PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM ESTUDO EMPÍRICO SOBRE A ADOÇÃO DE MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE EM ORGANIZAÇÕES PÚBLICAS

ISAQUE VACARI

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação, pelo Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Rafael Prikladnicki

Porto Alegre, Rio Grande do Sul, Brasil, 2015.

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP)

V120u Vacari, Isaque

Um estudo empírico sobre a adoção de métodos ágeis para desenvolvimento de software em organizações públicas / Isaque Vacari. - Porto Alegre, 2015.
206 p.

Dissertação (Mestrado) - Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Rafael Prikladnicki.

1. Informática. 2. Engenharia de Software. 3. Métodos Ágeis. 4. Organização Pública. I. Prikladnicki, Rafael. II. Título.

CDD 005.1

Ficha Catalográfica elabora pela

Biblioteca da Embrapa Informática Agropecuária

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Um Estudo Empírico Sobre a Adoção de Métodos Ágeis para Desenvolvimento de Software em Organizações Públicas" apresentada por Isaque Vacari como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 17/03/2015 pela Comissão Examinadora:

Prof. Dr. Rafael Prikladnicki– Orientador — PPGCC/PUCRS

Prof. Dr. Duncan Dubugras Alcoba Ruiz– — PPGCC/PUCRS

Prof. Dr. Rodrigo Penteado Ribeiro de Toledo– — UFRJ

Homologada em 18/06/2015, conforme Ata No. 010 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

DEDICATÓRIA

“Para você - pessoa de bem - que contribui de alguma maneira para o crescimento, fortalecimento e melhoria na qualidade de vida das pessoas em nosso país.”

AGRADECIMENTOS

Acima de tudo, agradeço a Deus pelo dom da vida e saúde.

A Empresa Brasileira de Pesquisa Agropecuária (Embrapa) pelo apoio financeiro possibilitando minha dedicação exclusiva ao mesmo. A Chefia da Embrapa Informática Agropecuária pela confiança na execução deste trabalho de pesquisa e sua aplicabilidade ao momento atual da Empresa.

Ao meu orientador Prof. Dr. Rafael Prikladnicki pela amizade, ótima convivência, enorme aprendizado e por abrir novas “portas” na comunidade ágil. Vivenciamos vários momentos de reflexão, os quais foram importantes para construção do conhecimento a qual se propôs este estudo. A sua tranquilidade e competência para conduzir-me e manter-me focado foram fundamentais para fazer-me chegar do outro lado “(...) mestrado ou doutorado bons são aqueles que conseguimos finalizar”.

Ao Prof. Dr. Duncan Ruiz pelo acompanhamento do trabalho com suas importantes considerações e experiência na condução de pesquisas acadêmicas.

Aos novos colegas e amigos, funcionários e professores do PPGCC pela amizade e experiências compartilhadas. Com muito carinho, ao Bernardo Estácio, Ciro Santos, Silvia Nunes, Alessandra Dutra e Douglas Monteiro, pela convivência e ajuda ao longo desses dois anos.

Ao meu grande amigo Marcos Visoli pelo acompanhamento das atividades enquanto empregado da Embrapa e disposição em buscar soluções para melhoraria na qualidade do serviço público, bem como, a todo(a)s o(a)s amigo(a)s e colegas “Embrapiano(a)s” que apoiaram-me e ajudaram-me nessa jornada.

As organizações públicas participantes dos estudos de caso e as pessoas entrevistadas pela excelente contribuição e por reconhecerem valor nesta pesquisa. Ao colega Jorge Horácio Audy pelo contato inicial com uma das organizações e as chefias por permitirem minha entrada em “campo” e publicação dos resultados.

Aos pesquisadore(a)s em métodos ágeis e a comunidade ágil pela dedicação em promover e disseminar os conceitos do desenvolvimento ágil de software e suas nuances.

Em especial, aos meus familiares e a minha esposa & filhos pelo ambiente de paz e alegria a qual vivemos e por tudo o que representam na minha vida.

UM ESTUDO EMPÍRICO SOBRE A ADOÇÃO DE MÉTODOS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE EM ORGANIZAÇÕES PÚBLICAS

RESUMO

Nesta pesquisa, executou-se um estudo empírico sobre a adoção de métodos ágeis para desenvolvimento de software em organizações públicas. Este estudo incluiu a revisão da literatura da área e entrevistas com quatro equipes de desenvolvimento de software em duas organizações públicas de grande e médio porte no Brasil. Constatou-se que métodos ágeis podem ser realmente adotados por organizações públicas. Em primeiro lugar, a análise de dados sugere que uma boa alternativa é iniciar a adoção de métodos ágeis com pessoas dispostas a mudanças fortemente apoiadas pela alta gerência em projetos-piloto importantes. Posteriormente, a mudança dependerá da ampliação e interação com outras equipes visando alcançar a grande maioria da organização; em seguida, essas pessoas trabalham pela constituição de um ecossistema de aprendizado e melhoria contínua para não se acomodar conscientemente ou inconscientemente com as primeiras conquistas e subsequentemente com a zona de conforto. Em segundo lugar, confirmou-se que os benefícios da adoção de métodos ágeis em organizações públicas são mais sobre a satisfação das pessoas com o resultado do trabalho realizado e a forma como ele foi alcançado. Em terceiro lugar, identificou-se algumas barreiras difíceis de superar em organizações públicas, incluindo o uso arraigado de métodos guiados por plano com a abordagem “Big Design Up-Front” (BDUF), bem como a falta de conhecimento e experiência em métodos ágeis. Em quarto lugar, comprovou-se que métodos ágeis não é percebido como um processo acabado que se finaliza quando uma determinada conquista é alcançada; pelo contrário, ele é algo que se processa de maneira contínua e depende de um conjunto de fatores ligados as pessoas e ao ambiente onde ele está inserido para alcançar novos e melhores resultados continuamente. Por último, constatou-se que métodos ágeis em organizações públicas é um caminho sem retorno, onde seus praticantes precisam compartilhar suas experiências e ajudar uns aos outros. Esta pesquisa contribui principalmente no sentido de propor um conjunto de recomendações para a adoção de métodos ágeis em organizações públicas, além da formação e ampliação da base teórica sobre o tema.

Palavras-chave: Engenharia de software empírica, métodos ágeis, organização pública, setor público.

AN EMPIRICAL STUDY ABOUT ADOPTING AGILE METHODOLOGIES FOR SOFTWARE DEVELOPMENT IN PUBLIC ORGANIZATIONS

ABSTRACT

This study reports from an empirical study planned and executed about adopting agile methodologies in public organizations. This study involved a literature review and interviews with four software development teams from two public organizations - large and medium-sized - located in Brazil. Through this method, it was found that agile methodologies could be adopted in public organizations. First, the analysis suggests that a good alternative is to start the adoption of agile methodologies with people willing to change - strongly supported by senior management - working on important pilot-projects. Subsequently, the change will depend on their interaction with other teams, in order to reach the vast majority of the organization; then these people work for the establishment of a learning ecosystem and continuous improvement to not accommodate - consciously or unconsciously - with the first winnings and later with the comfort zone. Second, we found that the benefits of agile methodologies in public organizations are more about people's satisfaction with the result of the work done and how it was achieved. Third, we also found some barriers difficult to overcome in public organizations, including the ingrained use of plan-drive methods, habit of “Big Design Up-Front” (BDUF) and lack of knowledge and experience in agile methodologies. Fourth, we see that agile methodologies is not perceived as a finished process that ends when a particular victory is achieved; on the contrary, it is something that takes place continuously and depends a set of people factors and the environment where it is inserted to achieve new and better results continuously. Finally, we note that the adoption of agile methodologies in public organizations is a way without return, where its practitioners also need to share their experiences and help each other. The main contribution of this research is the development of a set of recommendations for adopting agile methodologies in public organizations, and the systematization of the empirical evidence about this topic.

Keywords: Empirical software engineering, agile methodologies, public organizations, public sector.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS

AP	Administração Pública
ASD	*Adaptative Software Development*
BDD	*Behavior Driven Development*
BDUF	*Big Design Up Front*
CASE	*Computer-Aided Software Engineering*
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integration*
CNPTIA	Embrapa Informática Agropecuária
DoD	*United States Department of Defense*
DSDM	*Dynamic Systems Development Method*
EDUF	*Enough Design Up Front*
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
ES	Engenharia de Software
EUA	Estados Unidos da América
FDD	*Feature Driven Development*
GAO	*Government Accountability Office*
JAD	*Join Application Design*
ISO	*International Standards Organization*
NAO	*National Audit Office*
NDFU	*No Design Up Front*
OMB	*US Office of Management and Budget*
OOPSLA	*Object-Oriented Programming, Systems, Languages & Applications*
OP	Organização(ões) Pública(s)
PDCA	*Plan-Do-Check-Act*
PDS	Processo de Desenvolvimento de Software
PRINCE2	*PRojects IN Controlled Environments*
PROCERGS	Companhia de Processamento de Dados do Estado do Rio Grande do Sul
PSP	*Personal Software Process*
PO	*Product Owner*
RAD	*Rapid Application Development*
RIPP	*Rapid Iterative Production Prototyping*
ROE	Revisão de Obra Especializada
RSL	Revisão Sistemática da Literatura

RUP	*Rational Unified Process*
SEI	*Software Engineering Institute*
SISP	Sistema de Administração de Recursos de Tecnologia da Informação
SLTI	Secretaria de Logística e Tecnologia da Informação do Ministério do Planejamento, Orçamento e Gestão
SSADM	*Structured Systems Analysis and Design Method*
SW	Software
TDD	*Test Driven Development*
TI	Tecnologia da Informação
TICs	Tecnologias de Informação e Comunicação
TSP	*Team Software Process*
UK	Reino Unido
UX	*User Experience*
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

A pesquisa que agora se apresenta sob a forma de uma dissertação de mestrado centra a problemática da adoção de métodos ágeis em organizações públicas (OP). Este capítulo descreve a motivação e caracteriza o problema estudado, nas seções 1.1 e 1.2 respectivamente. Em busca de solução para o problema apresentado, são apresentados os objetivos do trabalho, na seção 1.3. Em seguida, é descrito o contexto geral de pesquisa onde esta proposta se encaixa, na seção 1.4. Por fim, é resumida a forma como os capítulos deste documento foram organizados, na seção 1.6.

1.1 Motivação

O mundo contemporâneo vem exigindo bem mais dos governos. Com o avanço tecnológico no final do século passado, como era esperado, as mudanças e os movimentos de modernização da Administração Pública (AP) passaram a incorporar as Tecnologias de Informação e Comunicação (TICs) para atender os compromissos atuais e emergentes da sociedade, como também, melhorar a qualidade dos serviços públicos prestados e agregar valor as atividades da AP [Bal10]. Na mesma direção, com o aumento da exigência por parte dos clientes e do mercado, muitas empresas de desenvolvimento de software estão tentando se tornar mais ágeis buscando cada vez mais oferecer serviços e produtos com maior alinhamento as necessidades do cliente, rapidez e qualidade [Coh11] e nas OP não é diferente [Wer12].

Equipes ágeis estão produzindo software de maior qualidade que atendem melhor às necessidades do usuário, com maior rapidez e a um custo menor do que equipes tradicionais [Ric08]. Isto tem levado também a uma maior satisfação do cliente, bem como maior grau de previsibilidade [Coh11]. Assim, OP poderiam beneficiar-se desses novos métodos para alcançar resultados melhores do que seria possível alcançar com métodos ditos como tradicionais.

Não obstante, muito tem sido escrito sobre o desenvolvimento ágil de software. Ele tem sido tema de interesse tanto da indústria [Ver15] como da academia [DNB+12] [DyD08] [MSK+13] e do governo [GAO12] [NAO12] [TCU13b]. Com o interesse gradativo das organizações públicas (OP) em métodos ágeis para o desenvolvimento de software e, muitas delas, já utilizando métodos ágeis [VaP14], bem como, algumas delas palestrando sobre o tema (conforme APÊNDICE D), as questões de como métodos ágeis são e podem ser melhor executados em ambientes de governo, quais são e como alcançar os benefícios que eles geram para as OP, quais são e como resolver os desafios para sua adoção em

OP estão se tornando cada vez mais urgentes. Assim, esta pesquisa, a luz da ciência, procura reunir e fornecer boas respostas para estas perguntas por meio da execução de estudos teóricos e empíricos executados no contexto das organizações públicas.

1.2 Caracterização do Problema

Um estudo recente do Standish Group [COM13] com 3.555 projetos de TI na esfera do governo norte-americano nos últimos 10 anos (2003-2012) revelou que somente 6% deles foram bem-sucedidos, como aqueles finalizados dentro do prazo; dentro do orçamento e que contemplam todas as funcionalidades originalmente especificadas. Outrossim, 52% dos grandes projetos foram considerados desafiadores, como aqueles que utilizam um orçamento maior do que o previsto, finalizados fora do prazo e que não contemplam todas as funcionalidades originalmente especificadas. Todos os demais projetos - cerca de 42% - fracassaram completamente, como aqueles cancelados em algum momento do ciclo de vida do desenvolvimento. Isso significa que muitas ideias transformam-se em grandes projetos, porém acabam com pouco ou nenhum êxito.

Especialistas em desenvolvimento de software e pesquisadores apontam para várias questões como a causa potencial de tais problemas, típicos de OP:

- Grandes projetos de TI são mais propensos ao fracasso [Wer12] (RSL04) (RSL14) (ROE03) [SaC03];
- Entregas e implantações *big-bang*[1] são mais propensas a falhas [Wer12] [GAO12] [NAO12] [HMT14] (RSL04) (RSL07) (RSL14);
- Várias empresas contratadas pelo governo trabalhando de maneira dispersa são mais difíceis de gerenciar do que o esforço em gerenciá-las de maneira ágil, onde os desenvolvedores estão localizados no mesmo espaço físico (RSL04) (RSL14);
- Cercado de cuidados e a fim de incorrer menos riscos, muitos governos blindam-se com elementos contratuais burocráticos que praticamente levam o projeto ao fracasso. Nesse pensamento, a última coisa em mente é tornar o sistema útil para seus clientes [Wer12].

Estes e outros problemas não são exclusivos do governo norte-americano. Em 2003, a Universidade de Oxford constatou que 84% dos projetos de TI executados no setor público e privado no Reino Unido (UK) fracassaram [SaC03]. Embora não houvesse muita diferença no desempenho dos projetos executados em ambos os setores, constatou-se que

[1] Implantação *big-bang* é uma implantação do tipo completa, em que todos os módulos do produto de software são implementados em todas as localidades ou em toda organização simultaneamente, com a mesma data para início da operação.

falhas de alto nível no setor público, em relação a entrega de projetos de TI, afetaram significativamente a sua reputação [Sac03]. Por exemplo, o projeto do sistema Libra ultrapassou os custos iniciais em mais de £318 milhões de libras esterlinas e atrasou 2 anos e meio, totalizando £416 milhões de libras esterlinas e 8 anos e meio para sua conclusão. No governo brasileiro, auditorias realizadas em diversos órgãos e entidades revelaram que embora os sistemas de informação dessas organizações contribuem efetivamente para as atividades das áreas de negócio e colaboram de maneira efetiva para o alcance de metas e objetivos revelando grande alinhamento entre TI e negócio, o elevado índice de descontentamento com o tempo para atendimento das demandas de sistemas é um fator de preocupação para a alta administração das organizações e indica que as unidades desejam maior agilidade na prestação desse serviço [TCU14]. Além das falhas de projetos de TI na AP e de relatórios questionando os seus benefícios e o tempo para conclusão de sistemas, estudos tem relatado a falta de uma evidência clara sobre os impactos positivos da TI na AP, já que em muitos casos os sistemas de TI não apresentam benefícios apenas expectativas exageradas e irrealistas [Gol07].

Para melhorar estes índices e alcançar melhores resultados, mudanças são necessárias e alguns governos têm encontrado boas soluções. O governo britânico, por exemplo, teve sucesso no desenvolvimento do gov.uk e na forma como ele modificou os critérios de aprovação financeira para executar projetos de desenvolvimento de software no governo com base na cultura ágil [HMT14]. A nova Unidade conhecida como Government Digital Service (GDS) [GDS14e] é responsável por todo esse trabalho. Ela é formada por equipes de técnicos internos, os quais são responsáveis pelo desenvolvimento e/ou contratação de fornecedores adequados para as necessidades do governo. Essas equipes tornaram o trabalho mais objetivo, flexível, adaptável e impreterivelmente focado nas necessidades dos clientes, o qual é uma demonstração clara de como o governo pode desenvolver a tecnologia certa no momento certo [GDS14b] [GDS14c] [GDS14d].

Na mesma direção, recentemente o *The Standish Group* publicou o relatório intitulado “*Chaos Manifesto 2013*” [SGI13]. Ele aponta como solução ao histórico de fracassos de projetos de TI a divisão de grandes projetos em projetos menores priorizados com base em valor de negócio, formados por equipes multifuncionais dedicadas integralmente ao projeto seguindo uma abordagem de desenvolvimento ágil de software. Ou seja, a melhor solução é adotar uma estratégia de entrega de projetos pequenos executados sequencialmente.

Por fim, o fato de que um sítio *web* deixou constrangido o então presidente dos Estados Unidos da América (EUA), Barack Obama, é digno de nota [NYT13]. Assim, os sistemas de governo na *web* tornaram-se verdadeiramente críticos para a sociedade. E isso é realmente positivo porque a *web* aparece como uma interface entre o governo e seus cidadãos. Esta realidade implica diretamente no aumento da demanda por produtos de software que agregam serviços importantes e eficientes para os cidadãos e a AP, e indiretamente implica na busca por melhores abordagens para desenvolvê-los ou adquiri-los. Desta forma, o incentivo à adoção de métodos ágeis é algo imprescindível para o futuro do desenvolvimento de software no governo. Assim, à luz da pesquisa científica, a área de investigação que norteou esta pesquisa foi métodos ágeis em organizações públicas. A questão de pesquisa foi assim definida:

É possível adotar métodos ágeis em organizações públicas e alcançar seus benefícios, superando os eventuais obstáculos organizacionais inerentes as organizações públicas?

1.3 Objetivos

O objetivo geral desta pesquisa é entender e ampliar o que se sabe sobre a adoção de métodos ágeis em OP, propondo um conjunto de recomendações para a sua adoção - com base em estudos teóricos e empíricos executados em OP - contribuindo para as OP alcançarem a agilidade no desenvolvimento de software e seus benefícios. De forma a complementar o objetivo geral proposto, os seguintes objetivos específicos foram definidos:

- Aprofundar a base teórica sobre modelos prescritivos e adaptativos de processo de desenvolvimento de software, com ênfase em métodos ágeis;
- Identificar, reunir, organizar e sintetizar aspectos sobre a adoção de métodos ágeis em OP encontradas na teoria;
- Identificar, reunir, organizar e sintetizar aspectos sobre a adoção de métodos ágeis em OP encontradas na prática;
- Consolidar os resultados encontrados na teoria e prática, que permita elaborar um arcabouço de resultados teóricos e empíricos sobre a adoção de métodos ágeis em OP;
- Propor um conjunto de recomendações para adoção de métodos ágeis em OP, a partir dos estudos teóricos e empíricos, contribuindo para as OP alcançarem os benefícios dos métodos ágeis.

1.4 Contexto

A expressão "administração pública" (AP) envolve dois sentidos: um, normalmente chamado de subjetivo, orgânico ou formal, segundo o qual essa expressão compreenderia as pessoas jurídicas, seus órgãos, suas entidades e agentes que executam a atividade administrativa; outro, conhecido como objetivo, material, operacional ou funcional, compreendendo a atividade realizada por esses últimos [Mei08]. Por conta disso, preferiu-se usar como substrato a expressão “organizações públicas” (OP) para representar as organizações cuja fonte de recursos é pública, na qual executam o Serviço Público, porque considera indispensável à sociedade a sua existência [Lim07]. Portanto, no âmbito desta pesquisa entendeu-se que organização pública é uma empresa financiada pelos cofres públicos - sendo formada por um conjunto de pessoas - cujo objetivo é atender à uma necessidade que é parte integrante do papel do Estado atender.

Atualmente, diversas OP iniciaram investimentos para adotar contratações de serviços de desenvolvimento de software utilizando métodos ágeis. Contudo, cada uma dessas organizações tem vivenciado diferentes experiências e vem compartilhando dificuldades comuns no que diz respeito às limitações impostas pelo normativo de contratação de software [TCU13b]. Uchoa [Uch13] e Pontes [Pon14] examinaram como contratar o desenvolvimento ágil de software na AP Federal de maneira que se garanta a adoção pelo contratante (governo) e contratado (fornecedor) de algumas práticas ágeis. Os autores concluíram que, mesmo havendo poucos instrumentos legais específicos para apoio é possível uma contratação de desenvolvimento ágil de software e que não existem barreiras para se desenvolver ou contratar soluções de software de maneira ágil dentro da legislação brasileira. Franco [Fra14], ao analisar a legislação para a terceirização do desenvolvimento de software na AP brasileira em relação as norte-americana e britânica, alerta para a real necessidade: “a demora na hora de se entregar alguma funcionalidade”, e sugere que a contratação de grandes soluções de software aconteça de forma modular, onde não se permita a construção e contratação de novas soluções de software com tempo superior a três meses.

Assim, esses estudos beneficiam as OP que “não desenvolvem software”, mas que no papel de contratantes precisam gerenciar as atividades de desenvolvimento de suas contratadas. Por outro lado, muitas outras OP desenvolvem software com equipes internas, além de gerenciar o desenvolvimento de alguns produtos de software de suas contratadas. Esta pesquisa aplica-se a esse perfil de OP, cuja missão é promover TI com maior autonomia tecnológica e menor dependência de empresas privadas. No âmbito da AP brasileira, algumas OP que melhor enquadram-se nesse aspecto são:

- Empresas públicas de TI. Exemplo: Empresa de Tecnologia e Informações da Previdência Social (DATAPREV) e Departamento de Informática do Sistema Único de Saúde (DATASUS);
- Serviço Federal e Companhias Estaduais e Municipais de Processamento de Dados. Exemplo: Serviço Federal de Processamento de Dados (SERPRO), Companhia de Processamento de Dados do Estado do Rio Grande do Sul (PROCERGS), Companhia de Processamento de Dados do Estado de São Paulo (PRODESP) e Companhia de Processamento de Dados do Município de Porto Alegre (PROCEMPA);
- Empresas e Centros de Pesquisa: Empresa Brasileira de Pesquisa Agropecuária (Embrapa), Instituto Nacional de Pesquisas Espaciais (INPE) e Instituto Tecnológico de Aeronáutica (ITA).

1.5 Métodos Ágeis ou Metodologias Ágeis?

A comunidade científica frequentemente se depara com uma dúvida em relação ao uso do termo métodos ágeis ou metodologias ágeis quando se descreve, em uma pesquisa, o conhecimento empírico identificado pelo(s) autor(es) para formalizar uma determinada área de estudo envolvendo o desenvolvimento ágil de software. Algumas revistas e congressos têm adotado o termo métodos ou *methods*. Outros aceitam como título metodologias ou *methodologies*, parecendo ser uma questão dispensável para a aceitação de artigos e publicação de obras abrangendo esse assunto. Porém, o dicionário básico de filosofia [JaM08] mostra que essas palavras não possuem o mesmo significado, sendo definidas da seguinte maneira:

> ***Método:*** *"Conjunto de procedimentos racionais, baseados em regras que visam atingir um objetivo determinado".*
>
> ***Metodologias:*** *"Literalmente, ciência ou estudo dos métodos. Investigação sobre os métodos empregados nas diferentes ciências, seus fundamentos e validade, e sua relação com as teorias científicas".*

Considerando o dicionário de filosofia, que apresenta uma suposta distinção entre metodologia (que seria a ciência ou estudo dos métodos) e os métodos em si, preferiu-se utilizar nesse estudo a expressão "métodos ágeis" para descrever as abordagens de desenvolvimento ágil existentes.

1.6 Organização do Volume

Este volume está organizado em seis capítulos. Em seguida, o capítulo 2 apresenta a metodologia de pesquisa, descrevendo cada uma das etapas do estudo, justificando a escolha e o uso dos métodos apresentados. No capítulo 3 apresenta-se o referencial teórico desta pesquisa, abrangendo os principais conceitos e implicações das áreas do estudo: processos e padrões de processos de software, métodos ágeis para desenvolvimento de software e métodos ágeis para desenvolvimento de software em OP. Esse capítulo também se propõe a construir um inventário de casos identificados na literatura sobre a adoção de métodos ágeis em OP, formando a base teórica sobre o tema.

No capítulo 4, descreve-se detalhadamente os estudos de caso executados em duas OP de Tecnologia de Informação (TI) de grande e médio porte respectivamente. Os estudos de caso relatam a experiência da OP durante o processo de adoção de métodos ágeis para o desenvolvimento de software, incluindo os benefícios alcançados, os problemas enfrentados, as lições aprendidas e as recomendações para o seu uso.

No capítulo 5, apresenta-se o arcabouço de resultados teóricos e empíricos consolidados a partir dos resultados dos estudos teóricos e empíricos descritos nos capítulos 3 e 4, respectivamente. O conjunto de recomendações para a adoção de métodos ágeis é apresentado no capítulo 5.6, como resultado do processo de pesquisa como um todo, apoiado nos estudos teóricos (capítulo 3) e empíricos (capítulo 4), os quais foram consolidados no capítulo 5.

2 METODOLOGIA DE PESQUISA

Neste capítulo apresenta-se a metodologia de pesquisa utilizada no estudo. Na seção 2.1 apresenta-se o desenho de pesquisa e as suas fases. Vale ressaltar que o detalhamento dos instrumentos e procedimentos metodológicos e os critérios de validade científica adotados para abordar a realidade estudada na prática serão detalhados no capítulo 4.

2.1 Desenho e Fases da Pesquisa

A seguir apresenta-se o desenho da pesquisa, identificando suas fases:

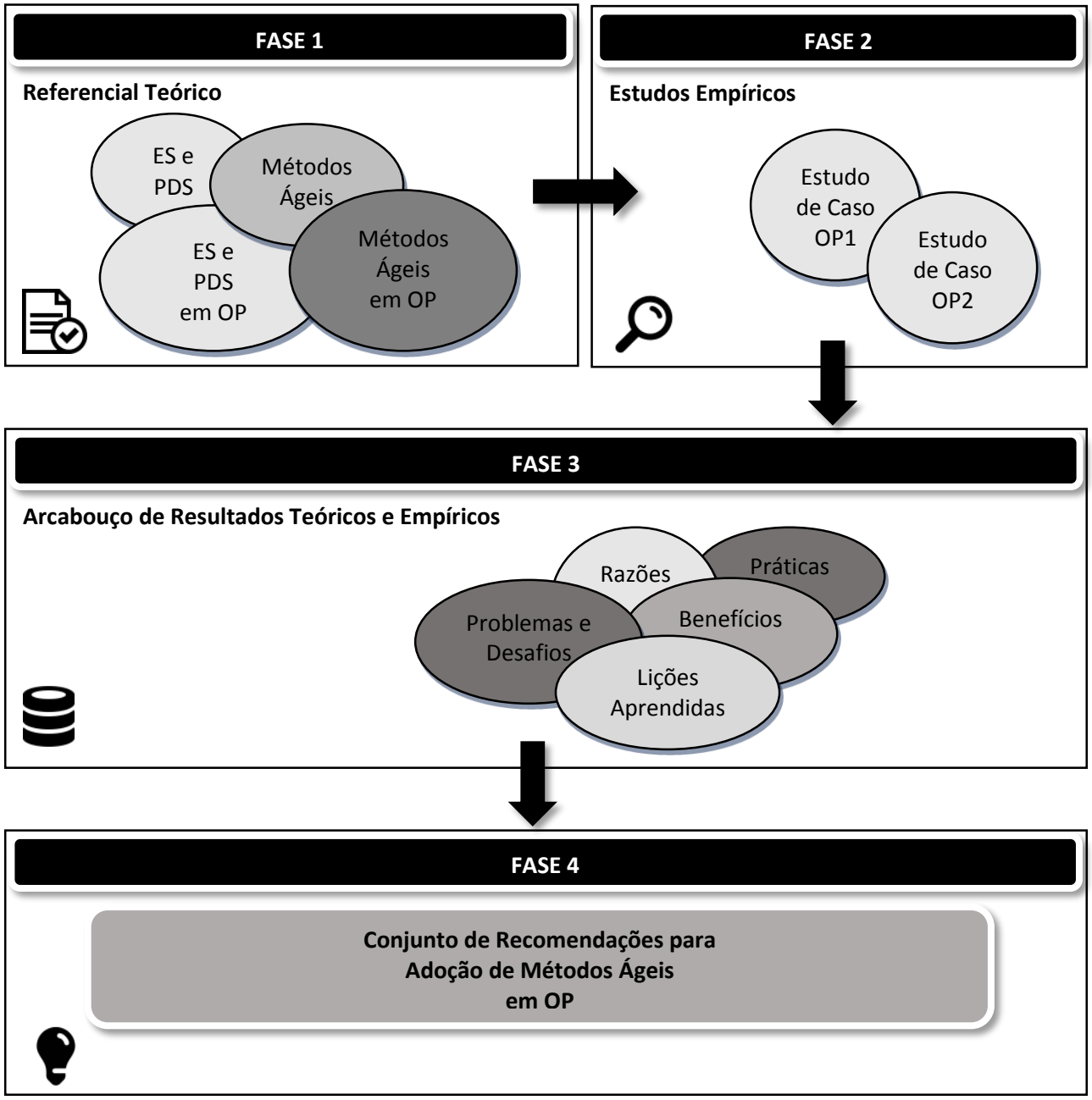

Figura 1 - Desenho e Fases da Pesquisa.

A seguir são descritas as quatro fases do estudo:

Fase 1: A fase de referencial teórico (F1) consistiu em aprofundar o conhecimento em modelos de processos de desenvolvimento de software e desenvolvimento de software em OP, com ênfase em métodos ágeis. Esta fase foi importante na medida em que permitiu caracterizar a área de estudo, formando uma base teórica sobre o assunto, contribuindo para a continuidade do estudo. A Tabela 1, apresenta a relação das publicações norteadoras desta pesquisa com a indicação da dimensão teórica, aspectos principais e autores relacionados.

Tabela 1 - Referencial teórico.

DIMENSÃO TEÓRICA	ASPECTOS PRINCIPAIS	AUTORES RELACIONADOS
Engenharia de Software e Processos de Desenvolvimento de Software	- Abordam aspectos gerais da ES. - Abordam a adoção dos métodos ágeis no âmbito da ES.	[BoT03] [Pre11] [Roy70] [Som11] [Waz13]
Engenharia de Software e Processos de Desenvolvimento de Software em Organizações Públicas	- Abordam o uso de TI no governo. - Abordam a criação e a adoção de métodos, processos e práticas de ES em OP.	[Bal10] [Ban08] [Bra85] [CFD+14] [Fre98] [GAO09] [Gol07] [Hei10] [HMT11] [Kro12] [Mcd10] [Mid99] [MPM09] [NAO06] [Pys06] [SaC03] [SIS12] [TCU13a] [TCU14] [VaP14]
Métodos Ágeis para Desenvolvimento de Software	- Abordam os conceitos de XP, Scrum e Kanban.	[Bas08] [Bec99] [BeA04] [Kni07] [Sho08] [Coh11] [Kni07] [Kni09] [Pic11] [Rub13] [Sch01] [Sch04] [Sch12] [Sch13] [And10] [Boe10] [Kni09] [Lea13] [Ohn97] [Pop11]
Métodos Ágeis para Desenvolvimento de Software	- Abordam os conceitos de métodos ágeis. - Abordam as razões, benefícios, problemas e desafios na adoção de métodos ágeis, bem como, suas vantagens e desvantagens. - Abordam os aspectos e mudanças organizacionais para transformação e adoção de métodos ágeis.	[App12] [ASR+02] [Bas08] [BBB+01] [Boe02] [BoT03] [Coc02] [CoH01] [Coh11] [DDM10] [DNB+12] [DyD08] [Hig02] [Iil10] [KaR02] [Kar13] [Koc04] [MSC+12] [MSK+13] [Pop11] [Pre11] [PWM14] [Ric08] [Sah12] [Sch12] [Sch13] [Sch95] [Scr12] [Som11] [Ver15] [Wer12]
Métodos Ágeis para Desenvolvimento de Software em Organizações Públicas	- Abordam experiências na adoção de métodos ágeis em OP, incluindo as razões, benefícios, problemas e desafios na sua adoção, tal como, recomendações para seu uso.	[GAO12] [GDS14a] [GDS14b] [GDS14c] [GDS14d] [GDS14e] [HMT14] [NAO12] [Fra14] [Pon14] [Ser14a] [Ser14b] [Ste13] [TCU13b] [Uch13] [VaP14] [Wer12] [APÊNDICE A]

Fase 2: O estudo de base teórica contribui para a definição do protocolo de estudo de caso, detalhado no APÊNDICE B. Na fase de estudos empíricos (F2) buscou-se entender e ampliar o conhecimento sobre a adoção de métodos ágeis em OP por meio da execução de um estudo empírico na forma de estudos de caso em duas OP brasileiras. A Tabela 2, apresenta a abordagem metodológica utilizada nesta fase.

Tabela 2 - Abordagem metodológica utilizada na fase 2.

ABORDAGEM	DESCRIÇÃO
Estratégia de pesquisa	Qualitativa
Tipo de pesquisa	Exploratória
Método de pesquisa	- Estudo de caso (interpretativa) - Duas organizações públicas, sendo quatro equipes/projetos de desenvolvimento de software e divididos em duas dimensões: organizacional e projeto
Tipo de instrumento de coleta de dados	- Entrevista com roteiro semiestruturado - Análise documental
Tipo do registro dos dados coletados	- Gravações - Anotações de campo - Transcrição das entrevistas
Tipo de dados analisado	Qualitativo
Forma de análise de dados	- Uso da técnica de análise de conteúdo dos dados coletados empiricamente - Identificação das categorias de análise, incluindo informações sobre a organização, projeto, equipe, relacionamento com o cliente e o negócio, processos e práticas, bem como, sobre o conhecimento e a experiência em métodos ágeis

Fase 3: Na fase 3 (F3) buscou-se consolidar os resultados teóricos e empíricos alcançados nas fases 1 e 2, formando o arcabouço de resultados teóricos e empíricos. O arcabouço contém as razões, benefícios, práticas utilizadas, problemas e desafios, bem como, as lições aprendidas concernentes a adoção de métodos ágeis em OP. Cada aspecto identificado contém uma descrição detalhada para facilitar a sua compreensão. Essa fase intermediária foi necessária para apoiar a formulação do conjunto de recomendações para adoção de métodos ágeis em OP.

Fase 4: A última fase (F4) da pesquisa consistiu na elaboração e consolidação do conjunto de recomendações para adoção de métodos ágeis em OP. Um conjunto preliminar de recomendações foi elaborado a partir do arcabouço de resultados teóricos e empíricos. Posteriormente, novas entrevistas com especialistas foram realizadas visando buscar algum grau de estabilidade no conjunto de recomendações. Vale ressaltar que embora uma fase intermediária (F3) foi estabelecida para consolidar os resultados das fases anteriores (F1 e F2), em algumas situações foi necessário consultar novamente os estudos teóricos e o texto transcrito dos estudos empíricos para esclarecer alguns pontos.

3 REFERENCIAL TEÓRICO

O referencial teórico constitui uma importante etapa da pesquisa, contendo os principais conceitos e teorias da área estudada. Dessa maneira, a seção 3.1 descreve a origem da introdução de processos e padrões de processo de software em OP, considerando os países protagonistas Estados Unidos da América e Reino Unido e o Brasil. A seção 3.2 traz os conceitos sobre os métodos ágeis de desenvolvimento de software. A seção 3.3 mostra a adoção de métodos ágeis em OP, bem como, algumas recomendações oficiais de governo sobre o tema. Por fim, a seção 0 consolida os resultados teóricos.

3.1 Processos e Padrões de Processos de Software

Quando se trabalha na elaboração de um produto ou sistema, é importante seguir uma série de passos previsíveis, um roteiro que ajude a criar um resultado de alta qualidade e dentro do prazo estabelecido. Esse roteiro é denominado “processo de software” [Pre11]. O Modelo Cascata [Roy70], primeiro modelo de processo de desenvolvimento de software a ser publicado, é um exemplo de processo de software dirigido a planos, onde deve-se planejar e programar todas as atividades do processo antes de trabalhar nelas. Os principais estágios do Modelo Cascata refletem diretamente as atividades fundamentais do desenvolvimento: (1) análise e definição de requisitos; (2) projeto de sistemas e software; (3) implementação e testes [Som11].

Em termos mais genéricos, um padrão de processo fornece um modelo (*template*), um método consistente para descrever soluções de problemas no contexto do processo de software [Amb98]. Combinando o modelo de processo em cascata e estabelecendo padrões, os governos norte-americanos e britânicos buscaram elaborar padrões de processos de desenvolvimento de software que melhor atendessem suas necessidades, disciplinando e estruturando o desenvolvimento de software. Infelizmente, em parte devido á conveniência na contratação de serviços de desenvolvimento de software, o governo norte-americano e o britânico produziram uma visão equivocada do Modelo Cascata, enfatizando uma interpretação puramente sequencial do modelo, onde uma nova etapa não poderia ser iniciada se a etapa anterior não tivesse sido completada [Boe06].

3.1.1 Processos e Padrões de Processos de SW em OP Norte-Americanas e Britânicas

Desde o estabelecimento da Engenharia de Software (ES), o governo dos Estados Unidos da América (EUA) e do Reino Unido (UK) sempre buscaram implantar padrões e normas para estruturar o desenvolvimento de software em OP. Nos EUA, o *United States Department of Defense* (DoD) instituiu os padrões MIL-STDs e DOD-STDs entre 1974 e

1998 [Mcd10]; igualmente, o governo do Reino Unido determinou a adoção do padrão *Structured Systems Analysis and Design Method* (SSADM) de 1981 até meados da década de 2000 [Mid99]. Os primeiros padrões foram inspirados implicitamente no modelo conhecido como *Waterfall* (Cascata) apresentado por Winston Royce na década de 1970 [Roy70], que possui em sua estrutura uma ordem formal de elementos do processo e um fluxo de trabalho que descreve como cada um destes elementos se relaciona uns com os outros [Pre11]. Entretanto, os padrões iniciais não consideraram as recomendações de Royce [Roy70] que sugeria iterações não apenas após o último passo denominado “Operações”, mas também recomendava, após cada passo, uma revisão do anterior; além disso, após a conclusão do primeiro ciclo, Royce sugeria que todo o ciclo fosse refeito ao menos duas vezes. Pelo contrário, eles foram concebidos e usados como um modelo sequencialmente ordenado, de apoio a contratos de governo, no qual o desenvolvimento de software é visto como uma via de mão única que fluí para frente através das suas diversas fases [Mcd10] [Mid99].

3.1.1.1 A Adoção dos Padrões MIL-STDs e DODs

Inicialmente, o DoD enfrentou diversas falhas de projeto que ocasionaram objeções aos padrões MIL-STD-1679 e DOD-STD-2167 [Mcd10]. Concomitantemente, novos modelos baseados no pensamento de desenvolvimento incremental de software surgiram para resolver as fraquezas reais e perceptíveis dos modelos baseados na abordagem Cascata. Barry Boehm foi o primeiro a enfatizar a importância da iteração no desenvolvimento de software. Ele publicou o Modelo Espiral em 1988 [Boe88], ao pensar que os usuários realmente não sabiam o que eles queriam até que eles visualizassem o software em funcionamento. O Modelo Espiral contém elementos semelhantes ao Modelo Cascata, porém o ciclo é repetido em iterações. A cada iteração, uma versão do software é construída e, à medida que as iterações avançam, as versões tornam-se mais completas, de maneira progressiva. O *feedback* do cliente obtido após o término de cada iteração fundamenta as próximas etapas de análise, planejamento e desenvolvimento. Isso é semelhante ao pensamento original de Royce de “fazê-lo duas vezes” [Roy70], o que muda são os ciclos ilimitados do Modelo Espiral.

Assim, sob forte influência de Barry Boehm, o DoD publicou em 1988 uma revisão da norma anterior [Mcd10]. O padrão posterior, DOD-STD-2167A, procurou esclarecer e enfatizar a necessidade de uma entrega incremental, entretanto o desenvolvimento de projetos de softwares baseados no Modelo Cascata continuava inabalável, embora esforços para neutralizar seu uso foram realizados através da norma MIL-STD-498,

publicada em 1994 [Wer12]. Em 1996, o congresso norte-americano aprovou a série DOD-5000 de regulamentos, com base no Modelo Espiral, para reforçar a necessidade do desenvolvimento iterativo e incremental. Porém, a abordagem de desenvolvimento evolutiva introduzida no padrão DOD-5000, adotada ao longo da última década, continha falhas no processo de contratação do desenvolvimento de software [Wer12]. Por exemplo, o *Australian Department of Defence* (AU DOD) adotou o padrão DOD-5000 para suas próprias contratações, mas depois descobriu que o processo de contratação do padrão DOD-5000 era uma instância do Modelo Cascata. Cada etapa só poderia começar quando a etapa anterior fosse concluída. Orçamentos para os projetos foram estimados em um estágio inicial somente quando o amplo escopo do projeto foi detalhado, o que aumentou o volume de trabalho nas etapas iniciais e dificultou a inclusão de mudanças nas etapas seguintes [JVC05]. Assim, embora o desenvolvimento com base no Modelo Espiral foi o indicado para ser o processo chave, os padrões usados para estruturar o desenvolvimento de software no DoD têm sido inerentemente Cascatas por natureza, mesmo que exista a intenção de ser evolutivo [Han08], mantendo uma abordagem inflexível no DoD [Wer12].

3.1.1.2 A Adoção do Padrão Structured Systems Analysis and Design Method

O estudo de Middleton [Mid99] revelou que o resultado do uso da abordagem *Structured Systems Analysis and Design Method* (SSADM) foi percebida como prescritiva, onerosa e de difícil aplicação, demonstrando algumas fragilidades relacionadas com: (1) a incerteza inerente de projetos de ES; (2) a comunicação com o usuário; (3) o crescimento profissional das pessoas; (4) além de não refletir a forma como as pessoas trabalhavam na prática. Além disso, o estudo mostrou que na prática ocorriam mudanças fundamentais de estratégias durante à execução de projetos, devido às mudanças da alta administração. Ou seja, a suposição de um contexto estratégico estável e coerente mostrou-se inválida na prática. Igualmente, a necessidade de começar com requisitos fixos em todos os casos foi um obstáculo e tornou-se inviável. Como consequência disso, várias funcionalidades foram entregues sem satisfazer as necessidades dos usuários. Outrossim, a falta de entregas parciais de software, assim como, o longo tempo de espera e a grande quantidade de documentação foram aspectos que levaram o índice de comprometimento dos projetos para baixo.

As evidências dos projetos estudados por Middleton [Midd99] mostrou que a abordagem SSADM limitou a contribuição do usuário para envolvimento em vez de sua participação com maior comprometimento. Ela concentrou o desenvolvedor sobre as técnicas de desenho técnico do projeto de software e não nas habilidades sociais de

facilitação do processo humano de comunicação. Ao incentivar esta prática, a comunicação real foi perdida, em parte porque desenhos técnicos tendem a ser imprecisos na prática. Por fim, Middleton concluiu que abordagens baseadas no Modelo Cascata, poderiam não ser a melhor maneira de desenvolver softwares para a maioria dos projetos da AP, sendo necessário investir em um modelo de desenvolvimento de software mais adequado para lidar com a incerteza estratégica.

3.1.1.3 A Criação do Capability Maturity Model Integration

O desenvolvimento do *Capability Maturity Model for Software* (SW-CMM) começou formalmente em 1986 com o *Software Engineering Institute* (SEI) da *Carnegie-Mellon University* e o *MITRE Corporation*. O objetivo era produzir um roteiro razoável para o governo federal dos EUA avaliar as empresas privadas que eram contratadas na área de desenvolvimento de software [BoT03]. Em 1987, uma equipe liderada por Watts Humphrey publicou a proposta inicial. Em 1991, o conceito evoluiu para a Versão 1.0 do SW-CMM [Pau+91]. Em 1993, uma melhoria da versão 1.1 foi lançada [Pau+93] e, posteriormente, publicada como livro em 1995 [CaS95]. Com o tempo, o SW-CMM ganhou força como um guia para melhorar a capacidade do desenvolvimento de software. O DoD incluiu níveis de maturidade do processo CMM como critério de exclusão de empresas privadas para muitos de seus maiores contratos de aquisição de software [BoT03].

Simultaneamente, a *International Standards Organization* (ISO) estava trabalhando em processos de software e desenvolveu dois padrões ISO/IEC 15504 [ISO12] e ISSO/IEC 12207 [NBR09]. Estas normas cobriam grande parte do conteúdo do SW-CMM. Em 1997, o DoD e o *National Defense Industrial Association* com o objetivo de integrar os diversos CMMs existentes e harmonizar a convivência com as normas ISO, iniciaram um esforço conjunto nessa direção. O resultado foi o lançamento da suíte de produtos *Capability Maturity Model Integration* (CMMI) em 2001 [ACT01].

3.1.1.4 Os Métodos Guiados por Plano

Com o aumento da importância da ES, métodos para “disciplinar” o desenvolvimento de softwares começou a surgir. Nesse sentido, o DoD desenvolveu uma série de documentos de orientação, representados por MIL-STD-1679, DOD-STD-2167, DOD-STD-2167A, MIL-STD-498 e seus sucessores. O governo do Reino Unido desenvolveu o padrão SSADM. Semelhantemente, o SEI e parceiros desenvolveram o SW-CMM, CMMI, *Personal Software Process* (PSP) [Hum96] e o *Team Software Process* (TSP) [Hum00]. Ao mesmo tempo, empresas comerciais, desenvolveram padrões semelhantes, tais como: *Rational*

Unified Process (RUP) [Kru01], uma implementação comercial do Processo Unificado [JBR99]. Igualmente, a ISO criou os padrões ISO/IEC 15504 e ISO/IEC 12207. Todos eles proporcionaram uma considerável contribuição quanto à sua estrutura utilizável no trabalho de ES e forneceram um roteiro razoavelmente eficaz para as equipes de software [Pre11].

Nesse sentido, Barry Boehm & Richard Turner cunharam o termo *plan-driven methods*[2] ("métodos guiados por plano") para descrever abordagens de desenvolvimento de software que são baseadas na expectativa de que um plano (roteiro) é um comprometimento [BoT03]. Eles definem um processo adequado guiado por plano como sendo um processo que tem "a capacidade inerente (...) de produzir resultados planejados" [BoT03]. Como visto, eles originaram-se no mundo de contratos governamentais, onde é um desafio administrar contratos nas extensões exigidas na AP sem criar um plano detalhado, que é considerado um comprometimento [PoP11].

3.1.2 Processos e Padrões de Processos de SW em OP Brasileiras

Como visto anteriormente, os primeiros padrões, o modelo CMM e, mais recentemente, o CMMI derivaram da preocupação em estimular as empresas locais de software a melhorar seus processos e melhor atender as demandas governamentais [BoT03]. Na mesma direção, o Brasil, durante o período da reserva de mercado, criou o Projeto Fábrica de Software (PFS) em 1985 [Bra85], cujo objetivo era dotar o Brasil de uma estrutura de produção industrial de software, incluindo o desenvolvimento de metodologias, tecnologias e ferramentas, bem como, a formação de um grupo de excelência em ES voltadas ao atendimento do mercado nacional e internacional de software. Mais especificamente, seu objetivo era aumentar significativamente a produtividade dos programadores e a qualidade dos programas produzidos pela implantação de metodologias e ferramentas com base na utilização intensiva de técnicas formais de especificação e desenho; em aspectos relativos a gerência da configuração do processo e do produto; na inclusão de métodos quantitativos na formulação de métricas de controle de qualidade. As entidades envolvidas no PFS eram as seguintes OP: Centro Tecnológico para Informática (CTI), Empresa Brasileira de Pesquisa Agropecuária (Embrapa), e o Banco do Brasil. O projeto encerrou-se pouco tempo depois do término da reserva de mercado.

[2] A área de ES apresenta diversos conceitos que, muitas vezes, são considerados sinônimos. Para diminuir a discrepância entre os termos, este estudo considera "métodos guiados por planos", "métodos tradicionais" e "métodos pesados (*heavyweight*)" como sendo métodos prescritivos, onde (1) um planejamento detalhado e antecipado do projeto; (2) uma descrição das atividades, papéis e artefatos; (3) uma execução controlada e rigorosa do processo de desenvolvimento de software são necessárias para produzir softwares.

Mais tarde, em 2003, sob o pano de fundo de como melhorar os processos de software no Brasil, a um custo acessível, a Associação para Promoção da Excelência do Software Brasileiro (Softex) criou seu próprio modelo de referência de processo e método de avaliação de processo, conhecido como Melhoria de Processo do Software Brasileiro (MPS.BR), mais adequado à realidade brasileira e em oposição ao alto custo das avaliações CMMI, sobretudo para a pequenas e médias empresas (PMEs) brasileiras [Web+05].

A proposta destas duas abordagens nacionais e de outras tantas abordagens conhecidas mundialmente têm inspirado muitas OP brasileiras a definirem, padronizarem e melhorarem seus processos de desenvolvimento de software. Por exemplo, em 2001, o Serviço Federal de Processamento de Dados (Serpro) desenvolveu o “Processo Serpro de Desenvolvimento de Soluções” (PSDS) e o “Programa Serpro de Melhoria do Processo de Desenvolvimento de Soluções” (PSMDS) inspirados no RUP e no CMMI respectivamente [MPM09]. Em 2012, o Sistema de Administração de Recursos de Tecnologia da Informação (SISP) publicou o “Processo de Software para o SISP” (PSW-SISP) [SIS12]. Em 2014, a Empresa Brasileira de Pesquisa Agropecuária (Embrapa) publicou o “Modelo Corporativo de Processos de Software da Embrapa” (MCPSE) [CFD+14].

Além disso, de acordo com o último levantamento de Governança de TI realizado em 2012 pelo Tribunal de Contas da União [TCU13a] - que contou com a participação de 337 organizações da AP Federal - revelou que 26% das OP possuem um processo formal aprovado, publicado e obrigatório, tal como, 34% das OP possuem um processo informal repetido várias vezes e que implementa conceitos de qualidade de processo. Em geral, o que ocorre é a existência de várias metodologias semelhantes visando maximizar a estrutura e a ordem no desenvolvimento de software, cada uma com suas próprias vantagens e desvantagens em relação às outras. Isso mostra como a área de ES ainda é incipiente na AP sobretudo pela inexistência de uma solução universalmente aceita ou definitiva.

Por outro lado, o crescimento da utilização de métodos ágeis de desenvolvimento de software no mercado internacional [Ver15] e nacional [MSK+13], motivado sobretudo pelos vários benefícios alcançados na sua adoção, tem levado algumas OP a acreditarem em melhores resultados com o uso de métodos ágeis em vez de métodos guiados por plano. A partir deste interesse gradativo na adoção de métodos ágeis por OP, esta pesquisa se propõe a entender e ampliar o que se sabe sobre a adoção de métodos ágeis em OP, fornecendo um conjunto de recomendações para a sua adoção.

3.2 Métodos Ágeis para Desenvolvimento de Software

De acordo com Sommerville [Som11], no início da década 1980 e início da de 1990, havia uma visão generalizada de que a melhor maneira para conseguir o melhor software era por meio de um planejamento cuidadoso do projeto, qualidade da segurança formalizada, do uso de métodos de análise e projeto apoiado por ferramentas CASE (*Computer-Aided Software Engineering*) e do processo de desenvolvimento de software controlado e rigoroso. Essa abordagem, comumente conhecida como prescritiva, trouxe uma considerável contribuição quanto à estrutura utilizável no trabalho de ES e forneceu um roteiro razoavelmente eficaz para as equipes de software [Pre11], principalmente para equipes responsáveis pelo desenvolvimento de sistemas grandes e duradouros, como sistemas aeroespaciais e de governo [Som11]. Por outro lado, tais abordagens envolvem um *overhead* significativo no planejamento, projeto e documentação do sistema, sendo justificado quando o trabalho de várias equipes de desenvolvimento precisa ser coordenado, quando o sistema é crítico e quando muitas pessoas diferentes estão envolvidas na manutenção do software durante sua vida [Som11].

Em uma economia moderna, as condições de mercado mudam rapidamente e novos desafios competitivos surgem sem aviso. Aprender sobre o mercado e construir um produto para satisfazê-lo são práticas importantes para o desenvolvimento de soluções bem-sucedidas [PoP11]. No entanto, quando abordagens pesadas de desenvolvimento dirigidas a planos são aplicadas aos sistemas onde os requisitos são voláteis e incertos, o *overhead* é tão grande que domina o processo de desenvolvimento de software e dificulta que a equipe entregue o software que o negócio precisa e quando ele precisa. Se gasta mais tempo em análises de como o sistema deve ser desenvolvido do que no desenvolvimento do software e testes em si [Som11]. Como os requisitos se alteram, o retrabalho é necessário, e, pelo menos em princípio a documentação do projeto deve mudar com o software.

A insatisfação com abordagens prescritivas de desenvolvimento de software levou um grande número de profissionais a proporem, na década de 1990, novos métodos para o desenvolvimento de software para lidar com as limitações de projetos baseados em abordagens prescritivas [Som11]. Esses métodos passaram a ser chamados de “leves” (*lightweight*), em oposição aos anteriores, “pesados” (*heavyweight*), por não utilizarem as formalidades que caracterizavam os métodos prescritivos e por evitarem a burocracia imposta pela utilização excessiva de documentos [Bas08].

Devido ao grande número de menções aos processos “leves”, que emergiam como resposta para sanar as fraquezas reais e perceptíveis de projetos utilizando abordagens

prescritivas de ES [Pre11], em fevereiro de 2001 um grupo de dezessete líderes do pensamento da área de software, incluindo desenvolvedores, autores e consultores praticantes das abordagens “leves” (incluindo *Adaptive Software Development*, *Extreme Programming*, Scrum, Crystal, *Feature Driven Development*, *Dynamic System Development Method*) e “*Pragmatic programming*” se reuniram em um final de semana na cidade de Utah para discutir sobre suas experiências de trabalho e pontos em comum. O resultado do encontro deu origem ao manifesto ágil [BBB+01], uma declaração que são, na visão de seus proponentes, determinantes para entender o desenvolvimento de software como um processo criativo, adaptativo, e movido a incertezas. Ademais, desse encontro surgiu o termo “métodos ágeis” (também referenciados como metodologias ágeis) e mais tarde, alguns de seus criadores formaram a *Agile Alliance* [Agi14], os quais se aplicam o manifesto ágil. Posteriormente, Alistair Cockburn, um dos proponentes do manifesto ágil, publicou em 2002 o primeiro livro sobre métodos ágeis, intitulado “*Agile Software Development*” [Coc02].

3.2.1 Valores e Princípios dos Métodos Ágeis

O manifesto ágil é o embasamento filosófico de todos os métodos ágeis. Ele é formado por um conjunto de quatro valores e doze princípios, e se inicia da seguinte maneira [BBB+01]:

Manifesto para o desenvolvimento ágil de software

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

Indivíduos e interação *entre eles mais que processos e ferramentas*

Software em funcionamento *mais que documentação abrangente*

Colaboração com o cliente *mais que negociação de contratos*

Responder a mudanças *mais que seguir um plano*

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

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

A primeira frase do manifesto ágil afirma que ainda “estamos descobrindo melhores maneiras de desenvolver software...”. Para Gomes *et al.* [GWR14], pode-se inferir a partir dessa declaração, que a área de Engenharia de Software (ES) ainda é muito incipiente, quando comparada com outras engenharias, como a civil ou mecânica, sendo uma área nova com cerca de 40 anos. Logo, não se propõe uma solução definitiva, mas bons indícios

do que possa ser um melhor caminho rumo ao sucesso nos projetos. Ou seja, a ES ainda está em fase de aprendizado.

O Manifesto Ágil ressalta o que mais tem valor para os métodos ágeis [Bas08]. Processos, ferramentas, documentação, contratos, e planejamento têm valor para o desenvolvimento de software, mas terão mais sentido e mais valor depois que saber como lidar com pessoas, ter o cliente colaborando para encontrar a melhor solução, entregar o software com qualidade e adaptar-se às mudanças [KaR02]. Ou seja, processos bem estruturados de nada adiantam se as pessoas não os seguem, software bem documentado também não adianta se não satisfaz os requisitos ou não funciona, e assim por diante [Waz13]. É importante notar que os valores do manifesto ágil são declarações relativas, e não absolutas. Eles representam uma ponderação de alternativas, em vez de uma escolha binária [BoT03].

Outrossim, o Manifesto Ágil não rejeita os processos nem as ferramentas, a documentação abrangente, a negociação de contratos ou o plano preestabelecido, ao reconhecer explicitamente que “...mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda”. Mas indica que eles têm importância secundária quando comparados com indivíduos e interações, com software funcionando, com colaboração com o cliente e com respostas rápidas a mudanças [PWM14]. Por outro lado, se esta última frase do Manifesto Ágil é negligenciada, então os valores podem ser erroneamente interpretados como sendo dicotomias exclusivas. Porém, eles não são. Por exemplo, embora os métodos ágeis valorizem mais software funcionando do que documentação abrangente, ainda há uma necessidade de alguns documentos, mesmo que no formato ágil.

Esses valores são descritos a seguir, inspirados na visão de Cockburn [Coc02] e Gomes *et al.* [GWR14]:

- **Indivíduos e interações** mais que processos e ferramentas: Métodos ágeis rejeitam a ideia de que pessoas envolvidas no desenvolvimento de software são peças substituíveis, onde um papel pode ser preenchido por outra pessoa, se um indivíduo deixa o projeto. Em determinados ambientes esse pensamento pode ser conveniente, porém em trabalhos criativos, como o de desenvolvimento de software, a individualidade e a competência pessoal é importante, portanto a substituição de um indivíduo é difícil. Ou seja, métodos ágeis enfatizam o trabalho individual das pessoas sobre papéis e incentivam a interação entre os indivíduos. Por outro lado, processos e ferramentas, também são importantes, sendo mais difícil trabalhar sem elas. Porém, novas alternativas e soluções para problemas

emergem das interações entre as pessoas, sendo preferível usar um processo não documentado com boas interações do que um processo documentado com interações hostis.

- **Software funcionando** mais que documentação abrangente: Em projetos ágeis, software de qualidade testado e funcionando é a medida principal de progresso. Documentos contendo requisitos e informações do projeto podem ser úteis e devem ser usados apenas o suficiente para apoiar o trabalho dos desenvolvedores, porém o código compilado e testado revela informações mais relevantes sobre a equipe e o processo de desenvolvimento, tal como, problemas a serem resolvidos e satisfação do cliente.
- **Colaboração com o cliente** mais que negociação de contratos: O pressuposto básico por trás dessa declaração de valor é a satisfação do cliente [KaR02], que pode ser alcançada quando clientes e desenvolvedores estão na mesma equipe, bem como, quando ambos estabelecem uma relação de estreita colaboração (o que não exclui a possibilidade de conflitos) para a tomada de decisão conjunta e melhoria da comunicação. Tentar criar barreiras de proteção entre clientes e desenvolvedores não irá resolver muita coisa se não houver colaboração entre ambas as partes. Então, ao invés de procurar solucionar os problemas incluindo novas cláusulas, redigindo contratos rígidos, é preferível trabalhar com uma outra abordagem com o cliente, através de um clima de colaboração e confiança. Ou seja, boa colaboração pode tornar contratos desnecessários, assim como, se o contrato está em perigo, uma boa colaboração pode salvar a situação do contrato.
- **Responder a mudanças** mais que seguir um plano: Métodos ágeis admitem que os planos são úteis e planejamento é incluído em projetos ágeis. Porém, em vez de seguir um plano rigoroso, métodos ágeis consideram que tudo muda em software. Os requisitos mudam, o projeto muda, os negócios mudam, a tecnologia muda, os membros da equipe mudam. Assim, é natural e inevitável que haja mudanças. Se tudo muda o tempo todo, então um movimento para frente torna-se arriscado. De acordo com Kent Beck [Bec99], o problema não é a mudança em si, porque ela vai acontecer; o problema é, na verdade, a falta de habilidade para se lidar com a mudança quando ela chega. Por conta disso, métodos ágeis através de diversas práticas oferecem instrumentos para lidar com as mudanças, permitindo que o desenvolvimento de software aconteça de maneira flexível e segura em ambientes onde as alterações são frequentes. Além disso, acredita-se que mudanças são ótimas oportunidades para que o produto de software

desenvolvido seja mais aderente às necessidades do cliente, além de contribuírem para os resultados desejados. Assim métodos ágeis preconizam que mudanças devem ser aceitas, além de serem bem-vindas.

Além dos quatro valores, doze princípios foram estabelecidos para fornecer uma definição mais detalhada e apoiar a filosofia do Manifesto Ágil, apresentados na Tabela 3.

