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

DESENVOLVIMENTO DE SOFTWARE NA ADMINISTRAÇÃO PÚBLICA: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Isaque Vacari e Prof. Dr. Rafael Prikladnicki

Relatório Técnico: n° 082.
Data: Dezembro, 2014.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

AP	Administração Pública
AUP	*Agile Unified Process*
CASE	*Computer-Aided Software Engineering*
CMM	*Capability Maturity Model* (CMM)
DoD	*United States Department of Defense*
ES	Engenharia de Software
EUA	Estados Unidos da América
GP	Gerenciamento de Projetos
IEEE	*Institute of Electrical and Electronics Engineers*
IV&V	*Independent Verification and Validation*
MA	Métodos ágeis
MSF	*Microsoft Solutions Framework*
MPS.BR	Melhoria de Processos do Software Brasileiro
OSS	*Open-source software*
PDCA	*Plan-Do-Check-Act*
PDS	Processo de Desenvolvimento de Software
PMBOK	*Project Management Body of Knowledge*
PRODEPA	Empresa de Processamento de Dados do Estado do Pará
RAD	*Rapid Application Development*
SI	Sistema de Informação
SWEBOK	*Software Engineering Body of Knowledge*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
USDP	*Unified Software Development Process*
TSP	*Team Software Process*
V&V	Verificação e validação de software
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

No âmbito do governo, as administrações públicas (APs) vem passando por um período de grandes mudanças para atender os compromissos atuais e emergentes da sociedade, bem como, para melhorar a qualidade dos serviços públicos prestados e agregar valor as atividades da AP, sendo que a maioria das transformações tem sido sustentada por um investimento pesado em tecnologia da informação e comunicação [Bal10]. Esta realidade implica diretamente no aumento da demanda por produtos de software e indiretamente implica na busca por melhores abordagens e processos para desenvolvê-los ou adquiri-los [Bas08]. Se durante muitos anos o desenvolvimento de software foi guiado por processos prescritivos, que tinham por finalidade colocar ordem no caos, proporcionando um roteiro razoavelmente eficaz para as equipes de projetos, nos dias atuais observa-se a introdução cada vez maior de processos adaptativos de desenvolvimento de software, focados no produto e nas pessoas que o desenvolvem e recomendados para ambientes onde os requisitos são voláteis [Pre11]. Modelos, processos e métodos de Engenharia de Software (ES) se encaixam neste contexto e tem gradativamente despertado o interesse da AP brasileira, principalmente se já foram experimentados e avaliados pelo setor privado. Neste sentido, o objetivo desta monografia é apresentar um estudo sobre o desenvolvimento de software na AP, apresentando evidências sobre a utilização de modelos, processos e métodos de ES no setor público, incluindo os problemas encontrados, desafios enfrentados, lições aprendidas, benefícios alcançados, além de recomendações para seu uso, formando assim a base teórica sobre o tema. Para isso, foi planejado e executado um estudo secundário do tipo Revisão Sistemática da Literatura (RSL). Cabe ressaltar, que até o momento, nenhuma RSL sobre desenvolvimento de software na AP foi publicada ou se encontra disponível para pesquisas acadêmicas. Isto significa que profissionais da indústria de software, servidores públicos e pesquisadores precisam recorrer de maneira exploratória a livros e artigos, a fim de obter uma visão geral sobre o tema. Espera-se que este estudo seja útil para todos os grupos, deixando mais claro que desenvolvimento de software na AP tem sido suportados por estudos científicos. Este estudo apresenta no capítulo 2 o método de pesquisa utilizado, o detalhamento e a execução da revisão sistemática, enquanto que o capítulo 3 apresenta os resultados encontrados, bem como, uma análise quantitativa e qualitativa dos mesmos. Finalmente, o capítulo 4 apresenta as limitações do estudo, e as considerações finais no capítulo 5.

2 MÉTODO DE PESQUISA

O objetivo deste estudo é identificar, reunir, organizar e sintetizar o maior número de evidências científicas sobre o desenvolvimento de software na Administração Pública (AP), o quanto possível, que permita caracterizar a área de estudo e formar uma base teórica sobre o tema. O resultado dessa caracterização pode auxiliar na identificação de novas oportunidades de pesquisa na área, além de direcionar os esforços em estratégias mais adequadas na área de estudo. Para alcançar este objetivo planejou-se e executou-se um estudo secundário do tipo Revisão Sistemática da Literatura (RSL). De acordo com Kitchenham & Charters [KiC07], uma RSL é um meio de identificar, avaliar e interpretar todo o estudo relevante sobre uma questão de pesquisa específica, área temática ou fenômeno de interesse. Estudos individuais que contribuem para uma RSL são chamados de estudos primários; uma RSL é uma forma de estudo secundário.

2.1 Planejamento da Revisão Sistemática

Para descrever e organizar o plano de estudo planejou-se e estabeleceu-se um protocolo de pesquisa, de acordo com as recomendações de Kitchenham & Charters [KiC07]. O protocolo abrange a questão de pesquisa que essa RSL se propõe a abordar, a estrutura da pergunta e sua estratégia de busca, assim como, as bases de dados utilizadas para recuperação de evidências cientificas e os critérios de inclusão e exclusão de estudos adotados.

2.1.1 Questão de Pesquisa

Essa RSL foi iniciada com a seguinte questão de pesquisa:

O que se sabe sobre o desenvolvimento de software na Administração Pública (AP), incluindo a utilização de modelos, processos e métodos de Engenharia de Software (ES)?

2.1.2 Estrutura da Pergunta

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

- **População:** Administração Pública (AP);
- **Intervenção:** Artigos científicos que abordam o desenvolvimento de software na AP;
- **Saídas:** Os modelos, os processos e os métodos de ES utilizados na AP, assim como, os problemas encontrados, desafios enfrentados, lições aprendidas, benefícios alcançados, além de recomendações para o seu uso.

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 Administração Pública e aqueles relacionados com a dimensão de Desenvolvimento de Software. A Tabela 1 apresenta as palavras-chave utilizadas em cada categoria. Além disso, as palavras-chave foram utilizadas para recuperar automaticamente estudos escritos em inglês, enquanto os trabalhos escritos em português foram recuperados por meio de uma pesquisa manual realizada em três fontes de informação científica no Brasil, a saber na seção 2.4. 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 na AP. Assim, 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)

Tabela 1 - Palavras-chave utilizadas por categoria.

Referência	Categoria	Palavras-chave
A	**Administração 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)

2.1.5 Bases de Dados

A abordagem inicial definiu que as seguintes bases de dados deveriam ser consultadas:

- ACM Digital Library (http://dl.acm.org/);
- Bielefeld Academic Search Engine (http://www.base-search.net/);
- ScienceDirect (http://www.sciencedirect.com/);
- Engineering Village (http://www.engineeringvillage.com/);
- IEEEXplore (http://ieeexplore.ieee.org/);
- Scopus (http://www.scopus.com/);
- Springer (http://www.springer.com/);
- Web of Knowledge (http://apps.webofknowledge.com/);
- Wiley Online Library – Wiley (http://onlinelibrary.wiley.com/).

Além das bases de dados citadas anteriormente, outras fontes de evidências científicas foram incluídas no estudo para pesquisa manual:

- Bases de Dados da Pesquisa Agropecuária – BDPA (http://www.embrapa.br/bdpa/);
- Biblioteca Digital Brasileira de Computação - BDBComp (http://www.lbd.dcc.ufmg.br/bdbcomp/);
- Workshop Brasileiro de Métodos Ágeis - WBMA (http://www.agilebrazil.com/).

2.1.6 Critérios de Seleção e Exclusão de Estudos

Os critérios de seleção de estudos destinaram-se a identificar os estudos primários que forneceram evidência direta sobre a questão de pesquisa. Para incluir um estudo na análise, os seguintes critérios foram adotados:

- O estudo deveria relatar a experiência do desenvolvimento de software na AP;
- O estudo deveria ter sido escrito em inglês ou português;
- O estudo deveria estar disponível em texto completo[1] para leitura e extração dos dados.

[1] O texto completo dos estudos foi obtido por meio dos serviços das bibliotecas da Pontifícia Universidade Católica do Rio Grande do Sul, Empresa Brasileira de Pesquisa Agropecuária e Universidade Estadual de Campinas.

Além disso, estudos sobre desenvolvimento de software realizados em universidades públicas sem a participação de outras empresas, órgãos ou agências de governo não fizeram parte do escopo desta RSL. Igualmente, foram excluídos estudos relacionados com:

- Adoção e implantação de Governança de TI, Governo Eletrônico, TI Verde, Segurança da Informação e Infraestrutura de TI na AP;
- Adoção de padrões abertos e implantação de dados abertos na AP;
- Adoção e implantação de *Enterprise Resource Planning* (ERP), *Data Warehouse*, *Data Mining*, *Business Intelligence* (BI) e *Geographic Information System* (GIS) e software livre na AP;
- Avaliação de usabilidade e acessibilidade em sítios *web* de governo.

2.1.7 Análise dos Resultados

O processo de classificação e análise de artigos com base em alguns critérios podem ser subjetivos. Para minimizar essa limitação, uma abordagem em duas fases foi planejada para seleção de artigos, explicada na seção 2.2, e uma outra abordagem de revisão de todos os artigos selecionados foi planejada para classificação e análise dos estudos. Todos os estudos foram lidos pelo menos três vezes pelo mesmo pesquisador, em momentos diferentes, visando buscar a estabilidade da análise realizada.

2.2 Execução da Revisão Sistemática

A ferramenta Start 2.0 [Zam+10] foi utilizada para gerenciar o grande número de referências que foram obtidas através da pesquisa bibliográfica, assim como, para apoiar o processo de extração de dados e análise dos resultados. Em geral, o resultado da pesquisa bibliográfica de cada base de dados foi exportado e importado diretamente na ferramenta Start 2.0 através do formato BibTeX. Além dessa ferramenta, o software Microsoft Excel® foi utilizado para apoiar a análise quantitativa dos resultados. A ferramenta Start 2.0 possui um mecanismo de exportação de dados para o software Microsoft Excel®, sendo este recurso largamente utilizado nessa pesquisa.

2.2.1 Seleção de Estudos

Os estudos selecionados para análise em profundidade foram obtidos a partir de três etapas, conforme Figura 1. 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. Na segunda etapa, realizou-se a leitura do título e do resumo dos trabalhos, sendo os artigos classificados em três categorias:

- **[Inc]** indica que o estudo está relacionado com desenvolvimento de software (DS) na AP.
- **[Exc]** indica que o estudo não está relacionado com DS na AP.
- **[Dup]** indica que o estudo está duplicado 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, somente a leitura do título e resumo não foi suficiente para classificar cada artigo corretamente. Dessa forma, um subconjunto de documentos relacionados com desenvolvimento de software na AP contidos na categoria **[Inc]** foi selecionado para a etapa de extração de dados e análise em profundidade.

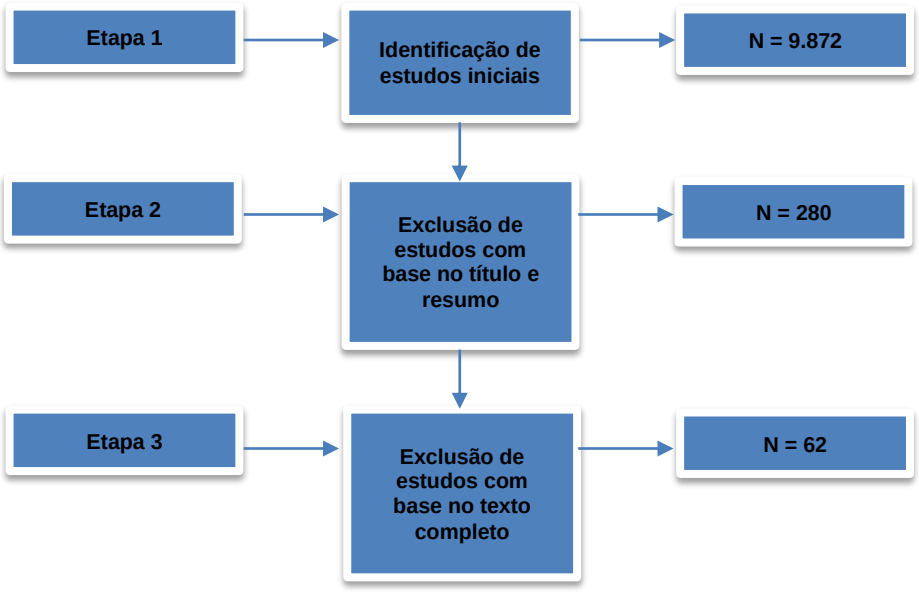

Figura 1 - Etapas do Processo de Seleção de Estudos

2.2.2 Extração dos Dados

Os trabalhos selecionados foram catalogados na ferramenta Start 2.0 de acordo com três categorias gerais de informação:

- **Informações gerais:** Base de dados, Título, Autoria, Palavras-chave, Ano de publicação, Fonte, Tipo da publicação (Journal, Conference ou Workshop), Abstract e Categoria (Incluído, Excluído, Duplicado ou Texto completo não disponível para leitura).
- **Informações relacionadas com a organização da pesquisa:** Tipo da pesquisa (Evaluation research, Validation research, Solution proposal, Philosophical paper, Experience paper ou Opinion paper), Método de pesquisa (Estudo de caso, Survey, Experimento, Etnografia, Revisão da literatura, Grounded theory ou Múltiplos métodos de pesquisa), Método de coleta de dados (Entrevista, Observação, Questionário, Análise de documentos, Análise de bases de dados, Workshop ou Múltiplos métodos de coleta de dados), Tipo de dados analisado (Qualitativo, Quantitativo ou Ambos), Método de análise de dados (Estatístico, Grounded theory, Análise de conteúdo ou Múltiplos métodos de análise de dados).
- **Informações relacionadas com o conteúdo da pesquisa:**
    - **Ambiente público:** (nome, sigla e país da Administração Pública);
    - **Aspectos de Engenharia de Software:** abrangendo a área de conhecimento do Software Engineering Body of Knowledge (SWEBOK), o Modelo / Processo / Método de ES, o Nível de experiência na utilização da abordagem escolhida, as informações sobre quem executou o desenvolvimento do software, assim como, aspectos humanos, organizacionais e técnicos incluindo os problemas encontrados, os desafios enfrentados, as lições aprendidas e os resultados obtidos na pesquisa;
    - **Comentários gerais:** informações relacionadas a temas explorados em cada estudo, assim como, outras informações complementares para orientar a análise qualitativa.

