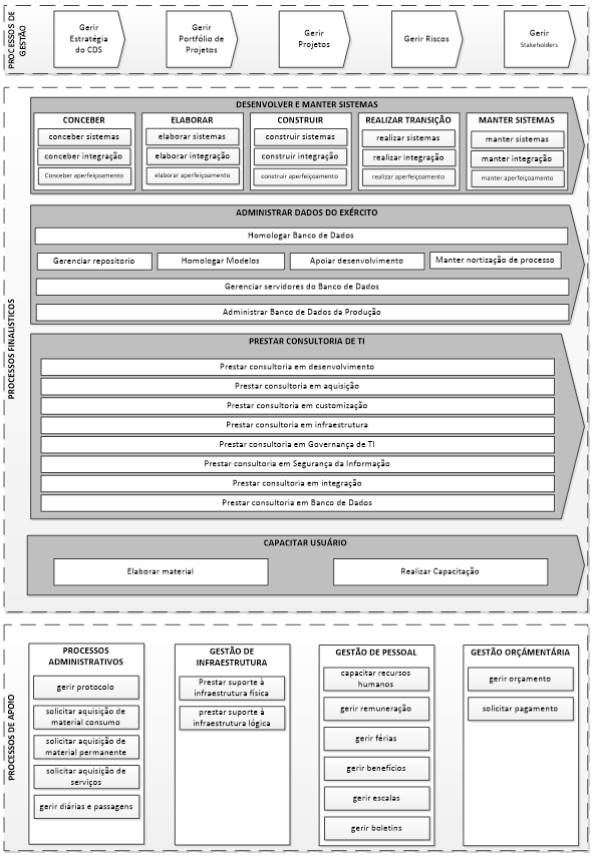

Figura 4.3: Cadeia de Valor do CDS, Fonte: desenvolvido pelo autor

No processo de gestão descritos na Figura 4.3, estão os processos de gerir estratégia do CDS, Portfólio de Projetos, Projetos, Riscos e *Stakeholders* das diversas atividades atribuídas ao Centro.

Já nos processos finalísticos estão os macroprocessos de desenvolver e manter sistemas sob responsabilidade do CDS, com foco nos processos de conceber, elaborar, construir, realizar transição a manter sistemas fundamentados em métodos reconhecidos e normativos de desenvolvimento, o processo de Administrar Dados do Exército, com as atividades de homologar Banco de Dados, Gerenciar repositório, Homologar modelos, Apoiar desenvolvimento, Manter normatização de processo, Gerenciar servidores de Banco de Dados e Administrar Banco de Dados da Produção.

Complementar aos processos finalísticos estão Prestar Consultoria de TI e Capacitar usuário, com atividades de Prestar Consultoria em desenvolvimento, em aquisição, customização, infraestrutura, Governança de TI, Segurança da Informação, Integração e Banco de Dados no processo de prestar consultoria de TI e Elaborar material e Realizar Capacitação como atividades do processo de Capacitar usuário de sistemas de TI do Exército.

Como mecanismo de apoio estão previstos os Processos Administrativos, Gestão de Infraestrutura, Gestão de Pessoal e Gestão Orçamentária, com atividades de gerir protocolo, solicitar aquisição de material de consumo e permanente, solicitar aquisição de serviços, gerir diárias e passagens, prestar suporte à infraestrutura física e lógica, capacitar recursos humanos, gerir remuneração, férias, benefícios, escalas e boletins, bem como, orçamento e solicitar pagamento, dos respectivos processos de apoio.

Destarte, como mecanismo produtivo de apoio ao cumprimento da missão prevista ao Centro, o tópico a seguir apresentará a análise dos processos de desenvolvimento de *software* normativo do Exército.

4.2 Estudo dos Processos

4.2.1 De Desenvolvimento de *Software* do Exército

O Manual Técnico para a Metodologia de Desenvolvimento de *Software* do Exército (MDS-EB) foi aprovado por intermédio da Portaria n. EB80-MT-78.001, elaborado com base nos princípios apregoados pelo RUP, reproduzindo o ciclo de vida de um software de acordo com as Instruções Gerais (IG) do Ciclo de Vida do *Software* do Exército, fundamentado nos processos de aquisição, fornecimento, desenvolvimento, produção e manutenção, dentre os quais são constituídos com atividades e tarefas de análise de requisito, projeto, codificação, integração, testes, instalação e aceitação relacionadas aos produtos de *software*. (BRASIL, 2012)[21]

Como finalidade, o MDS visa descrever e padronizar os processos para o desenvolvimento de *software* no âmbito do CDS e, posteriormente, servir como orientação às demais Organizações Militares do Exército. (BRASIL, 2012)[21]

O MDS (2012)[21] está dividido em quatro fases distintas, vide Figura 4.4: Iniciação, Elaboração, Construção e Transição, com a definição e aprovação do escopo na Iniciação, com a definição e validação da arquitetura na fase de Elaboração, com a codificação e teste na fase de Construção e finalmente com a homologação do *software* na fase de Transição.

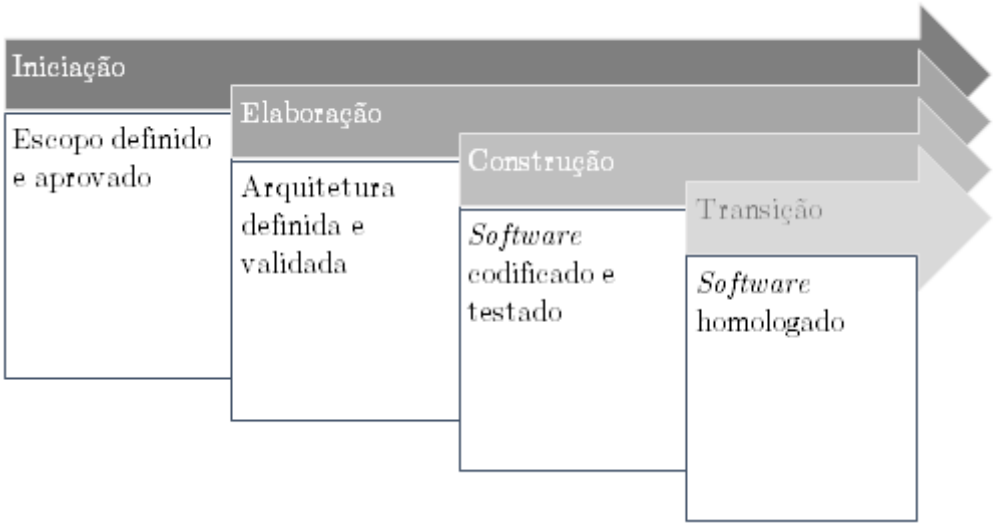

Figura 4.4: Fases do Desenvolvimento de *software* do EB, Fonte: (BRASIL, 2012)[21], adptado pelo autor

Conforme demonstrado no diagrama da Figura 4.5, a etapa de Iniciação do processo de Desenvolvimento de *Software* do MDS é composta de 10 atividades com 01 subprocesso, que resultam em 13 artefatos, envolvendo 09 atores no processo distribuídos nos papeis de Gerente de Projeto, Analista, Arquiteto, Projetista, Administrador de Banco de Dados (AD/DBA), Desenvolvedor, Testador, Cliente e Projetista de Infra estrutura, destacando que no subprocesso de Identificar e refinar requisitos são somadas outras 02 atividades com 07 artefatos.

Nesta fase, o Gerente de Projeto irá planejar o projeto, enquanto simultaneamente o Analista está definindo a visão do produto e o Arquiteto está descrevendo a arquitetura do sistema e preparando o ambiente de produção e homologação do sistema que será produzido. Neste momento, o Projetista deve analisar o ambiente de produção desenvolvido e elaborar o Relatório de Infraestrutura no sentido de fornecer insumo para os demais atores durante o planejamento da fase de Iniciação.

Dado o planejamento do projeto e a definição da visão, fazendo uso do Documento de Visão e do Plano de Projeto, o Gerente de Projeto irá planejar a iteração dos itens de trabalho. O Plano de Iteração será, portanto, entrada para o subprocesso de Identificação e refino dos requisitos do sistema os quais serão homologados, posteriormente, pelo cliente.

A fim de garantir a integração das funcionalidades do *software* que está sendo desenvolvido, o gerenciamento da iteração, bem como o planejamento da iteração seguinte deverão ocorrer em paralelo ao subprocesso citado. Por fim, a fase é encerrada com a elaboração do Relatório de Avaliação da Iteração.

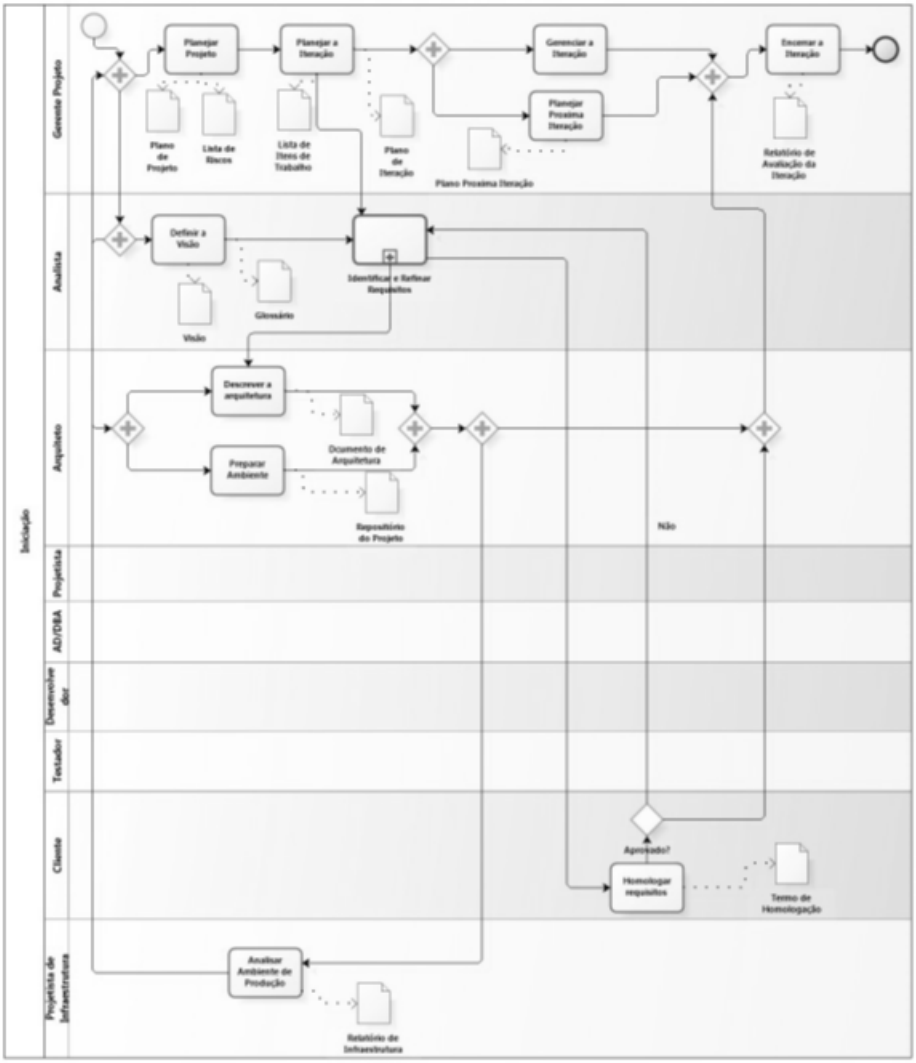

Figura 4.5: Processo de Iniciação do MDS, Fonte: (BRASIL, 2012)[21]

Na fase de Elaboração e Construção do processo de Desenvolvimento de *Software* do MDS, apresentada na Figura 4.6, de forma duplicada, são identificadas 24 atividades, 06 subprocessos e 26 artefatos. Dos subprocessos citados são apontadas 16 atividades com a participação de 09 atores envolvendo 12 artefatos. Para dar início a esta etapa o Gerente de Projeto deve revisar as iterações anteriores para produzir informação tanto para o subprocesso de Identificação e refino dos requisitos referentes a iteração corrente quanto para o gerenciamento da mesma. Além disso, a informação produzida também será utilizada no planejamento da próxima iteração.

Como resultado do subprocesso citado, o Arquiteto refina a arquitetura do sistema que está sendo desenvolvido para fornecer insumos ao Projetista durante o projeto da solução. O Administrador do Banco de Dados irá em sequência validar e implementar o modelo de dados. Este modelo validado é entrada do segundo subprocesso o qual se refere

ao desenvolvimento do incremento da solução da iteração.

Ao longo deste desenvolvimento, o Cliente irá homologar os requisitos gerados e avaliar a iteração, enquanto o Testador deve testar a solução da iteração. Por fim, esta fase é finalizada pelo Gerente de Projeto com o Relatório de Avaliação da Iteração.

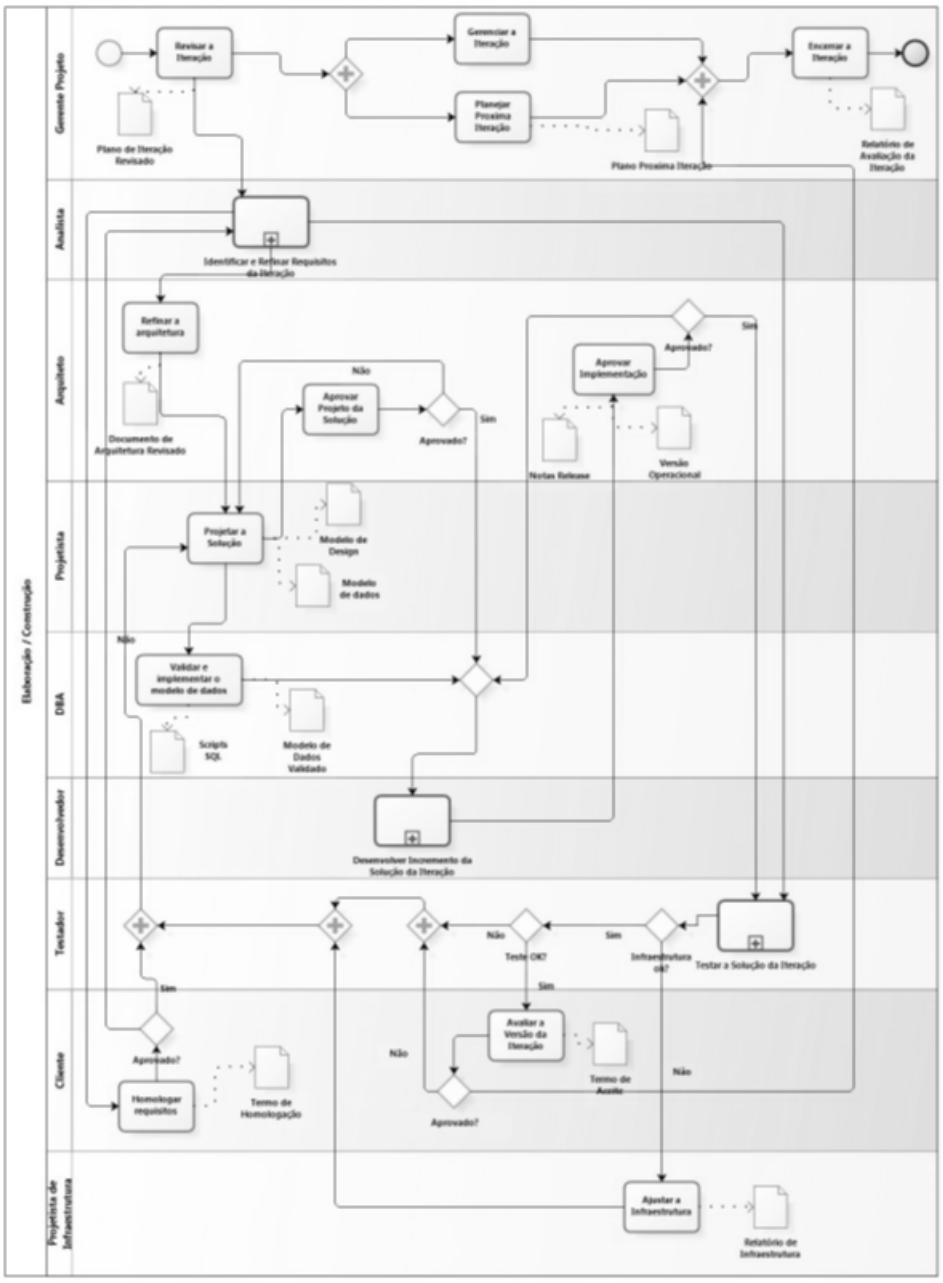

Figura 4.6: Processo de Elaboração do MDS, Fonte: (BRASIL, 2012)[21]

Finalmente na última fase do processo de Desenvolvimento de *Software* do MDS, nomeada fase de Transição apresentada na Figura 4.7, foram identificadas 10 atividades, 10 artefatos os quais foram produzidos por 09 atores já citados anteriormente.

O Gerente de Projeto deve, assim como na fase anterior, revisar as iterações anteriores para produzir informação para o gerenciamento da mesma e para o planejamento da próxima iteração, bem como para fornecer insumos ao Analista para o planejamento da implantação do *software*. Com o Plano de Implantação elaborado, este ator irá, em seguida, desenvolver material de suporte e treinar os usuários.

O Arquiteto, por sua vez, é responsável pela implantação do produto em produção, o qual ocorre utilizando informações do planejamento elaborado pelo Analista, e que ao final será validado pelo Cliente.

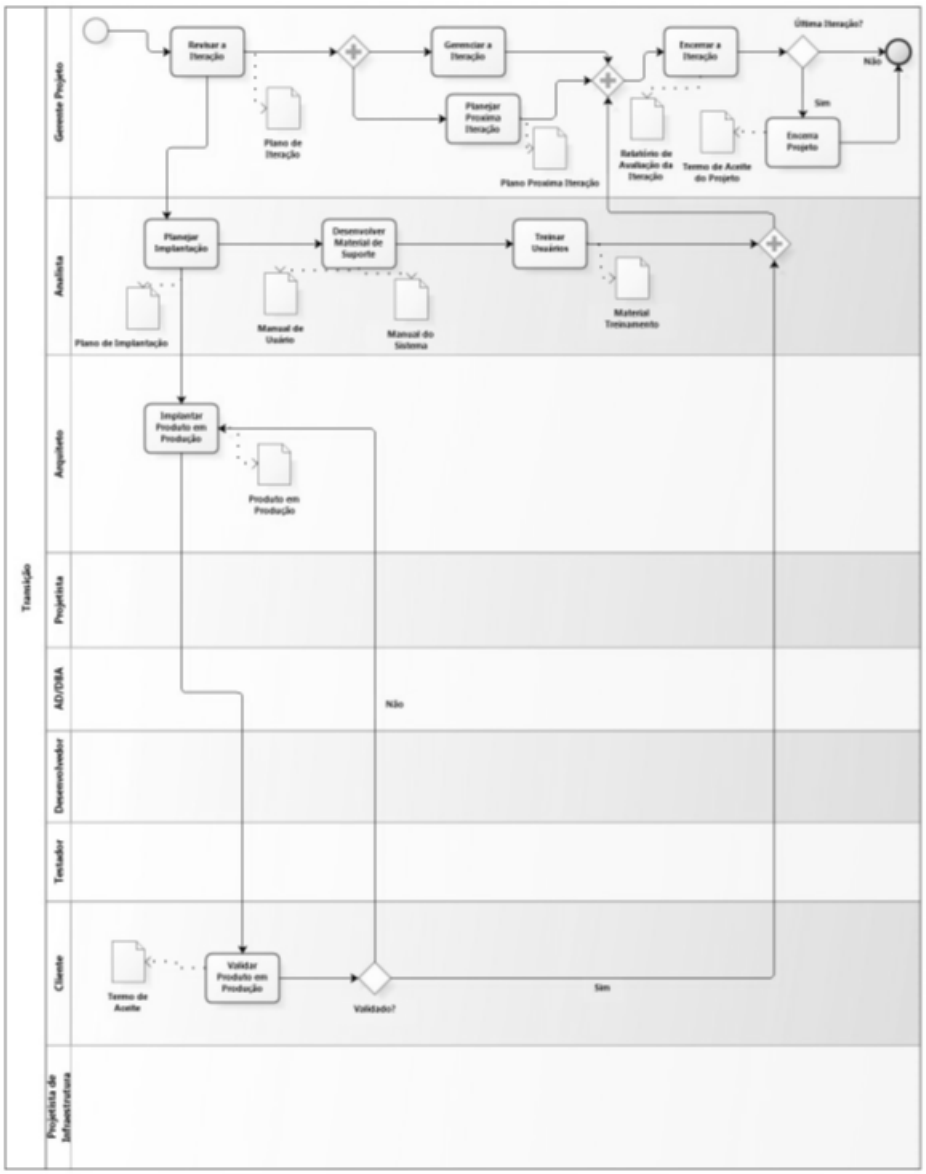

Figura 4.7: Processo de Transição do MDS, Fonte: (BRASIL, 2012)[21]

Por fim, é possível aferir que no processo atual de Desenvolvimento de *Software* adotado pelo EB são executadas 62 atividades, com 07 subprocessos, onde são gerados 68 artefatos no total, produzidos com o envolvimento de 09 atores.

A próxima seção apresenta os cases utilizados como referência em boas práticas de desenvolvimento de *software* que adotam métodos ágeis, utilizado pela Administração Pública Federal (APF), especificamente o IPHAN, TSE e o BACEN.

4.3 Referências de Órgãos que adotam métodos ágeis no processo de desenvolvimento de *software*

O objetivo deste tópico é identificar as práticas adotadas por autarquias da Administração Pública Federal (APF), no desenvolvimento de *software*. A pesquisa se deu por meio de coleta documental, entrevistas e aplicação de questionário que estabeleceram as características e processos utilizados na prática de métodos ágeis no desenvolvimento de *software*, internamente, ou mesmo, por intermédio de contratação externa.

A análise documental contribuiu como fonte para melhor entendimento das práticas adotadas pelas autarquias de forma essencial a evidenciar um estudo de caso e melhor alinhar os objetivos de desenvolver o modelo para o Exército Brasileiro.

Os objetivos das entrevistas realizadas nesta pesquisa foram para melhor compreender as práticas realizadas pelas autarquias e complementar as informações obtidas pelo exame documental, avaliar a importância do tema do uso de métodos ágeis no desenvolvimento de software, em regime de fábrica, ou seja, por intermédio de contratação pela administração pública e identificar atitudes, opiniões, ideias, procedimentos e processos não caracterizados na análise documental e nos questionários. O questionário aplicado foi elaborado com questões objetivas ligadas ao uso de métodos ágeis e formas de contratações e seus respectivos impactos, como forma complementar aos outros instrumentos de pesquisa.

Inicialmente, a execução da pesquisa contou com o apoio de alunos do curso de Engenharia de Produção da Universidade Federal de Brasília (UnB), que previamente participaram das entrevistas e na análise dos documentos disponibilizados, com objetivo de modelar as melhores práticas utilizadas pelas autarquias federais visitadas.

Os próximos tópicos irão apresentar as autarquias entrevistadas.

4.3.1 Instituto do Patrimônio Histórico e Artístico Nacional (IPHAN)

O Instituto do Patrimônio Histórico e Artístico Nacional (IPHAN) é um órgão vinculado ao Ministério da Cultura, criado em 1937 e transformado em autarquia federal por intermédio da Lei n. 8.113/1990, com objetivo de preservar o patrimônio cultural do país. Tem sede em Brasília/DF, Setor de Edifícios Públicos Sul (SEPS) 713/913, Lote D, e circunscrição administrativa em todo o território nacional. (IPHAN, 2014)[36]

Logisticamente, possui 27 superintendências, uma em cada estado da federação e no Distrito Federal; 27 Escritórios Técnicos em cidades com conjuntos urbanos tombados e, ainda, quatro unidades especializadas: o Centro Nacional do Folclore e Cultura Popular, o Sítio Roberto Burle Marx, o Centro Cultural Paço Imperial e o Centro Nacional de Arqueologia. Encontra-se em processo de instalação o Centro Lucio Costa, reconhecido pela UNESCO como Centro de Categoria II, voltado para a formação em Gestão do Patrimônio e tendo como público alvo países de língua portuguesa e espanhola da América do Sul, África e Oceania. (IPHAN, 2014)[36]

Como mecanismo de apoio tecnológico institucional o IPHAN dentro de sua estrutura organizacional conta com a Coordenação Geral de Tecnologia da Informação (CGTI), diretamente subordinada ao Departamento de Planejamento e Administração (DPA) para o gerenciamento do desenvolvimento de produtos de *software* na organização, tendo como modelo de arquitetura de processos de TI apresentado pela Figura 4.8.

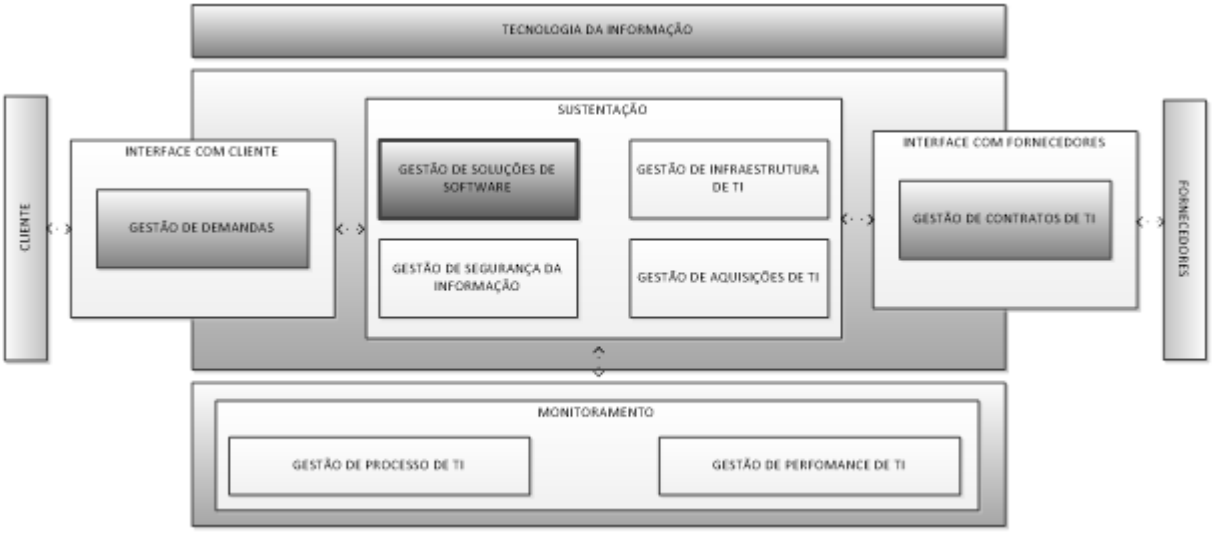

Figura 4.8: Contexto do modelo na arquitetura de processo de TI do IPHAN, Fonte: (BRASIL, 2013)[17]

Dentro do contexto arquitetural do processo de TI do IPHAN a CGTI, por intermédio de sua Divisão de Sistemas de Informação (DIVSIS), no âmbito do desenvolvimento e manutenção de Sistemas de Informação fundamentados na metodologia ágil utiliza

a Metodologia IPHAN de Gestão de Demandas de Desenvolvimento Ágil de *Software* (MIDAS) como um guia corporativo que estabelece o fluxo de gerenciamento de demandas, fundamentado na metodologia ágil *Scrum* e em boas práticas de mercado. (BRASIL, 2013)[17]

O MIDAS é um fluxo de trabalho e o passo-a-passo de como gerenciar as demandas de desenvolvimento de sistemas junto às empresas contratadas para tal fim, apesar da obrigatoriedade da elaboração de uma coleção de documentos públicos, esta metodologia tem por foco tão somente o acompanhamento do desenvolvimento de produtos de *software* (BRASIL, 2013)[17]. No âmbito do MIDAS, os conceitos, papéis e classificações do processo de desenvolvimento e gestão de contratos fundamentados na metodologia ágil são (BRASIL, 2013)[17]

1. Solução de *Software* é uma denominação genérica para *software* que automatiza, de modo parcial ou total, atividades de coleta, processamento, transmissão, armazenamento, recuperação e disseminação de dados que representam informação para o usuário ou cliente, ou para ambos.

2. Papéis principais:

    (a) *Product Owner*: representa a área requisitante do produto (área de negócio ou dono do produto) e os stakeholders (envolvidos), centra sua atuação nos itens relacionados ao cliente (histórias de usuário) garantindo que a solução agregue valor ao negócio. Prioriza funcionalidades, ajusta funcionalidade e prioridades, aceita ou rejeita o resultado dos trabalhos.
    (b) *Scrum Master*: é responsável pela remoção de impedimentos à capacidade da equipe para realizar as entregas. Sua atuação centra-se na manutenção do processo, no alinhamento às regras definidas e no foco da equipe às tarefas definidas. Garante a colaboração entre os diversos papéis e funções e atua como escudo às interferências externas.
    (c) *Team Scrum* (Equipe de Desenvolvimento): é a equipe responsável pela entrega do produto, composta pelas pessoas que executam o trabalho real (analisar, projetar, desenvolver, testar, documentar, etc.).
    (d) IPHAN: entidade governamental que utiliza sistemas de informação para cumprimento de seus objetivos institucionais e negociais; proprietária do processo atua como contratante e demanda atividades de gestão de serviços de TI.
    (e) Fábrica de *Softwares*: entidade responsável pela prestação dos serviços de desenvolvimento e manutenção de sistemas; atua como prestador externo de serviços contratado através de procedimento licitatório.
    (f) Fábrica de Qualidade: entidade responsável pela prestação dos serviços de controle da qualidade de sistemas; atua como prestador externo de serviços contratado através de procedimento licitatório.
    (g) Fábrica de Métricas: entidade responsável pela prestação dos serviços de medição de sistemas; atua como prestador externo contratado através de procedimento licitatório.

Dentro do contexto institucional do atendimento das atribuições previstas na IN SLTI/MP n. 04/2010 os papéis de Fiscal Técnico, atribuição essa prevista com representação funcional do órgão é representada no MIDAS com a função do "*Scrum Master* IPHAN". Já como representante da fabrica de *software* contratado, ou seja o preposto e ou Coordenador do Contrato, responsável por assegurar que a equipe de desenvolvimento contratada (*Team Scrum*) respeite as regras do projeto e realize as entregas definidas é representado pelo "*Scrum Master* fábrica de *softwares*", demonstrados na Tabela 4.1.

Tabela 4.1: Papéis MIDAS x Papéis IN SLTI/MP N.04/2010, Fonte: (BRASIL, 2013)[17]

Papéis	Papéis IN SLTI/MP 04/2010
Product Owner	-Área Requisitante da Solução - Integrante Requisitante -Fiscal Requisitante -Gestor do Contrato
Scrum Master IPHAN	-Área de TI - Integrante Técnico - Fiscal Técnico - Gestor do Contrato*
Scrum Master CONTRATADA	- Preposto **
Equipe de Desenvolvimento	- Equipe do fornecedor contratado

* Papel do Gestor do Contrato poderá ser exercido tanto pelo Product Owner quanto pelo Scrum Master IPHAN ou ainda por qualquer outro servidor que atenda aos requisitos do inciso "V" do art. 2º da IN SLTI/MP nº 04/2010. (BRASIL, 2013)

** As atribuições do papel de Preposto – conforme definidas pelo inciso "VIII" do art. 2º da IN SLTI/MP nº 04/2010 – se assemelhando às do papel de *Scrum Mastaer* CONTRATADA, porém, estes papéis poderão ser exercidos por pessoas diferentes. (BRASIL, 2013)

3. Papéis Auxiliares:

   (a) Áreas de Negócio do IPHAN: surgem como os principais requisitantes de soluções de *software*, seus representantes frequentemente atuarão nos projetos exercendo o papel de "*Product Owner*";
   (b) Área Administrativa do IPHAN: responsável por conduzir processos administrativos críticos, tais como licitações e pagamento de fornecedores;
   (c) Área de Infraestrutura de TI do IPHAN: responsável pela gestão da infraestrutura tecnológica, por vezes sua atuação será necessária do processo de provimento de soluções de *software*;

(d) Papéis de gestão contratual: os contratos da área de TI devem ser obrigatoriamente geridos por uma equipe formada pelo gestor do contrato e pelos fiscais técnico, requisitante e administrativo;

(e) Empresa Contratada: pessoa jurídica selecionada através do devido processo legal de licitação para prestar serviços ao órgão, segundo o modelo vigente na organização as fábricas são empresas contratadas (prestadores externos).