Tabela 3 - Princípios dos métodos ágeis.

PRINCÍPIO	DESCRIÇÃO
Satisfação do cliente	A maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
Aceitar as mudanças	Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
Entrega incremental	Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
Trabalho em equipe e envolvimento do cliente	Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
Pessoas, e não processos	Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
Comunicação frente a frente	O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa frente a frente.
Software funcionando	Software funcional é a medida primária de progresso.
Ambiente sustentável	Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
Qualidade	Contínua atenção a excelência técnica e bom *design* aumenta a agilidade.
Manter a simplicidade	Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
Equipes auto-organizáveis	As melhores arquiteturas, requisitos e *designs* emergem de times auto-organizáveis.
Aprendizado e melhoria contínua	Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Segundo Pressman [Pre11], nem todo método ágil aplica esses doze princípios atribuindo-lhes pesos iguais, e alguns métodos preferem ignorar (ou pelo menos relevam) a importância de um ou mais desses princípios. Porém, os princípios definem um espírito ágil mantido em cada um dos métodos ágeis que serão apresentados na sequência.

3.2.2 Visão Geral dos Métodos Ágeis

Jacobson [Jac02] destaca que a penetração da mudança é o principal condutor para a agilidade, uma equipe ágil é aquela rápida e capaz de responder apropriadamente a mudanças. Pressman [Pre11] acrescenta que agilidade consiste em algo mais que uma resposta à mudança, abrangendo a filosofia proposta no manifesto ágil. Ela incentiva a estruturação e as atitudes em equipe que tornam a comunicação mais fácil entre os todos

os membros do projeto. Enfatiza a entrega rápida do software e diminui a importância dos artefatos intermediários; assume o cliente como parte da equipe de desenvolvimento e trabalha para eliminar a atitude de “nós e eles”; reconhece que o planejamento em um mundo incerto tem seus limites e que o plano de projeto deve ser flexível. Prikladnicki *et al.* [PWM14] reforça que ser ágil está associado a uma mudança cultural, a uma nova forma de pensar, onde o enfoque maior é nas pessoas e não em processo, e o seu conjunto de valores, princípios e práticas. Abrahamsson *et al.* [ASR+02] discute que embora existam várias abordagens para o desenvolvimento ágil de software, elas compartilham algumas características fundamentais:

- **Desenvolvimento iterativo e incremental:** Os incrementos são pequenos e, normalmente, são entregues e disponibilizados aos clientes em ciclos curtos de desenvolvimento, também conhecidos como iterações.
- **Desenvolvimento colaborativo:** Os clientes são envolvidos no processo de desenvolvimento e ficam próximos da equipe de desenvolvimento para obter *feedback* rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos [Som11].
- **Desenvolvimento adaptativo:** O sistema é construído em uma série de versões, cada qual contendo uma série de incrementos. Os clientes, usuários finais e demais partes interessadas colaboram na especificação e avaliação de cada incremento. Eles podem sugerir modificações ao software e novos requisitos que devem ser implementados em uma versão ou incremento posterior. Os métodos ágeis são concebidos para lidar com mudanças em contextos onde os requisitos são incertos e voláteis, onde clientes e desenvolvedores não possuem o total conhecimento a respeito dos requisitos no início do projeto.

3.2.3 Alguns Exemplos de Métodos Ágeis

Segundo o relatório “*9th ANNUAL State of Agile Development Survey*” [Ver15] (uma pesquisa *online* com mais de 3.900 pessoas) conduzido pelo fornecedor de ferramentas de desenvolvimento ágil VersionOne, dentre os métodos ágeis Scrum, Scrum/XP híbrido, Ágil customizado, Scrum/Kanban e Kanban são os mais amplamente adotados atualmente. Embora muitos outros encontram-se em uso no setor, incluindo Dynamic Systems Development Method (DSDM), Adaptative Software Development (ASD), Feature Driven Development (FDD) e Família Crystal, somente Extreme Programming (XP), Scrum e Kanban serão descritos neste estudo por restrição de tamanho do volume.

3.2.4 Extreme Programming

Extreme Programming (XP) foi proposto por Kent Beck, Ward Cunningham e equipe na década de 1990, ao refletir sobre melhores caminhos para resolver problemas básicos no desenvolvimento de software, incluindo atraso no cronograma, cancelamento antecipado do projeto, sistema em produção com alta taxa de erros, funcionalidades entregues sem atender as necessidades de negócio, sistema com baixo valor agregado e de difícil manutenção [Bec99]. Kent Beck, através de um projeto na *DaimlerChrysler*, criou um novo estilo de desenvolvimento de software capaz de tratar esses riscos em todas as etapas do processo de desenvolvimento, o *Extreme Programming* (XP). A inovação de XP consistiu em juntar técnicas de desenvolvimento comprovadas por décadas em uma única abordagem e garantir que elas sejam executadas ao extremo, assim como, assegurar que elas apoiem umas às outras ao máximo [Bec99]. Segundo James Shore & Shane Warden, dentre todos os métodos ágeis conhecidos, XP é o mais completo [Sho08]. Ele assume que a volatilidade dos requisitos existe e, em vez de tentar eliminá-la, trata o desenvolvimento do software a partir de uma abordagem flexível e colaborativa, na qual desenvolvedores e clientes fazem parte de uma única equipe que tem o propósito de produzir software de alto valor agregado [Bas14]. XP é formado por um conjunto de valores, princípios e práticas que ditam como as várias atividades no desenvolvimento de software (planejamento, testes, desenvolvimento, projeto e implantação) devem ocorrer [Bec99].

3.2.4.1 Valores e Princípios do XP

Valores são ideais abstratos, porém identificáveis e distintos [Sho08]. Kent Beck & Cynthia Andres definiram os valores de XP, como sendo: comunicação, simplicidade, *feedback*, coragem e respeito [BeA04]. A comunicação deve ocorrer de maneira contínua entre os próprios desenvolvedores, bem como, entre a equipe de desenvolvimento e o cliente para criar um senso de equipe e cooperação eficiente [BeA04]. A simplicidade consiste em descartar elementos desejados, porém dos quais não são precisos realmente para fazer algo funcionar [Sho08]. Comunicação e simplicidade têm uma relação mútua de suporte, quanto mais se comunica, mais claramente se vê o que precisa ser feito e se tem mais certeza sobre o que não precisa ser feito [Bec99]. O *feedback* contribui para o aprendizado no sentido de aprender a melhorar o trabalho em todas as situações possíveis [Sho08]. A coragem utilizada de forma isolada, sem contrabalancear os demais valores é perigosa. Fazer algo sem levar em conta as suas consequências não é um exemplo de equipe de trabalho eficiente [BeA04]. Por outro lado, quando utilizada em conjunto com outros valores ela se torna poderosa. A coragem de falar a verdade para os envolvidos

quando eles precisam ouvi-las, promove a comunicação e confiança [Sho08]. O respeito entre as pessoas é a parte fundamental para que os demais valores colaborem entre si. Sem respeito, a comunicação e o *feedback* serão pouco eficientes e a coragem de um membro poderá ser nociva aos demais por não estar alinhada com os interesses da equipe [Bas08]. XP considera que todas as pessoas envolvidas no desenvolvimento de software têm o mesmo valor como ser humano. Isto significa que ninguém é mais importante do que qualquer outra pessoa, sendo assim as opiniões e considerações de cada indivíduo da equipe precisam ser respeitadas [BeA04].

Os princípios são o elo de ligação entre os valores abstratos e as práticas concretas de XP. Kent Beck & Cynthia Andres prescreveram 14 princípios para XP [BeA04]. Outros princípios podem coexistir, porém os 14 princípios norteadores são aplicáveis a todos os softwares.

1. **Humanidade:** Software são desenvolvidos por pessoas. É importante considerar que pessoas têm fraquezas e necessidades individuais que devem ser atendidas e equilibradas com as necessidades da equipe e objetivos de negócio.
2. **Economia:** A equipe precisa verificar se o que está sendo feito tem valor para o negócio e atende aos objetivos de negócio.
3. **Benefício Mútuo:** Cada atividade executada deve beneficiar todos os envolvidos. Testes, refatorações e metáforas são exemplos de atividades que trazem ganhos imediatos e futuros para os desenvolvedores pois melhoram o entendimento e a manutenibilidade do software. Igualmente, os clientes são beneficiados, pois menos erros são introduzidos no sistema. Da mesma maneira, gerentes se beneficiam, uma vez que a ausência de um dos membros da equipe pode ser superada mais facilmente.
4. **Auto-semelhança:** Este princípio sugere que quando equipes conseguem encontrar boas soluções que funcionem em um contexto específico, também procurem aplicá-las em um outro contexto, mesmo em condições ou escalas diferentes, correndo o risco de não funcionar.
5. **Melhoria:** A excelência na utilização de XP é alcançada através do aperfeiçoamento contínuo dos aspectos envolvidos no desenvolvimento de software, incluindo *design*, código, testes, etc. O ideal é encontrar um ponto de partida, e na sequência buscar a melhoria contínua.
6. **Diversidade:** Equipes de desenvolvimento de software precisam reunir uma variedade de competências, opiniões e perspectivas para identificar problemas,

pensar em várias maneiras de resolvê-los e implementar as melhores alternativas. Isso aumenta o universo de ideias e potenciais soluções.

7. **Reflexão:** Constantemente a equipe deve pensar sobre como estão trabalhando e porque estão trabalhando de uma determinada maneira, assim como, deve refletir sobre fatores de sucesso e falhas de projetos, identificando pontos de melhoria e experiências bem-sucedidas que podem ser repetidas em outros projetos.
8. **Fluxo:** A equipe deve ser capaz de entregar software funcionando e de qualidade frequentemente, no período de tempo mais curto possível. Pequenos incrementos de software, entregues constantemente, melhora o fluxo de trabalho da equipe e permite que ela receba *feedback* mais rápido sobre o que está sendo feito.
9. **Oportunidade:** Problemas e mudanças inesperadas ocorrem no desenvolvimento de software, transformá-los em oportunidades de aprendizado e melhoria contribui para o crescimento pessoal, trabalho em equipe e qualidade do software produzido.
10. **Redundância:** A qualidade de software deve ser garantida através de práticas redundantes que reduzam as chances de erros. Por exemplo, através da Programação em Par, Integração Contínua e Envolvimento do Cliente, problemas difíceis e defeitos críticos no desenvolvimento de software podem ser resolvidos.
11. **Falha:** Consiste em experimentar na prática diferentes abordagens para resolver o mesmo problema, ao invés de discutir várias alternativas demoradamente no nível teórico.
12. **Qualidade:** Existe um efeito humano gerado pela qualidade. Todos querem fazer um bom trabalho, e trabalham muito melhor se sentem que estão fazendo um bom serviço.
13. **Passos Pequenos:** Passos pequenos reconhece que a sobrecarga de avançar em pequenos passos é muito menor do que quando equipes abandonam alternativas provocadas por mudanças em grandes passos.
14. **Aceitação da Responsabilidade:** A responsabilidade não deve ser atribuída, mas sim aceita. Cabem as pessoas decidirem se elas são responsáveis ou não por determinadas atividades.

3.2.4.2 Práticas do XP

As práticas são os princípios aplicados a um tipo específico de projeto, sendo utilizadas no cotidiano de uma equipe XP para o desenvolvimento efetivo de software [Sho08], e refletem os princípios dos métodos ágeis [Som11]. O poder de XP está no conjunto das práticas [Bas14]. Onde uma prática é fraca, os pontos fortes das outras práticas compensarão essa fraqueza [Bec99]. Elas são dependentes do contexto, sendo que a equipe XP deve combinar diferentes práticas para atender a essas situações, criando sinergia, porém elas podem não ser suficientes para desenvolver softwares com resultados positivos [BeA04]. Elas formam o núcleo de excelência para equipes de desenvolvimento. Problemas não cobertos por uma das práticas podem surgir. Quando isso acontecer, é o momento de observar os valores e os princípios do XP, que não devem mudar para se adaptar a uma situação específica para encontrar uma solução adequada para o problema [BeA04]. Kent Beck & Cynthia Andres [BeA04] organizaram as práticas de XP em 13 práticas primárias e 11 práticas corolárias, totalizando 24 práticas. Na sequência, um subconjunto de práticas de XP para código é apresentado:

1. **Programação em Pares:** Equipes de dois desenvolvedores escrevem (analisam, desenvolvem e testam) todo o código em um único computador, sentados lado a lado, utilizando um mouse e um teclado. As duplas devem ser formadas naturalmente, nunca atribuídas, e são trocadas regularmente para melhorar a coesão da equipe, disseminar habilidades de desenvolvimento e conhecimento para toda a equipe.
2. ***Build* em 10 minutos:** O *build* é composto pela execução automatizada de todas as atividades necessárias para colocar o software em perfeito funcionamento, incluindo a compilação, testes e verificação de código, no máximo em 10 minutos.
3. **Integração Contínua:** O objetivo desta prática é manter o código-fonte produzido pelos desenvolvedores integrado e o software pronto para ser entregue com a maior frequência possível.
4. **Desenvolvimento Dirigido por Testes:** Ao praticar o Desenvolvimento Dirigido por Testes (*Test Driven Development* - TDD) é possível detectar erros de programação assim que eles são cometidos, para isso TDD inclui quatro atividades: (1) refletir sobre uma pequena funcionalidade a ser implementada; (2) adicionar um teste para esta nova funcionalidade, o que inevitavelmente irá falhar porque ele é escrito antes da funcionalidade a ser implementada; (3) implementar o código para que o teste seja satisfeito, sendo que o novo código implementado poderá não ser o ideal, isso é aceitável porque na sequência ele será melhorado.

Além disso, deve-se verificar se todos os testes anteriores continuam sendo satisfeitos; (4) refatorar o código implementado, nesse ponto o código-fonte é revisado e melhorias são incluídas.

5. ***Design* Incremental:** TDD é um exemplo de *Design* Incremental no nível de classes e métodos individuais. No entanto, *Design* Incremental vai além do TDD, ao considerar todos os níveis do projeto. Assim como no TDD, os desenvolvedores iniciam o projeto com a solução mais simples que funcione, posteriormente, eles fazem adições incrementais ao *design* na medida certa de sofisticação para suportar as necessidades de negócio atuais, sem agregar complexidade antecipada ou despender maiores esforços para construí-lo.
6. **Código Compartilhado:** Esta prática permite que qualquer desenvolvedor realize qualquer alteração necessária em qualquer parte do código a qualquer momento.
7. **Código e Testes:** O código e os testes são os únicos artefatos que devem ser preservados permanentemente pela equipe XP. Outros documentos que se façam necessários podem ser gerados automaticamente a partir do código e testes.
8. **Repositório de Código Unificado:** Esta prática complementa e vai além da abordagem de Código Compartilhado, recomendando que exista apenas um repositório de código, preferencialmente sem ramificações.

3.2.4.3 Ciclo de Vida do XP

Kent Beck definiu as 6 fases para um projeto XP ideal, como sendo: exploração, planejamento, iterações para as versões, produção, manutenção e morte [Bec99]. Equipes de desenvolvimento que utilizam XP executam quase todas as atividades de desenvolvimento de software simultaneamente, incluindo análise, projeto, codificação, testes e implantação [Sho08]. Estas atividades são realizadas através de iterações, que são incrementos de trabalho que levam uma semana para serem concluídos, por exemplo. Em geral, no início da semana, a equipe de desenvolvimento se compromete a entregar algumas funcionalidades, descritas geralmente como histórias, que tem um valor de negócio importante para o cliente. No decorrer da semana, a equipe XP atua em praticamente todas as atividades do desenvolvimento de software para implementar cada um dos itens de trabalho selecionados para a semana. Por último, os desenvolvedores disponibilizam o software para revisão interna, que em alguns casos pode incluir a instalação para os próprios clientes.

3.2.5 Scrum

O Scrum foi proposto em 1993 por Jeff Sutherland e sua equipe, ao pensar sobre equipes auto-organizadas e o papel da gestão no processo de desenvolvimento, assim como, os conceitos de desenvolvimento de software orientado a objetos, processos empíricos, desenvolvimento iterativo e incremental, processos de software, produtividade e sistemas adaptativos complexos. Dois anos mais tarde, Ken Schwaber formalizou o primeiro artigo sobre Scrum no *Object-Oriented Programming, Systems, Languages & Applications* (OOPSLA) [Sch95]. Desde então, Jeff Sutherland e Ken Schwaber, juntos ou separadamente, tem publicado vários trabalhos sobre Scrum, incluindo "*Agile Software Development with Scrum"* [Sch01], "*Agile Project Management with Scrum"* [Sch04], "*Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust"* [Sch12] e o "*The Scrum Guide"* [Sch13].

De acordo com o relatório "*9th ANNUAL State of Agile Development Survey*" [Ver15], Scrum é o método ágil mais conhecido e adotado na atualidade, sendo uma fonte inspiradora para muitos dos pensamentos por trás dos valores e princípios do manifesto ágil [Scr12]. Isto significa que os valores e princípios do manifesto ágil se aplicam diretamente ao Scrum. Ele é baseado em um conjunto de valores, princípios e práticas que formam o *framework* Scrum, com um maior foco nos aspectos gerenciais do projeto de desenvolvimento de software [PrM14] [Rub13] [Scr12] [Som11]. O Scrum não prescreve a utilização de práticas de codificação, como Programação em Par e Desenvolvimento Dirigido por Testes (TDD). Desta maneira, ele pode ser utilizado com abordagens que consideram aspectos técnicos, como XP [Kni07] [Som11].

O Scrum tem como premissa a existência de um processo iterativo e incremental para o desenvolvimento, trazendo uma nova dimensão na capacidade de resposta e adaptabilidade da gestão dos processos [Sch04]. Segundo Prikladnicki & Magno [PrM14], partindo do princípio de que desafios fundamentalmente empíricos não podem ser resolvidos com sucesso utilizando uma abordagem tradicional de controle, na qual para um mesmo conjunto de variáveis de entrada pode-se esperar o mesmo resultado sempre, foram buscadas alternativas, como o Scrum. Ele trata problemas bem conhecidos no desenvolvimento de software, como a alta variação dos requisitos e o alto grau de imprevisibilidade, não como deficiências ou como problemas gerados por algum processo, mas como características da natureza de um projeto de desenvolvimento de software.

O Scrum maximiza a entrega de software de modo eficaz, adaptando-se à realidade das mudanças [PrM14]. Ele adota uma abordagem empírica, aceitando que o problema pode não ser totalmente entendido ou definido na análise e que provavelmente os requisitos

mudarão com o passar do tempo, e mantém o foco na maximização da habilidade da equipe em responder de forma ágil aos desafios emergentes [PrM14]. As funcionalidades de maior valor são desenvolvidas antecipadamente, enquanto se reflete sobre a necessidade ou não das menos prioritárias. Se mudanças forem necessárias, a equipe ágil poderá facilmente alterar as prioridades [PrM14]. De acordo com Prikladnicki & Magno [PrM14], sua principal motivação é o fato de que o desenvolvimento de software envolve muitas variáveis técnicas e de ambiente, como requisitos, recursos e tecnologia, que podem mudar durante o processo, tornando-o imprevisível e requerendo flexibilidade para acompanhar as mudanças.

3.2.5.1 Papéis do Time de Scrum

As pessoas envolvidas no processo de desenvolvimento formam o Time de Scrum e são organizadas em três papéis principais: *Product Owner* (dono do produto), *ScrumMaster* e os membros do Time de Desenvolvimento [Rub13] [Sch13] [Scr12]. Outros papéis podem coexistir ao usar o Scrum, porém o Scrum requer apenas os três papéis listados [Rub13]. Um Time de Scrum é auto-organizável e multifuncional [Rub13] [Sch13]. Time auto-organizável escolhe qual a melhor forma para completar seu trabalho, em vez de ser dirigido por outros de fora da equipe [Sch13]. Time multifuncional possui todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe [Sch13]. Ou seja, Scrum requer que um time multifuncional de pessoas que, entre elas, tenham todas as habilidades necessárias para projetar, construir, testar e entregar o produto de software desejado [Gom13] [Rub13] [Sch13] [Scr12].

3.2.5.2 Funcionamento do Scrum

O Scrum se inicia quando alguns *stakeholders* (partes interessadas) necessitam de um produto [Scr12]. O PO (dono do produto) é o maior interessado no software, é aquele que teve (ou que representa quem teve) a necessidade do produto, e por isso detém a visão do produto [Gom13]. Em última instância, ele é responsável por definir o que deve ser feito, assim como, determinar a sequência do trabalho a ser realizado, mantendo os itens de trabalho organizados na forma de uma lista de prioridades ordenadas, conhecida como *product backlog* [Rub13]. Desta maneira, o *Product Owner* é responsável pelo *product backlog*, incluindo seu conteúdo, disponibilidade e ordenação [Sch13], podendo ser apoiado por outros indivíduos, mas deve ser uma única pessoa [Scr12].

O *ScrumMaster*, conhecido como facilitador, é responsável por auxiliar a organização e o time a fazer o melhor uso do Scrum [Rub13] [Sch13] [Scr12]. Ele ajuda a

tornar o Scrum um processo mais vantajoso e produtivo de acordo com as especificidades da organização [Sch13] [Scr12]. Ao mesmo tempo, ele ajuda o *Product Owner* a encontrar técnicas para o gerenciamento efetivo do *product backlog* e comunicar claramente a visão e os itens do *product backlog* para o Time de Desenvolvimento [Sch13] [Scr12]. Ele também é responsável por proteger o Time de Desenvolvimento de interferências externas e assume um papel de liderança na remoção de impedimentos que inibem a produtividade da equipe (quando os próprios indivíduos não podem razoavelmente resolvê-los) [Rub13]. É importante ressaltar que o *ScrumMaster* não tem autoridade para exercer o controle sobre a equipe porque ela é auto-organizável, de modo que este papel não é o mesmo que o papel tradicional do gerente de projeto ou gerente de desenvolvimento [Rub13]. Isto significa que o *ScrumMaster* não determina o que cada membro do time deve ou não fazer, pelo contrário, o time se compromete com a entrega das funcionalidades, então, se auto-organiza, definindo por quem e em qual momento as tarefas serão realizadas [Gom13].

O Time de Desenvolvimento constrói o produto incrementalmente, em iterações ou ciclos curtos de tempo, chamados de *sprints* [Rub13] [Sch13] [Scr12]. Uma *sprint* é um período de tempo fixo, de uma a quatro semanas, com uma preferência por intervalos mais curtos [Scr12]. Em cada *sprint*, o Time de Scrum deve construir e entregar um incremento do produto de valor tangível para o cliente ou usuário [Scr12] [Rub13]. Cada incremento é um subconjunto funcional, com melhorias evidentes e reconhecíveis do produto, atendendo a critérios de aceitação compreendidos e é construído com um nível de qualidade chamado *definition of done* (definição de pronto) [Scr12].

Uma nova *sprint* inicia-se imediatamente após a conclusão da *sprint* anterior, e como regra não deve permitir qualquer alteração de escopo durante sua execução, no entanto, as necessidades do negócio, por vezes, pode tornar a adesão a esta regra uma prática impossível [Rub13], sendo necessário ajustar o escopo com o *Product Owner*, de forma a atender as necessidades do negócio das organizações [Sch12].

As *sprints* são compostas por uma reunião de planejamento da *sprint*, reuniões diárias, um incremento do produto, uma reunião de revisão da *sprint* e uma reunião de retrospectiva da *sprint* [Rub13] [Sch13] [Scr12]. O *product backlog* pode representar muitas semanas ou meses de trabalho, o que é muito mais do que pode ser concluído em uma única *sprint* curta [Rub13]. Para determinar o subconjunto mais importante de itens do *product backlog* que será desenvolvido na próxima *sprint*, o *Product Owner*, o *ScrumMaster* e o Time de Desenvolvimento realizam uma reunião denominada planejamento da *sprint*, a qual é descrita como tendo duas partes, cada uma ocupando a metade do tempo [Rub13] [Sch13] [Scr12].

A primeira parte delas consiste em determinar que incremento será entregue na próxima *sprint*. O *Product Owner* apresenta os itens de trabalho ordenados do *product backlog* para o Time de Desenvolvimento, e o Time de Scrum inteiro colabora para compreender o trabalho [Scr12]. O número de itens do *product backlog* a ser aceito na *sprint* é decidido unicamente pelo Time de Desenvolvimento [Scr12]. Para decidir quantos itens deve escolher, o Time de Desenvolvimento considera o estado atual do incremento do produto, o desempenho histórico do time, a capacidade atual do time e o *product backlog* ordenado [Scr12]. O Time de Desenvolvimento decide sozinho quanto trabalho deverá assumir. Nem o *Product Owner*, nem qualquer outro agente, pode forçar mais trabalho para o Time de Desenvolvimento [Scr12].

A segunda parte consiste em determinar como o trabalho necessário para entregar o incremento será realizado. Então se analisa tarefa por tarefa com mais detalhes, mais informações de negócio são apresentadas e é possível tomar diversas decisões de negócio e decisões técnicas [Gom13]. O Time de Desenvolvimento planeja e modela apenas o suficiente para terem confiança de que serão capazes de completar o trabalho durante a *sprint* [Scr12].

O resultado da reunião é um conjunto de itens de trabalho selecionados a partir do *product backlog* para desenvolvimento na *sprint* atual, conhecido como *sprint backlog* [Rub13] [Sch13] [Scr12], juntamente com um plano para entregar o incremento do produto, tornando visível todo o trabalho que o Time de Desenvolvimento identifica como necessário para completar a *sprint* [Scr12]. No final do planejamento da *sprint*, o Time de Desenvolvimento deve ser capaz de explicar ao *Product Owner* e ao *ScrumMaster* como pretende trabalhar como equipe auto-organizada para completar o objetivo da *sprint* e criar o incremento do produto de software [Sch13].

Uma vez que o Time de Scrum termina o planejamento da *sprint* e concorda com o seu conteúdo, o Time de Desenvolvimento, orientado pelo *ScrumMaster*, realiza todo o trabalho necessário para completar o objetivo da *sprint* [Rub13] [Sch13] [Scr12]. Ninguém diz ao Time de Desenvolvimento em que ordem ou de que maneira o trabalho contido no *sprint backlog* deve ser realizado. Ao invés disso, o Time de Desenvolvimento define o seu próprio trabalho em nível de tarefa e sua estratégia de organização para entregar um incremento do produto de software de boa qualidade de acordo com as necessidades do *Product Owner* [Rub13].

Diariamente é realizada uma reunião, conhecida como *Daily Scrum*, para verificar o andamento da *sprint* e inspecionar se o progresso obtido tende para completar o trabalho previsto no *sprint backlog* [Gom13] [Rub13] [Sch13] [Scr12]. Gráficos de *burndown* e *burnup*

têm sido utilizados para comunicar visualmente a progressão do trabalho em direção a conclusão do objetivo da *sprint*, assim como, têm sido adotados para visualizar a quantidade total de trabalho que permanece inacabado em cada *sprint*, necessários para completar uma versão do produto de software [Rub13]. Além dos grandes gráficos visíveis, é comum um Time de Scrum usar dispositivos visuais de *feedback* adicionais em seu espaço de trabalho. Um dos mais comuns é uma lâmpada de lava que é ligada sempre que a *build* automatizada é interrompida [Coh11].

Ao final de cada *sprint*, mais duas reuniões acontecem, a reunião de revisão e a reunião de retrospectiva [Rub13] [Sch13] [Scr12]. Na reunião de revisão, o objetivo é inspecionar o produto e adaptar o *product backlog* se necessário. Nesta reunião, o Time de Scrum e as partes interessadas colaboram sobre o que foi feito na *sprint* [Sch13]. Com base nisso, todos os presentes conhecem claramente o que está acontecendo e tem a oportunidade de ajudar a orientar o desenvolvimento futuro para garantir que a solução mais apropriada para o negócio está sendo construída [Rub13]. À luz disso, é comum atualizar o *product backlog* como parte da reunião de revisão da *sprint* [Scr12].

A reunião de retrospectiva ocorre geralmente após o término da reunião de revisão da *sprint* e antes da reunião de planejamento da próxima *sprint* [Rub13] [Sch13] [Scr12]. Enquanto a reunião de revisão é um momento para inspecionar e adaptar o produto, a reunião de retrospectiva é uma oportunidade de inspecionar e adaptar o processo [Rub13]. O Time de Scrum identifica o que foi bem e o que não foi tão bem, e identifica melhorias em potencial [Gom13] [Scr12]. Eles criam um plano de melhorias a serem realizadas a partir do próximo *sprint* [Rub13] [Scr12]. A implementação destas melhorias é a forma de adaptação à inspeção que o Time de Scrum faz a si próprio [Sch13]. Após a conclusão da reunião de revisão da *sprint*, todo o ciclo se repete novamente, iniciando com a próxima sessão de planejamento da *sprint*, realizada para determinar o conjunto de itens de trabalho com mais alto valor para o Time de Scrum se concentrar [Rub13].

O Scrum reforça a ideia de que todas as reuniões precisam de um tempo definido que precisam ser respeitados [Gom13]. Os tempos sugeridos para uma *sprint* de quatro semanas, por exemplo, são [Sch12]:

- Reunião de planejamento: 8 horas.
- Reunião diária: 15 minutos.
- Reunião de revisão (demonstração): 4 horas.
- Reunião de retrospectiva: 3 horas.

3.2.6 Kanban

No final da década de 1940, a Toyota encontrou uma melhor maneira de gerenciar seu processo de produção através de uma fonte improvável, supermercados norte-americanos [Ohn97]. Taiichi Ohno, chefe de produção da Toyota, desenvolveu o que veio a ser conhecido como Sistema Toyota de Produção [Ohn97]. Ele estudou o sistema de produção da Ford e ampliou sua visão sobre o modo como os supermercados norte-americanos controlavam seus estoques [PoP11]. Para isso, ele acrescentou seu conhecimento de fiação e tecelagem, bem como, as percepções dos trabalhadores por ele supervisionados [PoP11]. Levou anos de experimentação para se desenvolver gradualmente o Sistema Toyota de Produção, um processo que Ohno considerou infindável [Ohn97]. Ele explica que a base do Sistema Toyata de Produção é a absoluta eliminação do desperdício sendo sustentado por dois pilares: *Just-in-time* e a autonomação, ou automação com um toque humano (também chamada de *Jidoka*). *Just-in-time* significa que, em um processo de fluxo, o produto ou matéria-prima chega ao local de utilização no momento e na quantidade exata em que são necessários [Ohn97]. Isto significa que nada deve ser produzido, transportado ou comprado antes da hora exata. Autonomação, ou *Jidoka* em japonês, significa que o trabalho é organizado de modo a ser imediatamente interrompido quando a menor anomalia for detectada, sendo necessária a resolução da causa do problema antes do recomeço de suas atividades [PoP11].

O Kanban é a ferramenta utilizada para operar o sistema, ideia inspirada no modelo de trabalho dos supermercados norte-americanos [Ohn97]. Kanban é uma palavra japonesa que significa “cartão” ou “sinal” visual. A linha de produção da Toyota utilizou cartões Kanban para sinalizar as etapas de seu processo de fabricação [Ohn97]. O resultado da utilização do sistema *Just-in-time*, juntamente, com a programação manual via cartões Kanban na Toyota foi um sucesso imediato e esmagador, em grande parte porque seus detalhes haviam sido projetados pelos próprios trabalhadores, que sabiam como resolver os pequenos problemas e melhorar continuamente o processo de trabalho [PoP11]. Além disso, a natureza visual do sistema Kanban melhorou a comunicação das equipes acerca do que precisava ser feito e quando, assim como, padronizou e aperfeiçoou os processos existentes, o que ajudou a reduzir o desperdício, revelar os problemas existentes e maximizar o valor do trabalho entregue [Lea13]. Após dez anos de trabalho, em 1962 o Kanban foi adotado em toda *Toyota Motor Companhy*. Embora pareça muito tempo, de acordo com Taiichi Ohno, é natural porque estava sendo introduzido conceitos completamente novos [Ohn97].

Uma nova aplicação de Kanban para o trabalho do conhecimento emergiu em 2005 e foi formalizada como metodologia de melhoria de processo incremental e evolutiva por David J. Anderson em 2007, ao pensar sobre uma maneira sistemática de obter um ritmo sustentável de trabalho e encontrar uma abordagem para introduzir mudanças de processo com o mínimo de resistência [And10]. O método Kanban como uma abordagem para mudança começou a crescer em adoção pela comunidade após as conferências "*Lean New Product Development*" e "*Agile"* realizadas em 2007, onde Rick Garber e David J. Anderson apresentaram os resultados da utilização de Kanban na Corbis, uma empresa de acervo fotográfico e de direitos de propriedade intelectual, com base no centro de Seattle EUA. Segundo o relatório "*9th ANNUAL State of Agile Development Survey*", em 2014 Kanban foi adotado em 5% dos casos estudados, o que significa um aumento de 1% ao ano [Ver15]. O corpo de conhecimento em torno de Kanban tem sido influenciado pelo Sistema Toyota de Produção, assim como, pelo trabalho realizado pelos pioneiros de Kanban e outros líderes do pensamento [Lea13].

Kanban, mais precisamente Kanban aplicado ao desenvolvimento de software, é uma abordagem para mudança gerencial [And10] [Boe10]. Ele não é um processo ou ciclo de vida de gerenciamento de projetos ou de desenvolvimento de software, como Scrum e XP. Pelo contrário, ele requer que algum processo já esteja em funcionamento de maneira que Kanban possa ser aplicado para alterar incrementalmente o processo base [And10]. Isto significa que Kanban é uma abordagem para introduzir mudanças em um ciclo de desenvolvimento de software ou metodologia de gerenciamento de projetos [Kni09].

O método Kanban não prescreve um conjunto específico de passos ou de atividades que deve ser seguido durante o processo de desenvolvimento. A adoção de um sistema de mudança baseado no Kanban começa com o mapeamento e a visualização do fluxo de trabalho atual, para posterior estímulo à melhoria contínua [Kni09]. O aperfeiçoamento do sistema é obtido através da busca constante de melhorias incrementais e evolucionárias no processo existente [Boe10]. Mudanças revolucionárias e drásticas não são bem vindas, pois apesar de parecerem mais efetivas, elas têm probabilidade de erro maior porque trabalhadores tendem a resistir [TCU13b]. A ideia é mudar o mínimo possível e estimular melhorias graduais no fluxo de trabalho a fim de se alcançar mudanças evolucionárias no processo existente pelo aperfeiçoamento [And10]. Respeitar o fluxo de trabalho atual, os papéis, as responsabilidades, nome dos cargos e práticas de trabalho específicas contribui para neutralizar as resistências iniciais de adoção de uma nova forma de trabalho, e a iniciativa de implantação do Kanban tem maior chance de aceitação, por consenso, entre os trabalhadores [And10].

3.2.6.1 Os Princípios do Kanban

Por natureza, Kanban não é prescritivo [Lea13]. Porém, por vários anos, equipes de desenvolvimento de software seguiram orientações prescritivas e muitas vezes são confundidas pela liberdade proposta pelo Kanban, e em alguns casos tem encontrado dificuldades para começar a utilizá-lo. Nesse sentido, sugestões têm sido propostas e publicadas para fornecer essa ajuda, incluindo "*Kanban: successful evolutionary change for your technology business"* [And10], "Kanban em 10 passos: otimizando o fluxo de trabalho em sistemas de entrega de software" [Boe10] e "*Kanban Roadmap: How to Get Started in 5 Steps*" [Lea13]. O LeanKit [Lea13] organizou os elementos fundamentais de Kanban em quatro princípios, listados a seguir.

3.2.6.1.1 Visualizar o Trabalho

Ao criar um modelo visual do fluxo de trabalho atual, é possível observar a movimentação de todo o trabalho que está sendo realizada através do sistema Kanban, como também, tudo o que impacta o desempenho da organização em termos de quantidade de trabalho de valor entregue e o tempo de ciclo necessário para entregá-lo, incluindo os gargalos, filas, variabilidade, desperdícios e impedimentos [And10]. Todas as atividades necessárias para se completar um item de trabalho devem ser visualizadas, e não apenas o desenvolvimento [Boe10]. Tornar o trabalho visível tem mudado o comportamento e incentivado uma maior comunicação e colaboração no trabalho [And10] [Lea13].

3.2.6.1.2 Limitar o Trabalho em Progresso

Pode parecer contra-intuitivo, mas limitar a quantidade de trabalho em progresso pode reduzir o tempo que um item de trabalho percorre no sistema Kanban, ou seja, um item de trabalho pode ser entregue mais rápido ao limitar o trabalho em progresso [Lea13], bem como, contribuir para definir a capacidade e equilibrar a demanda sobre a equipe em relação ao rendimento do trabalho entregue, evitando que trabalhadores fiquem sobrecarregados [And10]. Outrossim, ao limitar o trabalho em progresso tudo o que fica bloqueado por qualquer motivo tende a parar o sistema. Isso cria a necessidade de concentrar toda a equipe na solução do problema para desbloquear o item e restaurar o fluxo [And10]. Com isso, ao limitar o trabalho em progresso, um sistema puxado é estabelecido, porque o novo trabalho é puxado para o sistema quando existe capacidade para lidar com ele, em vez de ser empurrado para o sistema com base na demanda. De acordo com David J. Anderson, os limites para o trabalho em progresso devem ser acordados por consenso com os *stakeholders* em todos os níveis e gerência sênior [And10].

3.2.6.1.3 Medir e Gerenciar o Fluxo de Trabalho

Ao utilizar limites para o trabalho em progresso e ao tornar explícitas as políticas do processo que estão sendo seguidas é possível aperfeiçoar o fluxo de trabalho e obter métricas e indicadores a partir da análise do sistema Kanban [Lea13]. Como resultado, um fluxo aperfeiçoado ajuda a reduzir os prazos de entrega e melhora o desempenho da data de entrega e previsibilidade, estabelecendo uma cadência regular de entrega de software [And10]. Por causa disso, Kanban precisa reportar métricas um pouco diferente de outras abordagens ágeis. Ele define uma expectativa de gerenciamento quantitativa objetiva e orientada a dados mais do que um gerenciamento qualitativo subjetivo que é mais utilizado em reuniões de revisões de *sprints* [And10]. David J. Anderson [And10] e Jesper Boeg [Boe10] sugerem começar com as seguintes métricas ou indicadores:

- **Diagrama de Fluxo Cumulativo:** mostra a quantidade de trabalho em progresso em cada estágio do sistema e ajuda a identificar se o sistema Kanban está fluindo corretamente.
- ***Lead Time*:** é um indicador da agilidade do negócio e informa o quanto previsível o projeto é na entrega, sendo útil como um relatório de desempenho geral do sistema, mas não é muito útil como um indicador de previsibilidade ou como um meio para informar oportunidades de melhoria.
- **Desempenho de Entrega na Data:** é um indicador de previsibilidade e mostra se a entrega dos itens de trabalho tem ocorrido antes da data limite, sendo aplicável somente para itens de trabalho com data fixa de entrega.
- **Rendimento:** indica quantos itens de trabalho foram finalizados num período de tempo determinado, ele não deve ser usado para prever a quantidade de entregas em um intervalo de tempo ou qualquer compromisso específico de entrega, pelo contrário, deve ser usado como um indicador de quão bem o sistema está executando e para demonstrar melhoria contínua.
- **Problemas e Itens de Trabalho Bloqueados:** mostra um Diagrama de Fluxo Cumulativo de impedimentos reportados sobreposto a um gráfico de quantidade de trabalho em progresso, dando a indicação de quão boa é a organização em identificar, reportar, resolver problemas e os seus impactos.
- **Qualidade inicial:** mostra o número de defeitos descobertos pelos testadores dentro do sistema e indica quanta capacidade está sendo gasta devido à baixa qualidade inicial.

- **Carga de Falhas:** mostra quantos itens de trabalho têm sido executados devido à baixa qualidade de itens anteriores, ou seja, quantos itens de trabalho são defeitos de produção ou novas *features* que foram requisitadas devido à usabilidade fraca ou falhas em antecipar necessidades do usuário adequadamente.

3.2.6.1.4 Melhorar Continuamente

Uma vez que o sistema Kanban está estabelecido, ele torna-se uma via de acesso para a melhoria contínua [Lea13]. A grande distribuição de informações com a visualização do fluxo de trabalho e de políticas explícitas, proporciona à equipe uma capacidade de dialogar constantemente sobre oportunidades de melhoria [Boe10]. Ao se reunirem para discuti-las, o fluxo de trabalho pode mudar e, com ele, o mapa visual que o representa. Quando o mapa muda, uma nova forma de ver o trabalho se evidencia, e novas oportunidades de melhoria emergem, fazendo com que um ciclo evolucionário de mudanças se estabeleça [Val14]. Equipes querem demonstrar como elas podem ajudar a organização com uma melhor previsibilidade, mais rendimento, *lead time* menores, custos menores, e maior qualidade [And10]. Como são coletados dados reais, experimentos e análises podem ser executados para mudar o sistema e melhorar a eficiência da equipe [Boe10]. Um experimento, neste caso, não é muito diferente de uma experiência científica básica. Avaliar o estado atual, formular uma hipótese, testar a hipótese com um experimento, medir o resultado e, em seguida, chegar a uma conclusão [Lea13].

Como visto, o Kanban é baseado numa ideia muito simples que consiste em visualizar o trabalho, limitar o trabalho em progresso, medir a eficiência da equipe e do fluxo de trabalho, e por último, começar a melhorar continuamente o processo. O fluxo contínuo do sistema Kanban está menos interessado em reportar se um projeto está dentro do prazo ou se um plano específico está sendo seguido. O mais importante é mostrar que o sistema Kanban é previsível e está funcionando como projetado, que a organização tem agilidade no negócio, que há um foco no fluxo, e que existe um desenvolvimento claro de melhoria contínua [And10]. Segundo David J. Anderson, o Kanban parece uma mudança pequena, porém, muda tudo a respeito de uma empresa, incluindo o desempenho, cultura, capacidade e maturidade de uma equipe e a organização a qual ela está inserida [Kni09].

3.2.7 Alguns Desafios dos Métodos Ágeis

O sucesso dos métodos ágeis favorece o interesse em usá-los em diferentes cenários do desenvolvimento de software. Porém, a imagem bastante otimista no nível teórico se opõe a uma realidade dominada por desafios, dificuldades e problemas concretos.

3.2.7.1 Desafios Gerais

Segundo Sommerville [Som11], na prática, os princípios básicos dos métodos ágeis são, em algumas situações, difíceis de se concretizar:

- Embora a ideia de envolvimento do cliente no processo de desenvolvimento de software seja atraente, seu sucesso depende de um cliente disposto e disponível para passar o tempo necessário com a equipe de desenvolvimento, e que possa representar todas as partes interessadas do software, o que nem sempre é possível.
- Membros individuais da equipe podem não ter personalidade e disponibilidade adequada para o intenso envolvimento que é típico dos métodos ágeis e, portanto, não interagem bem com os outros membros da equipe.
- Priorizar as mudanças pode ser extremamente difícil, especialmente em sistemas nos quais muitas partes são afetadas. Normalmente, diferentes partes interessadas dão prioridades diferentes para mudanças diferentes.
- Manter a simplicidade pode ser difícil. Sob a pressão de prazos de entrega, membros da equipe podem não ter tempo suficiente para fazer as simplificações desejadas.
- Muitas organizações, principalmente as grandes empresas, passaram anos mudando sua cultura para que os processos fossem definidos e seguidos, sendo difícil mudar para um modelo de trabalho em que os processos são informais e definidos pela equipe de desenvolvimento.
- Os métodos ágeis são mais eficazes quando o sistema pode ser desenvolvido com uma pequena equipe colocalizada capaz de se comunicar de maneira informal. Isso pode não ser possível para sistemas de grande porte que exigem equipes de desenvolvimento maiores, podendo estas estarem distribuídas geograficamente, inclusive com a presença de empresas externas. Nesse caso, abordagens prescritivas podem ser combinadas com métodos ágeis e

documentos de projeto para a comunicação entre as equipes podem ser necessários.

- Outro problema não técnico, que é um problema típico do desenvolvimento e da entrega incremental, ocorre quando o cliente usa uma empresa externa para implementar o software. O documento de requisitos do software é normalmente parte integrante do contrato entre o cliente e a empresa externa. Como o desenvolvimento incremental é inerente aos métodos ágeis, escrever contratos para esse tipo de desenvolvimento pode ser difícil. Consequentemente, os métodos ágeis têm de contar com contratos nos quais os clientes pagam pelo tempo e recursos necessários para desenvolver o software, e não pelo conjunto de requisitos entregues.
- A literatura existente tem relatado principalmente a experiência do uso de métodos ágeis para o desenvolvimento de novos softwares. Porém, pouco ou nenhum estudo tem sido dedicado ao uso de métodos ágeis em projetos de manutenção, onde a falta de documentação adequada pode tornar a manutenção do software mais cara e difícil. Igualmente, em projetos de manutenção, pode ser difícil justificar o envolvimento de representantes dos clientes em tempo integral porque as mudanças não são contínuas. Representantes dos clientes são propensos a perder o interesse no sistema na fase de manutenção.

Na mesma linha, a publicação de Dingsøy *et al.* [DDM10] apontou três desafios para métodos ágeis:

1. Projetos de missão crítica demandam níveis mais alto de desempenho e segurança, consequentemente, arquiteturas mais robustas são exigidas e equipes maiores são formadas [BLK+10]. Conciliar a visão de aceitar e responder rapidamente a mudanças dos métodos ágeis com as questões de arquitetura em projetos de missão crítica, tem sido uma questão importante para manter o software viável a longo prazo, cujas modificações acontecem no curto prazo [BLK+10].
2. Usabilidade de software tem sido um desafio para qualquer abordagem de desenvolvimento [DDM10]. Como integrar métodos ágeis com a experiência do usuário durante o desenvolvimento do software tem sido uma questão latente para melhorar a usabilidade e diminuir o número de modificações no sistema por conta de defeitos de usabilidade [HHM10].
3. A cultura organizacional tem sido um fator que afeta potencialmente a adoção de métodos ágeis [DDM10]. Muitos dos benefícios dos métodos ágeis podem ser

> negados quando a cultura organizacional não tem evoluído com o mesmo entusiasmo dos métodos ágeis. Métodos ágeis são incompatíveis com estruturas hierárquicas e burocráticas [IiI10]. Além disso, supondo que métodos ágeis são suportados por organizações com uma forte cultura hierárquica, é de se esperar que eles são complementados com características prescritivas e, como consequência, eles podem começar a perder um pouco de sua agilidade [IiI10].

3.2.7.2 Desafios com o Desenvolvimento Distribuído de Software

O desenvolvimento ágil envolve trazer as partes interessadas para dentro do processo de desenvolvimento de software, o que exige mais colaboração e *feedback* rápido entre as pessoas envolvidas no projeto. Porém, em projetos de desenvolvimento de software dispersos geograficamente, a distância tem afetado a capacidade de comunicação e colaboração entre as pessoas [Her07]. Além disso, a distância física elimina a oportunidade da comunicação frente a frente, o qual é um dos aspectos mais importantes para que o desenvolvimento ágil aconteça com resultados positivos [Bec99].

Em uma equipe global de desenvolvimento de software, um membro da equipe distribuída geograficamente deve ser capaz de realizar as atividades típicas de desenvolvimento de software, incluindo codificação, testes, implantação e liberação para a produção durante o seu horário de trabalho (de acordo com o fuso horário local). Porém, o desenvolvimento distribuído de software possui várias preocupações, incluindo dispersão geográfica (distância física), dispersão temporal (diferenças de fuso horário) e diferenças socioculturais (idioma, tradições, costumes, normas e comportamento) [Car99]. Essas três características (não necessariamente as três ao mesmo tempo) acabam criando diferentes sensações de distância, que por sua vez multiplicam em diversas dificuldades na coordenação do trabalho necessário para desenvolver produtos de software [AuP07], e se não forem tratadas adequadamente podem desestabilizar o sucesso do desenvolvimento global de software [HeM03]. Além disso, itens de trabalho executados de maneira distribuída parecem levar, cerca de duas vezes e meia, mais tempo para serem concluídos do que itens de trabalho semelhantes executados em ambiente colocalizado [CiC10]. Desta maneira, melhorar a comunicação, colaboração e os resultados em ambientes distribuídos de desenvolvimento de software têm sido um desafio para métodos ágeis [Kar13].

Nesse contexto, estudos sobre a combinação do desenvolvimento ágil com a capacidade da computação em nuvem (incluindo ferramentas de suporte ao desenvolvimento distribuído de software) têm emergido como solução para tornar o desenvolvimento distribuído de software tão eficiente quanto o desenvolvimento

colocalizado de software, preservando os conceitos de agilidade preconizados no manifesto ágil [Kar13]. Além disso, sistemas que demandam várias implantações diárias no ambiente de produção têm exigido maior automação e flexibilidade para gerenciar e conduzir esse processo de entregas frequentes. A computação em nuvem pode conciliar o desenvolvimento ágil com operações de TI para produzir e entregar software seguro de bom desempenho com rapidez [Kar13]. Por último, profissionais e pesquisadores em TI são da opinião de que ela é importante não apenas para os tecnólogos, mas também para as empresas, pois proporciona uma oportunidade de melhoria e aceleração na colaboração entre as equipes e as partes interessadas [Kar13], o que é particularmente relevante para a adoção bem-sucedida de métodos ágeis.

3.2.7.3 Desafios com Contratos Ágeis

Se por um lado métodos prescritivos de desenvolvimento de software têm um longo histórico de confiabilidade jurídica, por outro lado, faltam maiores informações de como empresas e governo podem usar métodos ágeis “legalmente” [Ste13]. Nesse contexto, o estudo de Wernham [Wer12] revelou que a compreensão da dinâmica jurídica de modelos de contratos por profissionais que atuam com métodos ágeis é limitada. Novos modelos de contratos precisam ser construídos para lidar com as deficiências dos modelos de contratos tradicionais, o que exige uma compreensão muito maior da teoria da complexidade, teoria do caos e do pensamento sistêmico para atender a complexidade dinâmica dos ambientes atuais. Encontrar abordagens adequadas para unir empresas contratadas e contratantes tem sido um desafio. Mais pesquisas são necessárias para definir modelos de contratos que podem apoiar adequadamente o desenvolvimento ágil de soluções.

3.2.7.4 Desafios com Dívidas Técnicas

Estudos evidenciam que métodos ágeis têm proporcionado ganhos significativos em projetos de desenvolvimento de software [Ric08]. Por outro lado, desenvolvedores pressionados por entregas rápidas de soluções, muitas vezes ignoram a importância da qualidade do código e dívidas acabam sendo inseridas no software [Cri14]. Essas dívidas vêm sendo estudadas por Ward Cunnigham, que criou a metáfora da “Dívida Técnica” - em alusão à dívida financeira - justamente com o objetivo de melhorar a qualidade dos entregáveis de software [Cun92]. Visto que a partir do momento que dívidas são inseridas, juros começam a ser contraídos, e em algum momento futuro precisarão ser pagos. Dessa maneira, deixá-las visíveis e gerenciá-las, bem como, escolher o melhor momento para pagá-las ou não (se for o caso) são desafios atuais para o desenvolvimento ágil de software.

3.2.8 Algumas Vantagens e Desvantagens dos Métodos Ágeis

Apesar do discurso, aparentemente neutro, do uso de métodos ágeis como abordagem ideal para alcançar o desenvolvimento adaptativo e flexível, a agilidade não conta apenas com seguidores, mas também com críticos. Críticos mais fundamentados, como Boehm [Boe02], destacam as vantagens e desvantagens dos métodos ágeis. Eles propõem uma abordagem híbrida, na qual métodos ágeis incorporam algumas técnicas de abordagens prescritivas, que pode ser o melhor caminho a seguir [Pre11]. O estudo de Boehm [Boe02] identificou algumas vantagens e desvantagens dos métodos ágeis em comparação com abordagens prescritivas, apresentadas a seguir.

- **Desenvolvedores:** Cockburn e Highsmith [CoH01] enfatizam várias qualidades pessoais dos desenvolvedores necessárias para os métodos ágeis, incluindo colaboração, talento, habilidade e comunicação, onde a agilidade no desenvolvimento é alcançada através do conhecimento tácito incorporado na equipe ao invés do conhecimento explícito registrado em documentos de planejamento. Segundo Boehm [Boe02], quando o conhecimento tácito da equipe é suficiente para as necessidades do ciclo de vida do produto de software, as coisas funcionam bem. Por outro lado, existe o risco de que a equipe poderá cometer erros arquiteturais irreparáveis por conta de deficiências não reconhecidas em seu conhecimento tácito. Abordagens prescritivas podem reduzir esse risco e facilitar a avaliação de especialistas externos. Ao fazê-lo, no entanto, métodos ágeis aceitam o risco de que uma mudança rápida poderá tornar os documentos obsoletos, ou então, custosos para serem atualizados.
- **Clientes:** Métodos ágeis funcionam melhor quando os clientes atuam em estreita colaboração e dedicação com a equipe de desenvolvimento, bem como, quando o conhecimento tácito é suficiente para as necessidades do ciclo de vida do produto de software [Bec99]. Igualmente, abordagens prescritivas podem reduzir o risco de deficiências no uso do conhecimento tácito, assim como, pode compensar a falta de clientes reais no local de desenvolvimento [Boe02].
- **Requisitos:** Cockburn e Highsmith [CoH01] afirmam que métodos ágeis são mais adequados para ambientes complexos onde os requisitos são voláteis. De acordo com sua visão de mundo, as organizações são sistemas adaptativos complexos em que os requisitos são emergentes em vez de previsíveis. Por outro lado, as abordagens prescritivas são recomendadas para situações onde os requisitos são estáveis com pouca ou nenhuma variação [Boe02]. Nessas situações, a ênfase conservadora em ter os requisitos completos, consistentes,

precisos, verificáveis e rastreáveis dificilmente encontrará problemas intransponíveis de mudança [Boe02].

- **Arquitetura:** O manifesto ágil valoriza software em funcionamento mais que documentação abrangente, bem como, enfatiza a simplicidade: maximizar a quantidade de trabalho que não precisou ser feito [BBB+01]. Segundo Boehm [Boe02], esta abordagem funciona bem quando as necessidades futuras são imprevisíveis. Entretanto, em situações em que os requisitos futuros são previsíveis, esta prática desperdiça e não considera o apoio fundamental que o planejamento da arquitetura do sistema tem sobre a evolução do software. Alguns métodos ágeis como Crystal e DSDM reconhecem a importância do planejamento antecipado da arquitetura do sistema para acomodar mudanças de requisitos. Porém, tal como acontece com os requisitos, uma documentação abrangente da arquitetura do sistema encontrará problemas de atualização com mudanças frequentes de requisitos.
- **Refatoração:** De acordo com Boehm [Boe02], com desenvolvedores experientes e softwares pequenos a refatoração livre é válida. Porém, o estudo de Boehm [Boe02] mostrou que em sistemas grandes, 20% dos problemas que causam 80% dos retrabalhos são provenientes de atividades de refatoração, incluindo otimização de desempenho, tolerância a falhas e segurança, e que nenhuma refatoração foi capaz de trazer benefícios reais para o funcionamento do sistema.
- **Tamanho:** Cockburn e Highsmith [CoH01] concluíram que métodos ágeis são mais difíceis para equipes maiores, mas eles podem ser ampliados e aplicados a projetos de grande porte, ainda que com uma sobrecarga considerável [Coh11]. Por outro lado, abordagens prescritivas lidam melhor com projetos de grande porte, mas em organizações burocráticas, o uso de métodos orientado a planos tem sido ineficiente para projetos pequenos [Boe02].
- **Objetivo principal:** O primeiro princípio do manifesto ágil afirma que “A maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor” [BBB+01]. Segundo Boehm [Boe02], “entrega adiantada e contínua” são metas razoavelmente compatíveis para pequenos sistemas desenvolvidos por pessoas experientes. Porém, esse princípio quando aplicado em sistemas de grande porte, os primeiros resultados podem levar a retrabalhos de arquitetura, sendo necessário mais planejamento.

De acordo com Boehm & Turner [BoT03], métodos ágeis e abordagens prescritivas não podem garantir sozinhos o êxito que os projetos de software esperam. O

desenvolvimento de software precisa tanto de agilidade como de disciplina. Segundo Sommerville [Som11], a maioria dos projetos inclui práticas de abordagens prescritivas e ágil, sendo que o equilíbrio entre as duas abordagens pode ser obtido respondendo a uma série de questões técnicas, humanas e organizacionais:

1. É importante ter uma especificação de requisitos e um projeto bem detalhados antes de passar para a implementação? Se sim, uma abordagem prescritiva provavelmente será necessária.
2. É realista uma estratégia de entrega incremental em que o software é entregue aos clientes e rapidamente se obtém um *feedback*? Se sim, o uso de métodos ágeis deve ser considerado.
3. Quão grande é o sistema que está em desenvolvimento? Os métodos ágeis são mais eficazes quando o sistema pode ser desenvolvido por uma equipe colocalizada capaz de se comunicar de maneira informal. Isso pode não ser possível para sistemas de grande porte que exigem equipes de desenvolvimento maiores. Nesse caso, uma abordagem prescritiva pode ter de ser usada.
4. Que tipo de sistema está sendo desenvolvido? Sistemas que exigem uma análise profunda antes da implementação geralmente demandam um projeto bastante detalhado para atender a essa análise. Nessas circunstâncias, uma abordagem prescritiva pode ser a melhor opção.
5. Qual é o tempo de vida esperado do sistema? Sistemas de vida longa podem exigir mais da documentação de projeto, a fim de comunicar para a equipe de manutenção as intenções originais dos desenvolvedores do sistema. Porém, métodos ágeis argumentam que a documentação não é constantemente atualizada e não é de muita utilidade para a manutenção do sistema a longo prazo.
6. Que tecnologias estão disponíveis para apoiar o desenvolvimento do sistema? Métodos ágeis frequentemente contam com boas ferramentas para manter o controle de um projeto de desenvolvimento. A ausência de ferramentas de apoio ao desenvolvimento de software pode gerar necessidade de mais documentação do projeto.
7. Como é organizada a equipe de desenvolvimento? Se está distribuída, ou se parte do desenvolvimento está sendo terceirizado, então pode ser necessário o desenvolvimento de documentos do projeto para a comunicação entre as equipes.

8. Existem questões culturais que podem afetar o desenvolvimento do sistema? Organizações tradicionais de ES têm uma cultura de desenvolvimento baseada em planos. Geralmente, isso requer extensa documentação de projeto, no lugar do conhecimento informal, usado em projetos ágeis.
9. Quão bons são os projetistas e programadores na equipe de desenvolvimento? Às vezes, argumenta-se que os métodos ágeis exigem níveis mais altos de habilidade do que as abordagens dirigidas a planos, em que os programadores simplesmente traduzem um projeto detalhado em um código.
10. O sistema é sujeito à regulamentação externa? Se sim, então, provavelmente será obrigatória a produção de documentação detalhada como parte da documentação do sistema.

Cockburn [Coc02] argumenta que abordagens prescritivas, tem uma falha essencial: ela esquece das fragilidades das pessoas que desenvolvem o software. As pessoas não são robôs. As pessoas apresentam grande variação nos estilos de trabalho; diferenças significativas no nível de habilidade, criatividade, organização, consistência e espontaneidade. Algumas se comunicam bem na forma escrita, outros não. Cockburn afirma que os modelos de processos podem “lidar com as fraquezas comuns das pessoas com disciplina e/ou tolerância” e que a maioria dos modelos de processos prescritivos opta por disciplina. Segundo ele: “Como a consistência nas ações é uma fraqueza humana, as metodologias com disciplina elevada são frágeis”. Para que funcionem, os modelos de processos devem fornecer um mecanismo realista que estimule a disciplina necessária ou, então, devem ter características que apresentem “tolerância” com as pessoas que realizam trabalhos de ES. De acordo com Pressman [Pre11], invariavelmente, práticas tolerantes são mais facilmente adotadas e sustentadas pelas pessoas envolvidas, porém deve-se considerar os prós e os contras.

Por último, Pressman [Pre11] adverte que como toda argumentação sobre tecnologia de software, o debate sobre metodologia corre o risco de “descambar para uma guerra santa”. Se for deflagrada uma guerra, a racionalidade desaparece e crenças, em vez de fatos, orientarão a tomada de decisão. Além disso, Pressman mostra que a questão de como desenvolver softwares para atender as necessidades atuais dos clientes e que apresente características de qualidade que permitirão que seja estendido e ampliado para responder às necessidades dos clientes no longo prazo, continua sendo uma questão latente sem respostas absolutas, mesmo nas abordagens ágeis.

3.3 Métodos Ágeis para Desenvolvimento de Software em OP

Esta seção apresenta evidências científicas que métodos ágeis também são viáveis para OP, sendo que eles têm despertado gradativamente o interesse da AP de diversos países, incluindo o Brasil, Estados Unidos da América (EUA) e Reino Unido.