Os trabalhos que inspiraram a elaboração do formulário de extração de dados foram:

- O tipo da pesquisa foi classificado de acordo com a proposta de Wieringa [WMR06].
- O método de pesquisa, o método de coleta de dados, o tipo de dado analisado e o método de análise de dados foram classificados a partir dos seguintes trabalhos: Yin [Yin10], Wohlin [WRH+12], Creswell [Cre14], Oates [Oat06] e Tarozzi [Tar11].
- A área de conhecimento de ES foi preenchida de acordo com o Guia SWEBOK versão 3.0 [BoF14]. Este guia é um documento criado sob o patrocínio do Institute

of Electrical and Electronics Engineers (IEEE) com a finalidade de servir de referência em assuntos considerados, de forma generalizada pela comunidade, como pertinentes a área de ES.

- Os modelos, métodos e processos de ES foram preenchidos de acordo com a proposta de Pressman [Pre11].
- As informações referentes aos aspectos humanos, organizacionais e técnicos de ES foram preenchidas de acordo com a proposta de Hazzan [HaD08].

3 ANÁLISE DA REVISÃO SISTEMÁTICA

Neste capítulo será apresentado os resultados da revisão sistemática. Na seção 3.1 é apresentado a análise dos resultados sob uma perspectiva quantitativa, enquanto na seção 3.2 é relatada a análise qualitativa dos resultados. Uma abordagem em três etapas foi planejada para a seleção dos artigos, conforme Figura 1. Na primeira etapa, um total de 9.872 artigos foram encontrados, conforme Tabela 2. Após a triagem inicial, 280 artigos foram selecionados na segunda etapa, conforme Tabela 3. Na sequência, 62 artigos foram selecionados para uma análise em profundidade, conforme Tabela 3. Foi observado que a falta de terminologia para desenvolvimento de software na AP e a baixa precisão dos mecanismos de busca resultou em uma grande quantidade de estudos iniciais para investigação. Porém, somente alguns deles foram relevantes para responder a questão de pesquisa, conforme Apêndice A, evidenciando uma baixa precisão da estratégia de busca desta RSL quando submetida às bases de dados para consulta.

Tabela 2 - Execução da consulta, primeiros resultados.

Base de dados	**Número de artigos**	**Primeira classificação**		
		[Inc]	**[Exc]**	**[Dup]**
Scopus	1530	146	[P13]	3
Engineering Village	189	1	12	176
Bielefeld Academic Search Engine	329	12	288	29
IEEEXplore	110	14	51	45
Springer	2975	44	2822	109
Wiley Online Library	2012	15	1942	55
Web of Knowledge	93	8	28	57
ScienceDirect	2525	21	2347	157
ACM Digital Library	95	12	65	18
Bases de Dados da Pesquisa Agropecuária	1	1	-	-
Biblioteca Digital Brasileira de Computação	12	5	5	2
Workshop Brasileiro de Métodos Ágeis	2	2	-	-
Total	**9872**	**280**	**8941**	**651**
Porcentagem (%)	**100**	**2,84**	**90,57**	**6,59**

Todos os artigos selecionados para análise em profundidade foram classificados de acordo com as três categorias de informação, incluindo informações gerais, informações relacionadas com a organização da pesquisa e informações relacionadas com o conteúdo da pesquisa. Esta classificação permitiu realizar uma análise quantitativa dos resultados, detalhada na seção 3.1. Enquanto, a análise qualitativa dos resultados foi realizada através das informações relacionadas com o conteúdo da pesquisa, abrangendo os resultados obtidos em cada aspecto da ES (humano, organizacional e técnico). Os resultados alcançados pelos estudos individuais foram agrupados em algumas dimensões, sendo

inspiradas nas áreas do conhecimento do SWEBOK [BoF14] e elementos de ES abordados por Pressman [Pre11]. Esta ação de realizar a análise qualitativa dos resultados por dimensões facilitou o trabalho de caracterização da área de estudo, assim como, contribuiu para a formação da base teórica sobre o tema, uma vez que a análise qualitativa dos resultados ocorreu sobre um conjunto menor de estudos individuais pertencentes a uma determinada dimensão.

Tabela 3 - Execução da consulta, segundo resultado.

Base de dados	**Número de artigos**	**Segunda classificação**			**Total de artigos selecionados**
		[Exc]	**[Dup]**	**[Ntc]**	
Scopus	146	89	4	16	37
Engineering Village	1	-	-	1	0
Bielefeld Academic Search Engine	12	9	-	2	1
IEEEXplore	14	10	-	-	4
Springer	44	32	1	4	7
Wiley Online Library	15	8	1	1	5
Web of Knowledge	8	5	-	3	0
ScienceDirect	21	20	-	-	1
ACM Digital Library	12	8	-	1	3
Bases de Dados da Pesquisa Agropecuária	1	-		-	1
Biblioteca Digital Brasileira de Computação	5	2	1	-	2
Workshop Brasileiro de Métodos Ágeis	1	-	-	-	1
Total	**280**	**183**	**7**	**28**	**62**
Porcentagem (%)	**100**	**65,36**	**2,50**	**10,00**	**22,14**

3.1 Análise Quantitativa dos Resultados

Nesta seção será apresentada uma análise quantitativa dos estudos selecionados de acordo com as três categorias de informação, incluindo informações gerais, informações relacionadas com a organização da pesquisa e informações relacionadas com o conteúdo da pesquisa.

3.1.1 Informações Gerais

O conjunto final de artigos selecionados para análise em profundidade, listados no Apêndice A, foram organizados em três categorias: por base de dados, por ano de publicação e por tipo da publicação. Na sequência, as informações coletadas nesta dimensão são apresentadas de maneira quantitativa.

3.1.1.1 Por Base de Dados

A Tabela 4 apresenta os artigos selecionados por base de dados, sendo que trinta e sete estudos (59,68%) foram recuperados a partir da base de dados Scopus, justamente por

ter sido a primeira base de dados utilizada para consulta e oferecer uma maior cobertura de publicações científicas.

Tabela 4 - Artigos selecionados por base de dados.

Base de dados	Número de artigos	ID
Scopus	37 (59,68%)	P03, P04, P05, P06, P10, P11, P13, P15, P16, P17, P18, P19, P20, P22, P24, P25, P26, P27, P28, P29, P30, P31, P33, P34, P35, P38, P41, P44, P47, P48, P52, P55, P56, P57, P59, P60, P62
Bielefeld Academic Search Engine	01 (01,61%)	P23
IEEEXplore	04 (06,45%)	P01, P07, P12, P40
Springer	07 (11,29%)	P32, P37, P42, P43, P49, P50, P54
Wiley Online Library	05 (08,06%)	P02, P08, P09, P21, P46
ScienceDirect	01 (01,61%)	P14
ACM Digital Library	03 (04,84%)	P53, P58, P61
Bases de Dados da Pesquisa Agropecuária	01 (01,61%)	P45
Biblioteca Digital Brasileira de Computação	02 (03,23%)	P39, P51
Workshop Brasileiro de Métodos Ágeis	01 (01,61%)	P36

3.1.1.2 Por Ano de Publicação

A Tabela 5 apresenta os artigos selecionados por ano de publicação. No ano de 2013 os artigos foram selecionados até o mês de Outubro (mês em que a primeira etapa da execução da revisão sistemática foi encerrada). Pode-se observar que os maiores números de estudos foram publicados no ano de 2010. Entre 1991 e 1993 não foram encontrados estudos publicados sobre o tema.

Tabela 5 - Artigos selecionados por ano de publicação.

Ano de publicação	Número de artigos	ID
1990	01 (01,61%)	P14
1994	01 (01,61%)	P52
1995	01 (01,61%)	P08
1996	01 (01,61%)	P54
1998	01 (01,61%)	P30
1999	02 (03,23%)	P07, P38
2000	01 (01,61%)	P32
2001	03 (04,84%)	P40, P49, P53
2002	02 (03,23%)	P10, P45
2003	02 (03,23%)	P01, P60
2004	02 (03,23%)	P02, P55
2005	05 (08,06%)	P09, P13, P42, P43, P59
2006	07 (11,29%)	P17, P24, P28, P29, P35, P50, P57
2007	04 (06,45%)	P05, P11, P18, P41
2008	05 (08,06%)	P03, P15, P31, P47, P62
2009	06 (09,68%)	P04, P06, P20, P23, P33, P58
2010	10 (16,13%)	P12, P21, P22, P34, P36, P39, P44, P48, P56, P61
2011	03 (04,84%)	P16, P27, P51
2012	03 (04,84%)	P19, P26, P46
2013	02 (03,23%)	P25, P37

3.1.1.3 Por Tipo da Publicação

A Tabela 6 apresenta os artigos selecionados por tipo da publicação, sendo que *Journal* foi a opção mais utilizada para publicação, seguido por *Conference* e *Workshop*.

Tabela 6 - Artigos selecionados por tipo de publicação.

Tipo da publicação	Número de artigos	ID
Periódico	33 (53,23%)	P02, P03, P05, P06, P08, P09, P10, P11, P14, P16, P17, P18, P19, P21, P22, P25, P30, P32, P33, P34, P35, P38, P41, P43, P46, P53, P54, P55, P56, P57, P60, P61, P62
Conferência	22 (35,48%)	P01, P07, P12, P13, P15, P20, P23, P24, P26, P27, P29, P31, P37, P39, P42, P44, P45, P47, P48, P49, P52, P59
Workshop	07 (11,29%)	P04, P28, P36, P40, P50, P51, P58

3.1.2 Informações Relacionadas com a Organização da Pesquisa

O Apêndice B apresenta as informações relacionadas com a organização da pesquisa, incluindo o tipo da pesquisa (de acordo com Wieringa [WMR06]), o método de pesquisa, o método de coleta de dados, o tipo de dados analisado e o método de análise de dados. Na sequência, as informações coletadas nesta dimensão são apresentadas de modo quantitativo.

3.1.2.1 Por Tipo da Pesquisa

A Tabela 7 mostra os artigos selecionados por tipo da pesquisa. Seis estudos (9,68%) de opinião foram incluídos por apresentarem trabalhos relacionados e incluírem a experiência pessoal do(s) autor(es) na percepção de como o desenvolvimento de software na AP ocorreu na prática. Duas propostas de soluções (3,23%) foram incluídas por apresentarem uma discussão prática dos seus benefícios. Os demais artigos apresentaram resultados diretamente relacionados com a questão de pesquisa formulada, onde aspectos do desenvolvimento de software na AP foram implementados e avaliados na prática, e suas consequências foram investigadas.

Tabela 7 - Artigos selecionados por tipo da pesquisa.

Tipo da pesquisa	Número de artigos	ID
Evaluation research	54 (87,10%)	P01, P02, P03, P04, P05, P06, P07, P08, P09, P10, P11, P12, P13, P14, P15, P16, P17, P19, P20, P22, P23, P24, P25, P26, P27, P28, P29, P30, P31, P32, P33, P36, P38, P39, P40, P41, P42, P43, P44, P45, P46, P47, P48, P49, P51, P52, P53, P54, P56, P57, P58, P59, P60, P61
Opinion paper	06 (09,68%)	P18, P34, P35, P37, P55, P62
Solution proposal	02 (03,23%)	P21, P50

3.1.2.2 Por Método de Pesquisa

A Tabela 8 apresenta os artigos selecionados por método de pesquisa. O método de pesquisa mais utilizado foi Estudo de caso, seguido por *Survey* e múltiplos estudos de caso. Uma revisão da literatura foi incluída por apresentar informações históricas importantes sobre a utilização de modelos, processos e métodos de ES em projetos de software do governo dos EUA [P34]. Somente três artigos não foram classificados por não apresentarem informações claras sobre o método de pesquisa utilizado [P14] [P30] [P35].

Tabela 8 - Artigos selecionados por método de pesquisa.

Método de pesquisa	**Número de artigos**	**ID**
Estudo de caso	29 (46,77%)	P01, P07, P09, P10, P11, P12, P13, P15, P16, P19, P22, P23, P24, P25, P28, P31, P32, P36, P37, P39, P40, P43, P45, P48, P51, P53, P56, P57, P59
Estudo de caso Etnografia	02 (03,23%)	P05, P06
Experimento	04 (06,45%)	P02, P29, P42, P60
Grounded theory	01 (01,61%)	P58
Múltiplos estudos de caso	08 (12,90%)	P03, P04, P18, P21, P44, P46, P54, P62
Múltiplos estudos de caso Survey	01 (01,61%)	P38
Revisão da literatura	01 (01,61%)	P34
Survey	13 (20,97%)	P08, P17, P20, P26, P27, P33, P41, P47, P49, P50, P52, P55, P61
Não definido (ND)	03 (04,84%)	P14, P30, P35

3.1.2.3 Por Método de Coleta de Dados

Conforme Tabela 9, de maneira única, ou então, combinada com outros instrumentos, entrevista foi a técnica mais utilizada para coleta de dados. Entretanto, em vinte e quatro estudos (38,71%) não foi possível definir o método de coleta de dados utilizado, o que demonstra uma falta de informações relacionadas com a organização da pesquisa.

Tabela 9 - Artigos selecionados por método de coleta de dados.

Método de coleta de dados	Número de artigos	ID
Análise de documentos	04 (06,45%)	P18, P21, P34, P62
Análise de bases de dados	02 (03,23%)	P52, P53
Entrevista	10 (16,13%)	P03, P08, P10, P11, P19, P44, P46, P50, P55, P57
Entrevista Análise de documentos	01 (01,61%)	P39
Entrevista Análise de bases de dados	02 (03,23%)	P48, P54
Entrevista Análise de documentos Análise de bases de dados	01 (01,61%)	P20
Entrevista Análise de documentos Observação	02 (03,23%)	P06, P09
Entrevista Análise de documentos Observação Questionário	04 (06,45%)	P13, P15, P31, P36
Entrevista Análise de documentos Observação Workshop	01 (01,61%)	P58
Entrevista Observação	02 (03,23%)	P05, P47
Entrevista Observação Questionário	01 (01,61%)	P38
Entrevista Questionário	03 (04,84%)	P17, P27, P49
Observação Análise de bases de dados	01 (01,61%)	P23
Questionário	03 (04,84%)	P26, P33, P61
Questionário Análise de bases de dados	01 (01,61%)	P41
Não definido (ND)	24 (38,71%)	P01, P02, P04, P07, P12, P14, P16, P22, P24, P25, P28, P29, P30, P32, P35, P37, P40, P42, P43, P45, P51, P56, P59, P60

3.1.2.4 Por Tipo de Dados Analisado

A Tabela 10 apresenta os artigos selecionados por tipo de dados analisado. Quanto ao tipo de dados analisado, predominou a utilização da abordagem qualitativa, seguida pela combinação das abordagens quantitativa/qualitativa, o que corrobora com a questão de pesquisa formulada, onde informações descritivas sobre a prática de desenvolvimento de

software na AP são mais esperadas. Porém, em nove estudos (14,52%) não foi possível definir o tipo de dados analisado, o que demonstra novamente uma falta de informações relacionadas com a organização da pesquisa.