4. Fluxos do Processo.

   As demandas apresentadas à empresa contratada para executar os serviços de desenvolvimento de sistemas no âmbito do IPHAN são fundamentadas no *framework Scrum*, sempre respeitando os seguintes tipos:

   (a) Tipo I: MIDAS (Processo de Gestão de Demandas de Desenvolvimento Ágil de *Softwares*);

   (b) Tipo II: *Sprint* (subprocesso).

   Para melhor classificação dos tipos de projetos contratados, os quais são de desenvolvimento de *software* novo, Manutenção Evolutiva, Manutenção Corretiva Refatoração e documentação de *software* Legados, o MIDAS prevê o seguinte fluxo:

Tabela 4.2: Fluxos de desenvolvimento segundo os tipos de demanda, Fonte: (BRASIL, 2013)[17]

Tipo de Projeto	Fluxo MIDAS	Observações
Sistema Novo	Tipo I	
Manutenção Evolutiva	Tipo I	
Manutenção Corretiva	Não se aplica	A manutenção corretiva, mesmo sendo de grande complexidade, será iniciada por uma Ordem de Serviço especifica. A depender da urgência do caso e da criticidade do sistema, o prazo para execução do serviço será estipulado em contrato. A definição da criticidade de um sistema cabe única e estritamente ao IPHAN
Refatoração	Tipo I ou Tipo II	A decisão por tipo de fluxo dependerá dos níveis quantitativo e qualitativo das alterações demandadas. Cabe unicamente ao IPHAN a decisão do tipo de fluxo.
Documentação de Sistema Legado	Tipo I ou Tipo II	A decisão por tipo de fluxo dependerá da complexidade do sistema tratado e da pré-existência de algum tipo de documentação. Cabe unicamente ao IPHAN a decisão do tipo de fluxo.

Dentro de uma visão macro, o processo do MIDAS, o fluxo de gestão de demandas de desenvolvimento ágil de *software*, é fundamentado nas diretrizes do ciclo PDCA e da metodologia *SCRUM*, com o objetivo de permitir que os projetos sejam executados de forma incremental, com entregas frequentes e o progresso seja medido continuamente, demonstrados no fluxo apresentado no diagrama da Figura 4.9 (BRASIL, 2013)[17].

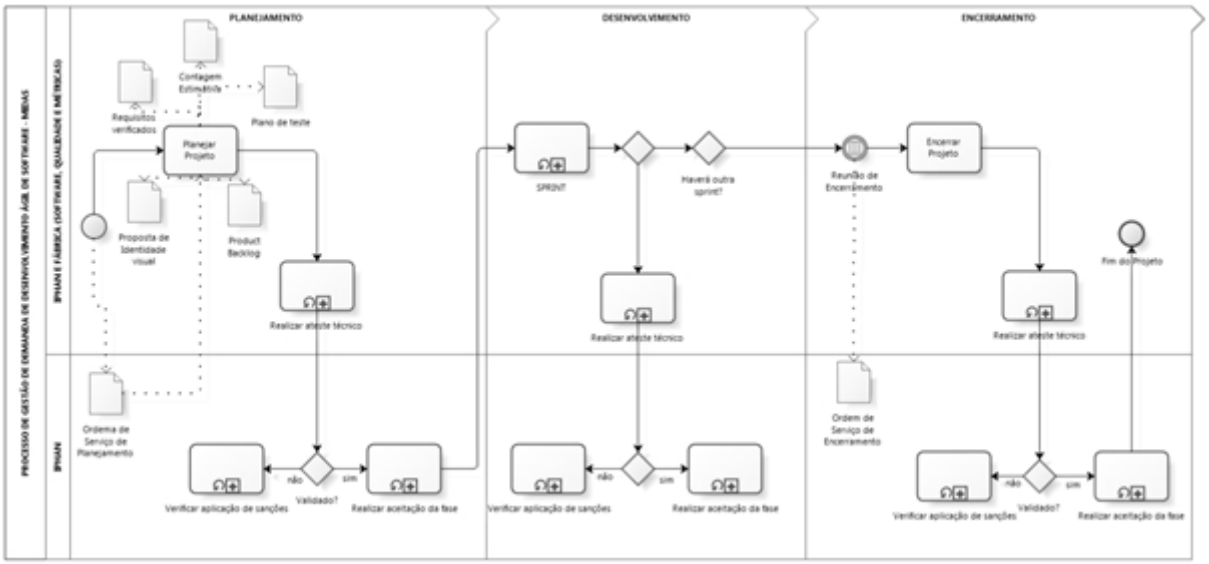

Figura 4.9: Processo MIDAS - Macro Fluxo, Fonte: (BRASIL, 2013)[17]

4.3.2 Banco Centro do Brasil (BACEN)

O Banco Central do Brasil (BACEN) é uma autarquia Federal vinculada ao Ministério da Fazenda constituída por intermédio da Lei n. 4.595, de 31 de dezembro de 1964, tendo como principal executor das orientações do Conselho Monetário Nacional e responsável por garantir o poder de compra da moeda nacional com os principais objetivos de zelar pela adequada liquidez da economia; manter as reservas internacionais em nível adequado; estimular a formação de poupança; zelar pela estabilidade e promover o permanente aperfeiçoamento do sistema financeiro nacional. Dentro de sua estrutura organizacional, o BACEN conta com o apoio na execução de suas atribuições institucionais com o Departamento de Tecnologia da Informação (DEINF), departamento funcionalmente subordinado a Diretoria de Administração (DIRAD).

Como método de desenvolvimento e manutenção de *software* corporativo, o BACEN por intermédio da DEINF conta com o Processo Ágil de Desenvolvimento do Banco Central do Brasil (PDS-BC Ágil) como guia para a construção do produto de software correto e com qualidade (BRASIL, 2014)[20].

Para os casos de subcontratação de desenvolvimento e manutenção, o PDS-BC Ágil tem como escopo o controle das mudanças significativas, ou mesmo de qualquer natureza, com aplicação de atividades realizadas internamente e a todos os produtos gerados pela contratada, tendo como resultados esperados:

1. O produto de *software* é desenvolvido, alinhado às necessidades do negócio;
2. O produto de *software* entrega o maior valor possível para o negócio com qualidade;
3. O processo de trabalho da equipe do projeto é adaptado continuamente;
4. As entregas de *software* são frequentes;
5. Os riscos são identificados e tratados;
6. A equipe do projeto trabalha colaborativamente;
7. Os envolvidos têm visibilidade do progresso do projeto;
8. A documentação do *software* é automatizada e mantém-se sempre atualizada;
9. A documentação para transferência do conhecimento é gerada.

Diante de um processo metodológico definido à entrada de produtos desenvolvidos, onde a qualidade desses produtos e o alinhamento qualitativo estão sendo impostos por uma demanda de diretriz e de uma coerência regulamentada por instruções governamentais, cada empresa necessita entender claramente cada etapa dos processos administrativos e produtivos, a fim de atender a uma demanda cada vez mais exigente de consumo.

Além do exposto, à medida que aumenta a qualidade dos projetos desenvolvidos fundamentados no PDS-BC Ágil e soluções de TI no mercado corporativo, surge a necessidade de padronização das ações necessárias para o desenvolvimento ágil dentro do BACEN. A partir dessa necessidade, o BC buscou descrever um processo de desenvolvimento adequado as necessidades de negócio do órgão e as normativas impostas pelo governo no processo de desenvolvimento de *software* por intermédio de contratação externa em um formato padronizado e inteligível tanto por requisitante quanto por empresas executoras.

Nesse contexto, o PDS-BC Ágil, demonstrado no diagrama da Figura 4.10, conta com 17 atividades fragmentadas em ações que estabelecem a visão do produto; planejam as liberações e o *backlog* do produto; planejam a liberação atual; executam a iteração; apresentam os resultados das iterações; realizam a retrospectiva da iteração; testam a qualidade do produto; homologam o produto e publicam o *software*.

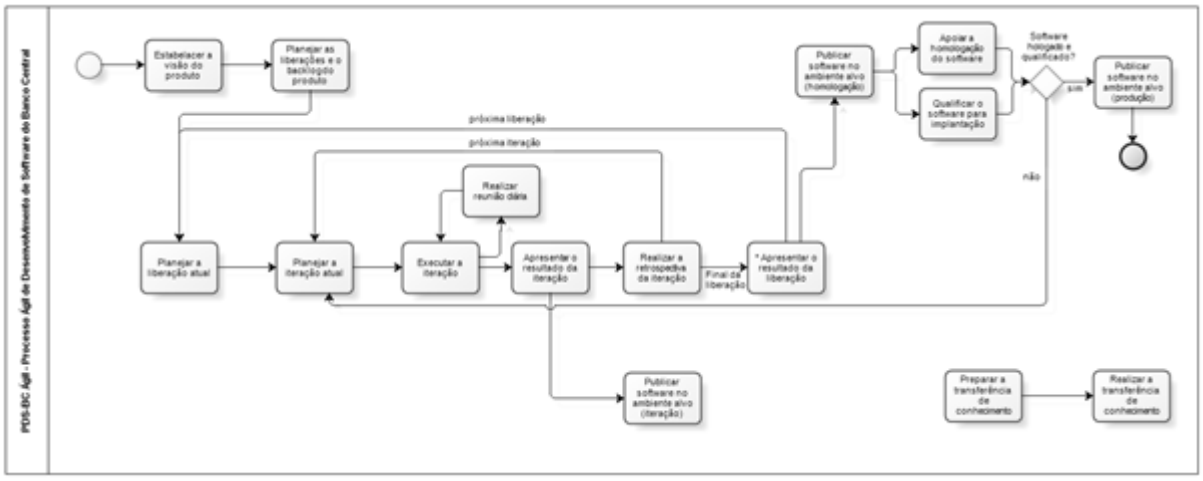

Figura 4.10: Processo Ágil de Desenvolvimento de *Software* do Banco Central, Fonte: (BRASIL, 2014)[20]

No âmbito do PDS-BC Ágil, papéis e características do processo de desenvolvimento e manutenção de *software* contratos fundamentados na metodologia ágil são (BRASIL, 2014)[20]

1. *Product Owner* (PO): indivíduo que representa os interesses dos *stakeholders* de negócio; deve ter conhecimento suficiente do negócio para responder aos questionamentos da equipe de desenvolvimento.

   (a) Características:

      i. Conhecer o processo de negócio e seus objetivos;
      ii. Gerenciar as expectativas dos *stakeholders*;
      iii. Comunicar e negociar;
      iv. Trabalhar em equipe;
      v. Aceitar mudanças;
      vi. Ter iniciativa.

   (b) *Coach*: indivíduo experiente no processo ágil que zela pela sua correta execução e ajuda os desenvolvedores a implementar *software* de alta qualidade alinhado às necessidades dos *stakeholders*; é um mentor que trabalha lado a lado com os outros membros da equipe de desenvolvimento em suas tarefas, disseminando as práticas do processo. É o líder técnico da equipe e deve pertencer à Divisão de Relacionamento com o Negócio responsável pelo projeto.

   (c) Características:

      i. Conhecer o processo e as práticas de desenvolvimento ágil;

ii. Liderar;
iii. Disseminar o conhecimento;
iv. Comunicar e negociar;
v. Trabalhar em equipe.

(d) Gerente de projeto: indivíduo que resolve os impedimentos da equipe de desenvolvimento executa as funções administrativas e acompanha o andamento do projeto.

(e) Características:

i. Conhecer o processo ágil de desenvolvimento de *software*;
ii. Comunicar e negociar;
iii. Gerenciar conflitos e expectativas;
iv. Liderar;
v. Trabalhar em equipe;
vi. Ter iniciativa.

(f) Desenvolvedor: indivíduo que constrói o produto de *software*.

(g) Características:

i. Conhecer o processo e as práticas de desenvolvimento ágil;
ii. Conhecer os padrões corporativos do BACEN;
iii. Comunicar e negociar;
iv. Trabalhar em equipe;
v. Aceitar mudanças;

2. Manual de usuário: orientação ao usuário sobre a forma de utilização do *software*, que pode se apresentar na forma de livro, ajuda do sistema, instalador etc.

3. Material de treinamento: conjunto de documentos destinados ao repasse de conhecimento aos usuários do sistema. Pode ser composto por apostilas, apresentações, folders, encartes ou outros considerados adequados a cada caso.

4. Melhoria no processo: ações de melhoria contínua no processo por meio de incorporação, supressão ou adaptação de práticas.

5. Necessidades do negócio: lista de problemas a serem resolvidos ou oportunidades a serem exploradas.

6. Plano de liberações: lista com a previsão das liberações do projeto que contém cronograma e o objetivo de todas as liberações do produto.

7. Plano de transferência de conhecimento: plano que registra as ações de transferência de conhecimentos previstas, cronograma de realização e publico alvo de cada ação.

8. Produto de *software*: conjunto de programas de computador (código fonte, testes automatizados, *scripts* de banco de dados, testes e operação), procedimentos, documentação e dados associados.

4.3.3 Tribunal Superior Eleitoral (TSE)

O Tribunal Superior Eleitoral (TSE) é um órgão do poder judiciário com abrangência Nacional, constituída pela restruturação da Justiça Eleitoral ocorrida em 1945 por intermédio do Decreto-Lei n. 7.586/1945, de 1 de junho do corrente ano, responsável por toda gestão executiva, operacional e normativo do processo eleitoral.

Em sua estrutura organizacional, conta com o apoio da Secretaria de Tecnologia da Informação (STI) nas ações de governança instituída por normas e procedimentos metodológicos, com a missão de "Prover o efetivo suporte tecnológico à Justiça Eleitoral com qualidade e segurança para o fortalecimento da democracia" (BRASIL, 2014)[24].

Com essa visão, o Escritório de Processos e Padrões (EPP) iniciou em 2011, um projeto para prover melhorias no desenvolvimento de *software* do TSE, tendo como premissa a identificação das questões de grande impacto que, na visão das equipes, dificultam ou impedem o seguimento de uma metodologia que as auxilie a controlar e monitorar o processo de desenvolvimento, minimizando os riscos, que resultou no Método de Desenvolvimento com Práticas Ágeis (MAgil) (BRASIL, 2014)[24].

O MAgil (2014)[24] tem como proposta agregar ao dia a dia das equipes as melhores práticas do desenvolvimento ágil, sendo recomendado a projetos nos quais se é possível atuar com agilidade e pouco formalismo e cujo foco seja a produção e entrega de funcionalidades do sistema de forma incremental. É considerado no TSE como uma "caixa de ferramentas" que pode ser utilizada conforme contexto e necessidade do projeto, na qual encontram-se práticas e artefatos das metodologias, métodos e *frameworks* disponíveis no mercado, como o Processo Unificado, PMBOK, UML, XP e *Scrum*.

Como forma de padronização, os papéis envolvidos dentro da estrutura da STI no processo de desenvolvimento de *software*, são apresentados na Tabela 4.3.

Tabela 4.3: Papéis envolvidos no desenvolvimento do *software* do TSE, Fonte: (BRASIL, 2014)[24]

Papel	Responsabilidade
Gerente de Projetos	▪ Conduzir o planejamento e gerenciamento do projeto; ▪ Coordenar as interações com os principais envolvidos; ▪ Manter a equipe de projeto focada em alcançar os objetivos.
Product Owner ou Cliente	▪ Representar o cliente final nas decisões sobre as necessidades e requisitos do sistema; ▪ Homologar os produtos e serviços entregues.
Scrum Master	▪ Garantir que as técnicas do *Scrum* sejam utilizadas no projeto, incentivando as práticas ágeis e removendo os impedimentos do time. O *Scrum Master* pode ser alguém do time *Scrum* (ou não), e, no TSE, o gerente de projeto pode assumir esse papel.
Analista de Conformidade	▪ Avaliar os processos e produtos de trabalho do projeto em relação à conformidade com descrição de processos, políticas, padrões e procedimentos aplicáveis à STI.
Apoiador do Projeto	▪ Apoiar a implantação de uma metodologia gerencial integrada ao processo de desenvolvimento que permita determinar, manter e evoluir processos e ferramentas de gestão de projetos; ▪ Colaborar com gerenciamento e o planejamento dos projetos, por meio de esforço cooperativo e coordenado.
Equipe de Infraestrutura	▪ Realizar a implantação do produto no ambiente de homologação e produção.
Equipe de suporte	▪ Participar da transferência de conhecimento recebendo as informações sobre a utilização do sistema que auxiliarão a equipe no suporte aos usuários finais.
Especialista em Ponto de Função	▪ Realizar a medição funcional do projeto; ▪ Validar as contagens recebidas da contratada; ▪ Auxiliar a equipe na especificação de requisitos.
Fiscal de Contrato	▪ Autorizar a abertura e o fechamento das ordens de serviços, assim como aprovar os demais documentos contratuais; ▪ Aplicar as regras de não conformidade com o contrato.
Fiscal Técnico de TI	▪ Solicitar estimativa de serviço; ▪ Abrir e fechar ordens de serviço. * Estas atividades são normalmente exercidas pelo gerente de projeto.
Preposta da Fábrica de Software	▪ Participar, quando necessário, da formalização de aceitas provisórios ou definitivos das entregas dos

O processo de desenvolvimento ágil do TSE está previsto em 04 etapas, as quais são: Iniciação do Projeto, Produção da *Sprint*, Encerramento da *Sprint* e Encerramento do

Projeto, executados com etapas distintas por subprocessos que detalham sua execução, conforme demonstrado no diagrama da Figura 4.11.

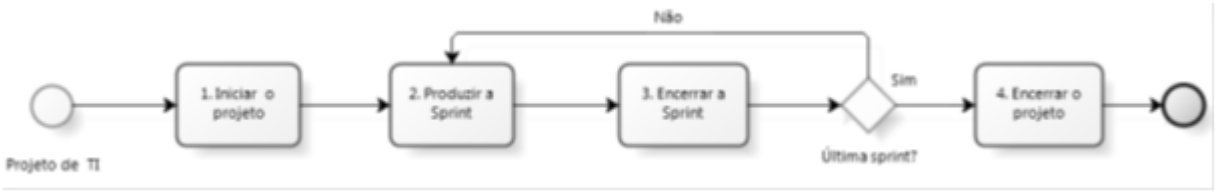

Figura 4.11: Macro processo de desenvolvimento ágil do TSE, Fonte: (BRASIL, 2014)[24]

No processo de Iniciação dos Projetos estão previstas as atividades de Definir o *Product Owner*, Definir a Visão do produto, Definir os itens do *Backlog* do Produto, Priorizar os itens do *Backlog* do Produto, Formar o *Time Scrum*, Estimar o Tamanho do Produto e Realizar o Planejamento do projeto, representado no diagrama da Figura 4.12.

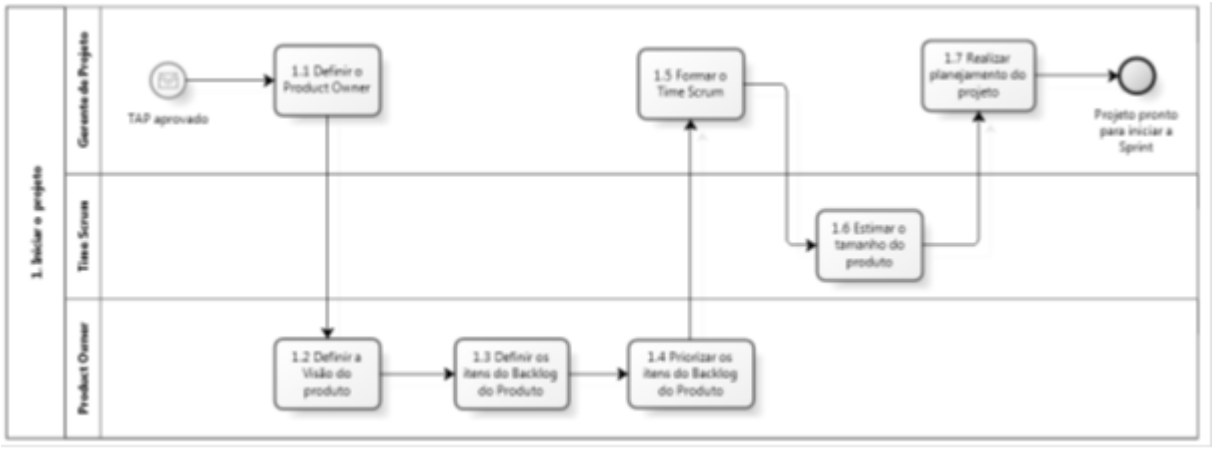

Figura 4.12: Processo de Iniciação do Projeto do TSE, Fonte: (BRASIL, 2014)[24]

Na produção da *Sprint* estão previstas as atividades de Definir escopo da *Sprint*, Planejar a Sprint e Executar a *Sprint*, conforme Figura 4.13.

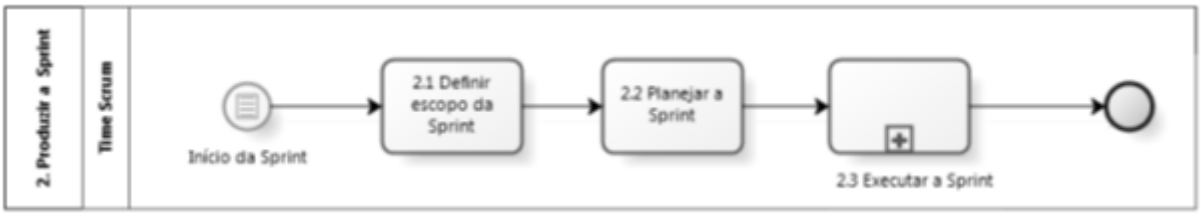

Figura 4.13: Processo de produção da *Sprint*, Fonte: (BRASIL, 2014)[24]

Como subprocesso de executar a *Sprint*, estão as atividades de Realizar reunião do dia, Elicitar os requisitos, Construir os itens da *Sprint*, Testar os requisitos construídos, Implantar versão no ambiente de homologação do TSE, Preparar a apresentação do *Sprint*,

Apresentar a *Sprint*, finalizando com Formalizar a entrega da *Sprint*, demonstrado no diagrama da Figura 4.14.

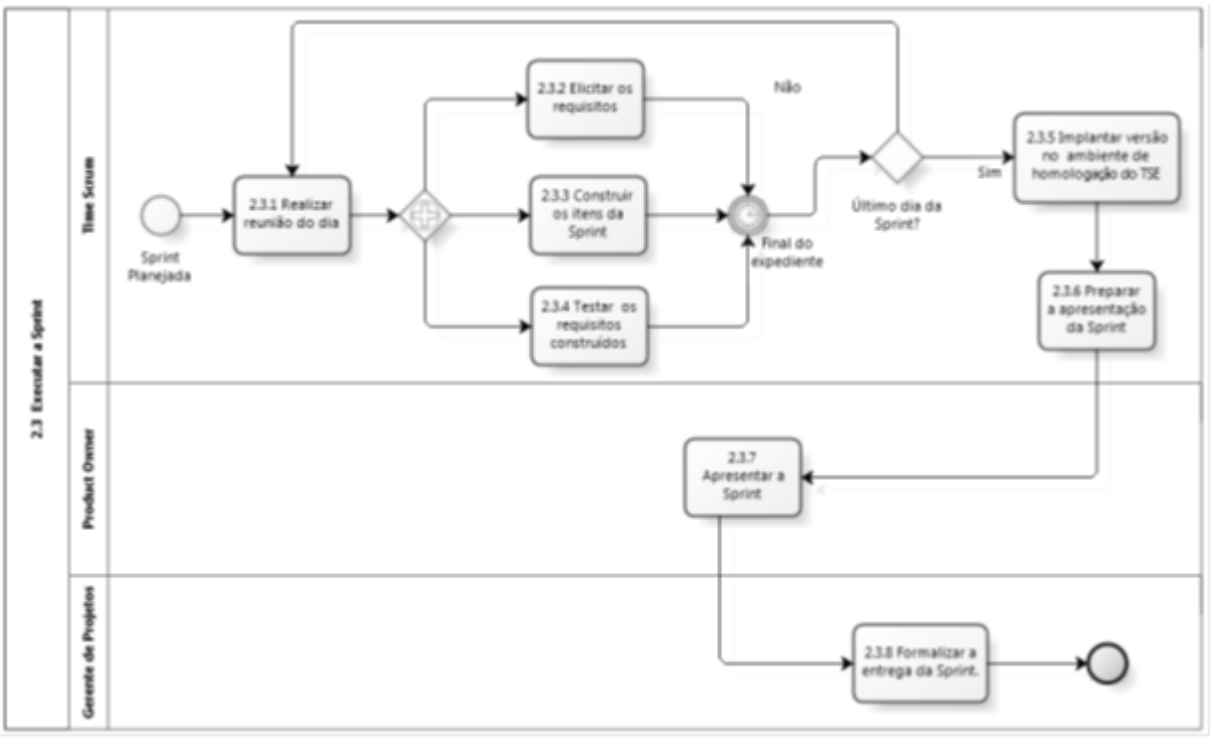

Figura 4.14: Subprocesso de Executar a *Sprint*, Fonte: (BRASIL, 2014)[24]

Continuando o processo de desenvolvimento de *software* ágil do TSE, no processo de Encerramento da *Sprint* estão previstas as atividades de Aprovar *Sprint*, Gerar laudo de garantia da qualidade, Rejeitar Sprint e Realizar reunião de retrospectiva da *Sprint*, conforme mostra a Figura 4.15.

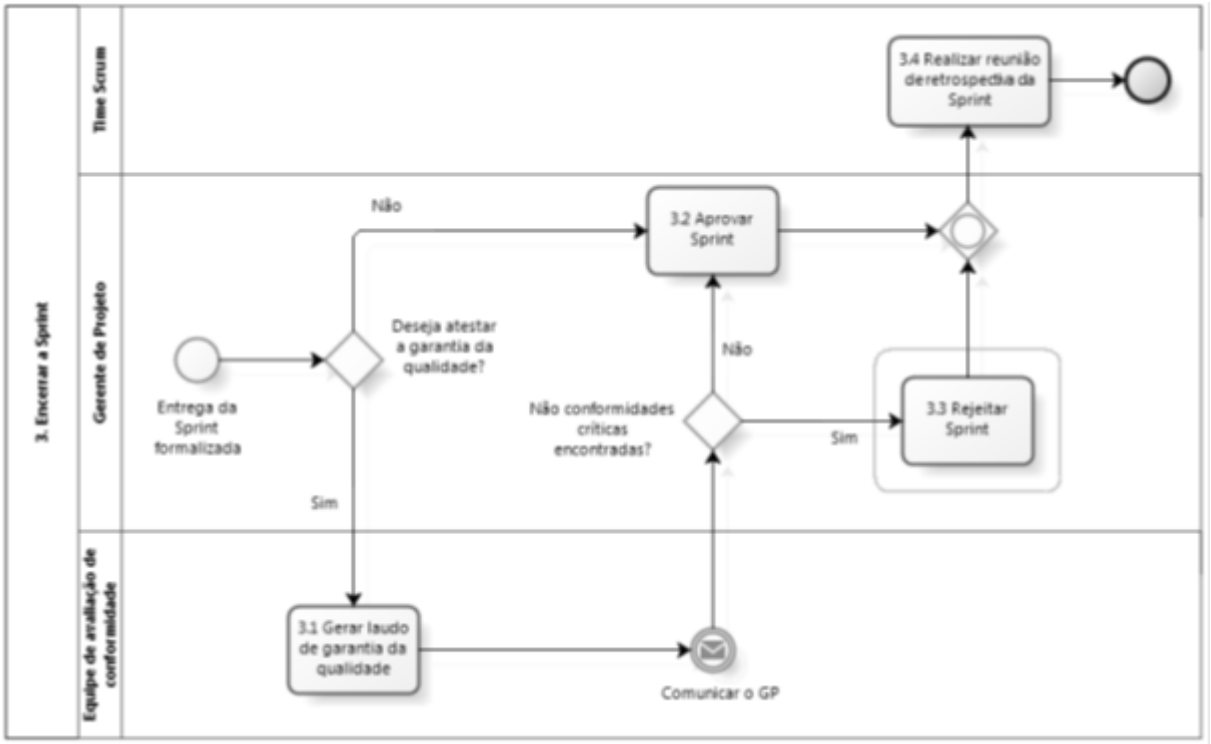

Figura 4.15: Processo de Encerramento da *Sprint*, Fonte: (BRASIL, 2014)[24]

Finalizando o processo de desenvolvimento ágil do TSE está o processo de encerrar Projeto com três atividades, as quais: Implantar versão na ambiente de produção do TSE, Transferir conhecimento e Encerrar o projeto, conforme o diagrama da Figura 4.16.

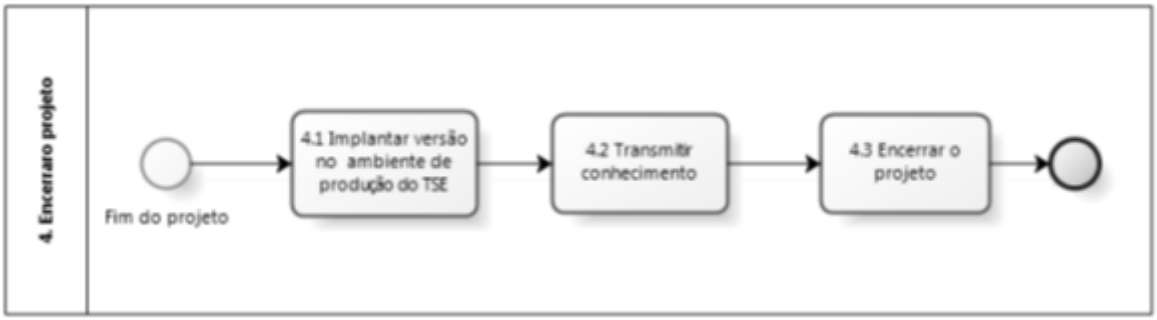

Figura 4.16: Porcesso de Encerramento do Projeto, Fonte: (BRASIL, 2014)[24]

A próxima seção visa complementar a visão processual do desenvolvimento de *software* dos diversos órgãos federais visitados, serão apresentadas as considerações gerais, com os resultados dos questionários aplicados.

4.4 Análise dos resultados (*"Benchmarks"*)

Neste tópico serão apresentados os resultados obtidos na aplicação dos questionários e entrevistas realizadas nos órgãos federais, utilizados na avaliação da metodologia proposta.

Conforme descrito na metodologia utilizada neste trabalho foram selecionados 3 órgãos para realização do *benchmarking*. O questionário foi encaminhado para 15 pessoas, no entanto, obteve-se resposta de apenas 10.

As entrevistas foram realizadas com coordenadores e gerentes, juntamente com suas respectivas equipes técnicas, de modo a se compreender o contexto a respeito do objetivo da pesquisa. As entrevistas com os agentes públicos foram realizadas por meio de agendamento prévio. O questionário foi aplicado por intermédio da ferramenta de apoio oferecido pelo Google Docs, inicialmente armazenado e posteriormente transformado em dado quantitativo para análise.

O questionário produzido foi formulado por questões fechadas e aplicadas aos agentes públicos que exercem as funções de coordenadores, gestores e técnicos, com atividades de gestão e execução de atividades de contratação e ou execução de desenvolvimento e manutenção de *software* organizacional, no período de março a agosto de 2014. A elaboração das questões e a seleção desses agentes procurou garantir o alinhamento com os objetivos da pesquisa, bem como da credibilidade dos dados coletados. Para tanto, foi considerado os órgãos visitados pelo Tribunal de Contas da União (TCU), relacionados no Acórdão n. 2314-33/2013 (BRASIL, 2013b)[18], o qual realizou o levantamento de auditoria de conhecimento acerco da utilização de métodos ágeis nas contratações para o desenvolvimento de *software* pela Administração Pública Federal (APF).

Na elaboração das questões foram considerados os critérios de correlação com os objetivos da pesquisa, os quais se destacam o desenvolvimento de *software* com uso da metodologia ágil pela APF de forma objetiva e agrupadas com as respectivas abordagens, com o objetivo de facilitar o entendimento dos entrevistados, fundamentado nos conceitos estudados na revisão bibliográfica. Complementar a aplicação do questionário foi disponibilizado pelos órgãos, de forma documental, suas metodologias de desenvolvimento e manutenção de sistemas de *software* coorporativos fundamentado no método ágil, como mecanismo de entendimento dos processos e atividades executadas pelos agentes internos e ou em regime de fábrica de *software*, ou seja, empresa contratadas para execução do desenvolvimento e manutenção dos sistemas.