3.3.1 Métodos Ágeis em Organizações Públicas: Recomendações de Governo

A história do esforço de implantar padrões e normas para estruturar o desenvolvimento de software em OP mostrou que não é possível mudar atitudes a favor do desenvolvimento adaptativo de software através de leis e regulamentos, sendo necessário pensar em novas abordagens para implementar o desenvolvimento adaptativo de software na AP [Wer12]. Além disso, essa história revelou inicialmente uma disputa pelo controle do desenvolvimento de software entre as OP (contratante) e as empresas contratadas da indústria de software [Mcd10]. Neste período, o governo foi o principal cliente da indústria e influenciou diretamente a maneira pela qual as empresas produziam softwares [Mcd10] [Mid99]. Entretanto, com o crescimento exponencial do mercado e falhas de projeto que ocasionaram o descrédito em padrões de governo, a indústria de software não estava mais disposta a implementar procedimentos caros e talvez tecnicamente falhos para satisfazer as necessidades de um cliente importante, mas não mais o único [Mcd10]. Uma das respostas da indústria de software para lidar com as limitações de projetos baseados em abordagens prescritivas de ES, típica de projetos de governo [Wer12], foi dada através do manifesto ágil. Desde então, cada vez mais, o governo tem renunciado as suas formas de trabalho e tem adotado métodos e práticas de desenvolvimento de software alinhado aos princípios do desenvolvimento ágil de software.

3.3.1.1 Recomendações de Governo: Estados Unidos da América e Reino Unido

Em 2012, órgãos fiscalizadores dos governos dos Estados Unidos da América (EUA) e do Reino Unido anunciaram grandes mudanças em suas abordagens para projetos de desenvolvimento de software financiados pelo governo. Tanto o *Government Accountability Office* (GAO) dos EUA quanto o *National Audit Office* (NAO) do Reino Unido, reconheceram valor em métodos ágeis e recomendaram seu uso para o desenvolvimento de software nos departamentos governamentais.

O relatório intitulado "*Effective Practices and Federal Challenges in Applying Agile Methods*" do GAO [GAO12] informou:

> *"As agências federais dependem da TI para realizar suas missões e gastaram mais de 70 bilhões de dólares em TI no ano fiscal de 2011. Entretanto, em vários exemplos de projetos de longo prazo, foram identificados estouros de orçamento e atrasos no cronograma, além de pouca contribuição para os objetivos propostos. De modo a reduzir esses riscos, o Ministério de Orçamento e Gestão recomenda a entrega modular de software, consistente com a abordagem conhecida como ágil, que prega o desenvolvimento de software em pequenos incrementos e intervalos curtos."*

O relatório norte-americano identificou padrões de governança eficazes e abordagens ágeis para projetos, os quais foram organizados em quatro princípios de governança:

- A governança deve espelhar a filosofia dos métodos ágeis - executar uma tarefa apenas se agregar valor ao negócio, sem introduzir atrasos;
- Equipes ágeis devem decidir a respeito de metas empíricas de desempenho e acompanhamento a serem utilizadas;
- A gerência sênior, assessores externos, usuários e a equipe de TI devem atuar em parceria para garantir a qualidade do projeto; essa abordagem colaborativa é uma mudança essencial na forma de pensar e agir;
- A avaliação externa e as revisões das entregas ágeis devem focar nos comportamentos das equipes, não apenas em processos e documentação.

Enquanto, o relatório "*Governance for Agile delivery*" do NAO [NAO12], de maneira similar, prescreveu:

> *"O Governo pretende utilizar métodos ágeis, para reduzir os riscos de fracasso dos projetos. Em audiência do Comitê de Prestação de Contas em maio de 2011, ficou claro que o governo não considera ágil apenas como método para melhorar o desenvolvimento de software, e sim como técnica para obter sucesso em mudanças organizacionais tecnológicas. Segundo um ex-secretário, "não existem projetos de TI; há projetos que envolvem TI". Tentar modificar todo o sistema nacional em uma implantação big-bang de um único dia estaria fadado ao fracasso em quase todas as situações."*

O relatório do Reino Unido, por sua vez, identificou desafios comuns para a adaptação e utilização da abordagem ágil no governo federal:

- Equipes tiveram dificuldades em colaboração e dificuldade em se adaptar ao trabalho auto-gerenciado;

- Procedimentos de licitação e outras formas de aquisição podem não se adaptar a projetos ágeis;
- Houve clientes que não confiavam em soluções iterativas;
- Equipes tiveram dificuldade em se comprometer com novos requisitos e suas alterações, e também em gerenciar requisitos iterativamente;
- Agências tiveram problemas para manter as equipes comprometidas;
- Revisões e auditorias se mostraram difíceis de executar dentro do período das iterações;
- A adoção de novas ferramentas foi difícil;
- As práticas de comunicação governamentais não se alinhavam com a abordagem ágil;
- Revisões tradicionais de artefatos conflitaram com o pensamento ágil;
- O monitoramento tradicional do progresso não se alinhava à abordagem ágil.

Além disso, o relatório do britânico identificou práticas e abordagens que se provaram bem-sucedidas nos projetos federais:

- Começar com diretrizes ágeis e uma estratégia de adoção de métodos ágeis;
- Reforçar a adoção dos conceitos ágeis utilizando seus termos, tais como histórias do usuário; além de fornecer exemplos ágeis, por exemplo demonstrar como escrever essas histórias;
- Melhorar continuamente a adoção de métodos ágeis tanto no nível de projetos quanto no nível organizacional;
- Buscar identificar e resolver impedimentos em nível organizacional e de projetos;
- Obter frequentemente *feedback* dos clientes e partes interessadas;
- Empoderar equipes e encorajar a formação de equipes pequenas e interdisciplinares;
- Incluir requisitos relacionados à segurança e ao monitoramento do progresso na fila de trabalho não concluído (o *backlog*);
- Conquistar confiança por meio da demonstração de valor ao final de cada iteração;
- Monitorar o progresso utilizando ferramentas e métricas, diariamente e de forma visual.

Historicamente, as recomendações recentes para o uso de métodos ágeis nos governos dos EUA e Reino Unido não surgiram da noite para o dia. Pelo contrário, foram fundamentadas em resultados de pesquisas realizadas no governo ao longo de vários anos.

O GAO, por exemplo, em 2009, constatou que o DoD, em alguns casos, utilizou mais tempo para especificar detalhes em seus projetos do que entregar soluções de software em tempo oportuno, o que aumentou consideravelmente o tempo de planejamento dos projetos extrapolando os custos ao longo de sua execução, com pouca contribuição para os objetivos propostos [GAO09]. Mais tarde, em 2011, ao estudar os fatores críticos de sucesso em sete projetos de TI, com um custo total em torno de 5 bilhões de dólares, o GAO descobriu que todos os sete projetos, de uma forma ou de outra, aplicaram o desenvolvimento incremental de software, o que demonstrou a importância do desenvolvimento adaptativo como um fator crítico de sucesso para os projetos de governo [Wer12]. Com base nisso, é que o GAO recomendou o uso de abordagens alinhadas ao desenvolvimento ágil para os departamentos governamentais dos EUA.

Da mesma maneira, o NAO em 2006, publicou o relatório "*Delivering successful IT-enabled business change*" [NAO06] advertindo sobre o risco da abordagem *big-bang*, com exemplos de projetos de sucesso no governo que tinham evitado essa abordagem. No mesmo ano, o NAO publicou um parecer em que OP poderiam se beneficiar de abordagens evolutivas, não apenas TI. Finalmente, em 2012, o NAO recomendou o uso de métodos ágeis nos departamentos governamentais do Reino Unido [NAO12].

Em 2011, no Reino Unido, o *HM Treasury* e o *Cabinet Office* [HMT11] publicaram orientações para estruturar o desenvolvimento de software em grandes projetos de governo com base no Modelo Cascata, contradizendo e negligenciando o conselho do NAO a favor do desenvolvimento incremental. Porém, foram necessários mais três anos, para que o *HM Treasury* reconhecesse valor em métodos ágeis para o desenvolvimento de software na AP, e juntamente com o *Government Digital Service* (GDS) e em comum acordo com o *Cabinet Office*, publicassem alguns esclarecimentos sobre como organizações governamentais conseguirão aprovação financeira para executar projetos de desenvolvimento de software [HMT14]. O objetivo da ação é reduzir a burocracia e incentivar a inovação para tornar mais fácil a aplicação de TI em todo o governo, apoiando uma cultura ágil em toda AP. O esclarecimento aplica-se a grandes projetos de governo [HMT14]; mas, é uma boa prática para organizações governamentais que seguem os mesmos princípios internamente, mesmo quando se trata de gastos menores [GDS14a]. Ele recomenda o uso de artefatos ágeis, como gráfico de *burns* e *product backlog* para acompanhar o progresso do projeto em vez de documentos convencionais de TI, muitas vezes, detalhados e custosos para atualização. Ele também orienta que o foco deve permanecer sobre as necessidades dos usuários e os resultados de negócio, sobre o acompanhamento dos custos e as metas associadas ao projeto.

3.3.1.2 Recomendações de Governo: Brasil

No Brasil, em agosto de 2013, o Tribunal de Contas da União (TCU) emitiu na forma de acórdão o resultado de uma auditoria acerca do uso de métodos ágeis em contratações para desenvolvimento de software pela AP Federal [TCU13b]. Com base em uma análise empreendida em oito OP federais, o estudo demonstrou a viabilidade da adoção de métodos ágeis em contratações destinadas ao desenvolvimento de software pela AP Federal. Por outro lado, foram relacionados alguns riscos inerentes passíveis de materialização nas contratações com métodos ágeis, organizados em três categorias de riscos: relativos a processo, pessoas e produto. A Tabela 4 apresenta os riscos identificados pelo TCU.

Tabela 4 – Riscos relacionados a contratações com métodos ágeis [TCU13b].

DIMENSÃO	RISCO
RISCOS RELATIVOS A PROCESSO	Contratação de desenvolvimento de software com adaptação de metodologia ágil que desvirtue sua essência.
	Alteração da metodologia ágil adotada no instrumento convocatório no decorrer da execução contratual.
	Ausência de definição dos artefatos ou alteração dos artefatos exigidos da contratada no instrumento convocatório durante a execução contratual.
	Exigência de artefatos desnecessários ou que se tornam obsoletos rapidamente.
	Utilização de contrato para desenvolvimento de software por metodologias tradicionais para desenvolvimento por métodos ágeis.
RISCOS RELATIVOS A PESSOAS	Falta de comprometimento ou colaboração insatisfatória do responsável indicado pela área de negócios (Product Owner) no desenvolvimento do software.
	Falta do conhecimento necessário do indicado pela área de negócios (Product Owner) para o desenvolvimento do software.
	Excessiva dependência da visão do indicado pela área de negócios (Product Owner).
	Equipe da empresa contratada não ter expertise em desenvolvimento de software com métodos ágeis.
	Dificuldade de comunicação entre a equipe de desenvolvimento da contratada com o indicado pela área de negócios (Product Owner).
RISCOS RELATIVOS A PRODUTO	Alteração constante da lista de funcionalidades do produto.
	Iniciação de novo ciclo sem que os produtos construídos na etapa anterior tenham sido validados.
	Falta de planejamento adequado do software a ser construído.
	Pagamento pelas mesmas funcionalidades do software mais de uma vez.
	Não disponibilização do software em ambiente de produção para a utilização e avaliação dos reais usuários.
	Forma de pagamento não baseada em resultados.

De acordo com o relatório, no caso específico de adoção de métodos ágeis, tratados como novidade no mercado especializado nacional, sobretudo no âmbito da AP Federal, a gestão de riscos inerentes às características do método merece atenção especial, no sentido de possibilitar que as OP possam fazer uso das práticas previstas sem incorrer em descumprimento dos normativos vigentes. Para tanto, o relatório apontou algumas recomendações na busca por mitigar os riscos identificados. Frisando-se que vários riscos não são exclusivos do uso de métodos ágeis, mas também se aplicam a contratações de desenvolvimento com outros tipos de metodologias [TCU13b]:

- Entrega de artefatos de documentação associados ao software produzido a cada iteração;
- Relação contratual prevalece sobre a possível colaboração entre as partes;
- O escopo das iterações é fixo;
- Níveis de serviço vinculados à qualidade do produto.

Embora um passo importante tenha sido dado em direção ao uso de métodos ágeis na AP brasileira, o relatório do TCU perdeu a oportunidade de salientar as desvantagens inerentes do desenvolvimento *big-bang* e as vantagens do desenvolvimento incremental (modular), comparativamente ao que ocorreu recentemente nos governos dos EUA e Reino Unido [GAO12] [NAO12] [HMT14]. Em vez disso, o TCU se abstém de qualquer impasse ao afirmar que todas as metodologias são viáveis [TCU13b]. Não obstante, o relatório deixou escapar a oportunidade de frisar que a falta de capacidade do governo em atuar no desenvolvimento de software pode introduzir um risco de dependência e exploração por parte das empresas terceirizadas, a qual pode tornar a indústria de software mais forte e ágil e o governo menos forte e frágil.

Por outro lado, mais recentemente, o SERPRO - uma das quinze maiores empresas de TI & Telecom do Brasil [VaE14] - anunciou a decisão de adoção de métodos ágeis em toda a Empresa [Ser14a], com alguns resultados positivos já demonstrados [Ser14b]. Na mesma direção, a partir do interesse na adoção de métodos ágeis por órgãos do SISP, a Secretaria de Logística e Tecnologia da Informação do Ministério do Planejamento, Orçamento e Gestão (SLTI) criou a Estratégia Ágil para o SISP, que representa um conjunto de ações para viabilizar e difundir o uso desse método nas instituições públicas. Entre as ações da estratégia ágil está a construção de um guia de projeto de sistemas com práticas de métodos ágeis e terceirização do desenvolvimento, o qual está em fase de Consulta Pública [Sis15]. Pretende-se, com a Consulta Pública, o recebimento de contribuições acerca de seu objeto para a consolidação da presente proposta.

3.3.2 Métodos Ágeis em Organizações Públicas: Uma Revisão Sistemática da Literatura

O principal objetivo deste tópico foi reunir e analisar evidências científicas sobre a adoção de métodos ágeis para desenvolvimento de software em OP através da execução de um estudo secundário do tipo Revisão Sistemática da Literatura (RSL).

3.3.2.1 Protocolo da Revisão Sistemática da Literatura

Esta RSL foi executada conforme as recomendações fornecidas por Kitchenham & Charters [KiC07], incluindo a elaboração do protocolo de pesquisa, a questão de pesquisa que a revisão se propõe a abordar e os critérios de seleção de estudos.

3.3.2.1.1 Questão de Pesquisa

Esta RSL foi iniciada com a seguinte questão de pesquisa.

O que se sabe sobre o desenvolvimento de software em organizações públicas (OP), incluindo a utilização de modelos, processos e métodos de Engenharia de Software (ES)?

3.3.2.1.2 Estrutura da Pergunta

Para subsidiar a atividade de construção da estratégia de busca, o escopo geral do estudo foi definido como sendo:

- **População:** Organizações públicas (OP).
- **Intervenção:** Artigos científicos que abordam o desenvolvimento de software em OP.
- **Saídas:** Os modelos, processos e métodos de Engenharia de Software adotados em OP, assim como, os resultados de sua utilização, incluindo os benefícios alcançados, os problemas e desafios enfrentados, as lições aprendidas e as recomendações para sua utilização.

3.3.2.1.3 Estratégia de Busca

Os termos de busca foram selecionados a partir de um estudo preliminar formado por um conjunto candidato de artigos, sendo organizados em duas categorias principais: aqueles relacionados com a dimensão da “Organização pública” e aqueles relacionados com a dimensão de “Desenvolvimento de software”. A Tabela 5 apresenta as palavras-chave utilizadas em cada categoria para recuperar automaticamente estudos escritos em inglês.

Tabela 5 - Palavras-chave utilizadas por categoria.

REFERÊNCIA	CATEGORIA	PALAVRAS-CHAVE
A	Organização pública	Government (1) Public sector (2) Public administration (3) Public organization (4)
B	Desenvolvimento de software	Software development life cycle (5) Software development methodology (6) Software development process (7) Software development projects (8) Software process (9) Unified process (10) Rational unified process (11) RUP (12) Microsoft solutions framework (13) Agile methodologies (14) Agile methods (15) Agile principles (16) Agile process (17) Agile software development (18) Extreme programming (19) Lean software development (20)

A estratégia de busca combinou os termos das categorias A e B com o operador booleano “AND”, sendo que os termos contidos em cada categoria foram combinados com o operador “OR”. A Categoria B possui mais termos e reflete o fato de haver muitas variações sobre a indexação dos artigos. Além disso, não se tinha conhecimento da quantidade de evidências que seriam encontradas sobre desenvolvimento de software em OP. Genericamente, a frase de busca foi definida como sendo:

(1 OR 2 OR 3 OR 4) AND (5 OR 6 OR 7 OR 8 OR 9 OR 10 OR 11 OR 12 OR 13 OR 14 OR 15 OR 16 OR 17 OR 18 OR 19 OR 20)

3.3.2.1.4 Bases de Dados

A abordagem inicial definia que as seguintes bases de dados científicas deveriam ser consultadas: ACM Digital Library, Bielefeld Academic Search Engine, ScienceDirect, Engineering Village, IEEEXplore, Scopus, SpringerLink, Web of Knowledge e Wiley Online Library. Além dessas, outras fontes de evidências científicas foram incluídas no estudo para pesquisa manual de estudos em português: Bases de Dados da Pesquisa Agropecuária, Biblioteca Digital Brasileira de Computação e Workshop Brasileiro de Métodos Ágeis.

3.3.2.2 Critérios de Seleção

Os critérios de seleção de estudos destinam-se a identificar os estudos primários que fornecem evidência científica direta sobre a questão de pesquisa. Para incluir um estudo na análise, os seguintes critérios foram adotados: i) o estudo deveria relatar a experiência na adoção de métodos ágeis para o desenvolvimento de software em OP; ii) o estudo deveria ter sido escrito em inglês ou português e iii) o estudo deveria estar disponível em texto completo para leitura e extração dos dados.

3.3.2.3 Execução da Revisão Sistemática da Literatura

Para gerenciar o grande número de referências que foram obtidos através da pesquisa bibliográfica, a ferramenta Start 2.0 [Zam+10] foi adotada para apoiar a execução desta RSL. Os estudos selecionados para análise em profundidade foram obtidos a partir de quatro etapas, conforme Figura 2.

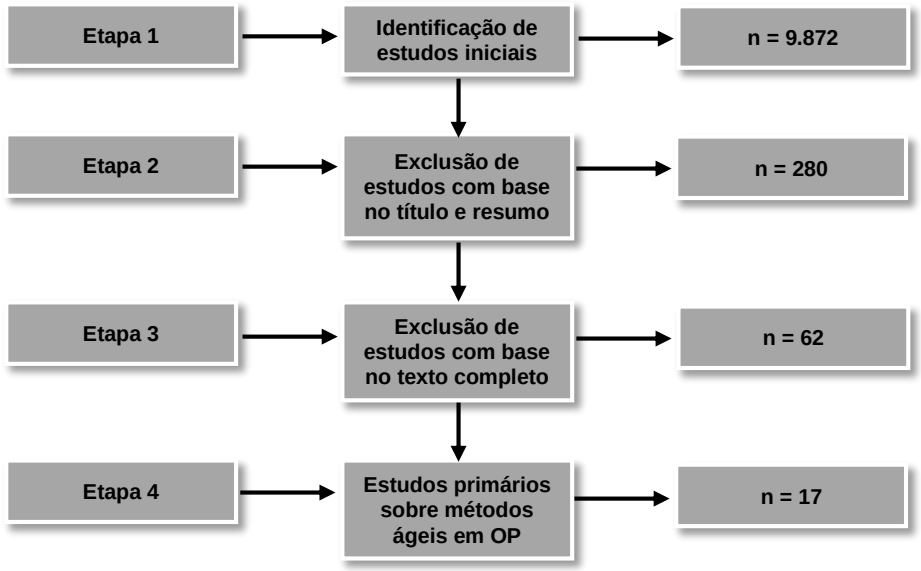

Figura 2 - Etapas do processo de seleção de estudos.

Na primeira etapa, executou-se a frase de busca em cada base de dados selecionada, sendo o resultado da pesquisa bibliográfica catalogado na ferramenta Start. A segunda etapa foi baseada na leitura do título e do resumo dos trabalhos. Nesta etapa os artigos foram classificados em três categorias:

- **[Inc]:** indica que o estudo está relacionado com a utilização de modelos, processos e métodos de ES em OP.
- **[Exc]:** indica que o estudo não está relacionado com a utilização de modelos, processos e métodos de ES em OP.
- **[Dup]:** indica que o estudo está duplicado, ou seja, repetido com outros estudos.

Todos os estudos da segunda etapa contidos nas categorias [Exc] e [Dup] foram excluídos. Na terceira etapa os trabalhos da categoria [Inc] foram analisados com mais cautela através da leitura do texto completo (introdução, conclusão, e partes específicas associadas com a contribuição principal). Nesta etapa, foi incluída uma nova categoria de exclusão de trabalhos para indicar que o texto completo do estudo não está disponível para leitura [Ntc]. Todos os estudos da terceira etapa contidos nas categorias [Exc], [Dup] e [Ntc] foram excluídos. A inclusão da terceira etapa foi adequada, pois em alguns casos, a leitura do título e resumo não foi suficiente para classificar cada artigo corretamente. Então, somente um subconjunto de documentos relacionados com desenvolvimento de software em OP contidos na categoria [Inc] foi selecionado para a etapa seguinte.

Na quarta etapa, foi incluído um classificador para indicar que o estudo está relacionado com métodos ágeis para desenvolvimento de software em OP. Somente os estudos classificados como “ágeis” foram selecionados para a fase seguinte de extração de dados e análise em profundidade. Os trabalhos analisados foram catalogados na ferramenta Start 2.0 de acordo com três categorias gerais de informação, incluindo informações gerais, informações relacionadas com a organização da pesquisa e com o conteúdo da pesquisa.

3.3.2.4 Análise dos Resultados da Revisão Sistemática da Literatura

Na primeira etapa, um total de 9.872 artigos foram encontrados. Após a triagem inicial, realizada na segunda etapa, um total de 280 artigos foram selecionados para a terceira etapa. Após a segunda triagem, realizada na terceira etapa, um total de 62 artigos foram selecionados para uma terceira triagem. Finalmente, dos 62 artigos encontrados, 17 estudos sobre métodos ágeis para desenvolvimento de software em OP foram selecionados para uma análise em profundidade, conforme Tabela 6. Foi observado que a falta de terminologia para desenvolvimento de software em OP e a baixa precisão dos mecanismos de buscas resultou em uma grande quantidade de estudos iniciais. Embora a estratégia de busca tenha retornado um número maior de artigos para seleção, somente alguns deles foram relevantes para responder a questão de pesquisa.

Tabela 6 - Conjunto final de artigos para análise em profundidade.

ID	TÍTULO	AUTORES	ANO	BASE DE DADOS
(RSL01)	A case study: Introducing eXtreme programming in a US government system development project	A. Fruhling, P. McDonald, and C. Dunbar	2008	Scopus
(RSL02)	Staying agile in government software projects	B. Upender	2005	Scopus
(RSL03)	Adoção de métodos ágeis em uma Instituição Pública de grande porte - um estudo de caso	C. de O. Melo, and G.R.M. Ferreira	2010	WBMA
(RSL04)	The FBI gets agile	C. Fulgham; J. Johnson; M. Crandall; L. Jackson; N. Burrows	2011	Scopus
(RSL05)	Collaborative development of public information systems: A case study of "Sambruk" e-services development	C.-O. Olsson, and A. Öhrwall Rönnbäck	2010	Scopus
(RSL06)	Making agile development work in a government contracting environment-measuring velocity with earned value	G.B. Alleman, and M. Henderson	2003	IEEE
(RSL07)	Agile development in a bureaucratic arena - A case study experience	H. Berger	2007	Scopus
(RSL08)	An industrial case study for Scrum adoption	H. Hajjdiab, A.S. Taleb, and J. Ali	2012	Scopus
(RSL09)	Army simulation program balances agile and traditional methods with success	J. Surdu, and D.J. Parsons	2006	Scopus
(RSL10)	A Case Study on the Adoption of Measurable Agile Software Development Process	M. Iliev, I. Krasteva, and S. Ilieva	2009	BASE
(RSL11)	Extreme Programming by example	M. Pedroso Jr, M.C. Visoli, and J.F.G. Antunes	2002	BDPA
(RSL12)	Lessons learned using agile methods on large defense contracts	P.E. McMahon	2006	Scopus
(RSL13)	Evolving to a "lighter" software process: a case study	R.J. Moore	2001	IEEE
(RSL14)	Is Agile the Answer? The Case of UK Universal Credit	R. Michaelson	2013	Springer
(RSL15)	Agile software development under university-government cooperation	S. Kaneda	2006	Scopus
(RSL16)	Exploring XP for scientific research	W.A. Wood, and W.L. Kleb	2003	Scopus
(RSL17)	Agile metrics at the Israeli Air Force	Y. Dubinsky, D. Talby, O. Hazzan, and A. Keren	2005	Scopus

A Tabela 7 apresenta uma descrição abreviada dos estudos selecionados para análise em profundidade.

Tabela 7 - Resultados e contribuições de artigos científicos sobre métodos ágeis em OP.

ID	RESULTADO OU CONTRIBUIÇÃO
(RSL01)	O estudo apresenta como XP pode ser adotado em OP. Ele apresenta um conjunto de lições aprendidas para apoiar futuras implementações de XP em ambientes governamentais e militares.
(RSL02)	O estudo apresenta uma combinação de XP e Scrum para o desenvolvimento de um sistema de gerenciamento de dados clínicos em uma organização pública.
(RSL03)	O trabalho apresenta um estudo de caso em uma organização pública brasileira de grande porte que optou por adotar métodos ágeis em alguns projetos após anos de experiência com métodos tradicionais de desenvolvimento.

ID	RESULTADO OU CONTRIBUIÇÃO
(RSL04)	O estudo apresenta o caso do projeto Sentinel, onde o *Federal Bureau of Investigation* (FBI) adotou métodos ágeis para transformar o desenvolvimento de um projeto mal sucedido em um programa bem-sucedido. Ele aponta que para ser ágil a organização pública precisa assumir o desenvolvimento do produto de software e atuar em estreita colaboração com os clientes e com as empresas da indústria de software, inclusive na mesma localização física.
(RSL05)	O estudo apresenta uma estratégia de desenvolvimento distribuído de software entre vários municípios com métodos ágeis, a qual tem oferecido uma abordagem mais ampla para enfrentar os desafios de desenvolvimento de serviços eletrônicos mais adequados às necessidades dos cidadãos da Suécia.
(RSL06)	O estudo apresenta uma abordagem ágil para atender aos requisitos de gerenciamento de valor agregado (*Earned Value Management* - EVM).
(RSL07)	O estudo apresenta uma visão de tensão que transpareceu quando a abordagem de desenvolvimento ágil foi executada em um ambiente burocrático e hierárquico. O resultado foi que as questões de conflito, confiança e uma "cultura de culpa" impediu que as partes interessadas promovessem um trabalho cooperativo e colaborativo com os desenvolvedores, o qual teve um impacto significativo sobre o progresso no desenvolvimento do projeto.
(RSL08)	O trabalho apresenta um estudo de caso em profundidade de falha na adoção de Scrum por uma equipe alocada em múltiplos projetos em uma entidade do governo dos Emirados Árabes Unidos. O estudo identifica problemas e desafios no processo de transição de abordagens prescritivas para métodos ágeis. Ele argumenta que todos os desafios podem ser superados ou minimizados se o processo de adoção de Scrum for cuidadosamente planejado, bem como, se os recursos humanos e financeiros são garantidos com antecedência.
(RSL09)	O estudo apresenta a experiência da adoção e combinação de práticas de XP e de outros métodos ágeis com modelos prescritivos de processo aplicados para o desenvolvimento de um determinado projeto de simulação em uma determinada organização pública.
(RSL10)	O estudo apresenta alguns benefícios alcançados a partir da adoção de métodos em ágeis em uma organização pública da Bulgária.
(RSL11)	O estudo apresenta a experiência da adoção de XP e suas práticas por uma organização pública brasileira em um determinado projeto. O estudo argumenta que o envolvimento do cliente requer um compromisso profundo e preparação, tanto do cliente como dos desenvolvedores, para alcançar o seu pleno potencial.
(RSL12)	O estudo compartilha várias lições aprendidas sobre o que está funcionando e o que não está funcionando ao adotar métodos ágeis em grandes projetos de software no setor de defesa do governo norte-americano. O resultado mostra que muitas das lições abordadas no estudo não são novas, e algumas podem parecer ter pouco - ou nada - a ver com métodos ágeis.
(RSL13)	O estudo descreve uma experiência de desenvolvimento de software em uma organização do governo dos EUA, onde um projeto de software submeteu-se a mudanças organizacionais, incluindo a adoção de métodos e práticas ágeis.
(RSL14)	O estudo apresenta alguns desafios para adoção de métodos ágeis em OP, incluindo a falta de comprometimento dos clientes, bem como, a necessidade dos métodos ágeis em manter conformidade com normas e regulamentos governamentais.
(RSL15)	O estudo descreve a experiência da prefeitura de Kyoto e da universidade pública de Doshisha no Japão que adotaram métodos ágeis para desenvolver dois softwares sem o apoio de empresas da indústria de software. O resultado desta estratégia mostra que os alunos e os professores beneficiaram-se com o conhecimento sobre desenvolvimento de software com métodos ágeis para o mundo real, onde ambos alcançaram uma maior compreensão da importância da ES.
(RSL16)	O estudo detalha a experiência na adoção de 12 práticas de XP em um projeto de software científico executado em um centro de pesquisa. Ele mostra que 8 das 12 práticas iniciais de XP são desafiadoras para a cultura organizacional dos centros de pesquisa e, posteriormente, o estudo mostra como superá-los.
(RSL17)	O estudo apresenta uma experiência de adoção de XP em um projeto de missão crítica em uma organização pública israelense. Posteriormente, o estudo mostra alguns benefícios alcançados com a adoção de métodos ágeis.

Uma vez que o principal interesse é o estudo da adoção de métodos ágeis em OP, a análise dos estudos incidiu sobre essa temática. Nesta análise, identificaram-se alguns aspectos da adoção de métodos ágeis, sendo organizados em duas dimensões: (1) benefícios alcançados e (2) problemas e desafios enfrentados, mostrados na Tabela 8.

Tabela 8 - Alguns aspectos da adoção de métodos ágeis em OP identificados na RSL.

DIMENSÃO	ASPECTO	FONTES
BENEFÍCIOS	Entrega de valor ao cliente mais cedo	(RSL02) (RSL09) (RSL12) (RSL13)
	Maior alinhamento e colaboração entre TI e negócios	(RSL01) (RSL02) (RSL04) (RSL05) (RSL07) (RSL09) (RSL10) (RSL17)
	Aumento na satisfação do cliente	(RSL01) (RSL02) (RSL03) (RSL04) (RSL10)
	Aumento na moral da equipe e redução da dependência de empresas contratadas	(RSL11) (RSL17)
	Melhoria na comunicação	(RSL01) (RSL02) (RSL04) (RSL09)
	Melhoria no aprendizado de novas tecnologias	(RSL03)
	Melhoria na qualidade do produto e do código	(RSL16)
	Melhoria na visibilidade do projeto	(RSL02) (RSL12) (RSL17)
	Aumento na produtividade das equipes	(RSL03) (RSL16)
	Redução de custos	(RSL07)
	Melhoria na capacidade de gerenciar mudanças e prioridades	(RSL05) (RSL17)
PROBLEMAS E DESAFIOS	A cultura organizacional e a resistência a mudanças	(RSL01) (RSL02) (RSL03) (RSL07) (RSL08) (RSL14) (RSL15) (RSL16) (RSL17)
	A falta de conhecimento e experiência em métodos ágeis	(RSL01) (RSL02) (RSL03) (RSL08) (RSL11) (RSL14) (RSL17)
	O pouco ou nenhum comprometimento das partes interessadas	(RSL07)
	O uso arraigado de abordagens prescritivas	(RSL01) (RSL03) (RSL04) (RSL08) (RSL15) (RSL17)
	A tendência de mega projetos de tecnologia da informação	(RSL04) (RSL14)
	O problema com contratos e contratações	(RSL01) (RSL03) (RSL04)
	A conformidade com normas e regulamentos	(RSL04) (RSL06) (RSL09) (RSL12)
	O apoio da alta administração	(RSL03) (RSL07) (RSL08) (RSL16) (RSL17)
	O problema com atrasos	(RSL01) (RSL04) (RSL07) (RSL08)

3.3.2.5 Algumas Lições da Revisão Sistemática da Literatura

Esta seção tem por objetivo descrever algumas lições oriundas da revisão sistemática.

Lição #1: Modelos adaptativos produzem resultados melhores para as OP.

A lição #1 é que modelos adaptativos produzem resultados melhores do que seria possível alcançar com métodos ditos como tradicionais em OP em determinadas situações. O principal argumento é que modelos prescritivos não fornecem a visibilidade do produto de software ou resultados concretos até o final do ciclo de vida do projeto, sendo considerado adequado para situações onde os requisitos são estáveis. Consequentemente, modelos adaptativos, principalmente métodos ágeis, evoluíram para proporcionar uma abordagem de desenvolvimento mais flexível e adequada para situações onde os requisitos são voláteis e incertos. O sucesso de modelos adaptativos deve-se principalmente ao fato de que os resultados priorizados são entregues mais cedo, e em incrementos muito menores do que em projetos prescritivos que utilizam o Modelo Cascata, onde cada etapa do projeto deve ser totalmente concluída antes de prosseguir para a etapa seguinte. Secundariamente, há um elemento de aprendizagem inserido em cada ciclo de entrega e uma elevação na moral da equipe do governo.

Lição #2: OP precisam assumir a sua responsabilidade no desenvolvimento de software.

A lição #2 mostra que para ser adaptativo as OP precisam criar as suas próprias capacidades organizacionais e técnicas para atuar em estreita colaboração, em todas as etapas do ciclo de vida do projeto, com as empresas contratadas da indústria de software, e não apenas monitorarem e verificarem os resultados finais das tarefas realizadas. Há indícios de que os governos deveriam parar de contratar o desenvolvimento de software por inteiro com grandes etapas de entrega, bem como, deveriam parar de transferir a responsabilidade do desempenho e resultados de projetos para as empresas contratadas. De acordo com Pyster [Pys06], o reflexo da fragilidade na capacidade do governo em atuar de maneira mais incisiva ao longo do desenvolvimento de software tem causado um impacto negativo no custo, cronograma, escopo e desempenho técnico em vários projetos de software em OP, mesmo que estes projetos tenham sido executados por empresas com níveis elevados de capacidade e maturidade. Heil [Hei10] concorda com esse pensamento e acrescenta que a opção do governo em transferir completamente o trabalho de desenvolvimento para a indústria raramente tem sido uma alternativa viável. O impacto de custo e cronograma tem feito com que as únicas opções de mitigação de risco sejam: aumentar significativamente os recursos financeiros, atrasar significativamente o

cronograma, reduzir ou eliminar significativamente as funcionalidades previstas, ou então, cancelar o projeto.

De acordo com Heil [Hei10], para reverter esse quadro, as OP precisam assumir a liderança e necessitam aumentar a sua participação em projetos de desenvolvimento de software, principalmente os de missão crítica para a AP [Hei10]. Na mesma direção, o estudo de Bannerman [Ban08] constatou que algumas OP aprenderam ao longo do tempo que melhores resultados são alcançados quando desenvolvedores terceirizados trabalham com profissionais do governo na mesma localização física (*insourcing*). Esta integração física tem melhorado a comunicação do projeto, a interação entre as pessoas, a resolução mais rápida de problemas e o maior acompanhamento do progresso do projeto, bem como, tem permitido que o desenvolvimento incremental aconteça na sua essência, o que tem resultado na entrega de sistemas mais aderentes as necessidades das OP [Ban08].

Lição #3: Existe uma necessidade por novos modelos de contrato para apoiar o desenvolvimento ágil de software entre empresas contratadas e OP.

Entende-se que o sucesso alcançado nos projetos *Sentinel* (RSL04) e *OneSAF-OSS* (RSL09) demonstra que o modelo *in-house* de equipe de desenvolvimento de software combinado com modelos adaptativos não é apenas conceitual, mas tem produzido melhores resultados na prática. Isto cria uma oportunidade para pesquisadores compreenderem e explorarem esta área a partir de uma perspectiva técnica-jurídica, incluindo informações sobre diferentes modelos de contrato, tais como "Tempo e Recursos Materiais" e “Escopo Negociável”, assim como, informações sobre o ambiente de execução do projeto, onde desenvolvedores da indústria e governo, clientes e usuários finais se instalam no mesmo ambiente e compartilham as mesmas expectativas e objetivos.

Lição #4: Existe a oportunidade de avaliar empiricamente a adoção de métodos ágeis em OP.

Uma vez que modelos adaptativos não têm sido testados e explorados suficientemente na AP, assim como, dado o conjunto restrito de evidências encontradas para extrair resultados conclusivos, mas ao mesmo tempo a existência de resultados promissores do uso de modelos adaptativos na AP, existe uma oportunidade para pesquisadores atuarem em OP para realizarem estudos empíricos para entender como estes modelos são executados na prática e que efeitos eles geram, incluindo suas vantagens e desvantagens.

3.3.3 Métodos Ágeis em Organizações Públicas: Uma Revisão de Obra Especializada

Após a execução da RSL vislumbrou-se a oportunidade de ampliar a quantidade de estudos primários sobre o tema por meio de uma consulta *ad hoc* em livros especializados e sítios *web* governamentais do Brasil, EUA e Reino Unido. Esta estratégia permitiu encontrar a publicação de Brian Wernham intitulada "*Agile Project Management for Government*" [Wer12], sendo a primeira obra conhecida que visualiza a adoção da abordagem ágil no âmbito das OP. Ele apresenta cinco casos de sucesso em OP mostrando que também é possível entregar softwares de missão crítica com métodos ágeis. Em seguida, uma descrição abreviada dos cinco estudos de caso abordados por Wernham [Wer12] é apresentado na Tabela 9.

Tabela 9 - Resultados e contribuições de casos governamentais sobre métodos ágeis em OP.

ID	RESULTADO OU CONTRIBUIÇÃO
(ROE01)	O estudo apresenta a adoção de DSDM em um projeto de software em uma organização pública do governo do Reino Unido. Ele mostra como um método de desenvolvimento de software superou as expectativas e foi determinante para resolver os problemas encontrados em projetos tradicionais baseados em modelos prescritivos de processo. Ele destaca a importância de contratos de escopo negociável para alterar o escopo do projeto em direção a solução ideal, sempre respeitando os custos que são fixos.
(ROE02)	O estudo apresenta como o *US Department of Veterans Affairs* (VA) conseguiu entregar um sistema complexo de processamento de subsídios para ex-militares norte-americanos em poucos meses com resultados positivos. Ele argumenta que o uso de modelos prescritivos de processo seria incapaz de construir uma solução de software em tão pouco tempo, sendo necessário uma abordagem ágil. Por último, o estudo identifica alguns problemas de implementação de software que podem ser úteis para outros projetos ágeis.
(ROE03)	O estudo mostra que a insistência do FBI em contratar o desenvolvimento de software por inteiro com grandes etapas de entrega baseado em abordagens prescritivas de processo levou o FBI a um histórico de falhas e atrasos em projetos de desenvolvimento de software. Posteriormente, ele apresenta como o FBI inverteu esse histórico, assumindo a liderança do projeto com métodos ágeis, o que reduziu a dependência de empresas contratadas da indústria de software e diminuiu os riscos do projeto. Por último, o estudo relata que o FBI conseguiu construir com somente 52% do orçamento 88% do sistema em apenas um ano, contra um histórico de 10 anos de atraso e uso ineficiente do dinheiro público. Vale ressaltar que este estudo está repetido com o estudo (RSL04) identificado na RSL.
(ROE04)	O estudo mostra como o Estado de Queensland na Austrália adotou o método ágil Scrum para mais de 30 projetos. Ele argumenta que a adoção inicial quase foi prejudicada por uma prática de trabalho que, de certa maneira, obrigava que todos os requisitos do sistema fossem identificados e especificados detalhadamente, à priori. Esse impasse foi superado com a decisão pelo uso combinado das abordagens Scrum e PRINCE2.
(ROE05)	O estudo apresenta uma combinação de XP, Scrum e DSDM para o desenvolvimento do novo sistema de armazenamento, processamento e provisionamento de informações meteorológicas do UK Met Office. Ele demonstra como a abordagem *Enough Design Up Front* (EDUF) pode ser usada para identificar a arquitetura geral do sistema e planejar os ciclos de entrega de software. Por último, o estudo relata que a implantação gradual do novo sistema permitiu que o sistema anterior fosse descontinuado mais cedo, o que gerou uma economia anual de 1 milhão de libras esterlinas.

A partir da experiência adquirida com a execução da revisão sistemática, preferiu-se repetir a mesma abordagem utilizada para análise de dados da RSL nesta revisão de obra especializada. A Tabela 10 apresenta os aspectos identificados.

Tabela 10 - Alguns aspectos da adoção de métodos ágeis em OP identificados na ROE.

DIMENSÃO	ASPECTO	FONTES
BENEFÍCIOS	Entrega de valor ao cliente mais cedo	(ROE05)
	Maior alinhamento e colaboração entre TI e negócios	(ROE01) (ROE03) (ROE05)
	Aumento na satisfação do cliente	(ROE03) (ROE04) (ROE05)
	Aumento na moral da equipe e redução da dependência de empresas contratadas	(ROE01) (ROE03) (ROE04)
	Aumento na qualidade do produto e do código	(ROE01) (ROE02)
	Redução de custos	(ROE03) (ROE04) (ROE05)
	Melhoria na capacidade de gerenciar mudanças e prioridades	(ROE02) (ROE03) (ROE04)
PROBLEMAS E DESAFIOS	A falta de conhecimento e experiência em métodos ágeis	(ROE01) (ROE04)
	O uso arraigado de abordagens prescritivas	(ROE01) (ROE03) (ROE04)
	A tendência de mega projetos de tecnologia da informação	(ROE03)
	O problema com contratos e contratações	(ROE01) (ROE03)
	A conformidade com normas e regulamentos	(ROE03) (ROE04)
	O problema com atrasos	(ROE01) (ROE03) (ROE05)

3.4 Consolidação dos Resultados Teóricos

Este capítulo evidenciou que métodos ágeis têm sido adotados em OP com resultados positivos, sendo apoiados por evidências científicas. Eles têm sido adotados em projetos: (1) grandes (ROE03) e pequenos (RSL16); (2) simples (RSL01), complicados (ROE01) (ROE02) e complexos (ROE04) (ROE05); (3) colocalizados (ROE03) e distribuídos (RSL05); (4) “puramente” ágeis (RSL01) (RSL17) e combinados com outras abordagens (RSL09) (ROE04); (5) de valor de curto prazo (ROE05) e valor de longo prazo (RSL16); (6) desenvolvidos inteiramente por empregados internos (RSL03) (RSL13) e com a participação de empresas contratadas (RSL09) (ROE03), ou então, em parceria com outras OP (RSL05) (RSL15); (7) em diferentes OP de diferentes países, incluindo departamentos de TI, centros de pesquisa, empresas públicas, agências de governo, etc.; (8) que atuam em vários segmentos da cadeia de valor do governo, incluindo o setor agropecuário, aeroespacial, militar, bancário, habitacional, etc. Além disso, seus benefícios são endossados pelos governos dos EUA [GAO12], Reino Unido [HMT14] [NAO12] e Brasil [TCU13b].

Vale salientar que os resultados coletados a partir dos estudos teóricos realizados em projetos de desenvolvimento ágil de software em OP serão amplamente consolidados no capítulo 5, formando o arcabouço de resultados teóricos e empíricos. Essa abordagem foi necessária para não repetir o mesmo texto em diferentes pontos desta pesquisa.

4 ESTUDOS DE CASO

Este capítulo mostra os instrumentos e procedimentos metodológicos e os critérios de validade científica adotados para abordar a realidade estudada, na seção 4.1. Na sequência, ele apresenta e mapeia os dados coletados juntos aos sujeitos da pesquisa, na seção 4.2 e 4.3. Por fim, uma consolidação dos resultados é descrita na seção 4.4.

4.1 O Estudo de Caso como Opção de Pesquisa Qualitativa

Diante de poucos estudos sobre a adoção de métodos ágeis para o desenvolvimento de software em OP, principalmente no Brasil, começou-se a vislumbrar a possibilidade de aumentar o inventário de casos e sua acessibilidade. Considerando que a pesquisa empreendida quer entender como se desenvolvem softwares no interior de uma organização pública, fica evidenciado que o objeto de estudo está circunscrito no âmbito das ciências humanas e sociais e que uma pesquisa com este caráter direciona para o uso de um método de pesquisa qualitativo, porque o que se procura são respostas que indiquem o estado real do objeto delimitado desvendando como ele ocorre a partir dos próprios sujeitos que o exercitam.

Dentre as várias alternativas de pesquisa qualitativa optou-se pelo estudo de caso porque este se fundamenta, conforme ressalta Yin [Yin10], na identificação de respostas para questões em que o problema de pesquisa consiste em buscar o “como” e o “por que”, bem como, na focalização de eventos contemporâneos complexos com contexto de vida real e no esforço para manter as características universais do contexto estudado.

Ademais, Graham [Gra10] destaca que estudos de caso propiciam uma ferramenta rica para apresentação de problemas na AP, de tal forma que os usuários - estudantes, profissionais ou professores - podem auferir benefícios significativos. Ele reforça que a riqueza da vida organizacional pública não está em suas bases de dados, mas sim nas pessoas, suas histórias e na forma como elas enfrentaram problemas, com maior ou menor êxito. O governo oferece um contexto repleto de complexidades, dramas e ambiguidades. Captar essa combinação de elementos é a vantagem dos estudos de caso, seja para o conhecimento da prática, seja para o estudo.

Do ponto de vista metodológico, o método de estudo de caso se mostrou a opção mais adequada porque, ao mesmo tempo, complementa os interesses da pesquisa e oferece oportunidades de enxergar a realidade entendendo-a e apreendendo-a na sua integralidade. Também é um método que garante a fundamentação nos dados e a facilidade na exploração de nuances, contradições e paradoxos.

4.1.1 A Seleção das Organizações Públicas da Pesquisa

Esta pesquisa iniciou-se com uma amostra diferenciada, com duas OP, incluindo a Embrapa Informática Agropecuária (CNPTIA) e a Companhia de Processamento de Dados do Estado do Rio Grande do Sul (PROCERGS) porque se supunha que existissem condições diferentes em contextos de governos diferentes que afetam o desenvolvimento de software em OP. A escolha destas duas OP foi feita *a priori* por razões meramente organizativas, referindo-se à disponibilidade de recursos financeiros, ao tempo à disposição, a oportunidade de acesso ao objeto de interesse (projetos de software com métodos ágeis) e a possibilidade de recrutar sujeitos para o estudo. Por outro lado, a amostragem de organizações típica de manual teria exigido, ao contrário, iniciar em um contexto e ampliar a amostra apenas quando as ideias que emergiam da análise tivessem exigido isso. Ampliar *a priori* a amostra não é, em si, um erro metodológico, porém obriga a coletar dados que podem não ser essenciais, perdendo para tanto tempo, energias e recursos [Tar11].

4.1.1.1 A Embrapa Informática Agropecuária

A Empresa Brasileira de Pesquisa Agropecuária (Embrapa) conta atualmente com 17 Unidades Centrais localizadas em Brasília, 46 Unidades Descentralizadas em todas as regiões do Brasil, 4 Laboratórios Virtuais no Exterior (EUA, Europa, China e Coréia do Sul) e 3 Escritórios Internacionais na América Latina e África[3]. Tanto as Unidades Centrais quanto as Unidades Descentralizadas estão vinculadas a um único modelo institucional, ou seja, a um sistema corporativo de estrutura administrativa e funcional formal e de normas operativas que se complementam, se articulam e se organizam entre si para favorecer a realização da missão institucional da Empresa [Tor08].

Dentre as suas 46 Unidades Descentralizadas encontra-se a Embrapa Informática Agropecuária, localizada na cidade de Campinas, Estado de São Paulo, unidade de pesquisa cuja missão atual é “viabilizar soluções de pesquisa, desenvolvimento e inovação em tecnologia de informação para a sustentabilidade da agricultura, em benefício da sociedade brasileira” [EIA08].

Em 1985, durante a vigência da Reserva de Mercado de Informática, não se visualizava com clareza o futuro da tecnologia no Brasil. Nesse contexto, a Diretoria-Executiva da Embrapa firmou convênio com o Centro Tecnológico para Informática (CTI), vinculado à Secretaria Especial de Informática (SEI) da Presidência da República, com o

[3] Capturado em https://www.embrapa.br/quem-somos, Setembro 2014.

objetivo de desenvolver o projeto denominado Fábrica de Software. Foi criado, dessa forma, o Núcleo Tecnológico para Informática Agropecuária (NTIA), funcionando nas instalações do CTI, o atual Centro Nacional de Pesquisas Renato Archer (CenPRA).

O surgimento do NTIA, segundo pondera Seixas Neto *et al* [SJP07], preencheu uma lacuna existente no País que era a ausência de um organismo institucionalizado que servisse de centro de integração de metodologias, ferramentas e processos avançados de produção de software para possibilitar tanto o aumento da produtividade quanto o de qualidade no desenvolvimento de software, de maneira que pudesse, ao mesmo tempo, disseminar as tecnologias geradas para a indústria e para as empresas brasileiras do setor e automatizar a produção de softwares. Acreditava-se que um órgão com tais características contribuiria para alavancar a independência tecnológica do Brasil no setor [Tor08].

A Embrapa, quando da criação do NTIA, soube adotar a informática como um fator estratégico porque percebeu que esta ferramenta poderia ampliar sua capacidade de gerar novos conhecimentos e de transformá-los em tecnologias de fácil acesso e adoção pela sociedade [Tor08]. Assim ficou consignado que a informática na Embrapa contribuiria tanto para a melhoria e agilidade dos processos de decisão administrativa quanto para avançar pesquisas sobre sensoriamento remoto, processamento digital de imagens, levantamentos agroecológicos e socioeconômicos, produção de cartografias de ocupação de terras, cartas morfopedológicas, perfis descritivos de agricultores, biotecnologia, sequenciamento genético e bioinformática, dentre outros [Tor08]. O desenvolvimento da pesquisa agropecuária “pode beneficiar-se e, até mesmo, ser acelerado através de adequada utilização dos instrumentos de informática” [EMB86].

Em 1993, o Núcleo recebeu status de centro nacional e transformou-se no atual Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura (CNPTIA). Nesse ano foi celebrado o convênio com a Universidade Estadual de Campinas (Unicamp) para a cessão de uso de terreno na Cidade Universitária Zeferino Vaz, onde seria construída a futura sede da Unidade, utilizando recursos do Banco Interamericano de Desenvolvimento (BID) por meio do Programa de Modernização Tecnológica da Agropecuária (Promoagro). No início de 1996, a inauguração da sede própria da Unidade representou a consolidação da infraestrutura adequada para o desenvolvimento dos trabalhos de pesquisa. Ainda em 1996, em decorrência da implantação da política de comunicação da Empresa e com o objetivo de fortalecer a marca Embrapa, foram criadas as assinaturas-sínteses das unidades e o CNPTIA passou a ser denominado Embrapa Informática Agropecuária.

Atualmente a Embrapa Informática Agropecuária conta com um quadro de 109 pessoas das quais 74 são dedicadas às atividades de pesquisa e desenvolvimento e os demais às atividades de transferência de tecnologia, comunicação & negócios e suporte à pesquisa, sendo que das 109 pessoas, 20 atuam com o desenvolvimento de software na unidade, 14 atuam mais no gerenciamento de projetos de software e 8 trabalham com infraestrutura de TI. Vale salientar que essas pessoas atuam em processos administrativos e/ou de pesquisa de caráter estratégico, tático e/ou operacional para viabilizar e suportar as ações de pesquisa da unidade.

4.1.1.2 A Companhia de Processamento de Dados do Estado do Rio Grande do Sul

A Companhia de Processamento de Dados do Estado do Rio Grande do Sul (PROCERGS) é uma empresa de economia mista vinculada à Secretaria da Fazenda, cujo maior acionista é o Governo do Estado do Rio Grande do Sul. Ela iniciou suas atividades em 28 de Dezembro de 1972 como órgão executor da política de informática do Estado. Hoje, a Empresa busca oferecer soluções em TI e Comunicação para a AP visando aumentar a eficiência e transparência do serviço público e aproximar Governo e Cidadão[4]. Ela não era conhecida da população como outras empresas públicas do Rio Grande do Sul (RS), a exemplo da Companhia Estadual de Energia Elétrica (CEEE). Entretanto, na última década, à medida que a Tecnologia da Informação ganhou força em vários setores, e o setor público não é exceção, a PROCERGS tornou-se uma força importante na AP. Sem ela, os serviços dos departamentos de trânsito, os serviços da Secretaria da Agricultura, Pecuária e Agronegócio, os serviços de interconexão dos diversos órgãos estaduais e muitos outros serviços da AP do Rio Grande do Sul, cada vez mais dependente de tecnologia, simplesmente não aconteceriam. Ou então, estariam em um estágio não muito desenvolvido. A Empresa extrapola o âmbito do governo ao oferecer alguns serviços à iniciativa privada. É o caso do Via-RS, o primeiro provedor de Internet do Estado do RS, que em 2015 completa 20 anos.

Hoje a PROCERGS conta com um quadro de aproximadamente 1.206 funcionários das quais aproximadamente 960 são dedicadas às atividades de TI, sendo que aproximadamente 600 servidores públicos atuam em projetos de desenvolvimento de software. Além disso, ela é formada por mais 6 Coordenadorias Regionais e 6 Unidades Descentralizadas. Vale salientar que o desenvolvimento de software acontece unicamente de maneira centralizada na Sede da Empresa em Porto Alegre/RS.

[4] Sítio web da PROCERGS: http://www.procergs.rs.gov.br/

4.1.2 A Seleção dos Sujeitos da Pesquisa

A partir de projetos ágeis de desenvolvimento de software executados em OP - como ponto de partida para a identificação de estratégias de coleta de dados - a seleção dos sujeitos que compuseram a pesquisa foi realizada em função de dois focos de atenção definidos nesta pesquisa como dimensão de análises. A *dimensão organizacional* é a primeira delas. Ela levantou, junto aos gestores, aspectos institucionais relevantes acerca da cultura organizacional. Parte-se do pressuposto que a cultura organizacional influencia, promove, facilita, dificulta, favorece ou e/ou torna a adoção de métodos ágeis com maior ou menor possibilidade de aceitação na organização. Enquanto a *dimensão do projeto* focou a percepção dos membros sobre como compreendem o trabalho em equipe, o alinhamento com os objetivos de negócio e a execução de processos e práticas de desenvolvimento de software; entretanto, se interessou por conhecer como o desenvolvimento de software acontece e se desenvolve no interior dos projetos. Considerando a definição das dimensões escolhidas e também a experiência profissional dos pesquisadores optou-se pelas quantidades de sujeitos por dimensão de amostras consignadas na Tabela 11.

A questão da quantidade de sujeitos em uma pesquisa qualitativa é controvérsia e se insere no debate sobre a representatividade da amostra. Patton [Pat02] admite que a riqueza de informações do caso analisado é que garante a validade e significância da pesquisa qualitativa e não a quantidade de sujeitos que estão envolvidos na coleta destas informações. Creswell [Cre14] e Yin [Yin10], por outro lado, concordam que o número de sujeitos que devem compor uma amostra na pesquisa qualitativa depende da questão de pesquisa e da visão do pesquisador quanto à robustez dos dados e do grau de saturação qualitativa. Tarozzi [Tar11], por sua vez, aponta que o critério para estabelecer quando se pode interromper o procedimento da amostragem é a saturação teórica, no sentido de que, para qualquer direção que se prossiga na coleta de dados, confirmam-se constantemente os mesmos resultados alcançados anteriormente, enquanto que Minayo [Min98] assegura que a fala de alguns indivíduos de um grupo é representativa de grande parte dos membros deste mesmo grupo.

Tabela 11 - Quantitativo de sujeitos da pesquisa por OP e dimensão.

ORG. PÚBLICA	DIMENSÃO	SUJEITOS
CNPTIA	Organizacional	1 Assessor
	Projeto (1)	2 Desenvolvedores 1 Gerente de Projeto
	Projeto (2)	1 Gerente de Projeto 1 Desenvolvedor
	Total	**6 sujeitos**
PROCERGS	Organizacional	1 Gestor(a)
	Projeto (3)	1 *Product Owner* 1 Analistas de Sistemas 1 *ScrumMaster* / Desenvolvedor 1 Analista de Qualidade 2 Desenvolvedores
	Projeto (4)	1 Analista de Sistemas 1 *ScrumMaster* / Desenvolvedor 2 Desenvolvedores 1 Analista de Qualidade
	Total	**12 sujeitos**

Nesta pesquisa a quantidade de sujeitos mostrou-se adequada para ajudar a responder a questão de pesquisa; além disso, o fato de eles viverem no mesmo contexto social os levou a desenvolver e reproduzir disposições semelhantes e, por isto mesmo, os significados individuais ofereceram maiores possibilidades de representarem significados grupais. A seleção dos sujeitos que compuseram cada uma das dimensões de amostras foi definida da seguinte maneira.

1) Na Embrapa Informática Agropecuária:
   a) Os dois projetos de desenvolvimento de software selecionados para esta pesquisa foram escolhidos pela chefia de Pesquisa & Desenvolvimento (P&D). Ambos apresentaram evidências do uso de práticas ágeis durante a execução do projeto;
   b) Uma vez feita a seleção dos projetos de desenvolvimento de software, solicitou-se aos gerentes de projeto, dentre os membros do projeto, pessoas para compor a amostra de sujeitos da dimensão de projetos. A amostra selecionada foi composta de 5 sujeitos, sendo distribuídos em 2 projetos;
   c) A dimensão organizacional foi composta por um dos assessores da chefia, sendo responsável pelo Núcleo da Garantia da Qualidade (NGQ). Essa decisão mostrou-se adequada por que esta assessoria, além de outros compromissos, é responsável por entender as demandas das pessoas que desenvolvem software na Empresa.

2) Companhia de Processamento de Dados do Estado do Rio Grande do Sul:
   a) Os dois projetos de desenvolvimento de software selecionados para esta pesquisa foram escolhidos pelo Setor de Desenvolvimento 1 (SD1). Ambos apresentaram evidências no uso do método ágil Scrum. Vale ressaltar que o contato inicial com o SD1 foi intermediado por um renomado especialista em métodos ágeis que atuou como consultor na PROCERGS em 2014;
   b) Uma vez feita a seleção dos projetos de desenvolvimento de software, solicitou-se a chefia do SD1, dentre os membros dos projetos, pessoas para compor a amostra de sujeitos da dimensão de projetos. A amostra selecionada foi composta de 11 sujeitos, sendo distribuídos em 2 projetos;
   c) A dimensão organizacional foi composta pela chefia do SD1. Essa decisão mostrou-se adequada por que este setor é responsável por desenvolver projetos e versões de sistemas que atendam às necessidades dos clientes da Empresa. Como também, este setor contribui na prospecção de metodologias e ferramentas que suportam o processo de desenvolvimento, as quais são experimentadas e avaliadas em casos reais de projeto.

4.1.3 Os Instrumentos de Coleta de Dados

De acordo com Yin [Yin10], há três princípios que norteiam os estudos qualitativos: 1) o uso de fontes múltiplas (duas ou mais) de instrumentos de coletas de dados para que o pesquisador possa ter evidências que convirjam para o mesmo conjunto de fatos ou descobertas; 2) a elaboração de uma base de dados construída a partir da reunião das evidências emergidas em função do princípio anterior; e 3) a existência de vinculação clara entre as questões, os dados coletados e as conclusões. No âmbito de instrumentos de coleta de dados, esta pesquisa está fundamentada na utilização de duas técnicas: (1) entrevistas e (2) análise de documentos. Vale salientar que a adoção da observação como instrumento de coleta de dados se tornou inviável porque os projetos do CNPTIA e da PROCERGS são em redes e, em muitas vezes, interinstitucionais ou interdepartamentais, o que impede a participação de outras pessoas como observadoras. Fora estes fatos, a observação exigiria uma imersão no ambiente onde os fatos acontecem, o que é de difícil aceitação por parte dos sujeitos e das organizações selecionadas.

4.1.3.1 Entrevistas

Uma vez que o desenvolvimento de software é executado por pessoas, percebeu-se que a questão de pesquisa só seria entendida se incorporasse a experiência e a intersubjetividade dos próprios indivíduos que habitam o espaço físico circunscritos nos estudos de caso analisados. Este fato por si só justifica a opção da entrevista como instrumento de coleta de dados. Além disso, entrevistas, mesmo não sendo o único, continua sendo o instrumento principal de coleta de dados em estudos de caso, especialmente aquela executada frente a frente com o entrevistado, onde o pesquisador atua diretamente com o sujeito para fazer perguntas e registrar suas respostas [Bha12].