Tabela 10 - Artigos selecionados por tipo de dados analisado.

Tipo de dados analisado	Número de artigos	ID
Quantitativo	10 (16,13%)	P04, P17, P20, P26, P28, P39, P41, P53, P54, P61
Qualitativo	31 (50,00%)	P05, P06, P08, P09, P10, P14, P16, P18, P19, P21, P22, P24, P30, P33, P34, P35, P38, P42, P43, P44, P45, P46, P47, P49, P50, P52, P55, P57, P59, P60, P62
Ambos	12 (19,35%)	P02, P03, P07, P11, P13, P15, P23, P27, P31, P36, P48, P58
Não definido (ND)	09 (14,52%)	P01, P12, P25, P29, P32, P37, P40, P51, P56

3.1.2.5 Por Método de Análise de Dados

A Tabela 11 mostra que em vinte estudos (32,26%), a informação sobre o método de análise de dados utilizado não estava clara. A análise de conteúdo foi predominantemente utilizada para análise qualitativa dos estudos selecionados, enquanto, o método estatístico foi adotado para análise quantitativa dos trabalhos escolhidos.

Tabela 11 - Artigos selecionados por método de análise de dados.

Método de análise de dados	Número de artigos	ID
Análise de conteúdo	20 (32,26%)	P06, P08, P09, P10, P18, P21, P22, P33, P34, P38, P44, P46, P47, P49, P50, P52, P55, P57, P59, P62
Análise de conteúdo Grounded Theory	01 (01,61%)	P05
Estatístico	12 (19,35%)	P02, P04, P17, P20, P26, P28, P39, P41, P48, P53, P54, P61
Estatístico Análise de conteúdo	08 (12,90%)	P03, P07, P13, P15, P23, P27, P31, P36
Estatístico Grounded theory	01 (01,61%)	P58
Não definido (ND)	20 (32,26%)	P01, P11, P12, P14, P16, P19, P24, P25, P29, P30, P32, P35, P37, P40, P42, P43, P45, P51, P56, P60

3.1.3 Informações Relacionadas com o Conteúdo da Pesquisa

O Apêndice C mostra as informações relacionadas com o conteúdo da pesquisa, incluindo o ambiente público, a área do conhecimento do SWEBOK [BoF14] e informações sobre os aspectos de ES dos estudos selecionados. Na sequência, as informações coletadas nesta dimensão são mostradas de maneira quantitativa.

3.1.3.1 Por País

A Tabela 12 mostra os artigos selecionados por país, onde predominou os artigos de APs dos Estados Unidos, Brasil e Reino Unido. Somente em dois estudos (3,23%) não foi possível determinar o país de origem, justamente por se tratarem de estudos gerais de desenvolvimento de software no setor público [P18] e [P61].

Tabela 12 - Artigos selecionados por país.

País	Número de artigos	ID
Alemanha	03 (04,84%)	P27, P28, P32
Argentina	02 (03,23%)	P07, P31
Austrália	02 (03,23%)	P03, P43
Barbados	01 (01,61%)	P11
Bélgica	01 (01,61%)	P58
Brasil	06 (09,68%)	P29, P33, P36, P39, P45, P51
Bulgária	01 (01,61%)	P23
Canadá	02 (03,23%)	P47, P48
Chile	01 (01,61%)	P42
Coréia do Sul	01 (01,61%)	P25
Emirados Árabes Unidos	02 (03,23%)	P19, P20
Espanha	01 (01,61%)	P49
Estados Unidos (EUA)	21 (33,87%)	P01, P02, P08, P12, P14, P15, P16, P17, P21, P22, P34, P35, P40, P50, P52, P53, P54, P55, P57, P59, P60
Israel	02 (03,23%)	P13, P41
Itália	02 (03,23%)	P04, P10
Japão	01 (01,61%)	P24
Noruega	02 (03,23%)	P09, P26
Reino Unido (UK)	05 (08,06%)	P05, P06, P30, P37, P38
UK, EUA e Nova Zelândia	01 (01,61%)	P62
Sérvia	01 (01,61%)	P56
Suécia	02 (03,23%)	P44, P46
Não definido (ND)	01 (01,61%)	P61
Não se aplica (NA)	01 (01,61%)	P18

3.1.3.2 Por Área do SWEBOK

Todos os artigos selecionados foram classificados de acordo com as áreas de conhecimento do SWEBOK versão 3.0 [BoF14]. Conforme mostra a Tabela 13: Modelos e Métodos de Engenharia de Software, Gerência de Engenharia de Software e Processos de Engenharia de Software prevaleceram nesta pesquisa, o que confirma a questão de pesquisa formulada, onde modelos, processos e métodos de ES aplicados em projetos de desenvolvimento de software na AP são de interesse deste estudo.

Tabela 13 - Artigos selecionados por área do SWEBOK.

Área do SWEBOK	Número de artigos	ID
Software Design	01 (01,61%)	P10
Software Engineering Economics	01 (01,61%)	P18
Software Engineering Management	15 (24,19%)	P01, P03, P11, P13, P17, P20, P21, P26, P35, P41, P50, P55, P58, P61, P62
Software Engineering Models and Methods	29 (46,77%)	P04, P05, P06, P09, P15, P16, P19, P22, P23, P24, P25, P27, P28, P32, P34, P36, P37, P38, P40, P42, P44, P45, P46, P47, P48, P56, P57, P59, P60
Software Engineering Process	05 (08,06%)	P07, P30, P31, P33, P51
Software Maintenance	02 (03,23%)	P08, P49
Software Quality	03 (04,84%)	P14, P52, P53
Software Requirements	01 (01,61%)	P43
Software Testing	05 (08,06%)	P02, P12, P29, P39, P54

3.1.3.3 Por Modelos, Processos e Métodos de Engenharia de Software

O resultado da Tabela 14 mostra a introdução cada vez maior de processos adaptativos de desenvolvimento de software no setor público, incluindo métodos ágeis e processo unificado, vinte e nove estudos (46,77%). Nove estudos (14,52%) abrangendo modelos, processos e métodos de ES não puderam ser classificados facilmente nas categorias de modelos prescrito ou adaptativo, e foram incluídos na categoria Outros. Dezesseis estudos (25,81%) sobre desenvolvimento de software no setor público não mencionaram claramente a abordagem de desenvolvimento utilizada, ou então, esta informação não fazia sentido para o contexto do trabalho. Por fim, três estudos (4,84%) sobre modelos de maturidade e capacidade de software foram incluídos por apresentarem resultados práticos que influenciaram diretamente as atividades de desenvolvimento de software na AP estudada.

Além disso, a utilização de termos específicos sobre métodos ágeis e processo unificado na composição da estratégia de busca maximizou o resultado de estudos selecionados que adotaram estas abordagens. Isto significa que quanto mais específico for o termo de busca, maior é a probabilidade de encontrar estudos relacionados a partir do termo de busca especificado. Por outro lado, a utilização de termos genéricos sobre desenvolvimento de software não foi capaz de encontrar muitos estudos na AP abrangendo outras abordagens de desenvolvimento de software, incluindo os modelos *waterfall*, iterativo e incremental, evolucionário, etc.

Tabela 14 - Artigos selecionados por modelos, processos e métodos de ES.

Modelo, Processo e Método de Engenharia de Software	Número de artigos	ID
Modelos Prescritivos de Processo	**05 (08,06%)**	
Structured Systems Analysis and Design Method (SSADM)	01 (01,61%)	[P38]
Modelo Cascata (Waterfall)	04 (06,45%)	[P10] [P34] [P53] [P54]
Modelos Adaptativos de Processo	**29 (46,77%)**	
Evolucionário	01 (01,61%)	[P32]
Iterativo e Incremental	01 (01,61%)	[P58]
Métodos Ágeis	**17 (27,42%)**	
Extreme Programming	06 (09,68%)	[P01] [P13] [P15] [P40] [P45] [P60]
Scrum	03 (04,84%)	[P16] [P19] [P44]
Extreme Programming e Scrum	02 (03,23%)	[P36] [P59]
Não definido (ND)	06 (09,68%)	[P05] [P23] [P24] [P35] [P37] [P57]
Processo Unificado	**06 (09,68%)**	
Agile Unified Process	01 (01,61%)	[P04]
Rational Unified Process	04 (06,45%)	[P29] [P47] [P48] [P56]
Unified Software Development Process	01 (01,61%)	[P42]
Rapid Application Development	01 (01,61%)	[P06]
Microsoft Solutions Framework	01 (01,61%)	[P09]

Modelo, Processo e Método de Engenharia de Software	Número de artigos	ID
V-Modell XT	02 (03,23%)	[P27] [P28]
Modelos de Maturidade e Capacidade	**03 (04,84%)**	
CompetiSoft	01 (01,61%)	[P31]
CMM	01 (01,61%)	[P07]
MPS.BR	01 (01,61%)	[P51]
Outros modelos, processos e métodos de ES	**09 (14,52%)**	
ABS Software Development Process	01 (01,61%)	[P43]
Computer-Aided Software Engineering	02 (03,23%)	[P08] [P30]
MODAF	01 (01,61%)	[P46]
OSS Development Models	01 (01,61%)	[P25]
Processo SERPRO de Desenvolvimento de Soluções	01 (01,61%)	[P33]
Software Engineering Evaluation System (SEES)	01 (01,61%)	[P02]
Team Software Process	01 (01,61%)	[P22]
Verificação e Validação Independente (IV&V)	01 (01,61%)	[P12]
Não definido (ND)	03 (04,84%)	[P03] [P39] [P49]
Não se aplica (NA)	13 (20,97%)	[P11] [P14] [P17] [P18] [P20] [P21] [P26] [P41] [P50] [P52] [P55] [P61] [P62]

3.1.3.4 Por Nível de Experiência

A Tabela 15 mostra o nível de experiência da AP na utilização do modelo, processo e método de ES escolhido, com predomínio do nível de experiência Iniciante. Isto significa que na maioria dos estudos selecionados a abordagem escolhida encontrava-se em um estágio de “projeto-piloto”, geralmente adotada para sanar problemas de outras abordagens utilizadas anteriormente.

Tabela 15 - Artigos selecionados por nível de experiência.

Nível de Experiência	Número de artigos	ID
Iniciante	27 (43,55%)	P02, P06, P07, P09, P12, P13, P15, P16, P19, P23, P24, P25, P28, P29, P31, P36, P37, P39, P40, P42, P45, P46, P47, P51, P58, P59, P60
Experiente	10 (16,13%)	P04, P05, P08, P22, P35, P43, P48, P53, P54, P57
Não definido (ND)	07 (11,29%)	P01, P10, P30, P32, P44, P49, P56
Não se aplica (NA)	18 (29,03%)	P03, P11, P14, P17, P18, P20, P21, P26, P27, P33, P34, P38, P41, P50, P52, P55, P61, P62

3.1.3.5 Por Quem Executou o Desenvolvimento do Software

A Tabela 16 apresenta os estudos selecionados por quem executou o desenvolvimento do software. Vinte e dois estudos (35,48%) de desenvolvimento de software foram executados inteiramente por servidores públicos em ambiente interno do governo. Enquanto nove estudos (14,52%) de desenvolvimento de software na AP foram executados de maneira conjunta com a indústria e/ou academia. Sete estudos (11,29%)

relataram que a responsabilidade do desenvolvimento de software foi transferida inteiramente para a indústria ou academia, ou seja, o governo atuou como cliente durante o processo de desenvolvimento de software. Por fim, vinte e quatro estudos (38,71%) sobre desenvolvimento de software na AP não relataram qual era o papel do governo durante o desenvolvimento do produto de software, ou então, esta informação não fazia sentido para o contexto do estudo.

Tabela 16 - Artigos selecionados por quem executou o desenvolvimento do software.

Quem executou o desenvolvimento do software?	Número de artigos	ID
Academia	02 (03,23%)	P24, P29
Indústria	05 (08,06%)	P01, P06, P10, P15, P59
Governo	22 (35,48%)	P04, P07, P09, P12, P13, P19, P22, P23, P28, P30, P31, P36, P39, P40, P42, P43, P45, P47, P48, P49, P51, P60
Governo e Academia	01 (01,61%)	P02
Governo e Indústria	05 (08,06%)	P08, P16, P25, P46, P57
Governo, Academia e Indústria	01 (01,61%)	P32
Governo Colaborativo	02 (03,23%)	P44, P58
Não definido (ND)	06 (09,68%)	P05, P35, P37, P53, P54, P56
Não se aplica (NA)	18 (29,03%)	P03, P11, P14, P17, P18, P20, P21, P26, P27, P33, P34, P38, P41, P50, P52, P55, P61, P62

Em resumo, os dados quantitativos mostraram uma maior utilização de modelos adaptativos de processo de software na AP oriundos da indústria de software (incluindo métodos ágeis e RUP), totalizando 21 estudos (33,87%), o que corrobora com a constatação de que o setor público tem adotado e aplicado tecnologias e processos correspondentes bem depois dessas abordagens terem sido experimentadas e avaliadas pelo setor privado [MEL09]. Outra constatação foi que Estudo(s) de Caso e/ou Survey (totalizando 53 – 85,48%), assim como, Entrevistas (totalizando 27 – 43,55%) e Métodos de análise qualitativa de dados (totalizando trinta estudos – 48,39%) foram as estratégias de organização da pesquisa mais utilizadas.

3.2 Análise Qualitativa dos Resultados

Nesta seção será apresentada uma análise qualitativa dos estudos selecionados com vista a responder a questão de pesquisa definida no protocolo da revisão sistemática. Uma vez que o principal interesse é sobre o desenvolvimento de software na AP, a análise incidiu sobre artigos com evidências práticas da experiência de desenvolvimento de software no setor público, organizadas em algumas dimensões, listadas a seguir.

3.2.1 Desenvolvimento de Software na Administração Pública

Desde o estabelecimento da ES, o governo sempre buscou implantar padrões para estruturar o desenvolvimento de software no setor público, inspirado inicialmente em modelos prescritivos, que possuem em sua estrutura uma ordem formal de elementos do processo, bem como, um fluxo de trabalho que descreve como cada um destes elementos se relaciona uns com os outros. A principal característica desses modelos é, portanto, a busca pela estrutura e a ordem [Pre11]. O *United States Department of Defense* (DoD) institui os padrões MIL-STDs e DOD-STDs entre 1974 e 1998 [P34]. Da mesma maneira, 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 [P38], enquanto V-Modell e suas variantes tem sido obrigatório para projetos de software no governo da Alemanha desde o início da década de 1980 [P27] [P28]. A história deste esforço revelou inicialmente uma disputa pelo controle do desenvolvimento de software entre o setor público (contratante) e empresas contratadas [P34]. Neste período, o governo foi o principal cliente da indústria e influenciou diretamente a maneira pela qual as empresas produziam softwares [P34] [P38].