A amostra do objeto da pesquisa estava limitada às APF que utilizam efetivamente o método ágil como forma de desenvolvimento e manutenção de *software* internamente e ou por intermédio de contratação de serviços de TI, em regime terceirizada. O campo de pesquisa se restringiu a três órgãos devido o pequeno número de administrações que

utilizam o método ágil de forma processual alinhada aos objetivos estratégicos, bem como das normas regulamentares vigentes do Governo Federal.

Adicionalmente no tópico 4.4.1 é apresentada à análise desenvolvida com o uso da ferramenta do *Failure Mode and Effect Analysis* (FMEA) no sentido de priorizar os riscos do Acórdão do TCU segundo o entendimento das APFs e dos gerentes de projeto do CDS.

As questões formuladas nesta pesquisa obedecem aos objetivos da gestão de desenvolvimento dos diversos órgãos visitados e garante a confidencialidade dos dados com a preservação da identidade dos respondentes.

Inicialmente, os dados da pesquisa são correlacionados com as questões individuais com relação à experiência dos entrevistados no uso de metodologias ágeis (Figura 4.17). Constata-se que 70% possui entre um e dois anos de experiência; 20% tem entre três e cinco anos de experiência; e 10% entre seis meses e um ano de experiência.

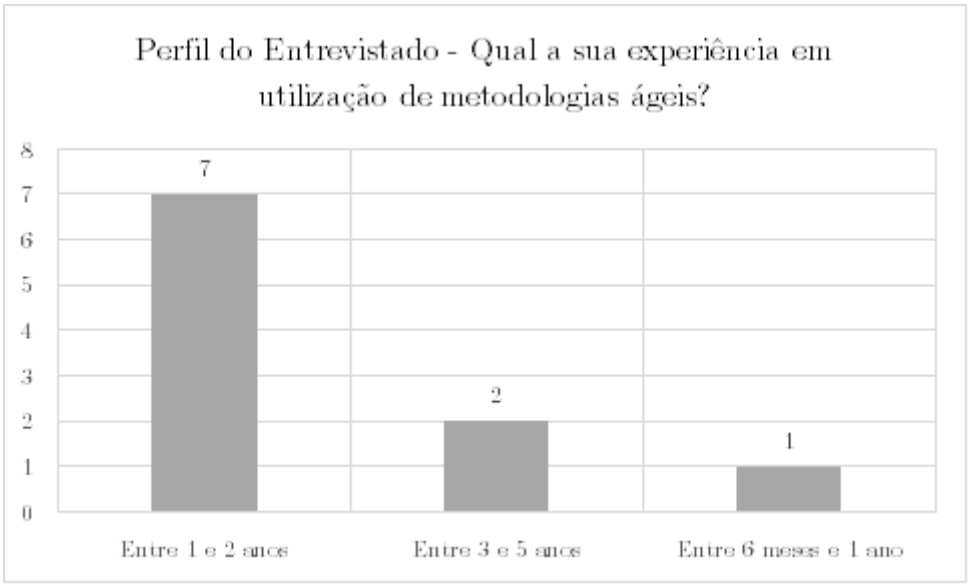

Figura 4.17: Gráfico de Perfil dos entrevistados - Cargo / Função

Com relação ao uso de métodos ágeis no desenvolvimento de *software* realizados internamente (Figura 4.18), observa-se que 20% dos entrevistados não utiliza métodos ágeis em seu desenvolvimento, 10% utiliza em menos de 25% de seus projetos, outros 10% utiliza entre 25 e 50% dos projetos e a grande maioria (60%) utiliza em pelo menos 75% dos projetos.

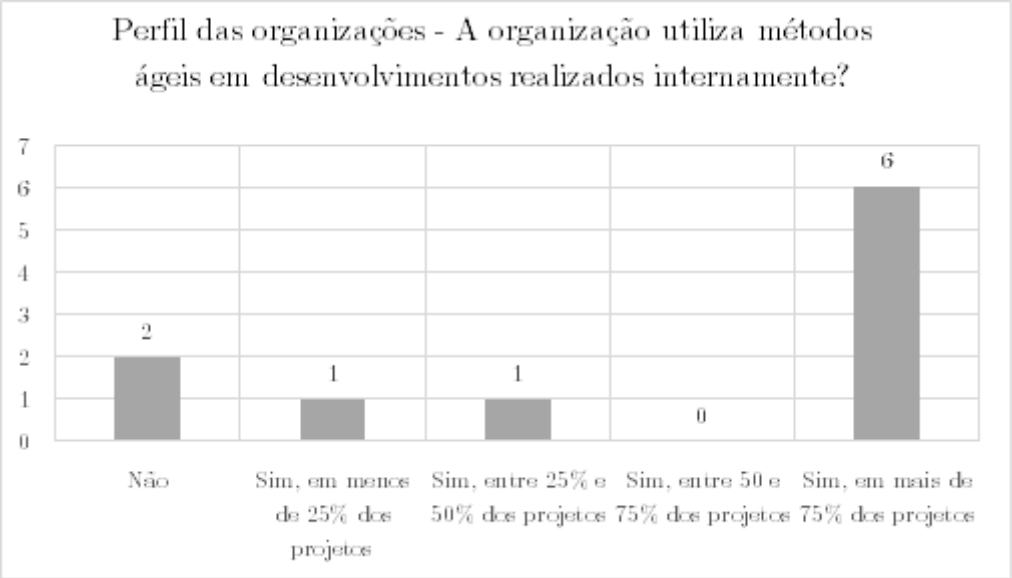

Figura 4.18: Gráfico de Perfil dos Entrevistado – Uso da metodologia ágil no desenvolvimento interno

Quando perguntado sobre a utilização de métodos ágeis para o acompanhamento e gerência dos contratos de desenvolvimento com empresas terceirizadas (Figura 4.19), 60% respondeu que 'sim, utilizam métodos ágeis' e 40% 'não'.

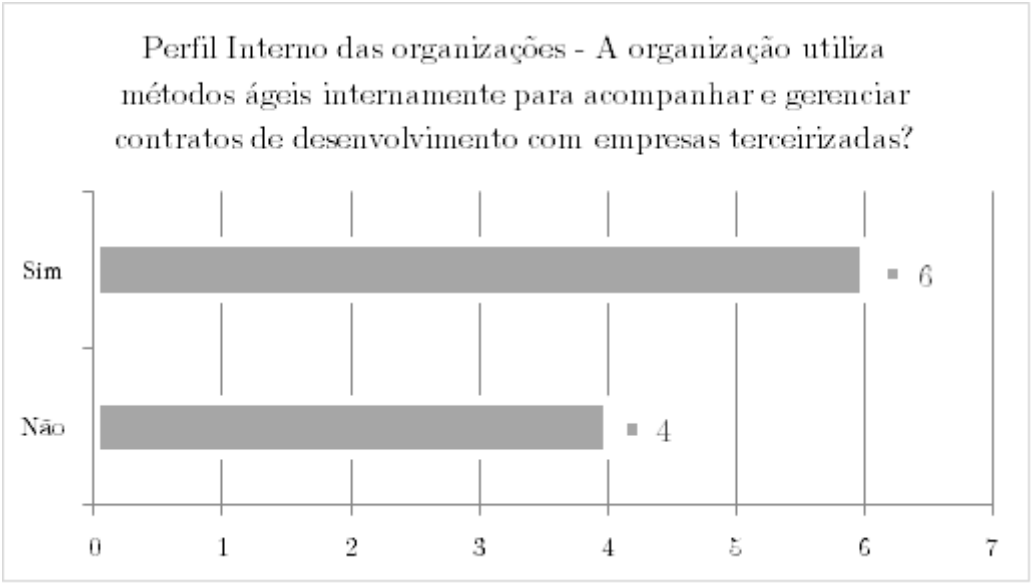

Figura 4.19: Gráfico de Perfil interno da organização no uso de metodologia ágil na gestão do contrato

Quando perguntados sobre o número de projetos ágeis realizados (ou em desenvolvimento) por meio de empresas terceirizadas (Figura 4.20), observa-se que metade das organizações não mantém nenhum contrato. A outra metade possui de um a quatro contratos, sendo 10% um contrato, outros 10% dois contratos, 20% três contratos e 10% quatro contratos.

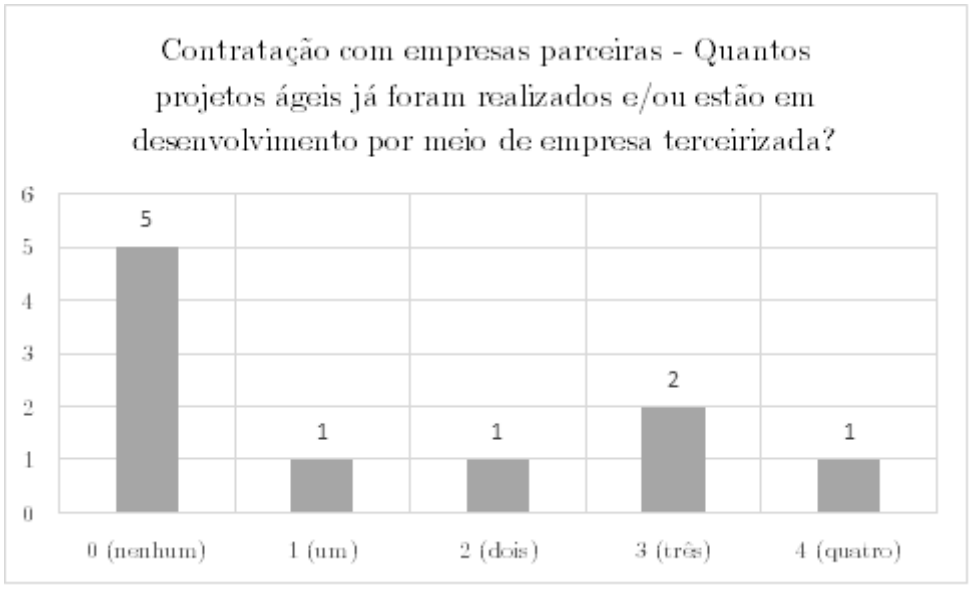

Figura 4.20: Gráfico de Quantitativo de contratação com o uso do método ágil

Observa-se que na maioria das organizações avaliadas (70%) a duração média dos contratos ou o desenvolvimento interno utilizando métodos ágeis dura até 2 anos (Figura 4.21), sendo 30% até 1 ano e 40% entre 1 e 2 anos. Organizações com tempos acima de dois anos somam 20%.

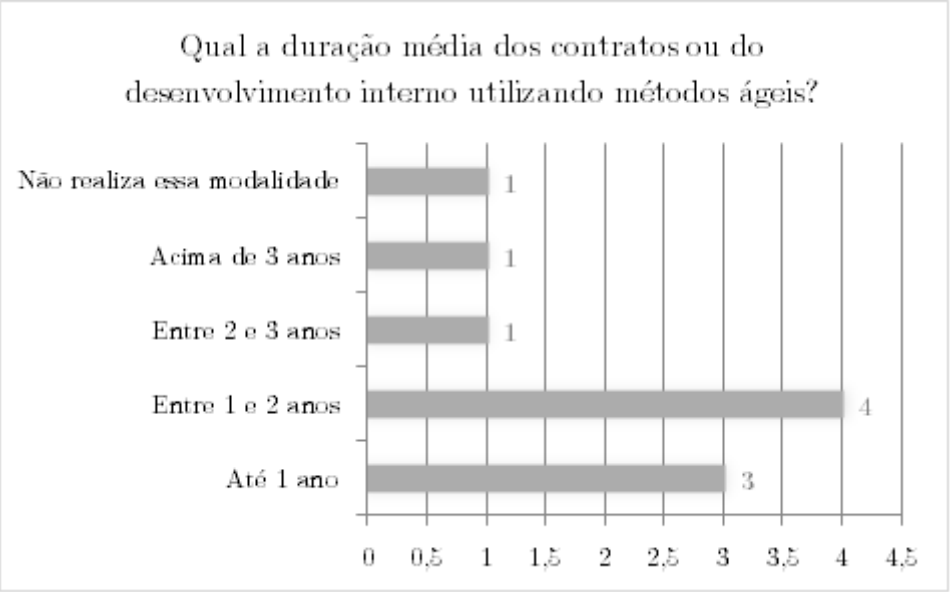

Figura 4.21: Gráfico de Tempo médio de duração dos contratos de desenvolvimento de software com uso de métodos ágeis

Sobre as metodologias ágeis utilizadas na execução dos projetos (Figura 4.22), 53% das organizações afirmaram utilizar a metodologia *Scrum*, 26% Kanban, 11% *eXtreme Program* (XP) e 10% Desenvolvimento orientado a testes. Embora bastante difundidos, o "desenvolvimento enxuto" e "Cristal" não foi citado por nenhuma organização participante da pesquisa.

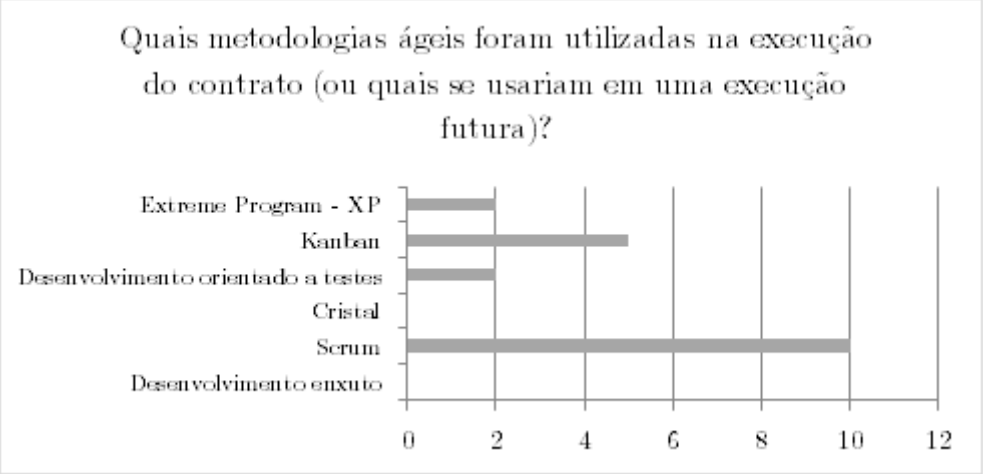

Figura 4.22: Gráfico de Tipos de métodos ágeis utilizados na execução do contrato

Quando perguntados sobre o nível de dificuldade na gestão de um contrato de desenvolvimento de sistemas com a utilização de métodos ágeis (Figura 4.23), 70% das organizações se mostraram neutras, ou seja, não acharam nem fácil nem difícil. 10% afirmou ter achado a gestão fácil, 10% difícil e outros 10% muito difícil.

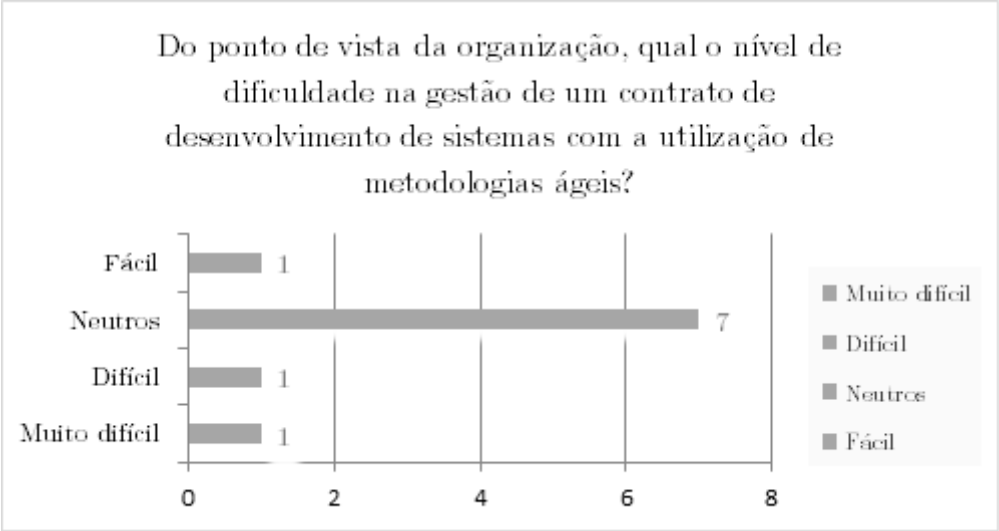

Figura 4.23: Gráfico de Grau de dificuldade no uso de método ágil na gestão de contrato com desenvolvimento ágil

Quando questionados sobre o grau de precisão da previsão de custo total nos momentos iniciais do projeto com métodos ágeis (Figura 4.24), 20% avaliaram em uma escala de 1 a 5 (onde 1 é pouco preciso e 5 muito preciso) que a precisão é 1. Outros 40% avaliaram com grau 2 e 40% como grau 3. Os graus 4 e 5, que retratam uma precisão maior não foram citadas.

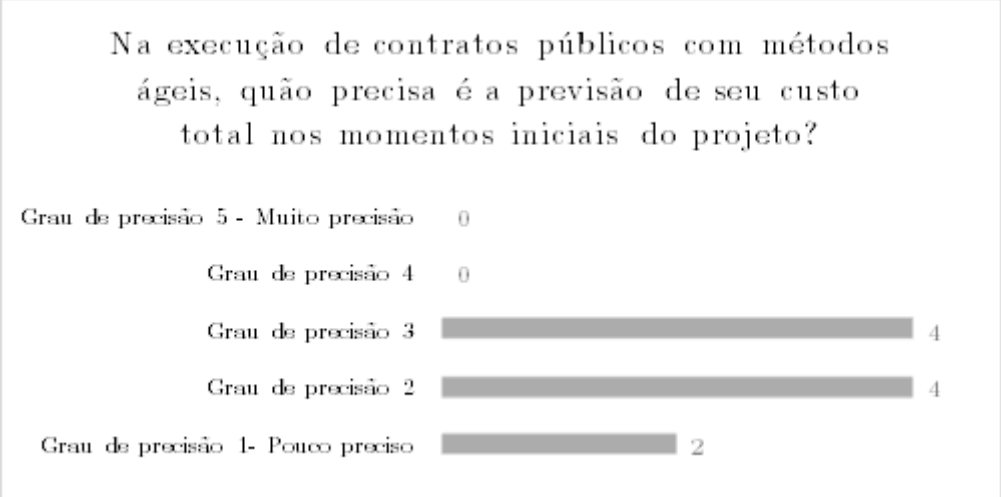

Figura 4.24: Gráfico de Grau de precisão do custo total do projeto de desenvolvimento ágil

Ainda com base na mesma escala, onde 1 revela pouca precisão e 5 precisão absoluta, observa-se que 10% das organizações classificam a precisão como grau 1, 20% como grau 2, 50% como grau 3 e outros 20% como grau 4 (Figura 4.25).

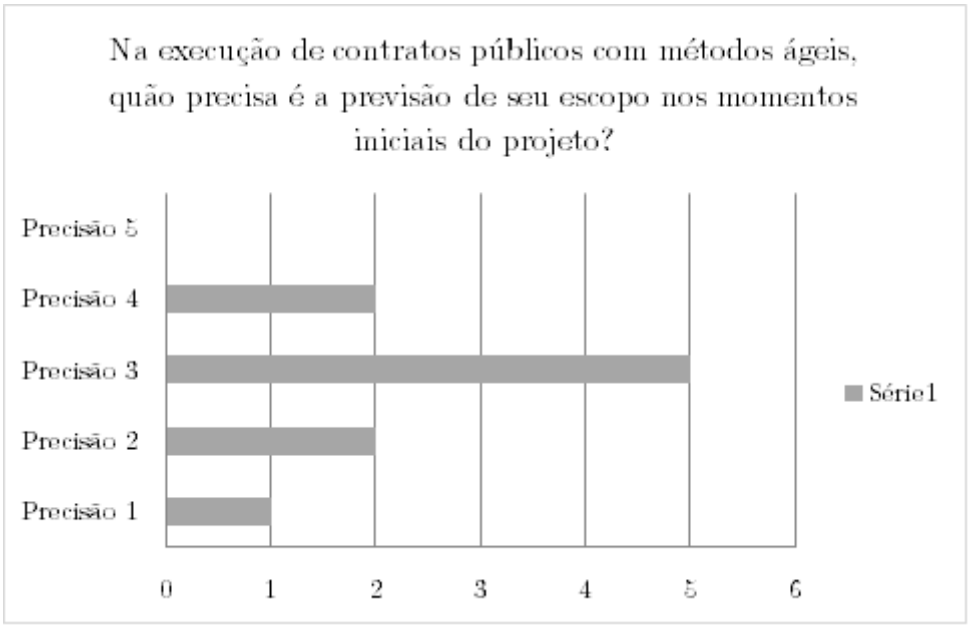

Figura 4.25: Gráfico de Grau de precisão do escopo do projeto no uso da metodologia ágil

Em relação à precisão da previsão de cronogramas em momentos iniciais do projeto (Figura 4.26), observa-se que as organizações entrevistadas tendem a avaliar essa previsão como pouco precisa. De acordo com a escala, 10% apontam grau de precisão 1, 40% como grau de precisão 2, 30% como precisão 3 e 20% como precisão 4, onde o grau 1 representa pouca precisão e 5 muita precisão.

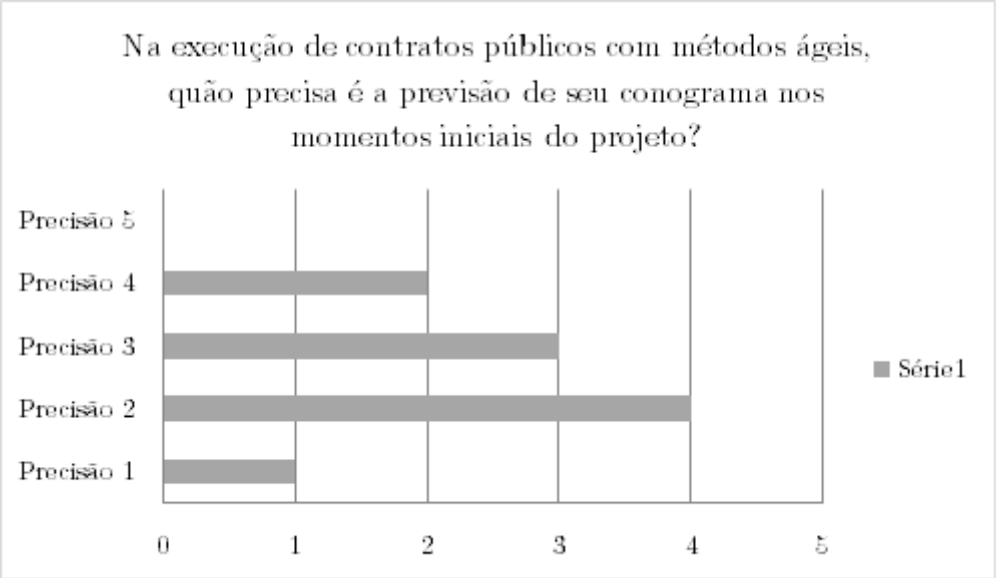

Figura 4.26: Gráfico de Grau de previsão do cronograma de inicial do projeto de desenvolvimento

Quando questionados sobre possíveis impactos à qualidade do desenvolvimento ao utilizar métodos ágeis (Figura 4.27), 90% das organizações respondeu que sim, ou seja, que acreditam que o uso de metodologias ágeis trazem impactos à qualidade e apenas 10% respondeu que não.

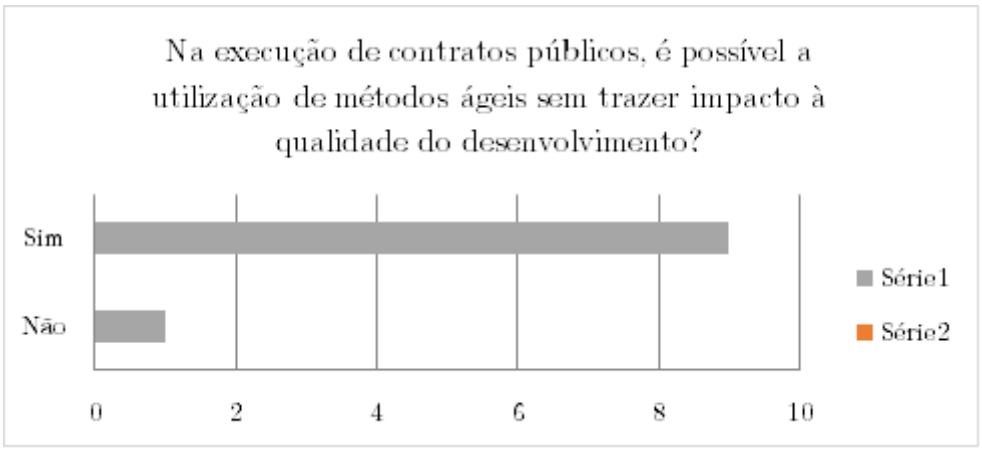

Figura 4.27: Gráfico de Análise de qualidade do desenvolvimento fundamentado no ágil

Quando perguntadas sobre como cada variável pode interferir na aplicação de métodos ágeis com empresas terceirizadas (Figura 4.28), em relação à variável 'tempo', 30% das organizações acreditam não interferir e 50% acreditam que interfere favoravelmente. Em relação aos 'custos', para 50% das organizações entrevistadas não há interferência, 30% interfere desfavoravelmente e 20% interfere favoravelmente. Em relação à variável 'riscos', observa-se um crescente consciência sobre a influência dessa variável, onde 10% acredita

ser desfavorável, 20% acredita não interferir, 30% acredita interferir favoravelmente e 40% acredita interferir muito favoravelmente. Coincidentemente as variáveis 'risco' e 'comunicações' apresentaram as mesmas respostas, onde chamando atenção o fato dessas duas variáveis serem relevantes para 70% das organizações entrevistadas, revelando uma preocupação e consciência da importância dessa variável nos projetos. A variável 'qualidade' também se destaca por 60% das organizações acreditarem que existe uma interferência favorável. Por fim, em relação à variável 'aquisições', 60% das organizações entrevistadas acreditam não interferir e 30% acredita interferir favoravelmente.

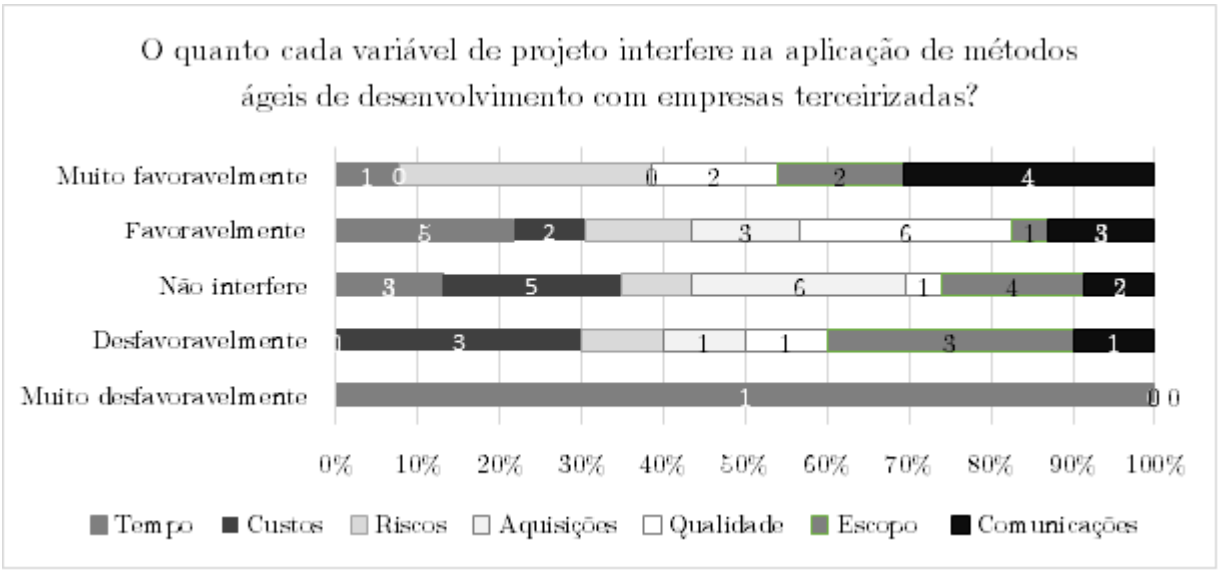

Figura 4.28: Variável de interferência no uso de métodos ágeis no contrato

As organizações entrevistadas se manifestaram ainda sobre o grau de limitação de restrições como cronograma (Figura 4.29), qualidade, escopo e preço no uso de práticas ágeis. Pôde-se apreender que em relação ao cronograma fixo, 30% das organizações acredita existir um baixo grau de limitação e outros 30% um médio grau de limitação. Sobre a alta qualidade constante, 50% acredita não haver limitação, outros 20% acreditam em uma limitação baixa e 30% uma limitação média. Em relação ao Escopo fixo, 40% diz não haver limitação, 10% limitação baixa, 30% média e 20% um alto grau de limitação. Por fim, preço fixo não é motivo de preocupação para 40% das organizações entrevistadas. Ainda 40% diz que o preço fixo representa um baixo grau de limitação e 20% um médio grau de limitação. De uma forma geral, destaca-se o entendimento de que não há limitação em relação da alta qualidade para metade das organizações e que o escopo fixo representa um alto grau de limitação para 20% das organizações.

.

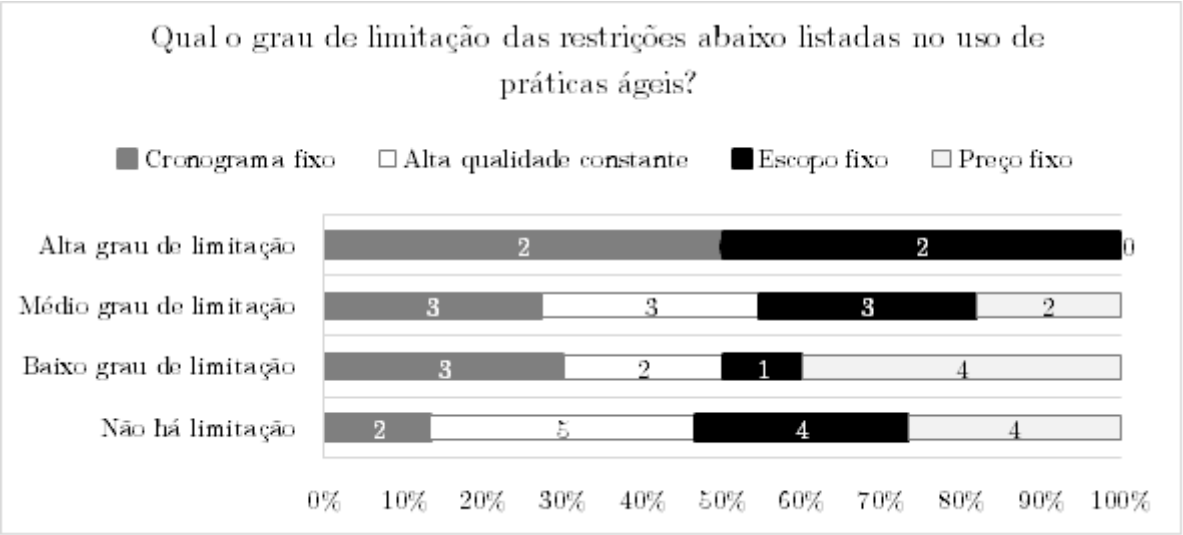

Figura 4.29: Gráfico de Grau de limitação das restrições no uso de práticas ágeis

As organizações foram perguntadas ainda sobre a maneira mais adequada para medição do esforço de desenvolvimento na execução de contratos públicos com métodos ágeis (Figura 4.30). Para 80% das organizações entrevistadas a 'Análise de ponto de função' é a maneira mais adequada, seguido por 'Unidade de Serviço Técnico' com 10%, e Pontos por História, também com 10%.