Dentre os vários tipos de entrevistas optou-se pela entrevista individual, seguindo um protocolo de entrevista semiestruturado, detalhado no APÊNDICE B, o qual foi aprovado em Setembro de 2014. Este tipo de entrevista se mostrou o mais indicado para coletar as informações acerca da experiência dos sujeitos no desenvolvimento de software em OP. Também, buscou-se conhecer a importância que os sujeitos da pesquisa dão aos métodos ágeis enquanto atuam profissionalmente em suas atividades.

4.1.3.2 Análise de Documentos

A análise de documentos foi o segundo instrumento de coleta de dados na qual fundamentou-se esta pesquisa para garantir a validade dos resultados. Os tipos de documentos analisados são apresentados na Tabela 12, os quais foram disponibilizados com acesso restrito à esta pesquisa (com exceção do Plano Diretor e Sítio *web* da OP).

Tabela 12 - Tipo de documento analisado por OP e dimensão.

ORG. PÚBLICA	DIMENSÃO	TIPO DE DOCUMENTO
CNPTIA	Organizacional	- Plano Diretor da Empresa - Sítio *web* da Empresa - Lista de empregados da Empresa
	Projeto (1)	- Documento de aprovação do projeto - Documento de visão do projeto - *Backlog* do produto - Relatório detalhado dos requisitos - Sítio *web* do projeto
	Projeto (2)	- Documento de aprovação do projeto - Relatório final do projeto - Atas do Comitê Técnico Interno - Ferramenta de apoio ao processo de desenvolvimento de software
PROCERGS	Organizacional	- Sítio *web* da Empresa - Manual da Organização
	Projeto (3)	- E-mail com informações sobre a caracterização do projeto - Ferramenta de apoio ao processo de desenvolvimento de software
	Projeto (4)	- E-mail com informações sobre a caracterização do projeto

4.1.3.3 Entrada em Campo

Na tentativa de aprender algo através da experiência dos sujeitos em suas organizações, tão logo o protocolo de estudo de caso foi validado na primeira semana de Setembro de 2014, providenciou-se a entrada em campo nas organizações selecionadas. No caso da Embrapa Informática Agropecuária, a primeira providência adotada, tão logo os projetos e os sujeitos foram selecionados, foi contatá-los para agendar data e horário para a entrevista, bem como, solicitar a sua colaboração deixando-os a vontade para optarem por participar ou não participar da entrevista. Uma vez obtido o consentimento, as entrevistas ocorreram de acordo com a disponibilidade dos sujeitos.

No que se refere a condução das entrevistas, optou-se por registrar as entrevistas eletronicamente em áudio (com o consentimento dos entrevistados) para referência futura. O áudio das entrevistas foi transformado em texto o mais rápido possível. Cada entrevista foi ouvida pelo menos três vezes pelo mesmo pesquisador, visando buscar a estabilidade do texto das entrevistas. Uma abordagem de transcrição completa das entrevistas não foi necessária. Isto quer dizer que as falas não foram transcritas tal qual foram faladas nas entrevistas, até porque, para tornar a leitura mais amena, foram retirados, sem prejuízo do conteúdo central, alguns vícios de linguagem, repetições e solecismos. Por outro lado, a fala coloquial dos sujeitos foi preservada por representar a cultura do espaço em que desenvolvem suas práticas profissionais.

Esta proposta de gravar em áudio as entrevistas e transformá-las em texto foi o que permitiu: (1) colher as nuances, sendo algumas delas reveladoras; (2) focar na relação com o sujeito durante a conversação, sem a preocupação de lembrar ou de tomar notas; (3) permanecer fiel ao fenômeno, o mais próximo possível das palavras dos entrevistados.

Cabe salientar que os textos das entrevistas consubstanciaram um documento com 97 páginas, denominado “Texto das Entrevistas”, que serviu para análise de conteúdo que será discutido mais adiante. A Tabela 13 registra os períodos de realização das entrevistas e os períodos de conversão em texto e escrita dos estudos de caso por OP e dimensão da amostra da pesquisa.

Por último, cada uma das falas selecionadas para representar as ideias constantes nas análises será sucedida das informações: nome fictício do sujeito que concedeu a entrevista; data da entrevista; e dimensão da amostra à qual o sujeito pertenceu. Esta última informação é designada da seguinte forma: *dp* (dimensão de projeto) e *do* (dimensão organizacional).

Tabela 13 - Período das entrevistas e conversão em texto por OP e dimensão.

ORG. PÚBLICA	DIMENSÃO	PERÍODO DE REALIZAÇÃO DAS ENTREVISTAS	PERÍODO DE CONVERSÃO EM TEXTO DAS ENTREVISTAS	PERÍODO DA ESCRITA DOS ESTUDOS DE CASO
CNPTIA	Organizacional	18/09/2014	16/09 a 22/09/2014	15/09 a 07/10/2014
	Projeto (1)	15/09 e 17/09/2014	16/09 a 22/09/2014	18/09 a 28/09/2014
	Projeto (2)	23/09/2014	29/09 e 30/09/2014	02/10 a 05/10/2014
PROCERGS	Organizacional	14/10/2014	22/10/2014	31/10 a 01/11/2014
	Projeto (3)	14/10 e 15/10/2014	16/10 a 22/10/2014	27/10 a 29/10/2014
	Projeto (4)	15/10/2014	16/10 a 22/10/2014	30/10 a 31/10/2014

Cabe salientar que algumas providencias foram adotadas no dia da entrevista com os sujeitos:

- Estar presente no mínimo quinze minutos de antecedência ao horário marcado;
- Informar ao sujeito sobre o tema da pesquisa, a sua importância e os seus benefícios para o governo, tal como, a questão e os objetivos da pesquisa;
- Evitar fazer anotações sobre a entrevista no seu decorrer. Não registrar qualquer informação observada sobre o sujeito na sua presença;
- Deixar claro ao sujeito como se chegou até ele explicando a forma de seleção e registrando que, além dele, estarão participando da pesquisa mais outros profissionais de sua organização;
- Informar ao sujeito que o objetivo deste estudo não é avaliar o participante, mas sim avaliar como o desenvolvimento de software acontece(u) na prática;
- Deixar claro ao sujeito que os dados coletados na entrevista serão usados unicamente na pesquisa. Assegurar a ele o anonimato das informações que dará e de seu nome. As análises das informações serão agregadas em estudos de caso, separados por organizações e projetos. As falas usadas na escrita deste estudo serão feitas com um nome fictício;
- Comunicar ao sujeito que a entrevista segue um roteiro semiestruturado que foi elaborado de acordo com referencial teórico pesquisado;
- Solicitar permissão ao sujeito para prosseguir com a entrevista através da assinatura do “Termo de Consentimento Livre e Esclarecido”, descrito no APÊNDICE C;
- Solicitar permissão ao sujeito para gravar a entrevista informando que a gravação somente será ouvida pelo pesquisador (aluno), e eventualmente pelo orientador acadêmico;
- Agradecer os sujeitos pela sua importante contribuição para os resultados desta pesquisa.

Em média as entrevistas duraram cerca de 34 minutos, sendo a mais longa com 54 minutos de duração e a mais curta com 16 minutos de duração. Embora de duração curta, não se observou, em nenhum momento, a indisposição da parte dos sujeitos de continuarem um diálogo. Além de as entrevistas terem sido cercadas por um clima de respeito mútuo e compreensão, as falas dos entrevistados foram marcadas pela sinceridade e disposição em contribuir com os objetivos desta pesquisa. Isto favoreceu substancialmente o acesso a outras informações que interessavam à pesquisa, especialmente documentos de uso interno da organização e dos projetos.

4.1.4 O Panorama dos Sujeitos da Pesquisa

Com o objetivo de manter o sigilo dos depoimentos, os 18 sujeitos entrevistados foram identificados com nomes fictícios. Na sequência, o panorama dos sujeitos da pesquisa é apresentado, conforme a Tabela 14.

No CNPTIA, a amostra foi composta por 6 sujeitos do gênero masculino. Um sujeito fez parte da amostra da dimensão organizacional e os outros 5 sujeitos compuseram a amostra da dimensão de projeto.

Em termos de tempo de serviço no CNPTIA os sujeitos possuem em média 13 anos. Além disso, considerando a experiência profissional com desenvolvimento de software, com vínculo empregatício, em outras empresas que não a Embrapa, registrou-se que dos 6 sujeitos apenas 2 atuaram por mais tempo como profissionais em outras empresas do que como empregados da Embrapa. Vale salientar que os sujeitos possuem em média 7 anos de experiência com métodos ágeis, sendo que todos eles adquiriram o conhecimento e a experiência sobre o assunto na própria Empresa.

Considerando a função na Empresa ou papel no projeto, os sujeitos estão distribuídos da seguinte maneira no CNPTIA: 1 sujeito é assessor da chefia, 2 são gerentes de projetos e 3 são desenvolvedores de software.

Quanto à formação acadêmica, dentre os 6 sujeitos da Embrapa apenas 2 não possuem pós-graduação *stricto sensu* em nível de mestrado ou doutorado, mas, apesar disto, eles desenvolvem suas atividades de pesquisa e/ou desenvolvimento normalmente. Dentre as universidades que contribuíram para formar os sujeitos estão a Pontifícia Universidade Católica de Campinas (PUC-Campinas), a Universidade Estadual de Campinas (Unicamp), a Université Blaise Pascal (Clermont-Ferrand / França) e o Instituto Nacional de Pesquisas Espaciais (INPE).

Na PROCERGS, a amostra foi composta por 12 sujeitos, sendo 8 sujeitos do gênero masculino e 4 sujeitos do gênero feminino. Um sujeito do gênero feminino fez parte da amostra da dimensão organizacional e os outros 11 sujeitos fizeram parte da dimensão de projeto.

Dentre os 12 sujeitos, 3 são recém contratados com menos de um ano de PROCERGS. Além disso, os sujeitos possuem, em média, 7 anos de Empresa, bem como, 12 anos em média de experiência no desenvolvimento de software e 2 anos em média de experiência com métodos ágeis. Vale ressaltar que 7 dos 12 sujeitos já possuíam conhecimento em desenvolvimento ágil antes de pertencerem ao quadro de empregados da PROCERGS.

Considerando a função na Empresa ou papel no projeto, os sujeitos estão distribuídos da seguinte maneira na PROCERGS: 1 sujeito é chefe de setor, 1 é *Product Owner*, 3 são desenvolvedores de software, 2 são analistas de testes, 2 são analistas de sistemas. Além disso, 2 sujeitos exercem o papel de *ScrumMaster* / Desenvolvedor simultaneamente.

Quanto à formação acadêmica, dentre os 12 sujeitos da PROCERGS apenas 2 não concluíram o ensino superior, estando em fase de conclusão. Um sujeito possui 2 cursos de pós-graduação *lato sensu* e os outros sujeitos possuem nível superior completo. Dentre as universidades que contribuíram para formar os sujeitos estão a Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), a Universidade Federal do Rio Grande do Sul (UFRGS), a Universidade de Santa Cruz do Sul (UNISC), a Universidade do Vale do Rio dos Sinos (Unisinos), o Centro Universitário Internacional (Uninter), a Universidade de Passo Fundo (UPF), a Universidade Federal de Pelotas (UFPEL), o Centro Universitário La Salle (Unilasalle), a Universidade Luterana do Brasil (Ulbra) e a Faculdade Cenecista Nossa Senhora dos Anjos (Facensa).

Tabela 14 - Panorama dos sujeitos da pesquisa.

Org. Pública	ID	Dim.	Sexo	Tempo de Empresa	Tempo de Experiência com Des. de Software	Tempo de Experiência com Métodos Ágeis	Formação Acadêmica	Função ou Papel no Projeto
Embrapa Informática Agropecuária	1	Proj.	Masc.	4 meses	14 anos	4 meses	Superior Completo	Desenvolvedor de Software
	2	Proj.	Masc.	8 anos	15 anos	3 anos	Doutorado	Gerente de Projeto
	3	Proj.	Masc.	20 anos	20 anos	2 anos	Doutorado	Desenvolvedor de Software
	4	Org.	Masc.	20 anos	25 anos	14 anos	Mestrado	Assessor da Chefia
	5	Proj.	Masc.	20 anos	25 anos	14 anos	Mestrado	Gerente de Projeto
	6	Proj.	Masc.	12 anos	18 anos	8 anos	Superior Completo	Desenvolvedor de Software
Companhia de Processamento de Dados do Rio Grande do Sul	7	Org.	Fem.	17 anos	17 anos	2 anos	Superior Completo	Chefe de Setor
	8	Proj.	Masc.	1 ano	4 anos	1 ano e meio	Superior Incompleto	*ScrumMaster* e Desenvolvedor de Software
	9	Proj.	Masc.	1 mês	8 anos	1 ano e meio	Superior Incompleto	Desenvolvedor de Software
	10	Proj.	Fem.	1 ano e meio	6 anos	3 anos	Superior Completo	Desenvolvedor de Software
	11	Proj.	Masc.	4 meses	2 anos	2 anos	Superior Completo	Analista de Qualidade
	12	Proj.	Fem.	20 anos	30 anos	3 anos	Superior Completo	Analista de Sistemas
	13	Proj.	Masc.	20 anos	22 anos	2 anos	Superior Completo	*Product Owner*
	14	Proj.	Masc.	2 anos e meio	8 anos	3 anos	MBA e Especialização	*ScrumMaster* e Desenvolvedor de Software
	15	Proj.	Masc.	3 meses	12 anos	3 meses	Superior Completo	Desenvolvedor de Software
	16	Proj.	Masc.	2 anos	11 anos	4 anos	Superior Completo	Desenvolvedor de Software
	17	Proj.	Fem.	13 anos	18 anos	2 anos	Superior Completo	Analista de Sistemas
	18	Proj.	Masc.	1 ano	5 anos	2 anos e meio	Superior Completo	Analista de Qualidade

4.1.5 O Tratamento e a Análise dos Dados

Godoy [God95] afirma que o desenvolvimento da perspectiva qualitativa gerou uma grande diversidade de métodos de trabalho, estilos de análise e a apresentação de resultados e diferentes considerações quanto aos sujeitos, tendo, porém, como técnica mais usual, a análise de conteúdo. Silva [SGS05] argumenta que, a aplicação da técnica de análise de conteúdo nas ciências sociais apresenta-se como uma ferramenta útil à interpretação das percepções dos atores sociais. O papel de interpretação da realidade social configura ao método de análise de conteúdo um importante papel como ferramenta de análise na pesquisa qualitativa nas ciências sociais aplicadas. Ele indica ainda que não há, no método, um esquema rígido de utilização e que o pesquisador pode e deve utilizar esta flexibilidade, entretanto permanece o compromisso de imprimir nitidez ao quadro teórico e a postura metodológica. Para isso, a precisão com que o investigador captou o ponto de vista dos participantes, testando-o junto aos próprios informantes ou confrontando sua percepção com a de outros pesquisadores, deve ser assegurada [Hof13]. Nessa direção, o tratamento e a análise de dados desta pesquisa inspirou-se na proposta de Bardin [Bar11], a qual prevê três fases fundamentais: (1) pré-análise, (2) exploração do material e (3) tratamento dos resultados - inferência e interpretação.

Um traço distinto desta pesquisa foi que a coleta de dados e a condução das entrevistas aconteceram simultaneamente à escrita dos estudos de caso. Não se esperou finalizar ou ter realizado a maior parte das entrevistas para enfrentar o laborioso trabalho de conversão do áudio em texto e tratamento dos resultados - inferência e interpretação - sendo fundamental que tais processos acontecessem simultaneamente. A Pré-análise incluiu a conversão do áudio das entrevistas em texto consubstanciando no documento “Texto das Entrevistas”. Ela também ajudou a definir os temas a serem tratados, sendo organizados em algumas categorias sínteses de informação, incluindo informações sobre o projeto, a equipe, o relacionamento com o cliente e com o negócio, o processo e as práticas, o conhecimento e a experiência dos sujeitos. Vale salientar que essas categorias derivaram-se principalmente do referencial teórico e uma única categoria foi criada após a coleta de dados (o conhecimento e a experiência dos sujeitos). Além disso, apesar do principal instrumento de coleta de dados ter sido a entrevista semiestruturada, a análise completa não se limitou aos dados provenientes deste único instrumento, mas também utilizou-se de documentos internos e externos ao projeto, conforme apresentado anteriormente na Tabela 12.

A segunda etapa de tratamento e análise de dados que foi executada consistiu na exploração do material e elaboração de um “Conjunto de Evidências” extraídas da leitura e interpretação do documento “Texto das Entrevistas” produzido na etapa anterior (Pré-análise). As evidências representam o grau de proximidade conceitual encontrado na leitura e interpretação dos dados, ou seja, os temas que se repetiram com certa frequência foram recortados do texto e agrupados. Por exemplo: alguns sujeitos de um determinado projeto falaram que nem sempre é possível garantir que todas as expectativas serão atendidas em tempo oportuno, o que produz algum tipo de frustração. Estas falas foram consideradas como uma descoberta descritiva a qual denominou-se evidência. Nelas percebeu-se que estavam apontando a compreensão que tinham sobre motivos de insatisfação no projeto. Vale ressaltar que em todo o processo de construção do “Conjunto de Evidências”, procurou-se preservar a fala do entrevistado.

Partindo desta racionalidade a terceira fase do processo de análise do conteúdo denominada tratamento dos resultados - inferência e interpretação aconteceu sob a lógica consignada na fala dos sujeitos e o “Conjunto de Evidências”, sendo redigidas na forma de estudos de caso procurando dar significado e validade aos dados brutos coletados. Vale salientar que para análise da dimensão organizacional foi necessário consignar as falas individuais dos sujeitos - acerca de suas percepções sobre métodos ágeis e o futuro do desenvolvimento de software na organização - com a fala do respondente das questões organizacionais.

4.2 Embrapa Informática Agropecuária: Estudos de Caso

Na Embrapa Informática Agropecuária, dois projetos recentes apresentaram evidências no uso de métodos e/ou práticas ágeis para o desenvolvimento de software. Estes dois projetos compõem as unidades de análise, sendo apresentados a seguir.

4.2.1 Caso 1

Uma série de contratempos afligiu a equipe de desenvolvimento que estava construindo o software “A”, entre os anos de 2012 e 2014. Pouco ou nenhum conhecimento sobre as metodologias e tecnologias adotadas no projeto eram evidentes. Embora representativas, equipes maiores de negócio pareciam tomar mais tempo na tomada de decisão. Outros problemas haviam aparecido e somados a esses, aumentaram o tempo de desenvolvimento do software “A”. Por outro lado, foi impressionante notar como as pessoas de negócio e TI continuavam engajadas na construção do sistema. Estas e outras informações encontradas no Caso 1 são apresentadas na sequência.

4.2.1.1 Informações sobre o Projeto

O desenvolvimento do Sistema A é parte integrante do Projeto 1, que tem como objetivo principal estabelecer um processo sistematizado e informatizado para o gerenciamento centralizado de dados experimentais na Embrapa. O Sistema A é composto por três módulos de software: (1) módulo *web*; (2) módulo de integração com dispositivos eletrônicos; (3) módulo de coletores eletrônicos de dados, sendo o módulo *web* o mais importante deles.

Além dos aspectos tecnológicos, o projeto apresenta desafios de processo (tais como, padronizar o fluxo e o registro de informações), bem como, desafios jurídicos (tais como, estabelecer políticas adequadas para a propriedade, acesso e uso dos dados armazenados). O projeto contempla 12 Unidades de Produto da Embrapa, afetando diretamente a forma com que a Embrapa executa suas pesquisas, bem como, impactando na forma de atuação de seus parceiros em projetos de pesquisa. Ele prevê um orçamento de aproximadamente R$1.100.000,00, incluindo a aquisição de equipamentos de informática (servidores, *desktops,* coletores de dados, etc.) e despesas com *workshops*, capacitações para equipe técnica e implantação do sistema, etc.

A previsão de encerramento do projeto é para o primeiro trimestre de 2015, totalizando 3 anos de execução. Vale salientar, que o papel da Embrapa Informática Agropecuária neste projeto é gerenciar as dimensões técnica, administrativa e financeira do projeto, tal como, executar o desenvolvimento de software do módulo *web*, o qual está sendo realizado por empregados da própria Unidade. Além disso, até o momento, não está prevista a distribuição do software para outras OP que compõem o Sistema Nacional de Pesquisa Agropecuária (SNPA), uma vez que ele precisa primeiro ser validado e testado na própria Empresa.

4.2.1.2 Informações sobre a Equipe

A equipe do projeto é formada por 12 Unidades de Produto e duas Unidades de TI. As Unidades de Produto estão localizadas em diferentes cidades brasileiras, abrangendo desde a região Sul até a região Nordeste. Enquanto, a Unidade responsável pelo desenvolvimento do software “A” está localizada na cidade de Campinas no Estado de São Paulo.

Vale ressaltar que em cada Unidade de Produto há uma Comissão Local (CL), a qual é responsável por representar os interesses de seus pesquisadores. As CLs (dono do produto) são responsáveis por determinar “o que deve ser feito”. Além disso, em cada CL há, pelo menos, um ponto focal de TI e um ponto focal de negócio, o que diminui a

complexidade na comunicação do projeto. O tamanho das CLs é variável (de 4 a 17 pessoas). Em setembro de 2014, o projeto era composto por aproximadamente 120 pessoas, envolvendo pessoal de TI e negócio, sendo todos eles empregados da Empresa. Já, a equipe de desenvolvimento do software “A” é formada por 3 pessoas, incluindo 2 desenvolvedores e 1 gerente de projeto, sendo separados por sala e andar. Essas 3 pessoas não tinham trabalhadas juntas em um mesmo projeto.

A comunicação entre os desenvolvedores tem acontecido, preferencialmente, por meio de conversa frente a frente, com a ida dos desenvolvedores nas salas uns dos outros. O relacionamento entre eles é bem tranquilo e colaborativo, conforme mostrou as falas de Shun e Ikki:

> *“Na minha opinião, a relação de trabalho entre os desenvolvedores é bem tranquila e colaborativa desde o começo do projeto, principalmente em momentos de dificuldades no desenvolvimento.” (Shun, 15/09/2014, dp)*
>
> *“Somente com a entrada do novo empregado é que a equipe ficou com duas pessoas. Acredito que a nossa relação de trabalho é bem tranquila.” (Ikki, 17/09/2014, dp)*

4.2.1.3 Informações sobre o Relacionamento com o Cliente e com o Negócio

A distância geográfica é uma das variáveis inerentes do projeto. Os desenvolvedores estão colocalizados, mas separados por sala e andar. Enquanto, as CLs estão dispersas geograficamente uma das outras. Também, há uma distância entre os membros da CL (dono do produto) e seus pesquisadores (*stakeholders*), sendo separados por sala e andar.

Outra variável inerente do projeto é o trabalho compartilhado com outras atividades. Um dos desenvolvedores do produto de software exerce atividades em outros projetos. Os pontos focais de TI das CLs são responsáveis por outras atividades de TI em suas Unidades. Os pesquisadores exercem atividades de campo e participam de eventos técnico-científicos de suas áreas.

O resultado da somatória dessas variáveis tem sido uma dificuldade em estabelecer uma agenda mais intensa em que todas as pessoas possam participar ao mesmo tempo, bem como, uma dificuldade em garantir que as pessoas certas estejam disponíveis no momento em que elas são requisitadas. Uma boa alternativa para o problema da agenda tem sido pré-agendar, de maneira sistemática, reuniões para a definição e consolidação dos requisitos do sistema em um único dia da semana (sexta-feira). Dessa maneira, as pessoas envolvidas nessa atividade conseguem “bloquear” suas agendas, quando possível, para este dia.

Entretanto, o problema do trabalho compartilhado é uma característica inerente de toda a Empresa, ficando sem solução. Vale salientar que todas as reuniões são realizadas por meio de videoconferências, as quais são gravadas e disponibilizadas no ambiente colaborativo do projeto, podendo ser vistas posteriormente para esclarecer o que foi tratado para os membros da equipe que não conseguiram visualizar a videoconferência com clareza, não estiveram presentes no dia ou precisaram de esclarecimento sobre o conteúdo exato do que foi discutido. Além disso, o ambiente colaborativo é utilizado para o compartilhamento de informações sobre “o que está acontecendo no projeto”, incluindo informações sobre os membros do projeto, sobre os requisitos do sistema, sobre “o que está pronto” no sistema para validação, etc.

Sobre o relacionamento com o cliente e com o negócio, a fala de Shiryu mostrou que o papel da Unidade de TI era, além de desenvolver o software propriamente dito, orientar e ajudar os CLs a criarem um objetivo comum, bem como, facilitar a tomada de decisão. Essa abordagem visava impedir que reuniões sem fim e com interesses conflitantes acontecessem sem que nenhum progresso real fosse alcançado. Em uma única situação, a intervenção da Unidade de TI foi fundamental para garantir a estabilidade do projeto, conforme relatou Shiryu:

> *“Algumas discussões são polarizadas e tornam-se prolongadas. Em um caso específico, foram seis meses de “discussão” em torno de um requisito importante. Haviam dois lados opostos com considerações fundamentadas. Como ambos os lados não conseguiram chegar a um consenso sobre o assunto; então, uma boa alternativa foi implementar uma solução intermediária capaz de satisfazer os dois lados.” (Shiryu, 15/09/2014, dp)*

Ainda sobre o relacionamento com o cliente e com o negócio, nem sempre é possível garantir que todas as expectativas serão atendidas em tempo oportuno, o que produz algum tipo de frustração:

> *“Sobre a insatisfação, existem várias linhas de pensamento sobre “como o sistema deve funcionar”. Quando as decisões de negócio favorecem uma determinada linha de pensamento, outras linhas de pensamento não são atendidas; isso gera um pouco de insatisfação.” (Ikki, 17/09/2014, dp)*

> *“...a insatisfação acontece quando determinadas sugestões de melhorias ainda não são contempladas na versão apresentada.” (Shiryu, 15/09/2014, dp)*

Também, aconteceram alguns problemas na comunicação entre os profissionais de TI e os profissionais de negócio, conforme relatou Shiryu:

> *"Há um anseio por parte dos desenvolvedores em obter rapidamente os requisitos necessários para as atividades de codificação. Porém, ainda existem desafios em adaptar a linguagem dos desenvolvedores para a linguagem do negócio. Em algumas reuniões, alguns membros da área técnica de TI estavam dialogando com os especialistas do domínio utilizando termos da Computação. Em várias situações, procurei intervir e solicitei para os membros técnicos de TI refazerem suas perguntas e considerações utilizando o vocabulário do negócio." (Shiryu, 15/09/2014, dp)*

4.2.1.4 Informações sobre o Processo e as Práticas

Inicialmente, foram identificados aproximadamente 80 requisitos, sendo escritos em uma visão de alto nível, sem qualquer detalhamento de implementação. Vale salientar que alguns deles foram obtidos por meio de um protótipo não funcional de software criado anteriormente. Ou seja, nessa etapa procurou-se descobrir "quais eram os requisitos do sistema".

O planejamento inicial definia a divisão desses requisitos em 12 iterações de seis semanas, com um tempo de 45 dias para cada iteração, sendo necessário duas semanas para o detalhamento do requisito, mais duas semanas para o desenvolvimento de todos os requisitos detalhados anteriormente, mais duas semanas de validação do sistema e mais 3 dias para homologação da versão. Assim, o desenvolvimento baseou-se no modelo de desenvolvimento iterativo e incremental e incluiu algumas práticas e artefatos ágeis (como *backlog* do produto e incrementos de software).

Porém, na prática, quando os requisitos "chegaram" no desenvolvimento alguns problemas surgiram, o que acarretou atrasos no cronograma: (1) alguns requisitos eram dependentes de integração com outros sistemas da Empresa; (2) a maioria dos requisitos, embora detalhados, não continham todas as informações necessárias para o seu desenvolvimento, sendo necessário consultar o gerente de projeto, ou então, o dono do produto para obter maiores informações; (3) alguns requisitos eram uma "mini-aplicação" com algumas questões não resolvidas, incluindo questões de segurança e controle de acesso. Assim, estabelecer iterações curtas com tempo fixo de entrega não foi possível, sendo necessário pensar em uma nova abordagem para o projeto, conforme relatou Shun:

> *"(...) apontei para a gerência do projeto que a divisão inicial de todos os requisitos em iterações e o tempo necessário para implementá-los falharam. A realidade mostra que os requisitos são complexos e difíceis de implementar, o que leva a um tempo muito maior para sua conclusão, inviabilizando o cronograma inicial." (Shun, 15/09/2014, dp)*

Uma boa solução para esse problema foi encontrada na fala de Shun e Ikki, onde a abordagem escolhida faz com que as pessoas validem novos conhecimentos, constatem acertos e erros de concepções com base na experimentação do sistema, o que é entendido como relevante, já que serve de instrumento potencializador da aprendizagem e apropriação do conhecimento:

> *"Para resolver esse problema, uma boa alternativa tem sido o desenvolvedor implementar o requisito com as informações obtidas até o momento; na sequência, a funcionalidade implementada, além de apresentada para o gerente de projeto, é mostrada à um usuário-chave para obter feedback sobre seu funcionamento, sendo esse ciclo repetido quantas vezes for necessário, até que a implementação da funcionalidade alcance uma certa "aceitação" do usuário-chave. Basicamente, a ideia é ajustar o funcionamento do sistema com base no feedback de alguns usuários-chave durante a iteração, antes de sua validação completa com todas as Unidades envolvidas no projeto". (Shun, 15/09/2014, dp)*

> *"Para resolver esse problema, nós implementamos os requisitos com base no nosso entendimento e apresentamos para o cliente. Na sequência, quando o cliente visualiza o software funcionando; ele nos dá um feedback (pelo menos uma vez por mês) sobre se estamos indo na direção certa ou não." (Ikki, 17/09/2014, dp)*

Embora que não foi possível determinar precisamente quando uma iteração ficaria pronta, a abordagem de desenvolvimento adotada contribuiu para o projeto alcançar alguns resultados positivos, conforme observado na fala de Shiryu e Ikki:

> *"A entrega do sistema em iterações contribuiu para manter todos os membros do projeto comprometidos com os resultados. É impressionante observar que após dois anos de projeto, todas as Unidades envolvidas e seus membros continuam comprometidos com o projeto. Certamente, as entregas incrementais, bem como, o desenvolvimento colaborativo com o cliente contribui para isso; senão fosse assim, acredito que muitos já teriam abandonado o projeto há muito tempo". (Shiryu, 15/09/2014, dp)*

> *"Apesar do atraso no cronograma, bem como, apesar das dificuldades em manter um contato mais frequente e próximo do cliente, eles continuam bem ativos no projeto. Assim, trabalhar com o cliente é importante para que isso aconteça." (Ikki, 17/09/2014, dp)*

> *"Eu senti que na última iteração, quando os usuários começaram a criar alguma coisa mais interessante no sistema, eles ficaram mais satisfeitos. Assim, o projeto está virando realidade; o sistema está funcionando." (Ikki, 17/09/2014, dp)*

> *“A satisfação acontece quando os membros dos projetos conseguem visualizar o funcionamento do software, que é o fruto do seu trabalho. A satisfação é muito clara quando novas versões são disponibilizadas.” (Shiryu, 15/09/2014, dp)*

Assim, a partir das concepções trazidas pelos sujeitos, o presente relato oferece indícios importantes para referendar que o desenvolvimento de software executado com base no modelo iterativo e incremental contribui para manter o interesse das pessoas e a credibilidade no projeto “vivos” durante toda sua execução. Por outro lado, as falas também anunciam que, mais do que alguns resultados promissores alcançados no projeto, os sujeitos também incorporaram neste aprendizado problemas, desafios e lições a aprender:

> *“Algo que precisa ser repensando no uso dessa abordagem de desenvolvimento são as questões de arquitetura do sistema. Mais informações são necessárias para definir a arquitetura do sistema, antes de começar a desenvolver o software. Atualmente, a principal reclamação do cliente está relacionada com a performance (tempo de resposta) do sistema; esse é o tipo de problema que deveria ter sido tratado desde o início do projeto (e não foi); deixá-lo para depois, torna mais difícil a sua correção. A falta de domínio nas tecnologias utilizadas no projeto também contribui para aumentar esse problema.” (Shun, 15/09/2014, dp)*

> *“Eu acredito que para métodos ágeis funcionar deve existir uma homogeneidade de conhecimento na equipe, principalmente relacionado ao conhecimento técnico da equipe em codificação. Como são várias pessoas que podem modificar o código-fonte; se não houver consciência sobre a importância da arquitetura do sistema, padrões de código e modelos de dados; então futuras implementações serão mais difíceis de se fazer.” (Ikki, 17/09/2014, dp)*

> *“Métodos ágeis oferecem pouco ou nenhuma informação de como documentar os processos de negócio para que novos usuários assimilem “como o processo de negócio é executado”. Estamos vislumbrando a possibilidade de escrever um livro sobre o assunto, detalhando porque determinadas decisões foram tomadas. Esse é um problema latente, e não sabemos como resolvê-lo. Também, ainda não sabemos quais documentos são necessários (e se realmente são necessários) para fins de auditoria em empresas públicas brasileiras”. (Shiryu, 15/09/2014, dp)*

4.2.1.5 Informações sobre o Conhecimento e a Experiência em Métodos Ágeis

Ninguém no projeto havia desenvolvido um software com as metodologias e tecnologias adotadas. Por outro lado, práticas ágeis e *Java Enterprise Edition* (Java EE) pareciam ser abordagens promissoras. Alguns treinamentos foram organizados e algumas pessoas de TI foram capacitadas em Scrum e Java EE, o que adicionou mais tempo ao

projeto devido ao aprendizado das novas tecnologias e metodologias. Assim, não houve tempo suficiente para as pessoas assimilarem as vantagens e desvantagens dessas abordagens, como também, não foi possível que outras pessoas com mais experiência nessas tecnologias e metodologias fossem incorporadas ao projeto. Ikki e Shun, expressaram suas considerações sobre a falta de conhecimento e experiência nas tecnologias e metodologias utilizadas no projeto:

> *"Inicialmente, não conhecíamos profundamente todos os elementos das tecnologias (Java EE) utilizadas no desenvolvimento do sistema. Quando problemas técnicos aparecem não conseguimos resolvê-los rapidamente. Com isso, muitas vezes, não conseguimos cumprir os prazos das iterações. O que fizemos foi prolongar o tempo de desenvolvimento das iterações com base no que tem acontecido." (Ikki, 17/09/2014, dp)*

> *"Eu nunca trabalhei com métodos ágeis antes, mas estou adaptando-me a eles. Estou procurando mudar a minha maneira de pensar. Além disso, o aprendizado das tecnologias associadas ao projeto introduziu um desafio pessoal enorme; ainda estou aprendendo sobre elas. Então, produzir algo sem dominar as metodologias e as tecnologias associadas ao projeto é um problema." (Shun, 15/09/2014, dp)*

4.2.2 Caso 2

Por trás de todo grande produto existe uma ou mais pessoas com uma grande empatia pelo cliente, uma percepção sobre o que é possível realizar separando o que é essencial e o que é supérfluo. Essas pessoas atuam sobre uma profunda relação de confiança entre o cliente e os desenvolvedores, pensando em termos de entrega de valor ao cliente e definindo bons produtos que podem ser criados. Estas e outras informações encontradas no Caso 2 são apresentadas na sequência.

4.2.2.1 Informações sobre o Projeto

O produto "B" é um software para o gerenciamento de bibliotecas, desenvolvido pela Embrapa Informática Agropecuária, cuja primeira versão foi iniciada em 1991 e desde esta data encontra-se em constante evolução. É utilizado principalmente, pelas bibliotecas da Embrapa com o objetivo de organizar, preservar e difundir o acervo das bibliotecas e os documentos gerados pelas atividades de Pesquisa e Desenvolvimento e Inovação (PD&I) da Empresa. O planejamento inicial do Projeto 2 foi o de evoluir o software "B" de uma arquitetura proprietária cliente/servidor (descentralizada) para uma versão melhor e mais moderna para *web* (centralizada) com base em software livre, incluindo recursos de atualização de dados, integração, recuperação e visualização das informações constantes nas bibliotecas do Sistema Embrapa de Bibliotecas (SEB) em tempo real a partir de

qualquer uma das bibliotecas e evoluir o sistema de busca, com nova interface, mais interativa e com novos e mais sofisticados recursos de busca e implementação de opções de navegação. Na versão anterior, o Sistema B operava individualmente em cada uma das bibliotecas gerenciando individualmente seus respectivos acervos. Isto demandava equipamento para hospedar o software, assim como recursos humanos do setor de infraestrutura de cada unidade para mantê-lo em operação. Além disso, a cada atualização de revisão ou versão, era necessário garantir que todas as unidades renovassem suas versões. Com este projeto, e a construção da nova versão, não há mais a necessidade de qualquer tarefa de infraestrutura das unidades, assim como softwares e equipamentos específicos para instalação e configuração do software “B”. Além disso o gerenciamento de versões passou a ser facilitado, assim como todo o acervo da Embrapa foi integrado.

Durante aproximadamente 7 anos de execução, que compreende o último trimestre de 2005 até o terceiro trimestre de 2013, o Projeto 2 consumiu aproximadamente R$ 600.000,00, incluindo gastos em: aquisição de equipamentos de informática (abrangendo servidores, computadores para desenvolvimento, computadores para todas as bibliotecas da Embrapa, impressoras, *scanners* de pequeno porte e um *scanner* de grande porte), contratação de estagiários, despesas com treinamento e implantação do novo sistema, digitalização do acervo documental, etc.

Por outro lado, considerando que o custo para manter uma instalação local da versão anterior estaria em torno de R$ 4.000,00 por ano, estima-se, em valores, que a nova versão do software “B”, tem gerado, desde 2010, uma economia anual para a Embrapa de algo em torno de R$ 160.000,00. Então, do ponto de vista econômico, a migração do sistema mostrou-se viável. Já do ponto de vista de negócio, os resultados do Projeto 2 também têm contribuído para manter o portal da Embrapa como o melhor classificado entre os centros de pesquisa da América Latina na edição de julho de 2014 do Ranking Mundial Web de Centros de Pesquisa, elaborado pelo Conselho Superior de Pesquisas Científicas da Espanha (CSIC)[5]. Além disso, a consulta ao módulo externo do sistema tem ultrapassado a marca de 1.000.000 de acessos mensais.

De um modo geral, as mudanças realizadas favoreceram o acesso às informações da Embrapa de maior utilidade para o cidadão, como publicações técnico-científicas para *download* gratuito. Nesse último, um dos repositórios frutos do Projeto 2, que traz informações científicas produzidas por pesquisadores da Embrapa, ocupou o 6º lugar na América Latina em 2014[6].

[5] http://research.webometrics.info/en/World
[6] http://repositories.webometrics.info/en/Americas/Latin_America

Vale salientar que o software “B” foi licenciado sob os termos específicos de uma licença da Embrapa, sendo disponibilizado gratuitamente para organizações de pesquisa. Segundo informações da equipe do projeto, além da própria Embrapa, atualmente a nova versão do software “B” está sendo utilizado por mais 15 organizações, sendo uma delas no exterior (*Instituto Nacional de Investigación Agropecuaria* - INIA do Uruguai).

4.2.2.2 Informações sobre a Equipe

Em 2006, o projeto era formado por uma equipe de desenvolvimento com 2 empregados e mais 2 ou 3 estagiários; 1 gerente de projeto; e uma Comissão Permanente (CP) com 6 ou 7 empregados de algumas Unidades da Empresa, sendo responsável por representar os interesses da organização, dos usuários e da sociedade. Mais tarde, no segundo semestre de 2008 o(a) presidente da CP foi substituído(a), bem como, no primeiro mês de 2009 o gerente de projeto deixou a equipe.

Os desenvolvedores e o gerente de projeto que eram empregados da Embrapa Informática Agropecuária estavam localizados em salas individuais no mesmo andar. Enquanto, os estagiários estavam localizados em um ambiente muito próximo das salas dos empregados. De acordo com a fala de Alexer, isso também foi importante para o bom andamento do projeto:

> *“A equipe conseguia fazer reuniões rapidamente; após uma reforma entre 2006 e 2007, nós ficamos com um laboratório quase com a configuração ideal para aplicar XP, com uma mesa para realizar reuniões rápidas, com um quadro branco, com salas para os empregados comunicantes, bem como, com o espaço dos estagiários comunicantes com as salas dos empregados desenvolvedores do sistema. Então, nós tínhamos uma proximidade entre todos os desenvolvedores, um ambiente favorável para fazer reuniões rápidas e integração entre a equipe. Certamente isso colaborou para o bom andamento do projeto.” (Alexer, 23/09/2014, dp)*

Sobre a participação de estagiários de graduação no processo de desenvolvimento do software, Alexer destacou que:

> *“A gente tinha uma filosofia que o estagiário, embora estivesse no projeto, ele era supervisionado por um empregado. Aquilo que estava sendo negociado entre o empregado supervisor e seu estagiário, os demais não iriam “invadir” esse espaço para solicitar algo diferente do que estava sendo combinado, o que poderia gerar ingerência e consequentemente conflitos entre os empregados. Então, a gente conversava o que precisava ser feito no projeto e isso desencadeava algumas atividades para os estagiários. E ambiente de estágio é um ambiente universitário. Felizmente, tivemos uma boa “safra” de estagiários.” (Alexer, 23/09/2014, dp)*

> *“Se por um lado, exigiu mais tempo nosso de capacitação e treinamento, por outro, desempenharam atividades que certamente ajudaram na evolução do sistema.” (Alexer, 23/09/2014, dp)*

Vale salientar que embora houvesse uma distância nacional entre os membros da CP e o(a) presidente da CP, essa distância não acontecia entre o(a) presidente da CP e a equipe de desenvolvimento, pelo menos até a metade de 2008, onde ambos estavam localizados na mesma Unidade da Empresa. Assim, durante aproximadamente 3 anos o projeto teve um cliente residente (presidente da CP). Mais tarde, no segundo semestre de 2008 com a substituição do(a) presidente da CP, a distância entre eles passou a ser de aproximadamente 20 quilômetros, que na fala de Cassios poderia ser um problema:

> *“Não tenho nenhum problema a destacar (...) talvez, a única questão é que o dono do produto não está mais fisicamente aqui, e ele mesmo gostaria de se transferir para cá, o que não foi possível.” (Cassios, 23/09/2014, dp)*

Além disso, o(a) presidente da CP e os desenvolvedores tinham disponibilidade de tempo quase que integral ao projeto. Mais tarde, em 2010, um dos desenvolvedores começou a exercer atividades em outros projetos da empresa de maneira simultânea. Por último, com exceção de Cassios e Saori que já trabalhavam juntos há cerca de 3 anos, essa foi a primeira experiência de trabalho juntos para os demais membros da equipe.

4.2.2.3 Informações sobre o Relacionamento com o Cliente e com o Negócio

Alexer trabalhou no Projeto 2 entre 2005 e 2008, sua missão inicial, enquanto gerente de projeto, era restaurar a confiança e a crença nos desenvolvedores, pelo menos foi o que aparentou sua fala:

> *“Havia problemas de relacionamento do Cassios com a Saori, e eu entrei no projeto para tentar acertar essa relação; pelo menos, minimizar os efeitos negativos dessa relação, sendo que naquele momento não existia uma relação de confiança entre eles. De certa maneira, o Cassios tinha uma sobrecarga de algumas coisas, porém o resultado dessa relação não muito boa foi uma quebra total da confiança, que gerou uma série de problemas.” (Alexer, 23/09/2014, dp)*

Por causa do nível de desconfiança e de atritos no relacionamento, Alexer optou por banir temporariamente o relacionamento direto da equipe de desenvolvimento com o dono do produto. A estratégia foi utilizar a comunicação por escrito para explicar melhor as mensagens verbais e evitar atritos, conforme relatou Alexer:

> *“(...) a estratégia adotada foi primeiramente intervir para não ter mais a relação direta; tudo deveria passar pelo gerente de projeto para a qual eu fui designado. Em segundo*

> *lugar, passamos a formalizar as demandas do dono do produto, que até então, eram feitas por telefone ou e-mail; passamos a utilizar uma ferramenta para gerir essas demandas, que naquela época era o GForge. Essa estratégia se demonstrou eficiente, porque cortando a relação com o dono do produto e colocando uma ferramenta, então ajudou a minimizar esses atritos, porém o mais importante era conquistar novamente a confiança do dono do produto. Quando isso acontecesse as coisas passariam a fluir bem melhor." (Alexer, 23/09/2014, dp)*

Mesmo sabendo do desafio que tinham diante de si, construir a confiança era necessário para a equipe de desenvolvimento, a qual foi alcançada por meio da demonstração de resultados favoráveis, conforme mostrou a fala de Alexer:

> *"A satisfação aconteceu quando foi publicada a nova versão da interface de busca externa do sistema, que incluiu recursos mais modernos e rápidos de recuperação de informação, bem como, uma interface mais amigável quando comparado com a versão anterior. E um ponto que foi fundamental, a redução considerável do tempo do registro de uma informação e sua disponibilização para consulta externa, antes a cada seis meses, depois uma vez por semana e agora diária. Então, veja, conseguimos criar um novo sistema de recuperação de informação muito mais eficiente, muito mais rápido, mais preciso e com informações atualizadas. Obviamente isso dá uma satisfação maior para o cliente e seus usuários. Internamente, houve uma satisfação para a própria equipe de desenvolvimento porque rapidamente conseguimos colocar um novo sistema no "ar" que o cliente estava satisfeito, começou a usar e dar "críticas" construtivas. Assim, começamos a ganhar a confiança do dono do produto e dos usuários do sistema." (Alexer, 23/09/2014, dp)*

Devido à confiança construída entre o dono do produto (presidente da CP e seus membros), gerente do projeto e os desenvolvedores, melhores resultados com o trabalho em equipe foram alcançados, culminando em 2010, com o prêmio na categoria destaque da Unidade em Pesquisa & Desenvolvimento (P&D).

No que tange ao relacionamento com o negócio, a CP se reunia uma ou duas vezes por ano de maneira presencial, porém uma comunicação mais intensa (quase que diária) acontecia por e-mail e listas de discussões, onde o(a) presidente da CP era o(a) principal agregador(a) das demandas para o desenvolvimento do novo sistema, sendo ele(a) o ponto focal nas questões de requisitos para o novo sistema. No primeiro momento, como também era um processo para mudar a forma de trabalho do dono do produto, a equipe de desenvolvimento começou a anotar em papel os requisitos; depois a equipe começou a registrá-los em uma ferramenta de apoio ao desenvolvimento de software; mais tarde, o próprio dono do produto começou a registrar suas necessidades na ferramenta. Na

sequência, essas necessidades de negócio eram priorizadas, sempre buscando fazer as coisas o mais rápido possível, sendo que a equipe não estimava o esforço do trabalho que seria executado, conforme relatou Alexer:

> *"(...) fazer um planejamento de quanto iria demorar as tarefas não foi feito, tinha coisas que eram difíceis saber quanto tempo iria durar. Então, em algumas situações mostrávamos alguma coisa em uma semana, e em outras situações em duas ou três semanas. Ou seja, não tinha uma disciplina de juntar todos os requisitos e mostrar em quanto tempo tudo ficaria pronto, algumas tecnologias ainda eram novas para a equipe. Além disso, ao mesmo tempo que estávamos desenvolvendo uma nova versão do sistema, tínhamos que garantir que a nova versão iria suportar a migração de dados do sistema antigo, bem como, continuar oferecendo suporte as versões mais antigas do sistema." (Alexer, 23/09/2014, dp)*

Sobre as mudanças de prioridades de negócio, conforme as falas dos sujeitos, não haviam resistências em realiza-las:

> *"Não vejo problemas quanto a isso. Mas, primeiramente "olhamos" a viabilidade da mudança e depois "partimos" para a implementação." (Cassios, 23/09/2014, dp)*

> *"As mudanças que ocorreram eu não irei me lembrar com detalhes, basicamente a gente discutia os efeitos das mudanças e implementava. A equipe de desenvolvimento não colocava resistência alguma quanto a fazer mudanças. Ninguém se sentia assim: "Puxa vida vai mudar de novo", a equipe não teve esse espírito. A gente não deixou isso acontecer na equipe. Por outro lado, o dono do produto entendeu que a mudança deveria ser bem justificada, bem pensada e bem elaborada para não se dar ao vento. O vento vai para um lado e muda, e o vento vai para o outro lado e muda também." (Alexer, 23/09/2014, dp)*

O uso de métodos ágeis levou o projeto a um maior comprometimento do cliente e uma maior consciência dos desenvolvedores aos objetivos de negócio, o que gerou uma mentalidade de responsabilidade compartilhada. Essa constatação foi observada na fala de Alexer:

> *"(...) em um outro projeto, eu sofri muito com o outro processo, um processo onde o cliente não se responsabilizava por "patavinas" nenhuma no sistema, e colocava toda a responsabilidade para a equipe técnica; e para agravar a situação, nós da equipe técnica caíamos na armadilha de implementar coisas que o cliente não precisava, e até mesmo, incluir novas tecnologias que não iriam agregar benefício algum ao sistema. Assim, nesse projeto, ágil ajudou muito a resolver isso." (Alexer, 23/09/2014, dp)*

4.2.2.4 Informações sobre o Processo e as Práticas

De 2000 a 2003, Alexer trabalhou com algumas práticas de XP em um outro projeto de desenvolvimento de software na Unidade. Os bons resultados alcançados motivaram Alexer a repetir essa experiência no Projeto 2, conforme observou-se em sua fala:

> *"Tive uma experiência em um outro projeto de 2000 a 2003, onde nós utilizamos algumas práticas de XP. Então, a gente nesse projeto resolveu utilizar algumas dessas práticas de XP, que no fim são boas práticas de desenvolvimento de software em várias metodologias, inclusive agora não somente em ágil. Escrever testes unitários, por exemplo, é muito importante independente da metodologia utilizada. Eu não consigo dizer que usamos XP em sua totalidade. Dado o tamanho da equipe, Programação em Par, por exemplo, não era possível. Revisão de código era feita juntamente com os estagiários. Teve Planning Game, controlador de versão, integração contínua e testes automatizados." (Alexer, 23/09/2014, dp)*

Muitos dos benefícios dos métodos ágeis podem ser negados quando o trabalho em equipe não acontece. Sabendo disso, Alexer elegeu como principal lição aprendida a importância do estabelecimento da relação de confiança, sendo mais importante do que processos e práticas:

> *"A questão principal para mim foi como conseguir estabelecer uma relação de confiança entre todas as pessoas do projeto. Por mais que tivesse práticas ágeis (tais como: integração contínua, testes unitários, testes de aceitação, etc.), tecnologias abertas, pessoas com domínio sobre essas tecnologias, se não tivesse a relação de confiança, então nós não estaríamos satisfeitos e nem o cliente. É o que eu levo para todos os outros projetos. Se existir essa relação de confiança entre a equipe de TI e o dono do produto, bem como, se existir um entendimento mútuo sobre as responsabilidades, problemas e desafios de cada um, melhores resultados podem ser alcançados. Vale salientar que depois da troca da presidência da CP, foi colocado como um ponto chave para o sucesso o fato de manter e continuar essa relação de confiança. Porque a quebra disso, como nós já vimos em outros sistemas, pode levar o projeto ao fracasso, sendo mais impactante, inclusive, do que os problemas técnicos." (Alexer, 23/09/2014, dp)*

Continuando esse assunto, de acordo com a fala de Alexer a relação de confiança é mais difícil de ser construída no setor público do que no setor privado:

> *"Olhando para as práticas que foram adotadas no projeto e olhando para o relacionamento com o dono do produto, eu não tenho dúvida alguma de que fazer trocas de pessoas que atuam sobre o mesmo código-fonte é importante e não irá atrapalhar o andamento do projeto. Eu não tenho dúvidas sobre os benefícios da integração contínua, eu não tenho dúvidas sobre os benefícios de escrever testes. Porém, o ponto*

> *chave é a relação com o dono do produto e a capacidade de estabelecer uma relação de confiança, principalmente em empresas públicas. Em empresas privadas, se isso não acontece, ambos ou alguns deles estarão indo embora da empresa; assim, essa relação acontece de maneira forçada, onde o nível de discussão vai até certo ponto e se nada acontece, vem um gerente lá de cima e resolve. No nosso caso essa possibilidade não existia, assim um exercício de fala e convencimento foi muito maior para conquistar a confiança. O que nota-se é que após a troca da CP, então esse processo foi facilitado, talvez por conta do aprendizado da importância de estabelecer uma relação de confiança entre a equipe de desenvolvimento e o dono do produto." (Alexer, 23/09/2014, dp)*

Ainda sobre esse assunto, para alcançar a tão desejada relação de confiança, não houve mais nada relevante do que apresentar resultados práticos que trouxessem benefícios reais aos objetivos de negócio, conforme mencionado na fala dos sujeitos:

> *"Para recuperar a confiança do dono do produto, passamos a fazer e a entregar coisas no menor espaço de tempo possível. Primeiro, passamos a identificar melhor o que o cliente estava fazendo atribuindo a ele a responsabilidade pelo o que estava sendo pedido, e mostrando a ele o quanto "custava" fazer aquele tipo de coisa. Então, existia uma lista enorme de coisas para fazer, a gente tentava priorizar com o dono do produto, mostrando a ele o quanto que "custava" cada coisa a ser feita. No início, o dono do produto não acreditava que esse custo era real, então era um trabalho a médio prazo. Passamos a trabalhar dessa forma e rapidamente mostrar resultados com algumas coisas funcionando em tempos curtos. Com isso, o dono do produto passou a perceber que a equipe de desenvolvimento estava fazendo as coisas acontecer; ao mesmo tempo, a equipe de desenvolvimento começou a atribuir a responsabilidade das coisas que estavam sendo feitas para o dono do produto, para que ele se sentisse responsável pelo sistema. "Você pediu isso e a gente fez". Além disso, a equipe de desenvolvimento passou a atuar em partes do sistema que traziam resultados imediatos para a empresa e a sociedade, como por exemplo a interface de busca de informações. Com isso, a equipe conseguiu rapidamente apresentar mais resultados para o dono do produto. Isso foi muito mais produtivo do que fazer versões do sistema com base em cadastros, que trazem pouco ou nenhum benefício perceptível. (Alexer, 23/09/2014, dp)*

> *"A principal lição é a forma de trabalho, a qual é muito diferente da forma como eu atuava anteriormente. O pensamento de ter um software funcional em uma semana, por exemplo, é interessante para o usuário. Se o projeto tem um ano de duração, o software não é mais entregue ao final desse um ano. Pelo contrário, várias funcionalidades são entregues para o usuário utilizar em espaços curtos de tempo. O usuário sempre tem algo novo, e isso é um "chamariz"." (Cassios, 23/09/2014, dp)*

Nem sempre métodos ágeis significa fazer menos, por exemplo, mais testes são recomendados. Além disso, métodos ágeis não elimina a importância dos documentos escritos. Estas constatações foram observadas na fala de Cassios:

> *"Outro ponto, é que talvez, continuamos a fazer atividades que não estão relacionadas diretamente com o desenvolvimento de software propriamente dito. Continuamos a fazer as documentações necessárias, as quais ainda são importantes. Uma outra questão, é que fazemos muitas coisas que os usuários não veem, e isso incluiu o arcabouço de testes, o que toma mais tempo, embora os testes trazem um benefício futuro (...) que dá maior segurança no momento de alterar qualquer coisa no código." (Cassios, 23/09/2014, dp)*

Vale salientar que as entregas de software aconteceram em tempos variáveis, sendo mais intensas no início do projeto e menos frequentes ao longo do tempo, conforme apontou a fala de Cassios:

> *"No início, as entregas eram quinzenais, mas depois passaram a ser mensais, e agora é sob demanda (justamente porque menos features novas têm aparecido). Hoje as maiores demandas são em relação a integração do sistema com outros sistemas da empresa (geralmente por serviços web), e não afetam os usuários do software." (Cassios, 23/09/2014, dp)*

4.2.2.5 Informações sobre o Conhecimento e a Experiência em Métodos Ágeis

Já que a transição para métodos ágeis envolve pedir que as pessoas trabalhem de maneiras que não lhes são familiares, então essas pessoas precisam ser convencidas de que o desenvolvimento ágil irá ajudá-las em suas atividades. Essa foi a maneira encontrada por Alexer para fazer com que XP tivesse aceitação imediata entre os desenvolvedores:

> *"Eu tinha uma experiência anterior com XP, porém, as outras duas pessoas que atuavam no desenvolvimento do sistema não tinham experiência alguma com métodos ágeis. Então, houve um processo de fazer essas pessoas acreditarem que fazer integração contínua era importante, que usar controlador de versão adequadamente era importante, que revisão de código era importante, que escrever testes era importante também." (Alexer, 23/09/2014, dp)*

4.2.3 Caso Organizacional

Na Embrapa Informática Agropecuária, as pessoas que atuam no desenvolvimento de software (aproximadamente 34 pessoas, incluindo desenvolvedores e gerentes de projetos) estão estruturadas em projetos e laboratórios de maneira distribuída, aonde elas não estão sob a mesma gestão. As pessoas podem atuar em mais de um projeto

concomitantemente, e não necessariamente conversam entre si, não estão fisicamente próximas, não trabalham com as mesmas tecnologias e muitas vezes trabalham em problemas distintos no desenvolvimento de software, incluindo Sistemas de Informação, sistemas voltados à computação científica, bioinformática, etc. Assim, de acordo com Seiya, a primeira dificuldade que surge é:

> *"(...) mesmo que você tenha alguma prática para ser implantada é muito difícil adotá-la completamente em todos os diferentes tipos de projeto." (Seiya, 18/09/2014, do)*

Nessa linha, segundo Seiya, embora que as pessoas compreendam que algumas práticas do desenvolvimento ágil são interessantes e exista uma predisposição para mudanças, elas não conseguem avançar sozinhas porque em muitos casos elas são a única pessoa de TI alocada naquele projeto. Durante as entrevistas ficou claro que a organização utiliza uma abordagem de execução de vários projetos concorrentes, sendo que as equipes são formadas por no máximo 2 desenvolvedores e eles podem atuar em mais de um projeto de maneira simultânea (multitarefa). Como efeito dessa abordagem, alguns indícios negativos apareceram na fala dos sujeitos, os quais são inibidores do trabalho em equipe e do desenvolvimento ágil de software:

> *"O que eu acho prejudicial é o fato de ter pessoas atuando em vários projetos simultaneamente, o que contribui para dispersar as pessoas do trabalho em equipe. O resultado é que as pessoas não conseguem conversar entre si. O que eu gostaria é formar equipes com mais pessoas que "tocam" um projeto e termina, e vai para outro projeto e termina também. O que temos hoje são projetos "tocados" por uma única pessoa; se essa pessoa se desliga do projeto, há uma grande chance do projeto não ir adiante. Então, falta mais gente para desenvolver softwares na instituição, o que iria permitir a formação de equipes." (Cassios, 23/09/2014, dp)*

> *"Eu vejo que falta uma integração entre as pessoas que atuam com o desenvolvimento de software na Unidade, principalmente para troca de experiências. Para os novos empregados seria importante que essa troca de experiências acontecesse de maneira sistemática. Eu sempre encontrei isso em outras empresas que trabalhei, era um pouco diferente. Estávamos mais próximos uns dos outros, e conhecíamos as pessoas a quem poderíamos pedir ajuda." (Shun, 15/09/2014, dp)*

> *"Faz-se necessário melhorar o conhecimento sobre as tecnologias que estão sendo utilizadas, bem como, faz-se necessário melhorar a troca de experiência entre as pessoas. Me parece que o desenvolvimento de um novo software é um esforço muito grande. Se houvesse um maior contato com outras pessoas, talvez algumas coisas andariam mais rápidas." (Ikki, 17/09/2014, dp)*

> *“No âmbito dos Sistemas de Informação, precisamos ajustar o número de projetos de desenvolvimento em relação ao número de desenvolvedores para não deixá-los sobrecarregados com vários projetos ao mesmo tempo. Na mesma direção, a rotatividade de pessoas no desenvolvimento também é prejudicial para o trabalho em equipe. Além disso, precisamos capacitar continuamente os empregados da área de desenvolvimento nas tecnologias e metodologias mais modernas.” (Shiryu, 15/09/2014, dp)*

> *“(...) a minha maior consideração é que nós não conseguimos aplicar essas práticas com equipes compartilhadas em vários projetos; a situação que temos hoje é que existem pessoas atuando em dois ou três projetos simultaneamente com papéis distintos em cada um deles, onde cada projeto segue uma linha, então é muito difícil aplicar qualquer prática de entregas curtas, reuniões diárias, etc.” (Alexer, 23/09/2014, dp)*

Após identificar essas vulnerabilidades, então a organização fica com o problema de como eliminá-las, que na visão de Seiya deveria perpassar por uma restruturação da alocação das pessoas:

> *“Eu espero que a estrutura das equipes de desenvolvimento seja alterada, pelo menos para os Sistemas de Informação. Nós já mostramos que projetos que utilizaram algumas práticas ágeis foram bem-sucedidos. Isso é visível, a Unidade sabe disso. Acredito que a reorganização das equipes irá fazer com que as pessoas se sintam apoiadas em continuar com a implantação de métodos ágeis. Provavelmente, nessa reorganização poderá existir uma pessoa que será um “evangelizador” dos métodos ágeis, e irá fazer com que isso aconteça nos projetos. Eu não vejo mais nada relevante do que a restruturação da área, para que o desenvolvimento ágil consiga alcançar seu pleno êxito. Ou então, mesmo se não houver uma reorganização dessa estrutura, que os projetos consigam formar um time para o desenvolvimento de software (composto por duas, três ou quatro pessoas).” (Seiya, 18/09/2014, do)*

Além disso, é difícil prever exatamente como as pessoas responderão às mudanças que serão necessárias no caminho de se tornar ágil, sendo necessário um apoio dos níveis mais altos:

> *“(...) precisa de uma compreensão maior das pessoas sobre a importância dos métodos ágeis, uma orientação institucional sobre sua utilização e uma reorganização das pessoas de desenvolvimento em relação aos projetos. Com certeza, a Unidade ganharia muito mais com isso.” (Alexer, 23/09/2014, dp)*

> *“Um tipo de apoio que pode ocorrer é uma decisão gerencial da gestão em falar: “Sim vai ser ágil”. “Sim nós vamos mudar”. Só queria ressaltar que quando se fala em ter uma decisão gerencial, não é uma crítica ao gestor que não toma a decisão; pelo contrário, é uma decisão difícil, principalmente com a demanda e a estrutura de projetos que nós temos. Precisa ter uma certa coragem em realizar essas mudanças. Porém, é uma coragem com responsabilidade, e não uma coragem sem responsabilidade. Precisa de um plano para fazer isso acontecer e um convencimento para que as pessoas mudem para essa nova abordagem. Embora seja difícil, o cenário é favorável e algumas questões têm avançado na Unidade. Alguns resultados positivos em projetos já foram alcançados. Já houve um treinamento de ágil para algumas pessoas, e um outro treinamento irá ocorrer para mais dez pessoas.” (Seiya, 18/09/2014, do)*

Vale salientar que o ato do uso de métodos ágeis pode levar a resultados não tão bons como aqueles preconizados na teoria. Basicamente, essa foi a opinião de alguns sujeitos ao comentarem o que eles não apreciavam em métodos ágeis:

> *“(...) o discurso teórico e favorável aos métodos ágeis é adequado para cenários perfeitos, o que pode diferir da realidade. Por exemplo, nesse projeto apareceram variáveis que aumentaram a complexidade no desenvolvimento de software, fazendo com que as iterações se tornassem longas e variáveis; assim, nem sempre o resultado teórico é igual ao resultado prático.” (Shiryu, 15/09/2014, dp)*

> *“Se tudo funcionasse como está escrito seria o ideal, mas a realidade é diferente. Me parece que eles funcionam melhor para equipes que são coesas e seguem uma determinada linha de construção das coisas. Isso produz um resultado mais coerente do software.” (Ikki, 17/09/2014, dp)*

> *“(...) muita gente coloca os métodos ágeis como solução para tudo. Então, no início surgiu um “boom” como se fosse resolver tudo, depois muita gente viu que não era bem assim, e daí a turma começou a entrar na mediana, fazendo crescer o uso de métodos ágeis.” (Alexer, 23/09/2014, dp)*

> *“Métodos ágeis não elimina as tarefas que não estão associadas com o desenvolvimento propriamente dito, isso pode frustrar um pouco as pessoas.” (Cassios, 23/09/2014, dp)*

Como visto nos Estudos de Caso 1 e 2 (ECa01) (ECa02), os projetos são de longa duração, sendo executados em 36 meses e 84 meses respectivamente. Em ambos os projetos, o trabalho expandiu-se de modo a preencher o tempo disponível para a sua realização. Eles são sistemas longevos, cujo o comportamento do software precisa ser mantido ao longo do tempo porque eles ainda são importantes na Empresa. Além disso, eles não são resistentes ao tempo e precisam ser atualizados para não se tornarem

obsoletos em pouco tempo. Assim, eles precisam de manutenções frequentes, o que contribui para agravar ainda mais os problemas no desenvolvimento de software na organização, onde algumas pessoas não são liberadas para trabalhar em novos projetos por conta da manutenção dos sistemas existentes, conforme relato de Cassios:

> *"Outro ponto é que a instituição prioriza o sistema novo, porém a realidade mostra que existem softwares que precisam ser mantidos (tem suporte a usuários, máquinas precisam ser mantidas, etc.), o que muitas vezes não libera tempo para fazer os sistemas novos e nem mesmo desenvolver novas funcionalidades para os existentes, assim precisamos de mais gente. E como fazer isso e como provar que precisamos de mais gente, estas e outras questões ainda são questões em aberto na instituição." (Cassios, 23/09/2014, dp)*

Em geral, conforme constatado, a criação de qualquer plano de melhoria no desenvolvimento de software no CNPTIA deveria suplantar algumas barreiras: (1) conseguir formar equipes com dedicação integral para o projeto; (2) avaliar a quantidade de projetos em andamento em relação ao número atual de desenvolvedores; (3) diminuir significativamente o tempo de entrega de novos produtos de software; (4) criar estratégias melhores para manutenção dos sistemas existentes; (5) criar condições para o compartilhamento de conhecimento e experiências entre as pessoas; (6) uma vez que essas barreiras não podem ser superadas sem o comprometimento das pessoas, o que se pode fazer é incluí-las nesse processo de transformação porque serão elas próprias que encontrarão as melhores respostas de como o desenvolvimento de software poderá ser melhor executado na Empresa.