O resultado da abordagem SSADM foi percebida como prescritiva, onerosa e de difícil aplicação, assim como, mostrou dificuldade em lidar com a incerteza inerente de projetos de ES, comunicação com o usuário, desenvolvimento pessoal e não refletiu a forma como as pessoas trabalhavam na prática [P38]. Igualmente, o DoD enfrentou diversas falhas de projeto que ocasionaram o desuso dos padrões MIL-STD-1679 e DOD-STD-2167 [P34]. Estas e outras constatações indicaram que abordagens baseadas no modelo prescritivo não eram a melhor maneira de desenvolver softwares para a maioria dos projetos do setor público [P34] [P38]. Para resolver este problema, novos modelos de desenvolvimento de software com base em modelos adaptativos, como DOD-STD-2167A e MIL-STD-498 [P34] e V-Modell XT [P27] [P28] foram propostos pelo governo.

Entretanto, com o crescimento exponencial do mercado, 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 [P34]. Uma das respostas da indústria de software, por exemplo, para lidar com as limitações de projetos baseados em abordagens prescritivas de ES, típica de projetos de governo [P16], foi dada em fevereiro de 2001, quando um grupo de dezessete renomados desenvolvedores, autores e consultores da área de software, praticantes de *Dynamic Systems Development Method* (DSDM), *Extreme Programming* (XP), Scrum e *Feature Driven Development* (FDD) se

reuniram para discutir sobre suas experiências de trabalho e pontos em comum. O resultado do encontro deu origem ao manifesto ágil [BBB+01], uma declaração com quatro valores e doze princípios que são, na visão de seus proponentes, determinantes entender o desenvolvimento de software como um processo criativo, adaptativo, e movido a incertezas. Desde então, cada vez mais, o governo tem renunciado as suas formas de trabalho e tem adotado métodos e práticas alinhados aos princípios do desenvolvimento adaptativo. Na sequência, os modelos, processos e métodos de Engenharia de Software (ES) adotados na AP encontrados nessa pesquisa são apresentados.

3.2.1.1 Modelos Prescritivos de Processo

O padrão *Structured Systems Analysis and Design Method* (SSADM) foi obrigatório para desenvolvimento de software no Reino Unido de 1981 até meados da década de 2000 [P38]. O estudo de Middleton [P38] sobre desenvolvimento de sistemas de informação (SI) em ambientes de governo, tipicamente burocrático, apontou que abordagens baseadas em modelos prescritivos não são susceptíveis de lidar bem com a incerteza estratégica, comunicação com o usuário e desenvolvimento pessoal. Além disso, o estudo mostrou que na prática ocorrem mudanças fundamentais de estratégias durante à execução de projetos, devido às mudanças da alta administração. A suposição de um contexto estratégico estável e coerente mostrou-se inválida na prática. Segundo o autor, isto significa que modelos prescritivos pode não ser a melhor maneira de desenvolver softwares para a maioria dos projetos do setor público, sendo necessário investir em um modelo de desenvolvimento de software que é mais adequado para lidar com a incerteza estratégica.

No mesmo estudo, o autor mostrou que a necessidade de começar com requisitos fixos em todos os casos foi um obstáculo e tornou-se inviável. Como consequência disto, 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. A evidência dos projetos estudados foi que a abordagem SSADM limitou a contribuição do usuário para envolvimento em vez de sua participação com maior comprometimento. Por fim, a abordagem SSADM concentrou o desenvolvedor sobre as técnicas de desenho técnico e não nas habilidades sociais de facilitação do processo humano de comunicação. Ao incentivar esta prática, a comunicação real foi perdida, em parte porque diagramas tendem a ser imprecisos na prática.

3.2.1.2 Modelos Adaptativos de Processo

Devido ao maior número de estudos encontrados sobre modelos adaptativos de PDS, foi necessário reorganizar os artigos em algumas subcategorias, incluindo Modelo Evolucionário, Processo Unificado, Métodos Ágeis, *Rapid Application Development* (RAD) e *Microsoft Solutions Framework* (MSF).

3.2.1.2.1 Modelo Evolucionário

A utilização do modelo evolucionário para o desenvolvimento de software foi relatada no estudo de Sohlenkamp *et al.* [P32]. Os usuários finais foram envolvidos de maneira intensiva, e práticas interativas como oficinas, reuniões e discussões em grupo foram aplicadas durante todo o processo de desenvolvimento. O processo de desenvolvimento foi conduzido de maneira cooperativa, proporcionando um ambiente de aprendizagem mútua entre usuários e desenvolvedores para troca de experiências, assim como, o aspecto cognitivo das pessoas para aceitação das mudanças necessárias foi melhorado. Os representantes dos usuários foram integrados nas reuniões de projeto para discutir as necessidades dos usuários, tal como, priorizar os resultados que poderiam ser antecipados. A definição dos requisitos para o sistema foi realizada de tal maneira que, um sistema com uma funcionalidade pronta foi instalado no ambiente do usuário, e os requisitos foram identificados a partir das experiências práticas obtidas durante o uso do sistema.

Segundo os autores, os benefícios da utilização da abordagem evolucionária foram vistos logo no primeiro ano. Os requisitos do usuário foram rapidamente adotados e integrados ao sistema. Este processo de melhoria contínua aumentou a satisfação do usuário pelo sistema de maneira significativa. Outrossim, a estreita colaboração com os usuários e a utilização prévia do sistema aumentou o número de requisitos que não poderiam ser obtidos de outra forma. Em particular, para o sistema desenvolvido descobriu-se que os detalhes só poderiam ser vistos quando o sistema fosse realmente utilizado. Os requisitos que os usuários identificaram antes do uso do sistema foram consideravelmente diferentes daqueles que surgiram ao longo da utilização do sistema.

A pesquisa recomendou que os usuários devem ser vistos como parceiros e não como concorrentes ou inimigos. A abordagem participativa levou a uma melhor compreensão do trabalho dos usuários e da organização. A estratégia envolveu reunir todos os membros do projeto em sessões de entrevistas, dando-lhes a oportunidade de estarem em contato direto

com os usuários do sistema. Isto provou ser uma estratégia valiosa, especialmente para os *designers*, que tiveram uma melhor compreensão sobre as exigências dos usuários.

3.2.1.2.2 Processo Unificado

Foram encontrados seis (9,68%) estudos sobre a utilização de processo unificado no governo, sendo apresentados a seguir.

3.2.1.2.2.1 Estudo de uma Organização de Governo no Chile

O estudo de Navón [P42] relatou a utilização da abordagem *Unified Software Development Process* (USDP) para criar uma representação do domínio de negócio, cujo os artefatos poderiam ser mapeados facilmente através de restrições adequadas para uma plataforma específica de desenvolvimento *web*. A proposta foi avaliada em um projeto piloto e o resultado mostrou que a metodologia foi facilmente aprendida e adotada pelos desenvolvedores, além disso, o software foi construído mais rapidamente. Além do mais, os usuários relataram que o projeto resultante foi superior em termos de elegância e flexibilidade quando comparado aos projetos anteriores.

3.2.1.2.2.2 Estudo de uma Grande Agência de Petróleo e Gás do Governo do Canadá

O trabalho de Pinheiro *et al.* [P47] apresentou um estudo sobre os principais benefícios alcançados com a transição do modelo prescritivo para o modelo adaptativo. Segundo os autores, o modelo prescritivo não estava sendo mais capaz de suportar o ritmo crescente de desenvolvimento, e muitos lançamentos foram adiados, resultando em uma baixa qualidade de software e custos acima do previsto. A solução encontrada pela empresa foi adotar modelos adaptativos, como o processo *Rational Unified Process* (RUP). A implementação de RUP no primeiro projeto, resultou em um projeto entregue dentro do prazo e no orçamento previsto, algo que não acontecia a mais de seis anos. Os autores relataram que os principais benefícios da adoção de RUP foram: um cronograma mais organizado, uma melhor negociação e priorização de requisitos, um melhor planejamento de lançamentos futuros, uma redução do número de defeitos encontrados após lançamento no ambiente de produção, assim como, um maior envolvimento dos parceiros de negócio e uma melhoria na execução de testes de aceitação do usuário. Além disso, a adoção de RUP forneceu uma mudança social inicial para adoção de algumas práticas ágeis, incluindo testes unitários, testes automatizados, integração contínua e reuniões diárias.

3.2.1.2.2.3 Estudo de uma Agência de Governo do Canadá

O mesmo cenário apresentado no trabalho [P47] foi encontrado no estudo [P48], inclusive com resultados semelhantes. Pinheiro *et al.* [P48] relatou que a transição de abordagens prescritivas para RUP em toda a empresa favoreceu a entrega de projetos dentro do prazo e no orçamento previsto, assim como, a porcentagem de defeitos encontrados no ambiente de produção após o lançamento de uma nova versão do software foi reduzida de 40% para 25% e o tempo de resposta para resolução de defeitos também foi reduzido substancialmente quando comparado ao modelo prescritivo. Outrossim, a execução do desenvolvimento envolveu iterações curtas, apresentação do software ao final de cada iteração, uso de protótipos e testes de aceitação com usuários.

3.2.1.3 Estudo do Inst. de Aeronáutica e Espaço e Inst. Nacional de Pesquisas Espaciais

O desenvolvimento adaptativo apoiado por doze artefatos do RUP, incluindo Caso de Uso, Plano de Testes e Diagrama de Sequência, foi apresentado no estudo de Loubach *et al*. [P29] sobre desenvolvimento de aplicações aeroespaciais. O estudo revelou que testes automatizados juntamente com métricas de software melhoraram a qualidade, confiabilidade e segurança do software entregue. Tal como, reduziu o tempo necessário para realização de testes de unidade e de integração do sistema.

3.2.1.3.1.1 Estudo de uma Agência de Medicina do Governo da Sérvia

Além de cobrir todo o ciclo de desenvolvimento de software, RUP tem sido importante para modelar processos de negócio em projetos de governo eletrônico. O estudo de Stojadinovic *et al.* [P56] forneceu um exemplo de como esta abordagem foi utilizada para modelar o processo de negócio relativo à comercialização de medicamentos na Sérvia, onde vários diagramas *Unified Modeling Language* (UML) foram utilizados para descrever o modelo do sistema.

3.2.1.3.1.2 Estudo do(a) Consiglio Nazionale delle Ricerche

Bei *et al.* [P04] concentrou seu estudo no problema da definição de um processo de desenvolvimento de software, com ênfase em ERP, para a implantação de uma fábrica de software no *Consiglio Nazionale delle Ricerche* (CNR) na Itália. Após uma investigação cuidadosa de várias abordagens de desenvolvimento de software, incluindo: processo unificado, métodos ágeis, XP e modelos prescritivos, a empresa escolheu a abordagem *Agile Unified Process* (AUP). Aspectos de capacidade de adaptação a mudanças e envolvimento

dos usuários, independência de especialistas técnicos, escalabilidade e foco em arquitetura e reutilização foram considerados na escolha de AUP. Esta abordagem tem sido utilizada em vários projetos ao longo de vários anos, os resultados têm mostrado que os softwares têm sido desenvolvidos e implantados satisfatoriamente, assim como, na opinião dos autores esta abordagem pode ser aplicada em ambientes públicos de pequeno e médio porte do governo italiano.

3.2.1.3.2 Métodos Ágeis

Foram encontrados dezessete (27,42%) estudos sobre a utilização de métodos ágeis no governo, sendo apresentados a seguir.

3.2.1.3.2.1 Estudo do(a) United States Strategic Command

O estudo de Fruhling *et al.* [P15] mostrou como métodos ágeis, mais especificamente XP, pode ser efetivamente adotado em organizações governamentais. Os autores criaram um arcabouço de lições aprendidas para auxiliar os profissionais da área de software alocados em ambientes governamentais e militares em futuras implementações de XP. Os autores enfatizaram que para adotar métodos ágeis com sucesso é preciso haver uma comunicação oportuna, precisa e completa entre todos os membros da equipe de desenvolvimento, gestores do processo de negócio e usuários do software. Outra constatação do estudo revelou que o trabalho colaborativo foi prejudicado quando membros da equipe de desenvolvimento foram alocados em vários projetos com equipes diferentes simultaneamente. Esta constatação corrobora com a opinião de Hajjdiab *et al.* [P19], onde todos os membros da equipe de desenvolvimento deveriam estar o tempo todo com a equipe e direcionar toda a sua atenção ao projeto. De acordo com ambos os estudos [P15] e [P19], esta atribuição de diversos projetos simultâneos a uma pessoa tem ocasionado interrupções no fluxo de trabalho e dificultado a inclusão de qualquer tipo de rotina diária para a equipe do projeto, assim como, tem elevado a aplicação resumida de várias práticas ágeis nos projetos. Por fim, Fruhling *et al.* [P15] sugerem que novas pesquisas sejam realizadas para entender melhor as vantagens, desvantagens e riscos na transição de abordagens prescritivas de processo para métodos ágeis em ambientes governamentais e militares.

3.2.1.3.2.2 Estudo do(a) National Institutes of Health

O estudo de Upender [P59] relatou a experiência da adoção e adaptação dos métodos ágeis XP e Scrum para um projeto de gerenciamento de dados clínicos. Em geral, a inclusão

de novas práticas de trabalho, alteração do gerenciamento de projetos e mudanças de hábitos de programação foi difícil, porém elas resultaram em uma melhor comunicação da equipe, um produto mais utilizável, e uma melhor parceria entre os usuários e a equipe de desenvolvimento. O autor acredita que as práticas de Scrum e XP são simples de entender, porém, internalizar e seguir essas práticas é rigorosamente difícil. Além disso, ele acredita que ficar ágil é tão difícil quanto se tornar ágil.