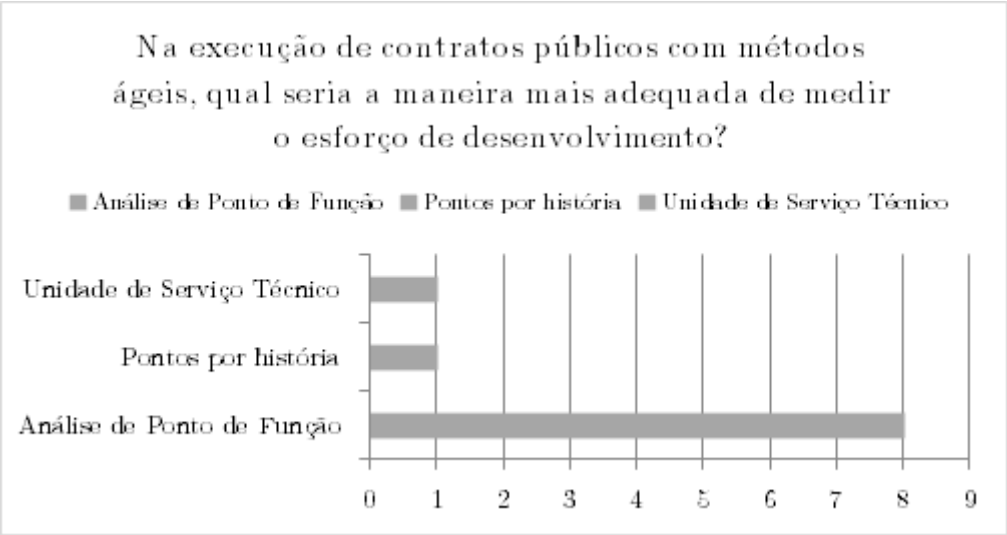

Figura 4.30: Gráfico de Métrica utilizada na aferição do esforço de desenvolvimento ágil

Sobre a organização das ordens de serviço (Figura 4.31), 90% das organizações apontaram o *"sprint"* como a forma mais adequado para a execução de contratos públicos com métodos ágeis e apenas 10% apontaram que a melhor forma seria por 'entrega'.

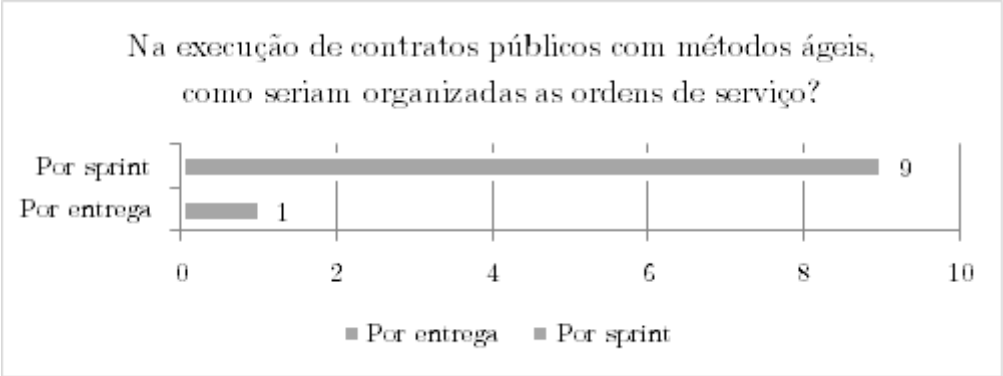

Figura 4.31: Gráfico de Método de demanda de execução do contrato ágil

As organização foram perguntadas sobre sua experiência na utilização de métodos ágeis em relação à necessidade de aditivo de valores em contratos (Figura 4.32), onde 60% revelou não realizar tal modalidade de contratação; 10% das organização não necessitaram de aditivo, 20% aditivou os contratos em cerca de 10% a 25% dos contratos e 10% em mais de 50% dos contratos.

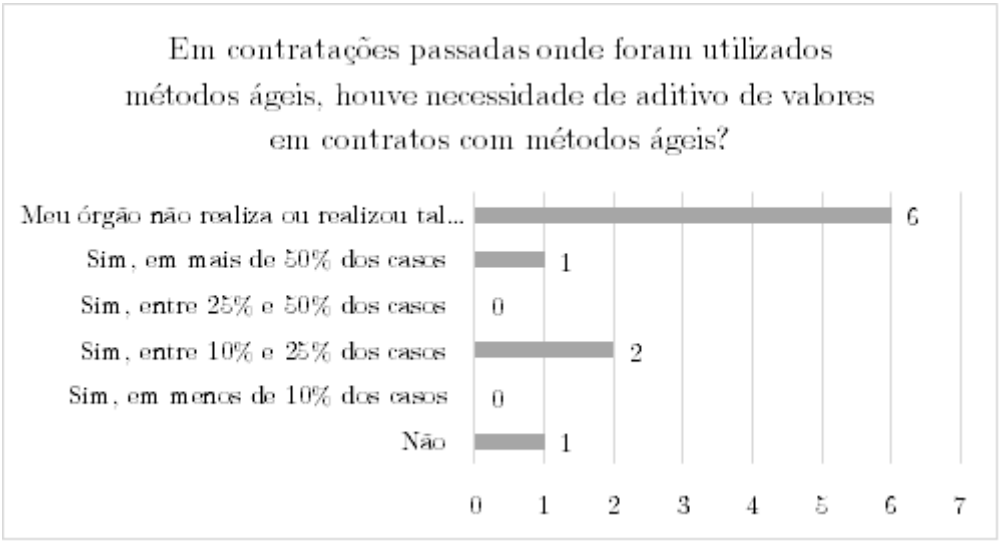

Figura 4.32: Gráfico de Análise de necessidade de aditivo de valor

As organizações também foram perguntadas sobre sua experiência na utilização de métodos ágeis em relação à necessidade de aditivo de tempo em contratos (Figura 4.33), onde 60% revelou não realizar tal modalidade de contratação; 10% das organizações não necessitaram de aditivo, 20% aditivou os contratos em cerca de 25% a 50% dos contratos e 10% em mais de 50% dos contratos.

.

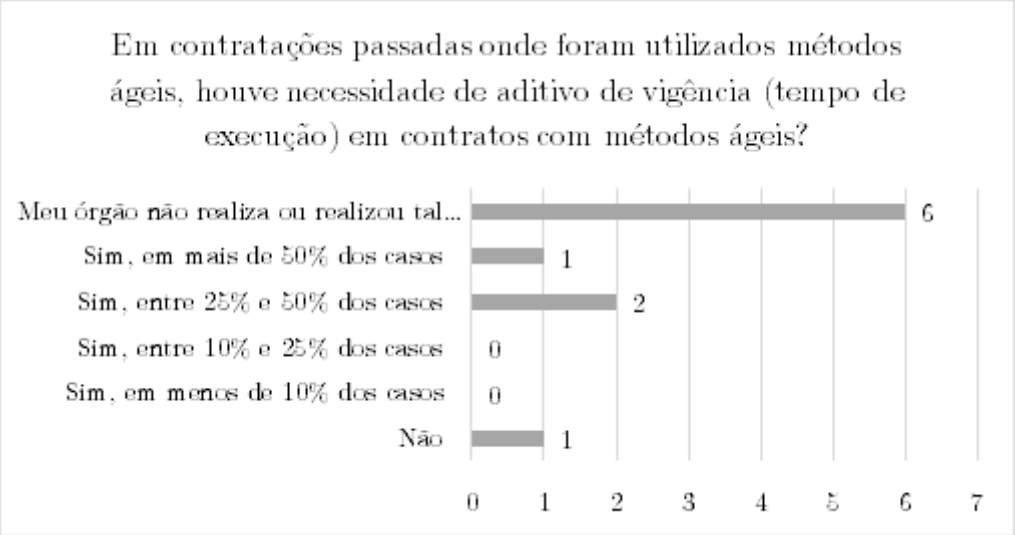

Figura 4.33: Gráfico de Análise de necessidade de aditivo de vigência de contrato

Em relação à pretensão dos órgãos entrevistados na execução de novos contratos de desenvolvimento de sistemas utilizando metodologias ágeis (Figura 4.34), 30% informou ter interesse em contrato com até 1 ano de duração, 10% contratos entre 1 e 2 anos de duração e 60% em contratos com duração acima de 2 anos.

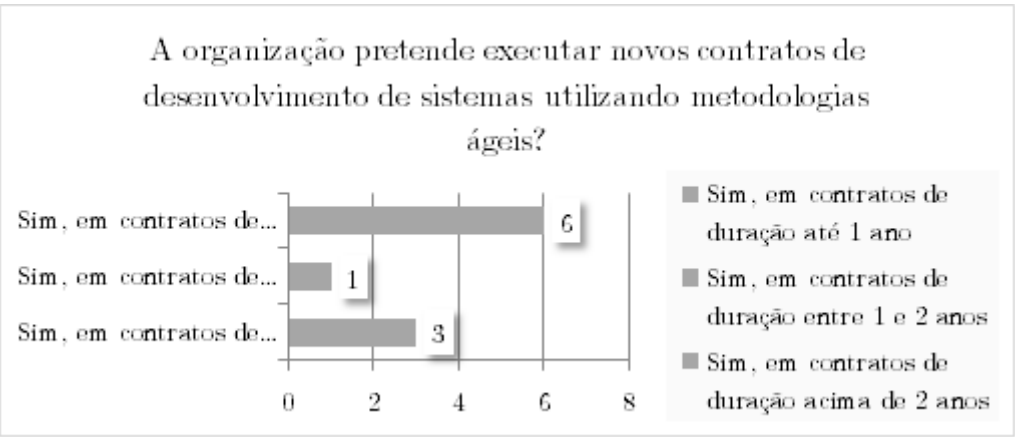

Figura 4.34: Gráfico de Análise de expectativa de novas contratações de desenvolvimento de *software* com método ágil

Por sua vez no próximo tópico é apresentada à análise dos riscos usando o FMEA, onde é possível observar que estes riscos estão concentrados ou no processo ou no produto.

4.4.1 Riscos no uso da metodologia ágil na Administração Pública Federal (TCU - Acórdão n. 2314-33/2013)

Neste tópico serão apresentados os resultados obtidos na aplicação dos questionários no IPHAN, BACEN, TSE e CDS. Estes foram utilizados na avaliação dos potencias riscos apontados na aplicação de métodos ágeis nas contratações para o desenvolvimento de *software* pela Administração Pública Federal (APF), segundo o Tribunal de Contas da União por intermédio do Acórdão n. 2314-33/2013, vide Figura 4.35 (BRASIL, 2013b)[18].

Figura 4.35: Riscos do uso do métodoágil, Fonte: (BRASIL, 2013b)[18]

Para Aguiar e Mello (2008)[5], o relacionamento por processo é um dos aspectos marcantes da gestão da qualidade. Os autores apontam que o princípio fundamental é estruturar e gerenciar as atividades por processos de forma sistemática e integrada, alinhada com as expectativas do cliente e da organização.

Segundo Lima *et al* (2006)[39], o FMEA consiste na técnica analítica utilizada com a finalidade de identificar as potencias falhas de um produto ou processo e suas causas e mecanismos associados que possam ser tratados.

O FMEA pode ser aplicado na identificação e avaliação de risco associado ao desempenho do processo, verificando quais serão os impactos no cliente de forma quantitativa,

priorizando os de maior potencial, determinando as ações a serem executadas, caso ocorra o risco. O FMEA deve ser revisto e atualizado sempre que os processos forem modificados, especialmente em casos onde o processo apresenta algum problema de qualidade, de maneira a assegurar que todas as medidas possíveis para evitar uma nova ocorrência sejam implementadas (AGUIAR e MELLO, 2008)[5].

O uso da técnica funciona também como ferramenta de gerenciamento de risco, por intermédio de uma planilha que auxilia na identificação de incertezas, suas causas e de seus efeitos sobre o produto ou processo, devendo ser elaborada de forma multidisciplinar, a qual busque responder as seguintes questões: i) Que tipo de risco/falha pode ocorrer? ii) Que partes do processo são afetados? iii) Quais são os efeitos do risco/falha sobre o processo? iv) Qual a importância do risco/falha? e v) Como preveni-los? (AGUIAR; MELLO, 2008)[5].

Para a presente pesquisa, foi desenvolvido um conjunto de medidas de análise para o processo de desenvolvimento de *software* fundamentado no método ágil, um dos elos mais importantes dos objetivos estratégicos da governança do TI. O método está intimamente ligado à qualidade do produto a ser entregue. Além disso, a análise seguiu a proposta de aplicação do FMEA do trabalho de Santos e Cabral (2008)[55].

A redução desse tempo por intermédio de melhorias de produtividade não só reduz o tempo de desenvolvimento, como também traz eficiência na execução do ciclo processual da metodologia adotada, uma vez que o tempo necessário para entregar uma solução é responsável por uma grande parcela da lentidão do processo de desenvolvimento da mesma.

Dada a relevância a ser atribuída na mensuração do processo que compõe o desenvolvimento de *software*, destaca-se que quanto mais adequado for o sistema de mensuração, mais consistente será o tempo de ciclo do processo e maiores níveis poderão ser oferecidos à organização.

De acordo com Araujo (1997)[8] a medição pode ser uma ferramenta de tomada de decisão, realizada com observações quantificadas de algum atributo de processo, produto ou projeto, que possibilita a capacidade de compreensão das situações imateriais do entendimento organizacional.

Para Ward *apud* Araujo (1997)[8] o uso de medidas de forma adequada fundamenta a atuação de um processo, ao se basear no monitoramento das atividades essenciais para qualquer esforço contínuo da busca da qualidade, observando que não se pode gerenciar aquilo que não se pode medir.

A aplicação proposta por Santos e Cabral (2008)[55] prevê três etapas principais, são elas: a elaboração do Diagrama do Risco, a elaboração da Matriz de Avaliação e, por fim, a elaboração da Lista de Ações.

Segundo a proposta de implementação do FMEA de Santos e Cabral (2008)[55], a primeira etapa é a preparação do diagrama do risco onde são identificados o objetivo, o efeito do risco, grupo de causas do risco e causas do risco, como apresentado na Figura 39.

A Matriz de Avaliação, segundo os autores, explora os riscos contidos em cada grupo por meio da atribuição de valores para cada um dos índices propostos: severidade, ocorrência e detecção, tomadas como critério a escala Likert de 1 a 5, apresentada na Tabela 4.4. A multiplicação destes três fatores resulta no índice de risco (IR) o qual aponta a prioridade que deve ser incorrida em cada um dos riscos.

Tabela 4.4: Critérios para identificação dos índices de ocorrência, severidade e detecção utilizado no FMEA, Fonte: STAMATIS (1995) *apud* MATOS, MILAN (2009)[42]

Índices	Critérios do FMEA		
	Ocorrência (Oc)	Severidade (Sv)	Detecção (Dt)
1	Probabilidade muito remota de acontecer	É razoável esperar que o cliente não perceberá a risco / falha	Probabilidade muito alta que o risco / falha seja detectada
2	Número de ocorrência baixo	O cliente perceberá o risco/ falha, mas não ficará insatisfeito por causa dela	Probabilidade alta de que o risco / falha seja detectada
3	Número de ocorrência moderado	O cliente perceberá o risco / falha e ficará insatisfeito	Probabilidade média de que o risco / falha seja detectada
4	Número de ocorrência alto	O cliente ficará insatisfeito, mas não tem uma segurança afetada	Probabilidade baixa de que o risco / falha seja detectada
5	Riscos / Falhas em proporções alarmantes	O cliente ficará muito insatisfeito e afeta a sua segurança	Probabilidade muito baixa de que o risco / falha seja detectada

O índice de Risco (IR) é calculado baseando-se nas informações relativas ao modo de risco/falha potencial, à severidade do risco/falha e à capacidade de detectá-los antes de chegarem ao cliente. Para isso, foi avaliado cada ponto em termos de índices de (Sv) Severidade do risco/falha, (Oc) possibilidade de ocorrência do risco/falha, (Dt) possibilidade de detecção do risco/falha e calculamos o índice de risco, IR = Oc x Sv x Dt.

Após o cálculo do IR é possível ranquear os riscos a fim de estabelecer uma estratégia para um deles, baseada na Norma de Gestão de Riscos ISO 31000/2009 (ABNT, 2009)[2], onde são estabelecidas as três estratégias de tratamento de riscos citado abaixo. A Lista de Ações é, portanto resultado desta terceira etapa e será insumo para a elaboração do Plano de Riscos apresentado no capítulo 5.

1. NEUTRALIZAÇÃO - consiste na modificação do uso ou do tipo de recurso informacional, nas condições ambientais ou qualquer outros fatores que tenham como consequência a eliminação da causa que está gerando o risco.
2. MITIGAÇÃO - consiste em medidas que diminuam o impacto e/ou as chances de um risco se concretizar.
3. ACEITAÇÃO - consiste na aceitação do risco tal como foi estimado sem medidas adicionais para seu controle.

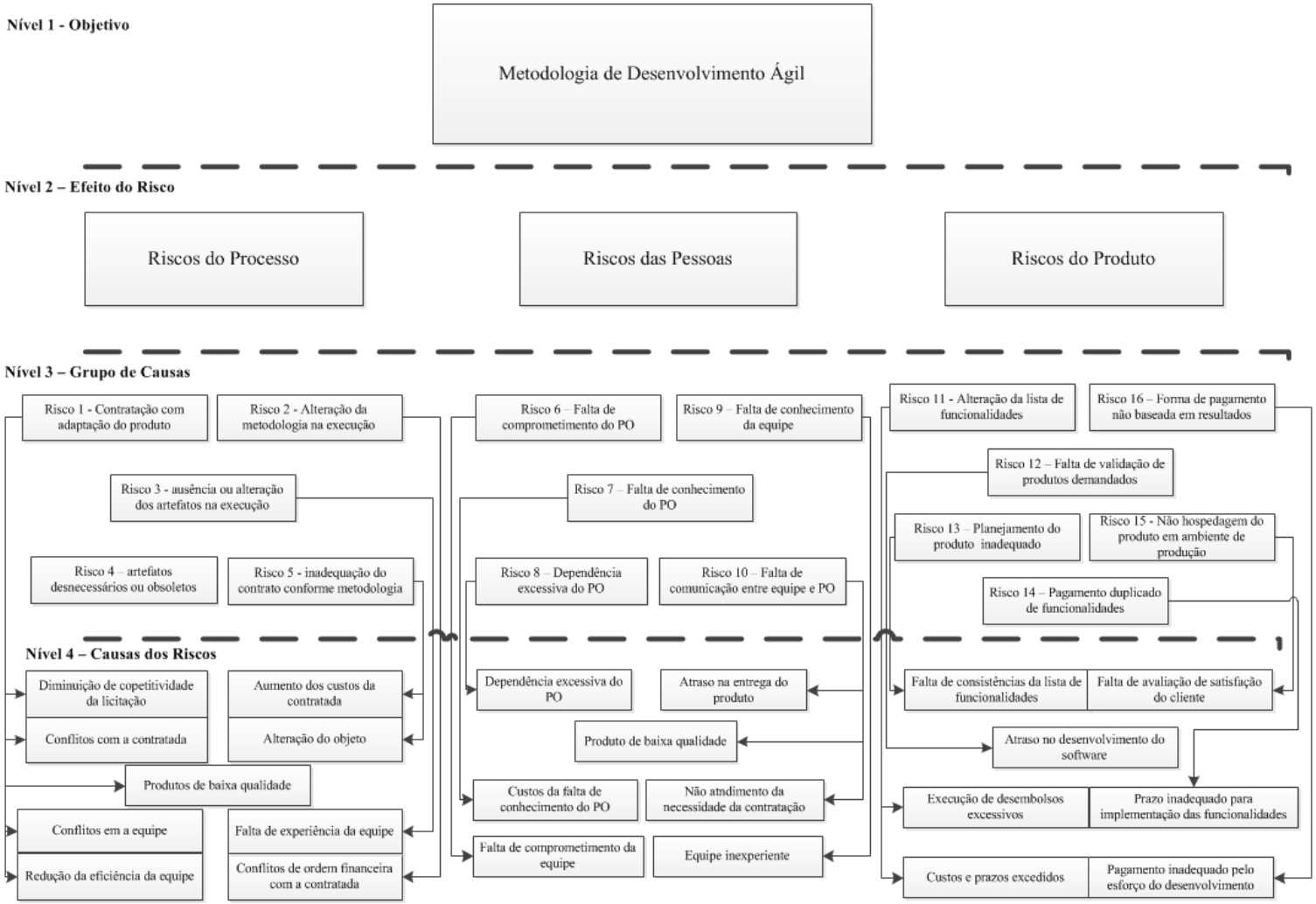

Figura 4.36: Diagrama do Risco, Fonte: (SANTOS e CABRAL, 2008)[55], adaptado pelo autor

Depois de realizadas as entrevistas com as APFs, bem como os Gerentes de Projetos do CDS, foram realizadas as identificações dos pontos críticos, por intermédio da análise do índice de risco. Nos tópicos a seguir serão apresentados os resultados referentes ao mapeamento de riscos dos processos citados.

4.4.1.1 Análise dos Riscos das Administrações Públicas Federais

Na seção 4.4 foram apresentados os três casos de referência entrevistados quanto à utilização da metodologia ágil no desenvolvimento de *software*, em um segundo momento foram analisados os riscos vinculados a aplicação desta metodologia, também segundo as três APFs.

Os riscos utilizados nesta análise estão de acordo com o Acórdão n. 2314-33/2013 (Figura 38) e foram agrupados em três grupos distintos: os que são relacionados ao processo propriamente dito, os que são relacionados às pessoas e por fim os relacionados aos produtos.

Os entrevistados foram questionados, portanto, quanto à ocorrência, a severidade e a detecção dos riscos listados. Para cada um deles foram atribuídos valores de 1 a 5 com o intuito de fornecer a análise necessária, como já explicado na Tabela 8.

Após a consolidação dos resultados provenientes das entrevistas com os representantes do IPHAN, do BACEN, e do TSE, alguns apontamentos foram evidenciados. O primeiro deles diz respeito à análise da participação de cada grupo de riscos na utilização da metodologia ágil. É possível observar que os riscos estão concentrados ou no processo ou no produto, como demonstrado na Tabela 4.5 de forma priorizada, conforme indicado Santos e Cabral (2008)[55] e nos gráficos a seguir.

Tabela 4.5: FMEA de falha/risco priorizados na utilização do ágil nas APFs, Fonte Autor

	Índices			
	Oc	Sv	Dt	Risco (IR)
Riscos relativos ao processo				
• Risco 4: exigência de artefatos desnecessários ou que se tornam obsoletos rapidamente.	1,7	5,0	2,3	19
• Risco 1: contratação de desenvolvimento de *software* com adaptação de metodologia ágil que desvirtue sua essência	1,3	5,0	1,0	7
• Risco 2: alteração da metodologia ágil adotada no instrumento convocatório no decorrer da execução contratual	1,3	5,0	1,0	7
• Risco 3: ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual	1,0	5,0	1,0	5
• Risco 5: utilização de contrato para desenvolvimento de *software* por metodologias tradicionais para desenvolvimento por métodos ágeis.	1,0	5,0	1,0	5
Riscos relativos a pessoas				
• Risco 9: equipe da empresa contratada não ter expertise em desenvolvimento de *software* com métodos ágeis.	1,7	5,0	1,0	8
• Risco 10: dificuldade de comunicação entre a equipe de desenvolvimento da contratada com o indicado pela área de negócios *Product Owner* (PO).	1,7	5,0	1,0	8
• Risco 7: falta do conhecimento necessário do indicado pela área de negócios *Product Owner* (PO) para o desenvolvimento do *software*.	1,0	5,0	1,0	5
• Risco 8: excessiva dependência da visão do indicado pela área de negócios *Product Owner* (PO).	1,0	5,0	1,0	5
• Risco 6: falta de comprometimento ou colaboração insatisfatória do responsável indicado pela área de negócios *Product Owner* (PO) no desenvolvimento do *software*.	2,3	1,0	2,0	5
Riscos relativos a produtos				
• Risco 11: alteração constante da lista de funcionalidades do produto.	2,3	4,0	1,3	12
• Risco 14: pagamento pelas mesmas funcionalidades do *software* mais de uma vez, em virtude de funcionalidades impossíveis de serem implementadas em um único ciclo, ou em virtude da alteração de funcionalidades ao longo do desenvolvimento do *software*.	1,3	5,0	1,0	7
• Risco 12: iniciação de novo ciclo sem que os produtos construídos na etapa anterior tenham sido validados.	2,0	2,7	1,0	5
• Risco 13: falta de planejamento adequado do *software* a ser construído.	1,0	5,0	1,0	5
• Risco 15: não disponibilização do *software* em ambiente de produção para a utilização e avaliação dos reais usuários.	1,0	5,0	1,0	5
• Risco 16: forma de pagamento não baseada em resultados.	1,0	5,0	1,0	5

Legenda:

Máx.:	125	
10%	30%	100%
12,5	37,5	125
1 a 12	12.1 a 37	37.1 a 125
Baixo	Moderado	Alto

Ainda analisando os três grupos de riscos, com o objetivo de obter informações a respeito da média encontrada baseada na resposta dos entrevistados quanto aos critérios apresentados, chama a atenção para a diferença encontrada quanto à severidade atribuída a cada um dos grupos quando comparada aos outros dois critérios, detecção e ocorrência, apresentado na Figura 4.37.

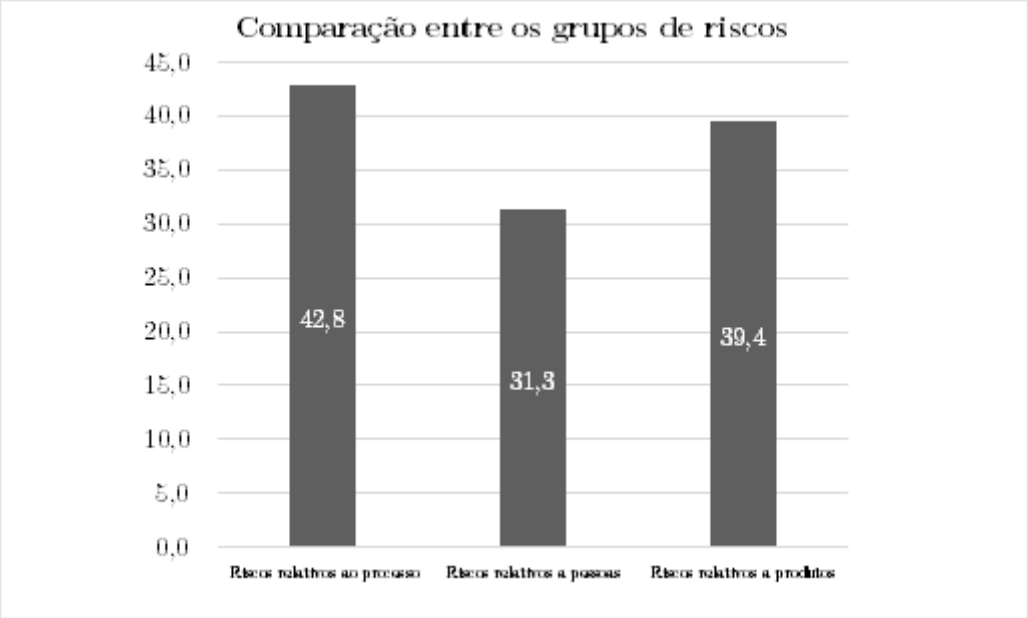

Figura 4.37: Gráfico de Comparação percentual dos grupos de riscos)

A Figura 4.38 deixa claro, portanto, que apesar dos riscos serem considerados severos frente à metodologia ágil estes não têm ocorrido com frequência segundo a experiência dos entrevistados, bem como os riscos apresentam alto poder de serem mitigados, caso ocorram.

Este cenário corrobora a implementação da metodologia ágil no desenvolvimento de *software* de instituições federais públicas federais considerando que seu impacto tende a ser positivo e seus riscos têm tendência positiva no sentido de se tornarem oportunidades de melhoria.

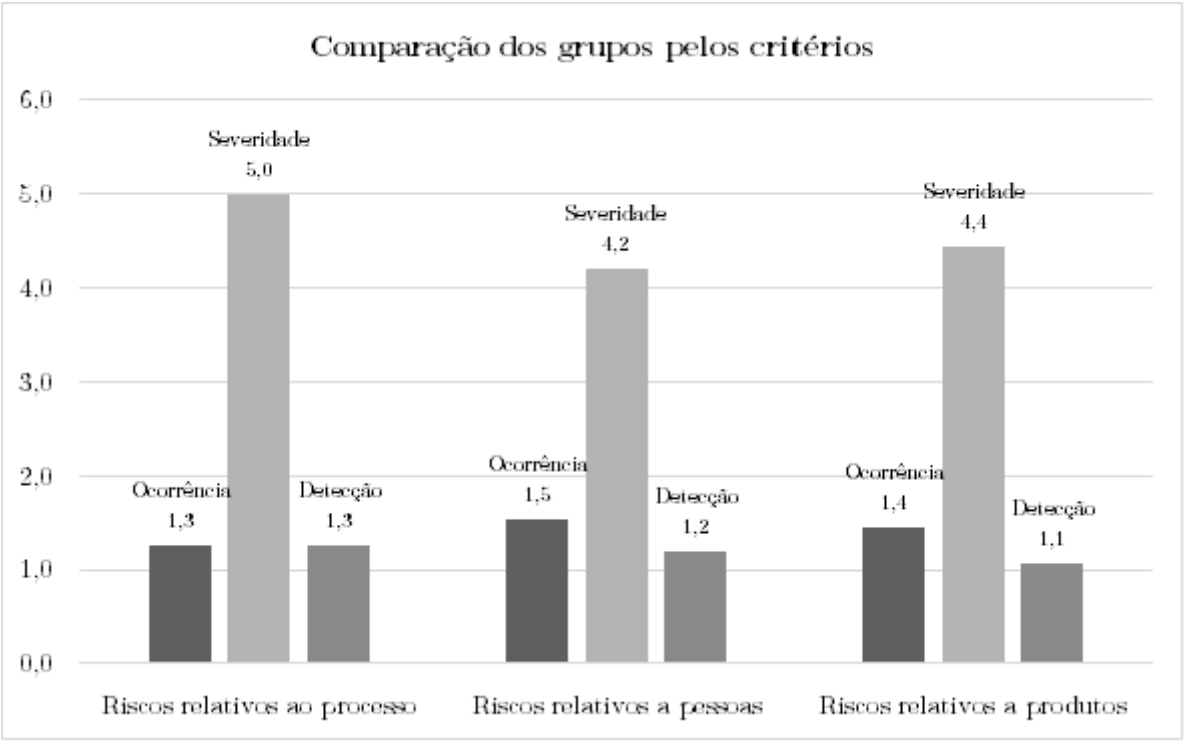

Figura 4.38: Gráfico de Comparação dos grupos pelos critérios

Por fim, cada um dos grupos pode ser analisado em termos dos riscos que os compõem.

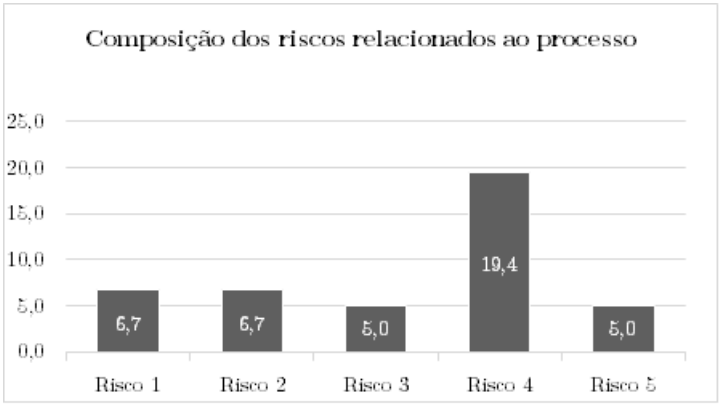

Figura 4.39: Gráfico de Composição dos riscos relacionados ao processo

Os riscos que foram agrupados por corresponderem aos processos apresentam disparidade clara quanto ao seu índice. Foram atribuídos pelos entrevistados valores similares quanto aos riscos 1, 2, 3 e 5 fazendo com que a média estivesse em torno de 5 ou 6. Por outro lado, o risco 4, que aponta a exigência de artefatos desnecessários, foi avaliado muito alto quando comparado aos demais riscos deste grupo. Esta avaliação é decorrência da implementação da filosofia ágil, dado que é objetivo desta metodologia diminuir a quantidade de documentos utilizados e, desta forma, quando ocorre a exigência de artefatos a mais este risco tem algo impacto nos processos.