4.3 Companhia de Processamento de Dados do Rio Grande do Sul: Estudos de Caso

Na Companhia de Processamento de Dados do Rio Grande do Sul são muitos os projetos ágeis. Porém, apenas 2 projetos executados recentemente no setor SD1 foram escolhidos e compõem as unidades de análise, sendo apresentados a seguir.

4.3.1 Caso 3

O caso a seguir é um exemplo de migração e integração de sistemas, do tipo difícil para métodos ágeis, que envolve alguns elementos arriscados ao mesmo tempo. Ele mostra como não é fácil desenvolver e implantar um novo sistema enquanto o sistema atual continua sendo utilizado. Ou seja, é muito difícil colocar partes de um novo sistema em produção e, ao mesmo tempo, ir desabilitando aos poucos algumas funcionalidades do sistema atual. Para agravar o problema, há outros sistemas que consomem informações do sistema atual e também precisam ser alterados para consumir essas mesmas

informações do novo sistema. Ou seja, além da migração de dados, da mudança tecnológica e da mudança de interfaces, também há o desafio da dependência entre os sistemas. Esta e outras informações encontradas no Caso 3 são apresentadas na sequência.

4.3.1.1 Informações sobre o Projeto

O Projeto 3 tem como objetivo tecnológico migrar o Sistema C de um modelo cliente/servidor para um modelo *web*, criando melhores condições de integração com outros sistemas da Empresa. Além disso, o projeto contempla a inclusão de melhorias no processo atual, incluindo o gerenciamento de empresas credenciadas ao Departamento Estadual de Trânsito do Rio Grande do Sul (DETRAN/RS), o gerenciamento de profissionais que possuem algum tipo de relação de trabalho com o DETRAN/RS e o gerenciamento de recursos físicos (salas, equipamentos, etc.) utilizados pelas empresas credenciadas. Além das empresas credenciadas, o sistema abrange cinco departamentos do DETRAN/RS, bem como, mais cinco departamentos da PROCERGS.

O desenvolvimento do novo Sistema C foi iniciado em Julho de 2014 com previsão de encerramento para Dezembro de 2016, totalizando 30 meses aproximadamente. Porém, o desenvolvimento e a implantação dar-se-ão com base no modelo iterativo e incremental, conforme observado na fala de Kasa:

> *“Pretendemos disponibilizar novas versões do sistema em produção a cada trinta dias, sendo duas semanas para o desenvolvimento e mais duas semanas para homologação do cliente.” (Kasa, 15/10/2014, dp)*

Além disso, na visão de Shunrei, o lançamento de software mais rápido contribui para aumentar o comprometimento das OP na utilização daquilo que está sendo feito:

> *“Eu já tive uma experiência de passar um ano todo desenvolvendo algo que o cliente nunca utilizou porque o cliente mudou; e isso é muito comum de acontecer em órgãos públicos. Troca-se pessoas e partidos administrativos, então muita coisa deixa de ser importante e nunca é utilizada. Assim, o quanto antes o cliente começar a usar o sistema, então ele terá um maior comprometimento em utilizar aquilo.” (Shunrei, 14/10/2014, dp).*

4.3.1.2 Informações sobre a Equipe

A equipe do projeto é composta por 1 *Product Owner*, 1 Analista de Sistemas, 2 desenvolvedores, 1 *ScrumMaster* / Desenvolvedor e 1 Analista de Qualidade, totalizando 6 pessoas. Com exceção do Analista de Qualidade que possui outras atividades em seu setor de origem, todos os membros da equipe possuem dedicação de tempo total ao projeto. Eles

estão localizados na mesma “ilha” bem próximos uns dos outros, sendo que alguns benefícios dessa configuração de ambiente de trabalho foram encontrados nas falas de Tokumaru e Miho:

> *“O convívio é muito tranquilo e a comunicação é bem aberta, o principal é isso.” (Tokumaru, 14/10/2014, dp)*

> *“(...) como nós trabalhamos nesse formato de mesa, que lembra uma mesa de reunião, muitas vezes as discussões já estão acontecendo e, de certa forma, nós já estamos participando porque estamos sentados um de frente para o outro. Então, dependendo da discussão e quando ela começa a dar mais problemas, é comum todos pararem o que estão fazendo e prestar mais atenção no que está sendo falado, o resultado é que sugestões começam a aparecer. Ou seja, (...) o diálogo pode acontecer a qualquer momento.” (Miho, 14/10/2014, dp)*

Além disso, a formação da equipe procurou maximizar as pessoas que já se conheciam a mais tempo, visando a formação de equipes com um elevado estado de colaboração entre as pessoas, o que pode ajudar as equipes a errarem menos nas fases iniciais do projeto, conforme relatou Shunrei:

> *“Além disso, repetir a equipe é muito bom porque é possível conhecer o ritmo da equipe. Se não for assim, então nas primeiras sprints a equipe irá errar muito.” (Shunrei, 14/10/2014, dp)*

Sobre o papel do *Product Owner* (PO), ele é relativamente novo para a Empresa. Nesse projeto, o papel do PO é executado por um funcionário da própria PROCERGS, ou seja, um cliente real não assumiu o papel de PO. O PO faz parte da equipe do projeto e colabora de perto com seus outros membros, ou seja, ser PO não é um ato isolado.

Pode ser tentador executar o papel do PO como o de gerente de projeto ou produto tradicional. Porém, métodos ágeis exige uma mudança de mentalidade (adaptação) sobre a forma como as pessoas estão acostumadas a fazer o seu trabalho, conforme revelou o discurso de Tokumaru e Kasa:

> *“Houve uma dificuldade inicial muito grande de adaptação por parte do PO aos métodos ágeis. No sentido de adquirir confiança no trabalho em equipe. Então, isso foi um desafio para o PO, mudar a sua forma de trabalho com o cliente e com a equipe. Depois de muitas conversações e trabalho também, então acredito que a equipe conseguiu passar mais tranquilidade para o PO.” (Tokumaru, 14/10/2014, dp)*

> *“É um pouco difícil encontrar pessoas que queiram trabalhar juntas, compartilhando o mesmo espaço físico porque elas estão acostumadas a trabalhar de uma maneira diferente e isso exige um esforço de adaptação para nova maneira de trabalhar. Então,*

> *isso causa um certo desconforto, a pessoa fica incomodada e acaba desistindo da mudança." (Kasa, 15/10/2014, dp)*

4.3.1.3 Informações sobre o Relacionamento com o Cliente e com o Negócio

Com a inclusão do papel do PO no projeto, ele passou a liderar o esforço de desenvolvimento para criar o produto de software certo para os clientes. Isso tem incluído: a descoberta e o planejamento dos itens da iteração; a organização do *backlog* do produto; o envolvimento com os clientes, usuários e outros *stakeholders*; a assistência às reuniões de Scrum e a colaboração com a equipe. Como são muitas as atribuições do PO, para evitar sobrecarregá-lo, então o PO é uma função de tempo integral, onde ele cuida de um produto e interage com uma equipe. Além disso, designou-se 1 Analista de Sistemas e 1 Analista de Qualidade a colaborar com o PO para não deixar o trabalho pesado para uma só pessoa, bem como, ajudá-lo a incluir aspectos técnicos no *backlog* do produto.

O trabalho do PO e seus 2 colaboradores técnicos ampliou o tempo dos desenvolvedores para atuar na codificação do sistema, uma vez que eles não precisam gastar mais tempo no diálogo com os clientes para poder determinar e organizar os itens de trabalho a serem implementados, assim como, não dispendem mais tempo com a elaboração de cenários de testes.

Outrossim, escolher a pessoa certa para ocupar o papel de PO é algo desafiador. Em projetos de migração de sistemas, existe o risco de simplesmente "copiar" o sistema atual para o novo sistema, sem agregar nenhum benefício de negócio para o cliente, principalmente quando o PO tem conhecimento amplo sobre o sistema atual. Esse problema foi denunciado nas falas de Kasa, Shunrei e Tokumaru:

> *"No início do projeto, como eu já tinha trabalhado com o sistema atual, então era comum eu me basear no sistema atual para manifestar as minhas opiniões, o que poderia ser prejudicial para a prospecção de novas alternativas para o novo sistema. (...) hoje nós não utilizamos o sistema atual como base, então procuramos enxergar de maneira diferente e incluir melhorias nos processos porque isso é uma demanda do próprio cliente, justamente porque o cliente percebeu que várias coisas precisam mudar." (Kasa, 15/10/2014, dp)*

> *"Esse é um projeto de migração. O ideal seria identificar funcionalidades que não estão sendo mais utilizadas e eliminá-las, bem como, incluir algumas novas funcionalidades. Porém, uma vez que o foco maior é a migração dos aspectos tecnológicos e por conta do prazo, então nem sempre é possível incluir todas as funcionalidades que o cliente gostaria de ter." (Shunrei, 14/10/2014, dp)*

> *“De certa maneira, o PO estava pressionado para efetuar a migração do sistema atual, o quanto antes para o novo sistema sob o risco do projeto ser cancelado. E esse fato inibia que o PO aceitasse ideias novas do cliente porque as novas ideias poderiam dificultar a migração tecnológica para o novo sistema. Porém, hoje o PO está mais propenso a incorporar as novas sugestões do cliente porque a equipe técnica do projeto tem conseguido transmitir mais tranquilidade para o PO, mostrando que as coisas irão funcionar tecnicamente muito bem nesse momento de transição para o novo sistema.” (Tokumaru, 14/10/2014, dp)*

Para criar o produto certo o PO, a Analista de Sistemas, o Analista de Qualidade, o *ScrumMaster* e os desenvolvedores trabalham para conhecer e satisfazer as necessidades dos clientes. Segundo eles, a melhor maneira de fazer isso é envolver os clientes desde cedo, e continuamente, no processo de desenvolvimento incluindo-o na identificação do problema, na construção de histórias de usuário, na avaliação de protótipos, na participação nas reuniões de revisão da iteração e na homologação de versões do sistema. Na mesma direção, o lançamento de software mais cedo e com frequência quinzenal tem contribuído para os clientes visualizarem o avanço no desenvolvimento do sistema, vislumbrarem outras ideias que poderão ser agregadas ao software e aumentarem a sua confiança com relação ao trabalho em equipe, conforme observado na fala de Tokumaru:

> *“Acredito que nessas primeiras sprints, conseguimos estabelecer uma relação de parceria com o cliente, incluindo fazer ele entender algumas necessidades da migração e as dificuldades que isso gera para o desenvolvimento. Ou seja, conseguimos estabelecer uma relação de confiança, onde as novas demandas de processo (provenientes do cliente) e os aspectos técnicos da migração (provenientes da nossa Empresa) são atendidos simultaneamente.” (Tokumaru, 14/10/2014, dp).*

Embora adjetivos positivos não tenham faltado para descrever os clientes, tais como:

> *“O nosso cliente é muito bom. Ele é bem aberto e comunicativo (...).” (Tokumaru, 14/10/2014, dp)*

> *“A gente tem sorte com o nosso cliente porque ele é muito bom. Ele é muito responsável, proativo e interessado no projeto. Ele tem uma visão muito clara do que ele quer e tem uma visão muito boa do processo e sabe exatamente que pontos do processo podem ser melhorados.” (Shunrei, 14/10/2014, dp)*

> *“(...) o cliente é flexível e muito colaborativo, o que tem ajudado bastante” (Kasa, 15/10/2014)*

Alguns problemas relacionados com a disponibilidade do cliente têm acontecido, conforme relatado por Kasa e Shunrei:

> *"A relação com o cliente sempre é a mais complicada porque é difícil o cliente se organizar e dispor de um tempo maior para atuar com a equipe de desenvolvimento. Por exemplo, mesmo que as reuniões sejam marcadas com antecedência, mesmo que alguns documentos sejam disponibilizados para avaliação prévia, ainda é comum que alguns clientes não conseguem preparar-se para as reuniões, sendo muita coisa vista e avaliada na hora da reunião mesmo." (Kasa, 15/10/2014, dp)*

> *Como o projeto envolve diversos setores e são muitos clientes, então é muito difícil conciliar a agenda de todo mundo. Em algumas reuniões, não conseguimos avançar na pauta por conta da ausência de determinadas pessoas. Isso acontece porque são muitos clientes. (Shunrei, 14/10/2014, dp)*

Ademais, a equipe enfrentou dificuldades em convencer os clientes a realizar publicações parciais do sistema em produção, mesmo que agregassem valor ao negócio, conforme observado nas falas de Kasa e Shunrei:

> *"Até o momento, executamos quatro sprints, porém o cliente demonstrou uma certa preocupação em colocar o sistema em produção desde o início, por conta das alterações tecnológicas que aconteceram, onde um novo sistema web está substituindo um sistema desktop (cliente/servidor). Porém, nesse momento o cliente já está mais seguro e uma versão deverá ser colocada em produção ainda esse mês, justamente para que os usuários possam ir acostumando-se com o novo sistema." (Kasa, 15/10/2014, dp)*

> *"Uma coisa que está acontecendo com o ágil é a seguinte: antes a gente corria atrás do cliente para entregar, agora nós conseguimos entregar algo de valor para ele de maneira antecipada. Porém, muitas das vezes ele não está preparado para efetivamente utilizar algo em produção tão rápido. Do ponto de vista do cliente, ele esperaria muito mais tempo. Por outro lado, para o desenvolvimento ágil é importante colocar algo em produção o quanto antes para identificar problemas e coletar feedback. Para isso, estamos trabalhando com uma abordagem de utilização do sistema com base em pilotos." (Shunrei, 14/10/2014, dp)*

4.3.1.4 Informações sobre o Processo e as Práticas

Scrum é o método ágil adotado no Caso 3, o qual tem contribuído para a formação de uma cultura de trabalho em equipe de maneira organizada. O espaço da equipe é propício ao trabalho colaborativo, ele facilita a comunicação e a interação entre as pessoas, tornando o trabalho agradável e permitindo a visualização holística do projeto através de

disseminadores visuais de informação (incluindo o quadro kanban e o gráfico de *burndown*) e ferramenta de suporte ao desenvolvimento de software (incluindo a ferramenta Redmine). Assim, Scrum incutiu uma mudança na forma de trabalho das pessoas a favor do trabalho em equipe, conforme observado na fala de Shunrei, Miho e Krishna:

> *"(...) no modelo anterior não tínhamos uma visão holística do que estava acontecendo no projeto, incluindo as tarefas que cada um estava fazendo, os problemas que estavam acontecendo e os impedimentos que estavam bloqueando o avanço do projeto. Métodos ágeis veio para deixar tudo isso visível, o que contribui para unir a equipe em torno da resolução de problemas e na busca das melhores soluções." (Shunrei, 14/10/2014, dp)*
>
> *"Se algum problema está acontecendo e as pessoas não demonstram o que está acontecendo, então isso acaba sendo ruim porque depois de vários dias você vai descobrir que havia um problema. Então, nas reuniões diárias é uma oportunidade de descobrir isso." (Miho, 14/10/2014, dp)*
>
> *"Métodos ágeis favorece a integração entre as pessoas. Nas reuniões diárias é possível trocar informações entre a equipe. Nessas reuniões também é possível expor os impedimentos, sendo que as pessoas sempre estão dispostas a sugerir alternativas para que haja uma solução mais rápida." (Krishna, 14/10/2014, dp)*

A forma com que a equipe executa o método Scrum é revista continuamente. Mais recentemente, o projeto beneficiou-se da contratação externa de um *agile coach*, que ajudou a equipe na melhoria da execução do método Scrum. As principais alterações foram: (1) o papel do PO foi adicionado ao projeto, o que liberou mais tempo para a equipe de desenvolvimento atuar na codificação do sistema; (2) a equipe passou a trabalhar com histórias de usuário em vez de casos de uso, o que ampliou o conhecimento da equipe em artefatos ágeis; (3) os itens de trabalho passaram a ser estimados em horas em vez de pontos de história, o que melhorou a compreensão da equipe no que tange ao tempo restante para fazer as coisas; (4) um Analista de Qualidade foi incluído no projeto para melhorar a qualidade do software e começar a disseminar a disciplina de testes ágeis no projeto; (5) um fluxo contínuo e sustentável de projeto com base em ciclos de *discovery* e *delivery, sprints* quinzenais e critério mínimo de definição de preparado para que haja suficiente entendimento por parte dos desenvolvedores foi estabelecido, o que tem evitado ociosidade, desperdício ou retrabalho.

Métodos ágeis veem a mudança como a única constante, o que permanece incerto é apenas em quanto tempo e com que frequência isso vai acontecer [Pic11]. Nesse sentido, uma vez que algumas práticas ágeis estão em pleno uso, sempre é possível encontrar algo a melhorar, o que não é diferente no Projeto 3. Em comum acordo com o Setor de Garantia

de Qualidade (SQS), alocou-se 1 Analista de Qualidade no projeto, o qual é responsável por disseminar na prática a cultura de testes ágeis no projeto. O primeiro passo relevante incluiu a criação de cenários de testes de acordo com a especificação de comportamento *Behavior Driven Development* (BDD). Na sequência, alguns testes funcionais foram implementados. O próximo grande passo consiste em fazer com que o próprio desenvolvedor consiga escrever e automatizar bons testes unitários e de integração, o que segundo a fala de Sorento é um desafio cultural:

> *"Do ponto de vista do desenvolvedor, enquanto ele não consegue visualizar os benefícios em escrever testes automatizados, então essa prática torna-se "chata" com pouca ou nenhuma aceitação. Esse é um desafio cultural porque a cultura de escrita de testes automatizados na Empresa não existe; assim, não será nada fácil fazer isso acontecer na Empresa, ou seja, existe um longo caminho a percorrer." (Sorento, 14/10/2014, dp).*

Por fim, está claro para a equipe que métodos ágeis não significa trabalhar menos. Pelo contrário, o trabalho é mais intenso, mas a satisfação com o trabalho realizado compensa, conforme observou-se nas falas de Tokumaru, Krishna e Shunrei:

> *"A participação constante do cliente é um indicio que ele tem gostado do que está sendo feito e que fique cada vez melhor." (Tokumaru, 14/10/2014, dp).*
>
> *"O cliente deixou claro que é interessante visualizar o avanço no desenvolvimento do sistema. Inclusive, isso permite vislumbrar outras ideias que poderão ser agregadas no projeto." (Krishna, 14/10/2014, dp)*
>
> *"Não temos insatisfações com o sistema porque os clientes participam de todo o processo de desenvolvimento, então dificilmente temos alguma surpresa. O que acontece são pequenos ajustes, mas nada que produza uma insatisfação com o sistema do tipo: não era isso que eu queria." (Shunrei, 14/10/2014, dp)*

4.3.1.5 Informações sobre o Conhecimento e a Experiência em Métodos Ágeis

Além de realizarem cursos de treinamento e participarem de palestras sobre o tema, os empregados estão sendo beneficiados com 6.000 horas de consultoria homologado pela Empresa desde março de 2014. Ou seja, apoio gerencial e suporte operacional as pessoas com menos e mais experiência em métodos ágeis é algo virtuoso na Empresa, o qual foi observado na fala de diversos sujeitos:

> *"Aqui na Empresa, já participei de diversos cursos, treinamentos e palestras (tanto interno como externo). Então, foi possível crescer bastante nessa área aqui na Empresa." (Tokumaru, 14/10/2014, dp)*

> *“O que não tem faltado são treinamentos e palestras. Na semana passada, tivemos uma apresentação sobre introdução as metodologias ágeis, onde aprendemos conceitos de Lean e Kanban.” (Krishna, 14/10/2014, dp)*

> *“Eu tinha uma experiência anterior com métodos ágeis em outras empresas, porém o meu maior entendimento foi conquistado aqui. A Empresa oferece muitos treinamentos, algo que eu não tive em outras empresas anteriormente. Ou seja, além de sair usando na prática nós temos esse suporte. A última consultoria foi excelente, trouxe muitas melhorias para nós e conseguirmos aprender muito também.” (Miho, 14/10/2014, dp)*

Por outro lado, a fala de Shunrei adverte que quando falta experiência na metodologia e na tecnologia, mesmo com todo esse apoio institucional, as dificuldades são maiores, mas os resultados são compensadores também:

> *“(...) no meu primeiro projeto com métodos ágeis eu não tinha nenhuma experiência na metodologia e na tecnologia (dispositivos móveis) que foram utilizadas. Foi uma experiência muito difícil, porém ela foi recompensada com o prêmio máximo SECOP (Prêmio Excelência em Governo Eletrônico) em 2013. Durante a execução do projeto tivemos treinamentos e apoio de consultores, o que foi fundamental. O cliente mostrou-se surpreso porque ele não esperava ver algo funcionando em tão pouco tempo. Nesse projeto, nós erramos nas sprints iniciais porque não conhecíamos muito bem a tecnologia mobile. A área de infraestrutura também enfrentou problemas porque era uma tecnologia nova para eles também. Então, consumimos muito tempo no início do projeto.” (Shunrei, 14/10/2014, dp)*

4.3.2 Caso 4

Este caso é um bom exemplo de como métodos, processos e práticas de ES não produzem resultados absolutos, mesmo quando executados em ambientes, cenários e condições praticamente iguais a de outros projetos. Ele é caracterizado por ter um problema específico na execução do papel do PO. Na sequência, esta e outras informações são apresentadas.

4.3.2.1 Informações sobre o Projeto

Igualmente ao Caso 3, o Caso 4 tem como objetivo técnico migrar o Sistema D de um modelo cliente/servidor para um modelo *web*, sendo que o principal desafio do projeto, do ponto de vista técnico, consiste em realizar a transição do sistema atual para o novo sistema de maneira gradual, ou seja, por algum tempo os dois sistemas serão usados simultaneamente. Como resultado, a Divisão de Remoção e Depósito do DETRAN/RS terá um sistema mais integrado, com novas funcionalidades e melhorias no processo, tudo isso

em uma tecnologia moderna. O novo sistema abrange os Centros de Remoção e Depósito (CRDs), envolvendo o próprio DETRAN/RS e dois setores da PROCERGS. O seu desenvolvimento teve início em março de 2013 e deve estender-se até o final de 2015, totalizando 30 meses aproximadamente. Mas, a implantação é realizada por módulos, conforme relatado por Kiki:

> *"É um grande sistema que está sendo migrado por módulos, sendo que o primeiro módulo foi entregue em Março de 2014 e a sua implantação ocorreu em Julho de 2014. Agora a equipe está trabalhando no segundo módulo." (Kiki, 15/10/2014, dp)*

4.3.2.2 Informações sobre a Equipe

Desde março de 2013, o projeto sofreu algumas alterações na composição da equipe. Atualmente, ela é formada por somente empregados da PROCERGS, incluindo 1 Analista de Sistemas, 1 *Product Owner*, 2 desenvolvedores, 1 ScrumMaster / Desenvolvedor e 1 Analista de Qualidade, totalizando 6 pessoas. Semelhantemente ao Projeto 3, com exceção do Analista de Qualidade que possui outras atividades em seu setor de origem, todos os membros da equipe possuem dedicação de tempo total ao projeto. Por conta de espaço físico, eles estão distribuídos em duas "ilhas" bem próximas umas das outras. Igualmente ao Projeto 3, a formação da equipe procurou maximizar as pessoas que já se conheciam a mais tempo, visando a formação de equipes com um elevado estado de colaboração entre as pessoas.

As falas de Seika, Kiki e Bian expressam claramente o que foi registrado anteriormente no Projeto 3 sobre os benefícios da proximidade física para os métodos ágeis.

> *"A gente troca mais ideias, discute mais os aspectos técnicos do sistema, conversa mais, trabalha muito mais juntos (...) Já no outro modelo, o (...) praticamente trabalhava "sozinho". Ele definia tudo a partir do que ele conversava com o cliente e encaminhava para o desenvolvedor implementar." (Seika, 15/10/2014, dp)*

> *"Dentro do time nós temos um clima muito bom de trabalho. Diga-se de passagem que após a metodologia ágil isso melhorou muito. Como estamos muito próximos, a comunicação é verbal e o e-mail é utilizado apenas para alguma formalização, ou então, para alguma coisa importante que todo o time deveria ficar sabendo." (Kiki, 15/10/2014, dp)*

> *"Como participamos de algumas reuniões, como temos o PO próximo, como podemos contatar o próprio cliente, então isso é suficiente para implementarmos as histórias de usuário." (Bian, 15/10/2014, dp)*

Trabalho em equipe faz parte da essência de qualquer processo ágil. As equipes ágeis são bem-sucedidas em conjunto e falham em conjunto. Não há “meu trabalho” e “seu trabalho” em uma equipe ágil; há apenas “nosso trabalho” [Coh11]. Assim, a responsabilidade do que precisa ser feito e os resultados alcançados é algo compartilhado pela equipe, conforme observado na fala de Kiki:

> *“No método waterfall as tarefas eram individuais, assim a responsabilidade e o pensamento sobre elas eram individuais também. Se a maioria dos membros do projeto concluíam suas tarefas, então não havia uma preocupação com as pessoas e tarefas que ainda não tinham sido concluídas. Métodos ágeis ajudou a corrigir isso, mudando a cultura de trabalho individual para trabalho em equipe, ao dizer que o produto de software é de responsabilidade da equipe. Atualmente, se uma pessoa está com dificuldades para completar uma tarefa, então a equipe se preocupa em ajudá-la para alcançar esse objetivo. Ou seja, o projeto é da equipe, onde todos ganham quando bons resultados são alcançados e todos perdem quando malfeitos acontecem também. Isso foi uma mudança de cultura interessante, porém ela exige mais transparência na comunicação.” (Kiki, 15/10/2014, dp)*

Além disso, o trabalho em equipe contribui para propiciar uma visão holística do projeto, bem como, equalizar o conhecimento técnico da equipe, conforme relatado na fala de Kanon:

> *“(...) Outro ponto é que todos ficam sabendo o que está acontecendo no projeto, e todos tem o mesmo nível de conhecimento técnico sobre o projeto, o que não torna as pessoas insubstituíveis.” (Kanon, 15/10/2014, dp)*

Mais recentemente, adicionou-se o papel do *Product Owner* (PO) ao projeto, sendo executado por um funcionário da própria PROCERGS, ou seja, um cliente real não assumiu o papel de PO. Como o novo sistema tem como objetivo substituir o sistema atual, então a pessoa com maior conhecimento no sistema atual foi escolhida para exercer o papel de PO, a qual deslocou-se de seu setor de origem para atuar juntamente com os demais integrantes da equipe.

Kiki, por outro lado, adverte que uma vez que métodos ágeis e o papel do PO é relativamente novo para a Empresa, então é comum acontecer algumas divergências até que um ponto de equilíbrio seja encontrado.

> *“Como outros setores não avançaram para métodos ágeis, ainda é comum acontecerem divergências sobre a forma de conduzir os trabalhos. Por exemplo, o PO prefere maximizar o uso da comunicação formal em vez da comunicação informal (frente a frente), enquanto as outras pessoas da equipe preferem mais a comunicação informal*

(frente a frente) por ser mais rápida e menos burocrática. Porém alguns avanços têm acontecido para equacionar essa situação." (Kiki, 15/10/2014, dp)

4.3.2.3 Informações sobre o Relacionamento com o Cliente e com o Negócio

Antes da inclusão do papel do PO, a equipe (principalmente os desenvolvedores) assumia mais responsabilidades com o ambiente externo do projeto, onde a equipe conversava mais com os clientes, prospectando soluções futuras de negócio, o que na visão de Kiki e Bian produzia resultados melhores:

> *"O time inteiro reunia-se com o cliente, pelo menos, uma vez por semana. Nessas reuniões, além das questões atuais, o cliente fornecia indícios do horizonte futuro do sistema para as pessoas irem refletindo sobre o assunto. Com isso, o time inteiro conseguia prospectar e apresentar soluções para o cliente, o qual na sequência, refletia sobre as alternativas propostas. Então, como o time conhecia o assunto, quando esses requisitos eram incorporados ao backlog da sprint, as pessoas sentiam-se mais prontas para desenvolver os itens de trabalho da iteração." (Kiki, 15/10/2014, dp)*

> *"Quando não tínhamos o papel do PO, acredito que as coisas fluíam melhor. Toda a equipe reunia-se com o cliente (...) Com a inclusão do papel do PO, então somente o PO participa dessas reuniões, o que na minha opinião trouxe dificuldades para o processo." (Bian, 15/10/2014, dp)*

Atualmente, o projeto "caiu" em uma armadilha potencial na seleção do PO inicial em projetos de migração de sistemas. O PO está "apegado" ao sistema atual e não consegue absorver as mudanças de negócio, conforme relatado por Bian e Kiki:

> *"A escolha do PO foi com base na sua experiência do sistema atual, o que muitas vezes direciona a repetir o que já existe, desertando de qualquer melhoria significativa." (Bian, 15/10/2014, dp)*

> *"De certa maneira, a equipe de desenvolvimento perdeu a comunicação e o contato direto com o cliente, o qual era muito importante para o andamento do projeto. Ou seja, atualmente com o papel do PO, existe o risco das necessidades do cliente não serem bem entendidas pelo PO, que por sua vez irão refletir no software." (Kiki, 15/10/2014, dp)*

Além disso, embora que a equipe tenha conseguido construir um relacionamento bem-sucedido com o cliente, a fala de Bian apontou dificuldades na convergência de interesses entre um determinado departamento interno da PROCERGS e o cliente.

> *"(...) como o PO está submisso ao departamento interno da Empresa (que não é o cliente), muitas vezes, ele enfrenta conflitos de interesses entre o departamento interno*

> *da Empresa e o cliente propriamente dito, o que dificulta o seu trabalho e o andamento do projeto." (Bian, 15/10/2014, dp)*

Com relação a mudanças, a equipe adotou o seguinte caminho. Algo pode mudar dentro da iteração, mas com ressalvas.

> *"Hoje temos uma consciência maior do que são as mudanças e os efeitos que elas geram. Se uma mudança acontece, então não vemos problemas em atendê-la. Porém, o momento de realizá-la é acordado com o cliente, de preferência para a sprint seguinte ou para o final da sprint atual (quando possível). Ou seja, não fazemos mais como antigamente, onde uma mudança quando aparecia, então ela era imediatamente atendida em detrimento dos outros itens que estavam no backlog da sprint." (Kiki, 15/10/2014, dp)*

> *"Quando mudanças acontecem não há resistências em implementá-las. Porém, uma negociação é realizada com o cliente, geralmente intermediada pelo PO." (Seika, 15/10/2014, dp)*

4.3.2.4 Informações sobre o Processo e as Práticas

Da mesma maneira que o Projeto 3, Scrum é o método ágil adotado no Projeto 4. A forma de execução do método Scrum em ambos os projetos é praticamente a mesma, incluindo o processo, as práticas, os artefatos e as ferramentas. O que varia são os resultados e as percepções dos sujeitos sobre o que aconteceu ou está acontecendo na prática. Por exemplo, a fala de Mitsumasa anuncia a importância do *feedback* do cliente sobre o novo incremento de produto de software entregue.

> *"Quando eu não trabalhava com métodos ágeis, então eu pensava que os requisitos deveriam ser escritos "em pedra" e se o cliente deseja mudar por que ele não viu isso antes. Hoje eu vejo o feedback do cliente como algo positivo e o quanto antes ele acontecer melhor será para o projeto. Além disso, se o cliente não conseguiu perceber algo antes foi porque, justamente, ele não viu nada em funcionamento que o fizesse chegar a essa nova direção" (Mitsumasa, 15/10/2014, dp)*

Com isso, como Scrum é um método de desenvolvimento iterativo, onde a cada iteração a equipe e o cliente podem revisitar indiscriminadamente implementações parcialmente satisfatórias porém funcionais e melhorá-las, então é possível aprender e adaptar o software com base no *feedback* do cliente.

> *"Em métodos ágeis nós temos as sprints, onde as entregas são realizadas a cada quinze ou trinta dias. Então, o cliente consegue visualizar antes algum resultado, o que o deixa muito satisfeito. Além disso, ter algo pronto logo ajuda o cliente a refletir mais sobre o*

> *sistema, o que permite fazer ajustes mais rápidos, geralmente nas sprints seguintes." (Seika, 15/10/2014, dp)*
>
> *"(...) esse é um projeto estratégico e por conta disso algo mais rápido precisa ser mostrado, não dá para ficar esperando seis meses para que algo seja apresentado. Ou seja, é necessário apresentar algo rápido para produtos mais importantes."*

Essa adaptação, segundo Kiki, aumenta o entusiasmo do cliente com o que está sendo feito:

> *"O cliente que nós temos hoje é muito participativo e nos dá um feedback muito rico e construtivo. Ele constantemente elogia o trabalho da equipe. Ele está muito entusiasmado com aquilo que nós estamos entregando, então me parece que ele está satisfeito com o sistema." (Kiki, 15/10/2014, dp)*

Consequentemente, quando o cliente está satisfeito e os seus problemas estão sendo resolvidos, então a satisfação da equipe e o comprometimento do cliente, bem como, a confiança em métodos ágeis aumenta também.

> *"Os métodos ágeis conseguem trazer uma satisfação profissional e pessoal maior, bem como, não somos vistos como robôs que fazem código. Nós passamos a conhecer melhor o negócio do cliente e o seu produto, nós apresentamos o produto que está sendo desenvolvido, nós fazemos análises, etc. Ou seja, o papel do desenvolvedor em métodos ágeis é muito diferente do papel do desenvolvedor de outras metodologias." (Mitsumasa, 15/10/2014, dp)*
>
> *"O cliente parece acreditar mais nos resultados, com as entregas rápidas ele fica mais entusiasmado e continua colaborando. O cliente também percebe que ele precisa dar respostas mais rápidas e não pode demorar muito para responder os questionamentos da equipe. Ou seja, ele também precisa ser mais ágil." (Seika, 15/10/2014, dp)*

Com relação as entregas, embora o tempo de ciclo de cada novo incremento tenha sido reduzido para duas semanas, ela tem de esperar por mais algumas iterações para que algo mais completo seja colocado em produção.

> *"(...) o sistema é entregue para homologação ao cliente externo a cada quinze dias. Vale salientar que são necessárias mais algumas sprints para que o sistema fique completo o suficiente para entrar em produção. A mudança que aconteceu foi que reduzimos as iterações de trinta dias para quinze dias; isso tem se mostrado adequado para a atual configuração do projeto." (Kiki, 15/10/2014, dp)*
>
> *"A Empresa, geralmente, direciona para que o sistema em desenvolvimento seja colocado em produção o mais rápido possível (principalmente ao final de cada sprint), o que muitas das vezes não é o desejo do cliente." (Bian, 15/10/2014, dp)*

Por fim, o Projeto 4 corrobora com o Projeto 3, ao apontar a escrita de testes ágeis como a próxima barreira a ser superada.

> *"O desenvolvedor mudou a sua forma de trabalho e outras pessoas também, porém continuamos testando software como antigamente. Embora já tenha acontecido alguns avanços, precisamos incluir aspectos dos métodos ágeis no trabalho das pessoas que atuam com qualidade de software. Muitos dos pensamentos antigos sobre qualidade de software, incluindo métricas de detecção de defeitos, ainda continuam arraigados nas pessoas que trabalham com qualidade e teste de software." (Kiki, 15/10/2014, dp)*

> *"Hoje o nosso desafio na área de testes é escrever testes unitários automatizados, o qual não se ensina muito em cursos universitários. Então, o desafio é: como fazer o desenvolvedor escrever código e ao mesmo tempo testar aquele código que ele construiu? O ideal é que o desenvolvedor consiga construir testes de maneira automatizada. Pensamos em fazer isso nesse projeto a partir da próxima sprint, inclusive com integração contínua." (Kanon, 15/10/2014, dp)*

4.3.2.5 Informações sobre o Conhecimento e a Experiência em Métodos Ágeis

Novamente, a importância do apoio organizacional por meio de treinamentos e serviços de *mentoring* nos estágios iniciais de adoção de métodos ágeis é destacada na fala dos sujeitos.

> *"Como eu nunca tinha trabalhado com métodos ágeis antes, então eu tinha a percepção que não iria funcionar. Porém, à medida que eu fui trabalhando com essa abordagem e com apoio de uma consultoria e mais treinamentos (ministrados por empregados da própria Empresa e instrutores externos), eu fui percebendo que as coisas funcionam. Isso foi importante nesse processo de aprendizado dos métodos ágeis." (Mitsumasa, 15/10/2014, dp)*

> *"Eu tinha muito pouco conhecimento sobre métodos ágeis. O que eu fiz foi ler muito sobre o tema. A Empresa também contribuiu com investimentos em capacitação profissional dos empregados, incluindo cursos e treinamentos. Além disso, alguns serviços de consultoria foram contratados nos estágios iniciais. E recentemente, uma nova consultoria foi contratada para ajudar-nos a aprimorar nossa maneira de executar métodos ágeis, a partir das primeiras experiências." (Kiki, 15/10/2014, dp)*

Por outro lado, o aprendizado teórico precisa ser executado, o quanto antes, em casos reais, conforme observado na fala de Seika:

> *"Nós tivemos treinamento e consultoria, os quais foram muito importantes e são necessários. Porém, eles precisam ser executados na prática, o quanto antes. Não*

> *adianta fazer muitos treinamentos e nunca começar a usar. Tem que ser algo meio junto, o treinamento e a prática em casos reais." (Seika, 15/10/2014, dp)*

Além disso, o aprendizado e a adoção setorial de métodos ágeis contribuem para a troca de experiência entre as pessoas, conforme relatado por Bian:

> *"Quando comecei a trabalhar na Empresa, foi exatamente o momento em que a Empresa decidiu usar métodos ágeis, então tive a oportunidade de realizar vários cursos e treinamentos. Além disso, o nosso setor é todo ágil, o que facilita a troca de experiência." (Bian, 15/10/2014, dp)*

4.3.3 Caso Organizacional

A mudança organizacional na PROCERGS aconteceu por meio de uma via de mão dupla, de cima para baixo e de baixo para cima. Ou seja, a Empresa apontou para um novo caminho, mas ela não seria capaz de realizar uma proeza tão boa se não houvesse o comprometimento das pessoas. Bem como, o desejo das pessoas em adotar métodos ágeis não seria atendido se não houvesse o apoio dos níveis mais altos da Empresa.

O primeiro passo consistiu em executar um projeto-piloto (em 2012), o qual demonstrou que as coisas poderiam melhorar com métodos ágeis. O resultado positivo do projeto-piloto conscientizou as pessoas de que a mudança seria apropriada. Na sequência, a transição para métodos ágeis aconteceu dentro do próprio setor que executou o piloto, sendo gradativamente expandida para os demais setores de desenvolvimento de software da Empresa. As falas de Eurydice e Kiki demonstram como tudo aconteceu:

> *"Nós fizemos um grande movimento na Empresa para revisar a nossa metodologia de desenvolvimento de software. Ao todo, noventa pessoas participaram desse trabalho. E o resultado foi uma metodologia simplificada com base em métodos ágeis. Na sequência, executou-se um projeto-piloto no setor de fábrica de software com ajuda de um consultor. Esse projeto deu certo. Mais tarde, essa experiência foi repetida em todos os projetos do mesmo setor juntamente com a equipe de qualidade. Depois disso, essa experiência de métodos ágeis foi disseminada na Empresa, com ajuda de um consultor, para todas as áreas que atuam com desenvolvimento de software." (Eurydice, 14/10/2014, do)*

> *"Aqui na Empresa, Scrum nasceu de uma maneira diferente. Geralmente, Scrum nasce em células e depois sua adoção se expande para toda a organização (de baixo para cima). Porém, na nossa Empresa, a alta administração solicitou a adoção de métodos ágeis em toda a organização (de cima para baixo). Então, o Setor de Desenvolvimento de Software (SD1) foi escolhido como setor piloto para experimentar os métodos ágeis em projetos de desenvolvimento de software, o que foi muito bem recebido pelas*

> *pessoas que trabalham nesse setor. Assim, a partir dos resultados alcançados espera-se ampliar sua adoção em toda a Empresa, a qual já está acontecendo." (Kiki, 15/10/2014, dp)*

Desde então, de 553 sistemas cadastrados, de acordo com o indicador de métodos ágeis da Empresa, 252 sistemas são atendidos de maneira ágil, incluindo novos projetos e projetos de manutenção de software. Na opinião de Eurydice, a Empresa está feliz com a transição para métodos ágeis porque o tempo de entrega foi reduzido com o uso de um processo ágil.

> *"Eu acredito que nós tínhamos muita dificuldade em entregar software. Atualmente, estamos conseguindo entregar software de maneira mais rápida, com mais qualidade, com mais valor agregado, com menos custos, com menos retrabalhado e mais aderente as necessidades do cliente. Os resultados atuais são "fantásticos". Hoje em dia, a tecnologia está dominada e os clientes compraram a ideia dos métodos ágeis." (Eurydice, 14/10/2014, do)*

Como também, as pessoas estão felizes e seus princípios são transmitidos para os novos empregados.

> *"A Empresa pratica uma pesquisa de satisfação interna com os funcionários anualmente, então o setor que trabalhamos apresentou um diferencial em que as pessoas estão muito satisfeitas com a forma de trabalho. Além disso, quando um novo empregado entra para trabalhar no setor, então colocamos algumas questões como o trabalho colaborativo, o trabalho em equipe e a entrega de valor." (Eurydice, 14/10/2014, do)*

O princípio doze do manifesto ágil incentiva a inclusão de melhorias constantes no processo de produção de software, ao dizer que "em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo." [BBB+01]. Assim, os processos de mudanças contínuas passaram a figurar de forma corriqueira no cotidiano das pessoas e as têm estimulado a pensar em melhorias para alcançar melhores resultados. Mais recentemente, alguns projetos estão buscando garantir a qualidade no código desde o início, em vez de testar a qualidade no final. Isso quer dizer que o trabalho das pessoas que desenvolvem e executam testes, será prevenir defeitos, e não encontrá-los, o que exigirá maior conhecimento na escrita e automação de bons testes.

Muitas equipes param de melhorar no ponto em que começaram a funcionar como unidade, perdendo muitas das vantagens que o Scrum pode trazer [Coh11]. Para evitar esse problema, como dito anteriormente, a Empresa busca continuamente melhorias em seu processo por meio da experimentação em projetos-piloto.

> *“Este ano continuamos a melhorar a forma com que nós usamos métodos ágeis por meio da contratação de um serviço de mentoring em métodos ágeis, o qual foi muito rico. Passamos a considerar algumas coisas novas, tais como: pré-game, visão, mapping, discovery e delivery. Então, já fazem uns três meses que passamos a praticar essas melhorias no processo ágil de desenvolvimento no setor de fábrica. O próximo passo é coletar os resultados, avaliar e ampliar essa experiência para toda a Empresa.” (Eurydice, 14/10/2014, do)*

Do ponto de vista do aprendizado e compartilhamento do conhecimento, em vez de esperar passivamente o aprendizado ocorrer, a Empresa tem procurado novas maneiras de aprender e compartilhar conhecimento entre as pessoas, especialmente entre iguais (pessoas que executam tarefas parecidas, mas não estão próximas fisicamente e não atuam no mesmo projeto).

> *“(...) estamos procurando criar encontros para PO e ScrumMaster. O objetivo é que eles troquem experiências e prospectem melhorias no processo.” (Eurydice, 14/10/2014, do)*

Por fim, a fala de Eurydice, anuncia agilidade em todas as áreas da Empresa como o próximo grande desafio organizacional.

> *“Atualmente, a alta gestão está puxando para que toda a Empresa utilize métodos ágeis, não somente o desenvolvimento de software. Mas também, toda a parte de operação e infraestrutura, bem como, outros setores, tais como: a área administrativa, o departamento de gestão de pessoas, etc. Eu acredito que o próximo passo da Empresa, que deverá acontecer no próximo ano, será implantar métodos ágeis em nível de gestão também, porque o desenvolvimento foi todo, agora a empresa está buscando agilidade nas outras áreas. Na próxima semana, várias pessoas de diferentes setores estarão participando de uma apresentação sobre agilidade a nível empresarial, então a principal mudança para o próximo ano é escalar métodos ágeis para toda as áreas da Empresa.” (Eurydice, 14/10/2014, do)*

4.4 Consolidação dos Resultados Empíricos

Os quatro estudos de caso realizados evidenciaram diversos aspectos da ES e do desenvolvimento ágil de software, sendo consolidados na sequência.

4.4.1 Características das Organizações Públicas

A Tabela 15 apresenta algumas características das OP estudadas. Em seguida, essas características são discutidas detalhadamente.

Tabela 15 - Características das organizações públicas estudadas.

ORG. PÚBLICA	CARACTERÍSTICAS	
CNPTIA	Natureza jurídica	Empresa pública de direito privado
	Tempo de existência da empresa	29 anos
	Tamanho	109 pessoas, sendo 42 pessoas de TI
	Missão	TI para o setor agropecuário brasileiro
	Processo de desenvolvimento de software	Livre (há vários processos informais executados por diferentes equipes de desenvolvimento)
	Total de projetos executados com métodos e/ou práticas ágeis	Não mensurado, sendo o primeiro projeto em 2001
PROCERGS	Natureza jurídica	Sociedade anônima fechada - Economia Mista
	Tempo de existência da empresa	42 anos
	Tamanho	1.206 pessoas, sendo 960 pessoas de TI
	Missão	TI e Comunicação para a AP do Estado do Rio Grande do Sul
	Processo de desenvolvimento de software	Em otimização (o processo é periodicamente revisado e melhorado com base nas suas mensurações)
	Total de projetos executados com métodos e/ou práticas ágeis	252 projetos, sendo o primeiro projeto em 2012

Com relação a natureza jurídica, as organizações estudadas pertencem à AP indireta (pessoas jurídicas constituídas para o desempenho especializado de um serviço público), que mesmo sendo vinculadas à AP direta, usufruem de autonomia de gestão.

No setor privado, o tempo de vida das grandes corporações está ficando cada vez mais curto por conta das mudanças tecnológicas, onde grandes empresas são sufocadas pelas inovações feitas em outros lugares. Richard N. Foster [MIT13], ao observar o índice S&P 500[7], constatou que antigamente uma empresa conseguia permanecer no S&P 500 por 61 anos. Atualmente, a média é de apenas 18 anos. Ele atribui isso, essencialmente, a mudanças de tecnologia. Por outro lado, há indícios de que as empresas públicas de TI conseguem sobreviver ao tempo e as mudanças tecnológicas e de mercado. Por exemplo, a PROCERGS e o CNPTIA continuam sendo empresas públicas reconhecidas em sua área de atuação desde sua criação.

Com relação ao número de projetos executados com métodos ágeis na organização, é notável o avanço quantitativo na PROCERGS em tão pouco tempo. Por outro lado, o CNPTIA, embora protagonista em métodos ágeis no governo (RSL11), muito pouco avançou no âmbito organizacional, o que permite dizer que métodos ágeis não está implantado na organização.

[7] S&P 500 trata-se de um índice composto por quinhentos ativos (ações) qualificados devido ao seu tamanho de mercado, sua liquidez e sua representação de grupo industrial. É um índice ponderado de valor de mercado (valor do ativo multiplicado pelo número de ações em circulação) com o peso de cada ativo no índice proporcional ao seu preço de mercado.

4.4.2 Características dos Projetos

A Tabela 16 apresenta algumas características dos projetos estudados. Em seguida, essas características são discutidas detalhadamente.

Tabela 16 - Características dos projetos estudados.

PROJETO	CARACTERÍSTICAS	
CNPTIA (ECa01)	Tamanho da equipe	120 pessoas - 2 desenvolvedores - 1 gerente de projeto - 117 representantes do usuário
	Tempo médio de experiência com métodos ágeis	2 anos e meio
	Tempo de duração do projeto	36 meses
	Tempo de duração das iterações	Tamanho variável
	Clientes	Internos, porém distribuídos em várias Unidades da Empresa em todo o Brasil
	Característica do software	Sistema *web* para gerenciamento de dados de experimentos
	Orçamento	R$ 1.000.000,00
	Método de desenvolvimento	Algumas práticas de Scrum
CNPTIA (ECa02)	Tamanho da equipe	13 pessoas - 2 desenvolvedores - 1 gerente de projeto - 3 estagiários - 7 representantes do usuário
	Tempo médio de experiência com métodos ágeis	11 anos
	Tempo de duração do projeto	84 meses
	Tempo de duração das iterações	No início semanal, depois quinzenal, no final mensal e atualmente por demanda.
	Clientes	Internos, porém distribuídos em várias Unidades da Empresa em todo o Brasil
	Característica do software	- Sistema *web* para gerenciamento do acervo documental e digital - Migração da arquitetura cliente/servidor para *web* e integração com outros sistemas
	Orçamento	R$ 600.000,00
	Método de desenvolvimento	Algumas práticas de XP

Por razões meramente visuais, a apresentação da Tabela 16 continua na próxima página.

PROJETO	CARACTERÍSTICAS	
PROCERGS (ECa03)	Tamanho da equipe	6 pessoas - 1 PO - 1 analista de sistemas - 1 scrummaster / desenvolvedor - 2 desenvolvedores - 1 analista de qualidade
	Tempo médio de experiência com métodos ágeis	2 anos
	Tempo de duração do projeto	36 meses
	Tempo de duração das iterações	No início mensal e atualmente quinzenal
	Clientes	DETRAN/RS e o Setor de Trânsito da PROCERGS
	Característica do software	- Migração do modelo cliente/servidor para o modelo *web* - Inclusão de melhorias no processo de trabalho atual
	Orçamento	Não divulgado
	Método de desenvolvimento	Método de Desenvolvimento PROCERGS (MDP), baseado no Scrum
PROCERGS (ECa04)	Tamanho da equipe	6 pessoas - 1 PO - 1 analista de sistemas - 1 scrummaster / desenvolvedor - 2 desenvolvedores - 1 analista de qualidade
	Tempo médio de experiência com métodos ágeis	2 anos e meio
	Tempo de duração do projeto	30 meses
	Tempo de duração das iterações	No início mensal e atualmente quinzenal
	Clientes	DETRAN/RS e o Setor de Trânsito da PROCERGS
	Característica do software	- Migração do modelo cliente/servidor para o modelo *web* - Inclusão de melhorias no processo de trabalho atual
	Orçamento	Não divulgado
	Método de desenvolvimento	Método de Desenvolvimento PROCERGS (MDP), baseado no Scrum

Os levantamentos permitiram constatar que os orçamentos estão subdimensionados, tendo em vista que dotações para tecnologia da informação que dizem respeito a despesas com pessoal ativo que atua diretamente no projeto de desenvolvimento do produto de software não estão sendo alocadas como dotações para tecnologia da informação no orçamento geral do projeto. Isso vem acontecendo apesar da existência de modernas técnicas orçamentárias.

No âmbito da AP, não é incomum que a OP (desenvolvedor) também seja cliente para seu próprio esforço de desenvolvimento, nos quais enquadram-se os quatro casos estudados. Da mesma maneira, todos os projetos estudados são sistemas longevos, cujo o comportamento do software precisa ser mantido ao longo do tempo, os quais não são resistentes ao tempo. Assim, por conta das mudanças tecnológicas e de processos, eles precisam ser migrados para novas plataformas e evoluídos do ponto de vista de negócio. Os casos (ECa02) (ECa03) (ECa04) enquadram-se como exemplos de projetos de migração e integração de sistemas.

Outra característica comum encontrada foi o tempo de execução dos projetos que ocorre em vários anos, o que pode aumentar os custos para sua execução. Porém, com métodos ágeis, as entregas de software acontecem semanalmente ou mensamente, bem como, as implantações em produção acontecem em meses em vez de anos. Assim, os benefícios advindos de uma solução de TI podem retornar aos seus “investidores” o mais cedo possível. Além disso, constatou-se que os quatro projetos tiveram tempo e condições para experimentar, avaliar e usar várias práticas e tecnologias para o desenvolvimento de software.

Na PROCERGS, constatou-se pouco espaço individual e mais espaço coletivo e tudo bem próximo para o desenvolvimento de software, o que exige um exercício diário da essência da auto-organização. Por outro lado, no CNPTIA constatou-se mais espaço individual e menos espaço coletivo e pessoas distantes, o que exige a ajuda de ferramentas, e-mails e telefones para comunicação. O desafio nestes ambientes, embora com abordagens diferentes, é aprender a trabalhar juntos, bem como, garantir que as pessoas estarão disponíveis no momento em que elas são solicitadas. Conforme constatado nos casos (ECa01) (ECa02), um grande inibidor do trabalho em equipe é a alocação de pessoas concomitantemente em vários projetos.

Por último, três projetos estudados são de migração tecnológica, incluindo a migração de dados, migração do modelo cliente/servidor para um modelo web e integração com outros sistemas existentes (ECa02) (ECa03) (ECa04). De acordo com essas equipes, esse tipo de projeto é mais desafiador para métodos ágeis - especificamente para o princípio de entrega de valor mais cedo para o cliente - porque o novo produto de software precisa conviver em produção por algum tempo com o sistema existente, ou seja, o lançamento de software em produção com base em incrementos incluí desafios técnicos difíceis de resolver para manter ambos os sistemas em funcionamento na prática. Porém, essas três equipes recomendaram fortemente que essas questões técnicas sejam consideradas e tratadas desde o início do projeto para evitar surpresas e problemas futuros.

4.4.3 Características e Aspectos da Engenharia de Software

Conhecimento e experiência em gerenciamento de projetos foram encontrados como críticos para o desempenho e resultados dos projetos de software estudados, o que não deve ser uma descoberta nova. Em dois projetos estudados, esse aspecto da ES foi compartilhado entre algumas pessoas da equipe (*Product Owner*, Analista de Sistemas, Analista de Qualidade e o *ScrumMaster*), o que ilustra um exemplo de responsabilidade compartilhada pela equipe em vez de pessoas específicas, sendo completado com algumas capacidades organizacionais, incluindo serviços de *mentoring* e programas de treinamentos (ECa03) (ECa04). Por outro lado, nos outros dois projetos estudados, o papel específico do gerente de projeto ainda é obrigatório na formulação, aprovação e execução do projeto (ECa01) (ECa02), entretanto durante sua execução constatou-se um comprometimento compartilhado em relação ao cumprimento dos objetivos que a equipe aceitaria.

Com relação ao alinhamento com os objetivos de negócio, os quatro projetos estudados apresentaram uma nova solução de negócio em vez de apenas o fornecimento de um produto de software. Nestes casos, a gestão dos impactos organizacionais oriundas das mudanças de negócio foram gerenciadas desde o início do projeto. Eles constituíram comitês ou grupos - formados por clientes, representantes dos usuários e alguns projetos com pessoas de TI - para fornecer orientação e direção visando alcançar objetivos comuns e não coibir as iniciativas da equipe. Essa estratégia de governança do projeto mostrou-se adequada para aceitação efetiva do produto de software na organização. Porém, ela exige que os comitês ou grupos tenham participação ativa no processo de desenvolvimento e na resolução de problemas organizacionais, bem como, representem também os interesses dos usuários do sistema. Por outro lado, um projeto composto por um comitê formado por um número maior de pessoas encontrou dificuldades na tomada de decisão, o que ocasionou atrasos no andamento do projeto (ECa01). A partir disso, pode-se inferir que comitês ou grupos maiores são propensos a atrasos porque decisões consensuais e rápidas são mais difíceis de acontecer, principalmente quando não existe uma pessoa orientando o trabalho do comitê ou do grupo, ajudando e facilitando a tomada de decisão.