3.2.1.3.2.3 Estudo do(a) Banco Central do Brasil

O trabalho de Melo & Ferreira [P36] apresentou um estudo de caso de uma Instituição Pública Brasileira de grande porte que optou por adotar métodos ágeis após anos de experiência com métodos tradicionais de desenvolvimento. O artigo detalhou o contexto organizacional que motivou e apoiou o processo de adoção. Havia dificuldade de comunicação derivada da alta especialização em papéis. Os prazos de entrega eram longos e faltava objetividade na definição do escopo de sistemas. Os funcionários estavam desmotivados por trabalhar grande parte do tempo em tarefas burocráticas de gerenciamento das fábricas ou em atividades muito especializadas. Além disso, criou-se uma dependência das fábricas para todas as atividades relacionadas à implementação. Neste contexto, a adoção de métodos ágeis apareceu como uma proposta de solução para os diversos problemas apresentados, inclusive como uma alternativa para a solução do problema de escala de produção. Na sequência o estudo descreveu dois projetos pilotos e discutiu os resultados observados, sob perspectivas técnicas e gerenciais. O estudo revelou que a adoção de métodos ágeis em uma organização é um processo lento e complexo. Em organizações públicas, onde os processos burocráticos prezam pelo maior controle, em detrimento dos resultados mais rápidos, é particularmente mais complicado. A simples realização de alguns projetos pilotos não têm sido suficientes para tornar as práticas, valores e princípios ágeis de fato implantados. Igualmente, o estudo constatou que as principais dificuldades enfrentadas na implantação de métodos ágeis não estão relacionadas ao aprendizado das práticas ágeis e sim com a necessidade de mudança da cultura organizacional. Enquanto apenas o projeto de desenvolvimento de software pensar e agir de forma ágil e o restante da organização mantiver os vícios e culturas derivadas de processos prescritivos não será possível usufruir realmente dos benefícios ágeis. Por outro lado, os resultados após dezoito meses de implantação mostraram que a adoção teve um efeito positivo no aprendizado de novas tecnologias e na satisfação dos clientes e um discreto aumento na qualidade do código e na produtividade das equipes estudadas e foram

considerados decisivos para encorajar novas experiências com métodos ágeis dentro da organização.

3.2.1.3.2.4 Estudo do(a) Federal Bureau of Investigation

Uma boa alternativa para o problema da tomada de decisão em ambientes de governo foi encontrada no projeto *Sentinel*, onde o *Federal Bureau of Investigation* (FBI) usou métodos ágeis para transformar o desenvolvimento de um projeto mal sucedido em um programa bem sucedido [P16]. De acordo com Fulgham *et al.* [P16], a solução escolhida envolveu uma redução drástica do tamanho da equipe original do projeto de aproximadamente trezentas pessoas para quarenta e cindo pessoas, sendo quinze desenvolvedores e trinta gestores de processos de negócios, bem como, uma aproximação e contato diário entre a equipe de desenvolvimento e os gestores dos processos de negócios. As modificações realizadas diminuíram a complexidade de comunicação e reduziu o tempo para a tomada de decisão, tal como, forneceu um suporte adequado para a equipe de desenvolvimento e tornou o ambiente do projeto mais cooperativo e colaborativo. Além disso, para comunicar o andamento do projeto, reuniões de avaliação e demonstração do produto de software foram realizadas e adaptadas para públicos específicos, bem como, para garantir a qualidade do produto de software, testes em todos os níveis foram planejados e executados preferencialmente de maneira automática. Finalmente, os autores afirmaram que o uso de métodos ágeis em contratos de “escopo fixo” parece não proporcionar vantagens reais, que não foi o caso do projeto *Sentinel*.

3.2.1.3.2.5 Estudo do(a) Swedish Association of Municipalities for Joint Development of eServices

O estudo de Olsson & Rönnbäck [P44] revelou que a estratégia de desenvolvimento colaborativo entre municípios, que reúne várias competências e diferentes experiências, tem oferecido uma abordagem mais ampla para enfrentar os desafios de desenvolvimento de serviços eletrônicos mais adequados às necessidades dos cidadãos. Esta abordagem tem fornecido um cenário maior de utilização de serviços de governo que são necessários para a criação do projeto conceitual do software. Métodos ágeis têm sido adotados para melhorar o envolvimento dos municípios, onde uma compreensão forte e mútua dos processos, papéis, responsabilidades e desafios entre as partes interessadas são importantes para o sucesso do projeto. Da mesma maneira, o estudo revelou que métodos ágeis têm permitido uma maior facilidade de ajustes no escopo e direção do projeto, onde todos os requisitos não

podem ser definidos no início do projeto, mas emergem no decorrer do desenvolvimento do produto de software.

3.2.1.3.2.6 Estudo do(a) Rocky Flats Environmental Technology Site

O desejo de utilizar métodos ágeis e manter a conformidade com regulamentos contratuais pode parecer um conflito, mas não é, conforme [P01] [P16]. Métodos ágeis têm sido utilizado para relatar e atender aos requisitos de Gerenciamento do Valor Agregado (*Earned Value Management* - EVM), típico de projetos de desenvolvimento de software de governo executados com a participação de empresas contratadas da indústria de software [P01] [P16]. O estudo de Alleman *et al.* [P01] em um projeto de missão crítica de governo revelou que uma equipe de XP pode cumprir com as exigências de EVM. Isso compreendeu substituir a velocidade da equipe por métricas de valor agregado, definir o Custo Orçado do Trabalho Executado (*Budgeted Cost Of Work Performed* - BCWP) através de “requisitos testáveis”, estabelecer o Custo Orçado do Trabalho Agendado (*Budgeted Cost of Work Scheduled* - BCWS) no início de cada iteração, etc. Da mesma maneira, o estudo de Fulgham *et al.* [P16] combinou EVM com gráfico de *burn-down* e métricas ágeis de velocidade utilizando story points para calcular o custo de cada entrega.

3.2.1.3.2.7 Estudo do(a) UK Regional Government Departament e do(a) Her Majesty’s Revenue and Customs

Um clima de confiança, de cooperação, com práticas de trabalho flexíveis e colaborativas, juntamente com a tomada rápida de decisões são fatores importantes para o desenvolvimento ágil ter sucesso [P05]. Entretanto, nem todos os ambientes tem evoluído com o mesmo entusiasmo. Berger [P05] mostrou 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, que teve um impacto significativo sobre o progresso do desenvolvimento do projeto. Por exemplo, a falta de capacidade para a tomada de decisão sobre as necessidades do negócio afetou o progresso do desenvolvimento e ocasionou atrasos no cronograma do projeto, assim como, prejudicou o trabalho dos desenvolvedores que necessitavam de uma priorização das atividades para cumprir os prazos de desenvolvimento. Outra constatação foi que a falta de confiança entre todos os envolvidos no projeto (desenvolvedores, gerentes de negócio e demais partes interessadas) ocasionou

um impacto negativo sobre a capacidade de promover um ambiente de desenvolvimento colaborativo, bem como, enfraqueceu as iniciativas de integração de equipe, necessária para o desenvolvimento adaptativo. Em outras palavras, a cultura da organização influenciou o comportamento das partes interessadas, que por sua vez, influenciou o resultado da abordagem de desenvolvimento. Isso introduziu uma ameaça ao desenvolvimento ágil, onde os benefícios dos métodos ágeis foram negados quando práticas de trabalho equivocadas inerente à cultura organizacional foram levadas para o ambiente do projeto [P05]. Melo & Ferreira [P36] e Michaelson [P37] concordam com este pensamento e acrescentam que métodos ágeis tem requerido mudanças reais na cultura organizacional, incluindo aspectos políticos e processos de gestão, e não está claro que os profissionais de TI do governo se sentem suficientemente apoiados e confiantes em “desafiar” os líderes de departamentos e líderes políticos para executar o desenvolvimento de software de maneira diferente de modo a entregar serviços melhores e mais ágeis, em parte devido à falta de competência no assunto [P37]. Além disso, Michaelson [P37] revelou que no caso do projeto *Universal Credit* do Reino Unido, métodos ágeis tornou-se uma abordagem retórica, em vez de uma resposta ao fracasso de Tecnologia da Informação (TI) em projetos de grande porte do setor público. O autor acredita ser improvável que a adoção de um novo método de desenvolvimento de software e gestão de contratos seja capaz de resolver os principais problemas estruturais de projetos de grande porte do setor público, tais como: ambiente burocrático, pressão política, problemas com mudanças de negócios e implantação *big bang*.

3.2.1.3.2.8 Estudo de um Departamento de Telecomunicações e TI do Governo dos Emirados Árabes Unidos

O trabalho de Hajjdiab *et al.* [P19] apresentou 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. Os autores identificaram problemas e desafios no processo de transição de abordagens prescritivas para métodos ágeis, incluindo ausência de projeto piloto, falta de apoio da alta administração, grande pressão por resultados imediatos, falta de *coach* ágil, ambiente altamente burocrático que exigiam aprovações e assinaturas antes de passar de uma etapa para outra etapa, alocação de desenvolvedores em vários projetos de maneira simultânea e falta de preparo da equipe de desenvolvimento para lidar com métodos ágeis. Os autores apontaram que todos esses desafios poderiam ser superados ou minimizados se o processo de adoção de Scrum tivesse

sido cuidadosamente planejado, bem como, se os recursos humanos e financeiros fossem garantidos com antecedência.

3.2.1.3.2.9 Estudo do(a) United States Army

O estudo de Surdu & Parson [P57] apresentou a experiência da adoção de práticas de XP e de outros métodos ágeis, assim como, a combinação de métodos ágeis com modelos prescritivos para o desenvolvimento do projeto de simulação conhecido como *One Semi-Automated Forces* (OneSAF) *Objective System* (OOS) gerenciado pela *United States Army*. Para alcançar melhores resultados, incluindo melhoria na eficiência da comunicação e envolvimento do cliente, desenvolvedores de software do governo, desenvolvedores de software da indústria e representantes do processo de negócio se instalaram na mesma localização. Entregas incrementais e frequentes foram adotadas, bem como, métricas ágeis e documentos escritos foram adotados para manter conformidade com regulamentos contratuais de governo. Para melhorar a qualidade do produto de software, testes em todos os níveis foram planejados e executados, preferencialmente de maneira automática. Além disso, desde que o primeiro projeto ágil emergiu de maneira primordial, as empresas têm perguntado se as metodologias ágeis são compatíveis com o *Capability Maturity Model Integration* (CMMI) do *Software Engineering Institute* (SEI) [Coh11]. Uma contribuição importante para preencher essa lacuna foi encontrada neste estudo, onde uma combinação de práticas ágeis de XP com processos CMMI Nível 5 foram essenciais para o sucesso do projeto [P57].

3.2.1.3.2.10 Estudo de uma Administração Pública da Bulgária

O estudo de Iliev *et al.* [P23] mostrou que métodos ágeis tornou o processo de desenvolvimento mais aberto para o cliente. O *feedback* do cliente ao longo do desenvolvimento foi útil para melhorar o desenvolvimento das funcionalidades requeridas e agregar mais valor ao negócio. A introdução de práticas de desenvolvimento de software, testes e análise de diversas métricas levou a melhorias na execução técnica do projeto. A implementação de novas funcionalidades tornou-se algo previsível e transparente para o cliente. Por fim, os autores pretendem complementar o trabalho com informações sobre a preparação das pessoas e da organização para a adoção de métodos ágeis.

3.2.1.3.2.11 Estudo do(a) Embrapa Informática Agropecuária

O estudo de Pedroso Júnior *et al.* [P45] descreveu como as práticas de XP foram adotadas para o sucesso de um projeto crítico de governo, em resposta ao fracasso de um projeto anterior. O estudo revelou que o envolvimento do cliente tem requerido um compromisso profundo e preparação, tanto do cliente como dos desenvolvedores, para alcançar o seu pleno potencial. Propriedade coletiva do código e padrões de projeto foram adotadas sem nenhum problema. Integração contínua foi fundamental, enquanto TDD e Programação em Par não foram adotadas por falta de exemplos reais. Além disso, entregas frequentes de software foram realizadas, mas nenhuma delas foram definitivas, a equipe de desenvolvimento precisou retomar e introduzir mudanças nas versões entregues. Por fim, os autores mostraram-se satisfeitos com os resultados alcançados, e estavam ansiosos por refinar as práticas de XP em projetos futuros.

3.2.1.3.2.12 Estudo em Grandes Projetos no Setor de Defesa do Governo Americano

O estudo de McMahon [P35] compartilhou várias lições aprendidas sobre o que está funcionando e o que não está funcionando ao aplicar métodos ágeis em grandes projetos de software no setor de defesa do governo americano. O resultado mostrou que muitas das lições abordadas no estudo não são novas, e algumas podem parecer ter pouco - ou nada - a ver com métodos ágeis. Segundo o autor, o papel do Gerente de Projeto tem sido afetado pela maneira como ele/ela interage com a equipe ágil em um projeto, particularmente no que diz respeito aos requisitos e listas de tarefas. Em projetos de grande porte, por exemplo, tem sido recomendado ter várias listas de requisitos, geralmente organizadas em subprojetos, como também, que cada subprojeto possua seu próprio proprietário do produto e sua própria lista de requisitos. Da mesma maneira, os proprietários dos produtos deveriam reunir-se regularmente para coordenar mudanças entre si e acordar estratégias de integração dos subprojetos. Segundo McMahon [P35], um engenheiro de sistemas poderia ser um candidato perfeito para o papel de proprietário do produto. Entretanto, para Michaelson [P37], tem sido difícil identificar proprietários de produtos adequados e comprometidos para projetos, como também, garantir que eles estarão disponíveis para reuniões diárias com equipes de desenvolvimento. Outrossim, métodos ágeis não têm exigido menos requisitos escritos [P35].

3.2.1.3.2.13 Estudo de uma Organização do Governo dos Estados Unidos da América

O estudo de Moore [P40] descreveu uma experiência no desenvolvimento de software que aconteceu dentro de uma organização do governo dos EUA, onde um projeto de software foi submetido a mudanças organizacionais e técnicas para adaptar-se as novas tendências do desenvolvimento, incluindo métodos ágeis. O autor apresentou algumas características onde o projeto evoluiu. Para melhorar a detecção de defeitos no sistema, testes unitários foram criados e sua execução foi automatizada. Na sequência, o processo de compilação, integração dos componentes e disponibilização do sistema foi automatizado. Por fim, para garantir a qualidade do software, um conjunto de métricas foi adotado. O resultado do esforço permitiu que a equipe do projeto adaptasse as mudanças, bem como, entregasse o produto de software para o cliente mais de uma vez por semana.