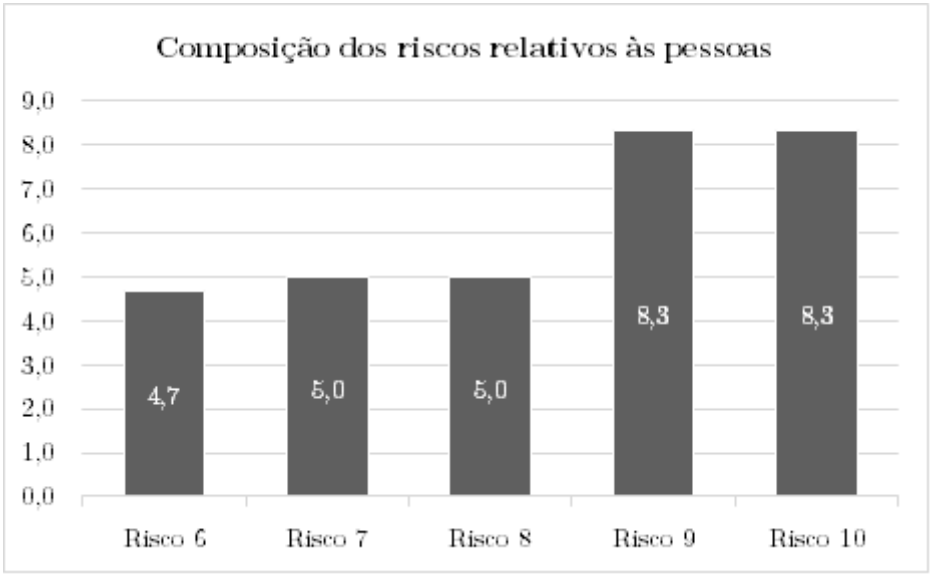

Figura 4.40: Gráfico de Composição dos riscos relativos às pessoas

Em contrapartida, os riscos relacionados ao pessoal tiveram uma avaliação mais uniforme. Como pode ser visto na figura 4.40, os riscos 6, 7 e 8 receberam notas medianas em torno de 5, enquanto aos riscos 9 e 10 foram atribuídos valores mais altos. Estes dois riscos correspondem, respectivamente, à equipe da empresa contratada não ter expertise em desenvolvimento de *software* com métodos ágeis e à dificuldade de comunicação entre a equipe de desenvolvimento da empresa contratada com o indicado pela área de negócios *Product Owner* (PO).

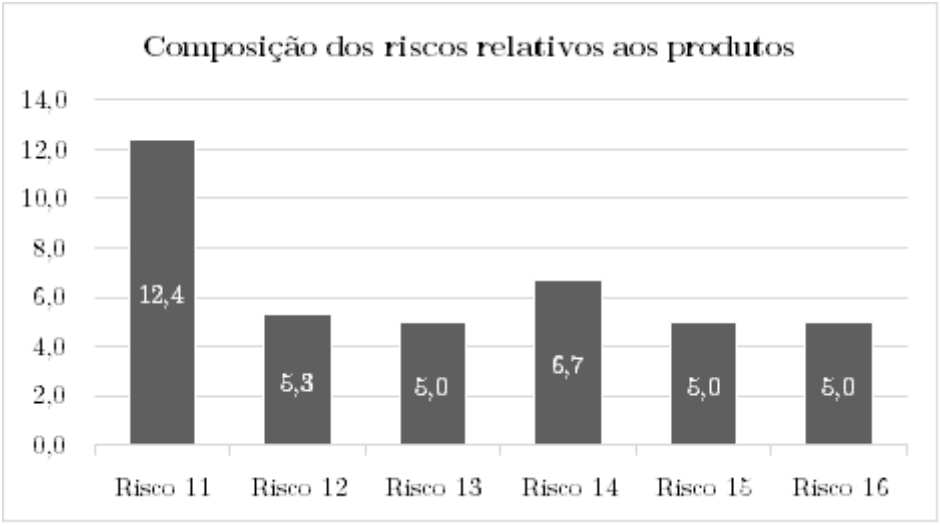

Figura 4.41: Gráfico de Composição dos riscos relativos aos produtos

Por fim, na figura 4.41 são apresentados os riscos relacionados ao produto, os quais, assim como os riscos de processos, apresentaram uniformidade apenas em parte da sua composição. A alteração constante da lista de funcionalidades do produto foi considerada um ponto crucial para o sucesso do desenvolvimento da metodologia exatamente pelo retrabalho causado.

4.4.1.2 Análise dos Riscos do Centro de Desenvolvimento de Sistemas

Proveniente dos resultados do questionário de incidência dos riscos associados ao entendimento do TCU proferidos por intermédio do Acórdão n. 2314-33/2013, no âmbito das instituições federais visitadas, o mesmo questionário foi aplicado aos Gerentes de Projeto do CDS. Com o objetivo de avaliar a viabilidade de aplicação e/ou usabilidade do método proposto no presente trabalho de pesquisa do ponto de vista da gestão dos riscos ora identificados pelo órgão de controle citado. Conforme citado no tópico acima e baseado no trabalho de Santo e Cabral (2008), foi desenvolvida um FMEA contendo a priorização em termos de índices de (Sv) Severidade do risco/falha, (Oc) possibilidade de ocorrência do risco/falha, (Dt) possibilidade de detecção do risco/falha e calculamos o índice de risco, IR = Oc x Sv x Dt, conforme demonstrado na Tabela 4.6, devidamente priorizados.

Tabela 4.6: FMEA de falha/risco priorizados na utilização do ágil no CDS, Fonte Autor

Riscos relativos ao processo	Índices			
	Oc	Sv	Dt	Risco (IR)
oRisco 3: ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual	3,2	3,0	2,6	25,0
oRisco 2: alteração da metodologia ágil adotada no instrumento convocatório no decorrer da execução contratual	3,0	2,4	2,8	20,2
oRisco 4: exigência de artefatos desnecessários ou que se tornam obsoletos rapidamente.	2,4	2,6	3,0	18,7
oRisco 5: utilização de contrato para desenvolvimento de *software* por metodologias tradicionais para desenvolvimento por métodos ágeis.	2,4	2,0	3,0	14,4
oRisco 1: contratação de desenvolvimento de *software* com adaptação de metodologia ágil que desvirtue sua essência	2,2	2,0	3,0	13,2
Riscos relativos a pessoas				
oRisco 6: falta de comprometimento ou colaboração insatisfatória do responsável indicado pela área de negócios *Product Owner* (PO) no desenvolvimento do *software*.	2,8	3,6	2,4	24,2
oRisco 9: equipe da empresa contratada não ter expertise em desenvolvimento de *software* com métodos ágeis.	3,6	4,0	2,4	34,6
oRisco 8: excessiva dependência da visão do indicado pela área de negócios *Product Owner* (PO).	3,8	2,8	2,6	27,7
oRisco 7: falta do conhecimento necessário do indicado pela área de negócios *Product Owner* (PO) para o desenvolvimento do *software*.	2,6	3,8	2,2	21,7
oRisco 10: dificuldade de comunicação entre a equipe de desenvolvimento da contratada com o indicado pela área de negócios *Product Owner* (PO).	2,0	3,0	2,4	14,4
Riscos relativos a produtos				
oRisco 11: alteração constante da lista de funcionalidades do produto.	3,6	3,2	2,8	32,3
oRisco 12: iniciação de novo ciclo sem que os produtos construídos na etapa anterior tenham sido validados.	2,8	3,6	2,4	24,2
oRisco 13: falta de planejamento adequado do *software* a ser construído.	3,0	3,4	2,2	22,4
oRisco 14: pagamento pelas mesmas funcionalidades do *software* mais de uma vez, em virtude de funcionalidades impossíveis de serem implementadas em um único ciclo, ou em virtude da alteração de funcionalidades ao longo do desenvolvimento do *software*.	2,0	3,8	2,4	18,2
oRisco 16: forma de pagamento não baseada em resultados.	1,8	3,4	2,0	12,2
oRisco 15: não disponibilização do *software* em ambiente de produção para a utilização e avaliação dos reais usuários.	1,6	4,2	1,4	9,4

Legenda:

Máx.:	125	
10%	30%	100%
12,5	37,5	125
1 a 12	12.1 a 37	37.1 a 125
Baixo	Moderado	Alto

Observando o resultado da Tabela 10 e da figura 4.42 de comparação entre os grupos de

riscos é possivel indentificar que o maior índice de incidência de risco está ligado às pessoas, resultado factível de mitigação tendo em vista que o CDS é uma Organização Militar (OM) apenas responsável pelo desenvolvimento do *software* e totalmente dependente das OM demandantes, ou seja, clientes do Centro.

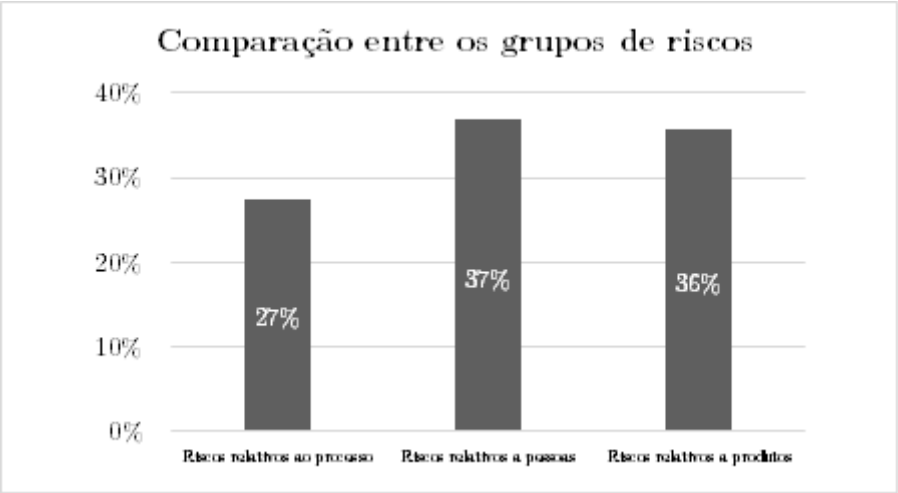

Figura 4.42: Gráfico de Comparação entre os grupos de riscos

Diferente do resultado do gráfico acima, a figura 4.43 traz como referência a comparação dos grupos pelos critérios associados ao processo, pessoas e produtos. O indicador de severidade do grupo de risco ligado ao produto teve resultado aumentado pelo nível de probabilidade da ocorrência de riscos, indicando a gestão da qualidade dos produtos a serem entregues aos clientes do Centro.

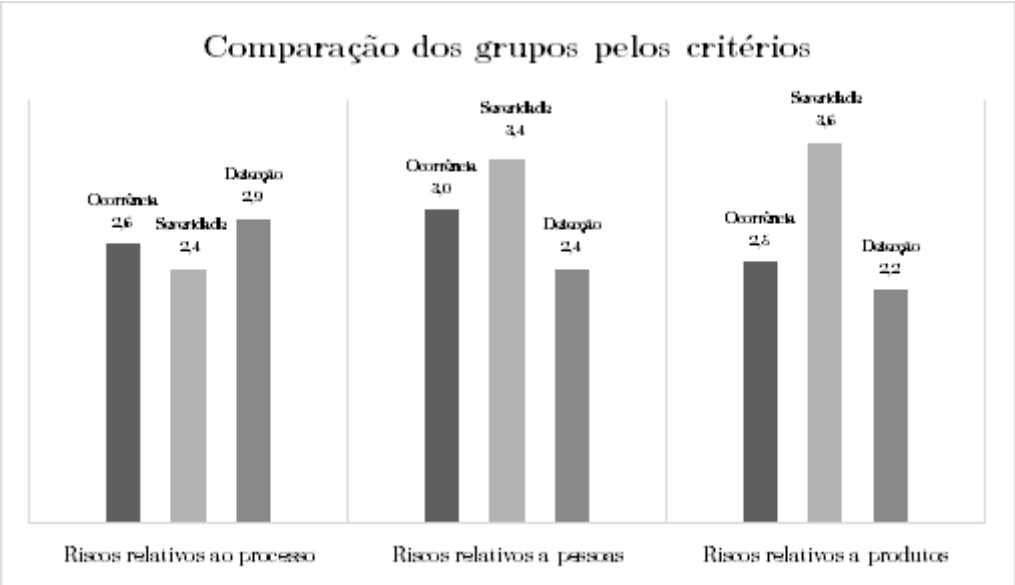

Figura 4.43: Comparação dos grupos pelos critérios

Observando a figura 4.44 de composição dos riscos relacionados ao processo, foi obtida uma média de 13, destacando o indicador do risco 3 de ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual, com o resultado de 25 motivado pela existência de uma normativa dentro do Exército no que diz respeito ao cumprimento pelos Gerentes de Projeto e/ou Gestores de Contratos que sejam estritamente o previsto no instrumento de contrato não sendo aceitável a mudança de requisitos previstos no instrumento, deixando claro a necessidade de realizar a mitigação do risco.

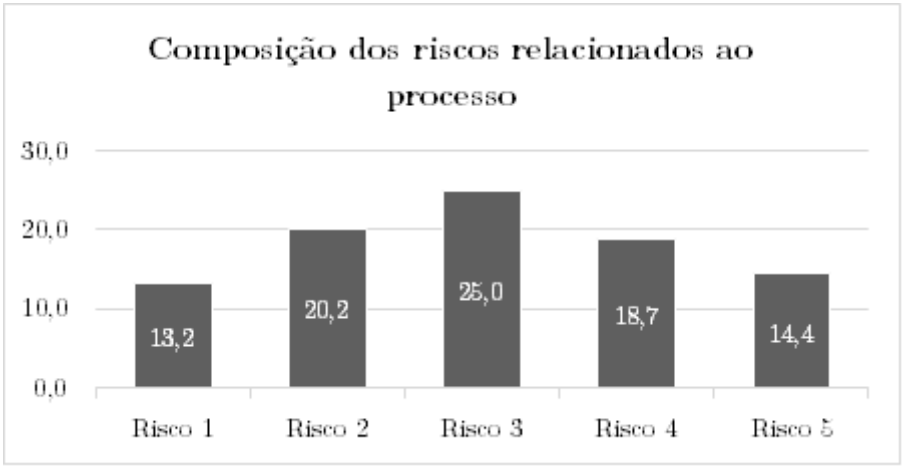

Figura 4.44: Gráfico de Composição dos riscos relacionados ao processo

Na composição dos riscos relativos às pessoas demonstrado na figura 4.45 é chamado a atenção para o indicador do risco 9 com 34,6 pontos superior a média de 20 pontos obtidos nos riscos 6,7,8, e 10. Em que existe a necessidade de mitigação do risco uma vez que possa ocorrer na equipe contratada não tenha expertise em desenvolvimento de *software* com métodos ágeis, devendo a mesma ser capaz de comprovar sua experiência na execução do desenvolvimento fundamentado no ágil, requisito essencial para o bom resultado do projeto.

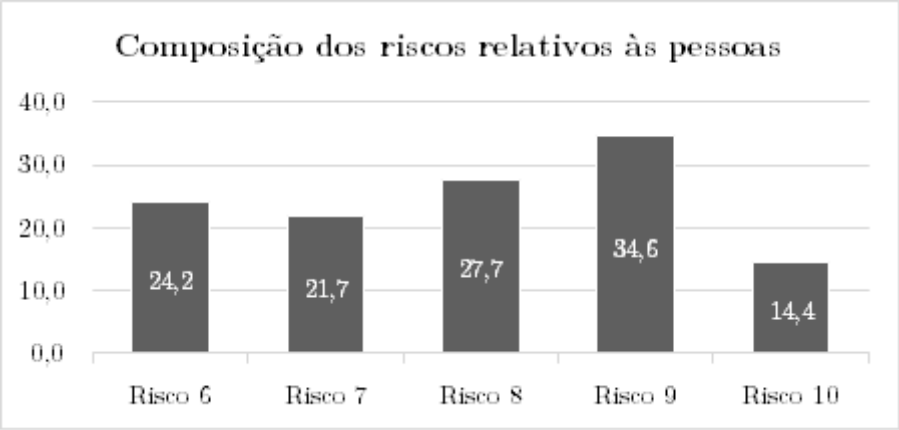

Figura 4.45: Gráfico de Composição dos riscos relativos às pessoas

Finalmente é apresentado a figura 4.46 a composição dos riscos relativos ao produto, semelhante ao resultado obtido nas APFs o indicador do risco 11 obteve resultado alto no qual ressalta que a alteração constante da lista de funcionalidades do produto foi considerada um ponto de risco. Não obstante, é necessário observar que para o sucesso do desenvolvimento da metodologia ágil é importante que a equipe envolvida no processo de desenvolvimento de *software* seja capaz de gerenciar as possíveis modificações das funcionalidades, uma vez que a essência do ágil é garantir que as possíveis modificações dos requisitos sejam aceitas como forma agregadora ao produto, não devendo ser entendido como prejuízo ou mesmo retrabalho.

Figura 4.46: Gráfico de Composição dos riscos relativos aos produtos

Todavia, com as resultados das pesquisas realizadas é possível identificar que o uso da metodologia ágil pela Administração Pública Federal, assim como para o Exército Brasileiro é viável, uma vez que os riscos apresentados na presente pesquisa são passiveis de mitigação.

Como um dos objetivos propostos na presente dissertação no tópico 5.1.2 é apresentado o Plano de Gestão de Risco do Processo de Desenvolvimento ágil para o CDS com as boas práticas e ferramentas de gestão dos riscos relacionados pelo TCU, consubstanciado nos resultados da análise do FMEA. Este foi elabora tendo como base na aplicação dos questionários realizados nas APFs e com os Gerentes de Projetos do CDS.

Tabela 4.7: FMEA de falha/risco priorizados na utilização do ágil baseado nos estudos de caso, Fonte Autor

Riscos		Índices CDS				Índices APF				Média (IR)	Estratégia de Tratamento
		Oc	Sv	Dt	Risco (IR)	Oc	Sv	Dt	Risco (IR)		
Riscos relativos ao processo	oRisco 4: exigência de artefatos desnecessários ou que se tornam obsoletos rapidamente.	2,4	2,6	3,0	18,7	1,7	5,0	2,3	19	19,1	Neutralizar
	oRisco 3: ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual	3,2	3,0	2,6	25,0	1,0	5,0	1,0	5	15,0	Neutralizar
	oRisco 2: alteração da metodologia ágil adotada no instrumento convocatório no decorrer da execução contratual	3,0	2,4	2,8	20,2	1,3	5,0	1,0	7	13,4	Neutralizar
	oRisco 1: contratação de desenvolvimento de *software* com adaptação de metodologia ágil que desvirtue sua essência	2,2	2,0	3,0	13,2	1,3	5,0	1,0	7	9,9	Neutralizar
	oRisco 5: utilização de contrato para desenvolvimento de *software* por metodologias tradicionais para desenvolvimento por métodos ágeis.	2,4	2,0	3,0	14,4	1,0	5,0	1,0	5	9,7	Neutralizar
Riscos relativos a pessoas	oRisco 9: equipe da empresa contratada não ter expertise em desenvolvimento de *software* com métodos ágeis.	3,6	4,0	2,4	34,6	1,7	5,0	1,0	8	21,4	Mitigar
	oRisco 8: excessiva dependência da visão do indicado pela área de negócios *Product Owner* (PO).	3,8	2,8	2,6	27,7	1,0	5,0	1,0	5	16,3	Aceitar
	oRisco 6: falta de comprometimento ou colaboração insatisfatória do responsável indicado pela área de negócios *Product Owner* (PO) no desenvolvimento do *software*.	2,8	3,6	2,4	24,2	2,3	1,0	2,0	5	14,4	Mitigar
	oRisco 7: falta do conhecimento necessário do indicado pela área de negócios *Product Owner* (PO) para o desenvolvimento do *software*.	2,6	3,8	2,2	21,7	1,0	5,0	1,0	5	13,4	Mitigar
	oRisco 10: dificuldade de comunicação entre a equipe de desenvolvimento da contratada com o indicado pela área de negócios *Product Owner* (PO).	2,0	3,0	2,4	14,4	1,7	5,0	1,0	8	11,4	Mitigar

Riscos relativos a produtos	oRisco 11: alteração constante da lista de funcionalidades do produto.	3,6	3,2	2,8	32,3	2,3	4,0	1,3	12	22,4	Aceitar
	oRisco 12: iniciação de novo ciclo sem que os produtos construídos na etapa anterior tenham sido validados.	2,8	3,6	2,4	24,2	2,0	2,7	1,0	5	14,8	Mitigar
	oRisco 13: falta de planejamento adequado do *software* a ser construído.	3,0	3,4	2,2	22,4	1,0	5,0	1,0	5	13,7	Mitigar
	oRisco 14: pagamento pelas mesmas funcionalidades do *software* mais de uma vez, em virtude de funcionalidades impossíveis de serem implementadas em um único ciclo, ou em virtude da alteração de funcionalidades ao longo do desenvolvimento do *software*.	2,0	3,8	2,4	18,2	1,3	5,0	1,0	7	12,5	Neutralizar
	oRisco 16: forma de pagamento não baseada em resultados.	1,8	3,4	2,0	12,2	1,0	5,0	1,0	5	8,6	Neutralizar
	oRisco 15: não disponibilização do *software* em ambiente de produção para a utilização e avaliação dos reais usuários.	1,6	4,2	1,4	9,4	1,0	5,0	1,0	5	7,2	Mitigar

Legenda:

Máx.:	125	
10%	30%	100%
12,5	37,5	125
1 a 12	12.1 a 37	37.1 a 125
Baixo	Moderado	Alto

Conclusivamente, as visitas técnicas permitiram uma melhor visualização de como os métodos ágeis podem ser aplicados no gerenciamento de desenvolvimento de *software* por uma empresa contratada, sobretudo com relação ao obtido no IPHAN e no BACEN, dois órgãos que realizam tais modalidades de contratação.

O importante a ser destacado é que não existe a necessidade de que o desenvolvimento do *software* seja realizado com um método ágil específico. Segundo ambos os órgãos, o relacionamento entre as partes (desenvolvedor, TI e demandante) é que precisa se dar em função de métodos ágeis. Nos dois casos, foram criadas metodologias próprias para tal gerenciamento: o IPHAN possui a Metodologia IPHAN de Gestão de Demandas de Desenvolvimento Ágil de *Softwares* – MIDAS; o BACEN tem como guia o PDS-BC Ágil - Processo Ágil de Desenvolvimento de *Software* do Banco Central. Ambos os documentos propiciam uma padronização do fluxo de trabalho e das atividades que compõem o processo de planejamento, controle do desenvolvimento e aceitação do sistema. Assim,

o que pode ser chamado de aplicação de métodos ágeis é, na verdade, um conjunto de medidas e práticas que visam gerenciar e realizar o trabalho de acompanhamento do desenvolvimento, que está sendo realizado pela contratada.

Os métodos ágeis trabalham com a ideia de reuniões diárias para que sejam debatidos os problemas, as limitações e os avanços do desenvolvimento. No caso da contratação, no entanto, tais reuniões são mais difíceis e, portanto, não são realizadas diariamente, por uma limitação temporal e espacial (uma vez que a fábrica de *software* não estará instalada no órgão contratante). Assim, as reuniões são mais esparsas, limitadas, muitas vezes, à coleta de requisitos e à apresentação do produto.

Portanto, a definição de *sprints* com curto prazo de duração (entre semanas e um mês) concorre tanto para uma maior velocidade nas entregas como uma maior interação entre demandante e contratada, de forma que, ao final de uma *sprint*, a fábrica terá outra atividade definida para iniciar e, até a próxima entrega, a contratante precisa testar e validar a entrega anterior.

Na pesquisa, percebeu-se que nesses procedimentos de gestão do desenvolvimento ágil alguns poucos centralizam o controle da criação e difusão do conhecimento. Além disso, as produtividades desses especialistas ficam comprometidas com as diversas investidas da equipe de analistas em busca da solução dos problemas decorrentes da rotina diária do processo de desenvolvimento de *software*.

A pesquisa mostrou também que a experiência na função e a maturidade pessoal são fatores relevantes para que o processo seja executado com qualidade. Isso é comprovado pelo tempo de exercício da função, como também a maturidade do método escolhido pela organização, destacando o dado relevante para a pesquisa a escolha do *framework Scrum*, com 3 das 5 instituições que responderam o questionário aplicado.

Foi possível identificar que o uso da metodologia ágil pela administração pública federal é viável no que tange à produtividade e satisfação da qualidade dos produtos entregues, levando em consideração que 98% dos indicadores apontados nos resultados obtidos são favoráveis ao uso do método ágil. Desta forma atende-se a hipótese levantada no Capítulo 1 em que o processo de maturidade no uso do método ágil garante a inovação no conhecimento da administração pública federal.

Após a consolidação dos resultados provenientes das entrevistas com os representantes do IPHAN, do BACEN e do TSE, alguns apontamentos foram evidenciados. O primeiro deles diz respeito à análise da participação de cada grupo de riscos quanto à utilização da metodologia ágil.

A análise desenvolvida por meio da utilização da ferramenta do FMEA trouxe a priorização dos riscos listados pelo TCU por intermédio do Acórdão n. 2314-33/2013, preservando os grupos já citados e evidenciando as possíveis fragilidades que as APFs

enfrentaram e que o CDS poderá vir a se deparar. Dando continuidade a pesquisa, no próximo capítulo será apresentada a proposta do processo de desenvolvimento de *software* ágil para o Exército Brasileiro, a ser executada por intermédio do Centro de Desenvolvimento de Sistemas (CDS).

Capítulo 5

Proposta de Desenvolvimento de *Software* fundamentado no Método Ágil para o EB

O processo de desenvolvimento de *software* tem por finalidade criar um *software* que seja confiável e trabalhe eficientemente. Portanto, neste capítulo será apresentado a proposta de um processo de desenvolvimento ágil para o EB, dividido em dois tópicos.

No primeiro tópico é apresentado a proposta de processo de desenvolvimento de *software* para o EB com base na metodologia ágil, e no segundo tópicos é apresentado o Plano de Gestão de Riscos identificados pelo TCU por intermédio do Acórdão n. 2314/2013.

Inicialmente é importante destacar que as metodologias ágeis são adaptativas, partem do princípio de que o conhecimento sobre o problema aumenta à medida que o sistema vai sendo desenvolvido, levando a uma busca constante por melhores soluções. O desenvolvimento ocorre de forma iterativa e evolutiva, com iterações tão curtas quanto possíveis (de 2 a 4 semanas). Ao final de cada iteração, obtém-se uma versão com a implementação de um novo subconjunto de funcionalidades, pronta para ser colocada em produção.

As metodologias ágeis são orientadas a pessoas, para melhor gerenciar projetos de *software*, pessoas devem ser tratadas como indivíduos e não como recursos, pois cada pessoa apresenta um ritmo de trabalho único e completamente diferente, que é fruto da sua vivência pessoal. Os membros da equipe escolhem as tarefas que irão desenvolver durante a iteração seguinte, além de estimarem o tempo a ser gasto com elas, o que aumenta a motivação, o comprometimento e a produtividade da equipe.

As interações são flexíveis e iterativas: adaptam-se constantemente ao conjunto de requisitos mais atual: a cada interação, usuários, clientes e desenvolvedores decidem sobre

quais características devem ser adicionadas, modificadas e até retiradas do *software*.

Além de ser desenvolvido da forma mais iterativa possível, é possível escolher os pontos mais críticos ou que mais agregam valor ao sistema para serem desenvolvidos em primeiro lugar. Estão mais voltadas para o bem do negócio: seu caráter adaptativo aumenta as chances de oferecer um melhor resultado ao projeto e ao negócio.

Para Pressman (2011)[50], os pontos fracos são risco de desorganização, pouca documentação e alta dependência do cliente/usuário, o que muitas vezes não é viável. Parte do princípio de que é preferível fazer algo simples de imediato e pagar um pouco mais para alterá-lo depois, se necessário, do que fazer algo complicado de imediato e acabar não utilizando depois. Assim, com os estudos sobre metodologias ágeis é possível definir que o desenvolvimento ágil é uma atitude, não um processo descritivo, é uma forma efetiva de se trabalhar em conjunto para atingir as necessidades das partes interessadas no projeto. Reúnem boas práticas de programação e de gerenciamento de projetos a fim de produzir software com qualidade. Propõem uma alternativa que busca a melhoria do processo, tornando-o mais ágil e menos burocrático.

As metodologias ágeis possuem em comum com a metodologia tradicional o fato de ser aplicado em projetos não muito complexos, utilizando ciclos iterativos curtos, planejamento guiado por funcionalidades, constantes mudanças e tolerância com as mudanças, proximidade da equipe, intimidade com o cliente e um foco no ambiente geral de trabalho do time. A maioria das metodologias ágeis possui métodos e práticas muito parecidas, o que realmente mudam são os enfoques e os valores de cada método.

Nesse sentido, no próximo tópico é apresentado o macro processo de desenvolvimento ágil, cuja missão é apoiar a atividade de assessoria técnica de desenvolvimento de *software* do CDS com boas práticas da metodologia ágil adotada na APF. Destaca-se que sua estrutura é composta de todos os macrodiagramas, representada na Figura 40, como também é determinado o processo que será estudado e os atores envolvidos, sistemas, infraestruturas, instrumentos, interfaces de entrada e saída e artefatos de entrada e saída. São definidas as leis, normas e políticas que impactam o processo. São determinados os requisitos, necessidades e expectativas do processo. Além disto, os indicadores de desempenho para medir o processo atual.

5.1 Processo de desenvolvimento ágil proposto

Para De Carvalho e Mello (2012)[26] o *software* deve utilizar: métodos, ferramentas e procedimentos para a compreensão de como é possível projetar, implementar, avaliar e melhorar seu processo de seu desenvolvimento. Da mesma forma, o desenvolvimento de *software* tem como objetivo a "criação de sistemas de software que satisfaçam às

necessidades de clientes e usuários", de forma que é essencial que se realize uma correta especificação dos requisitos do *software* para se obter o sucesso do processo.

Em geral um processo é considerado como uma coleção de atividades para a realização de tarefas. Cada sucessão de atividades tem que ser executada de acordo com o modelo do processo. Estas atividades podem ser conduzidas por humanos ou máquinas de uma forma integrada.

Desta forma, o processo de desenvolvimento ágil do EB tem como objetivo definir um conjunto de procedimentos, métodos e artefatos para orientar o desenvolvimento e a manutenção de *software* coorporativo, bem como, a mensuração da qualidade e produtividade.

Fruto do estudo realizado nas práticas dos Órgãos Federais visitados e no referencial teórico, o modelo proposto está fundamentado nas melhoras práticas e artefatos do *framework Scrum*, respeitando às orientações do Modelo de Desenvolvimento de *Software* do EB (MDS-EB).

Dentro da estrutura proposta, os Papéis (atores) envolvidos no processo são (Tabela 5.1):

Tabela 5.1: Atores do processo ágil, Fonte: desenvolvido pelo autor

Papéis (atores)	Responsabilidades
Gerente de Projeto Indivíduo responsável pelo planejamento do projeto, coordena as interações com os clientes e mantém a equipe de projeto focada em alcançar os objetivos do projeto.	▪ Instruir a equipe para conduzir um resultado de êxito no projeto e aceitação do produto pelo cliente; ▪ Responsável pelo resultado do projeto e pela aceitação do produto pelo cliente; ▪ Gerir a avaliação dos riscos do projeto e pelo controle destes riscos através de estratégias de atenuação; e ▪ Aplicar conhecimentos, habilidades, ferramentas e técnicas ágeis de gestão em uma grande quantidade de tarefas a fim entregar o resultado desejado para o projeto dentro dos prazos estabelecidos, recursos planejados e com os padrões de qualidade estabelecidos.
Product Owner (PO) Indivíduo que representa as pessoas que conhecem ou utilizam o processo para o qual o *software* será desenvolvido. É responsável por fornecer as informações que permitam o levantamento dos requisitos do *software*, validar os produtos gerados no processo, bem como receber o *software* desenvolvido.	▪ Representar o cliente final nas decisões sobre os requisitos funcionais do *software*; ▪ Conhecer o processo de negócio e seus objetivos; ▪ Trabalhar com o Time do *Scrum*; ▪ Homologar os produtos e serviços do projeto; ▪ Testar o *software* desenvolvido; ▪ Dar o aceite final ao produto desenvolvido.
Scrum Master Indivíduo experiente no processo ágil que zele pela boa aplicação dos valores no projeto	▪ Conhecer o processo e as práticas de desenvolvimento ágil; ▪ Fazer garantir que as técnicas do *Scrum* sejam aplicadas no projeto; ▪ Proteger a equipe de dificuldades externas, facilitando a resolução de obstáculos; ▪ Realizar treinamento e orientar os demais papéis em relação ao processo *Scrum*.