Com relação ao acompanhamento do projeto, dois projetos (ECa03) (ECa04) apresentaram evidências de como um gráfico de *burndown* bem feito oferece a percepção real de quanto o trabalho em andamento será concluído na data prevista ou não. A estratégia adotada em ambas as equipes consistiu em reestimar diariamente em horas o tempo restante para conclusão das tarefas, o que tornou o gráfico de *burndown* mais preciso. Essas equipes relataram o uso de ferramentas de gestão visual na forma de um quadro kanban para o acompanhamento do projeto também. Elas apontaram as reuniões

diárias como importantes para auxiliar o acompanhamento do projeto, bem como, as reuniões de revisão com o cliente para avaliar se o projeto está na direção certa. Por outro lado, as equipes dos projetos (ECa01) (ECa02) encontraram dificuldades para estabelecer uma rotina de reuniões diárias com os desenvolvedores por conta da participação deles em outros projetos de maneira simultânea. O acompanhamento do projeto foi realizado baseado em informações contidas em planilhas eletrônicas (ECa01) e ferramentas de apoio ao processo de desenvolvimento de software (ECa02).

Com relação ao desenvolvimento de software, os quatro projetos apresentaram evidências do uso da abordagem iterativa e incremental, estabelecendo ciclos de *feedback* bem definidos ajustando o software para criar uma interpretação ótima do produto. Além disso, três projetos (ECa02) (ECa03) (ECa04) preferiram incorporar aspectos de testes de software no início e durante as iterações em vez de deixá-los para o final. Dois projetos (ECa03) (ECa04) declararam estar praticando a criação de cenários de testes com base na técnica BDD com apoio do analista de qualidade, porém evidências da escrita de testes de unidade e automação de testes não foram encontradas para esses projetos, o qual foi declarado como desafio cultural na organização. Por outro lado, o projeto (ECa02) apresentou evidências na criação e execução de testes automatizados, abrangendo testes de unidade e testes funcionais; além disso, essa equipe apresentou bons conhecimentos em integração de código e lançamento de software em produção de maneira automatizada. Por último, nenhum dos projetos estudados apresentou evidências do uso de TDD.

Com relação a ambientes e ferramentas de apoio ao desenvolvimento de software, todas as equipes relataram a importância de ter ambientes específicos para o projeto, incluindo ambiente de desenvolvimento, testes, homologação, treinamento e produção. Duas equipes (ECa03) (ECa04) alegaram dificuldades em obter dados reais para desenvolvimento e testes, uma vez que muitos deles são sigilosos, não podendo ser disponibilizados para a equipe. Todas as equipes relataram o uso de ferramentas de apoio ao desenvolvimento de software, preferencialmente de código aberto para projetos escritos em Java. Uma equipe (ECa01) informou o uso frequente de ferramenta de gerenciamento de conteúdo para compartilhar informações com todos os membros do projeto, uma vez que eles estão distribuídos em várias regiões do Brasil. Essa mesma equipe apontou a importância do uso de outras formas de comunicação quando os clientes e usuários estão dispersos geograficamente, incluindo equipamentos de videoconferência, e-mail e telefone. Por último, não foram encontradas características plenas de multidisciplinaridade pessoais, pelo contrário, gerentes, arquitetos, analistas, desenvolvedores, projetistas de interfaces e testadores ainda são uma realidade em OP por opção.

4.4.4 Características dos Métodos Ágeis

Esta seção completa as consolidações anteriores com algumas informações identificadas nos estudos de caso.

4.4.4.1 Razões para Adoção de Métodos Ágeis

Como dito anteriormente, o CNPTIA foi protagonista na adoção de métodos ágeis na AP com o uso de XP em um importante projeto no início da década de 2000 (RSL11). Sua utilização foi motivada como resposta ao fracasso de um projeto anterior. O sucesso do projeto motivou seus protagonistas a disseminarem os princípios e as práticas de XP para outras pessoas, sendo algumas delas adotadas em alguns projetos subsequentes com alguns resultados positivos. Porém, a adoção de XP e de qualquer outro(a) método de desenvolvimento de software não é completo e unânime na Unidade.

Com relação a PROCERGS, a adoção de métodos ágeis começou em 2012, ou seja, bem depois de sua experimentação e avaliação pelo setor privado. Embora iniciada mais tarde, o uso de métodos ágeis para desenvolvimento de software está praticamente completo na Empresa. Sua utilização foi motivada por uma restruturação iniciada recentemente que visa tornar a Empresa mais eficiente e menos burocrática. Atualmente, a PROCERGS é uma boa referência na adoção de métodos ágeis em OP, principalmente para as companhias estaduais e empresas públicas de TI.

4.4.4.2 Benefícios na Adoção de Métodos Ágeis

Os quatro estudos são muito bons para evidenciar que o desenvolvimento ágil de software com base em entregas curtas de valor contribui para que os clientes mantenham-se entusiasmados e comprometidos com os resultados do projeto até o final, o que aumenta a confiança e a satisfação de todos com o trabalho realizado e com o sistema criado. Além disso, parece haver um fato novo, embora os projetos de software em OP ainda sejam de longa duração, a entrega em incrementos de valor para o cliente, bem como, a implantação de software em períodos de tempo menores cria maior capacidade de aceitação do sistema na Empresa, sendo mais resistentes às eventuais mudanças administrativas e políticas, típicas da AP.

4.4.4.3 Dificuldades, Problemas e Soluções na Adoção Métodos Ágeis

De acordo com as informações coletadas, pode-se dizer que as dificuldades do desenvolvimento ágil no CNPTIA estão centradas na alocação de pessoas para trabalhar em vários projetos concomitantemente, na dificuldade da alta administração em determinar

sua utilização de maneira institucional, na dificuldade de compartilhar conhecimento e informações entre as pessoas, na falta de treinamento aliado a *coaching in loco* e nos diferentes tipos de sistemas existentes. Apesar de existirem diversas maneiras de soluções aplicáveis visando minimizar os problemas na adoção de métodos ágeis em OP, algumas boas soluções foram encontradas na PROCERGS, onde as virtudes para sair melhor no desenvolvimento ágil começaram orientado principalmente à urgência da empresa e de algumas pessoas em usar métodos ágeis, no treinamento e *coaching* específico nas práticas de gestão de produtos e na formação de equipes alocadas para apenas um projeto, de preferência com pessoas que se conhecem a mais tempo.

4.4.4.4 Formação da Cultura Ágil

De acordo com as informações coletadas, ambas as empresas perceberam oportunidades externas e absorveram o uso de métodos ágeis em projetos-piloto com pessoas dispostas a experimentar o novo, sendo apoiadas pela alta administração. A partir daí suas experiências positivas foram refinadas e expandidas para outros projetos dentro da Empresa. Isso significa que a cultura ágil nasceu nessas duas empresas a partir da experimentação do objeto de estudo em subculturas[8]. Assim, de fato pode-se inferir que as OP aprendem e modificam-se a partir de si próprias. Por outro lado, a cultura ágil preconiza que as pessoas devem procurar, proativamente, novas maneiras de aprender e compartilhar conhecimento [Coh11]. Ou seja, outros conhecimentos precisam ser procurados deliberadamente em vez de esperar passivamente o aprendizado ocorrer para não prender-se as primeiras conquistas.

Neste sentido, a PROCERGS está procurando avançar na adoção de práticas de desenvolvimento ágil de software e estratégias de compartilhamento de conhecimento entre equipes. Por outro lado, o CNPTIA não tem conseguido estabelecer novos avanços permanecendo como está, inclusive com algumas dificuldades na formação de equipes e no compartilhamento de experiências entre as pessoas. Por esta razão, métodos ágeis não é percebido como um processo acabado que se finaliza quando uma determinada conquista é alcançada. Pelo contrário, métodos ágeis é algo que se processa de forma contínua e depende de um conjunto de fatores ligados as pessoas e ao ambiente onde ele está inserido para alcançar novos e melhores resultados continuamente. Isto sinaliza para o caráter interativo, contínuo e dinâmico dos métodos ágeis abstraído dos estudos de caso realizados.

[8] Subcultura é o conjunto de particularidades culturais de um grupo que se distancia do modo de vida dominante sem se desprender dele.

5 ARCABOUÇO DE RESULTADOS TEÓRICOS E EMPÍRICOS

Há muito a ser aprendido com outras experiências. Embora cada situação apresente características e desafios únicos, a experiência de um ministério, órgão ou governo pode subsidiar decisões com as quais outros se defrontaram. Observar como seus pares em outros países, desenvolvidos ou em desenvolvimento, abordaram questões que repercutem no âmbito de suas próprias experiências é um exercício bastante valioso para o governo [Gra10]. Seguindo essa linha de raciocínio, este capítulo se propõe a apresentar um arcabouço de resultados teóricos e empíricos sobre a adoção de métodos ágeis em OP construídos a partir das evidências encontradas. O arcabouço de resultados está organizado em quatro grandes categorias, incluindo as razões e os benefícios alcançados na adoção de métodos ágeis, os problemas e desafios enfrentados na sua adoção, algumas lições aprendidas e alguns métodos, práticas e métricas adotadas. Uma vez que métodos ágeis não abrange apenas os aspectos técnicos do desenvolvimento de software, mas também o ambiente de trabalho onde as pessoas estão inseridas [HaD08], outros três domínios de conhecimento serão utilizados para melhorar a classificação e visualização dos resultados encontrados. Vale salientar que o APÊNDICE A apresenta a lista completa de estudos teóricos e empíricos que compõem o arcabouço de resultados.

5.1 Três Domínios de Conhecimento de Métodos Ágeis

Ao pensar sobre os domínios de conhecimento de métodos ágeis no âmbito da Engenharia de Software, Hazzan & Dubinsky [HaD08] propuseram três perspectivas de informação, incluindo: a perspectiva humana, a perspectiva organizacional e a perspectiva técnica, as quais são denominadas como HOT (*Human*, *Organizational*, *Technical*). A perspectiva humana abrange os aspectos cognitivos e sociais, os processos de aprendizagem e o relacionamento interpessoal entre desenvolvedores, clientes e gerentes. Enquanto a perspectiva organizacional contempla os aspectos culturais, gerenciais e as políticas da organização. Por último, a perspectiva técnica inclui as práticas e os aspectos técnicos (*design*, teste, codificação, integração, entrega e manutenção de software). Esta proposta de subcategorias refletir-se-á sobre duas das quatro categorias principais do arcabouço de resultados, incluindo os benefícios alcançados e os problemas e desafios enfrentados.

5.2 Algumas Razões e Benefícios na Adoção de Métodos Ágeis em OP

Uma das maiores razões para a adoção de métodos ágeis são os benefícios que eles podem trazer para as OP, os quais são vistos como uma resposta ao histórico de fracassos de projetos de TI no setor público. Com relação aos benefícios alcançados na adoção de métodos ágeis em OP identificaram-se os seguintes aspectos (Figura 3).

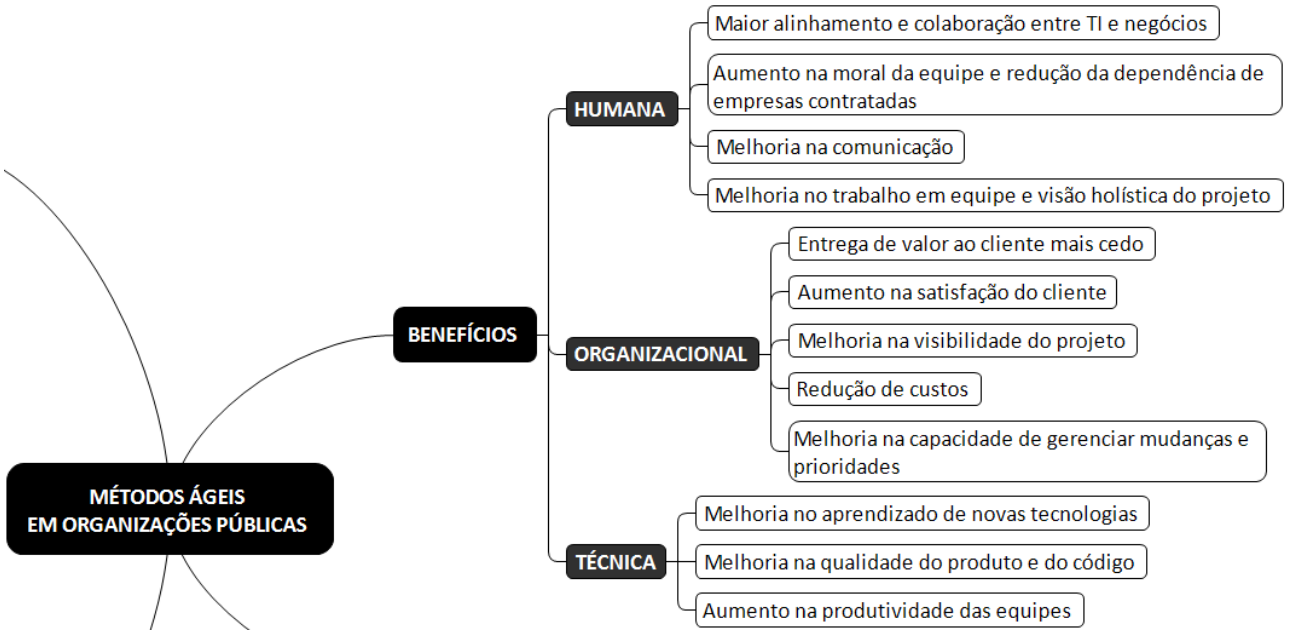

Figura 3 - Benefícios alcançados na adoção de métodos ágeis em OP.

A Tabela 17 apresenta a fonte de onde foi identificado cada aspecto.

Tabela 17 - Benefícios alcançados na adoção de métodos ágeis em OP.

PERSPECTIVA	ASPECTO	FONTES
HUMANA	Maior alinhamento e colaboração entre TI e negócios	(Teoria) (ECa01) (ECa02) (ECa03) (ECa04)
	Aumento na moral da equipe e redução da dependência de empresas contratadas	(Teoria)
	Melhoria na comunicação	(Teoria) (ECa03) (ECa04)
	Melhoria no trabalho em equipe e visão holística do projeto	(ECa03) (ECa04)
ORGANIZACIONAL	Entrega de valor ao cliente mais cedo	(Teoria) (ECa02) (ECa03) (ECa04)
	Aumento na satisfação do cliente	(Teoria) (ECa02) (ECa03)
	Melhoria na visibilidade do projeto	(Teoria)
	Redução de custos	(Teoria)
	Melhoria na capacidade de gerenciar mudanças e prioridades	(Teoria) (ECa01) (ECa02)
TÉCNICA	Melhoria no aprendizado de novas tecnologias	(Teoria) (ECa02)
	Melhoria na qualidade do produto e do código	(Teoria)
	Aumento na produtividade das equipes	(Teoria)

5.2.1 Maior Alinhamento e Colaboração entre TI e Negócios

Desde sempre tem se falado da necessidade de um alinhamento maior de TI aos negócios [Pre11]. Se os negócios estão, cada vez, mais mutáveis e incertos, então o trabalho de TI para estar mais alinhada só aumenta [Pre11]. Para isso, é necessário trabalhar em estreita colaboração com os parceiros de negócios, aproximando desenvolvedores e clientes, entregando o maior valor de negócio possível, no menor tempo e custo possíveis, ajudando-os a empregar o sistema de forma efetiva e entregando mais e melhores funcionalidades ao longo do tempo.

5.2.2 Aumento da Moral da Equipe e Redução da Dependência de Empresas Contratadas

Ainda que o assunto de contratação do desenvolvimento de software no governo não seja novo, o papel do governo nesse processo tem sido um dos temas mais discutidos nos últimos anos [Hei10] [Pys06]. Dentre os aspectos mais relevantes do governo, está a sua capacidade em lidar com novas metodologias e tecnologias para o desenvolvimento de software [Hei10]. Embora a participação de empresas da indústria de software seja vista com “bons olhos”, cabe ao governo, buscar a eficiência e a liderança na execução do desenvolvimento de software [Hei10], dois objetivos importantes para a promoção da autonomia, crescimento e desenvolvimento do governo. Quando o governo tornou-se totalmente dependente de empresas da indústria de software, então constatou-se uma desmotivação nos empregados de governo, por trabalhar grande parte do tempo em tarefas burocráticas de gerenciamento de empresas ou em atividades não relacionadas a implementação de software em si. Por outro lado, quando o governo assumiu a liderança e participou de todas as etapas do desenvolvimento de software com empresas contratadas, bem como, quando o governo adotou novos e mais modernos métodos de desenvolvimento de software (que inclui métodos ágeis) houve uma melhoria significativa na moral da equipe e na disciplina de ES, o que tem reduzido a dependência de empresas privadas.

5.2.3 Melhoria na Comunicação

O fato de métodos ágeis tornar o processo de desenvolvimento de software mais aberto para o cliente exige uma comunicação clara e objetiva, que consiste em fornecer informações certas às pessoas certas e no momento certo, para que elas possam utilizá-las proveitosamente, o que exige bons níveis de comunicação entre as pessoas [Sho08].

5.2.4 Melhoria no Trabalho em Equipe e Visão Holística do Projeto

Equipes conquistam resultados positivos não apenas pelos talentos individuais das pessoas, mas também por seus relacionamentos e pelo que as pessoas são capazes de alcançar juntas [Bec99]. As responsabilidades são compartilhadas pela equipe do projeto aumentando a colaboração para a entrega de funcionalidades em vez da conclusão de atividades individuais. Parte do mérito dos resultados positivos é devido a bons níveis de colaboração, comunicação e visão holística do projeto.

5.2.5 Entrega de Valor ao Cliente Mais Cedo

A medida de uma organização madura é a velocidade na qual consegue, confiável e repetidamente, executar seus processos principais [PoP11]. O processo principal no desenvolvimento de software é o processo fim a fim de tradução de uma necessidade do cliente na entrega de um produto [PoP11]. Dessa maneira, a maturidade de uma organização é medida pela velocidade com a qual ela consegue confiavelmente e repetidamente traduzir as necessidades de clientes em software funcional de alta qualidade que seja incluído em um produto que resolve o problema inteiro do cliente [PoP11]. Embora a velocidade para entrega de um produto para o cliente não seja um verdadeiro atributo de qualidade no sentido técnico, é uma medida de qualidade do ponto de vista de negócio [Pre11]. Do ponto de vista de negócio, dividir o software em um conjunto mínimo de funcionalidades úteis e entregar cada conjunto no seu devido tempo, o de mais alto valor primeiro, tem ajudado os clientes a fazerem melhor uma parte útil de seus trabalhos de maneira antecipada, bem como, quando esses conjuntos de funcionalidades úteis começam a produzir mais cedo um retorno de investimento, sistemas anteriores têm sido descontinuados, gerando uma economia com a sua manutenção; enquanto mais valor são entregues ao cliente mais cedo, ganhos em produtividade com reduções equivalentes no custo têm acontecido. Outrossim, o lançamento de software de valor mais cedo tem contribuído para aumentar o comprometimento das OP na utilização daquilo que está sendo feito, sendo mais resistente a eventuais mudanças administrativas e políticas, as quais são comuns na AP (ECa03) (ECa04).

5.2.6 Aumento da Satisfação do Cliente

O resultado de um alinhamento maior de TI aos negócios e de uma entrega rápida de valor ao cliente mais cedo tem sido uma maior satisfação do cliente com o produto de software desenvolvido, aumentando a confiança e o comprometimento das pessoas com os resultados do projeto durante toda a sua execução. Além disso, ao tornar o processo de

desenvolvimento mais aberto para o cliente, a implementação de funcionalidades tem se tornado algo constante e transparente, permitindo acompanhar o progresso do desenvolvimento do produto ao longo do tempo. Vale salientar que estas funcionalidades têm sido adicionadas com maior qualidade, de maneira mais rápida, sem retrabalhos, com menos defeito, sem atrasos e mais aderente as necessidades do cliente.

5.2.7 Melhoria na Visibilidade do Projeto

A essência da visibilidade do projeto são as informações. Elas devem estar disponíveis para todos os membros da equipe e devem comunicar claramente o andamento do projeto em relação aos seus objetivos, problemas e necessidades reais da equipe, pois a visibilidade dessas informações permite que a gestão e a equipe identifiquem questões importantes e resolvam problemas rapidamente, o que melhora a confiança das pessoas envolvidas no desenvolvimento de software.

5.2.8 Redução de Custos

Funcionalidades não utilizadas requereram provavelmente código, testes e documentação desnecessária [PoP11]. Equipes ágeis têm menos probabilidade de desenvolver funcionalidades desnecessárias por conta do maior comprometimento do cliente com o que está sendo feito, a qual é uma crítica comum feita ao processo de desenvolvimento dito tradicional, que quando o software é entregue, os usuários não precisam mais da funcionalidade que está sendo fornecida [Coh11].

5.2.9 Melhoria na Capacidade de Gerenciar Mudanças e Prioridades

O desenvolvimento iterativo sugere que novos incrementos de software sejam entregues em intervalos de duração o mais curto possível, o que permite realizar mudanças de prioridades nas iterações seguintes. Durante o planejamento da iteração, o cliente pode mudar suas prioridades com impacto reduzido na produtividade da equipe porque planeja-se detalhadamente apenas aquilo que está mais próximo de ser feito [Gom13]. A equipe necessita estar aberta a essas mudanças prioridades pois o objetivo é ter um software relevante ao final de cada iteração que possa ser considerado como progresso.

5.2.10 Melhoria no Aprendizado de Novas Tecnologias

A utilização de determinadas práticas ágeis de desenvolvimento de software, tais como: TDD, refatoração e integração contínua unidas a visão de qualidade proposta pelos métodos ágeis, contribuem para a incorporação de novas tecnologias no âmbito do projeto.

O aprendizado destas e de outras tantas tecnologias necessárias para a construção do produto de software tem acontecido através da troca colaborativa de aprendizado.

5.2.11 Melhoria na Qualidade do Produto e do Código

A baixa qualidade do produto afeta a satisfação dos clientes e a baixa qualidade de código dificulta a habilidade de adicionar novas funcionalidades ao sistema em tempo oportuno. Como visto anteriormente, TDD e *Design* Incremental são abordagens para melhorar a qualidade do código no nível de classes e métodos individuais e a arquitetura do projeto de software, enquanto ciclos de *feedback* frequentes com o cliente são fundamentais para ajustar o software na direção certa melhorando a qualidade do produto.

5.2.12 Aumento na Produtividade das Equipes

Produtividade tem sido um tema intensamente estudado ao longo dos anos havendo diferentes sentidos para o termo [Car14]. A noção de produtividade de software para equipes ágeis está mais relacionada à capacidade de entregar funcionalidades úteis para o cliente o quanto antes do que com quantidade de código [Car14]. Se a equipe e os clientes colaboram mutuamente para atender aos objetivos de negócio, então é de se esperar que sistemas mais aderentes as necessidades dos clientes sejam entregues, o que a pode tornar mais produtivas. Porém, nas pesquisas supracitadas (RSL03) (RSL16), os resultados afirmam ter alcançado maior produtividade em termos quantitativos com métodos ágeis.

5.3 Alguns Problemas e Desafios na Adoção de Métodos Ágeis em OP

Em alguns casos, a imagem bastante otimista no nível teórico dos métodos ágeis pode ser contraposta por uma realidade prática dominada por desafios, dificuldades e problemas concretos. Com relação aos problemas e desafios enfrentados identificaram-se os seguintes aspectos (Figura 4).

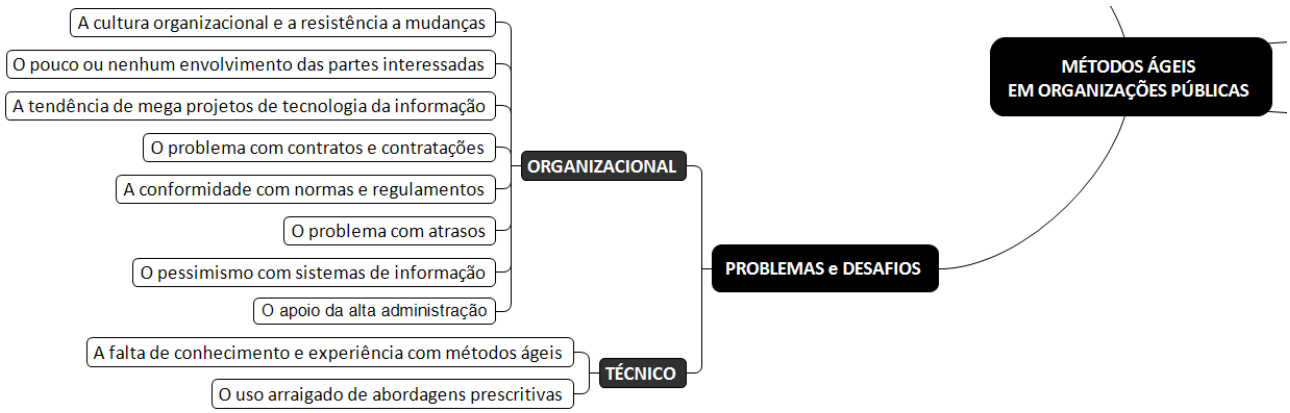

Figura 4 - Problemas e desafios enfrentados na adoção de métodos ágeis em OP.

A Tabela 18 apresenta a fonte de onde foi identificado cada aspecto.

Tabela 18 - Problemas e desafios enfrentados na adoção de métodos ágeis em OP.

PERSPECTIVA	ASPECTO	FONTES
ORGANIZACIONAL	A cultura organizacional e a resistência a mudanças	(Teoria)
	O pouco ou nenhum envolvimento das partes interessadas	(Teoria) (ECa01)
	A tendência de mega projetos de tecnologia da informação	(Teoria)
	O problema com contratos e contratações	(Teoria)
	A conformidade com normas e regulamentos	(Teoria) (ECa01)
	O apoio da alta administração	(Teoria)
	O problema com atrasos	(Teoria) (ECa01) (ECa02)
	O pessimismo com sistemas de informação	(ECa01)
TÉCNICA	A falta de conhecimento e experiência com métodos ágeis	(Teoria) (ECa01) (ECa02) (ECa03) (ECa04)
	O uso arraigado de abordagens prescritivas	(Teoria)

5.3.1 A Cultura Organizacional e a Resistência a Mudanças

Segundo Kent Beck [Bec99], um dos signatários do Manifesto Ágil, a maior barreira para o sucesso de XP é a cultura organizacional. Isso é uma verdade para métodos ágeis como um todo, e não apenas para XP. De acordo com o relatório "*9th ANNUAL State of Agile Development Survey*" [Ver15], capacidade de mudança da cultura organizacional continua sendo a maior barreira para a adoção de métodos ágeis, com mais da metade dos entrevistados citando esse aspecto como maior problema. Igualmente, o relatório "Métodos Ágeis no Brasil: Estado da Práticas em Times e Organizações" [MSC+12] obteve o mesmo resultado.

A cultura organizacional possui várias faces [Koc04], sendo algumas delas apresentadas na sequência. Por exemplo, qualquer projeto executado em estruturas hierárquicas e burocráticas, típicas de governo, terá conflitos com equipes que preferem práticas de trabalho flexíveis e colaborativas [Iil10]. O caso do(a) *UK Regional Government Departament* (RSL07) apoia esse pensamento, onde aspectos inerentes da cultura organizacional impediram que as partes interessadas conseguissem cooperar e colaborar com a equipe de desenvolvimento, gerando um impacto negativo sobre o progresso do projeto e a confiança da equipe. A "cultura da culpa", diminui a capacidade para a tomada de decisão sobre os objetivos do negócio, a qual afetou o trabalho dos desenvolvedores que necessitavam de uma priorização das atividades para cumprir os prazos de desenvolvimento, provocando atrasos e a perda da confiança no trabalho flexível e colaborativo.

Outra cultura que não contribui para métodos ágeis é o pensamento determinístico, que começa elaborando uma definição completa do produto e então cria uma realização a partir desta definição [PoP11]. O caso do(a) *United States Strategic Command* (RSL01) mostrou que, embora a adoção de métodos ágeis tenha sido bem-sucedida, várias partes interessadas ainda continuavam acreditando que com o planejamento prévio e detalhado melhores resultados teriam sido alcançados. Isso aconteceu apesar do fato de que as percepções da qualidade do produto resultante eram elevadas, e que a entrega frequente de novas funcionalidades, a cada duas ou três semanas em vez de dois meses, foi muito apreciada. Ou seja, qualquer projeto que tente apontar a direção certa logo de início terá conflitos com equipes que preferem ir acertando a direção continuamente [Bec99].

Uma cultura pronta para mudanças, disposta a encarar o novo, com coragem, descobrindo novos caminhos e novas soluções é a condição ideal para métodos ágeis começar (RSL01) (RSL07). Porém, em alguns casos, OP preferem abordagens que estão mais estreitamente alinhadas aos seus processos existentes, permanecendo como está, desertando da transformação (RSL08). Assim, em muitas organizações as pessoas sabem e dominam apenas o que elas estão acostumadas. Por isso, a personalização de métodos ágeis, bem como, a não experimentação de determinadas práticas ágeis onde elas não são compatíveis com a cultura organizacional, não é incomum em OP (RSL01). Por outro lado, uma abordagem mais flexível, na qual a equipe aprende métodos ágeis e, ao mesmo tempo, encontra respostas para os problemas culturais da organização a partir da experimentação do método em projetos-piloto tem alcançado impactos positivos (RSL03) (RSL16) (ECa03) (ECa04).

Um bom exemplo que ilustra essa situação foi encontrado no(a) centro de pesquisa *NASA Langley Research Center* (Langley) (RSL16), onde pesquisadores tiveram que lidar com várias barreiras culturais antes de aplicar XP, incluindo: (1) o valor de negócio de curto prazo e os objetivos de pesquisa de longo prazo; (2) atendimento a normas e regulamentos; (3) valorização do desempenho individual acima do desempenho de equipes; (4) salas individuais de trabalho; (5) equipe de duas pessoas. Esses obstáculos, embora difíceis de superar, foram “vencidos” pelos pesquisadores sem alterar a essência das práticas de XP. O resultado com XP foi aproximadamente duas vezes mais produtivo do que projetos semelhantes executados anteriormente. Esse exemplo, reforça o pensamento de que a adoção de métodos ágeis em OP não é apenas uma questão de treinamento e capacitação; mas, de aprendizado contínuo através da realidade do trabalho, visando produzir uma mudança que supere as barreiras.

5.3.2 O Pouco ou Nenhum Envolvimento das Partes Interessadas

Projetos de governo precisam do apoio e participação ativa das partes interessadas (incluindo os donos do produto, clientes e/ou usuários finais) [Wer12]. Quando isso aconteceu, as melhores soluções de software foram criadas, inclusive com uma maior aceitação na OP e na sociedade. Normalmente, as partes interessadas precisam investir tempo e recursos para tornarem um projeto viável. Porém, muitos projetos podem tornarem-se inviáveis quando as partes interessadas não são aliviadas de seus trabalhos diários para participar ativamente do projeto de desenvolvimento; quando falta o conhecimento necessário do dono do produto para o desenvolvimento do software; quando existem dificuldades de comunicação entre a equipe de desenvolvimento de empresas contratadas com o dono do produto; quando uma hierarquia burocrática de comando e controle é colocada em prática, não reconhecendo a necessidade de coordenação entre as partes interessadas potencialmente divergentes. Historicamente, o resultado do pouco ou nenhum comprometimento e/ou conhecimento das partes interessadas em projetos de software em OP, tem levado a construção de sistemas que não atendem as expectativas da sociedade e nem da própria AP, ou seja, são sistemas inservíveis [Mid99] [Wer12].

5.3.3 A Tendência de Mega Projetos de Tecnologia da Informação

De acordo com o estudo de Kipp *et al.* [KRW08], as duas características principais que definem Mega Projetos de TI são: (1) eles envolvem o desenvolvimento de grandes e extensos artefatos de TI; 2) eles exigem uma grande quantidade de recursos, em termos de tempo para sua execução, número de pessoas/departamentos/órgãos envolvidos para seu desenvolvimento, recursos financeiros, etc. Assim, eles são complexos, tanto do ponto de vista técnico, como também, do ponto de vista organizacional e social. Secundariamente, outra característica importante é que eles são executados principalmente na AP e colecionam problemas associados com extrapolação de gastos públicos, atrasos significativos no cronograma, problemas no gerenciamento de projetos, o que na maioria das vezes resulta no fracasso do projeto, despertando maior atenção da mídia. Esses projetos, geralmente, são incorporados em uma rede de interesses públicos, onde os objetivos muitas vezes complexos e interdependentes se opõem uns aos outros. Isso é particularmente desafiador, pois os objetivos podem mudar ao longo do tempo e as decisões políticas de várias áreas podem impactar o andamento do projeto negativamente.

Por conta disso, projetos maiores e complexos, muitas vezes, não conseguem identificar os principais riscos para o seu desenvolvimento [Wer12]. Eles possuem riscos que são mais difíceis de prever com impactos maiores, caso aconteçam. O efeito é um

provável aumento exponencial no custo e no prazo à medida que o tamanho do projeto aumenta. Foi o que aconteceu em alguns grandes projetos de TI da AP, incluindo as fases de insucesso do projeto *Sentinel* (RSL04) (ROE03) e o projeto *Universal Credit* do Reino Unido (RSL14). Por outro lado, métodos ágeis tendem para construir pequenos incrementos de trabalho em tempos regulares com equipes menores; ou seja, projetos ágeis são menores com menos custos associados.

5.3.4 O Problema com Contratos e Contratações

O governo precisa saber exatamente quando o projeto será entregue e quanto custará aos cofres públicos. Para isso, durante anos, a única estratégia adotada pelo governo foi ter como base um escopo suficientemente detalhado e documentado com estimativas mais próximas da certeza para tal escopo. Com essa abordagem, o governo procurou prever exatamente o que receberia, quando e com que preço. A princípio, essa abordagem parecia simplificar o processo de compras para contratação de empresas da indústria de software [Wer12]; os marcos obrigatórios para conclusão de cada fase aparentavam dar segurança sobre o progresso do projeto; a definição detalhada do escopo parecia garantir o êxito que o governo esperava, isentando-o no futuro, de eventuais discordâncias ou desvios sobre o que foi contratado. Porém, esse cenário previsível e com poucas mudanças, em muitos dos casos, não tem se configurado nas OP; pelo contrário, a natureza, cada vez mais, dinâmica e complexa dos negócios e da sociedade provocam mudanças nas OP, sendo desejável que projetos absorvam essas mudanças naturalmente.

Por conta disso, a previsibilidade sobre o escopo é inviável na maioria dos casos, e fixar o escopo pode prejudicar o governo em situações de mudanças em seus processos de negócio. A insistência nessa abordagem tem aumentado o risco de enfrentar e resolver o problema errado, além de inserir processos onerosos na tentativa de fechar o escopo, os quais dificultam a absorção de mudanças [Wer12]. Por essas razões, diversos projetos de “escopo fixo” em OP fracassaram, onde soluções foram entregues sem resolver os problemas reais dos usuários, com pouca ou nenhuma contribuição para os objetivos propostos no projeto [KRW08] [Mcd10] [Midd99] [Wer12].

Esse problema é agravado quando faltam incentivos para desenvolver projetos com base na confiança mútua entre o governo (contratante) e empresas contratadas, quando licitações públicas são pensadas com base na abordagem BDUF (típicas da União Europeia) e quando há falta de conhecimento e condições legais para criar e estabelecer contratos ágeis [Wer12]. Além disso, se uma OP estiver executando um projeto de desenvolvimento de software internamente, muitas vezes, ela tende a aplicar as mesmas

práticas de gestão como se estivesse terceirizando o desenvolvimento de software junto as empresas contratadas. As políticas organizacionais muitas vezes criam relações contratuais entre os departamentos dentro de uma mesma OP, o que pode produzir os mesmos efeitos dos contratos tradicionais de “escopo fixo” [Wer12].

Uma solução interessante para o problema de contratos de “escopo fixo”, conhecida como contrato de “escopo negociável”, quando combinada com métodos ágeis, tem emergido na AP com resultados positivos (RSL04) (ROE01) (ROE03). Nesse modelo, o escopo é negociado e discutido diversas vezes ao longo do projeto; ele não está vinculado ao contrato, assim, não há risco da empresa contratada deixar de cumprir com o contrato por um erro de interpretação da equipe ou alterações no escopo efetuadas pelo governo (cliente) ao longo do projeto [Des14]. Entretanto, projetos ágeis com escopo negociável dependem de decisões com base na confiança mútua, sendo bem adequados para projetos internos [Wer12]. Assim, contratar o desenvolvimento ágil quando a solução está sob a liderança de empresas contratadas externamente parece não proporcionar benefícios reais [Bec99] [Wer12]; pelo contrário, aumenta a dependência do governo em relação a indústria de software, bem como, diminui a competência do governo em manter sistemas críticos internamente, o qual deveria deter a competência em lidar com as metodologias e tecnologias utilizadas, tal como, a compreensão associada ao impacto em custo e cronograma [Bec99] [Hei10].

5.3.5 A Conformidade com Normas e Regulamentos

Assim como, em diversas organizações, as OP são pautadas por burocracias que incluem normas, procedimentos e regulamentos. O desejo de usar métodos ágeis e manter a conformidade com exigências burocráticas de governo pode parecer um conflito cultural, mas para alguns não é. Métodos ágeis têm atendido aos requisitos de Gerenciamento do Valor Agregado (*Earned Value Management* - EVM) (RSL04) (RSL06). Eles, também, mostraram-se compatíveis com o *PRojects IN Controlled Environments* (PRINCE2) (ROE04) e o *Capability Maturity Model Integration* (CMMI) (RSL09). Somente em dois casos, a renúncia das normas e regulamentos vigentes foi necessária para viabilizar a adoção de métodos ágeis em projetos de software em OP (RSL16) (RSL17). Assim, em teoria e na prática, tem sido possível atender diversas obrigações “legais” de governo com métodos ágeis. Porém, em algumas situações, as exigências de governo direcionam para o uso do Modelo Cascata e a abordagem BDUF sem qualquer nuance, reforçando a cultura prescritiva no governo em vez de ajudá-lo a se tornar mais ágil [Wer12]. Este é o caso em que o *US Office of Management and Budget* (OMB) e o GAO forçaram uma abordagem

prescritiva no FBI, a qual desencadeou uma grande contratação no desenvolvimento de software que falhou (ROE03).

Vale salientar que no Estudo de Caso 1 (ECa01) executado nesta pesquisa, não estava claro para os membros do projeto quais eram exatamente os documentos que precisavam ser mantidos para fins de auditoria na AP brasileira:

> *"(...) ainda não sabemos quais documentos são necessários (e se realmente são necessários) para fins de auditoria em empresas públicas brasileiras." (Shiryu, 15/09/2014, dp)*

5.3.6 O Apoio da Alta Administração

O apoio da alta administração é fundamental para a adoção de métodos ágeis em OP. Equipes ágeis dependem desse suporte gerencial para superar os obstáculos organizacionais e diminuir a resistência de mudanças para alcançar resultados positivos (RSL03) (RSL07) (RSL08) (RSL16) (RSL17) (ECa03) (ECa04), sendo confirmado também na pesquisa realizada na Embrapa Informática Agropecuária (item 4.2.3). Quando esse apoio institucional não aconteceu, então iniciativas de adoção de métodos ágeis foram abandonadas prematuramente nas primeiras situações contrárias (RSL08).

5.3.7 O Problema com Atrasos

Projetos de governo são conhecidos como grandes e duradouros [KRW08] [Som11]. Tudo que não funciona corretamente nesse processo acaba provocando um atraso de tempo. Esperar pela disponibilidade de desenvolvedores que estão trabalhando em outros projetos simultaneamente é uma grande causa do desperdício proveniente do atraso (ECa01) (ECa02); desenvolvedores sobrecarregados rapidamente tornam-se gargalos e limitam o progresso do projeto (RSL01) (RSL08) (ECa01) (ECa02). Pouca ou nenhuma responsabilidade na tomada de decisão aparece tomando mais tempo (RSL07). Organizações com estruturas hierárquicas e burocráticas retardam as coisas (RSL08). Partes interessadas com dedicação parcial fazem as coisas irem muito devagar (RSL07). Vários comitês com opiniões divergentes que prolongam a tomada de decisão aparecem tomando mais tempo ainda (ECa01). Construir o produto errado adiciona um enorme atraso (ROE03). Coisas demais no processo retardam o desenvolvimento (ROE01). Dependência de equipes externas ocasionam mais atraso ainda (ROE05). Ou seja, fatores que geram atrasos não são bons para criar agilidade; equipes menores, colocalizadas realizando iterações curtas com *feedback* frequente dos clientes podem diminuir drasticamente os atrasos, ao mesmo tempo que aumentam a qualidade das decisões (RSL04) (ROE03)

(ECa02); no entanto, esta não é a única abordagem para reduzir os atrasos, não importando muito onde os membros da equipe estão localizados fisicamente, o importante é garantir que o conhecimento esteja disponível exatamente quando e onde ele for necessário [PoP11].

5.3.8 O Pessimismo com Sistemas de Informação

Mais do que a visão positiva encontrada em grande parte da literatura ágil, esta pesquisa encontrou indícios que sugerem um pessimismo quando se trata do desenvolvimento de Sistemas de Informação em OP. No Estudo de Caso 1 (ECa01), Shiryu relatou a existência de empregados públicos desconfiados e céticos em relação a TI.

> *"A empresa é muito grande e complexa. Há pessoas que acreditam e pessoas que não acreditam que uma solução de TI irá ajudá-las a melhor executar o seu trabalho. Então, o maior desafio ainda é cultural, no sentido de mostrar para as pessoas a segurança e os benefícios oriundos de uma solução de TI." (Shiryu, 15/09/2014, dp)*

Pensando nisso, Shiryu procurou maximizar a participação das Unidades da Empresa em todas as decisões, visando aumentar o nível de aceitação da solução de software em toda a Embrapa.

> *"(...) nesse projeto existe uma questão séria de regulamentação do uso dos dados armazenados. Existem dados que não podem ser visíveis para todos, enquanto não forem autorizados por procedimentos formais. Se o usuário não sentir confiança no sistema, então ele não irá utilizá-lo. Por isso, esse sistema precisa ser de altíssima qualidade e com grande envolvimento das Unidades na concepção de como "as coisas devem ser". As Unidades participantes precisam se sentir "donas do negócio". Isso aumenta as chances de adoção da solução de software na empresa." (Shiryu, 15/09/2014, dp)*

5.3.9 A Falta de Conhecimento e Experiência com Métodos Ágeis

A falta de conhecimento e experiência dos servidores públicos (incluindo desenvolvedores, gerentes, clientes e usuários finais) e de empresas contratadas da indústria de software para executar o desenvolvimento de software de maneira diferente, de modo a entregar serviços melhores e mais ágeis tem sido uma questão latente nas iniciativas de adoção de métodos ágeis em OP. Há várias características sobre métodos ágeis que são novas e desconhecidas para muitos membros da equipe. Eles têm exigido uma mentalidade muito diferente do que as pessoas estão acostumadas. A experiência de abordagens baseadas em modelos prescritivos é completamente diferente do que realizar

reuniões diárias, trabalhar com *timebox*[9], entregar em pequenos incrementos de software, bem como, manter histórias de usuário e *backlogs* de produto. Por exemplo, donos do produto precisam construir os requisitos do sistema de maneira incremental ao invés de produzir especificações detalhadas, à priori; além disso, em vez de completar essa tarefa em duas semanas, por exemplo, agora uma iteração inteira é de apenas duas semanas; ou seja, um ciclo completo de detalhamento dos requisitos, codificação e testes acontece dentro dessas duas semanas. Ademais, equipes enfrentaram dificuldades em convencer os clientes a realizar publicações parciais do sistema em produção, mesmo que agregassem valor ao negócio, evidenciando uma falta de confiança dos clientes em soluções iterativas (RSL03) (ECa03).

Além disso, em algumas situações, o conhecimento técnico e as habilidades da equipe de desenvolvimento não foram suficientes para assimilar e executar diversas práticas ágeis. Alguns estudos relataram que:

- Equipes não reconheceram valor na Programação em Par (RSL01) (RSL11);
- Equipes tiveram dificuldades em escrever testes de unidade (RSL03) (ECa03);
- Equipes não adotaram TDD por falta de exemplos reais (RSL11);
- Equipes não encontraram benefícios na escrita de testes de aceitação com base em interfaces *web*, por conta de alterações frequentes (RSL02);
- Equipes tiveram dificuldades em colaboração e dificuldade em se adaptar ao trabalho auto-gerenciado [NAO12];
- Equipes tiveram dificuldades em estabelecer entregas de tempo fixo (ECa01);
- Equipes não conseguiram realizar entregas de software com funcionalidades relevantes em menos de quatro semanas (ROE02) (ECa01);
- Equipes não sabiam como documentar processos de negócio no desenvolvimento ágil (ECa01) (ECa03);
- Equipes tiveram dificuldades em se comprometer com novos requisitos e suas alterações, bem como, em gerenciar requisitos iterativamente [NAO12];
- Desenvolvedores tiveram dificuldades com o trabalho intermediado pelo *Product Owner* (ECa04).

Um risco comum causado pela falta de experiência com métodos ágeis é que a equipe quando encontra uma prática ágil difícil de aplicar, ela tenta alterar o método ágil para seu contexto específico (o que pode desvirtuar a essência do método), em vez de

[9] Timebox é uma técnica de gerenciamento de tempo comum no gerenciamento de projetos (típica de projetos de software), aonde o plano é dividido em períodos de tempo fixo, sendo que cada parte possui os itens de trabalho que serão entregues e a data de entrega.

aprender mais sobre os benefícios da prática visando mudar a forma como a equipe trabalha. Por isso, é importante ter o acompanhamento de pessoas experientes nos estágios iniciais de adoção de métodos ágeis; caso contrário, as iniciativas para sua adoção irão falhar, foi o que aconteceu em um departamento de Telecomunicações e TI do governo dos Emirados Árabes Unidos (RSL08).

Finalmente, é importante destacar que, em alguns casos, inserir novas práticas de trabalho, alterar o gerenciamento de projetos e inserir novas técnicas de programação foram difíceis de implementar. Isso significa que métodos ágeis requerem mais prontidão para mudanças, mais disciplina, mais maturidade e conhecimento da equipe.

5.3.10 O Uso Arraigado de Abordagens Prescritivas

Embora vários aspectos do desenvolvimento adaptativo têm sido defendidos e valorizados pela comunidade de ES durante anos, ainda existe um viés em direção a abordagens prescritivas em OP, incluindo o Modelo Cascata, a abordagem *Big Design Up Front* (BDUF) e entregas *big-bang*. Abordagens de cima para baixo são muito atraentes para organizações hierárquicas, típica de governo [Wer12]. Consultores de grandes OP são motivados a dizer ao cliente o que eles “querem”, especialmente se esta abordagem resulta em estudos de estratégias e análises prolongadas [Wer12]. O uso de abordagens que produzem um conjunto abrangente de documentação parece ser atraente para as organizações com maior burocracia, o que é uma característica de OP [Wer12].

As abordagens prescritivas mencionadas anteriormente corroboram com o modelo tradicional de planejamento, onde uma Estrutura Analítica de Projeto (EAP) é seguida em um roteiro prescrito de atividades, sendo acompanhado pelo controle do término de cada uma delas, o que muitas vezes, não traz o resultado esperado pelo cliente [MiR14]. Esse modelo baseia-se no pensamento: “se fazer tudo certo na primeira vez, então o desenvolvimento será mais rápido, melhor e mais barato”, sendo recomendado para situações onde os requisitos são previsíveis e estáveis (com poucas ou nenhuma alteração de escopo nas etapas seguintes) [Hig02]. Porém, esse pensamento é incoerente com a realidade, onde mudanças acontecem com frequência [Hig02]. O maior agravante disso é que o contato mais próximo do cliente com o produto começa quando a EAP é finalizada, e ele só irá receber outro *feedback* do produto apenas ao final, quando realizada a entrega do seu produto [MiR14].

Essa situação foi evidenciada no caso do projeto *Sentinel* (ROE03) (RSL04), onde o plano inicial do projeto mostrou-se irrealista, bem como, o desenvolvimento e a entrega de software executadas em grandes períodos de tempo, por empresas contratadas, não

conseguiu atender as necessidades dos usuários. A solução encontrada pelo FBI envolveu a participação de desenvolvedores de software de governo em atividades de codificação e uma mudança na estratégia de gerenciamento do projeto; o FBI assumiu literalmente a liderança do projeto e adotou o método Scrum para ajudá-lo nos aspectos gerenciais do produto de software; ele substituiu um extenso plano de projeto por um *backlog* de produto (priorizado e organizado em histórias de usuário) e incluiu o desenvolvimento incremental em tempos curtos de entrega com maior *feedback* e envolvimento dos usuários.

A abordagem Cascata é apropriada para alguns projetos da área de Engenharia Civil, que são monolíticos por natureza, tais como a construção de prédios [Wer12], mas quando aplicados em projetos de desenvolvimento de software tenderão para o que Kent Beck nomeou de *Big Design Up Front* (BDUF) [BeA04], que busca a precisão e a lógica perfeita na definição do escopo do projeto antes de iniciar o desenvolvimento do software, sendo recomendado apenas para situações onde os requisitos são razoavelmente estáveis [Pre11]; assim, o impasse entre projetar uma solução ideal antecipadamente e iniciar o desenvolvimento de software pode impedir que resultados de projetos sejam alcançados em tempo oportuno, como também, discussões para definição de tecnologias específicas podem desvirtuar o foco na resolução de problemas reais dos usuários [Wer12]. Por outro lado, a abordagem extremamente oposta, conhecida como *No Design Up Front* (NDUF), pode negligenciar e não considerar o apoio fundamental que o planejamento da arquitetura do sistema tem sobre a evolução do software [Boe02]. Nesse sentido, uma abordagem mais equilibrada, conhecida como *Enough Design Up Front* (EDUF) - sugerida por alguns métodos ágeis - tem emergido com resultados positivos (ROE01) (ROE05); essa abordagem considera que algum planejamento inicial é necessário; porém, ela enfatiza que o planejamento deve ser apenas o suficiente para que o desenvolvimento de software possa começar e avançar com segurança [DCO14].

O Modelo Cascata combinado com a abordagem BDUF são propensos para serem utilizados em contratações de grandes projetos com entregas *big-bang* no setor público, onde os testes e a implantação do sistema acontecem em uma única grande etapa após a entrega do software. O resultado pode ser projetos implementados em anos em vez de meses, com pouca ou nenhuma contribuição aos objetivos de negócios, conforme aconteceu nas fases de insucesso do projeto *Sentinel* no FBI (ROE03). Wernham [Wer12] vai além, ao afirmar que:

> *“A implementação e entrega big-bang ao final de grandes projetos faz com que os usuários finais tenham que mudar suas práticas de trabalho de uma só vez e começar a usar imediatamente o novo produto de software, que muitas vezes possui vários defeitos e não apresenta funções vitais implementadas. Implementação e entrega big-bang raramente tem entregado produtos de software ao menor custo, porque ela tende a adiar a execução do produto de software até que todas as funcionalidades estejam prontas ao mesmo tempo. Essa abordagem milita contra a obtenção de quaisquer ganhos reais imediatos.”*

Por fim, essas três abordagens, juntas ou separadamente, estão arraigadas e são inibidoras do desenvolvimento ágil em OP, porém relatórios recentes de governo desaconselharam sua utilização, sendo favoráveis ao desenvolvimento ágil [NAO12] [GAO12] [HMT14].

5.4 Algumas Lições Aprendidas sobre a Adoção de Métodos Ágeis em OP

Uma das melhores maneiras de se aumentar a memória organizacional é com a reunião de lições aprendidas. O armazenamento e a disseminação de lições aprendidas fazem com que os trabalhadores da organização reflitam sobre as experiências passadas e utilizem a memória organizacional como ponto de partida para as decisões presentes e futuras. Na sequência algumas lições aprendidas são apresentadas nas seguintes dimensões.

5.4.1 Algumas Lições sobre a Equipe

- Métodos ágeis requerem conhecimento e experiência da equipe (RSL01) (ROE04) (ECa01) (ECa02) (ECa03) (ECa04);
- Treinamentos e *coaches* contribuem para a formação de equipes com menos experiência em métodos ágeis (RSL01) (RSL02) (RSL03) (RSL08) (RSL09) (RSL17) (ECa03) (ECa04);
- Métodos ágeis encorajam a formação de equipes pequenas e interdisciplinares, o que diminui a complexidade na comunicação e o tempo para a tomada de decisão (RSL04) (ROE03) (ECa03) (ECa04);
- Métodos ágeis criam capacidade de resiliência nas pessoas (ROE04);
- Métodos ágeis podem ser adotado por equipes de desenvolvimento distribuídas geograficamente (RSL05);
- Métodos ágeis podem ser adotado em projetos com um ou mais clientes distribuídos geograficamente (ECa01) (ECa02);

- Equipes que se conhecem a mais tempo possuem um estado elevado de colaboração entre as pessoas, o que pode ser um diferencial nas fases iniciais do projeto, onde mais erros aparecem (ECa03) (ECa04).

5.4.2 Algumas Lições sobre o Relacionamento com o Cliente

- Métodos ágeis requerem um compromisso profundo e preparação, tanto do cliente como dos desenvolvedores para alcançar o seu pleno potencial (RSL05) (RSL11) (ECa03) (ECa04);
- Métodos ágeis tornam o processo de desenvolvimento de software mais aberto para o cliente através de uma comunicação direta entre o cliente e os desenvolvedores (RSL01) (RSL04) (RSL07) (RSL09) (RSL10) (RSL11) (RSL17) (ECa01) (ECa02) (ECa03) (ECa04);
- Métodos ágeis requerem uma comunicação oportuna, precisa e completa entre todos os membros da equipe de desenvolvimento, cliente e usuários do software (RSL01) (ECa01) (ECa03) (ECa04);
- As melhores soluções são criadas quando o cliente participa ativamente no desenvolvimento de software (RSL07) (ECa02) (ECa03);
- O *Product Owner* não é o cliente, ele comunica com o cliente e *stakeholders* e trabalha com a equipe em um regime de colaboração mútua (ECa03) (ECa04).

5.4.3 Algumas Lições sobre o Relacionamento com o Negócio

- Entregas incrementais e regulares de software, com o maior valor de negócio primeiro, são importantes para demonstrar a solução emergente para as partes interessadas, o que aumenta a confiança entre os membros do projeto (RSL02) (ROE01) (ROE04) (ECa02) (ECa03) (ECa04);
- Para alcançar um maior retorno sobre o investimento, os itens com maior valor agregado são priorizados para o desenvolvimento (ROE03);
- Para comunicar o andamento do projeto, reuniões de avaliação e demonstração do produto de software são realizadas e adaptadas para públicos específicos (RSL01) (RSL02) (RSL04) (ECa01);
- Métodos ágeis requerem que a equipe de desenvolvimento aprenda mais sobre os processos de negócio para produzir soluções ideais (ECa01) (ECa02) (ECa03) (ECa04);

- Medição de valor agregado e progresso são necessários para manter conformidade com alguns regulamentos contratuais de governo (RSL04) (RSL06) (RSL09) (RSL12);
- Métricas ágeis são fundamentais para tornar o processo decisório mais preciso e objetivo, bem como, são importantes para promover a visibilidade do projeto mais cedo (RSL02) (RSL12) (RSL17) (ECa03) (ECa04).

5.4.4 Lições sobre Processos e Práticas

- Práticas ágeis são simples de entender, porém, internalizá-las e segui-las é rigorosamente difícil (RSL02) (RSL03) (RSL08) (RSL17) (ROE04);
- Métodos ágeis alcançam melhores resultados quando desenvolvedores de software do governo atuam em estreita colaboração com desenvolvedores de software da indústria privada em todas as fases do desenvolvimento de software, na mesma localização física e próximos dos clientes (RSL04) (RSL09) (ROE03);
- Métodos ágeis requerem mais testes escritos por desenvolvedores, o que permite sua execução de maneira automatizada, que por sua vez aumenta a qualidade do produto de software (RSL03) (RSL04) (RSL09) (RSL10) (RSL11) (RSL13) (RSL16) (RSL17) (ROE05) (ECa02);
- Métodos ágeis podem conviver com abordagens prescritivas (RSL09) (ROE04) (ROE05);
- Métodos ágeis oferecem pouco ou nenhum suporte para a construção de sistemas que envolvem migração de dados e integração entre sistemas, que são desafios difíceis de resolver tecnicamente, principalmente com entregas parciais de software (ROE02) (ROE03) (ECa03) (ECa04);
- Aspectos de usabilidade do produto de software precisam ser considerados desde as primeiras iterações no desenvolvimento do software (RSL01), bem como, testes exploratórios com usuários finais e cenários reais ainda são necessários (ROE01);
- Em ambientes contratuais, métodos ágeis requerem contratos de escopo negociável, visando alterar o escopo do projeto em direção a solução ideal (RSL04) (ROE01) (ROE03).

5.4.5 Lições sobre Ferramentas

- Ferramentas eletrônicas de apoio ao desenvolvimento de software facilita a adoção de métodos ágeis. Elas são úteis para registrar os defeitos encontrados

e as solicitações de melhorias, bem como, para fornecer informações mais realistas sobre “o que está acontecendo” e “quem está fazendo o que” no projeto (RSL03) (RSL09) (RSL11) (RSL13) (ECa01) (ECa02) (ECa03) (ECa04). Elas também são utilizadas quando a colocalização entre os membros da equipe de desenvolvimento e o cliente não é possível (ROE05) (ECa01);

- Ferramentas eletrônicas de código aberto são uma alternativa economicamente viável para apoiar o trabalho de desenvolvimento de software (ROE05) (ECa02) (ECa03) (ECa04);
- Ferramentas visuais, como parede de cartões, também são usadas para o gerenciamento do projeto e valorização do trabalho em equipe (ROE04) (ECa03) (ECa04).

5.5 Alguns Métodos, Práticas e Métricas Adotadas em OP

Métodos ágeis ao valorizar a colaboração com o cliente faz com que a prática Cliente Presente (XP) seja uma das práticas mais utilizadas. Como a maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado, então práticas de planejamento como Jogo de Planejamento (XP), Reunião de Planejamento da Sprint e Backlog de Produto Priorizado (Scrum) são essenciais. O princípio da entrega frequente de software funcionando com preferência aos tempos menores faz com que as práticas Releases Curtas (XP), Desenvolvimento em Sprints (Scrum) e Iteração (DSDM) sejam amplamente adotadas em projetos ágeis. Como indivíduos e interações são mais importantes do que processos e ferramentas, então as Reuniões Diárias (Scrum) e as Reuniões de Revisão (Scrum) e Retrospectivas (Scrum) são espaços importantes para as pessoas saberem o que está acontecendo no projeto e se ele está no caminho certo, bem como, são essenciais para as pessoas refletirem sobre pontos de melhorias na sua forma de realizar o trabalho. Contínua atenção a excelência técnica e bom *design* faz com que algumas práticas de desenvolvimento, *design* e teste ágil de software - tais como: TDD, BDD e automação de testes - sejam aceitas e adotadas em projetos ágeis de software para atender aos requisitos de qualidade do produto. O suporte as necessidades da gerência de informação são atendidas por meio da mensuração e análise de métricas de software, incluindo gráficos de *burns* e cálculo de valor agregado, por exemplo. Vale salientar que a técnica para medição Análise de Pontos de Função - fortemente recomendada em processos de subcontratação de software - não foi mencionada nos estudos teóricos e empíricos. A Tabela 19 apresenta alguns métodos, práticas e métricas identificados neste estudo.

Tabela 19 - Alguns métodos, práticas e métricas adotadas em OP.