3.2.1.3.2.14 Estudo do(a) Prefeitura de Kyoto

A importância de métodos ágeis tem aumentado no Japão [P24]. Entretanto, o ciclo *Plan-Do-Check-Act* (PDCA) requer uma visão repetida do software, que não é aceita pelo sistema de orçamento do governo japonês, por que, ciclos repetitivos geram despesas repetidas, que por sua vez, significam mau uso do dinheiro público. Uma solução para este problema foi encontrada no estudo de Kaneda [P24]. Neste trabalho, o autor mostrou uma combinação das abordagens *Project Based Learning* (PBL) e métodos ágeis. Estas abordagens foram utilizadas pela prefeitura de Kyoto e a universidade pública de Doshisha no Japão que desenvolveram dois softwares sem a ajuda de empresas da indústria de software. O resultado desta estratégia foi que os alunos e os professores foram beneficiados com o conhecimento sobre desenvolvimento de software com métodos ágeis para o mundo real e ambos tiveram uma maior compreensão da importância da ES. Da mesma maneira, esta abordagem permitiu que o governo ajustasse os softwares desenvolvidos o quanto necessário, sem um aumento de custo. Outrossim, a execução do ciclo PDCA, importante para as atividades diárias da prefeitura de Kyoto, foi mantida.

3.2.1.3.2.15 Estudo do(a) NASA Langley Research Center

Kent Beck, um dos proponentes do manifesto ágil, enumerou nove ambientes que, segundo ele, XP não deveria ser utilizado [Bec99]. Seis destas nove características estavam presentes em um projeto piloto de adoção de XP na *NASA Langley Research Center* (Langley) [P60]. Na medida em que o projeto começou, William A. Wood & William L. Kleb [P60], tiveram que lidar com vários conflitos culturais antes de implementar XP. Em primeiro

lugar, o conflito entre o valor de negócio de curto prazo e os objetivos de pesquisa de longo prazo levou a um choque cultural ao aplicar práticas básicas de XP em projetos de pesquisa científica. O estabelecimento de ciclos de desenvolvimento incremental com *feedback* em poucas semanas parecia ser incoerente com o papel do centro de pesquisa de buscar soluções ao longo prazo através de projetos revolucionários.

Em segundo, a política interna da empresa atribuía dois terços do tempo do projeto para as atividades de levantamento de requisitos, documentação e projeto. A atividade de codificação deveria ser realizada em apenas um terço do tempo restante do projeto, não sendo permitida sua execução desde o início do projeto. Em terceiro, as políticas internas da empresa incentivavam o uso de grandes especificações para o planejamento e desenvolvimento de software, com pouca menção ao teste de software (por exemplo). O não cumprimento desta prática poderia penalizar o projeto por falta de garantia de qualidade e inconformidade com os processos institucionais. Em quarto, os pesquisadores eram reconhecidos e valorizados por desempenho individual ao invés de trabalho em equipe. Em quinto, em centros de pesquisa, os pesquisadores geralmente são alocados em salas individuais, que poderiam estar localizadas em outros prédios dentro do próprio campus da empresa. Esta característica de ambiente não favorecia à comunicação efetiva das pessoas envolvidas no projeto. Por último, as equipes eram formadas por duas pessoas, manter os papéis distintos de programador, cliente, mediador, *coach* para equipes pequenas era um problema. No contexto da pesquisa de longo prazo, as tecnologias que estão sendo exploradas podem ser imaturas ou incertas, como também, estarem a anos de distância de alcançarem seu potencial comercial. Nesta situação, o financiador é muito distante da pesquisa para servir como um cliente adequado. Assim, o pesquisador é essencialmente o "cliente" para o seu próprio esforço de desenvolvimento, pelo menos por vários anos.

As soluções encontradas pelos autores para resolver os conflitos culturais apresentados anteriormente incluíram:

- Inicialmente, os autores não sabiam se poderiam reformular as metas de pesquisa de longo prazo em pequenos incrementos tangíveis, adequados aos ciclos de iteração do XP de duas a três semanas. Felizmente, na prática, embora a escala de tempo de *feedback* de investigação continue grande, os autores conseguiram decompor com sucesso as características técnicas em pequenos pedaços de iteração, seguindo a prática do *Design* Simples do XP.

- As políticas internas de desenvolvimento de software do centro de pesquisa não foram seguidas;
- Os participantes suprimiram o desejo de serem reconhecidos individualmente por acreditar que duas pessoas fazendo XP seria mais produtivo do que a soma dos esforços individuais;
- O *layout* da sala dos pesquisadores envolvidos no projeto foi alterado para suportar a Programação em Par, como também, acomodar a presença de colaboradores de outras instituições, por exemplo: professores universitários;
- A Programação em Par foi executada exclusivamente durante a construção de testes e refatoração do código;
- O pesquisador com mais interesse no uso do software assumiu o papel de cliente durante a prática do Jogo de Planejamento.

Os pesquisadores se esforçaram em aplicar as doze práticas de XP no projeto piloto descritas no livro [Bec99]. Devido aos desafios culturais, oito das doze práticas apresentaram desafios de implementação. No entanto, os resultados indicaram que a abordagem XP para o desenvolvimento de software foi aproximadamente duas vezes mais produtiva do que projetos semelhantes executados anteriormente. Além disso, a base de código funcional foi cerca de metade do número de linhas de código em relação ao esperado, e a legibilidade do código foi melhor. Refatoração, conforme previsto pelo XP, são em grande parte responsáveis pela melhoria da estética do código fonte. A partir dos resultados alcançados com a execução do projeto piloto, o centro de pesquisa *Langley* decidiu ampliar o uso de XP para outro projeto de missão crítica, conhecido como "*High-Energy Flow Solver Synthesis*".

3.2.1.3.2.16 Estudo do(a) Israeli Air Force

O estudo de Dubinsky *et al*. [P13] descreveu uma pesquisa realizada com uma equipe de desenvolvimento de software da *Israeli Air Force*. Com o objetivo de reduzir o tempo de entrega do software, melhorar a comunicação e a colaboração com o cliente, a equipe optou pela adoção de XP para o desenvolvimento de um projeto de software de missão crítica. Dentre os vários temas pesquisados, o estudo focou em métricas ágeis. As métricas ágeis foram adotadas e aperfeiçoadas ao longo do projeto. O resultado foi um aumento da confiança dos membros da equipe de desenvolvimento e dos gestores administrativos, onde a adoção de métricas ágeis objetivas contribuiu para verificar o cumprimento de prazos mais cedo, assim como, permitiu que a tomada de decisão acontecesse de maneira precisa. Esta

constatação comprova a opinião de McMahon [P35], que diz o seguinte: ”Um dos maiores benefícios de métodos ágeis tem sido a sua capacidade em promover a visibilidade do projeto para os gestores mais cedo”. Além disso, os gestores administrativos apontaram que as métricas estudadas poderiam ajudar a escalar o XP, por exemplo, para gerenciar várias equipes de desenvolvimento em projetos de larga escala. Em trabalhos futuros, os autores pretendem investigar a escalabilidade das métricas adotadas de modo a promover a melhoria contínua das métricas usadas.

Vale ressaltar que os princípios e várias práticas de XP contradiziam e pareciam ser incompatíveis com muitos dos processos de trabalho existentes. A solução encontrada pela gerência administrativa foi uma renúncia da maioria dos regulamentos existentes. Isso permitiu que a equipe executasse seus próprios processos de trabalho, desde que demonstrasse para os níveis superiores que o projeto estava sendo gerenciado e estava efetivamente sob controle, bem como, a qualidade e o atendimento das necessidades dos clientes não estavam sendo comprometidas. O resultado foi que após a primeira iteração os gestores ficaram surpresos ao ver algo em execução e todos concordaram que “a pressão para entregar algo a cada duas semanas levou a resultados surpreendentes”.

3.2.1.3.3 Rapid Application Development

Rapid Application Development (RAD) foi aplicado em um projeto de larga escala para resolver problemas frequentes do desenvolvimento de software em uma AP do Reino Unido, incluindo entregas com atraso e que nem sempre atendiam às necessidades do cliente, assim como, uma crescente incapacidade de responder as novas demandas [P06]. O resultado foi que as partes interessadas encontraram dificuldades em ajustarem-se as exigências do RAD (incluindo comprometimento e disponibilidade) e adotaram uma postura passiva, o que dificultou a tomada de decisão críticas de negócio. A cultura organizacional estabeleceu que a tomada de decisão deveria ser realizada no ponto apropriado da hierarquia organizacional. Isto fez com que os gerentes de negócios se mantivessem relutantes em tomar decisões importantes sobre processos de negócios que não se sentiam responsáveis, apesar de estarem habilitados a fazê-lo. Tais dificuldades impactaram sobre a trajetória do projeto, causando atrasos no cumprimento de prazos importantes do projeto. Este resultado fortalece a ideia de que quando uma cultura organizacional enfatiza uma “cultura de culpa”, muito dos benefícios do desenvolvimento adaptativo podem ser negados, tal como, um nível de conflito e uma falta de confiança pode ocorrer entre desenvolvedores e gestores de negócios [P05]. Porém, evidências sugerem que esses problemas tendem a

diminuir ao longo do tempo devido à evolução e familiarização da abordagem do desenvolvimento dentro da organização [P06].

3.2.1.3.4 Microsoft Solutions Framework

O estudo de Bygstad [P09] mostrou que aplicar casos de uso, *design* de interfaces, técnicas de programação e testes foi difícil com muitos participantes inexperientes. O resultado foi que o custo foi maior do que o estimado, a produtividade e a qualidade foram inferiores, e a colaboração entre os gestores do processo de negócio e os desenvolvedores não foi suficiente para alcançar os melhores resultados. Por outro lado, o estudo identificou uma janela de oportunidade para a adaptação mútua. Em um projeto de desenvolvimento de SI, a flexibilidade do SI é alta no início do projeto, em seguida, ela gradualmente se torna menor à medida que mais componentes de software são produzidos e introduzidos no sistema. Por outro lado, o processo de negócio tem um baixo grau de flexibilidade no início, em seguida, ela aumenta lentamente por conta dos primeiros resultados do projeto. Neste ponto do tempo, a janela de oportunidade se abre, onde a flexibilidade tanto do SI quanto do processo de negócio é suficiente para uma mudança mútua. Isto significa que a flexibilidade do SI e do processo de negócio variam ao longo do tempo. No início do projeto, tem sido relativamente fácil alterar os requisitos, enquanto que no final do projeto, tem sido muito mais difícil e mais caro. Uma outra constatação do estudo foi que, o envolvimento de desenvolvedores e representantes do processo de negócio em *workshops* e sessões de prototipagem permitiu que os desenvolvedores tivessem um *feedback* mais detalhado do processo de negócio, enquanto os representantes dos processos de negócio aprenderam mais sobre as complexidades e limitações da tecnologia.

3.2.1.3.5 V-Modell XT

A adoção de V-Modell e suas variantes tem sido obrigatória para projetos de desenvolvimento de software no governo da Alemanha [P27] [P28]. O estudo de Kuhrmann *et al.* [P27] apresentou os primeiros resultados da utilização da nova variante V-Modell, conhecida como XT, em um projeto piloto do *German Department of the Interior*. Uma das constatações foi que a nova abordagem V-Modell XT foi muito mais simples de se utilizar do que a abordagem antiga V-Modell 97. Além do mais, o principal benefício identificado foi que V-Modell XT tem permitido uma adaptação do processo para contextos específicos, onde somente os artefatos necessários podem ser criados e atividades desnecessárias podem ser ignoradas. Logo, os autores acreditam que esta capacidade permitirá que projetos

construídos a partir do V-Modell XT tenham uma versão personalizada que atenda suas necessidades específicas da melhor maneira possível.

3.2.1.4 Outros Modelos de Processo

Nesta dimensão foram incluídos outros modelos, processos e métodos de ES que não foram classificados facilmente nas categorias de modelos prescritos e adaptativos de processo de desenvolvimento de software.

3.2.1.4.1 Team Software Process

O estudo de Humphrey [P22] apontou que muito dos problemas do desenvolvimento de software tem sido causado por modelos tradicionais de gestão, onde os gestores não sabem realmente o que devem mudar para melhorar seus resultados e são relutantes em mudar para um novo método de gestão que não é de uso geral por outros projetos similares. Além disso, com o trabalho do conhecimento realizado em grande escala, os gerentes não têm conseguido visualizar os pequenos problemas diários e os desenvolvedores não tem obtido os dados necessários para descrevê-los. Como resultado, os gerentes não têm conseguido tomar medidas para recuperar os atrasos diários. No momento em que os atrasos de agendamento têm se tornado grande o suficiente para serem visíveis, tem sido tarde demais para se fazer qualquer coisa sobre eles. Segundo o autor, este é o motivo pelo qual os projetos executados por gestores competentes e experientes continuam a ter problemas de custo, cronograma e qualidade. Os gestores não têm obtido o *feedback* que eles precisam para tentar impedir que os problemas ocorram a tempo. Outrossim, em projetos grandes, com muitas dependências entre si, atrasos em qualquer parte tem afetado muitas outras. Isto significa que várias partes de um grande projeto provavelmente irão entrar em problema de programação ao mesmo tempo.

A solução encontrada pela organização foi a utilização da metodologia *Team Software Process* (TSP), onde os desenvolvedores têm se reunido e utilizado os dados coletados para gerenciar seu próprio trabalho, assim como, tem medido com precisão o andamento do projeto no intervalo de um dia. Equipes TSP tem relatado o *status* do projeto semanalmente, e a gestão tem visto exatamente o que está acontecendo. De acordo com o autor, com informações de *status* precisas, a administração do projeto tem visto pequenos problemas de custo e cronograma, antes que eles se tornem sérios. Desta forma, a gerência pode tomar medidas cabíveis para identificar causas e resolver problemas a tempo.

Depois de utilizar o TSP por vários anos, a organização informou que os seus níveis de qualidade do produto melhoraram em cerca de dez vezes e que os tempos de teste foram reduzidos de meses para semanas. Cronograma e custo tem sido muito mais previsível do que antes, e as reuniões de *status* semanais que aconteciam as segunda-feira, quarta-feira, e sexta-feira não foram mais necessárias. Cooperação e coordenação da equipe também foram melhoradas. A conclusão final da organização foi a de que, "Esta é a única maneira de gerenciar grandes projetos de conhecimento de trabalho".

Com relação às equipes de desenvolvimento, elas têm se esforçado para cumprir as metas do cronograma da gerência do projeto, tal como, elas têm negociado seus próprios compromissos com a gerência. As equipes têm se sentido responsável pelo controle do seu trabalho, bem como, elas têm conseguido informar o *status* do projeto, e tem obtido informações adequadas para defender suas estimativas. Segundo Humphrey [P22], as equipes deveriam ser treinadas em métodos de gerenciamento de equipe. Elas deveriam ser responsabilizadas por produzir os seus próprios planos, negociar os seus próprios compromissos e cumprir esses compromissos com produtos de qualidade. Isto significa que o papel do gerente não tem sido simplesmente gerenciar equipes de trabalho, mas sim, liderá-las, motivá-las, apoiá-las e orientá-las [P22].