Time Scrum	
Composição de membros multidisciplinar responsáveis pelo desenvolvimento do software, respeitando o número de 3 a 9 pessoas.	
Analista de Sistemas	▪ Elaborar os documentos do projeto sob sua responsabilidade, conforme os prazos estabelecidos e em conformidade com a Metodologia de desenvolvimento de *software*; ▪ Validar junto ao PO e o Gerente de projetos todos os documentos de requisitos gerados sob sua responsabilidade.
Analista de Pontos por Função	▪ Validar e ou estimar o esforço necessário para o desenvolvimento do *software*.
Arquiteto	▪ Identificar e documentar os aspectos arquiteturalmente significativos do *software* como visões que descrevem requisitos, *design*, implementação e distribuição; ▪ Reduzir os riscos técnicos e assegurar que as decisões sejam eficazmente comunicadas, validadas e seguidas;
Desenvolvedor	▪ Transformar o *design* em código, implementar a estrutura do *software* na linguagem fonte escolhida; ▪ Implementar o comportamento do sistema definido nos requisitos funcionais; ▪ Escrever o código que permita às diferentes partes da aplicação; ▪ Identificar e construir os testes de desenvolvedor que verifique o comportamento desejado dos componentes técnicos.
Administrador de Banco de Dados	▪ Identificar as possibilidades de reuso de dados; ▪ Integrar os modelos de dados de projeto com o modelo de dados corporativo; ▪ Implementar o modelo físico de dados.

Testador	▪ Identificar os testes que necessitem ser executados; ▪ Implementar testes individuais; ▪ Preparar e executar os testes; ▪ Registrar a execução e resultados dos testes; ▪ Analisar e recuperar os erros de execução; e ▪ Comunicar os resultados do teste ao Time.

A Figura 5.1 apresenta o modelo de negócio do Centro de Desenvolvimento de Sistemas do EB (CDS). A metodologia é um recurso indispensável para a entrega do valor proposto aos clientes. Nesse sentido, a proposta da utilização da metodologia ágil, resultado desta pesquisa, tem por objetivo suportar o processo de desenvolvimento de *software* do EB.

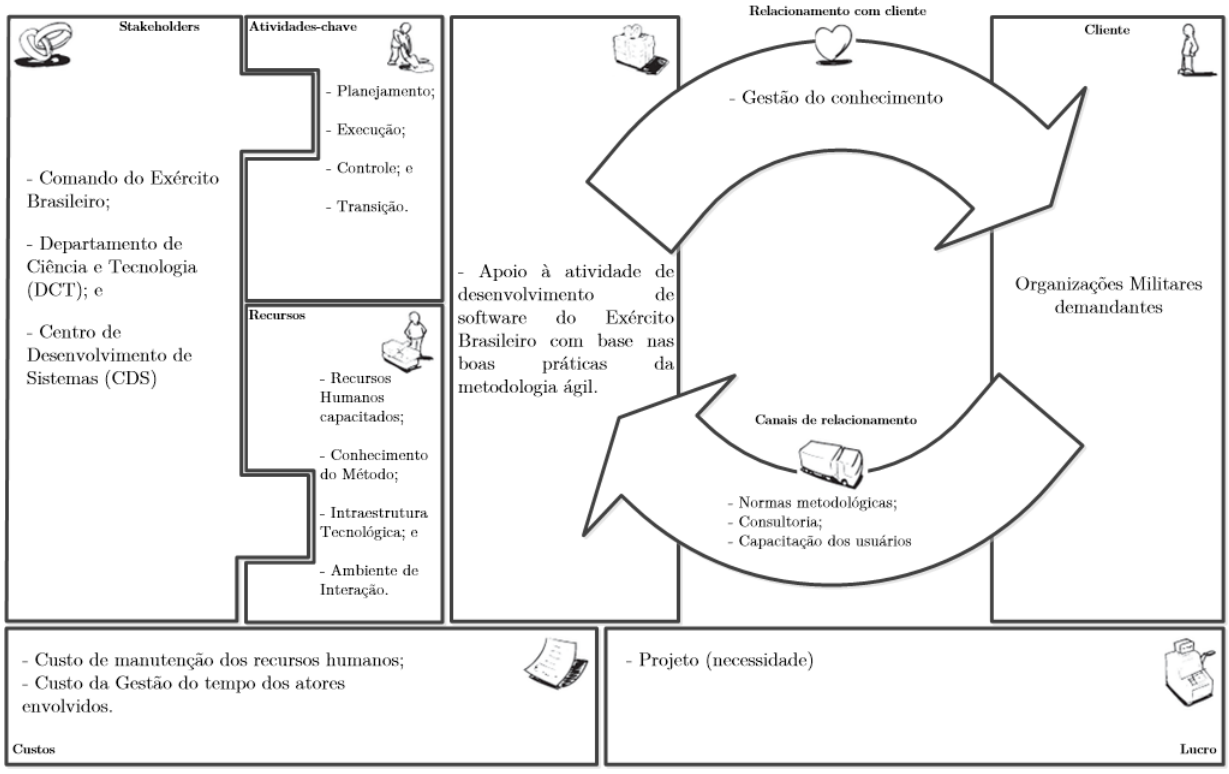

Figura 5.1: CANVAS do Processo de Desenvolvimento Ágil do EB, Fonte: (OSTERWALDER; PIGNEUR, 2009)[47], adaptado pelo autor

É pelo cenário apresentado que a proposta de valor é apoiar atividade de desenvolvimento com base nas práticas ágeis, a qual é entregue às Organizações Militares demandantes por meio de normas metodológicas, prestação de consultoria e capacitação dos usuários. Foi estabelecido que o mecanismo de relacionamento com o cliente está

baseado na gestão do conhecimento que, segundo Nonaka e Takeuchi (1997)[46], ocorre na transformação do conhecimento tácito em explícito.

A gestão do conhecimento está inserida na última fase da metodologia, a de Transição, e que garante a revisão do que foi realizado e a transferência do conhecimento adquirido.

Entende-se por receita o mecanismo necessário para a existência do método. Nos modelos apresentados anteriormente a receita estava totalmente vinculada à necessidade de recurso financeiro, neste caso, a receita é proveniente do projeto em si.

Considerando o sucesso da captação de projeto, os custos incorridos são cobertos em sua totalidade, são eles: o custo de manutenção dos recursos humanos e o custo da gestão do tempo dos atores envolvidos.

Para que a metodologia seja aplicada, dois fatores ainda são necessários, as atividades-chave e os recursos. As atividades-chave correspondem aos grupos de tarefas necessárias à implementação da metodologia ágil proposta e que foram nomeadas fases, as quais estão divididas entre planejamento, execução, ponto de controle e transição. Estas estão descritas no tópico 5.1.1.

Os recursos necessários listados estão agrupados entre Recursos Humanos capacitados, Conhecimento do método, Infraestrutura Tecnológica e Ambiente de Interação. E as partes interessadas neste modelo dizem respeito ao Comando do EB, o Departamento de Ciência e Tecnologia e o próprio Centro de Desenvolvimento de Sistemas.

Para finalizar a apresentação dos três modelos é importante destacar que a interação entre cada um deles ocorre, prioritariamente, por meio da entrega do valor. Isto quer dizer que à medida que a metodologia de desenvolvimento de *software* ágil oferece às OM demandantes a sua proposta de valor, esta impacta na entrega dos sistemas coorporativos e operacionais e na assessoria prestada pelo CDS. Todo este conjunto, posteriormente, será um dos pilares para a agregação de valor proposta pelo EB e entregue à sociedade brasileira do EB.

Este vínculo existente entre os modelos ocorre de forma construtiva quando se trata da metodologia ágil oferecer ao EB insumo suficiente à sua proposta de valor e do CDS ser base para a prestação do serviço do EB. Porém, o outro sentido também se torna verdadeiro ao passo que o modelo é decomposto em cada nível. É papel do EB oferecer a estrutura necessária ao CDS para que ele tenha sucesso em sua entrega e por sua vez o CDS disponibiliza recurso humano e físico, indispensáveis para a aplicação da metodologia.

Estes dois caminhos apontados, de construção de valor vindo da metodologia até o EB e da decomposição do modelo desde o EB até a metodologia são observados em todos os componentes apresentados nas três Figuras 4.1, 4.2 e Figura 5.1, são eles: Cliente, Proposta de Valor, Canais e Relacionamento com o cliente, Atividades-chave, Recursos, Receita, Custos e *Stakeholders*.

O modelo principal proposto pelas Figuras 5.2 e 5.3 é complementado por outros quatro subprocessos. O diagrama da Figura 5.4 apresenta os detalhes do subprocesso que verifica os requisitos necessários para o desenvolvimento do *software* ágil.

Levando em consideração a cultura organizacional do EB, que por sua vez é regida por uma estrutura hierarquizada e extremamente gerenciada por intermédio dos fundamentos do PMBOK, propõe-se uma estrutura de um processo, representado na figura 5.2.

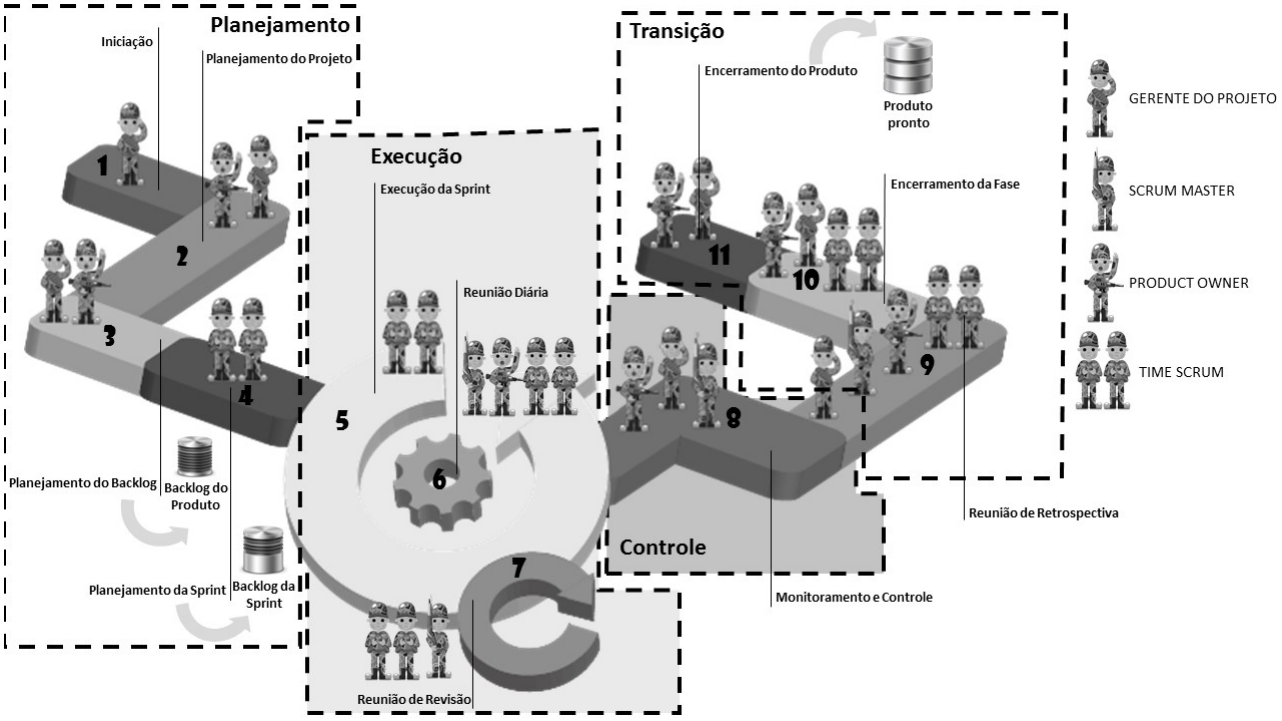

Figura 5.2: Proposta de um Processo de desenvolvimento ágil para o EB x PMBOK, Fonte: produzido pelo autor

O processo de desenvolvimento ágil proposto na figura 5.3 é resultado da necessidade de informatização do processo de negócio observada pelo cliente e transformada na demanda de desenvolvimento de um projeto de *software*. Este está dividido em 4 etapas, são elas, Planejamento da Execução, Execução, Controle e Transição.

A primeira etapa é composta por 08 atividades e 15 artefatos e visa, de maneira geral, garantir o escopo e os meios necessários para se desenvolver o *Software* solicitado. Nesse sentido, a etapa de planejamento é divida em 3 fases: a Iniciação, quando a demanda é recebida, os *stakeholders* identificados e alguns membros da equipe definidos; o Planejamento do Projeto, no intuito de delimitar o escopo do projeto que dará insumo à elaboração dos planos de projeto; o Planejamento do *Backlog*, quando o escopo do produto será definido no sentido de estimar requisitos e custos; e, por fim, o Planejamento da *Sprint*, que, dado o universo de informação coletado, irá, principalmente, estimar as atividades a serem desenvolvidas, tanto em tempo quanto em esforço de produção (Pontos

de Função), além de identificar os riscos e garantir a qualidade. A coordenação destas fases vem oferecer a esta etapa a flexibilidade necessária para geração de valor ao cliente o mais rápido possível, à luz de suas necessidades e em conformidade com a metodologia ágil.

A etapa de Execução possui 08 atividades, que geram 06 artefatos. Tendo como insumo o *Backlog* do Produto, o planejamento da *Sprint* citado anteriormente irá fornecer as informações necessárias para a execução da mesma, destacando as tarefas necessárias para o desenvolvimento de *software*, a exemplo da codificação, desenho do Banco de Dados, entre outros. A reunião diária, como o próprio desenho sugere, é atividade rotineira, com a presença de todo o *Time Scrum* e do PO, para esclarecer as atividades / tarefas executadas no dia anterior e o que será executado no dia corrente, relatando as dificuldades e sanando as dúvidas para o total cumprimento do planejado, é importante ressaltar que seguindo a orientação do método *Scrum* essas reuniões não devem ultrapassar 15 minutos. O acompanhamento da execução da *Sprint* é feito por meio das reuniões de revisão, quando o cliente (*Product Owner*) aprova ou não o resultado alcançado. Cabe destacar que na execução de uma *Sprint* não pode ocorrer mudança dos requisitos inicialmente planejados.

O espaço criado entre a etapa de Execução e a de Transição foi nomeado de Controle e tem o propósito de monitorar e controlar o desenvolvimento tanto do projeto quanto do produto. Sendo assim, o envolvimento dos *Stakeholders*, as Mudanças, o Escopo, os Custos e o Cronograma serão controlados juntamente ao monitoramento do desenvolvimento do projeto.

Finalmente o ciclo processual é concluído na etapa de Transição, com a execução de 04 atividades que resultam em 08 artefatos, composta pela Reunião de Retrospectiva, pelo Encerramento da Fase e do Produto. Neste momento, após o ponto de controle e durante a reunião citada, os Planos do Projeto serão revisados e as Lições Aprendidas registradas. Dada a aprovação das *Sprints* pelo *Time Scrum* e a homologação do produto pelo cliente, o projeto é encerrado.

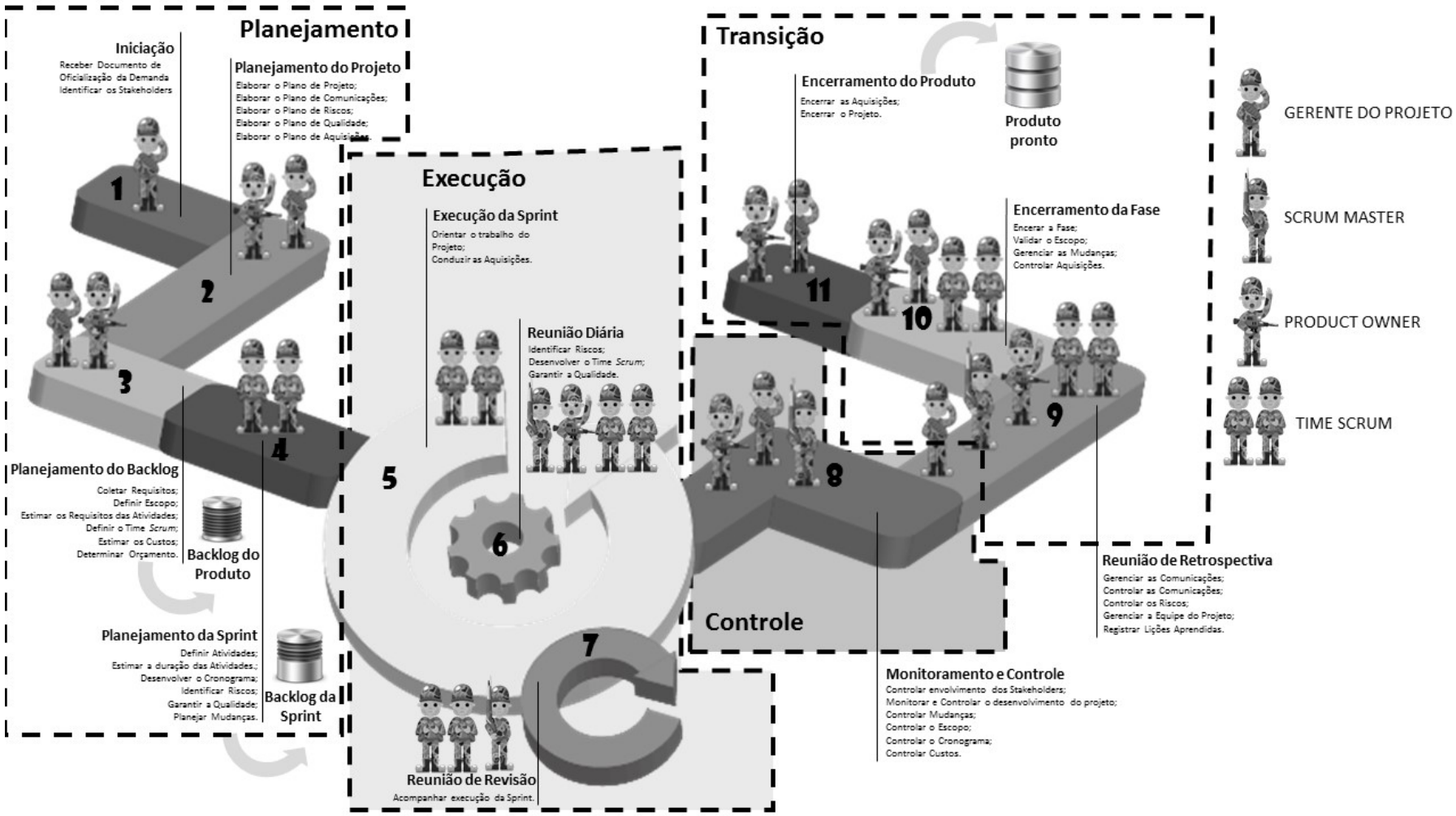

Figura 5.3: Proposta de um Processo de desenvolvimento ágil para o EB x PMBOK, com os artefatos, Fonte: produzido pelo autor

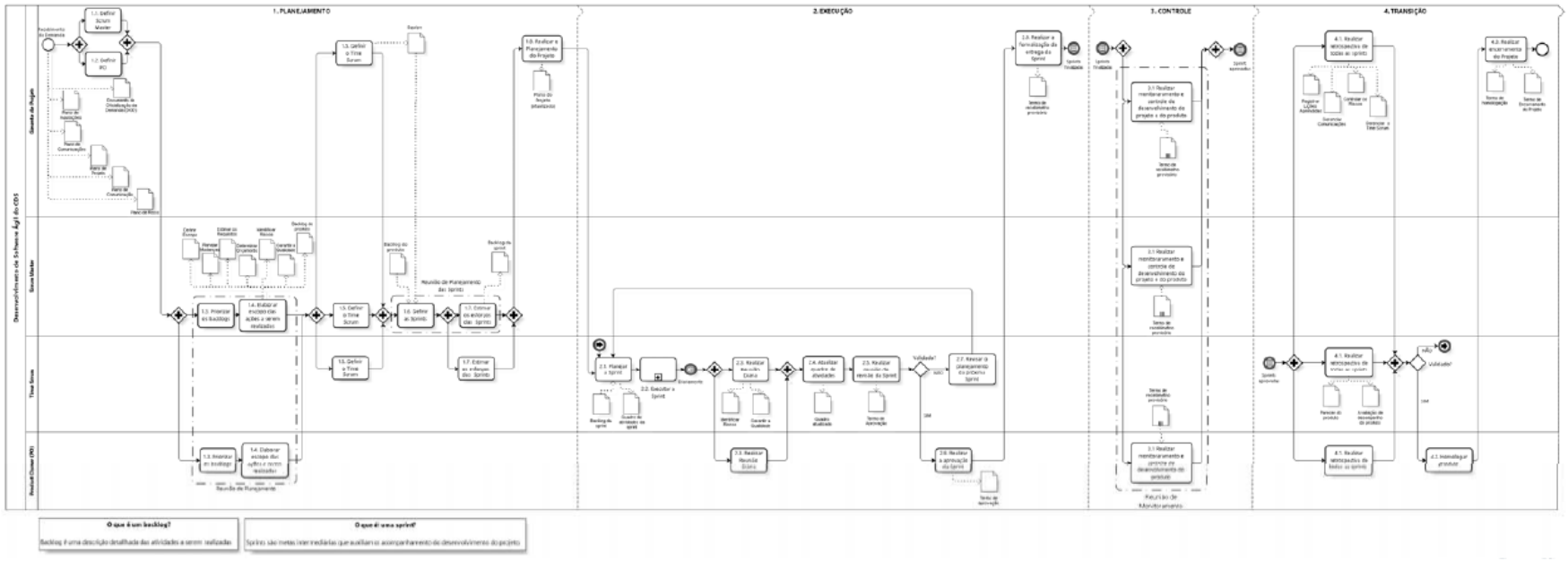

Figura 5.4: Diagrama da Proposta de um Processo de desenvolvimento ágil para o EB x PMBOK, com os artefatos, Fonte: produzido pelo autor.

Por fim, no processo de desenvolvimento ágil do EB é executado o subprocesso de transição com o conjunto de *Sprints* concluídas do subprocesso de execução, possibilitando assim iniciar a atividade de retrospectiva de todas as *Sprints* realizadas no release em conjunto com o Gerente de Projeto, *Time Scrum*, PO e o *Scrum Master*, que resulta num parecer do produto e a avaliação de desempenho do produto. Uma vez aprovado é executado outra atividade de homologar o produto pelo PO e o Gerente do Projeto, com a finalidade de assinar o Termo de homologação seguido do Termo de Encerramento do Pronto. Caso seja reprovado, a determinada release é reiniciada pelo *Time Scrum*, destacando que a homologação do produto só é realizada com a apresentação de toda a documentação produzida no projeto de desenvolvimento, disponibilizado no ambiente de produção do CDS.

Para melhor entendimento do processo de desenvolvimento ágil do EB, no próximo tópico é apresentado com maior especificação às atividades executadas com as etapas, entradas e saídas, resultados esperados, desvios e atitudes, que foram apresentadas na Figura 5.4.

5.1.1 Atividades do Processo de Desenvolvimento

Estão previstas as seguintes atividades:

5.1.1.1 De Planejamento

Atividade 1.1: Definir *Scrum Master*

Descrição: O Gerente de Projeto irá determinar, segundo Documento de Oficialização da Demanda e conforme disponibilidade do efetivo, o participante responsável pela gestão de cada área do projeto a fim de planejar o desenvolvimento do software.

Resultados esperados: Integrante indicado com experiência na execução dos princípios da metodologia ágil *Scrum*. Destacando que a presente função poderá ser executada pela mesma pessoa que exercerá a função de Gerente de Projeto.

Atividade 1.2: Definir PO

Descrição: O Gerente de Projeto irá determinar, segundo Documento de Oficialização da Demanda e conforme disponibilidade do efetivo, o participante capaz de subsidiar as decisões e informações necessárias para a construção da solução a ser proposta, além de

conhecer integralmente as necessidades do cliente.

Resultados esperados: Integrante indicado com domínio do processo organizacional na íntegra, capaz de tomar decisões necessárias para o cumprimento das necessidades negociais da solução proposta.

Atividade 1.3: Priorizar o *Backlog* do Produto

Descrição: Durante a Reunião de Planejamento, o *Product Owner* (PO), com a participação do Scrum Master e do Gerente de Projeto, irá priorizar os itens do *backlog* do produto e ainda indicar os riscos associados, tendo como base o Documento de Oficialização da Demanda.

Resultados esperados: Juntamente com a lista das prioridades do *Backlog* do Produto, ter também realizado a indicação dos riscos da execução das funcionalidades.

Atividade 1.4: Elaborar escopo das ações a serem realizadas

Descrição: Durante a Reunião de Planejamento, o *Product Owner* (PO), com a participação do *Scrum Master*, do Gerente de Projeto e de demais clientes interessados no produto, irá descrever o problema atual, as necessidades e as expectativas que subsidiarão a razão do projeto, considerando as informações presentes no Documento de Oficialização da Demanda bem como no *Backlog* do Produto atualizado e priorizado, resultando assim em uma visão atualizada do produto que tenha sido provada pelo PO.

Resultados esperados: Identificação das dificuldades do projeto com a causa e efeito, garantindo o entendimento do problema e as ações necessárias para o cumprimento do projeto.

Atividade 1.5: Definir Time Scrum

Descrição: O Gerente de Projeto, junto ao *Scrum Master* irá definir a equipe do projeto tendo como base a visão do produto, a tabela de riscos e o *Backlog* do Produto e irá formalizá-la na Lista da equipe do Projeto.

Resultados esperados: Visão do projeto atualizado, juntamente com a tabela de riscos e o *Backlog* do Produto.

Atividade 1.6: Definir Sprints

Descrição: O *Product Owner* e o *Time Scrum* durante a Reunião de Planejamento das *Sprints*, selecionam os itens do *Backlog* do Produto que comporão o escopo da entrega final da Sprint. Para que as metas das *Sprints* sejam definidas, os atores devem entender os objetivos de cada uma delas e definir a arquitetura do Projeto.

Resultados esperados: Definição das metas das *sprints* com definição de indicadores de produtividade e os riscos da execução.

Atividade 1.7: Estimar os esforços das *Sprints*

Descrição: O *Scrum Master* e o *Time Scrum* irão definir o esforço das *Sprints* em termos de tempo e produtividade, fazendo uso das metas que foram definidas anteriormente e do Plano de Execução de cada uma delas. Com isto, irão quantificar os Pontos de Função necessários para a execução da *Sprint* além de atualizar suas metas e Plano de Execução, por fim, irão elaborar o *Backlog* da *Sprint*.

Resultados esperados: Definição do esforço necessário para o cumprimento da *Sprint*.

Atividade 1.8: Realizar o Planejamento do Projeto

Descrição: O Gerente de Projeto irá planejar escopo, objetivos, riscos, duração inicial e os entregáveis do projeto, consultando o *Backlog* do Produto priorizado, o documento de Visão atualizado, o Plano de Execução da *Sprint* e a Tabela de Riscos. O Plano do Projeto será atualizado à medida que este progredir, com base nas informações colhidas sobre o andamento dos trabalhos e mudanças no ambiente.

Resultados esperados: Elaboração do Plano do Projeto com os dados de entrada, especificando as ferramentas de gerenciamento de riscos do projeto.

5.1.1.2 De Execução:

Atividade 2.1: Planejar *Sprints*

Descrição: O *Time Scrum*, em reunião, irá decompor os itens da *Sprint* em tarefas passíveis de serem executadas em um dia, irá ainda definir as responsabilidades de execução das tarefas e o método de acompanhamento do projeto (quadro *Kanban* e gráfico *Burndown*) no intuito de especificar como os itens selecionados para a *Sprint* serão construídos visando cumprir as metas pré-definidas. Para isto, são utilizados o *Backlog* da *Sprint*, as histórias de usuários, os critérios de aceite, as regras do negócio e os padrões de arquitetura e são produzidos o quadro de atividades da *Sprint*, a arquitetura do projeto, a modelagem do banco de dados, os testes unitários e os protótipos de tela.

Resultados esperados: São resultados esperados desta atividade a modelagem do Banco de Dados; produzir os modelos UML necessários para a realização das funcionalidades; atender os padrões de arquitetura; realizar testes unitários; construir as interfaces validadas pelo usuário; definir os métodos de acompanhamento do projeto.

Atividade 2.2: Executar *Sprints*

Descrição: Fazendo uso da história de usuário, dos critérios de aceite, das regras de negócio, dos protótipos de tela, da meta da *Sprint* e dos itens da *Sprint*, o *Time Scrum* irá executar as tarefas necessárias para o cumprimento da *Sprint* respeitando as orientações de qualidade do código, segurança e arquitetura. Através do conjunto de atividades necessárias para a construção dos itens da *Sprint* o produto será então, construído.

Resultados esperados: Produto construído com qualidade e as expectativas do negócio devidamente atendidas.

Atividade 2.3: Realizar reunião diária

Descrição: Diariamente, o *Product Owner* (PO) e o *Time Scrum* se reúnem, preferencialmente em pé, de posse das metas e dos itens da *Sprint* a fim de relatar as atividades executadas no dia anterior; apontar as dificuldades enfrentadas; entrevistar o PO para detalhar o requisito a ser construído; documentar o requisito detalhado na história de usuário; definir critérios de aceite de cada funcionalidade; definir e documentar especificações de tela de interface; validar os requisitos com o PO e, também, identificar

e documentar os itens do glossário do software. Com isto, ambos o quadro de atividade e o gráfico de produtividade serão atualizados.

Resultados esperados: Métodos de acompanhamento atualizados.
Atividade 2.4: Atualizar quadro de atividade

Descrição: O *Time Scrum* identifica as atividades da *Sprint* a serem executadas e, à medida que elas são finalizadas, o quadro de atividades é atualizado. Algumas atualizações podem ocorrer durante a reunião diária, mas o Time irá manter o quadro atualizado durante toda a fase de execução.

Resultados esperados: Quadro de atividade devidamente atualizado diariamente.

Atividade 2.5: Realizar reunião de revisão da **Sprint**

Descrição: Deverão realizar Reunião de revisão da *Sprint* o *Time Scrum* com *Scrum Master*, apresentando as funcionalidades concluídas, demonstrando o que foi realizado e o que não foi e relatando as lições aprendidas, com base nos itens da *Sprint* planejados anteriormente. Por fim, é elaborado o Termo de Aprovação que será submetido ao *Product Owner* (PO).

Resultados esperados: Termo de aprovação devidamente submetido ao *Product Owner* (PO).

Atividade 2.6: Realizar aprovação da *Sprint*

Descrição: O *Product Owner* (PO) irá aprovar ou não a *Sprint* executada quando da apresentação dos itens constituintes da mesma pelo *Time Scrum*.

Resultados esperados: Aprovação ou não da *Sprint*.

Atividade 2.7: Realizar o planejamento da próxima *Sprint*

Descrição: Dado a possível reprovação da *Sprint* o *Time Scrum* deverá realizar o planejamento das adequações solicitadas pelo PO, para isso serão atualizados os Plano de execução e o *Backlog* da *Sprint*.

Resultados esperados: Adequações solicitadas devidamente tratadas.

Atividade 2.8: Realizar formalização da entrega da *Sprint*

Descrição: Após aprovação das *Sprints* executadas e a assinatura do Termo de Aprovação, o Gerente de Projeto e o *Product Owner* (PO) formalizam a entrega do Produto com a elaboração e assinatura do Termo de recebimento provisório pelo Gerente.

Resultados esperados: Assinatura do Termo de Recebimento Provisório.

5.1.1.3 Do Controle

Atividade 3.1: Realizar monitoramento e controle do desenvolvimento do projeto e do produto

Descrição: Após a finalização de todas as *sprints* que foram planejadas para a execução do projeto, o Gerente de Projeto em conjunto com o *Scrum Master* irão apresentar o que foi desenvolvido para o *Product Owner* (PO) durante a Reunião de Monitoramento com o intuito de aprovar os resultados obtidos das *sprints* e com isso ter o Termo de Recebimento Provisório assinado. Essa reunião será a entrada para a próxima etapa de Transição.

Resultados esperados: Assinatura do Termo de Recebimento Provisório.

5.1.1.4 De Transição

Atividade 4.1: Realizar retrospectiva de todas as *Sprint*

Descrição: Gerente de Projeto, Scrum Master e Time Scrum se reúnem para consolidar os resultados das atividades desenvolvida, além de analisar os pontos positivos e negativos durante a execução do projeto. Por fim, é elaborado o Relatório do Projeto com as discussões pontuadas nesta Reunião de Retrospectiva.

Resultados esperados: Lições Aprendidas e Relatório do Projeto.