MÉTODOS, PRÁTICAS E MÉTRICAS		ID
EXTREME PROGRAMMING	Jogo de planejamento	(RSL01) (RSL11) (RSL16) (ECa02)
	Releases curtas	(RSL01) (RSL03) (RSL04) (RSL07) (RSL11) (RSL13) (RSL17) (RSL16) (ECa02)
	Metáfora	(RSL01) (RSL03) (RSL11) (RSL16)
	Design simples	(RSL01) (RSL03) (RSL11) (RSL16)
	Desenvolvimento Dirigido por Testes (TDD)	(RSL01) (RSL03) (RSL09) (RSL16)
	Refatoração	(RSL01) (RSL11) (RSL16) (ECa02)
	Programação em Par	(RSL01) (RSL16)
	Propriedade coletiva do código	(RSL01) (RSL03) (RSL11) (RSL16) (ECa02)
	Integração contínua	(RSL01) (RSL03) (RSL09) (RSL11) (RSL16) (RSL17) (ROE05) (ECa02)
	Semana de 40 horas	(RSL11) (RSL16)
	Cliente presente	(RSL01) (RSL07) (RSL09) (RSL10) (RSL11) (RSL16) (RSL17) (ECa02)
	Ritmo sustentável	(RSL01) (RSL03) (RSL04) (RSL09) (RSL16) (RSL17)
	Padrões de codificação	(RSL01) (RSL03) (RSL11) (RSL16)
SCRUM	Backlog priorizado do produto	(ROE03) (ROE04) (ECa01) (ECa03) (ECa04)
	Discovery / Delivery	(ECa03) (ECa04)
	Story mapping	(ECa03) (ECa04)
	Reunião de planejamento da sprint	(RSL04) (RSL10) (ROE03) (ROE04) (ECa01)
	Desenvolvimento iterativo em sprints	(RSL03) (RSL04) (RSL07) (RSL09) (RSL10) (RSL17) (ROE03) (ROE04) (ROE05) (ECa01) (ECa03) (ECa04)
	Reunião diária	(RSL01) (RSL04) (ROE04) (ROE05) (ECa03) (ECa04)
	Definition of Done (DoD)	(ROE04)
	Definition of Ready (DoR)	(ECa03) (ECa04)
	Revisão da sprint	(RSL04) (RSL10) (ROE03) (ROE04) (ECa01) (ECa03) (ECa04)
	Retrospectiva da sprint	(RSL04) (ROE04) (ECa03) (ECa04)
	Scrum de Scrum	(ROE04)
DSDM	Análise de negócio	(ROE01)
	Iteração (Exploração e Engenharia)	(ROE01) (ROE02)
	Implantação	(ROE01)
	Gerenciamento de risco	(ROE05)
DESIGN E TESTES	Testes de unidade	(RSL03) (RSL09) (RSL10) (RSL11) (RSL13) (ECa02) (ROE05)
	Testes de aceitação	(RSL03) (RSL09) (RSL10) (RSL11) (RSL17) (ROE04) (ROE05) (ECa02) (ROE05) (ECa03) (ECa04)
	Testes automatizados	(RSL03) (RSL04) (RSL09) (RSL10) (RSL11) (RSL13) (RSL17) (ROE05) (ECa02)
	Testes exploratórios	(ROE05)
	BDD	(ECa03) (ECa04)
	TDD	(RSL01) (RSL03) (RSL09) (RSL16)
MÉTRICAS	Cobertura de testes	(RSL03) (RSL10)
	Product size	(RSL17)
	Pulse	(RSL17)
	Burns	(RSL04) (RSL17) (ECa03) (ECa04)
	Story points	(RSL04) (ROE03) (ECa03) (ECa04)
	Faults	(RSL17)
	KLOC	(RSL13)
	Efferent Coupling	(RSL10)
	Cyclomatic Complexity	(RSL10) (RSL13)
	EVM	(RSL04) (RSL06)

5.6 Considerações Finais sobre o Arcabouço de Resultados

O arcabouço de resultados sugere que o desenvolvimento ágil de software em OP pode ser mais desafiador do que em outros ambientes porque pessoas com pouca experiência precisam conduzir equipes e projetos ao sucesso demonstrando resultados positivos de curto prazo, sendo que em algumas situações elas não possuem o apoio organizacional e o ambiente necessário para executarem o seu trabalho, dependendo de especialistas externos. Embora essa situação possa mudar ao longo do tempo, este estudo encontrou pistas de que em determinadas OP esse processo pode ser muito lento e complexo porque isso exige que as pessoas aprendam novos comportamentos e adotem novas maneiras de interação com as outras pessoas, sendo o problema agravado quando as novas práticas de trabalho são conflitantes com dinâmicas de abordagens arraigadas no setor público, dificultando a mudança. Por outro lado, encontraram-se indícios de que essas barreiras não são intransponíveis, sendo que as pessoas de TI e as pessoas com poder de decisão nas OP têm um papel importante nesse processo de estimulo à descoberta do que pode ser produzido de forma mais eficiente e com menos burocracia no setor público. Ou seja, cabe aos próprios servidores públicos à experimentação de métodos ágeis, descobrindo suas vantagens, seus benefícios, o que pode dar errado e o que funciona e em que condições para o seu próprio contexto.

Por último, alcançar os benefícios prometidos pelos métodos ágeis leva tempo e exige esforço e dedicação de seus proponentes. Com relação aos benefícios alcançados, o arcabouço de resultados sugere que nas OP eles são mais sobre a entrega de valor antecipada e regular aos clientes (aumentando a satisfação do cliente com o produto entregue), bem como, sobre a satisfação das pessoas com o trabalho e a forma como ele foi realizado (aumentando a satisfação no trabalho). Isto reduz o risco do produto de software não ser aceito na organização porque os clientes e os usuários estão comprometidos e participam ativamente do desenvolvimento do produto de software visualizando o crescimento e a evolução do sistema ao longo do tempo, o que diminui consideravelmente o descontentamento da área de negócios com o tempo para atendimento das demandas de sistemas porque as entregas em produção são realizadas em tempos menores e são mais aderentes as prioridades de negócio, agregando mais valor. O resultado motivacional é a satisfação das pessoas com o trabalho realizado e o estabelecimento de uma relação de confiança e parceria entre a área de TI e a área de negócio da organização.

6 RECOMENDAÇÕES PARA A ADOÇÃO DE MÉTODOS ÁGEIS EM OP

Competência para mudar a cultura organizacional é a característica mais desejada para iniciar a adoção de métodos ágeis [Ver15] [MSC+12]. No contexto das OP, a luta de forças se manifesta entre o "moderno e o antigo", ou seja, as inovações das organizações no mundo contemporâneo ante uma dinâmica de normas e processos arraigados [PiM06]. No geral, as OP se deparam com a necessidade da modernidade. Mais que isso, elas precisam integrar de maneira inovadora as necessidades da sociedade à sua capacidade administrativa e técnica de realização dos serviços públicos, sendo essa junção inerente e fundamental para as ações nesse campo. Essa disputa de forças leva as OP a refletirem sobre seus métodos de trabalho, onde se possa encontrar as melhores abordagens para alcançar seus objetivos, que consistem em serviços eficientes à sociedade. Assim, este capítulo apresenta - com base nos estudos teóricos e empíricos - um conjunto de recomendações para introduzir de maneira responsiva nas OP a visão de modernidade e agilidade proposta pelos métodos ágeis.

6.1 Definições de Adoção, Transformação e Transição

Nesta seção apresenta-se as definições de adoção, transformação e transição no escopo deste estudo, estabelecido por Michael Sahota [Sah12]:

- **Adoção:** é um termo aplicado tanto a um produto quanto a um processo. Assim, pode-se dizer seguramente que se está adotando métodos ágeis, como o Scrum.
- **Transformação:** significa uma mudança de uma forma de ser para outra, ou seja, é uma mudança de grande porte, como a transformação para ágil, que representa uma alteração importante nos comportamentos e valores.
- **Transição:** significa “movimento, passagem; mudança de uma posição, estado, estágio, conceito, etc., para outro(a)”. O termo transição é ambíguo podendo ser utilizado tanto para descrever uma adoção quanto uma transformação.

6.2 Conjunto Preliminar de Recomendações para a Adoção de Métodos Ágeis em OP

Uma proposta preliminar de recomendações para adoção de métodos ágeis em OP foi elaborada a partir do arcabouço de resultados. Como dito anteriormente, em algumas situações foi necessário consultar novamente os estudos teóricos e o texto transcrito dos estudos empíricos para esclarecer alguns pontos. As recomendações foram divididas em fases, incluindo: Preparação, Execução e Aprendizado. A Tabela 20 apresenta o conjunto preliminar com cada fase e suas respectivas recomendações.

Tabela 20 - Conjunto preliminar de recomendações para a adoção de métodos ágeis em OP.

FASE	RECOMENDAÇÃO	FONTE
1. PREPARAÇÃO	Estabeleça uma estratégia de gestão de mudanças com base nas pessoas	(ECa03) (ECa04)
	Comece com pessoas dispostas a mudanças	(RSL03) (RSL11) (ECa02) (ECa03) (ECa04)
	Envolva e comprometa os donos do poder	(RSL03) (RSL07) (RSL08) (RSL16) (RSL17) (ECa03) (ECa04)
	Comece com projetos piloto-importantes	(RSL01) (RSL03) (RSL08) (RSL11) (RSL16) (RSL17) (ECa01) (ECa02)
	Forneça as condições necessárias para as pessoas executarem o seu trabalho	(RSL03) (RSL07) (RSL14) (RSL16) (ECa01) (ECa02) (ECa03) (ECa04)
2. EXECUÇÃO	Promova o trabalho em equipe preferencialmente com equipes pequenas e perenes	(RSL04) (ROE03) (ECa03) (ECa04)
	Envolva, comprometa e satisfaça os clientes e os usuários	(RSL01) (RSL02) (RSL03) (RSL04) (RSL10) (ROE01) (ROE02) (ROE03) (ROE04) (ROE05) (ECa01) (ECa02) (ECa03) (ECa04)
	Considere a incorporação do analista de qualidade na equipe	(ECa03) (ECa04)
	Estabeleça o tempo das entregas de software preferencialmente em tempos mais curtos	(ECa01) (ECa02) (ECa03) (ECa04)
	Capacite a equipe no método e nas tecnologias utilizadas no projeto real	(RSL01) (RSL02) (RSL03) (RSL08) (RSL09) (RSL17) (ECa03) (ECa04)
	Forneça ferramentas de apoio ao processo de desenvolvimento de software	(RSL03) (RSL09) (RSL11) (RSL13) (RSL13) (ROE04) (ROE05) (ECa01) (ECa02) (ECa03) (ECa04)
	Acompanhe o progresso do projeto diariamente e visivelmente	(RSL04) (RSL17) (ECa03) (ECa04)
	Assegure o uso de controlador de versão com a prática de integração contínua	(RSL01) (RSL03) (RSL09) (RSL11) (RSL16) (RSL17) (ROE05) (ECa02)
	Explore múltiplas formas de comunicação	(ECa01)
	Evite a utilização da abordagem multitarefa	(RSL08) (ECa01) (ECa02)
	Satisfaça as exigências legais da organização	(RSL04) (RSL06) (RSL09) (RSL12) (ROE03) (ROE04)
	Estabeleça expectativas de documentação	(ECa01) (ECa03)
3. APRENDIZADO	Experimente métodos e práticas emergentes	(ECa03) (ECa04)
	Incentive a criação de grupos ou comunidades de prática	(ECa03) (ECa04)

6.3 Estratégias para Consolidação das Recomendações

Após a elaboração do conjunto preliminar de recomendações para a adoção de métodos ágeis em OP, planejou-se e executou-se uma avaliação das recomendações com os dois sujeitos que compuseram a dimensão organizacional dos estudos empíricos realizados (Seiya e Eurydice). Esses sujeitos apresentam-se como protagonistas em suas organizações na condução de estratégias para implantação de métodos ágeis em nível organizacional. A abordagem utilizada abrangeu a realização de entrevistas individuais; na Embrapa, a entrevista foi realizada de maneira presencial, enquanto na PROCERGS a entrevista aconteceu por telefone. Uma cópia (com apenas uma página) da versão preliminar do conjunto de recomendações foi encaminhada para os entrevistados por e-mail, com pelos menos três dias de antecedência em relação a data da entrevista, visando introduzir o assunto. O tempo das entrevistas não ultrapassou 30 minutos. Em ambas as entrevistas, as considerações dos sujeitos foram anotadas e suas observações foram avaliadas e incorporadas ao conjunto final de recomendações para a adoção de métodos ágeis em OP. Para isso, um roteiro contendo três perguntas foi estabelecido visando buscar algum grau de estabilidade no conjunto de recomendações:

1. O protocolo de recomendações representa as ações executadas na sua organização para a adoção de métodos ágeis? Se não, o que está faltando, ou então, o que está em excesso?
2. Na sua opinião, quais são as recomendações mais relevantes na adoção de métodos ágeis?
3. O que pode ser mais desafiador para as organizações públicas comparativamente a outras organizações?

Como resultado, os sujeitos entrevistados corroboraram com a versão preliminar do conjunto de recomendações e suas fases (com poucas sugestões de melhorias), o qual representa o conjunto de ações executadas no processo de adoção de métodos ágeis em suas organizações, ou então, a serem executadas em um futuro próximo. Como sugestão de melhoria, ambos os sujeitos solicitaram acrescentar na fase de Aprendizado uma recomendação de avaliação da adoção de métodos ágeis de modo a informar a alta gerência da organização os resultados alcançados a partir do seu uso. Com relação as recomendações mais relevantes, ambos destacaram os aspectos que compõem a fase de Preparação como as mais importantes e fundamentais para o sucesso na adoção de métodos ágeis, principalmente o envolvimento e o comprometimento dos donos do poder e dos primeiros adeptos.

Por último, sobre o que pode ser mais difícil para as OP comparativamente a outras organizações na adoção de métodos ágeis, os sujeitos apresentaram pontos de vista diferentes. Seiya argumenta que engajar os donos do poder e os primeiros adeptos em OP é mais difícil:

> *"Como as instituições públicas não visam lucro financeiro e os sistemas criados não possuem potencial comercial, então é mais difícil criar um senso de urgência a favor de métodos ágeis (...) consequentemente conquistar e comprometer os donos do poder, bem como alcançar os primeiros adeptos é mais difícil, o que exige um exercício maior de convencimento." (Seiya, 15/12/2014)*

Enquanto Eurydice menciona que a criação de ambientes ágeis em OP é mais difícil:

> *"É sabido que comunicação sempre aberta e direta, bem como, a convivência diária entre as pessoas aumenta a confiança no trabalho em equipe. Porém, no âmbito do governo, muitas pessoas topam experimentar métodos ágeis, mas não querem modificar alguns comportamentos, mantendo o receio de exposição preferindo espaços individuais em vez de espaços coletivos e abertos, por exemplo." (Eurydice, 16/12/2014)*

6.4 Consolidação das Recomendações para a Adoção de Métodos Ágeis em OP

Métodos ágeis para o desenvolvimento de software, como qualquer outro método, exigem conhecimento técnico para sua execução, mas seria um erro concentrar-se apenas em aspectos técnicos do desenvolvimento de software. Como os métodos ágeis representam uma mudança quando comparados as abordagens prescritivas, então fatores organizacionais e culturais precisam ser considerados e não podem ser "subestimados". Como constatado nos estudos teóricos e empíricos, alguns projetos de software executados em OP têm utilizado efetivamente práticas de desenvolvimento e gerenciamento ágil. Elas são conhecidas e podem ser experimentadas e repetidas continuamente. Entretanto, o que se apresenta como desafio são as questões de gestão de mudança que surgem a partir de uma nova forma de realizar o trabalho e encontrar um caminho de pequenos sucessos iniciais para a sua expansão como um método de desenvolvimento comumente aceito e utilizado na organização. Nesta direção, o conjunto de recomendações sugere que a mudança aconteça devagar sem "queimar etapas" porque a direção certa é mais importante do que a velocidade. Além disso, ele fornece informações úteis, de modo que as OP consigam tornar as promessas dos benefícios dos métodos ágeis uma realidade para o seu contexto. A Tabela 21 apresenta o conjunto final de recomendações.

Tabela 21 - Conjunto final de recomendações para a adoção de métodos ágeis em OP.

FASE	RECOMENDAÇÃO	FONTE
1. PREPARAÇÃO	Estabeleça uma estratégia de gestão de mudanças com base nas pessoas	(ECa03) (ECa04) [App12] [Fre98]
	Comece com pessoas dispostas a mudanças	(RSL03) (RSL11) (ECa02) (ECa03) (ECa04) [Fre98]
	Envolva e comprometa os donos do poder	(RSL03) (RSL07) (RSL08) (RSL16) (RSL17) (ECa03) (ECa04) [Fre98]
	Comece com projetos-piloto importantes	(RSL01) (RSL03) (RSL08) (RSL11) (RSL16) (RSL17) (ECa01) (ECa02) [Coh11]
	Forneça as condições necessárias para as pessoas executarem o seu trabalho	(RSL03) (RSL07) (RSL14) (RSL16) (ECa01) (ECa02) (ECa03) (ECa04) [App12]
2. EXECUÇÃO	Promova o trabalho em equipe preferencialmente com equipes pequenas e perenes	(RSL04) (ROE03) (ECa03) (ECa04)
	Envolva, comprometa e satisfaça os clientes e os usuários	(RSL01) (RSL02) (RSL03) (RSL04) (RSL10) (ROE01) (ROE02) (ROE03) (ROE04) (ROE05) (ECa01) (ECa02) (ECa03) (ECa04)
	Assegure a qualidade do código e do produto	(RSL01) (RSL03) (RSL09) (RSL10) (RSL11) (RSL13) (RSL16) (RSL17) (ROE01) (ROE05) (ECa02) (ECa03) (ECa04)
	Estabeleça o tempo das entregas de software preferencialmente em tempos mais curtos	(ECa01) (ECa02) (ECa03) (ECa04)
	Capacite a equipe no método e nas tecnologias utilizadas no projeto real	(RSL01) (RSL02) (RSL03) (RSL08) (RSL09) (RSL17) (ECa03) (ECa04)
	Forneça ferramentas de apoio ao processo de desenvolvimento de software	(RSL03) (RSL09) (RSL11) (RSL13) (RSL13) (ROE04) (ROE05) (ECa01) (ECa02) (ECa03) (ECa04)
	Acompanhe o progresso do projeto diariamente e visivelmente	(RSL04) (RSL17) (ECa03) (ECa04) [Hum10]
	Explore múltiplas formas de comunicação	(ECa01) (ECa02) (ECa03) (ECa04)
	Evite alocar a mesma pessoa em vários projetos concomitantes	(RSL08) (ECa01) (ECa02) [Coh11] [Pop11]
	Satisfaça as exigências legais da organização	(RSL04) (RSL06) (RSL09) (RSL12) (ROE03) (ROE04)
	Estabeleça expectativas de documentação	(ECa01) (ECa03)
	Experimente métodos e práticas emergentes	**(Entrevistas de consolidação das recomendações)** (ECa03) (ECa04) [Coh11]
3. APRENDIZADO	Avalie a adoção de métodos ágeis e avalie o produto gerado fornecendo *feedback* para os donos do poder	(ECa03) (ECa04)
	Incentive a criação de grupos ou comunidades de prática	**(Entrevistas de consolidação das recomendações)** (ROE03) (ROE04) (RSL16)

6.4.1 Preparação

O estudo teórico e a experiência prática das quatro equipes estudadas apontaram cinco recomendações que precisam ser devidamente consideradas antes de começar a adoção ou transformação ágil (Figura 5). Em primeiro lugar, as pessoas precisam ser consideradas nas estratégias de mudanças. Em segundo lugar, precisa haver pessoas receptivas a mudanças. Em terceiro lugar, o compromisso da gestão para este esforço é extremamente importante. Em quarto lugar, a criticidade e a importância dos projetos-piloto precisam ser cuidadosamente consideradas. Por último, precisa haver um ambiente e uma infraestrutura de TI adequada para apoiar o desenvolvimento ágil, visando facilitar as entregas frequentes de software com qualidade.

Figura 5 - Fase de Preparação.

6.4.1.1 Estabeleça uma Estratégia de Gestão de Mudanças com Base nas Pessoas

Transformar uma organização significa implementar mudanças profundas, muitas vezes radicais, sobre o *status quo* existente. No cerne das mudanças organizacionais está a mudança dos recursos humanos da organização [Fre98]. Qualquer processo de transformação ágil ou adoção de métodos ágeis, que não leve em consideração as pessoas envolvidas e as interações entre elas em primeiro lugar, tem grandes chances de falhar. Portanto, informar, ouvir, envolver, convencer, respeitar e comprometer as pessoas da organização, de forma a minimizar as barreiras humanas e organizacionais contra as mudanças que visam levar a OP ao estado futuro desejado é uma recomendação importante dentro das estratégias de transformação ágil ou adoção de métodos ágeis. Não é o intuito deste estudo cobrir abordagens conhecidas sobre mudanças organizacionais utilizadas pela comunidade ágil. Porém, este tópico inspirou-se na obra de Jurgen Appelo intitulada “Como Mudar o Mundo: Gestão de Mudanças 3.0” [App12].

6.4.1.2 Comece com Pessoas Dispostas a Mudanças

Todo programa de mudança começa também com uma primeira seleção de pessoas. A recomendação é começar com pessoas que estão mais propensas e dispostas a trabalhar com métodos ágeis, a experimentar o novo, a superar os obstáculos organizacionais da cultura dominante e não investir muito tempo combatendo céticos e críticos. Isso inclui também os clientes e seus representantes.

6.4.1.3 Envolva e Comprometa os Donos do Poder

Mudanças de porte em OP não se iniciam e se processam com sucesso caso não haja envolvimento e o real comprometimento dos executivos da organização, isto é, dos detentores do poder posicionado no nível mais alto da organização [Fre98]. Ou seja, a adoção de métodos ágeis em um processo de transformação ágil em OP, requer o aval, apoio e real comprometimento da alta direção da organização. Porém, os donos do poder precisam saber exatamente o que eles precisam fazer, por exemplo, eles podem demonstrar o seu apoio por meio da contratação de capacitações técnicas, implementação de melhorias na comunicação, aquisição de ferramentas eletrônicas e visuais, mudanças organizacionais e no ambiente que apoiem as equipes de desenvolvimento ágil, etc.

6.4.1.4 Comece com Projetos-Piloto Importantes

Apesar de “Ágil” representar uma mudança profunda na forma de pensar, este estudo apoia a adoção incremental de métodos ágeis orientada a projetos-piloto, sendo mais conveniente em OP, onde os valores e princípios do “Ágil” podem ser incompatíveis com a cultura organizacional. A estratégia é melhorar continuamente a cultura dominante através de bons exemplos em vez de muda-la radicalmente. Nesse sentido, a experiência dos primeiros projetos são bons exemplos de como as coisas estão indo (com acertos e erros) e como elas poderiam ser ainda melhores (melhoria contínua), fornecendo orientações para os projetos subsequentes, então, repetir a experiência para outros projetos é uma decisão mais sustentável. A recomendação é começar com projetos-piloto importantes e evoluir a partir deles e então disseminar os métodos ágeis em toda a empresa. Um projeto insignificante não chamará a atenção necessária do resto da empresa [Coh11].

Essa estratégia mais conservadora e menos arriscada não é para todos. O FBI, por exemplo, seguiu a estratégia oposta (a estratégia da transformação). Ele apostou tudo no Scrum e no desenvolvimento colocalizado para recuperar anos de fracassos consecutivos. O resultado mostrou-se bem-sucedido (RSL04) (ROE03). Por outro lado, as transformações

demandam esforços monumentais e riscos significativos, o que exige mais preparo e recursos das OP para executar uma transformação bem-sucedida, sendo recomendada em situações onde resultados negativos permanecem constantes no tempo [Sah12]. Quando a OP não entendeu o que significava e o que representava o termo transformação, o resultado converteu-se em fracasso (RSL08).

Por fim, é importante frisar que os estudos teóricos e empíricos apontam que a maior parte das OP não vislumbraram a transformação imediata para ágil. Pelo contrário, sua adesão aconteceu de maneira incremental por meio de experiências bem-sucedidas em projetos-piloto que foram, estão sendo ou serão repetidas em outros projetos com novas pessoas interessadas.

6.4.1.5 Forneça as Condições Necessárias para as Pessoas Executarem o seu Trabalho

Mesmo quando as pessoas querem mudar, a situação em que se encontram pode, muitas vezes, impedi-las de progredir [App12]. Apoio significativo da alta gerência remove barreiras para que as pessoas possam praticar métodos ágeis e não fracassem, o que exige mais tempo, recursos e dedicação da própria gestão.

Uma vez que a falta de conhecimento e experiência das pessoas em métodos ágeis é uma questão latente em OP. Essas pessoas precisarão de ajuda para executar métodos ágeis, sendo importante considerar como elas serão apoiadas e quem as apoiará. Cursos, treinamentos, aquisição de livros, serviços de *mentoring* e *coaches* podem direcionar as pessoas em sua jornada rumo à adoção de métodos ágeis.

As ferramentas e a infraestrutura de que as pessoas dispõem vão influenciar e guiar, significativamente, o comportamento delas. Vale-se perguntar se é possível mudar, de alguma forma, a infraestrutura física ou digital na qual as pessoas estão trabalhando e vivendo, a favor do desenvolvimento ágil de software [App12]. A equipe precisa de vários recursos tecnológicos, incluindo computadores para desenvolvimento, software de controle de versão, ferramentas e ambientes para testes, Internet rápida, etc. Isso é importante para que ela não seja prejudicada por uma infraestrutura de TI menos adequada. Ou seja, independente de saber se o novo método de desenvolvimento irá atender aos anseios da organização, o resultado final não irá atender grande parte das expectativas, se a infraestrutura de TI é superficial (RSL03) (ECa02). Portanto, além de haver pessoas dispostas a mudanças fortemente apoiadas pela alta administração da organização, uma infraestrutura de TI pronta para apoiar a execução de métodos ágeis pode ser amplamente eficiente.

6.4.2 Execução

Uma vez que existe uma subcultura disposta e apoiada pela alta administração para trabalhar com métodos ágeis, então a próxima etapa consiste em adotá-los na prática em projetos-piloto. Na sequência, são apresentadas doze recomendações identificadas como essenciais para a fase de execução de métodos ágeis em projetos de software (Figura 6).

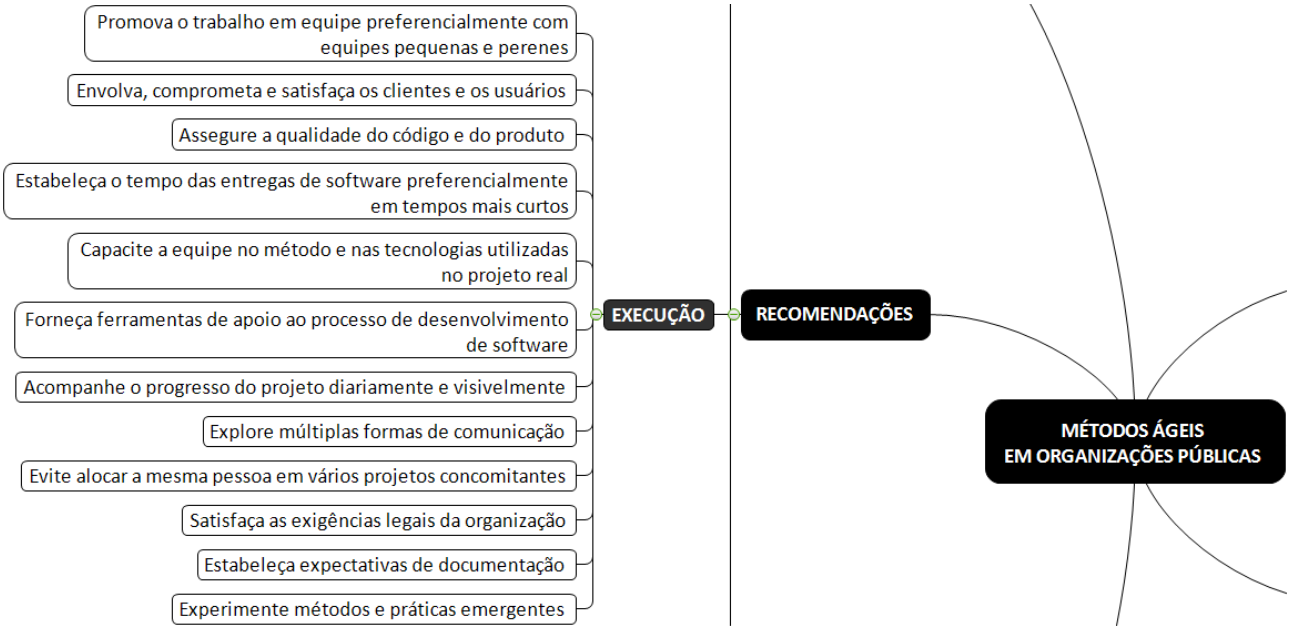

Figura 6 - Fase de Execução.

6.4.2.1 Promova o Trabalho em Equipe preferencialmente com Equipes Pequenas e Perenes

Duas equipes entrevistadas (ECa03) (ECa04) relataram que métodos ágeis promoveram o trabalho em equipe. Ao invés de as pessoas serem responsáveis por atividades individuais, a equipe como um todo responsabiliza-se pelos resultados. Reuniões diárias e retrospectivas da equipe foram encontradas como práticas essenciais para saber exatamente como o trabalho está indo, como problemas podem ser resolvidos e como o trabalho pode ser melhorado. Assim, o esforço da gestão é sobre o andamento do projeto e não sobre o desempenho individual das pessoas.

Um pré-requisito importante para o desenvolvimento ágil de software é construir a confiança entre os membros da equipe e saber como eles irão comunicar-se. Esse processo é facilitado quando as pessoas se conhecem e trabalham juntas a um certo tempo e quando as equipes são menores. Em vez de equipes serem formadas para um projeto específico, com equipes perenes o projeto é atendido pela equipe mais adequada, a qual deve estar em um estado elevado de colaboração entre as pessoas, o que pode ser um diferencial nas fases iniciais do projeto, onde mais erros aparecem (ECa03) (ECa04).

6.4.2.2 Envolva, Comprometa e Satisfaça os Clientes e os Usuários

A colaboração e a satisfação dos clientes são componentes importantes preconizados no manifesto ágil [BBB+01]. As pessoas patrocinadoras dos projetos de software precisam estar satisfeitas com o trabalho realizado, bem como, as pessoas que estão ou irão utilizar a solução técnica. Um projeto ágil bem-sucedido é aquele em que o cliente é o ponto focal da equipe - tanto em termos de produto que está sendo desenvolvido (a saída) como em termos de melhoria de negócio (o resultado). Projetos que utilizam abordagens prescritivas de software são frequentemente caracterizados com entregas de software de uma só vez, onde os clientes e usuários esperam por meses ou até anos para ver os resultados do seu trabalho. Por outro lado, métodos ágeis enfatizam a entrega das funções essenciais do sistema de maneira antecipada para o cliente. Para que isso seja possível os clientes e a equipe precisam colaborar e comprometer-se de maneira mais intensa com o projeto. Do lado do cliente, eles precisam conhecer o processo de trabalho da equipe; eles precisam rever e fornecer *feedback* sobre o sistema durante as iterações e as retrospectivas; eles precisam dar importância para a priorização dos requisitos; eles precisam comunicar claramente a visão do produto para a equipe; eles precisam respeitar os limites da equipe. Do lado da equipe, ela precisa entender o negócio e as demandas do cliente; ela precisa ajudar na definição de valor para o cliente; ela precisa deixar claro para o cliente o que será entregue no final de cada iteração, ela precisa ganhar a confiança do cliente demonstrando resultados com base na entrega de valor por meio de funcionalidades úteis e benefícios ao final de cada iteração.

Por último, todo esse esforço é necessário para não resultar em produto diferente do esperado. Neste sentido, para haver uma mudança real a favor dos métodos ágeis é desejável a existência do cliente e da equipe comprometida, que participa ativamente e reserva a quantidade certa de tempo para o desenvolvimento de software.

6.4.2.3 Assegure a Qualidade do Código e do Produto

O papel do Analista de Qualidade é essencial para o desenvolvimento ágil porque o teste é importante para a qualidade global do produto de software. Ele é responsável por escrever cenários de testes e definir critérios de aceitação no início de cada iteração (ECa03) (ECa04). Ele ajuda os desenvolvedores a escrever e automatizar bons testes de unidade e de integração. Ele cria *scripts* para automatizar o processo de execução dos testes. Ou seja, em vez atuar na descoberta de defeitos no final, o Analista de Qualidade atua na prevenção de defeitos apresentando características de qualidade de software desde o princípio. Práticas como BDD, TDD e automação de testes foram largamente

recomendadas nos estudos teóricos e empíricos; porém, isso implica mudanças nos hábitos de programação dos desenvolvedores, onde para alguns deles testes ainda são “caros”, demorados e “chatos” de escrever e manter (RSL03) (RSL17) (ECa03). Além disso, é de grande importância para o desenvolvimento ágil de software que os desenvolvedores sejam capazes de realizar compilações e integrações frequentes de código. Para isso, equipes relataram a importância do uso de uma ferramenta de controle de versão associada a prática de integração contínua, onde cada integração é verificada através de uma compilação e execução de testes automatizados para detectar erros de compilação, defeitos no sistema e problemas de integração o mais rápido possível.

Duas equipes entrevistadas (ECa01) (ECa03) mencionaram dificuldades em estabelecer arquiteturas de software com pouca ou nenhuma informação inicial, o que para elas é prejudicial para responder as necessidades dos clientes no longo prazo afetando a qualidade do produto no futuro. Dessa maneira, essa recomendação reforça a necessidade de que algum planejamento inicial é necessário; porém, ela enfatiza que o planejamento deve ser apenas o suficiente (abordagem EDUF) para que o desenvolvimento de software possa começar e avançar com segurança e qualidade.

6.4.2.4 Estabeleça o Tempo das Entregas de Software Preferencialmente em Tempos mais Curtos

A equipe e o cliente precisam chegar a um consenso sobre o cronograma dos ciclos de entregas de software. Três equipes (ECa02) (ECa03) (ECa04) preferiram iterações curtas de uma ou duas semanas. Enquanto, uma equipe (ECa01) não conseguiu definir exatamente o tempo de entrega. Além disso, equipes relataram dificuldades em estabelecer entregas em produção mais cedo (ECa03) (ECa04), e outras não conseguiram realizar entregas de software em produção até que o sistema estivesse completo (ECa01) (ECa02). Ou seja, o desenvolvimento ocorre em uma cadeia fixa de tempo, porém o negócio decide quando o sistema ou partes dele será liberado para uso em produção realmente. O ideal é evitar tempos prolongados para o lançamento de software em produção.

Duas equipes entrevistadas (ECa03) (ECa04) apresentaram um modelo de entrega de software mais cadenciado e consistente. Enquanto, o PO, o Analista de Qualidade e o Analista de Sistemas trabalham na etapa de *discovery* para a próxima *sprint*, os desenvolvedores trabalham na *sprint* de construção. Essa abordagem estabelece um trabalho prévio de detalhamento das coisas que precisam ser feitas, antes do desenvolvimento propriamente dito, por meio de ciclos curtos e sustentáveis, de modo que todos sintam-se confortáveis com o que precisa ser feito no tempo disponível.

6.4.2.5 Capacite a Equipe no Método e nas Tecnologias Utilizadas no Projeto Real

O treinamento formal em métodos ágeis como XP, Scrum e Kanban para as equipes é fortemente recomendado. Porém, uma vez que aulas teóricas raramente remetem ao contexto do objeto de estudo, é melhor transmitir a ideia sobre métodos ágeis através de capacitações internas focadas em questões práticas do gerenciamento e desenvolvimento do projeto real. Algumas equipes recomendaram a contratação de serviço de *mentoring* em métodos ágeis para acompanhar a execução do projeto, principalmente nas primeiras iterações, as quais são mais difíceis (ECa03) (ECa04). Na mesma direção, a ausência de conhecimento e experiência nas tecnologias utilizadas também é prejudicial ao andamento do projeto. A equipe (ECa01) relatou atrasos por conta da falta de domínio nas tecnologias adotadas. Assim, a capacitação e o acompanhamento de pessoas mais experientes nas metodologias e tecnologias adotadas no projeto são fortemente recomendados.

6.4.2.6 Forneça Ferramentas de Apoio ao Processo de Desenvolvimento de Software

As equipes precisam de boas ferramentas de apoio ao processo de desenvolvimento de software. Isso pode parecer uma recomendação óbvia, mas em ambientes de governo, identificação, análise, avaliação e aquisição podem acarretar tempos substanciais para aprovação. Por conta disso, ferramentas de código aberto são uma alternativa economicamente viável e imediata para apoiar o trabalho de desenvolvimento de software. Na mesma direção, ferramentas visuais, como parede de cartões, também são usadas para o gerenciamento do projeto e valorização do trabalho em equipe.

6.4.2.7 Acompanhe o Progresso do Projeto Diariamente e Visivelmente

Em vários projetos, os gerentes não conseguem visualizar os pequenos problemas diários e os desenvolvedores não conseguem descrevê-los com clareza. Eles não conseguem tomar medidas eficientes para recuperar os atrasos diários, então no momento em que os atrasos se tornam grandes o suficiente para serem vistos, em muitos casos, é tarde demais para se fazer qualquer coisa sobre eles [Hum10]. Este é o motivo pelo qual os projetos executados por gestores competentes e experientes continuam a ter problemas de custo, cronograma e qualidade [Hum10]. Para solucionar esse problema, equipes têm coletado informações diárias para gerenciar seu próprio trabalho, bem como, têm medido com precisão o andamento do projeto no intervalo de um dia por meio de ferramentais visuais e eletrônicas e gráficos de *burns*. Dessa maneira, a gerência pode visualizar pequenos problemas diariamente antes que eles se tornem mais sérios, podendo tomar medidas cabíveis para identificar suas causas e solucionar problemas a tempo.

6.4.2.8 Explore Múltiplas Formas de Comunicação

Métodos ágeis encorajam a comunicação frente a frente aproximando as pessoas e valorizando o relacionamento humano [Bec99]. Equipes colocalizadas dependem muito da comunicação direta e constante e do *feedback* imediato (ECa02) (ECa03) (ECa04). Porém, no caso de equipes distribuídas, onde os clientes, seus representantes e a equipe de desenvolvimento estão distantes geograficamente, outras formas de comunicação também são úteis e necessárias (ECa01). E-mails são usados para comunicação e ferramentas de gerenciamento de conteúdo são usadas para o compartilhamento e disseminação de informações. Reuniões semanais são realizadas por meio de videoconferências, tal como, pessoas estão sempre disponíveis por telefone. Assim, não há outra saída, equipes distribuídas terão que explorar múltiplas formas de comunicação mais do que equipes colocalizadas.

6.4.2.9 Evite Alocar a Mesma Pessoa em Vários Projetos Concomitantes

A forma corporativa da abordagem multitarefa é a criação de muitos projetos concomitantes [Coh11]. Quando uma empresa assume projetos demais, as pessoas são divididas entre vários projetos, o que leva à abordagem multitarefa individual. O efeito danoso da abordagem multitarefa apareceu nos estudos de caso (ECa01) (ECa02), onde esses projetos demoraram mais e impediram qualquer prática de trabalho diário e colaborativa. Mary e Tom Poppendieck [Pop11] recomendam às empresas que limitem o trabalho segundo a capacidade. Uma empresa com mais projetos executados concomitantemente do que a mão de obra que pode ser alocada está tentando trabalhar além de sua capacidade. Como eles colocam, “Se você espera que as equipes cumpram prazos agressivos, deve limitar o trabalho segundo a capacidade delas”. Na mesma direção, Cohn [Coh11] aponta que pessoas alocadas para trabalhar em vários projetos inevitavelmente produzem menos resultados e recomenda alocar as pessoas para apenas um projeto, o que corrobora com a forma com que os projetos (ECa03) (ECa04) estruturam suas equipes para o desenvolvimento de software.

Embora seja uma recomendação conhecida pela comunidade ágil, na AP não é incomum organizações ignorarem-na. Isso se explica nas OP porque muitos projetos de software são de uso interno da própria organização e eles precisam ser mantidos ao longo do tempo. Assim, em algum momento a quantidade de sistemas existentes para manutenção e a quantidade de novos sistemas para desenvolvimento serão bem maiores do que a própria capacidade da organização em executá-los. Como resultado, os desenvolvedores acabam responsabilizando-se por mais de um sistema

concomitantemente, o que aumenta consideravelmente o tempo de execução de novos projetos e diminui consideravelmente a quantidade de pessoas disponíveis para participar de novos projetos (ECa01) (ECa02).

6.4.2.10 Satisfaça as Exigências “Legais” da Organização

As OP são pautadas por normas, regulamentos, regras e preenchimento de documentos obrigatórios. Embora o preenchimento de muitos desses artefatos possa não adicionar valor algum ao produto, recomenda-se fortemente satisfazê-lo, uma vez que muitos deles são exigidos pelo ambiente naquele momento e não podem ser removidos imediatamente. Pode ser que essa abordagem não seja sustentável no longo prazo, mas é um passo inicial numa iniciativa de mudança organizacional mais abrangente.

6.4.2.11 Estabeleça Expectativas de Documentação

Três equipes estudadas (ECa01) (ECa02) (ECa03) afirmaram ser aceitável alguma documentação, principalmente a documentação relacionada ao processo de negócio. Assim, os documentos escritos - definitivamente - não devem ser abandonados. Pelo contrário, eles precisam ser definidos e escritos ao longo do projeto. Porém, no governo brasileiro, para algumas equipes entrevistadas não está claro quais são as documentações obrigatórias para fins de auditoria, mesmo para projetos executados internamente.

6.4.2.12 Experimente Métodos e Práticas Emergentes

A primeira frase do manifesto ágil afirma que “Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo” [BBB+01]. Pode-se inferir, a partir dessa declaração, que ainda há muito a ser aprendido e ensinado. Como dito anteriormente, o desenvolvimento ágil de software não é um processo acabado, ele precisa ser continuamente melhorado e suas experiências precisam ser continuamente transmitidas a outras pessoas. Taiichi Ohno, criador do Sistema Toyota de Produção, escreveu que “há algo chamado trabalho padrão, mas padrões devem ser alterados constantemente. Se você considera o padrão como o melhor que pode fazer, não há avanço” [apud Coh11]. Por conta disso, o CNPTIA e a PROCERGS permitem a experimentação de novos métodos e práticas de desenvolvimento de software em novos projetos, de modo a amadurecer as práticas e as tecnologias existentes visando encontrar o que é mais adequado para o projeto. A recomendação é melhorar continuamente a adoção de métodos ágeis, identificando, experimentando e avaliando métodos e práticas emergentes de modo a melhorar e aperfeiçoar continuamente o conhecimento existente.

6.4.3 Aprendizado

No âmbito das OP, faltam estudos teóricos subsequentes as primeiras adoções de métodos ágeis. Por esse motivo, as recomendações sugeridas nesta fase foram construídas com base somente nos estudos empíricos. A última fase proposta diz respeito ao aprendizado, sendo formado por duas recomendações (Figura 7).

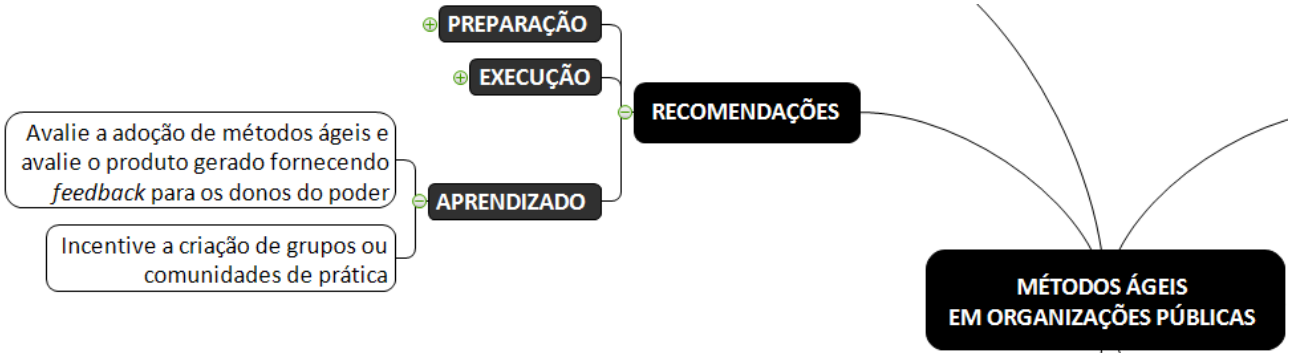

Figura 7 - Fase de Aprendizado.

6.4.3.1 Avalie a Adoção de Métodos Ágeis e Avalie o Produto Gerado Fornecendo Feedback para os Donos do Poder

O conjunto de recomendações para a adoção de métodos ágeis começa obtendo o apoio dos donos do poder, recrutando pessoas dispostas a mudanças e colocando métodos ágeis em prática em projetos-piloto importantes. É crucial que as equipes dos projetos-piloto demonstrem para os donos do poder (alta administração) os benefícios advindos na adoção de métodos ágeis. No CNPTIA e na PROCERGS, por exemplo, não se encontrou nada mais útil na promoção do sucesso do projeto do que um cliente que alcançou mais resultados do que esperava. Um cliente conversando com a alta administração que um projeto recente aplicou métodos ágeis e conquistou resultados melhores do que projetos anteriores seria ideal para fazer a alta administração pedir a outras equipes que também experimentem e avaliem a nova abordagem.

De maneira objetiva, algumas equipes coletaram métricas quantitativas, o que permitiu às OP olharem para métodos ágeis de diferentes pontos de vista. Por exemplo, uma equipe indicou que XP foi aproximadamente duas vezes mais produtivo do que projetos semelhantes executados anteriormente com outras abordagens (RSL16). Em um outro caso, o resultado foi que a OP conseguiu construir com somente 52% do orçamento 88% do sistema em apenas um ano contra anos de atraso (ROE03). Por último, equipes demonstraram a redução da dependência de pessoas externas, o que reduziu custos e mostrou o desenvolvimento de habilidades e aptidões na equipe (ROE04). Esses são

alguns exemplos de pontos de vista sob os quais a adoção de métodos ágeis pode ser examinada em OP.

6.4.3.2 Incentive a Criação de Grupos ou Comunidades de Prática

Em métodos ágeis, pessoas e equipes auto-organizáveis possuem a tendência de otimizar e melhorar as coisas para elas próprias. Mas é importante que elas também fiquem atentas ao que está acontecendo na organização como um todo, bem como, a forma com que seus pares (pessoas que executam a mesma função) têm realizado o seu trabalho em outras equipes. A recomendação é que a troca de experiências entre as equipes e os pares é algo positivo para socializar o conhecimento e estabelecer condutas na organização, bem como, melhorar a forma como a organização utiliza métodos ágeis. Na PROCERGS, por exemplo, encontros periódicos entre *ScrumMaster* e *Product Owner* são realizados para gerar sinergia e aprendizado contínuo entre as pessoas, o que sugere mais tempo e condições para as pessoas compartilharem conhecimento.

6.5 Considerações Finais do Conjunto de Recomendações

O conjunto de recomendações sugere que OP querendo realizar mudanças para métodos ágeis, precisam, antes de mais nada, trabalhar em si mesmas gradualmente. Há indícios de que não se pode transformar imediatamente a cultura do desenvolvimento de software de uma OP para ágil. As evidências mostram que para ser bem-sucedido alguns projetos-piloto precisam ser executados à priori; e se os resultados forem sustentáveis, então novos investimentos são realizados, refinando e expandindo as primeiras experiências para outros projetos dentro da organização. Além disso, encontrou-se indícios de que não existe nada que se pareça novo que seja de aceitação imediata, à medida em que os resultados e as evidências vão aparecendo as incertezas e as resistências também vão diminuindo. Dessa maneira, se os benefícios forem visíveis para a organização e seus empregados, então, o resultado emergente será um processo de mudança da cultura da organização, o qual precisa ser fortemente apoiado pela alta administração (de cima para baixo) e precisa de engajamento dos empregados que estão executando os novos métodos de trabalho (de baixo para cima).

Por último, com a identificação das fases e as recomendações envolvidas em cada uma delas, foi possível consolidar uma figura mostrando todas as recomendações e suas fases, as quais estão envolvidas na adoção de métodos ágeis em OP (Figura 8).

PREPARAÇÃO

1

- ☑ Estabeleça uma estratégia de gestão de mudanças com base nas pessoas
- ☑ Comece com pessoas dispostas a mudanças
- ☑ Envolva e comprometa os donos do poder
- ☑ Comece com projetos-piloto importantes
- ☑ Forneça as condições necessárias para as pessoas executarem o seu trabalho

EXECUÇÃO

2

- ☑ Promova o trabalho em equipe preferencialmente com equipes pequenas e perenes
- ☑ Envolva, comprometa e satisfaça os clientes e os usuários
- ☑ Assegure a qualidade do código e do produto
- ☑ Estabeleça o tempo das entregas de software preferencialmente em tempos mais curtos
- ☑ Capacite a equipe no método e nas tecnologias utilizadas no projeto real
- ☑ Forneça ferramentas de apoio ao processo de desenvolvimento de software
- ☑ Acompanhe o progresso do projeto diariamente e visivelmente
- ☑ Explore múltiplas formas de comunicação
- ☑ Evite alocar a mesma pessoa em vários projetos concomitantes
- ☑ Satisfaça as exigências legais da organização
- ☑ Estabeleça expectativas de documentação
- ☑ Experimente métodos e práticas emergentes

APRENDIZADO

3

- ☑ Avalie a adoção de métodos ágeis e avalie o produto gerado fornecendo *feedback* para os donos do poder
- ☑ Incentive a criação de grupos ou comunidades de prática

Figura 8 - Consolidação das recomendações.

7 CONSIDERAÇÕES FINAIS

Este estudo encontrou evidências sobre a viabilidade da adoção de métodos ágeis em OP, mostrando ser possível aplicá-los neste contexto também. Porém, muito do que se sabe sobre os efeitos da adoção de métodos ágeis em OP não são relativamente novos para a comunidade de ES porque eles praticamente repetiram-se em outros tipos de organizações e corroboram com as teorias existentes de métodos ágeis. Isto se explica porque nos dias atuais há pistas de que as OP têm aplicado tecnologias e metodologias bem depois delas terem sido experimentadas e avaliadas pela indústria de software e a academia. Ou seja, a indústria de software e a academia, muitas das vezes, são protagonistas das mudanças em tecnologias e metodologias, criando-as, experimentando-as e avaliando-as primeiramente. Mais tarde, as OP percebem oportunidades externas e as absorvem, procurando repetir muitas das experiências já conhecidas.

Mesmo que as histórias e os problemas encontrados no setor público possam parecer repetidos comparativamente a outros setores aparentando não haver algo novo para a comunidade de ES, o universo tecnológico está em constante mudança e evolução e essas histórias e problemas sempre apresentam nuances que tornam a busca de soluções diferentes e inovadoras algo extremamente estimulante [Car14]. Assim, a luz da ciência, este estudo contribui com um conjunto de recomendações - além de um arcabouço de resultados teóricos e empíricos - para as OP começarem a refletir sobre seus métodos atuais de trabalho encorajando-as aos métodos ágeis de modo a alcançar seus benefícios de maneira responsiva.

O arcabouço de resultados teóricos e empíricos e o conjunto de recomendações para a adoção de métodos ágeis é mais uma tentativa na busca de respostas e passos consistentes em direção ao desenvolvimento ágil de software em OP, trazendo informações úteis de como adotar métodos ágeis nesse contexto. As análises apresentam indícios de que uma boa alternativa é iniciar a adoção de métodos ágeis com equipes piloto dispostas a mudanças fortemente apoiadas pela alta gerência em projetos-piloto importantes; depois a mudança dependerá da ampliação e interação com outras equipes visando alcançar a grande maioria da organização; na sequência, essas pessoas trabalham pela constituição de um ecossistema de aprendizado e melhoria contínua para não acomodar-se conscientemente ou inconscientemente com as primeiras conquistas e subsequentemente com a zona de conforto. Isto demonstra que as pessoas realmente precisam praticar métodos ágeis, a fim de entendê-los melhor. Do ponto de vista da pesquisa científica, a legitimação do conjunto de recomendações proposto para a adoção de métodos ágeis em

OP é endossada pelo processo de pesquisa como um todo, descrito no capítulo 2 e na seção 4.1.

Por último, em 2001, quando os dezessete proponentes do desenvolvimento ágil de software chegaram a uma nova visão para o desenvolvimento de software, eles individualmente continuaram por muito tempo ensinando sobre práticas essenciais, como “escrever estórias em cartões de papel”, “trabalhar em equipe”, “escrever bons testes”, “tornar o trabalho visível”, “obter *feedback* do cliente toda semana”, etc. Isto mostra que a compreensão dos valores e princípios do manifesto ágil é de fácil assimilação, porém praticá-lo é mais difícil. Como constatado, o desenvolvimento de software no governo guiou-se por métodos prescritivos durante décadas. Isto significa que aqueles que trabalham com desenvolvimento ágil em OP precisam compartilhar suas experiências e ajudar uns aos outros, a fim de dar a este método a chance de acontecer o mesmo.

7.1 Contribuições da Pesquisa

Muitas pessoas estão interessadas em métodos ágeis no governo, porém elas não sabem como começar, então elas precisam de estudos introdutórios sobre a adoção de métodos ágeis no setor público. Neste sentido, as principais contribuições desta pesquisa são o arcabouço de resultados teóricos e empíricos e o conjunto de recomendações para a adoção de métodos ágeis. O conjunto de recomendações é mais uma contribuição para as OP refletirem, atualizarem, melhorarem e expandirem sua competência no desenvolvimento de software por meio da visão de modernidade e agilidade proposta pelos métodos ágeis.

Não obstante, esta pesquisa retroalimenta o conhecimento científico da área agregando à academia algumas características sobre as OP, bem como, alguns aspectos sobre o desenvolvimento de software e a adoção de métodos ágeis no setor público. Por último, este estudo aprofundou os conhecimentos do aluno em métodos ágeis, tornando-o um profissional apto a contribuir com a academia e OP como potencial especialista no assunto.

Com relação a questão de pesquisa deste estudo, ela foi respondida ao longo do processo de elaboração do conjunto de recomendações, identificaram-se algumas evidências científicas, algumas publicações governamentais e uma obra especializada sobre o tema, formando a base teórica sobre o assunto e mostrando a viabilidade da adoção de métodos ágeis em OP. Nesta fase, identificaram-se as principais razões, benefícios, problemas e desafios, práticas utilizadas e lições aprendidas sobre o tema. Adicionalmente, o resultado do estudo teórico formou o arcabouço de resultados teóricos e empíricos

preliminar. Posteriormente, este arcabouço foi ampliado e consolidado com os resultados dos estudos empíricos executados em duas OP de médio e grande porte respectivamente. Na sequência, estes estudos teóricos e empíricos forneceram informações para criação do conjunto de recomendações para a adoção de métodos ágeis em OP, o qual foi revisto, avaliado e endossado por dois servidores públicos com experiência em gestão de mudanças em OP envolvendo TI.

7.2 Limitações da Pesquisa

Durante a execução deste estudo, identificou-se algumas limitações, as quais são apresentadas a seguir. Em primeiro lugar, a revisão sistemática contribuiu para identificar estudos primários dentro do domínio pesquisado. Embora a escolha das bases de dados para consulta tenha procurado abranger o máximo das principais alternativas científicas disponíveis para execução de uma revisão sistemática, é possível que outras fontes de publicações científicas não utilizadas neste estudo também contenham artigos sobre o desenvolvimento ágil de software em OP. Portanto, não é possível garantir a cobertura total de artigos científicos sobre esse assunto.

Em segundo lugar, a definição à priori da estratégia de busca faz com que o conhecimento seja visto como algo bem definido e explícito. Entretanto, nesta pesquisa novos conhecimentos foram descobertos durante a execução da revisão sistemática, o que poderia ter resultado em uma reformulação da estratégia de busca, juntamente com uma nova execução de todo o processo com as novas descobertas. Isto significa que uma descrição inicial precisa da realidade torna difícil a aplicação desta prática com êxito, uma vez que o conhecimento pode ser também mal definido, tácito e difuso.

Em terceiro lugar, o conjunto de recomendações para a adoção de métodos ágeis que deve ser interpretado com cautela. Ele foi construído com base em estudos teóricos e empíricos executados especificamente em OP, onde recomendações para projetos de órgãos governamentais não são generalizáveis para projetos do setor privado [Ban08]. Porém, ao contrário da percepção comum, o estudo de Krogstie [Kro12] realizado com OP norueguesas não encontrou diferenças significativas na eficiência no desenvolvimento de novos sistemas e manutenção de sistemas existentes entre o setor público e o setor privado. Ademais, Ewan Ferlie [apud Ban08] argumenta que, apesar das diferenças, as OP não são radicalmente diferentes das organizações privadas na sua estrutura organizacional.

Em quarto lugar, o número de projetos estudados na parte empírica da pesquisa, restringe a generalização dos resultados obtidos. Não obstante, eles foram dominados por casos que foram percebidos como bem-sucedidos pelos sujeitos. Isto pode ter limitado ou influenciado os resultados da pesquisa. Deve-se, entretanto, destacar que especificamente o arcabouço de resultados teóricos e empíricos e o conjunto de recomendações são apoiados também na base teórica estudada, o que permite uma maior segurança nas constatações e conclusões obtidas.

Em quinto lugar, esta pesquisa concentrou-se apenas em projetos de desenvolvimento de software. Outros tipos de projeto, como infraestrutura de TI e subcontratação também são comuns e poderiam contribuir com percepções relevantes. Por último, o conjunto de recomendações trata-se de uma proposta inicial.

7.3 Trabalhos Futuros

Se esta proposta inicial se generalizar, então várias implicações práticas surgem principalmente para gerentes de projeto, desenvolvedores, clientes e usuários. Essa perspectiva aponta para a realização de uma revisão pós-aplicação (RPA) incluindo explicitamente uma avaliação das recomendações e suas fases. A RPA é uma oportunidade para avaliar e melhorar o conjunto de recomendações e garantir que as lições aprendidas serão incorporadas em projetos subsequentes.

Como visto, métodos ágeis para o desenvolvimento de software estão sendo gradativamente adotados em vários projetos de governo. Sua abordagem fornece meios para resolver muitos problemas comuns enfrentados no desenvolvimento de software na AP, incluindo o tempo de entrega e a aderência do produto de software as necessidades dos clientes. Governos têm aplicado diversas práticas para superar esses desafios comuns, incluindo o desenvolvimento iterativo com base em entregas frequentes priorizadas por valor de negócio, maior participação do cliente no processo de desenvolvimento de software, testes automatizados, integração contínua, etc. Como estas e outras práticas estão tornando-se cada vez mais prevalentes na indústria e no governo para o desenvolvimento de sistemas de informação, vislumbra-se a oportunidade de incorporá-las no desenvolvimento de software científico também, semelhantemente ao que aconteceu no projeto da *NASA Langley Research Center* e conforme apontou um dos entrevistados do CNPTIA. Identifica-se grande potencial de crescimento nesta linha de pesquisa, onde os pontos fortes envolvem uma parceria estável entre a academia e centros de pesquisa, criando condições de experimentação e aprendizagem únicas para o estudo de métodos ágeis no desenvolvimento de software científico.

Além disso, com o crescimento gradativo de métodos ágeis em OP, levantar como está o estágio atual de adoção de métodos ágeis no governo brasileiro é importante para mostrar como as OP como um todo estão em relação aos métodos ágeis. Assim, vislumbra-se a oportunidade de executar uma pesquisa quantitativa para identificar o estado da prática de métodos ágeis em OP, tomando como base o trabalho da empresa VersionOne [Ver15], bem como, a pesquisa executada pelo Departamento de Ciência da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP) [MSC+12].

Por último, como OP não visam lucros financeiros e os maiores benefícios de métodos ágeis para a AP concentram-se na satisfação das pessoas com os resultados alcançados e a forma como eles foram atingidos. Então, novas estratégias de gerenciamento de benefícios e novas medidas de performance e resultados precisam ser estudadas, elaboradas e avaliadas, incluindo, por exemplo, a percepção da melhoria na qualidade de vida pelos empregados, a percepção do cliente da melhoria no processo de trabalho e na qualidade do produto, os benefícios que o produto de software estão proporcionando para as pessoas produzindo lucros sociais entre outros.

REFERÊNCIAS BIBLIOGRÁFICAS

[ACT01] Ahern, D.; Clouse, A.; Turner, R. “CMMI(SM) Distilled: A Practical Introduction to Integrated Process Improvement”. Boston: Addison-Wesley, 2001, 336p.

[Agi14] Agile Alliance. “Agile Alliance: Home”. Capturado em: http://www.agilealliance.org/, Novembro 2014.

[Amb98] Ambler, S. “Process Patterns: Building Large-Scale Systems Using Object Technology”. Cambridge University Press/SIGS Books, 1998, 582p.

[And10] Anderson, D. J. “Kanban: successful evolutionary change for your technology business”. Sequim: Blue Hole Press, 2010, 261p.

[App12] Appelo, J. “Como Mudar o Mundo: Gestão de Mudanças 3.0”. Tradução de André Faria Gomes. 2012, Não paginado.

[ASR+02] Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. “Agile software development methods: Review and analysis”, VTT Publications 478, 2002, 107p. Capturado em: http://goo.gl/T79xvR, Dezembro 2014.

[AuP07] Audy, J. L. N.; Prikladnicki, R., “Desenvolvimento Distribuído de Software: Desenvolvimento de Software com Equipes Distribuídas”. Rio de Janeiro: Editora Campus/Elsevier, 2007, 232p.

[Bal10] Balbe, R. da S. “Uso de tecnologias de informação e comunicação na gestão pública: exemplos no governo federal”. *Revista do Serviço Público*, vol. 61-2, Abr-Jun 2010, pp.189-209.

[Ban08] Bannerman, P. L. “Risk and risk management in software projects: A reassessment”. *Journal of Systems and Software*, vol. 81-12, Dez 2008, pp. 2118-2133. DOI: 10.1016/j.jss.2008.03.059

[Bar11] Bardin, L. “Análise de conteúdo”. São Paulo: Edições 70, 2011, 279p.

[Bas08] Bassi Filho, D. L. “Experiências com desenvolvimento ágil”, Dissertação de Mestrado (Mestrado em Ciências - Área de concentração: Ciência da Computação), Instituto de Matemática e Estatística da Universidade de São Paulo, IME-USP, 2008, 154p.