3.2.1.4.2 Computer-Aided Software Engineering

O estudo de Love & Siddiqi [P30] investigou a hipótese de que a introdução de uma ferramenta *Computer-Aided Software Engineering* (CASE) renderia processos melhorados compatíveis com o CMM Nível 2 em um Departamento de Sistemas de Informação (DSI) do governo do Reino Unido. A investigação mostrou que a combinação de uma ferramenta CASE com mudanças em práticas de trabalho foi suficiente para o DSI alcançar CMM Nível 2, e que muitas das melhorias realizadas antecederam à adoção da ferramenta.

Em geral, o uso da ferramenta CASE contribuiu para melhorar o processo de desenvolvimento de software, incluindo protótipo de telas para especificação de requisitos com usuários, planejamento e acompanhamento de projetos, gerencia de configuração e garantia da qualidade por meio de relatórios de inconsistência de itens que estão nos repositórios.

3.2.1.4.3 Open-source Software Development Models

O estudo de Kim & Teo [P25] sobre a utilização do modelo de desenvolvimento baseado em software livre para a construção do *framework* de governo eletrônico

eGovFrame, revelou que o desenvolvimento de aplicações baseada em padrões abertos melhorou a qualidade do software desenvolvido, tal como, tornou o desenvolvimento mais eficiente e com uma maior reutilização do software. A abordagem utilizada foi articulada na forma de uma estratégia de inovação aberta, que englobou código aberto, licença de software livre, processos abertos, saídas abertas e um ecossistema aberto, assim como, uma parceria entre agências governamentais e empresas da indústria de software. O resultado mostrou que empresas públicas e privadas se beneficiaram com o compartilhamento ou venda de produtos comerciais oriundos do eGovFrame. Além disso, os autores apontaram seis lições aprendidas sobre o modelo de desenvolvimento baseado em ecossistemas abertos:

- Todas as partes interessadas devem ser envolvidas no desenvolvimento, bem como, os benefícios do trabalho colaborativo e o compartilhamento de informações deve ser enfatizado para todas as partes.
- A inovação aberta é essencial para a construção da estrutura do ecossistema de desenvolvimento. O desafio desta abordagem consiste em convencer os participantes a compartilhar os seus conhecimentos, uma vez que eles estão preocupados com a possibilidade do proprietário da plataforma ou outros participantes poderem replicar sua tecnologia e começarem a oferecer um produto concorrente.
- Mesmo que o código-fonte esteja disponível abertamente, somente pessoas autorizadas podem realizar as mudanças necessárias. Isto significa que uma estratégica de governança deve ser estabelecida e colocada em prática.
- A construção do *framework* e a disponibilização de aplicações devem ocorrer de maneira simultânea. Quanto mais aplicativos estiverem disponíveis na plataforma, mais clientes estarão dispostos a reconhecer o *framework* como uma fonte de aplicações adequadas.
- Os governos são geralmente relutantes em abrir seus processos decisórios internos para o público. No entanto, a experiência do projeto eGovFrame ilustrou como a relação de confiança entre o governo e seus cidadãos contribuiu para uma colaboração eficiente entre o governo e o público, assim como, contribuiu para a construção de uma mentalidade coesa e comum para o ecossistema.

- O governo da Coréia do Sul reconheceu que mudanças tecnológicas certamente iriam impactar o eGovFrame. Desta maneira, estratégias de melhoria contínua e atualização tecnológica foram estabelecidas.

3.2.1.4.4 Model-Based Development

O estudo de Pilemalm *et al.* [P46] explorou as experiências práticas de desenvolvimento e implementação de *model-based* em quatro projetos de desenvolvimento de software nas Forças Armadas da Suécia. Uma das constatações foi que os problemas vivenciados no desenvolvimento *model-based* não se desviaram dos problemas gerais de desenvolvimento de software. Um conhecimento sólido em engenharia de requisitos e uma participação representativa das partes interessadas com conhecimento do domínio da aplicação foram destacados como cruciais para o desenvolvimento de sistemas aderentes as necessidades dos usuários. Uma outra constatação mostrou que a implementação de *model-based* como uma abordagem madura de desenvolvimento de software irá levar mais tempo. Nos projetos estudados, não foi encontrado nenhum benefício real, isto é, o que foi implementado é de uma escala muito menor do que o planejado inicialmente. Isto significa que existe uma diferença clara entre o efeito teórico e as situações reais vivenciadas nos projetos estudados.

3.2.1.4.5 Desenvolvimento Centrado no Usuário

O estudo de Catarci *et al.* [P10] analisou aspectos de usabilidade e desenvolvimento centrado no usuário em dois projetos da AP da Itália executados por empresas contratadas. Em primeiro lugar, o problema da representatividade dos usuários envolvidos no projeto surgiu de uma maneira significativa. A questão era como muitos usuários e grupos de usuários deveriam ser envolvidos no ciclo de desenvolvimento e em que fases. As funcionalidades dos sistemas analisados pareciam ser construídas a partir de informações fornecidas por um único usuário, em um dos casos analisados, e por dois usuários, no outro caso.

De um modo geral, as pessoas que não eram os usuários finais do software encomendaram os sistemas, e atuavam em nome de todos os usuários finais do projeto. Os testes e as avaliações realizadas ao longo do desenvolvimento dos softwares não tiveram a participação dos usuários que iriam utilizar os sistemas. Em segundo lugar, os contratos de desenvolvimento de software não envolviam adequadamente os usuários e, geralmente, não consideravam fatores de usabilidade e satisfação do usuário como critérios de contrato com

a indústria. Por fim, usabilidade representou um custo adicional a ser pago a indústria, em termos da diminuição da taxa de produção de software, aumento do controle do produto e aumento da curva de aprendizado, contra benefícios que não eram fáceis de avaliar. O resultado da pouca participação dos usuários finais ao longo do desenvolvimento de software foi que a solução escolhida foi difícil de assimilar e alternativas comerciais eram mais fáceis de utilizar. Outra constatação mostrou que a indústria possui muito pouco conhecimento sobre métodos de desenvolvimento de software centrado no usuário e, em particular, dão pouca atenção ao contexto de utilização dos produtos de software que desenvolvem. Segundo os autores, isto significa que características de usabilidade de produtos de software são praticamente ignoradas pela indústria de software.

3.2.1.5 Algumas Comparações com o Setor Privado

Três estudos comparativos sobre o desenvolvimento de software no setor público e privado foram encontrados [P17] [P26] [P61]. O estudo de Krogstie [P26] mostrou que o tempo de desenvolvimento de novos produtos de software e melhoria de sistemas em produção é relativamente igual em todo o setor público e privado. Isto significa que baseado somente neste estudo não se poderia afirmar que o desenvolvimento e manutenção de sistemas é mais eficiente no setor privado do que no setor público. Porém, foi encontrado dois estudos que contrapõem estes resultados e apontam que o desempenho de projetos de desenvolvimento de software no setor público é pior do que o setor privado [P17] [P61].

O estudo de Furumo *et al.* [P17] mostrou que há indícios de que projetos em organizações do setor público são mais propensos de serem entregues com atraso do que projetos organizados pelo setor privado. Segundo os autores, esta constatação pode estar associada a uma maior coordenação e níveis adicionais de aprovação por conta do aumento do número de clientes e partes interessadas. O estudo sugere que, no mínimo, contingências de tempo deveriam ser construídas para projetos do setor público. Por outro lado, não há evidências que equipes de projetos em organizações do setor público são mais propensas a estourar o orçamento do que equipes em organizações do setor privado.

Enquanto o estudo de Wright & Capps [P61] mostrou que em ambos os setores, a maioria dos grandes projetos ultrapassaram seus orçamentos e cronogramas originais em mais de 50%, com muito mais intensidade no setor público. Outrossim, o estudo apontou que falhas tem ocorrido com frequência, e evidências empíricas mostraram que elas ocorreram com mais frequência no governo.

3.2.2 Práticas de Desenvolvimento de Software na Administração Pública

O *Software Engineering Body of Knowledge* (SWEBOK) [BoF14], organizado pela *IEEE Computer Society*, define o corpo de conhecimentos usualmente aceito relacionado à Engenharia de Software. Ele relacionada dez áreas do conhecimento, das quais modelos, processos e métodos de ES é o foco dessa pesquisa. Entretanto, o desenvolvimento de software depende de outras práticas de ES para alcançar seu pleno potencial. Neste sentido, secundariamente, este trabalho apresenta uma lista de outras práticas de ES usadas para o desenvolvimento de software no setor público. Vale ressaltar que a lista de práticas e estudos recuperados não é exaustiva e nem conclusiva, como também, não foi necessariamente produzida em uma ordem esperada, uma vez que a estratégia de busca não foi construída para essa finalidade.

3.2.2.1 Gerenciamento de Projetos

Nenhum estudo foi encontrado com foco em gerenciamento de projetos de produtos de software no setor público. Entretanto, a partir de algumas evidências encontradas, sugere-se que o planejamento e execução de um projeto de desenvolvimento de software no setor público tem acontecido com base em duas abordagens distintas:

1. Envolvendo um planejamento de projeto de longa duração e mais detalhado;
2. Envolvendo um planejamento com base em iterações de curta duração e menos detalhado.

O exemplo do projeto *Sentinel* executado pelo *Federal Bureau of Investigation* (FBI) [P16] contribui para materializar essa constatação. Na primeira fase do projeto, empresas da indústria de software foram contratadas. Por conta disso, um planejamento de projeto de longa duração e mais detalhado foi necessário. O resultado foi um produto de software entregue em anos em vez de meses, com pouca ou nenhuma contribuição aos objetivos de negócios. 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 engenheiros de software do governo em todas as atividades do desenvolvimento de software, bem como, uma mudança na estratégia de gerenciamento do projeto. O FBI assumiu literalmente a liderança do projeto e adotou o método Scrum para ajuda-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), assim como, incluiu o desenvolvimento incremental em tempos curtos de entrega com maior *feedback* e envolvimento dos usuários. O resultado foi que o FBI conseguiu construir com somente 52% do orçamento, 88% do sistema em apenas um ano, contra um histórico de dez anos de atraso e desperdícios do dinheiro público.

3.2.2.2 Gerenciamento de Risco

O estudo de Dey *et al.* [P11] sobre gerenciamento de risco em projetos de desenvolvimento de software identificou que a indisponibilidade e dificuldade de contato com clientes, assim como, a perda de pessoas importantes durante a execução de projetos de software tem comprometido o andamento do projeto. Outra constatação foi que requisitos incompletos ou incorretos quando encaminhados para desenvolvimento tem ocasionado tempos adicionais para realizar as modificações necessárias. Os autores mostraram que a aplicação eficiente do gerenciamento de riscos tem garantido uma execução bem sucedida de projetos, incluindo uma maior satisfação dos clientes e um melhor desempenho financeiro das organizações públicas. As recomendações têm requerido o envolvimento das partes interessadas, bem como, a integração do gerenciamento de risco ao processo de tomada de decisão do gerenciamento de projetos.

Modelos prescritivos predominaram nos projetos estudados por Bannerman [P03], uma das constatações foi que a concepção de requisitos estáveis e fixos em ambientes incertos e mutáveis tem sido uma importante fonte de risco de projeto em si mesmo. Nesta direção, o estudo de Pyster [P50] relatou que uma relação tensa entre o governo e a indústria de software tem ocorrido por conta de alterações de requisitos.

Outra constatação do estudo de Bannerman [P03] mostrou que vários projetos descobriram que a contratação de empresas da indústria de software poderia ser uma alternativa, ou então, um risco significativo para o projeto, dependendo de como elas seriam gerenciadas. A abordagem de contratação utilizada era orientada por planos com especificação formalmente estruturada, mas não tinham as condições necessárias para lidar com a incerteza estratégica. Algumas agências governamentais aprenderam ao longo do tempo que os melhores resultados foram alcançados com a equipe contratada localizada no próprio ambiente do governo. Esta integração física melhorou muito a comunicação do projeto, interação, resolução de problemas e acompanhamento do progresso, bem como, permitiu estabelecer um ciclo de desenvolvimento incremental, o que resultou na entrega de um sistema mais próximo das necessidades das agências governamentais. Outras agências descobriram que ao invés de ser uma fonte de ameaça para o projeto, a empresa contratada

contribuiu positivamente para o projeto através de sua experiência e competência, o que compensou a falta de experiência interna do governo. Esta constatação vai de encontro aos resultados do estudo [P16], onde a utilização da abordagem de desenvolvimento *in-house* trouxe melhores resultados para o projeto.

O trabalho de Zhou *et al.* [P62] apresentou um estudo sobre gerenciamento e estratégias de mitigação de risco em projetos de desenvolvimento de SI no setor público. As principais constatações envolveram questões de cliente, de gerenciamento de projetos e tecnológicas. Na dimensão cliente, foi constatado que projetos com maior possibilidade de provocar uma mudança significativa na estrutura e cultura organizacional tem maiores chances de enfrentar uma forte resistência dos usuários e causar uma série de riscos para o projeto. Isto significa que representantes do cliente tem um papel muito importante na mediação e negociação de mudanças, bem como, na preparação da organização para aceitar o novo sistema, incluindo treinamento e migração de usuários finais para o novo sistema [P50]. Outrossim, conflitos entre departamentos de usuários e dificuldades políticas internas têm criado dificuldades para a execução das atividades de desenvolvimento de software, assim como, para aceitação final do sistema [P62].

Enquanto na dimensão de gerenciamento de projetos, o estudo [P62] identificou que a maioria dos gerentes do setor público falharam ao relatar e documentar o andamento do projeto. Os autores acreditam que esta constatação pode ter sido provocada por uma cultura prevalente do setor público, ou apenas por uma falta de treinamento específico nas tarefas de gerenciamento de projetos de TI.

Com relação às questões tecnológicas, a pesquisa [P62] mostrou que as tecnologias não comprovadas ou não familiares utilizadas em projetos de desenvolvimento tem causado decepção e expectativas irreais, assim como, tem provocado baixo desempenho. Outrossim, a combinação da falta de informação para a tomada de decisão em relação as tecnologias com a pressão por melhores resultados têm causado efeitos negativos no desenvolvimento de software. Por fim, o estudo revelou que a falta de consulta ao usuário e a falta de uma consciência holística das necessidades organizacionais tem sido um dos maiores fatores de falha na maioria dos casos estudados.