Atividade 4.2: Homologar produto

Descrição: O *Product Owner* irá validar a arquitetura, os testes, o Banco de Dados a fim de verificar a qualidade do produto quanto aos padrões corporativos e do projeto. Para isto, faz uso do produto que foi disponibilizado no ambiente de homologação, do repositório do projeto atualizado com todos os documentos produzidos além dos artefatos e do código-fonte produzido na *Sprint*. Por fim, o PO elabora o Temo de Homologação.

Resultados esperados: Produtos produzidos dentro dos padrões.

Atividade 4.3: Realizar o encerramento do projeto

Descrição: O Gerente de Projeto junto ao *Product Owner* (PO) formalizam o encerramento do projeto em conformidade com os padrões do processo de Desenvolvimento Ágil através da assinatura do Termo de Encerramento.

Resultados esperados: Encerrar o projeto mediante assinatura do Termo de Encerramento do Projeto pelo Gerente do Projeto e PO.

Concluído o detalhamento do processo de desenvolvimento de *software* ágil do EB, o tópico 5.1.2 apresenta o Plano de Gestão de Riscos do processo, levando em consideração as observações da Secretaria de Fiscalização de Tecnologia da Informação (SEFTI) do Tribunal de Contas da União (TCU) realizada por intermédio do Acórdão n. 2314/2013[19], que realizou o levantamento em cinco Administrações Públicas Federais acerca da utilização de métodos ágeis nas contratações para desenvolvimento de *software*, haja vista que a adoção dos métodos ágeis pela Administração Pública ainda é pouco difundida nacionalmente.

5.1.2 Plano de Gestão de Risco do Processo de Desenvolvimento ágil

O processo de desenvolvimento de *softwares* é uma atividade de risco, uma vez que os problemas ocorridos em projetos de grande porte estão mais associados a falhas em atividades de gerenciamento do que falhas de atividades técnicas. As organizações são muito dependentes do sucesso ou fracasso dos projetos que desenvolvem então à medida que os projetos se tornam mais complexos aumenta a necessidade da utilização de metodologias de gerenciamento de riscos, as quais irão dar mais apoio aos gerentes de projetos no cumprimento das metas do projeto, garantindo a qualidade do produto gerado.

Conforme extraído do Acórdão TCU-Plenário n. 2314/2013[19], observa-se que a grande maioria dos desenvolvimentos de sistemas na Administração Pública segue o Processo Unificado, ou seja, "um conjunto de atividades executadas para transformar um conjunto de requisitos do cliente em um sistema de *software*", sendo que tal processo pode ser personalizado de acordo com as necessidades e recursos de um projeto. Tal processo tem como elementos descrever quem, o que, como e quando será realizada uma tarefa. Ainda, tal desenvolvimento é feito em divisões, chamados ciclos de iteração, onde existe uma análise de requisitos, projeto, implementação e teste; é baseado em casos de uso, ou seja, utiliza casos que já foram experimentados pelos clientes para resolver determinada situação; e é centrado na organização do sistema como um todo, ou seja, tem como alicerce a sua arquitetura.

No entanto, o próprio corpo técnico do TCU, no relatório que fundamenta o Acórdão, relata que o Processo Unificado e suas variantes (como o RUP que, em tradução livre, significa Processo Unificado da Rational) não são aplicados conforme seus preceitos iniciais, ou seja, de iterações e incrementos no sistema desenvolvido, sendo que, nos desenvolvimentos mais comuns no Brasil, é adotada uma postura de desenvolvimento que se assemelha a uma cascata: de maneira sequencial, executado em uma única vez, sem os *feedbacks* e as entregas sequenciais e incrementais que o processo unificado pretende.

Neste contexto, surgem as Metodologias Ágeis, representadas por um conjunto de valores e princípios a serem seguidos por desenvolvedores, reunidos na forma de um Manifesto, em 2001. Suas bases, embora pareçam modernas, remontam aos tempos da Segunda Guerra Mundial e à Manufatura Enxuta criada pela Toyota com o objetivo de aumentar a eficiência da produção, com base em práticas de como evitar o desperdício, construir o necessário, realizar entregas contínuas e estar aberto a eventuais mudanças.

Segundo o Relatório do TCU, os seguintes princípios do Manifesto Ágil podem ser descritos como, em pequena adaptação de linguagem: i) satisfazer o cliente através da entrega contínua de *software* de valor; ii) aceitar mudanças de requisitos, de forma a propiciar vantagens competitivas; iii) entregar frequentemente os sistemas, em escalas de semanas e meses, preferindo períodos curtos; iv) pessoas relacionadas ao negócio e desenvolvedores devem trabalhar em conjunto e diariamente; v) utilizar indivíduos motivados, confiando que eles farão o trabalho, dando-lhes ambiente e ferramentas; vi) priorizar as conversas "cara a cara"; vii) "*software* funcional é a medida primária do progresso"; viii) processos ágeis promovem um ambiente sustentável; ix) atenção à excelência técnica e ao design aumenta a agilidade; x) simplicidade; xi) "as melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis"; xii) o time reflete como ser mais efetivo.

Como apontamento, a SEFTI do TCU elencou dada à natureza dos negócios com a

União, três principais áreas de riscos, que totalizam 16 riscos no uso da metodologia ágil pela APF.

Uma vez entendida a diferença da utilização da Metodologia Ágil para os métodos tradicionais, entende-se a necessidade de se trabalhar de forma a melhorar o processo de contratação e desenvolvimento interno de tecnologias dos órgãos do serviço público. Entretanto, para agir nesse sentido, é necessária atenção às resoluções contidas no Acórdão TCU-Plenário n. 2314/2013 (BRASIL, 2013)[18].

Sendo assim, foram levantados insumos que traduziram algumas das melhoras práticas encontradas em diversos órgãos que utilizam dessa metodologia e, com base nessa informação, procurou-se traduzir esse modelo para uma solução aplicável ao Exército.

O primeiro passo para desenvolvimento do projeto foi voltado para a assimilação e reconhecimento da metodologia ágil como um todo. Essa análise possibilitou o entendimento e a necessidade de se migrar para uma nova forma, mais eficiente e vantajosa de gerenciamento de projetos, que facilitaria a forma como é feita a contratação de projetos e desenvolvimento de *software* no serviço público.

Entretanto, para que as mudanças sejam feitas, é preciso entender o objeto a ser alterado e como é o seu funcionamento antes da adoção do método proposto. Foi feita então um estudo do processo atual do EB, apresentado na seção 4.2, enxergando que o processo apresenta uma estrutura de planejamento extensivo, onde não podem ser feitas alterações após a aprovação do planejamento, ocorrendo riscos e retrabalhos.

Levando em conta toda essa atividade de pesquisa, foi desenvolvido o questionário que visou levantar todas as informações coerentes com a problemática. Esse questionário foi feito de forma presencial, durante as entrevistas, e virtualmente, através de plataforma escolhida a fim de melhorar a coleta de dados. Esses dados coletados compõem respectivamente as informações qualitativas do projeto.

As premissas para a utilização de metodologias ágeis no desenvolvimento interno e/ou externo das organizações seriam o investimento em treinamento adequado para toda a equipe que participa diretamente dos processos, a utilização adequada dos recursos visuais e a atualização de todas as informações para que nenhum dado de qualquer origem seja perdido ou fique defasado, de acordo com o processo. E também a liberdade dos próprios analistas solicitarem o treinamento para a disciplina que sentirem necessidade ao longo do projeto, ou antes, de iniciar a execução de um novo projeto.

A participação e o envolvimento de toda a área nos estudos de como aplicar as metodologias ágeis nos processos internos e uma segmentação da equipe, dividindo-a em setores os quais cuidariam de cada parte do processo, ou de um processo em si também é válida e importante para o desenvolvimento das atividades. Mas não necessariamente sempre as mesmas equipes nas mesmas áreas, e sim, a divisão das tarefas entre a equipe

toda.

Com o conhecimento mais aprofundado da utilização dessas metodologias, é mais fácil realizar a divisão dos papéis de cada um e cada papel se agrupa com os outros, a finalidade de montar um time o qual desenvolverá o projeto.

Atrelado aos projetos os quais irão sendo desenvolvidos e finalizados, a criação de um portfólio com os projetos internos que já utilizaram ou estão utilizando metodologias ágeis é útil para a disseminação do conhecimento.

Com o desenvolvimento de projetos simultâneos, surgirão alguns que são prioritários em relação a outros. Frente a isso, é necessária a intervenção e o envolvimento de funcionários com poder de decisão, sendo criado um comitê gestor, decidindo o que tem de ser feito primeiro em cada etapa dos projetos.

Dentre as equipes que são divididas, é necessária a criação de uma equipe que estime o tempo que cada atividade durará. Isso também será um ponto de análise para priorização das atividades. A partir dos prazos e prioridades definidas, os times assumem os projetos, participando do andamento do projeto o analista de negócios, a área de desenvolvimento que irá desenvolver os sistemas, a área de testes que irá testar os sistemas, a área de qualidade que irá analisar a qualidade dos sistemas e o andamento do projeto periodicamente e a área de arquitetura que irá estruturar os sistemas os quais serão desenvolvidos.

É importante dar às equipes certa autonomia para trabalharem com a metodologia ágil, a qual se adaptarem melhor para o desenvolvimento das atividades não engessando os processos com certo tipo de metodologia.

Com os métodos ágeis no desenvolvimento interno, é necessário ter uma área de Qualidade, que irá planejar e controlar certos parâmetros de processos: Processo de Melhoria Contínua, Contramedidas para minimizar riscos e aperfeiçoar o projeto, Qualidade de produto e também de projetos.

O desenvolvimento de *softwares* para órgãos do Serviço Público Federal não é considerado uma atividade finalística na grande maioria dos órgãos públicos. A exceção ocorre nos órgãos que tem como finalidade o desenvolvimento, acompanhamento e controle de *software* de gestão pública, notadamente o Serviço Federal de Processamento de Dados (SERPRO) e a Secretaria do Tesouro Nacional (STN), esta última por ter desenvolvido e ser gestora do Sistema Integrado de Administração Financeira do Governo Federal (SIAFI), poucos são as entidades cujo processo de negócio é o desenvolvimento de programas e ferramentas para outros órgãos ou para consumo interno.

Nesta lógica, o desenvolvimento de sistemas internos ou externos por um órgão federal pode se dar de forma contratada. Nesta modalidade, o órgão estabelece suas diretrizes, disponibiliza um instrumento público de concorrência, recebe propostas e escolhe aquela

que apresentar a melhor viabilidade de acordo com suas premissas, seja por ter apresentado melhor preço ou por demonstrar melhor técnica, seguindo os preceitos da legislação vigente, em especial a Lei Federal n. 8.666/93 (BRASIL, 1993)[14]. A empresa vencedora deste certame passa a fornecer um serviço à contratante, passando a ser chamada, no jargão da área, de fábrica de *software*.

As fábricas de *software* precisam seguir vários pontos à risca para poderem ser remuneradas pelo trabalho desenvolvido para a contratante. É necessário que ela se atenha à solicitação da intervenção proposta, aos requisitos que um sistema necessite ter, é preciso entregar um *software* operacionalmente estável e que atenda aos objetivos do negocio da contratante. Para tal, os normativos estabelecem um regime de verificação e validação do funcionamento e, também, de uma métrica específica que é utilizada para que se contabilize o real esforço empregado pela empresa, de forma a remunerá-la justa e corretamente por seu trabalho. Tal métrica é chamada Ponto de Função e foi estabelecida por Allan J. Albrecht, da IBM, por meio da publicação "Medindo a Produtividade do Desenvolvimento de Aplicativos" (1979)[**?**] e vem sendo amplamente utilizada na análise de *software* e atribuição do seu real valor.

5.1.2.1 Contexto do processo de gestão de riscos

Para fins do presente plano, deve-se adotar a seguinte conceituação (ABNT, 2009)[2], (ABNT, 2012)[3]

1. RISCO de TI- possível evento que representa uma ameaça em potencial aos processos de contratação e/ou desenvolvimento de *software* interno e que pode se concretizar por meio da exploração de uma ou mais incerteza, causando impacto nos objetivos e, por conseguinte, os objetivos das demais fases dos Projetos que dele dependam.
2. ANÁLISE DE RISCOS de TI- análise realizada sobre as fases do processo de desenvolvimento cuja primeira etapa é realizar o planejamento e, em relação a esses recursos, determinar o Plano de: Projeto, Comunicações, Riscos, Qualidade e Aquisições. Essa análise pode ser quantitativa ou qualitativa dependendo da metodologia empregada.
3. ESTIMATIVA DO VALOR DO RISCO de TI- processo que associa um valor ao risco identificado na análise de riscos.
4. MATRIZ DE RISCOS - Matriz que relaciona uma associação de valores de impacto e chances de concretização de uma ameaça com um valor de risco.

5. GESTÃO DE RISCOS - processo que visa manter os riscos em patamares aceitáveis para o processo a que é aplicado e que é realizada por meio do seguinte processo: análise de riscos;
6. IDENTIFICAÇÃO DO RISCO - processo que visa identificar os riscos que o processo de desenvolvimento oferece.
7. ANÁLISE QUALITATIVA DE RISCOS - análise de riscos que estima o valor dos riscos por meios não estatísticos, ou seja, por estimativas fornecidas por especialistas de área.
8. ANÁLISE QUANTITATIVA DE RISCOS - análise de riscos que conta com dados em quantidade e qualidade tal que seja possível utilizar técnicas estatísticas para calcular e interpretar o risco.

5.1.2.2 Objetivo do Plano de Gestão de Riscos no CDS

O Plano de Gestão de Riscos (PGR) tem como objetivo documentar as ações estratégicas de tratamento dos riscos do processo de desenvolvimento de *software* do CDS.

Dentre os objetivos específicos do Plano, destaca-se:

1. Gerar critérios para tomada de decisão sobre ações a serem executadas no gerenciamento de risco.
2. Prover referenciais doutrinários sobre gestão de risco.
3. Orientar a execução de processos de análises de risco qualitativas nos processos de desenvolvimento de *software* do CDS.
4. Prover um mecanismo útil na aplicação de processos de análise e identificação de riscos.
5. Estabelecer as principais responsabilidades no processo de análise de riscos.

5.1.2.3 Objetivos estratégicos do EB com relação a TI

Plano Estratégico do Exército – PEEx

1. Objetivo: Reestruturar o Sistema de Informação do Exército
2. Estratégias Aperfeiçoamento do Sistema de Tecnologia da Informação do Exército.
3. Ações Estratégicas Fortalecer o papel do DCT como gestor da TI.
4. Projetos em curto prazo Promover a transformação do CDS. (AçExec) (DCT).

5. Objetivos Estratégicos do CDS Aumentar a capacidade de desenvolvimento de *software* e de acompanhamento de contratos de TI.

6. Ameaças Falta de estruturação dos Processos de desenvolvimento de *software*, o que gera muito atraso nos projetos.

5.1.2.4 Responsabilidades

O pessoal envolvido no processo deverá ser selecionado e autorizado de acordo com o prescrito nas Instruções Gerais da Organização para implementar e manter o processo de gestão de riscos e nos instrumentos normativos e legais, além de outras legislações ou documentos normativos internos e ou externo que se façam necessários.

O pessoal técnico designado para aplicar o processo de análise de riscos deverá ser escolhido conforme o perfil técnico necessário.

O Chefe do CDS deverá designar um militar que fará os levantamentos iniciais para identificar o perfil técnico necessário às situações específicas a serem abordadas no processo de análise e, assim, tornar precisa a indicação dos técnicos que executarão o processo.

5.1.2.5 Identificação dos riscos

A identificação do risco visa caracterizar o evento que, em decorrência da exploração de uma incerteza, pode redundar em um impacto negativo ao processo.

Como etapa inicial do processo para a identificação dos riscos, é necessária a sua categorização. A divisão dos escopos possíveis em que os riscos devem ser abordados visa facilitar o processo da análise de riscos. Como referencial inicial, deve-se levar em conta as seguintes categorias:

1. TÉCNICOS;
2. HUMANOS;
3. AMBIENTAIS;
4. ADMINISTRATIVOS;
5. PROJETOS; e
6. EXTERNOS - categoria que leva em consideração os riscos à informação ou aos recursos informacionais advindos de fatores externos à atividade do CDS em que a análise de riscos estiver sendo aplicada.

As técnicas passíveis de aplicação para identificar o risco são inúmeras, o que faz com que a escolha da técnica seja resultante das particularidades do ambiente analisado, assim como a experiência dos condutores do processo.

5.1.2.6 Avaliação de riscos

Na análise qualitativa dos riscos estabelece prioridades de riscos para análise ou ação adicional. Esse processo avalia a prioridade dos riscos identificados, usando sua probabilidade de ocorrência e o impacto correspondente nos objetivos do projeto, se os riscos ocorrerem.

No que diz respeito à análise quantitativa dos riscos o objetivo é analisar numericamente o efeito dos riscos identificados nos objetivos do projeto. Este processo geralmente segue-se à análise qualitativa dos riscos.

5.1.2.7 Análise de riscos

As técnicas que devem ser utilizadas para preenchimento do Plano de Gestão de Riscos, são:

1. REUNIÕES DE *BRAINSTORMING*. O gerente do processo dirigirá uma discussão na qual gerentes, especialistas, usuários, além de outros integrantes da organização poderão, sem críticas dos demais participantes, falar sobre o que identificam como risco;
2. REVISÃO CRÍTICA DE DOCUMENTAÇÃO. Pesquisa e análise dos processos e procedimentos documentados, relativos ao escopo analisado, e de documentações relativas a outras análises de risco já realizadas, devendo ser dada especial atenção às medidas de tratamento do risco que foram estabelecidas e o seu cumprimento. A revisão tem como objetivo descobrir informações sobre características importantes do processo e que podem gerar algum tipo de risco;
3. LISTAS DE VERIFICAÇÃO. Esta técnica faz uso de listagens onde estão registrados, em geral, em forma de perguntas, os estados em que devem estar as informações ou recursos informacionais críticos. O objetivo desta técnica é, por meio da aplicação da lista de verificação, verificar se os riscos aventados pelas perguntas se confirmam;
4. TÉCNICA DE *DELPHI*. Esta técnica é baseada na busca de um consenso entre especialistas de área que opinam, por meio de questionários, sobre o escopo analisado. Esta técnica tem por objetivo estimar as chances de um risco ser explorado; e
5. ENTREVISTAS. Contato direto do gerente do projeto com quem detém as informações necessárias. Esta técnica tem por objetivo melhor caracterizar pontos do projeto.

5.1.2.8 Indicadores de quantidade

O processo de aplicação da análise de riscos quantitativa fará uso de técnicas estatísticas e pode ser empregado quando os dados disponíveis são em número suficiente para tratamento estatístico e, em especial, em situações em que a análise de riscos qualitativa se revele insuficiente.

5.1.2.9 Indicadores de qualidade

O processo de aplicação da análise de riscos qualitativa é como se segue:

1. Estimar as chances de um risco se concretizar. As técnicas que podem ser utilizadas para a escolha dos valores são: técnica de *Delphi*, caso se conte com mais de um especialista de área, ou entrevista com o(s) especialista(s) que entenda de contratação de serviços;
2. Estimar o tipo de impacto sobre o escopo analisado. O fecho da análise de riscos qualitativa fornece a lista dos riscos estimados, com destaque para aqueles que forem considerados prioritários, assim como aqueles que devam passar por análises adicionais, como a análise de riscos quantitativa.

5.1.2.10 Tratamento de riscos

Cada risco detectado deve ser tratado de acordo uma das seguintes estratégias:

1. NEUTRALIZAÇÃO - consiste na modificação do uso ou do tipo de recurso informacional, nas condições ambientais ou qualquer outros fatores que tenham como consequência a eliminação da causa que está gerando o risco.
2. MITIGAÇÃO - consiste em medidas que diminuam o impacto e/ou as chances de um risco se concretizar.
3. ACEITAÇÃO - consiste na aceitação do risco tal como foi estimado sem medidas adicionais para seu controle.

A sequencia de ações para efetuar o controle apropriado deverá ser a seguinte:

1. a partir dos resultados da análise de riscos contidos no relatório, estabelecer quais os tipos de estratégias serão implementadas para cada risco detectado;
2. as responsabilidades pela execução das ações devem ser formalmente atribuídas;

3. estabelecimento das prioridades para tratamentos dos riscos conforme seu grau de severidade;
4. identificação das medidas de controle possíveis;
5. avaliação da viabilidade técnica e financeira, assim como outro critério que se faça necessário, das medidas de controle possíveis.

Nesse sentido no próximo tópico será apresentada a proposta de mitigação dos riscos apontados pelo TCU, por intermédio do Acórdão n. 2314/2013 (BRASIL, 2013b)[18].

5.1.2.11 Proposta de Análise dos riscos identificados pelo TCU

Visando encontrar formas de mitigar os riscos relacionados pelo TCU por intermédio do Acórdão n. 2314/2013 e analisados pelas APFs visitadas e pelo CDS, conforme os resultados obtidos no tópico 4.4.1 é proposto as seguintes ações, vide Tabela 13:

Tabela 5.2: Plano de Risco proposto, Fonte: desenvolvido pelo autor

Item	Risco	Probabilidade	Dano	Estratégia de Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
Riscos relativos a processos								
1.	Contratação de desenvolvimento de *software* com adaptação de metodologia ágil que desvirtue sua essência.	Media	Baixo	Neutralizar	▪ Garantir que o Gerente do Projeto e o *Scrum Master* sejam capacitados na metodologia ágil, previamente à assinatura do contrato.	▪ Gerente do Projeto ▪ *Scrum Master*	▪ O processo de gerenciamento ou de desenvolvimento deve ser normativo; e ▪ Estabelecer prazo curto de *sprint*, a fim de estimular o desenvolvimento.	▪ Gerente do Projeto
2.	Alteração da metodologia ágil adotada no instrumento convocatório no decorrer da execução contratual.	Baixa	Alto	Neutralizar	▪ Garantir a participação do Gerente do Projeto na fase de Planejamento da Contratação	▪ Gerente do Projeto	▪ O processo de gerenciamento ou de desenvolvimento deve ser normativo.	▪ Chefe do CDS
3.	Ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual.	Baixa	Alto	Neutralizar	▪ Garantir que na fase de planejamento do desenvolvimento o Gerente do Projeto, juntamente com PO elabore o *Backlog* do Produto.	▪ Gerente do Projeto; ▪ PO	▪ O *Product Owner* (PO) deve ser um servidor disponível da área demandante, uma vez que ele pode compreender melhor a necessidade da intervenção e, ao final, verificar e validar cada entrega; ▪ Evitar a substituição do PO, a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ O PO deverá realizar a priorização do trabalho, selecionando quais *sprints* a equipe deseja que sejam entregues primeiro, mantendo sempre o foco nas principais histórias do *backlog*. ▪ A quebra de projetos grandes, que demandem grande espaço de tempo em projetos menores que representem uma etapa útil.	▪ Gerente do Projeto; e ▪ PO

Item	Risco	Probabilidade	Dano	Estratégia de Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
4.	Exigência de artefatos desnecessários ou que se tornam obsoletos rapidamente.	Baixa	Alto	Neutralizar	▪ Trabalhar adequadamente com o PO o *Backlog* do Produto, especificando as prioridades.	▪ Gerente do Projeto; ▪ PO	▪ O PO deverá realizar a priorização do trabalho, selecionando quais pacotes de trabalho a equipe deseja que sejam entregues primeiro, mantendo sempre o foco nas principais histórias do *backlog*. ▪ A quebra de projetos grandes, que demandem grande espaço de tempo em projetos menores que representem uma etapa útil.	▪ PO
5.	Utilização de contrato para desenvolvimento de *software* por metodologias tradicionais para desenvolvimento por métodos ágeis.	Baixo	Alto	Neutralizar	▪ Garantir o fiel cumprimento do Termo de Referência (TR).	▪ Gerente do Projeto	▪ O processo de gerenciamento ou de desenvolvimento deve ser documentado, publicado e normatizado.	▪ Gerente do Projeto

Item	Risco	Probabilidade	Dano	Estratégia de Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
Riscos relativos a pessoas								
6.	Falta de comprometimento ou colaboração insatisfatória do responsável indicado pela área de negócios *Product Owner* no desenvolvimento do *software*.	Media	Alto	Mitigar	▪ Garantir que na atividade de indicação do PO o demandante indique o militar que tenha conhecimento do negócio e autorização para tomada de decisão.	▪ Chefe do CDS	▪ O *Product Owner* (PO) deve ser um militar disponível da área demandante, uma vez que ele pode compreender melhor a necessidade da intervenção e, ao final, verificar e validar cada entrega; ▪ Deve-se escolher um PO que possibilite uma grande integração de toda a área demandante ao projeto, para evitar que o produto tenha um viés fortemente influenciado por apenas uma pessoa; ▪ Evitar a substituição do PO, a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ Escalonar as ausências previstas (férias, por exemplo), combinando-as com períodos onde não haja o levantamento de requisitos ou a validação de liberações produzidas, a fim de evitar atrasos pela ausência do mesmo; ▪ O PO deverá realizar a priorização do trabalho, selecionando quais pacotes de trabalho a equipe deseja que sejam entregues primeiro, mantendo sempre o foco nas principais histórias do *backlog*.	▪ Gerente do Projeto

Item	Risco	Probabilidade	Dano	Estratégia do Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
7.	Falta do conhecimento necessário do indicado pela área de negócios *Product Owner* para o desenvolvimento do *software*.	Media	Alto	Mitigar	▪ Garantir que na atividade de indicação do PO o demandante indique o militar que tenha conhecimento do negócio e autorização para tomada de decisão.	▪ Chefe do CDS	▪ O processo de gerenciamento ou de desenvolvimento deve ser normativo; ▪ O *Product Owner* (PO) deve ser um militar disponível da área demandante, uma vez que ele pode compreender melhor a necessidade da intervenção, mobilizar os colegas da área e, ao final, verificar e validar cada entrega; ▪ O PO deve ser capacitado na metodologia que se está utilizando. Assim, ele terá conhecimento suficiente para entender como se dará o desenvolvimento, poderá se planejar para as reuniões e validações e se torna, realmente, o "dono" do produto; ▪ Evitar a substituição do PO, a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ O PO deverá realizar a priorização do trabalho, selecionando quais pacotes de trabalho a equipe deseja que sejam entregues primeiro, mantendo sempre o foco nas principais histórias do *backlog*.	▪ Gerente do Projeto; ▪ *Time Scrum*
8.	Excessiva dependência da visão do indicado pela área de negócios *Product Owner*.	Alta	Baixa	Aceitar	▪ Realizar a capacitação do PO na metodologia ágil, afim de conscientizá-lo dos princípios ágil.	▪ *Scrum Master*	▪ Deve-se escolher um PO que possibilite uma grande integração de toda a área demandante ao projeto, para evitar que o produto tenha um viés fortemente influenciado por apenas uma pessoa;	▪ Gerente do Projeto

Item	Risco	Probabilidade	Dano	Estratégia do Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
9.	Equipe da empresa contratada não ter expertise em desenvolvimento de *software* com métodos ágeis.	Baixa	Alto	Mitigar	▪ Garantir o perfil do *Time Scrum* no Termo de Referência (TR)	▪ Gerente do Projeto; ▪ Equipe de Planejamento da COntratação	▪ Estabelecimento de prazo curto de *sprint*, a fim de estimular o desenvolvimento; ▪ Quando o desenvolvimento for realizado de forma contratada, a contratação de fábrica de *software* por projeto pode vir a encarecer o mesmo, devido a mudanças na ordem de priorização, nos parâmetros de aceitação, dentre outras ocorrências. A solução encontrada é a da contratação de uma fábrica de *software* com um banco anual de pontos de função, que são consumidos à medida que são realizadas intervenções e desenvolvimentos.	▪ Gerente do Projeto
10.	Dificuldade de comunicação entre a equipe de desenvolvimento da contratada com o indicado pela área de negócios *Product Owner*.	Media	Alto	Mitigar	▪ Realizar a capacitação do PO, com objetivo de conscientizá-lo do método; ▪ Indicar Militar que tenha disponibilidade.	▪ Gerente do Projeto; ▪ *Scrum Master*	▪ O *Product Owner* deve ser capacitado na metodologia que se está utilizando. Assim, ele terá conhecimento suficiente para entender como se dará o desenvolvimento, poderá se planejar para as reuniões e validações e se torna, realmente, o "dono" do produto; ▪ Evitar a substituição do PO, a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ Escalonar as ausências previstas (férias, por exemplo), combinando-as com períodos onde não haja o levantamento de requisitos ou a validação de liberações produzidas, a fim de evitar atrasos pela ausência do mesmo.	▪ Gerente do Projeto; ▪ *Scrum Master*

Item	Risco	Probabilidade	Dano	Estratégia do Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
Riscos relativos a produtos								
11.	Alteração constante da lista de funcionalidades do produto.	Baixa	Alto	Aceitar	▪ Garantir o cumprimento do princípio do método ágil.	▪ PO	▪ O processo de gerenciamento ou de desenvolvimento deve ser normativo; ▪ O *Product Owner* (PO) deve ser um militar disponível da área demandante, uma vez que ele pode compreender melhor a necessidade da intervenção e, ao final, verificar e validar cada entrega; ▪ O *Product Owner* (PO) deve ser capacitado na metodologia que se está utilizando (seja uma metodologia consagrada ou uma adaptação); ▪ Deve-se escolher um PO que possibilite uma grande integração de toda a área demandante ao projeto, para evitar que o produto tenha um viés fortemente influenciado; ▪ Evitar a substituição do PO, a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ O PO deverá realizar a priorização do trabalho, selecionando quais pacotes de trabalho a equipe deseja que sejam entregues primeiro, mantendo sempre o foco nas principais histórias do *backlog*; ▪ A quebra de projetos grandes, que demandem grande espaço de tempo em projetos menores que representem uma etapa útil. Projetos muito grandes tendem a consumir muitos recursos, propiciar muitas mudanças nos requisitos (ou até no escopo) e até no *Product Owner*, dada a	▪ Gerente do Projeto; *Scrum Master*

Item	Risco	Probabilidade	Dano	Estratégia do Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
							rotatividade grande de militares, tanto em remanejamento interno quanto em desligamento da instituição;	
12.	Iniciação de novo ciclo sem que os produtos construídos na etapa anterior tenham sido validados.	Baixo	Alto	Mitigar	▪ Realizar a capacitação da Equipe do Projeto com o uso das ferramentas de acompanhamento do projeto.	▪ Gerente do Projeto; ▪ *Scrum Master*	▪ O processo de gerenciamento ou de desenvolvimento deve ser normativo; ▪ O *Product Owner* (PO) deve ser capacitado na metodologia que se está utilizando (seja uma metodologia consagrada ou uma adaptação). Assim, ele terá conhecimento suficiente para entender como se dará o desenvolvimento, poderá se planejar para as reuniões e validações e se torna, realmente, o "dono" do produto; ▪ Deve-se escolher um PO que possibilite uma grande integração de toda a área demandante ao projeto, para evitar que o produto tenha um viés fortemente influenciado por apenas uma pessoa; ▪ Evitar a substituição do PO, a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ Escalonar as ausências previstas (férias, por exemplo), combinando-as com períodos onde não haja o levantamento de requisitos ou a validação de liberações produzidas, a fim de evitar atrasos pela ausência do mesmo.	▪ Gerente do Projeto; ▪ *Scrum Master*

Item	Risco	Probabilidade	Dano	Estratégia de Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
13.	Falta de planejamento adequado do *software* a ser construído.	Baixa	Alto	Mitigar	▪ Garantir a execução da fase de planejamento do *Backlog* do Produto	▪ Gerente do Projeto; ▪ PO	▪ O processo de gerenciamento ou de desenvolvimento deve ser normativo; ▪ O *Product Owner* (PO) deve ser capacitado na metodologia que se está utilizando (seja uma metodologia consagrada ou uma adaptação). Assim, ele terá conhecimento suficiente para entender como se dará o desenvolvimento, poderá se planejar para as reuniões e validações e se torna, realmente, o "dono" do produto; ▪ Deve-se escolher um *Product Owner* (PO) que possibilite uma grande integração de toda a área demandante ao projeto, para evitar que o produto tenha um viés fortemente influenciado por apenas uma pessoa; ▪ Evitar a substituição do *Product Owner* (PO), a fim de que haja harmonia no desenvolvimento de todo o projeto; ▪ Escalonar as ausências previstas (férias, por exemplo), combinando-as com períodos onde não haja o levantamento de requisitos ou a validação de liberações produzidas, a fim de evitar atrasos pela ausência do mesmo; ▪ O *Product Owner* (PO) deverá realizar a priorização do trabalho, selecionando quais pacotes de trabalho a equipe deseja que sejam entregues primeiro, mantendo sempre o foco nas principais histórias do	▪ Gerente do Projeto; ▪ *Scrum Master;* ▪ PO

Item	Risco	Probabilidade	Dano	Estratégia de Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
							backlog ▪ A quebra de projetos grandes, que demandem grande espaço de tempo em projetos menores que representem uma etapa útil. Projetos muito grandes tendem a consumir muitos recursos, propiciar muitas mudanças nos requisitos (ou até no escopo) e até no *Product Owner* (PO), dada a rotatividade grande de servidores, tanto em remanejamento interno quanto em desligamento da instituição;	
14.	Pagamento pelas mesmas funcionalidades do *software* mais de uma vez, em virtude de funcionalidades impossíveis de serem implementadas em um único ciclo, ou em virtude da alteração de funcionalidades ao longo do desenvolvimento do *software*.	Baixa	Alto	Neutralizar	▪ Garantir a participação do *Time Scrum* um profissional capacitado em Ponto por Função.	▪ Gerente do Projeto	▪ Para que os sistemas entregues sejam avaliados de forma técnica e eficiente, faz-se necessária a criação de um conceito de Fábrica de Testes e de Fábrica de Métricas, que podem ser compostas por militares ou podem ser contratadas, mediante os ditames dos normativos vigentes. A Fábrica de Testes visa submeter o software entregue a testes de estresse ou de conformidade, de forma a verificar se o mesmo responde segundo o padrão de aceitação solicitado e se há algum tipo de problema que trave ou impeça o regular funcionamento do sistema. A Fábrica de Métricas, por outro lado, analisará a quantidade de pontos de função consumidos pela contratada na execução do serviço, garantindo uma remuneração justa e evitando que a contratante efetue pagamento em mais de uma vez pelo mesmo serviço prestado;	▪ Gerente do Projeto

Item	Risco	Probabilidade	Dano	Estratégia do Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
15.	Não disponibilização do *software* em ambiente de produção para a utilização e avaliação dos reais usuários.	Baixa	Alto	Mitigar	▪ Garantir que na fase de planejamento do *Backlog* da *Sprint* seja apresentado as formas de aceito e ou homologação do Produto	▪ Gerente do Projeto; ▪ *Time Scrum*	▪ O *Product Owner* (PO) deve ser capacitado na metodologia que se está utilizando. Assim, ele terá conhecimento suficiente para entender como se dará o desenvolvimento, poderá se planejar para as reuniões e validações e se torna, realmente, o "dono" do produto; ▪ Deve-se escolher um *Product Owner* (PO) que possibilite uma grande integração de toda a área demandante ao projeto, para evitar que o produto tenha um viés fortemente influenciado; ▪ Todo *software* precisa de uma verificação e validação pela equipe demandante. Portanto, a cada entrega, o *Product Owner* (PO) deve estimular a validação do módulo por parte da equipe, fazendo-o de forma ampla e intensa, de forma a verificar se todos os requisitos são atendidos e o sistema realmente representa a vontade da equipe e o atendimento à demanda.	▪ Gerente do Projeto; ▪ *Time Scrum*; ▪ PO.