[Bas14] Bassi, D. “Capítulo 4: Programação Extrema (XP)”. In: Métodos Ágeis para Desenvolvimento de Software, Porto Alegre: Bookman, 2014, pp. 37-57.

[BBB+01] Beck, K; Beedle, M.; Bennekum, A. van; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R. C.; Mellor, S.; Schwaber, K.; Sutherland, J.; Thomas, D. “Manifesto for agile software development”. Capturado em: http://agilemanifesto.org/, Junho 2014.

[BeA04] Beck, K; Andres, C. “Extreme Programming Explained: Embrace Change”. Boston: Addison-Wesley, 2004, 2nd Edition, 224p.

[Bec99] Beck, K. “Extreme programming explained: embrace change”. Boston: Addison-Wesley, 1999, 190p.

[Bha12] Bhattacherjee, A. “Social Science Research: Principles, Methods, and Practices”, University of South Florida, USF Tampa Library Open Access Collections, 2012, 149p. Capturado em: http://goo.gl/ZQlMc0, Dezembro 2014.

[BLK+10] Boehm, B.; Lane, J. A.; Koolmanojwong, S.; Turner, R. “Architected Agile Solutions for Software-Reliant Systems”. In: Agile Software Development, Springer Berlin Heidelberg, 2010, pp. 165-184. DOI: 10.1007/978-3-642-12575-1_8

[Boe02] Boehm, B. “Get Ready for Agile Methods, with Care”. *IEEE Computer*, vol. 35-1, Jan 2002, pp. 64-69. DOI: 10.1109/2.976920

[Boe06] Boehm, B. “A view of 20th and 21st century software engineering”. In: 28th international conference on Software engineering (ICSE ’06), 2006, pp. 12-29.

[Boe10] Boeg, J. “Kanban em 10 passos”. Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt. C4Media, 2010, 43p. Capturado em: http://www.infoq.com/br/minibooks, Dezembro 2014.

[Boe88] Boehm, B. W. “A spiral model of software development and enhancement”. *IEEE Computer*, vol. 31-5, Mai 1988, pp. 61-72. DOI: 10.1109/2.59

[BoT03] Boehm, B.; Turner, R. “Balancing Agility and Discipline: A Guide for the Perplexed”. Boston: Addison-Wesley, 2003, 304p.

[Bra85] BRASIL. Secretaria Especial de Informática. “Master plan: Software Plant Project”. Campinas: CTI, EMBRAPA, SERPRO, EMBRATEL, 1985, Não paginado.

[Car14] Caroli, P. (Ed.). “ThoughtWorks - Antologia Brasil - Histórias de aprendizado e inovação”. São Paulo: Casa do Código, 2014, 267p.

[Car99] Carmel, E. “Global software teams: collaborating across borders and time zones”. Upper Saddle River, NJ: Prentice Hall, 1999, 269p.

[CaS95] Carnegie Mellon University, Software Engineering Institute. “The Capability Maturity Model: Guidelines for Improving the Software Process”. Addison-Wesley, 1995, 464p.

[CFD+14] Coelho, E. A.; Fouro, A. M. M.; D' Oliveira, F. M.; Silva, A. C. da; Souza, D. R. Q. de; Macedo, C. F. M. de; Crespo, M. da S.; Almeida, C. F. A. de; Visoli, M. C.; NUNES, L. C. de M.; Lopes Junior, S. “Guia de uso do modelo corporativo de processos de software da Embrapa (MCPSE)”. Belém, PA: Embrapa Amazônia Oriental, 2014, 33p.

[CiC10] Ciccozzi, F.; Crnković, I. “Performing a project in a distributed software development course: lessons learned”. In: Proceedings of the 5th IEEE International Conference on Global Software Engineering (ICGSE), 2010, pp. 187-191.

[Coc02] Cockburn, A. “Agile Software Development”. Boston: Addison-Wesley, 2002, 278p.

[CoH01] Cockburn, A; Highsmith, J. “Agile Software Development: The People Factor”. *IEEE Computer*, vol. 34-11, Nov 2001, pp. 131-133. DOI: 10.1109/2.963450

[Coh11] Cohn, M. “Desenvolvimento de software com Scrum: aplicando métodos ágeis com sucesso”. Tradução de Aldir José Coelho Corrêa da Silva. Porto Alegre: Bookman, 2011. 496p.

[COM13] COMPUTERWORLD. “Healthcare.gov website 'didn't have a chance in hell'“. Capturado em: http://goo.gl/6lhG5c, Novembro 2014.

[Cre14] Creswell, J. W. “Research Design: Qualitative, Quantitative, and Mixed Methods Approaches”. Thousand Oaks, CA: SAGE Publications Inc, 2014, 4.ed., 273p.

[Cri14] Crisp's Blog. “The Solution to Technical Debt”. Capturado em: http://goo.gl/bFTwVJ, Dezembro 2014.

[Cun92] Cunningham, W. “The WyCash portfolio management system”. In: OOPSLA '92 Addendum to the proceedings on Object-oriented programming systems, languages, and applications, 1992, pp. 29-30. DOI: 10.1145/157709.157715

[DCO14] DSDM Consortium. “DSDM Consortium”. Capturado em: http://www.dsdm.org/, Novembro 2014.

[DDM10] Dingsøy, T.; Dybå, T.; Moe, N. B. “Agile Software Development: Current Research and Future Directions”. Berlin: Springer, 2010, 238p.

[Des14] DesenvolvimentoAgil.com.br. “Aprenda sobre Desenvolvimento Ágil de Software”. Capturado em: http://desenvolvimentoagil.com.br/, Novembro 2014.

[DNB+12] Dingsøyr, T.; Nerur, S.; Balijepally, V.; Moe, N. B. “A decade of agile methodologies: Towards explaining agile software development”. *Journal of Systems and Software*, vol. 85-6, Jun 2012, pp. 1213-1221. DOI: 10.1016/j.jss.2012.02.033

[DyD08] Dybå, T.; Dingsøyr, T. “Empirical studies of agile software development: A systematic review”. *Information Software Technology*, vol. 50-10, Ago 2008, pp. 833-859. DOI: 10.1016/j.infsof.2008.01.006

[EIA08] Embrapa Informática Agropecuária. “IV Plano Diretor da Embrapa Informática Agropecuária 2008-2011-2023”. Campinas, 2008, 47p. Capturado em http://goo.gl/V7BWEx, Setembro 2014.

[EMB86] Embrapa. Núcleo Tecnológico para Informática Agropecuária. “Plano Diretor de Informática - 1986/1990”. Campinas, 1986, 55p.

[Fra14] Franco, C. A. de C. “Análise da legislação para a terceirização do desenvolvimento de software na administração pública brasileira em relação as norte-americana e britânica”, Dissertação de Mestrado, Programa de Pós-Graduação em Informática, UFRJ, 2014, 141p.

[Fre98] Fresneda, P. S. V. “Transformando organizações públicas: a tecnologia da informação como fator propulsor de mudanças”. *Revista do Serviço Público*, vol. 49-1, Jan-Abr 1998, pp. 71-91.

[GAO09] US GAO. “Defense Acquisitions: Measuring the Value of DOD's Weapon Programs Requires Starting with Realistic Baselines”. United States Government Accountability Office, 2009, 16p. Capturado em: http://goo.gl/mxhTrM, Novembro 2014.

[GAO12] US GAO. “Software Development: Effective Practices and Federal Challenges in Applying Agile Methods”, United States Government Accountability Office, 2012, 34p. Capturado em: http://goo.gl/gBg7jH, Novembro 2014.

[GDS14a] UK GDS. “Blog Government Digital Service: Getting approval for agile spending”. Capturado em: http://goo.gl/jFHUl7, Novembro 2014.

[GDS14b] UK GDS. “Blog Government Digital Service: 200 days of transformation, 200 days to go”. Capturado em: http://goo.gl/J9aH5D, Dezembro 2014.

[GDS14c] UK GDS. “Blog Government Digital Service: The role of the agile wall at GDS”. Capturado em: http://goo.gl/beihkq, Dezembro 2014.

[GDS14d] UK GDS. “Blog Government Digital Service: A year in the making - the Digital by Default Service Standard”. Capturado em: http://goo.gl/MWDsaf, Dezembro 2014.

[GDS14e] UK GDS. “Blog Government Digital Service”. Capturado em: https://gds.blog.gov.uk/, Dezembro 2014.

[God95] Godoy, A. S. “Introdução à pesquisa qualitativa e suas possibilidades”. *Revista de Administração de Empresas*, vol. 35-2, Jul-Ago 1995, pp. 57-63.

[Gol07] Goldfinch, S. “Pessimism, Computer Failure, and Information Systems Development in the Public Sector”. *Public Administration Review*, vol. 67-5, Set-Out 2007, pp. 917-929. DOI: 10.1111/j.1540-6210.2007.00778.x

[Gom13] Gomes, A. F. “Agile - Desenvolvimento de software com entregas frequentes e foco no valor de negócio”. São Paulo: Casa do Código, 2013, 151p.

[Gra10] Graham, A. “Como escrever e usar estudos de caso para ensino e aprendizagem no setor público”. ENAP Escola Nacional de Administração Pública, 2010, 212p. Capturado em http://goo.gl/PYuDJJ, Novembro 2014.

[GWR14] Gomes, A.; Willi, R.; Rehem S. “Capítulo 1: O Manifesto Ágil”. In: Métodos Ágeis para Desenvolvimento de Software, Porto Alegre: Bookman, 2014, pp. 3-15.

[HaD08] Hazzan, O.; Dubinsky, I. “Introduction to Agile Software Development”. In: Agile Software Engineering, 2008, pp. 1-24. DOI: 10.1007/978-1-84800-198-5_1

[Han08] Hantos, P. “Defense Acquisition Performance Assessment: The Life-Cycle Perspective of Selected Recommendations”, Aerospace Report TOR-2008 (8550)-8027, The Aerospace Corporation, 2008, Não paginado.

[Hei10] Heil, J. W. “Addressing the Challenges of Software Growth and Rapidly Evolving Software Technologies”. *Naval Engineers Journal*, vol. 122-4, Dez 2010, pp. 45-58. DOI: 10.1111/j.1559-3584.2010.00279.x

[HeM03] Herbsleb, J. D.; Mockus, A. “An empirical study of speed and communication in globally distributed software development”. *IEEE Transactions on Software Engineering*, vol. 29-6, Jun 2003, pp. 481-494. DOI: 10.1109/TSE.2003.1205177

[Her07] Herbsleb, J. D. “Global software engineering: the future of socio technical coordination”. In: Proceedings of the Future of Software Engineering (FOSE’07), 2007, pp. 188-198. DOI: 10.1109/FOSE.2007.11

[HHM10] Hellmann, T. D.; Hosseini-Khayat, A.; Maurer, F. “Agile Interaction Design and Test-Driven Development of User Interfaces - A Literature Review”. In: Agile Software Development - Current Research and Future Directions, Springer Berlin Heidelberg, 2010, pp. 185-201. DOI: 10.1007/978-3-642-12575-1_9

[Hig02] Highsmith, J., “Agile Software Development Ecosystems”. Boston: Addison-Wesley, 2002, 448p.

[HMT11] HM Treasury. “Major Project approval and assurance guidance”. HM Treasury, CabinetOffice, 2011, 19p. Capturado em: http://goo.gl/59oYLj, Novembro 2014.

[HMT14] HM Treasury. “Guidance: Agile digital and IT projects: clarification of business case guidance”. Capturado em: http://goo.gl/GZ3HHV, Novembro 2014.

[Hof13] Hoffman-Câmara, R. “Análise de Conteúdo: da teoria à prática em pesquisas sociais aplicadas às organizações”. *Revista Interinstitucional de Psicologia*, vol. 6-2, Jul-Dez 2013, pp. 179-191.

[Hum00] Humphrey, W. S. “The Team Software Process”. Software Engineering Institute, Carnegie Mellon, 2000, 36p. Capturado em: http://goo.gl/sTZNmq, Novembro 2014.

[Hum10] Humphrey, W. S. “Why Can’t We Manage Large Projects?”. *CrossTalk: The Journal of Defense Software Engineering*, vol. 23-4, Jul-Ago 2010, pp. 4-7. Capturado em: http://goo.gl/3mGgzR, Dezembro 2014.

[Hum96] Humphrey, W. S. “Using a defined and measured Personal Software Process”. *IEEE Software*, vol. 13-3, Mai 1996, pp. 77-88. DOI: 10.1109/52.493023

[Iil10] Iivari, J.; Iivari, N. “Organizational Culture and the Deployment of Agile Methods: The Competing Values Model View”. In: Agile Software Development - Current Research and Future Directions, Springer Berlin Heidelberg, 2010, pp. 203-222. DOI: 10.1007/978-3-642-12575-1_10

[ISO12] ISO/EIC 15504-5:2012. “Information technology -- Process assessment -- Part 5: An exemplar software life cycle process assessment model”. International Organization for Standardization, 2012, 196p.

[Jac02] Jacobson, I. “A Resounding ‘Yes’ to Agile Processes - But Also More”. *Cutter IT Journal*, vol. 15-1, Jan 2002, pp. 18-24.

[JaM08] Japiassú, H.; Marcondes, D. “Dicionário Básico de Filosofia”. Rio de Janeiro: Zahar, 2008, 5.ed., 320p.

[JBR99] Jacobson, I.; Booch, G.; Rumbaugh, J. “The Unified Software Development Process”. Boston: Addison-Wesley, 1999, 512p.

[JVC05] Jamieson, D.; Vinsen, K.; Callender, G. “Agile Procurement: New Acquisition Approach to Agile Software Development”. In: 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005, pp. 266-273. DOI: 10.1109/EUROMICRO.2005.12

[KaR02] Kalermo, J.; Rissanen, J. “Agile software development in theory and practice”. Master’s thesis, University of Jyväskylä, Finland, 2002, 188p.

[Kar13] Karunakaran, S. “Impact of Cloud Adoption on Agile Software Development”, 2014. In: Software Engineering Frameworks for the Cloud Computing Paradigm, Springer London, 2010, pp. 213-234. DOI: 10.1007/978-1-4471-5031-2_10

[KiC07] Kitchenham, B; Charters, S. “Guidelines for performing Systematic Literature Reviews in Software Engineering”, Technical Report, Keele University and Durham University, 2007, 65p.

[Kni07] Kniberg, K. “Scrum e XP direto das Trincheiras: Como nós fazemos Scrum”. Tradução de Vinícius Assef, Cássio Marques, Álvaro Maia, José Inácio Serafini, Rafael Benevides, Rafael Recalde Caceres, Fernanda Stringassi de Oliveira, Renato Willi, Rodrigo Palhano, Daniel Wildt, Francisco M. Soares Nt., Eduardo Bobsin Machado, Juliana Berossa Steffen, Mauricio Vieira, Rodrigo Russo, Adam Victor Brandizzi, Alexandre Gomes, Bruno Pedroso, Marcos Machado, Victor Hugo Germano, Acyr José Tedeschi, Emanoel Tadeu da Silva Freitas, Gustavo Grillo, Israel Rodrigo Faria, Marcos Vinícius Guimarães, Pablo Nunes Alves e Taiza Sousa. C4Media, 2007, 126p. Capturado em: http://www.infoq.com/br/minibooks, Dezembro 2014.

[Kni09] Kniberg, K.; Skarin, M. “Kanban e Scrum obtendo o melhor de ambos”. Tradução de Juliana Berossa Steffen, Marcelo Andrade, Eduardo Bobsin, Rodrigo Russo, Daniel Wildt, Luciano Costa, Renato Willi, Marcos Vinícius Guimarães, Adam Brandizzi, André Pantalião, Cássio Marques, Ismael Stahelin, Rafael Fuchs, Gerson Dias, Ian Gallina, Rafael Dantas, Vitor Machel, Vinicius Assef, Bruno Pedroso, Cassiano Alves, Gustavo Grillo, Igo Coelho, Rafael Prikladnicki e Adriana Luppi. C4Media, 2009, 139p. Capturado em: http://www.infoq.com/br/minibooks, Dezembro 2014.

[Koc04] Koch, A. S. “Agile Software Development: Evaluating the Methods for Your Organization”. Boston/London: Artech House Publishers, 2004, 280p.

[Kro12] Krogstie, J. “Comparing Private and Public Sector on Information Systems Development and Maintenance Efficiency”. In: Electronic Government, Springer Berlin Heidelberg, 2012, pp. 222-233. DOI: 10.1007/978-3-642-33489-4_19

[Kru01] Kruchten, P. “The Rational Unified Process: An Introduction”. Boston: Addison- Wesley, 2001, 2nd Edition, 320p.

[KRW08] Kipp, A.; Riemer, K.; Wiemann, S. “IT Mega Projects - What they are and why they are special”. In: ECIS 2008 Proceedings, 2008, 12p.

[Lea13] LeanKit. “Kanban Roadmap: How to Get Started in 5 Steps”. Capturado em: http://goo.gl/cM13Qa, Novembro 2014.

[Lim07] Lima, R. C. “Princípios de Direito Administrativo”. Revista e reelaborada por Paulo Alberto Pasqualini. São Paulo: Malheiros, 2007, 589p.

[Mcd10] McDonald, C. “From art form to engineering discipline? A history of us military software development standards, 1974-1998”. *IEEE Annals of the History of Computing*, vol. 32-4, Out-Dez 2010, pp. 32-47. DOI: 10.1109/MAHC.2009.58

[Mei08] Meirelles, H. L. “Direito administrativo brasileiro”. São Paulo: Malheiros, 2008, 839p.

[Mid99] Middleton, P. “Managing information system development in bureaucracies”. *Information and Software Technology*, vol. 41-8, Jun 1999, pp. 473-482. DOI: 10.1016/S0950-5849(99)00015-4

[Min98] Minayo, M. C. S. “O desafio do conhecimento: pesquisa qualitativa em saúde”. São Paulo: Hucitec; Rio de Janeiro: ABRASCO, 1998.

[MiR14] Milani, F.; Roriz, H. “Capítulo 12: Planejamento”. In: Métodos Ágeis para Desenvolvimento de Software, Porto Alegre: Bookman, 2014, pp. 225-240.

[MIT13] MIT Technology Review. “Technology Is Wiping Out Companies Faster than Ever”. Capturado em: http://goo.gl/KLIII9, Novembro 2014.

[MPM09] Malheiros, V.; Paim, F. R.; Mendonça, M. “Continuous Process Improvement at a Large Software Organization”. *Software Process: Improvement and Practice*, vol. 14-2, Mar-Abr 2009, pp. 65-83. DOI: 10.1002/spip.376

[MSC+12] Melo, C. de O.; Santos, V. A.; Corbucci, H.; Katayama, E.; Goldman, A.; Kon, F. “Métodos Ágeis no Brasil: Estado da Práticas em Times e Organizações”, Relatório Técnico, Departamento de Ciência da Computação, IME-USP, 2012, 9p. Capturado em: http://goo.gl/w56t8y, Novembro 2014.

[MSK+13] Melo, C. O.; Santos, V.; Katayama, E.; Corbucci, H.; Prikladnicki, R.; Goldman, A.; Kon, F. “The evolution of agile software development in Brazil”. *Journal of the Brazilian Computer Society*, vol. 19-4, Nov 2013, pp. 523-552. DOI: 10.1007/s13173-013-0114-x.

[NAO06] UK NAO. “Delivering successful IT-enabled business change”, Report, National Audit Office, 2006, 53p. Capturado em: http://goo.gl/ck68TS, Dezembro 2014.

[NAO12] UK NAO. “Governance for Agile delivery”, National Audit Office, 2012, 35p. Capturado em: http://goo.gl/YHc3Et, Dezembro 2014.

[NBR09] NBR ISO/EIC 12207:2009. “Engenharia de Sistemas e Software - Processos de Ciclo de Vida de Software”. Rio de Janeiro: ABNT, 2009, 121p.

[NYT13] THE NEW YORK TIMES. “Why the Government Never Gets Tech Right: Getting to the Bottom of HealthCare.gov’s Flop”. Capturado em: http://goo.gl/NbjW7F, Novembro 2014.

[Ohn97] Ohno, T. “O Sistema Toyota de Produção: Além da Produção em Larga Escala”. Bookman, 1997, 149p.

[Pat02] Patton, M. Q. “Qualitative research & evaluation methods”. Thousand Oaks: Sage, 2002, 3.ed.

[Pau+91] Paulk, M. *et al*. “Capability Maturity Model for Software”. CMU/SEI-91-TR-24, DTIC Number ADA240603, Pittsburgh: Software Engineering Institute, Carnegie-Mellon University, 1991.

[Pau+93] Paulk, M. *et al*. “The Capability Maturity Model for Software V1.1”. CMU/SEI-93-TR-24, DTIC Number ADA263403, Pittsburgh: Software Engineering Institute, Carnegie-Mellon University, 1993.

[Pic11] Pichler, R. “Gestão de produtos com Scrum: implementando métodos ágeis na criação e desenvolvimento de produtos”. Tradução de Daniel Vieira. Rio de Janeiro: Elsevier, 2011, 152p.

[PiM06] Pires, J. C. de S.; Macêdo, K. B. “Cultura organizacional em organizações públicas no Brasil”. *Revista de Administração Pública*, vol. 40-1, Jan-Fev 2006, pp.81-104.

[Pon14] Pontes, R. E. da S. “Contratação do Desenvolvimento Ágil de Software na Administração Pública Federal: Riscos e Ações Mitigadoras”, Dissertação de Mestrado, Programa de Pós-graduação *Stricto Sensu* em Gestão do Conhecimento e da Tecnologia da Informação, UCB, 2014, 110p.

[PoP11] Poppendieck, M.; Poppendieck, T. “Implementando o desenvolvimento Lean de software: do conceito ao dinheiro”. Tradução de Luiz Cláudio Parzianello e Jean Felipe Patikowski Cheiran. Porto Alegre: Bookman, 2011, 279p.

[Pre11] Pressman, R. S. “Engenharia de Software: Uma abordagem profissional”. Tradução de Ariovaldo Griesi. Porto Alegre: AMGH, 2011, 7.ed., 780p.

[PrM14] Prikladnicki, R.; Magno, A. “Capítulo 3: O Framework do Scrum”. In: Métodos Ágeis para Desenvolvimento de Software, Porto Alegre: Bookman, 2014, pp. 22-36.

[PWM14] Prikladnicki, R.; Willi R.; Milani, F. (Org.). “Métodos Ágeis para Desenvolvimento de Software”. Porto Alegre: Bookman, 2014, 289p.

[Pys06] Pyster, A. “What Beyond CMMI Is Needed to Help Assure Program and Project Success?”. In: Unifying the Software Process Spectrum, Springer Berlin Heidelberg, 2006, pp. DOI: 10.1007/11608035_8

[Ric08] Rico, D. F. “What is the ROI of agile vs. traditional methods? An analysis of extreme programming, test-driven development, pair programming, and Scrum (using real options)”. 2008. Capturado em http://davidfrico.com/agile-benefits.xls, Novembro 2014.

[Roy70] Royce, W. W. “Managing the development of large software systems”. In: Proceedings of the IEEE WESCON, vol. 26, 1970, pp. 1-9. Reprinted in Proceedings of the International Conference on Software Engineering (ICSE) 1987, ACM Press, 1987, pp. 328-338.

[Rub13] Rubin, K. S. “Essential Scrum: a practical guide to the most popular agile process”. New Jersey: Addison-Wesley, 2013, 452p.

[SaC03] Sauer, C.; Cuthbertson, C. “The State of IT Project Management in the UK 2002-2003”, Templeton College, University of Oxford, 2003, 82p.

[Sah12] Sahota, M. “Transformação e Adoção Agile: Um Guia de Sobrevivência”. Tradução de Leonardo Campos, Rafael Buzon, Paulo Rebelo, Marcelo Costa, Daniel Wildt e Leonardo Galvão. InfoQ Brasil/C4Media, 2012, 54p. Capturado em: http://www.infoq.com/br/minibooks, Dezembro 2014.

[Sch01] Schwaber, K.; Beedle, M. “Agile Software Development with Scrum”. Prentice Hall, 2001, 158p.

[Sch04] Schwaber, K. “Agile Project Management with Scrum”. Redmond, Washington: Microsoft Press, 2004, 192p.

[Sch12] Schwaber, K.; Sutherland, J. “Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust”. Hoboken, New Jersey: John Wiley & Sons, 2012, 216p.

[Sch13] Schwaber, K.; Sutherland, J. “The Scrum Guide”. Tradução de Fábio Cruz. Scrum.org, 2013, 19p.

[Sch95] Schwaber, K. 1995. “Scrum Development Process”. In: Business Object Design and Implementation, Springer London, 1997, pp. 117-134. DOI: 10.1007/978-1-4471-0947-1_11

[Scr12] ScrumAlliance. “Scrum a description”. Tradução de Heitor Roriz Filho, Rafael Sabbagh, Marcos Garrido, Daniel Wandarti e Filipe Albero Pomar. ScrumAlliance, 2012. 12p. Capturado em: http://agileatlas.org/atlas/scrum, Dezembro 2014.

[Ser14a] SERPRO. “Serpro decide adotar método ágil”. Capturado em: http://goo.gl/TwEKaZ, Dezembro 2014.

[Ser14b] SERPRO. “Método ágil já demonstra resultados na empresa”. Capturado em: http://goo.gl/etcV1o, Dezembro 2014.

[SGI13] The Standish Group International. “Chaos Manifesto 2013: Think Big, Act Small”, The Standish Group, 2013, 48p.

[SGS05] Silva, C. R.; Gobbi, B. C.; Simão, A. A. “O uso da análise de conteúdo como uma ferramenta para a pesquisa qualitativa: Descrição e aplicação do método”. *Organizações Rurais Agroindustriais*, vol. 7-1, Jan-Abr 2005, pp. 70-81.

[Sho08] Shore, J.; Warden, S. “A arte do desenvolvimento ágil”. Rio de Janeiro: Alta Books, 2008, 420p.

[SIS12] SISP. “Processo de Software para o SISP: Versão 1.0”, Sistema de Administração dos Recursos de Tecnologia da Informação, SISP, 2012, 108p. Capturado em: http://www.sisp.gov.br, Dezembro 2014.

[Sis15] BR SISP. “Guia de Projeto de Sistemas com Práticas de Métodos Ágeis e Terceirização do Desenvolvimento para o SISP”. Secretaria de Logística e Tecnologia da Informação, 2015, 95p. Capturado em: http://goo.gl/IGxjk9, Março 2015.

[SJP07] Seixas Neto, A.; Jardine, J. G.; Pereira, N. R. “Informática e Agronegócio: a história da Embrapa Informática Agropecuária”. Campinas: Embrapa Informática Agropecuária, 2007. No prelo.

[Som11] Sommerville, I. “Engenharia de Software”. Tradução de Kalinka Oliveira e Ivan Bosnic. São Paulo: Pearson, 2011, 9.ed., 544p.

[Ste13] Stenbeck, J. G. “Agile Government Contracting - Expert Guidance for Department, Command and Agency Leaders, Contracting Officers, Procurement Professionals, Program and Project Managers, and Prime Contractors and Sub-Contractors”. GR8PM, 2013, 150p.

[Tar11] Tarozzi, M. “O que é a Grounded Theory? Metodologia de pesquisa e de teoria fundamentada nos dados”. Tradução de Carmem Lussi. Rio de Janeiro: Vozes, 2011, 189p.

[TCU13a] BR TCU. “Levantamento de Governança de TI 2012”. Tribunal de Contas da União, 2013, 56p. Capturado em: http://goo.gl/iabL4u, Dezembro 2014.

[TCU13b] BR TCU. “Levantamento de Auditoria. Conhecimento acerca da utilização de métodos ágeis nas contratações para desenvolvimento de software pela Administração Pública Federal. Arquivamento”. Tribunal de Contas da União, 2013, 42p. Capturado em: http://goo.gl/Q2uywd, Novembro 2014.

[TCU14] BR TCU. “Fiscalização de Orientação Centralizada (FOC). Governança de TI. Recomendações. Arquivamento”. Tribunal de Contas da União, 2014, 71p. Capturado em: http://goo.gl/IsBk3r, Novembro 2014.

[Tor08] Torres, T. Z. “Cartografia de aprendizagens individuais, coletivas e organizacionais na Embrapa Informática Agropecuária”, Tese de Doutorado em Educação, Universidade Federal de São Carlos, 2008, 264p.

[Uch13] Uchoa, A. P. “Contratação de Desenvolvimento Ágil de Software pelo Governo”, Dissertação de Mestrado, Programa de Pós-graduação em Engenharia de Sistemas e Computação, UFRJ, 2013, 270p.

[VaE14] VALOR ECONÔMICO. “Valor 1000”. Capturado em: http://goo.gl/Un4STS, Dezembro 2014.

[Val14] Vale, A. “Capítulo 8: Kanban”. In: Métodos Ágeis para Desenvolvimento de Software, Porto Alegre: Bookman, 2014, pp. 119-145.

[VaP14] Vacari, I.; Prikladnicki, R. “Desenvolvimento de Software na Administração Pública: Uma Revisão Sistemática da Literatura”, Relatório Técnico n° 082, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2014, 90p.

[Ver15] “VERSIONONE: 9th Annual State of Agile Development Survey”. VersionOne, 2014, 14p. Capturado em: http://goo.gl/gd7nyj, Abril 2015.

[Waz13] Wazlawick, R. S. “Engenharia de Software: Conceitos e Práticas”. Elsevier/Campus, 2013, 343p.

[Web+05] Weber, K. C. et al. “Modelo de referência e método de avaliação para melhoria de processo de software - versão 1.0 (MR-MPS.BR e MA-MPS.BR”. In: Anais do IV simpósio brasileiro de qualidade de software (SBQS 2005), 2005, pp. 347-360.

[Wer12] Wernham, B. “Agile Project Management for Government”. London: Maitland and Strong, 2012, 371p.

[Yin10] Yin, R. K. “Estudo de caso: planejamento e métodos”. Porto Alegre: Bookman, 2010, 4.ed., 248p.

[Zam+10] Zamboni, A. B. et al. “StArt Uma Ferramenta Computacional de Apoio à Revisão Sistemática”. In: Proc.: Congresso Brasileiro de Software (CBSoft’10), Salvador, Brazil. 2010.

APÊNDICE A

CONJUNTO FINAL DE ESTUDOS

Tabela 22 - Conjunto final de estudos.

ID	ORGANIZAÇÃO PÚBLICA	PAÍS	TÍTULO	AUTORIA	ANO	FONTE	MÉTODO ÁGIL	ORIGEM
(RSL01)	United States Strategic Command (USSTRATCOM)	Estados Unidos da América (EUA)	A case study: Introducing eXtreme programming in a US government system development project	A. Fruhling, P. McDonald, and C. Dunbar	2008	Hawaii International Conference on System Sciences - HICSS	Extreme Programming (XP)	Revisão Sistemática da Literatura (RSL)
(RSL02)	National Institutes of Health (NIH)	EUA	Staying agile in government software projects	B. Upender	2005	AGILE Conference	Scrum XP	RSL
(RSL03)	Banco Central do Brasil	Brasil	Adoção de métodos ágeis em uma Instituição Pública de grande porte - um estudo de caso	C. de O. Melo, and G.R.M. Ferreira	2010	Workshop Brasileiro de Métodos Ágeis - WBMA	Scrum XP	RSL
(RSL04)	Federal Bureau of Investigation (FBI)	EUA	The FBI gets agile	C. Fulgham, J. Johnson, M. Crandall, L. Jackson, and N. Burrows	2011	IT Professional	Scrum	RSL
(RSL05)	Swedish Association of Municipalities for Joint Development of eServices (Sambruk)	Suécia	Collaborative development of public information systems: A case study of “Sambruk” e-services development	C.-O. Olsson, and A. Öhrwall Rönnbäck	2010	eChallenges Conference	Scrum	RSL
(RSL06)	Rocky Flats Environmental Technology Site	EUA	Making agile development work in a government contracting environment-measuring velocity with earned value	G.B. Alleman, and M. Henderson	2003	Agile Development Conference - ADC	XP	RSL
(RSL07)	UK Regional Government Departament	Reino Unido	Agile development in a bureaucratic arena - A case study experience	H. Berger	2007	International Journal of Information Management	Não classificado (NC)	RSL

ID	Organização Pública	País	Título	Autoria	Ano	Fonte	Método Ágil	Origem
(RSL08)	Um departamento de Telecomunicações e TI do governo dos Emirados Árabes Unidos	Emirados Árabes Unidos	An industrial case study for Scrum adoption	H. Hajjdiab, A.S. Taleb, and J. Ali	2012	Journal of Software	Scrum	RSL
(RSL09)	United States Army	EUA	Army simulation program balances agile and traditional methods with success	J. Surdu, and D.J. Parsons	2006	CrossTalk	NC	RSL
(RSL10)	Uma organização pública da Bulgária	Bulgária	A Case Study on the Adoption of Measurable Agile Software Development Process	M. Iliev, I. Krasteva, and S. Ilieva	2009	International Conference on Software, Services & Semantic Technologies - S3T	NC	RSL
(RSL11)	Embrapa Informática Agropecuária	Brasil	Extreme Programming by example	M. Pedroso Jr, M.C. Visoli, and J.F.G. Antunes	2002	International Conference on Extreme Programming and Agile Processes in Software Engineering	XP	RSL
(RSL12)	Government Defense	EUA	Lessons learned using agile methods on large defense contracts	P.E. McMahon	2006	CrossTalk	NC	RSL
(RSL13)	Uma organização do governo dos Estados Unidos	EUA	Evolving to a “lighter” software process: a case study	R.J. Moore	2001	Annual NASA Goddard Software Engineering Workshop	XP	RSL
(RSL14)	Her Majesty’s Revenue and Customs (HMRC)	Reino Unido	Is Agile the Answer? The Case of UK Universal Credit	R. Michaelson	2013	International Working Conference on Transfer and Diffusion of IT - TDIT	NC	RSL
(RSL15)	Prefeitura de Kyoto	Japão	Agile software development under university-government cooperation	S. Kaneda	2006	World Multi-Conference on Systemics, Cybernetics and Informatics - WMSCI	NC	RSL
(RSL16)	NASA Langley Research Center	EUA	Exploring XP for scientific research	W.A. Wood, and W.L. Kleb	2003	IEEE Software	XP	RSL

ID	Organização Pública	País	Título	Autoria	Ano	Fonte	Método Ágil	Origem
(RSL17)	Israeli Air Force	Israel	Agile metrics at the Israeli Air Force	Y. Dubinsky, D. Talby, O. Hazzan, and A. Keren	2005	AGILE Conference	XP	RSL
(ROE01)	UK Ministry of Defense (MoD)	Reino Unido	Agile Project Management for Government: Chapter 1 - Case Study at the UK Ministry of Defense	Brian Wernham	2012	London - New York - Sydney: Maitland and Strong Publishing	Dynamic Systems Development Method (DSDM)	Revisão de Obra Especializada (ROE)
(ROE02)	US Department of Veterans Affairs	EUA	Agile Project Management for Government: Chapter 2 - Case Study at the US Department of Veterans Affairs	Brian Wernham	2012	London - New York - Sydney: Maitland and Strong Publishing	DSDM	ROE
(ROE03)	Federal Bureau of Investigation (FBI)	EUA	Agile Project Management for Government: Chapter 3 - Case Study at the FBI	Brian Wernham	2012	London - New York - Sydney: Maitland and Strong Publishing	Scrum	ROE
(ROE04)	Estado de Queensland	Austrália	Agile Project Management for Government: Chapter 4 - Case Study in Queensland, Australia	Brian Wernham	2012	London - New York - Sydney: Maitland and Strong Publishing	Scrum	ROE
(ROE05)	UK Met Office	Reino Unido	Agile Project Management for Government: Chapter 5 - Case Study at the UK Met Office	Brian Wernham	2012	London - New York - Sydney: Maitland and Strong Publishing	Scrum XP DSDM	ROE
(ECa01)	Embrapa Informática Agropecuária	Brasil	Estudo de Caso 1: Sistema de gerenciamento de dados experimentais	NA	2014	NA	Scrum	Estudo de Caso (ECa)
(ECa02)	Embrapa Informática Agropecuária	Brasil	Estudo de Caso 2: Sistema de gerenciamento de acervos documentais e digitais	NA	2014	NA	XP	ECa

ID	ORGANIZAÇÃO PÚBLICA	PAÍS	TÍTULO	AUTORIA	ANO	FONTE	MÉTODO ÁGIL	ORIGEM
(ECa03)	Companhia de Processamento de Dados do Rio Grande do Sul	Brasil	Estudo de Caso 3: Sistema de gerenciamento de empresas, pessoas vinculadas ao DETRAN/RS	NA	2014	NA	Método de Desenvolvimento PROCERGS (MDP)	ECa
(ECa04)	Companhia de Processamento de Dados do Rio Grande do Sul	Brasil	Estudo de Caso 4: sistema para a Divisão de Remoção e Depósito do DETRAN/RS	MA	2014	NA	MDP	ECa

APÊNDICE B

PROTOCOLO DE ESTUDO DE CASO

Métodos Ágeis na Administração Pública brasileira: Um Estudo Empírico

Objetivo

Entender e ampliar o que se sabe sobre o uso de métodos ágeis na AP brasileira, incluindo suas vantagens e desvantagens, bem como, os benefícios alcançados, desafios enfrentados, lições aprendidas e recomendações para seu uso.

Característica-chave do método de estudo de caso

Este método de pesquisa é bem adequado para o estudo de organizações que estão envolvidas em processos de mudanças (implantação de novos métodos, sistemas, processos ou ferramentas) onde seus participantes são afetados. Uma vez que o principal método de coleta de dados em estudo de caso é a entrevista, um protocolo para entrevista semiestruturada, com questões abertas e fechadas, foi estabelecido. Na sequência os dados coletados nas entrevistas serão complementados com observações, bem como, com análise de documentos existentes. A técnica de triangulação será usada para garantir a confiabilidade dos dados coletados e dos resultados alcançados, onde as respostas dos entrevistados serão comparadas com os dados observados.

Unidades de análise

Projetos de desenvolvimento de software na AP brasileira em andamento ou concluídos.

Organização do protocolo

O protocolo está organizado da seguinte maneira:

1. Procedimentos

a. Levantamento das questões e estruturação do guia para a entrevista	
Participantes	Isaque Vacari
Data	Julho de 2014
Local	FACIN PUCRS - Faculdade de Informática da PUCRS

b. Revisão do guia para a entrevista	
Participantes	Prof. Dr. Rafael Prikladnicki
Data	Agosto de 2014
Local	TECNOPUC PUCRS - Parque Científico e Tecnológico da PUCRS

c. Autorização das empresas participantes	
Participantes	Comitê Técnico Interno (CTI) da Embrapa Informática Agropecuária
Data	Julho de 2014
Local	Campinas/SP

d. Validação de face e conteúdo	
Participantes	Prof. Dr. Duncan Dubugras Alcoba Ruiz
Data	Setembro de 2014
Local	Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

e. Pré-teste	
Participantes	Leandro Henrique Mendonça de Oliveira
Data	Setembro de 2014
Local	Embrapa Informática Agropecuária (Campinas/SP)

f. Aplicação das entrevistas - Dimensão Organizacional	
Embrapa Informática Agropecuária	**Participante 1** Assessor da Chefia de P&D (Núcleo de Garantia da Qualidade) 18 de Setembro de 2014
Companhia de Processamento de Dados do Rio Grande do Sul	**Participante 2** Divisão de Desenvolvimento de Soluções (DDS) 14 de Outubro de 2014

g. Aplicação das entrevistas - Dimensão do Projeto	
Embrapa Informática Agropecuária	**Participante 1 (Projeto 1)** Desenvolvedor de Software 15 de Setembro de 2014 **Participante 2 (Projeto 1)** Gerente de Projeto 15 de Setembro de 2014 **Participante 3 (Projeto 1)** Desenvolvedor de Software 17 de Setembro de 2014 **Participante 4 (Projeto 2)** Gerente de Projeto 23 de Setembro de 2014 **Participante 5 (Projeto 2)** Desenvolvedor de Software 23 de Setembro de 2014
Companhia de Processamento de Dados do Rio Grande do Sul	**Participante 6 (Projeto 3)** ScrumMaster e Desenvolvedor de Software 14 de Outubro de 2014 **Participante 7 (Projeto 3)** Desenvolvedor de Software 14 de Outubro de 2014 **Participante 8 (Projeto 3)** Desenvolvedor de Software 14 de Outubro de 2014 **Participante 9 (Projeto 3)** Analista de Testes 14 de Outubro de 2014 **Participante 10 (Projeto 3)** Analista de Sistemas / Product Owner 14 de Outubro de 2014 **Participante 11 (Projeto 3)** Analista de Sistemas / Product Owner 15 de Outubro de 2014

g. Aplicação das entrevistas - Dimensão do Projeto	
Companhia de Processamento de Dados do Rio Grande do Sul	**Participante 12 (Projeto 4)** ScrumMaster e Desenvolvedor de Software 15 de Outubro de 2014 **Participante 13 (Projeto 4)** Desenvolvedor de Software 15 de Outubro de 2014 **Participante 14 (Projeto 4)** Líder Técnico e Desenvolvedor de Software 15 de Outubro de 2014 **Participante 15 (Projeto 4)** Analista de Sistemas 15 de Outubro de 2014 **Participante 16 (Projeto 4)** Analista de Testes 15 de Outubro de 2014

2. Escolha das pessoas entrevistadas

Respondentes:

a. Chefes de desenvolvimento de software

b. Servidores públicos membros de projetos de desenvolvimento de software na AP

3. Outros recursos utilizados

a. Recursos tecnológicos:

   i. Microsoft® Word e Microsoft® Excel

b. Recursos financeiros:

   i. Bolsa Embrapa do Programa de Pós-Graduação

c. Recursos materiais:

   i. Um gravador para gravar as entrevistas

   ii. Papel e caneta para anotações de campo

4. Informações da organização da pesquisa

	Organização da Pesquisa
Natureza	Aplicada
Estratégia	Qualitativa
Tipo de Pesquisa	Exploratória
Área do Conhecimento	Ciências Exatas
Tipo de Desenho de Pesquisa	Não experimental (transversal)
Método de Pesquisa	Estudo de Caso (interpretativa)
Método de Coleta de Dados	Entrevista Observação neutra
Tipo de Dados Analisado	Qualitativo
Método de Análise de Dados	Análise de conteúdo das entrevistas e observações
Registro dos Dados Coletados	Anotações de campo Gravações (caso autorizado)
Local de Desenvolvimento	Campo

5. Protocolo para entrevistas

O protocolo para entrevista é composto por uma lista de perguntas abertas e fechadas organizadas nas seguintes dimensões:

QUESTÕES (A)	
DADOS DEMOGRÁFICOS	Nome:
	Curso (nível mais alto):
	Instituição: Concluído em: _____.
	Tempo de experiência profissional na área de Informática: ___ anos.
	Tempo de experiência profissional com desenvolvimento de software: ___ anos.
	Tempo de experiência profissional com métodos ágeis: ___ anos.
	Tempo de experiência profissional no setor público: ___ anos.
	Tempo de empresa: ___ anos.
	Vínculo empregatício:
	Área/departamento/setor:
	Função atual:

a. Dimensão Organizacional

	VARIÁVEIS	QUESTÕES (B)	AUTORES

ORGANIZACIONAL	Caracterização da organização e transformação cultural.	Dados gerais:
		• Nome da instituição
		• Sítio *web* da instituição
		• Negócio da instituição
		• Missão da instituição
		• Principais clientes da instituição
		• Número total de empregados
		  ○ Menor que 100
		  ○ Entre 101 e 300
		  ○ Entre 301 e 500
		  ○ Entre 501 e 1000
		  ○ Maior que 1000
		• Número total de empregados lotados na área de TI
		  ○ Menor que 100
		  ○ Entre 101 e 300
		  ○ Entre 301 e 500
		  ○ Entre 501 e 1000
		  ○ Maior que 1000
			(RSL01)
			(RSL02)
			(RSL03)
			(RSL07)
			(RSL08)
			(RSL14)
			(RSL15)
			(RSL17)
			(ROE03)

		Quais são principais desafios do desenvolvimento de software na instituição?	
		Em que tipo melhor se enquadra o processo de desenvolvimento de software da instituição? • **Ad hoc** (não há processo e nem conceito de qualidade do processo). • **Inicial** (não há processo nem seu controle, mas já há conceitos de qualidade de processo em implantação). • **Livre** (há vários processos informais executados por diferentes equipes de desenvolvimento). • **Gerenciado** (há um processo informal repetido várias vezes e que implementa conceitos de qualidade de processo. • **Definido** (há um processo formal - aprovado e publicado - e obrigatório). • **Mensurado** (o processo é controlado por meio de mensurações e há metas de processo a cumprir). • **Em otimização** (o processo é periodicamente revisado e melhorado com base nas suas mensurações).	

		Conhece o Manifesto Ágil? Em que profundidade?	
		Fale-me das virtudes e dificuldades do alinhamento da cultura da instituição aos valores do manifesto ágil?	
		Que mudanças aconteceram ou são necessárias na cultura da instituição em favor de métodos ágeis? Quais são as barreiras?	
		Que tipo de apoio a instituição têm oferecido ou pode oferecer para que os métodos ágeis consigam alcançar seu pleno êxito?	

b. Dimensão do Projeto:

	VARIÁVEIS	QUESTÕES (C)	AUTORES
CARACTERIZAÇÃO DO PROJETO	Caracterização do projeto.	Dados gerais: • Nome do projeto • E-mail para contato • Sítio web público • Licença do produto de software • Tipo de software • Breve descrição	[KRW08] (RSL04) (RSL05) (RSL09) (RSL15)
		Características do projeto: • Orçamento: ○ Menor que R$100.000,00 ○ Entre R$100.000,01 e R$500.000,00 ○ Entre R$500,000,01 e R$1.000.000,00 ○ Entre R$1.000.000,01 e R$5.000.000,00 ○ Maior que R$5.000.000,00 • Duração do projeto. • Número de pessoas/departamentos/órgãos/empresas envolvido(a)s. • Tipo de desenvolvimento: ○ Interno ○ Governo e Indústria de Software ○ Governo e Academia ○ Governo e Governo ○ Múltiplas formas de desenvolvimento	

	VARIÁVEIS	QUESTÕES (D)	AUTORES
EQUIPE	Caracterização da equipe e cooperação mútua.	Características da equipe: • Tamanho • Distância entre os desenvolvedores • Distância entre a equipe de desenvolvimento e o dono do produto • Distância entre o dono do produto e os *stakeholders*	(RSL03) (RSL11) (RSL16) (RSL17) (ROE01) (ROE03) (ROE04)
		Como é a relação entre os desenvolvedores? Como e com que frequência eles interagem? Há quanto tempo eles trabalham juntos? Que problemas têm acontecido?	

	VARIÁVEIS	QUESTÕES (E)	AUTORES
RELACIONAMENTO COM O CLINTE	Cooperação mútua e o envolvimento do dono do produto e dos *stakeholders*.	Como é a relação entre a equipe de desenvolvimento e o dono do produto? Há quanto tempo eles trabalham juntos? Como e com que frequência eles interagem? Que problemas têm acontecido?	(RSL03) (RSL11) (RSL16) (RSL17) (ROE01) (ROE03) (ROE04)
		Como é a relação entre o dono do produto e os *stakeholders*? Como e com que frequência eles interagem? Há quanto tempo eles trabalham juntos? Que problemas têm acontecido?	

	VARIÁVEIS	QUESTÕES (F)	AUTORES
RELACIONAMENTO COM O NEGÓCIO	Alinhamento entre TI e negócio, gerenciamento de mudanças de prioridades, informações de progresso do projeto, velocidade de *time-to-market* e satisfação do cliente.	Como as necessidades e os requisitos de negócio chegam e são organizados para o desenvolvimento?	(RSL01) (RSL02) (RSL04) (RSL05) (RSL07) (RSL09) (RSL10) (RSL17) (ROE01) (ROE02) (ROE03) (ROE05)
		Como a equipe de desenvolvimento tem absorvido as mudanças de prioridades de negócio?	
		Como e com que frequência o progresso do projeto é informado para o dono do produto e os *stakeholders*?	
		Como e com que frequência novas versões do produto de software são entregues? Por que com essa frequência?	
		Pode descrever-me as principais satisfações e insatisfações do(s) cliente(s) com o produto de software? Por que isso acontece?	

	VARIÁVEIS	QUESTÕES (G)	AUTORES
PROCESSOS E PRÁTICAS	Nível de experiência, resultados e lições aprendidas com o uso dos métodos/processos adotados.	Que métodos/processos têm sido adotados? Por quê?	(RSL01) (RSL02) (RSL03) (RSL04) (RSL06) (RSL08) (RSL09) (RSL11) (RSL12) (RSL16) (ROE01)
		Que mudanças, positivas e negativas, aconteceram desde eles foram adotados? Em caso positivo, como os resultados podem ser melhorados? Em caso negativo, quais são os desafios?	
		Pode descrever-me quais lições mais importantes você aprendeu com respeito a eles? Como eles podem ser melhor utilizados? Em que situações eles não são recomendados?	
		Que conhecimento técnico e experiência a equipe possui em relação a eles? Em caso de pouco ou nenhum conhecimento e/ou experiência, o que tem sido feito?	
		Que práticas têm sido eficientes e ineficientes? Que problemas elas têm conseguido e não têm conseguido resolver? Em caso positivo, como elas podem ser melhor executadas? Em caso negativo, quais são os desafios?	

	QUESTÕES (H)
Opinião	Fale-me de seus pensamentos e de suas percepções sobre métodos ágeis? O que você mais aprecia e não aprecia neles?
	De seu ponto de vista, como o desenvolvimento de software poderia ser melhor executado na sua instituição? Como chegou a essa conclusão?

6. Roteiro das entrevistas

Dimensão Organizacional

- **Dados demográficos**

Nome:
Curso (nível mais alto):
Instituição: Concluído em: _____.
Tempo de experiência profissional na área de Informática: ___ anos.
Tempo de experiência profissional com desenvolvimento de software: ___ anos.
Tempo de experiência profissional com métodos ágeis: ___ anos.
Tempo de experiência profissional no setor público: ___ anos.
Tempo de empresa: ___ anos.
Vínculo empregatício:
Área/departamento/setor:
Função atual:

- **Aspecto: Organizacional**

1. Dados gerais:
    - Nome da instituição
    - Sítio web da instituição
    - Negócio da instituição
    - Missão da instituição
    - Principais clientes da instituição
    - Número total de empregados
        - Menor que 100
        - Entre 101 e 300
        - Entre 301 e 500
        - Entre 501 e 1000
        - Maior que 1000
    - Número total de empregados lotados na área de TI
        - Menor que 100
        - Entre 101 e 300
        - Entre 301 e 500
        - Entre 501 e 1000
        - Maior que 1000
2. Quais são principais desafios do desenvolvimento de software na instituição?
3. Em que tipo melhor se enquadra o processo de desenvolvimento de software da instituição?
    a. **Ad hoc** (não há processo e nem conceito de qualidade do processo).
    b. Inicial (não há processo nem seu controle, mas já há conceitos de qualidade de processo em implantação).
    c. **Livre** (há vários processos informais executados por diferentes equipes de desenvolvimento).
    d. **Gerenciado** (há um processo informal repetido várias vezes e que implementa conceitos de qualidade de processo.
    e. **Definido** (há um processo formal - aprovado e publicado - e obrigatório).
    f. **Mensurado** (o processo é controlado por meio de mensurações e há metas de processo a cumprir).
    g. **Em otimização** (o processo é periodicamente revisado e melhorado com base nas suas mensurações).
4. Conhece o Manifesto Ágil? Em que profundidade?
5. Fale-me das virtudes e dificuldades do alinhamento da cultura da instituição aos valores do manifesto ágil?
6. Que mudanças aconteceram ou são necessárias na cultura da instituição em favor de métodos ágeis? Quais são as barreiras?
7. Que tipo de apoio a instituição têm oferecido ou pode oferecer para que os métodos ágeis consigam alcançar seu pleno êxito?

Dimensão do Projeto

- **Dados demográficos**

Nome:
Curso (nível mais alto):
Instituição: Concluído em: _____.
Tempo de experiência profissional na área de Informática: ___ anos.
Tempo de experiência profissional com desenvolvimento de software: ___ anos.
Tempo de experiência profissional com métodos ágeis: ___ anos.
Tempo de experiência no setor público: ___ anos.
Área/departamento/setor:
Vínculo empregatício:
Tempo de empresa: ___ anos.
Função atual:

- **Aspecto: Caracterização do projeto**

8. Dados gerais:
    - Nome do projeto
    - E-mail para contato
    - Sítio web público
    - Licença do produto de software
    - Tipo de software
    - Breve descrição
    - Orçamento
        - Menor que R$100.000,00
        - Entre R$100.000,01 e R$500.000,00
        - Entre R$500,000,01 e R$1.000.000,00
        - Entre R$1.000.000,01 e R$5.000.000,00
        - Maior que R$5.000.000,00
    - Duração do projeto
    - Número de pessoas/departamentos/órgãos/empresas envolvido(a)s
    - Tipo de desenvolvimento
        - Interno
        - Governo e Indústria de Software
        - Governo e Academia
        - Governo e Governo
        - Múltiplas formas de desenvolvimento

- **Aspecto: Equipe**

9. Características da equipe
    - Tamanho
    - Distância entre os desenvolvedores
    - Distância entre a equipe de desenvolvimento e o dono do produto
    - Distância entre o dono do produto e os stakeholders

10. Como é a relação entre os desenvolvedores? Como e com que frequência eles interagem? Há quanto tempo eles trabalham juntos? Que problemas têm acontecido?

- **Aspecto: Relacionamento com o cliente**

11. Como é a relação entre a equipe de desenvolvimento e o dono do produto? Há quanto tempo eles trabalham juntos? Como e com que frequência eles interagem? Que problemas têm acontecido?
12. Como é a relação entre o dono do produto e os stakeholders? Como e com que frequência eles interagem? Há quanto tempo eles trabalham juntos? Que problemas têm acontecido?

- **Aspecto: Relacionamento com o negócio**

13. Como as necessidades e os requisitos de negócio chegam e são organizados para o desenvolvimento?
14. Como a equipe de desenvolvimento tem absorvido as mudanças de prioridades de negócio?
15. Como e com que frequência o progresso do projeto é informado para o dono do produto e os stakeholders?
16. Como e com que frequência novas versões do produto de software são entregues? Por que com essa frequência?
17. Pode descrever-me as principais satisfações e insatisfações do(s) cliente(s) com o produto de software? Por que isso acontece?

- **Aspecto: Processos e práticas**

  18. Que métodos/processos têm sido adotados? Por quê?
  19. Que mudanças, positivas e negativas, aconteceram desde eles foram adotados? Em caso positivo, como os resultados podem ser melhorados? Em caso negativo, quais são os desafios?
  20. Pode descrever-me quais lições mais importantes você aprendeu com respeito a eles? Como eles podem ser melhor utilizados? Em que situações eles não são recomendados?
  21. Que conhecimento técnico e experiência a equipe possui em relação a eles? Em caso de pouco ou nenhum conhecimento e/ou experiência, o que tem sido feito?
  22. Que práticas têm sido eficientes e ineficientes? Que problemas elas têm conseguido e não têm conseguido resolver? Em caso positivo, como elas podem ser melhor executadas? Em caso negativo, quais são os desafios?

- **Aspecto: Opinião**

  23. Fale-me de seus pensamentos e de suas percepções sobre métodos ágeis? O que você mais aprecia e não aprecia neles?

De seu ponto de vista, como o desenvolvimento de software poderia ser melhor executado na sua instituição? Como chegou a essa conclusão?

APÊNDICE C

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Métodos Ágeis na Administração Pública brasileira: Um Estudo Empírico

Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Faculdade de Informática (FACIN)
Avenida Ipiranga, 6681 - Prédio 32 - 90619-900 - Porto Alegre/RS
Telefone: (51) 3320-3558

Termo de Consentimento Livre e Esclarecido

A PUCRS, através do **grupo de pesquisa de Metodologias Ágeis e Desenvolvimento Distribuído de Software (Munddos)** da Faculdade de Informática**,** agradece a sua atenção e a inestimável contribuição que prestarão para o auxílio do avanço da pesquisa na área de Engenharia de Software.

O objetivo desta pesquisa é investigar questões relacionadas a adoção da Métodos Ágeis na Administração Pública brasileira. Para isto, os participantes são convidados a participar com seus relatos de experiência e opiniões sobre essa temática.

Lembramos que o objetivo deste estudo **não é avaliar o participante**, **mas sim** avaliar como o desenvolvimento de software no governo acontece(u) na prática. O uso que se faz dos registros efetuados durante a entrevista é **estritamente** limitado a atividades de pesquisa e desenvolvimento, garantindo-se para tanto que:

1. O anonimato dos participantes será preservado em todo e qualquer documento divulgado em foros científicos (tais como conferências, periódicos, livros e assemelhados) ou pedagógicos (tais como apostilas de cursos, *slides* de apresentações, e assemelhados).
2. Todo participante poderá ter acesso a cópias destes documentos após a publicação dos mesmos.
3. Todo participante que se sentir constrangido ou incomodado durante a realização da atividade pode interromper a sua participação e estará fazendo um favor à equipe se registrar por escrito as razões ou sensações que o levaram a esta atitude. A equipe fica obrigada a descartar os dados do participante para fins da avaliação a que se destinaria.
4. Todo participante tem direito de expressar por escrito, na data de realização da atividade, qualquer restrição ou condição adicional que lhe pareça aplicar-se aos itens acima enumerados (1, 2 e 3). A equipe se compromete a observá-las com rigor e entende que, na ausência de tal manifestação, o participante concorda que rejam o comportamento ético da equipe somente as condições impressas no presente documento.
5. A equipe tem direito de utilizar os dados das atividades, mantidas as condições acima mencionadas, para quaisquer fins acadêmicos, pedagógicos e/ou de desenvolvimento contemplados por seus membros.

Declaro que estou de pleno acordo com os termos acima.

_____
Assinatura do participante

_____
Isaque Vacari
Pesquisador Responsável

Nome do Participante:
Pesquisadores Responsáveis: Isaque Vacari e Prof. Dr. Rafael Prikladnicki
Contato: isaque.vacari@acad.pucrs.br e rafael.prikladnicki@pucrs.br
Faculdade de Informática - PUCRS

APÊNDICE D

PALESTRAS MINISTRADAS SOBRE MÉTODOS ÁGEIS NO GOVERNO EM 2014

Tabela 23 - Algumas palestras ministradas sobre métodos ágeis no governo em 2014.

	PALESTRA
AgileTrends	**Metodologias Ágeis no Governo - Mitos e Verdades** Rafael Prikladnicki e Isaque Vacari
	Licitação, Copa do Mundo, 60 Desenvolvedores e Agilidade Rafael Nascimento
Regional Scrum Gathering® Rio	**Atacando os problemas certos com Kanban, Métricas e Visualização** Amanda Varella
	Scrum na administração pública é possível? Herbert Parente
	Uma retrospectiva sobre a utilização do Scrum em uma empresa pública: o que funcionou e o que precisa melhorar Luiz Carlos Júnior
	Metodologias Ágeis no Governo - Mitos e Verdades Rafael Prikladnicki (PUCRS) e Isaque Vacari (Embrapa)
	Agile quebrando mais paradigmas: a inclusão de um desenvolvedor cego em um time SCRUM Sonia Moreira Goldzweig e Avelino Ferreira Gomes
AgileBrazil	**Governo e Agilidade na Prática** James Brum
	Metodologias Ágeis na Administração Pública: Uma Revisão Sistemática da Literatura Rafael Prikladnicki e Isaque Vacari
	Aferição da Qualidade do Código-Fonte em uma autarquia da Administração Pública Federal Hilmer Rodrigues Neri e Guilherme Baufaker Rêgo
	Agilidade no Judiciário - Um relato de experiência de Agile Coaching no TJ-RS Guilherme Lacerda e Felipe Pinheiro de Souza
GUMA Agile Day 2014	**Governo e Agilidade na Prática** James Brum
	Metodologias Ágeis na Administração Pública: Uma Revisão Sistemática da Literatura Rafael Prikladnicki e Isaque Vacari
	Agilidade no Judiciário - Um relato de experiência de Agile Coaching no TJ-RS Guilherme Lacerda e Felipe Pinheiro de Souza

Este trabalho foi custeado pela Empresa Brasileira de Pesquisa Agropecuária, por meio da Resolução Normativa 037.009.004.004 do Programa de Educação Corporativa - Pós-Graduação Stricto Sensu.

Este trabalho foi editado de acordo com o Modelo de Teses, Dissertações e Monografias para o Programa de Pós-Graduação em Ciência da Computação (PPGCC), sendo derivado de uma compilação das regras do documento elaborado pela Biblioteca Central Irmão José Otão da PUCRS, o qual foi obtido de uma simplificação da NBR 14724 (trabalhos acadêmicos), usando para a bibliografia um formato derivado daquele empregado pela IEEE.