Item	Risco	Probabilidade	Dano	Estratégia de Tratamento	Ação Preventiva	Responsável	Ação de Contingência	Responsável
16.	Forma de pagamento não baseada em resultados.	Baixa	Alto	Neutralizar	▪ Garantir que no Termo de Referência (TR) esteja claro o uso do método de Análise pó Ponto de Função.	▪ Gerente do Projeto; ▪ Equipe do Planejamento da Contratação	▪ Para que os *softwares* entregues sejam avaliados de forma técnica e eficiente, faz-se necessária a criação de um conceito de Fábrica de Testes e de Fábrica de Métricas, que podem ser compostas por servidores ou podem ser contratadas, mediante os ditames dos normativos vigentes. A Fábrica de Testes visa submeter o software entregue a testes de estresse ou de conformidade, de forma a verificar se o mesmo responde segundo o padrão de aceitação solicitado e se há algum tipo de problema que trave ou impeça o regular funcionamento do sistema. A Fábrica de Métricas, por outro lado, analisará a quantidade de pontos de função consumidos pela contratada na execução do serviço, garantindo uma remuneração justa e evitando que a contratante efetue pagamento em mais de uma vez pelo mesmo serviço prestado;	▪ Gerente do Projeto

Por fim, este tópico teve como objetivo descrever formas de mitigação para os riscos identificados no processo de desenvolvimento de *software* com base em metodologias ágeis. Estas aplicações e de outras práticas que venham a se mostrar vantajosas no desenvolvimento de *softwares* e não choquem com princípios consagrados da Administração Pública, acredita-se que grande parte dos riscos levantados pelo TCU por intermédio do Acórdão n. 2314/2013, serão mitigados ou até eliminados pelo CDS. No entanto, é necessário lembrar que não é possível eliminar toda fonte de risco, uma vez que existem situações que podem, inclusive, concorrer para o aprimoramento do processo. Portanto, o foco da aplicação dessas práticas deve ser a proteção aos recursos públicos e o seguimento dos princípios da Administração Pública, de forma a evitar situações que ofendam os normativos e que garantam aos cidadãos um serviço prestado com qualidade.

O próximo capítulo apresenta uma síntese da avaliação do projeto de pesquisa, realizada com o apoio do referencia teórico, através dos estudos realizados, bem uma análise da pesquisa obtida sobre o Processo de desenvolvimento ágil adotado por outras Administrações públicas e por fim as lições aprendidas com a pesquisa.

Capítulo 6

Conclusão

As demandas atuais de serviço de TI pelo governo exigem crescentes níveis de produtividade e qualidade na prestação de serviços de desenvolvimento de *software* e na adoção da gestão de riscos associados aos objetivos estratégicos da organização. Para isso, na presente pesquisa foram apresentados os fundamentos e os conceitos da Metodologia Ágil e a Gestão de Riscos associados ao processo de desenvolvimento de *software* em regime ágil adotado pela APF, proporcionando uma possível quebra de incertezas, melhorando os indicadores de satisfação e aceitação do cliente com maior valor agregado ao negócio. Isso é necessário porque o modelo de desenvolvimento atualmente utilizado pela APF já não tem respondido às expectativas dos órgãos.

As ações estratégicas dos órgãos tem se tornado cada vez mais emergenciais sob a ótica competitiva ao atendimento do cliente final com relação aos seus serviços. O comportamento no Centro de Desenvolvimento de Sistemas do Exército (CDS) não é diferente, pela complexidade e a cultura do seu processo de desenvolvimento de *software*, considerando que seus os clientes diretos exigem, cada vez mais qualidade e agilidade na entrega dos produtos. As principais contribuições deste trabalho foram o estudo teórico que serviu para corroborar os resultados obtidos por intermédio dos conceitos adquiridos, a partir da revisão das literaturas que tratam da metodologia ágil e de gestão de riscos; bem como a apresentação de um processo de desenvolvimento de *software* do EB fundamentado no método ágil com o respectivo plano de gestão de riscos.

Durante a realização deste trabalho, foi possível verificar e discutir a aplicabilidade dos métodos ágeis no contexto da Administração Pública Federal. O resultado em geral das visitas e os questionários aplicados em órgãos que já executam o método ágil internamente e em regime de contratação, foi amplamente positivo, não somente do ponto de vista técnico como também negocial, do cliente, gerando a expectativa de continuidade no novo processo de desenvolvimento de *software* do EB.

Além disso, foi reforçada a prática da gestão de riscos, por intermédio de técnicas e ferramentas adequadas, para apoiar a tomada de decisão da alta gestão do EB.

Dos fatores que elucidaram o interesse pela pesquisa pode-se citar a possível agregação de valor mais frequente para o cliente em forma de *software* funcionando e ao mesmo tempo permitir uma maior flexibilidade de requisitos em função do contexto proposto pelo método ágil e na técnica de gestão de risco como mecanismo de mitigação e maturidade do processo organizacional.

Na execução do processo de desenvolvimento utilizado pelos diversos órgãos, ora analisados, ficou identificado que a governança dos ativos advindos do método ágil ganham destaque. As possibilidades de interações constantes com o cliente, área do negócio, representado pelo *Product Owner* (PO) juntamente com o *Time Scrum* foram a principal ferramenta de criação e difusão do conhecimento percebida pelos respondentes, haja vista a constante entrega de valor que necessariamente tem como referência os objetivos estratégicos da organização.

O grande desafio do método ágil perante APF do ponto de vista do Tribunal de Contas da União (TCU) está associado à gestão dos riscos promovendo um ponto fraco na gestão do desenvolvimento sem torna-lo um método pesado, nem tão pouco inviável do ponto de vista dos princípios da Administração Pública, a saber: (BRASIL, 1988)[13] i) Princípio da Legalidade: os atos da Administração têm que estar em conformidade com os princípios legais; ii) Princípio da Impessoalidade; iii) Princípio da Finalidade: orienta que as normas administrativas tem que ter sempre como objetivo o interesse público; iv) Princípio da Moralidade: ligando-se à moral e à ética administrativa; e v) Princípio da Publicidade: é a divulgação oficial do ato da Administração para a ciência do público em geral, com efeito de iniciar a sua atuação externa, ou seja, de gerar efeitos jurídicos.

Dessa forma, ao longo da execução do trabalho, os indicadores que avaliam os processos de desenvolvimento de *software* adotado por alguns órgãos federais apresentaram um razoável indicador de viabilidade do uso do método ágil pela APF. Destaca-se que, as organizações que participaram da pesquisa, por intermédio dos questionários enviados, ou mesmo pelas visitas realizadas já utilizam a metodologia ágil de forma institucionalizada e com um grande índice de satisfação tanto pelos gestores, bem como seus respectivos técnicos.

A pesquisa identificou a existência de inovação no conhecimento originado da prática dos métodos ágeis no desenvolvimento de *software* que deve ser integrado ao processo de contratação e/ou desenvolvimento interno na APF. Essa percepção se deu por intermédio da análise dos princípios e fundamentos dos métodos ágeis. Percebeu-se, ainda, que esses princípios informais, advindos das experiências vivenciadas no ciclo continuo do desenvolvimento de *software*, são de grande relevância aos interesses públicos.

Embora se ache que a inexperiência da APF inviabiliza o uso desse método, os resultados dos estudos realizados e os indicadores dos questionários aplicados materializam um grande potencial de inovação do método de desenvolvimento atualmente utilizado. Isso pode significar grande ganho de satisfação das organizações públicas, por intermédios de seus diversos gestores de negócios e a queda da insatisfação e improdutividades negociais, proporcionada pela má qualidade dos produtos desenvolvidos, bem como funcionalidades desconexas com as reais necessidades do negócio.

A possibilidade desses princípios informais de trabalho organiza e direciona o conhecimento compartilhado para se tornar um ativo de valor para o negócio. A percepção da existência desses ativos e a contribuição que eles proporcionam tanto para melhoria da qualidade do produto quanto para padronizar procedimentos e reduzir os tempos de espera, evidencia a hipótese de que existe uma inovação no conhecimento relevante no desenvolvimento de *software* fundamentado na metodologia ágil que ainda não são apropriados pela APF.

O objetivo de propor um processo de desenvolvimento de *software* e a mitigação dos riscos relacionados pelo TCU fundamentado no uso da metodologia ágil a luz das práticas adotadas pela APF foi verificado e atendido por intermédio da revisão de literatura. Sobre os métodos de desenvolvimento de *software* tradicional e ágil, apresentaram-se perspectivas de diversos autores, apresentados na literatura, que defendem que o uso do método ágil é um fator seguro, além de defenderem que o *framework Scrum* é o mais utilizado por organizações, por sua vez o mais maduro.

O trabalho empírico possibilitou o desenvolvimento de indicadores de risco e métodos de tratamento dos riscos no uso do método ágil no sentido de responder aos objetivos propostos. Com a realização da pesquisa foi possível compreender que o uso do método ágil é viável pela APF, o qual não quebra os princípios da Administração Pública, haja vista que não há prejuízos ao erário público e é facilmente adaptável a cultura organizacional com possíveis melhorias. Foi possível também identificar que o *framework Scrum* é o método mais adotado perante aos órgãos visitados e consultados, corroborando com a pesquisa bibliográfica conceitual.

Por fim, após a realização da pesquisa bibliográfica, juntamente com a pesquisa empírica, a grande contribuição do trabalho foi apresentar uma proposta para o processo de desenvolvimento de *software* do EB com base na metodologia ágil e elaborar um plano de gestão dos riscos indicados pelo TCU por intermédio do Acórdão n. 2314-33/2013 para o processo proposto. Espera-se que o resultado da pesquisa possa ser utilizado como referência para o atendimento das atividades de assessoria técnica no desenvolvimento de *software* do CDS como forma apoiadora a missão do EB.

Trabalhos futuros

Dada à restrição do tempo de aplicação do produto desta pesquisa foi evidenciada a necessidade do EB desenvolver a criação de indicadores de satisfação e qualidade dos produtos produzidos. Para ampliar esse conhecimento, outros trabalhos são necessários como fazer pesquisas de indicadores de produtividade do desenvolvimento de *software* fundamentado do método ágil e o método tradicional, para verificar se existem diferenças do uso do método ágil e o tradicional no âmbito da organização.

Outra pesquisa interessante seria desenvolver um método de análise de produtividade do desenvolvimento ágil diferente do usual que é o Ponto de Função, haja vista a impossibilidade de viabilizar a análise do indicador esforço x produtividade. As realizações dessas pesquisas devem resultar em uma melhora qualitativa do uso do método ágil de forma evolutiva.

Referências

[1] Associação Brasileira das Empresas de Software ABES. Mercado brasileiro de software - panorama e tendências. Report ISBN 978-8586700-03-3, Brazilian Software Market: scenario and trends, 2014. 1

[2] ABNT. *ABNT NBR ISO/IEC 31000-2009 – Gestão de riscos - Princípios e diretrizes*. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2009. Objetivo: fornecer princípios e diretrizes genéricas para a gestão de riscos. x, 5, 13, 15, 16, 18, 19, 118, 157

[3] ABNT. *ABNT NBR ISO/IEC 31010-2012 – Gestão de riscos - Técnicas para o processo de avaliação de riscos*. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2012. Objetivo: fornecer orientações sobre a seleção e aplicação de técnicas sistemáticas para o processo de avaliação de riscos. xiii, 20, 157

[4] ABPMP. Guia para o gerenciamento de processos de negócio corpo comum de conhecimento (bpm). Report, 2009. 82

[5] D. C AGUIAR and C. H. P MELLO. Femea de processo: Uma proposta de aplicação baseada nos conceitos da iso 9001:2000. *XXVIII Encontro Nacional de Engenharia de Produção (ENGEMEP)*, 2008. 116, 117

[6] A. L. Albertin. Relatório de pesquisa n. 7/2005, 2005. 10

[7] H. Alfaraj and A. Qin. Journal of engineering, design and technology. *Operation alising CMMI: integrating CMMI and COBIT perspective*, IX(1):323–335, 2011. 2, 3

[8] A. L. M. ARAUJO. *Indicadores de Qualidade e Produtividade como Instrumento de Apoio à Decisão no Processo de Expedição de Veículos*, volume VII. Produção, Belo Horizonte, 1997. 117

[9] Antônio C. O. Barroso, Beatriz Duarte, Caspar Van Rijinbach, Cindy Gordon, David Kato, Felipe Fioravante, Hermínia Aydar, Jean-Yves Prax, Larry Prusak, Marcos Felipe Magalhães, Paulo Roberto Floriano, Pierre Fayard, Rui Santo, Soumodip Sakar, Steve Wunker, Tatiana Gonoretzky, and Tomohiro Takanashi. *Inovação: quebrando paradigmas para vencer*. Saraiva, São Paulo, 2007. 6

[10] K. Beck, M. Beedle, and A. V. Bennekum. Agile manifesto, 2001. 41, 42

[11] E. W. N. Bernroider and M Ivanov. It project management control and the control objectives for it and related technology (cobit) framework. *Elsevier*, 29:325–336, 2011. 3

[12] Rosa Maria Villares BERTO and Davi Noborou NAKANO. A produção cientifíca nos anais do encontro nacional de engenharia de produção: Um levantametno de métodos e tipos de pesquisa. *Produção*, IX(2):65–76, 2000. 72, 73

[13] BRASIL. Constituição federal brasileira. Report, 1988. 78, 175

[14] BRASIL. Lei nº 8.666, de 21 de junho de 1993, junho 22 1993. 4, 157

[15] BRASIL. Programa excelência gerencial do exército brasileiro (peg-eb). Report, 2007. 78

[16] BRASIL. Tribunal de contas da união (tcu), 2012a. 2, 3

[17] BRASIL. *Metodologia IPHAN de Gestão de demandas de desenvolvimento Ágil de Software (MIDAS)*. Instituto do Patrimânio Histórico e Artístico Nacional (IPHAN), Brasília, 2013. xi, xiii, 74, 90, 91, 92, 93, 94

[18] BRASIL. Tribunal de contas da união (tcu), 2013b. x, xii, 2, 12, 13, 103, 116, 155, 162

[19] BRASIL. Tribunal de contas da união (tcu), 2013c. 153, 154

[20] Banco Central do Brasil (BACEN) BRASIL. Processo Ágil de desenvolvimento de software do banco central. Report, 2014. xi, 74, 94, 96

[21] Centro de Desenvolvimento de Sistemas BRASIL. *Manual Técnico para Metodologia de Desenvolvimento de Software do Exército*. Exército Brasileiro, Brasília, 2012. xi, 84, 85, 86, 87, 88

[22] Exército Brasileiro BRASIL. Exército brasileiro, 2009a. 79

[23] Exército Brasileiro BRASIL. Sobre o comando do exército, 2015b. 78

[24] Tribunal Superior Eleitoral BRASIL. Método de desenvolvimento com prática Ágeis (magil). Report, 2014. xi, xiii, 74, 98, 99, 100, 101, 102

[25] Donald L. BURESH. Customer satisfaction and agile methods. *This article is part of the IEEE Reliability Society 2008 Annual Technology Report.*, 2008. 7

[26] Bernado Vasconcelos de CARVALHO and Carlos Henrique Pereira MELLO. Aplicação de método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológia. *Gestão e Produção*, XIX(3):557–573, 2012. 8, 137

[27] Amado Luiz CERVO, Pedro Alcino BERVIAN, and Roberto da SILVA. *Metodologia Cientifica*. Pearson Prentice Hall, São Paulo, 2007a. 72

[28] M. Cohn. *Desenvolvimento de Software com Scrum - Aplicando métodos ágeis com sucesso*. Bookman, Porto Alegre, 2011. x, 41, 50, 59

[29] C. da S. Cruz, E. L. P. Andrade, and R. M. da C. Figueiredo. *Processo de Contratação de Serviços de Tecnologia da Informação para Organizações Públicas*. PBQP Software, Brasília, 2011. 5

[30] A. A. Fernandes and V. F. Abreu. *Implantando a Governança de TI: da estratégia a gestão de processos e serviços*. Brasport, Rio de Janeiro, 2008. 11, 12

[31] Aguinaldo Aragon Fernandes and Vladimir Ferraz de Abreu. *Implantando a governança de TI: da estratégia à gestão dos processos e serviços*. Brasport, Rio de Janeiro, 2012. 82

[32] A. B. H. Ferreira. *Aurélio século XXI: o dicionário da Lingua Portuguesa*. Nova Fronteira, Rio de Janeiro, 2011. 24, 35

[33] A. C. Gil. *Como Elaborar Projetos de Pesquisa*. Atlas, São Paulo, 2007. 73

[34] M. GIROD-SERVILLE and V. PERRET. Fondements épistémologiques de la recherche. *THIETART, R.A.*, pages 13–33, 1999. 70, 71

[35] IBM. Web services globalization model, 2013. 38, 40

[36] Instituto do Patrimônio Histórico e Artístico Nacional IPHAN. *Carta ao cidadão*. Brasilia, 2014. 90

[37] ISACA. Serving it governance professionals, 2013. 10

[38] H. Kniberg. *Scrum e XP direto das Trincheiras*. C4media: InfoQ, 2007. 55, 57, 58, 59, 61, 63, 66, 67

[39] P. F. A LIMA, L. A. S FRANZ, and F. G. AMARAL. Proposta de utilização do fta como ferramenta de apoio ao femea em uma empresa do ramo automotivo. *XIII SIMPEP*, 2006. 116

[40] Felipe N. R. Machado. *Análise e gestão de requisitos de software - onde nescem os sistemas*. Érica, São Paulo, 2011. 35

[41] M. A. Marconi and E. M. Lakato. *Técnicas de pesquisa - planejamento e execução de pesquisas, amostragens e técnicas de pesquisas, elaboração, análise e interpretação de dados*. Atlas, São Paulo, 2007a. 69, 72

[42] Roselaine Biangaman de MATOS and Marcos MILAN. *Aplicação Sistêmica do Modo de Análise de Falhas e Efeitos (FMEA) para o Desenvolvimento de Indicadores de Desempenho de Empresas de Pequeno Porte*. Scielo, São Paulo, 2009. xiii, 118

[43] F.N. MATTAR. *Pesquisa de Marketing: Metodologia e Planejamento*. Atlas, São Paulo, 1996. 73

[44] J. B. Medeiros. *Redação Científica – A prática de Fichamentos, Resumos, Resenhas*. Atlas, São Paulo, 2008. x, 71

[45] Grigori Melnik and Frank Maurer. Comparative analysis of job satisfaction in agile and non-agile software development teams. *Extreme Programming and Agile Processes in Software Engineering*, 4044:32–42, 2006. xiii, 7, 8

[46] Ikujiro NONAKA and Hirotaka TAKEUCHI. *Criação de Conhecimento na Empresa*. Campus, Rio de Janeiro, 1997. 142

[47] Alexander OSTERWALDER and Yves PIGNEUR. *Business Model Generation: A Handbook for Visonaries, Game Changers, and Challengers*. USA, 2009. xii, 78, 141

[48] A. Phan. *Scrum em Ação - Gerenciamento e Desenvolvimento Ágil de Projetos de Software*. Novatec, São Paulo, 2011. 41, 48, 49, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67

[49] Michael E. PORTER. *Competição - Estratégias competitivas essenciais*. Elsevier, Rio de Janeiro, 1999. 6, 7

[50] Roger S. Pressman. *Engenharia de Software - Uma abordagem profissional*. AMGH, Porta Alegre, 2011a. x, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 40, 41, 42, 43, 46, 47, 48, 49, 50, 51, 61, 68, 137

[51] R. Q. Reis, C. A. Reis, and D. J. Nunes. *Automoção no Gerenciamento do Processo de Engenharia de Software*. Escola de Informática Norte, Belém, 2002. 25

[52] D. A. Rezende and A. F. de Abreu. Revista de administração mackenzie. *Planejamento Estratégico da Tecnologia de Informação alinhado ao Planejamento Estratégico de Empresa*, 2:39–51, 2001. 10

[53] L. RISING and N. S. JANOFF. The scrum software development process for small teams. *IEEE*, XVII:26–32, 2000. 1

[54] C. C SAAVEDRA, P. SCHRIEVEROOFF, and U. LINDEMANN. Analysing the concept os calue to identify relevant stakeholders preferences to design for adaptability. *Internacional Design Conference*, pages 141–152, 2014. 2, 8

[55] Flavio Roberto Souza dos SANTOS and Santos CABRAL. Fmea and pmbok applied to project risk management. *Gestão da Tecnologia e Sistemas de Informação*, V(2):347–364, 2008. xii, 117, 118, 120, 121

[56] K. Schwaber and J. Sutherland. Scrum org, 2011. xiii, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67

[57] Pedro Serrador and Jeffrey K. Pinto. Does agile work? — a quantitative analysis of agile. *International Journal of Project Management*, 2014. 5

[58] Ian Sommerville. *Engenharia de Software*. Person Prentice Hall, São Paulo, 2011. x, xiii, 22, 24, 25, 27, 28, 29, 30, 31, 33, 34, 38, 43, 44, 45

[59] A. C. Tonini, M. M. Carvalho, and M. M. Spinola. Produção. *Contribuição dos modelos de qualidade e maturidade na melhoria dos processos de software*, 18:275–286, maio/agosto 2008. 3

[60] C. et al. VOSS. Case research in operations management. *International Journal of Operations and Production Management*, XXII(2):195–2019, 2002. 73

[61] P. Weill and J. W. Ross. *Governança de TI – Tecnologia de Informação*. M. Books, São Paulo, 2006. 10, 11

[62] Roberto K. Yin. *Estudo de caso: planejamento e métodos*. Bookman, Porto Alegre, 2001. 72, 73

Apêndice A

Apêndice

Os apêndices apresentam os documentos complementares à pesquisa. Este documento envolve o questionário auxiliar resultantes da pesquisa do uso da metodologia ágil em organizações públicas federais para a compreensão dos objetivos da pesquisa.

Questionário

Pesquisa sobre a adoção de metodologias ágeis em contratações de TI no Serviço Público Federal

Pesquisa de Alunos da Engenharia de Produção da Universidade de Brasília sobre a adoção de metodologias ágeis no desenvolvimento de softwares no Serviço Público Federal, bem como entender as melhores práticas e formas de gerenciamento de tais contratos e/ou desenvolvimentos.

*Obrigatório

Nome do Órgão:*

Escreva o nome do órgão que você trabalha por extenso (ex: Advocacia Geral da União)

A qual Departamento você está vinculado?*

Escreva o nome de seu Departamento por extenso (ex.: Departamento de Tecnologia da Informação)

Perfil do Entrevistado*

Qual função descreve melhor sua posição atual no órgão?

- ○ Coordenador-Geral
- ○ Coordenador de Área
- ○ Técnico de Área
- ○ Fiscal de Contrato
- ○ Desenvolvedor
- ○ Gerente de Projeto
- ○ Outro: [ ]

Perfil do entrevistado*

Qual a sua experiência em utilização de metodologias ágeis?

- ○ Nenhuma
- ○ Menor que 6 meses
- ○ Entre 6 meses e 1 ano
- ○ Entre 1 e 2 anos
- ○ Entre 3 e 5 anos
- ○ Acima de 5 anos

Perfil interno da Organização*

A organização utiliza métodos ágeis em desenvolvimentos realizados internamente?

- ○ Não
- ○ Sim, em menos de 25% dos projetos
- ○ Sim, entre 25% e 50% dos projetos
- ○ Sim, entre 50% e 75% dos projetos
- ○ Sim, em mais de 75% dos projetos

Perfil interno da Organização*

A organização utiliza métodos ágeis internamente para acompanhar e gerenciar contratos de desenvolvimento com empresas terceirizadas?

- ○ Sim
- ○ Não

Contratação com Empresas parceiras*

Quantos projetos ágeis já foram realizados e/ou estão em desenvolvimento por meio de empresa terceirizada?

- ○ Nenhum
- ○ Um
- ○ Dois
- ○ Três
- ○ Quatro ou mais

Qual a duração média dos contratos ou do desenvolvimento interno utilizando métodos ágeis? *

- ○ Até 1 ano
- ○ Entre 1 e 2 anos
- ○ Entre 2 e 3 anos
- ○ Acima de 3 anos
- ○ Meu órgão não realiza ou realizou tal modalidade de contratação/desenvolvimento

---

Quais metodologias ágeis foram utilizadas na execução do contrato (ou quais se usariam em uma execução futura)? *

Se necessário, selecione mais de uma resposta

- ☐ Desenvolvimento Enxuto
- ☐ Crustal
- ☐ Programação Extrema (Extreme Programming - XP)
- ☐ *Kanban*
- ☐ Desenvolvimento Orientado a Testes
- ☐ *Scrum*
- ☐ Outro: ____

Do ponto de vista da organização, qual o nível de dificuldade na gestão de um contrato de desenvolvimento de sistemas com a utilização de metodologias ágeis? *

Escala em ordem crescente de facilidade.

	1	2	3	4	5	
Muito Difícil	○	○	○	○	○	Muito Fácil

Na execução de contratos públicos com métodos ágeis, quão precisa é a previsão de seu custo total nos momentos iniciais do projeto? *

Escala em ordem crescente de precisão.

1	2	3	4	5

Em contratações passadas onde foram utilizados métodos ágeis, houve necessidade de aditivo de valores em contratos com métodos ágeis? *

- ○ Não
- ○ Sim, em menos de 10% dos casos
- ○ Sim, entre 10% e 25% dos casos
- ○ Sim, entre 25% e 50% dos casos
- ○ Sim, em mais de 50% dos casos
- ○ Meu órgão não realiza ou realizou tal modalidade de contratação/desenvolvimento

---

Em contratações passadas onde foram utilizados métodos ágeis, houve necessidade de aditivo de vigência (tempo de execução) em contratos com métodos ágeis? *

- ○ Não
- ○ Sim, em menos de 10% dos casos
- ○ Sim, entre 10% e 25% dos casos
- ○ Sim, entre 25% e 50% dos casos
- ○ Sim, em mais de 50% dos casos
- ○ Meu órgão não realiza ou realizou tal modalidade de contratação/desenvolvimento

---

A organização pretende executar novos contratos de desenvolvimento de sistemas utilizando metodologias ágeis? *

- ○ Não
- ○ Sim, em contratos de duração até 1 ano
- ○ Sim, em contratos de duração entre 1 e 2 anos
- ○ Sim, em contratos de duração acima de 2 anos

Apêndice B

Apêndice

	Riscos	Índices			
		Ocorrência	Severidade	Detecção	Risco (IR)
Riscos relativos ao processo	oRisco 1: contratação de desenvolvimento de *software* com adaptação de metodologia ágil que desvirtue sua essência				0
	oRisco 2: alteração da metodologia ágil adotada no instrumento convocatório no decorrer da execução contratual				0
	oRisco 3: ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual				0
	oRisco 4: exigência de artefatos desnecessários ou que se tornam obsoletos rapidamente.				0
	oRisco 5: utilização de contrato para desenvolvimento de *software* por metodologias tradicionais para desenvolvimento por métodos ágeis.				0
Riscos relativos a pessoas	oRisco 6: falta de comprometimento ou colaboração insatisfatória do responsável indicado pela área de negócios *Product Owner* (PO) no desenvolvimento do *software*.				0
	oRisco 7: falta do conhecimento necessário do indicado pela área de negócios *Product Owner* (PO) para o desenvolvimento do *software*.				0
	oRisco 8: excessiva dependência da visão do indicado pela área de negócios *Product Owner* (PO).				0
	oRisco 9: equipe da empresa contratada não ter expertise em desenvolvimento de *software* com métodos ágeis.				0
	oRisco 10: dificuldade de comunicação entre a equipe de desenvolvimento da contratada com o indicado pela área de negócios *Product Owner* (PO).				0

Riscos		Índices			
		Ocorrência	Severidade	Detecção	Risco (IR)
Riscos relativos a produtos	oRisco 11: alteração constante da lista de funcionalidades do produto.				0
	oRisco 12: iniciação de novo ciclo sem que os produtos construídos na etapa anterior tenham sido validados.				0
	oRisco 13: falta de planejamento adequado do *software* a ser construído.				0
	oRisco 14: pagamento pelas mesmas funcionalidades do *software* mais de uma vez, em virtude de funcionalidades impossíveis de serem implementadas em um único ciclo, ou em virtude da alteração de funcionalidades ao longo do desenvolvimento do *software*.				0
	oRisco 15: não disponibilização do *software* em ambiente de produção para a utilização e avaliação dos reais usuários.				0
	oRisco 16: forma de pagamento não baseada em resultados.				0

Índices	Critérios do FMEA		
	Ocorrência (Oc)	Severidade (Sv)	Detecção (Dt)
1	Probabilidade muito remota de acontecer	É razoável esperar que o cliente não perceberá a falha	Probabilidade muito alta que a falha seja detectada
2	Número de ocorrência baixo	O cliente perceberá a falha, mas não ficará insatisfeito por causa dela	Probabilidade alta de que a falha seja detectada
3	Número de ocorrência moderado	O cliente perceberá a falha e ficará insatisfeito	Probabilidade média de que a falha seja detectada
4	Número de ocorrência alto	O cliente ficará insatisfeito, mas não terá sua segurança afetada	Probabilidade baixa de que a falha seja detectada
5	Falhas em proporções alarmantes	O cliente ficará muito insatisfeito e afeta a sua segurança	Probabilidade muito baixa de que a falha seja detectada