O estudo de Goldfinch [P18] mostrou que em muitos casos, o setor público não tem a competência, recurso e pessoas adequadas para desenvolver, conduzir e monitorar projetos de desenvolvimento de software com empresas da indústria. Governos com níveis mais elevados de contratação têm sido menos propensos a terem projetos entregues no prazo e dentro do orçamento previsto. Em vários países, principalmente os países em

desenvolvimento, a falta de capacidade do governo em atuar com o desenvolvimento de software tem introduzido um risco de dependência e exploração de empresas terceirizadas, que torna a indústria de software mais forte e ágil, e comparativamente o governo menos forte e frágil.

3.2.2.3 Gerenciamento de Requisitos

No estudo de Martin & Gregor [P43] a prática de engenharia de requisitos foi examinada no *Australian Bureau of Statistic* (ABS). Uma das principais lições aprendidas é que a priorização de requisitos tem sido sobre uma escolha entre as necessidades de negócio e as necessidades técnicas, e que uma especificação pode não estar completa em todos os seus aspectos. Além disso, os grupos gestores tem assegurado que um grupo substancial de usuários tenham sido consultados para garantir que os requisitos de negócios sejam identificados, extraídos e priorizados. Esta abordagem tem sugerido que os gestores dos processos de negócio assumam responsabilidades conjuntas durante o desenvolvimento de software, como também, tem feito com que os usuários finais participem das atividades de testes de software.

3.2.2.4 Estimativas de Esforço

O estudo de Hamdam *et al*. [P20] revelou que aspectos da cultura organizacional e fatores de liderança afetaram de diversas maneiras o esforço necessário para a conclusão de projetos de software em mais de vinte organizações governamentais dos Emirados Árabes Unidos (EUA). A experiência da equipe de desenvolvimento juntamente com a interação dos membros da equipe foram os fatores mais impactantes nas estimativas de esforço e custo.

O estudo de Morgenshtern *et al.* [P41] foi realizado para identificar os principais fatores que tem afetado as estimativas de esforço e cronograma em um departamento de TI de uma grande organização pública do governo de Israel. O resultado do trabalho mostrou que projetos com maior nível de incerteza e mais complexos foram os mais suscetíveis de incorrer erros de estimativa de esforço e cronograma. Além disso, o resultado da análise revelou que erros de estimativa de esforço e cronograma aumentam à medida que o nível de incerteza do projeto também aumenta. Por fim, os resultados têm apoiado a ideia de que o planejamento de atividades mais curtas com tarefas menores contribui para a melhoria da precisão das estimativas.

3.2.2.5 Verificação & Validação, Teste e Qualidade de Software

O estudo de Arthur & Gröner [P02] mostrou que verificação e validação de software (V&V) tem sido importante para o desenvolvimento de software dentro do prazo, do orçamento, do escopo e com a qualidade requerida. Neste artigo, a eficiência da metodologia *Software Engineering Evaluation System* (SEES) sobre o processo de desenvolvimento de software foi examinada. O resultado revelou que a abordagem permitiu detectar defeitos logo no início do processo de desenvolvimento, o que reduziu o tempo necessário para corrigi-los. Por outro lado, os autores identificaram características que impediram o uso eficiente da abordagem SEES e propuseram um conjunto de recomendações para melhoria do processo, incluindo automatização do processo, redução da redundância de atividades e execução do processo sobre requisitos “maduros”. Além disso, a adesão da abordagem SEES foi difícil por conta da natureza abrangente e complexa de metodologias V&V.

O estudo de Driskell *et al.* [P12] apresentou os resultados da aplicação da abordagem *Independent Verification and Validation* (IV&V) em conjunto com a análise de risco para avaliar a adequação dos requisitos de segurança do software no início do desenvolvimento. O resultado mostrou que a partir de casos de uso de alto nível do sistema, diagramas de transição de estado e diagramas de atividade foi possível identificar uma lista de falhas críticas de segurança e fornecer informações importantes para testes específicos de segurança do software, incluindo entradas para testes de software e casos de teste de segurança.

O estudo comparativo de [P54] sobre três diferentes abordagens de testes de software utilizadas no *Software Engineering Laboratory* (SEL) revelou que testes de aceitação tem sido um processo consistente para medir a qualidade do software antes da liberação para o cliente. Os sistemas que tem realizado testes de aceitação tem demonstrado um alto nível de qualidade em suas operações, além disso, os testes escritos por desenvolvedores não têm sido tão eficientes na detecção de erros quando comparados aos testes baseados em cenários operacionais. Esta constatação foi confirmada pelos estudos [P47] e [P48], onde testes de aceitação realizados com os usuários reduziu drasticamente o número de defeitos encontrados após a liberação do software no ambiente de produção. Outra constatação foi que a implementação em pequenos incrementos de software acelerou o processo de execução dos testes de aceitação, e permitiu que os desenvolvedores tivessem um *feedback* mais rápido sobre a qualidade do software produzido.

O estudo de Monteiro *et al.* [P39] relatou a experiência da definição e implantação de um processo de testes na Empresa de Processamento de Dados do Estado do Pará (PRODEPA). Inicialmente, não existia nenhuma padronização a ser seguida durante a execução das atividades de teste e a importância dada a fase de testes (em termos de esforço e custo) era muito baixa se comparada com as demais (análise, projeto e construção). O resultado foi que houve um aumento na qualidade dos projetos no âmbito de achados de defeitos, além disso, a equipe de teste adquiriu uma visão mais crítica na execução e avaliação dos sistemas para um maior controle de defeitos. Do ponto de vista técnico, as principais lições aprendidas foram:

- Os artefatos de entrada para a produção dos casos de testes deveriam ser bem definidos e mantidos;
- Os casos de teste deveriam ter relação com o registro gerado na ferramenta de gestão de *bugs*, para manter a consistência;
- Os cenários desenvolvidos baseados em telas do sistema auxiliam na confecção dos casos de teste;
- A elaboração dos casos de teste deveria ser realizada antes da finalização do desenvolvimento, o que agiliza o processo de testes;
- A elaboração do projeto de interfaces facilitou a construção dos casos de teste;
- A comunicação entre os membros do projeto deveria ser mantida durante todo o ciclo de vida do projeto, principalmente em caso de modificações.

O estudo de Smith [P53] sobre custos de qualidade de software aplicado a um projeto de sistema operacional mostrou que as tarefas relacionadas com alta garantia da qualidade adicionaram um montante de 26% aos custos de desenvolvimento de software, assim como, o trabalho necessário para lidar com a alta garantia foi traduzido em tempo adicional do projeto também. Por outro lado, as tarefas de alta garantia da qualidade, que utilizaram métodos formais foram responsáveis por identificar 68% das falhas de segurança no sistema. Porém, apenas 14% de todas as falhas descobertas eram do tipo que poderiam ser detectadas através de métodos formais, e que o trabalho da equipe de garantia formal, representava apenas 19% de todas as falhas detectadas.

O estudo de Fairley & Bush [P14] sobre a utilização de práticas de garantia da qualidade, segurança, confiabilidade e eficiência do software revelou que garantia e qualidade de produtos de software melhorou quando profissionais da área atuaram como membros da equipe do projeto. Especialistas em garantia de software atuaram como

facilitadores e forneceram apoio em vários aspectos de qualidade de software, produtividade e garantia do produto. De acordo com os autores, esta estratégia foi adequada e evitou conflitos entre desenvolvedores, gerentes de projeto e especialistas em garantia da qualidade de software, uma vez que desenvolvedores poderiam interpretar o envolvimento de profissionais de qualidade como uma falta de confiança na sua competência de realizar o seu trabalho. O principal desafio encontrado foi determinar quais eram os artefatos e as métricas que realmente seriam úteis para os gerentes de projeto e desenvolvedores melhorarem a qualidade de seus produtos de software.

O estudo de Rosenberg & Sheppard [P52] mostrou que o desenvolvimento de software apoiado por processos de medição foi um sucesso. Por exemplo, os indicadores foram úteis para determinar se as técnicas de desenvolvimento de software utilizadas contribuíram realmente para a produção de um código mais eficiente, assim como, se o tempo e o custo investido com treinamento de pessoal nas ferramentas de desenvolvimento antes do início do projeto foram positivos. Ao comparar os dados atuais com os dados de projetos anteriores que não utilizavam a nova abordagem, foi possível convencer a alta administração de que as mudanças proporcionaram um aumento da produtividade. Além disso, as métricas foram utilizadas para argumentar satisfatoriamente que partes do código eram confiáveis, de fácil manutenção e não deveriam ser refatorados. Isto significa que as métricas podem ser utilizadas em benefício do desenvolvimento de software, para que ele seja mais eficiente e previsível, ajudando a produzir produtos de maior qualidade e mais fácil de manter.

Métodos ágeis têm sido adotado para testar continuamente o software desenvolvido, de modo que iterações anteriores permaneçam funcionando corretamente juntamente com novas funcionalidades [P35]. Esta capacidade tem sido alcançada principalmente através da escrita e execução de testes automatizados [P13] [P16] [P23] [P36] [P40] [P45] [P57], incluindo testes de unidade e testes de aceitação. Isto significa que métodos ágeis nem sempre denota menos, por exemplo, menos testes não têm sido planejados [P35].

Além dos testes automatizados, uma avaliação mais formal de usabilidade e testes exploratórios do software no ambiente do usuário final tem sido recomendada para melhorar a satisfação do usuário [P57]. Esta abordagem tem aproveitado o potencial do especialista em usabilidade para melhorar os aspectos de usabilidade do software, assim como, do testador para investigar e descobrir cenários, e eventualmente defeitos e efeitos colaterais, que não haviam sido previstos anteriormente.

Por outro lado, alguns estudos têm apresentado certa discrepância em relação à adoção de testes ágeis. Nos estudos de [P57] [P60] testes para todos os níveis foram implementados e práticas como *Test-driven development* (TDD) foi adotada sem restrição alguma. Enquanto, no estudo de Melo & Ferreira [P36] a adoção de testes de unidade apresentou dificuldades, assim como, no estudo de Pedroso Júnior *et al.* [P45] a prática de TDD não foi adotada pela equipe por falta de exemplos reais. Outrossim, o estudo de [P13] revelou que a maioria das pessoas não gostam de escrever testes, e quando escrevem, o fazem de maneira insuficiente.

3.2.2.6 Manutenção de Software

O estudo de Brown *et al.* [P08] apontou que a pressão sobre os prazos de desenvolvimento tem levado vários projetos a cortes na criação e manutenção de documentos. Em alguns casos, as empresas contratadas têm produzido e entregado artefatos de código não favoráveis a mudanças e sem a devida documentação para equipes de manutenção de software. Embora tais cortes tenham beneficiado as equipes de desenvolvimento, as equipes de manutenção, muitas vezes tem suportado os efeitos negativos de tais decisões, por exemplo, um determinado software foi reescrito totalmente, fruto do resultado de um trabalho inadequado executado por uma empresa contratada [P08]. Além disso, este estudo demonstrou que muitas pessoas acreditavam que as atividades de manutenção possuíam um baixo prestígio, com baixo apoio da administração, e com uma prioridade baixa no nível corporativo. O resultado foi um isolamento cultural de projetos dentro da organização, onde uma troca de experiências valiosa entre projetos foi impedida e lições aprendidas não foram disseminadas na organização, o que prejudicou a eficiência organizacional e a melhoria em ferramentas, técnicas e práticas de desenvolvimento de software.

O estudo de Polo *et al.* [P49] avaliou a maturidade do processo de manutenção de software em duas APs da Espanha. O resultado mostrou que departamentos de TI de entidades públicas produzem softwares semelhantes para as mesmas pessoas, para resolver problemas parecidos, tal como, ferramentas de desenvolvimento permanecem as mesmas ao longo do tempo. As pessoas, geralmente, foram envolvidas em todas as etapas do ciclo de vida dos produtos de software, incluindo análise de requisitos, projeto, codificação, teste, implantação e manutenção do sistema. A organização do cliente foi a mesma, normalmente um outro grupo de servidores públicos da mesma AP com diferentes funções. Os autores concluíram que quando uma organização "sempre faz o mesmo", com pouca

mobilidade de pessoal, a sua experiência no desenvolvimento, provavelmente, poderá substituir a existência de documentos, assim como, procedimentos escritos para realizar seu trabalho.

3.2.2.7 Ferramentas de Apoio ao Desenvolvimento de Software

As ferramentas de apoio ao desenvolvimento de software têm facilitado a utilização de modelos adaptativos no governo [P13] [P36] [P40] [P45] [P47] [P57]. No estudo de Moore [P40], a infraestrutura de testes permitiu que o sistema fosse entregue aos seus clientes mais do que uma vez por semana. Da mesma maneira, o uso de ferramentas para especificação de requisitos e execução de testes de aceitação automatizados contribuiu para a entrega de funcionalidades mais rapidamente [P13], assim como, a utilização de *scripts* automatizados melhorou a execução de tarefas comuns do desenvolvimento de software, incluindo compilação e testes [P36] [P40]. Por outro lado, a falta de uma ferramenta tem tornado a execução de várias atividades do processo de desenvolvimento de software uma tarefa informal e caótica [P58].

Ferramentas para organizar as informações públicas do projeto, como também, registrar os defeitos encontrados e as solicitações de melhorias têm sido utilizadas em projetos de governo [P45] [P47] [P57]. O uso de ferramentas tem sido importante para mostrar informações mais realistas sobre o que está acontecendo no projeto, como também, quando as mudanças serão entregues e quando os defeitos serão corrigidos [P47]. Além disso, o fato da instituição possuir um ambiente tecnológico adequado para novas experiências facilitou o processo de aprendizado e adoção de práticas como Integração contínua, *Releases* curtas, *Design* simples, Código padronizado, Propriedade coletiva do código e TDD em uma instituição pública de grande porte do Brasil [P36].

3.2.2.8 Avaliação e Melhoria de Processos de Software

Modelos de maturidade, como *Capability Maturity Model* (CMM), Melhoria de Processos do Software Brasileiro (MPS.BR) e *Modelo de Referencia de Procesos* (COMPETISOFT) têm sido utilizados para aumentar a produtividade, melhorar a qualidade do software entregue, melhorar a satisfação dos clientes e fortalecer a relação entre desenvolvedores e usuários, ou seja, melhorar a capacidade de desenvolvimento de software no governo [P07] [P51] [P31].

Ao adotar CMM Nível 2 a *Dirección General de Informática de Rosario* situada na Argentina relatou vários benefícios, incluindo melhoria substancial de estimativas e

planejamento de tarefas, baixo índice de falhas identificadas pós-implantação do software, redução do retrabalho em até 35% devido ao detalhamento maior de requisitos durante o desenvolvimento, assim como, maior satisfação do cliente e redução de requisitos ambíguos. Por outro lado, o estudo recomendou que a adoção de CMM deve ser definida juntamente com a alta administração e ter uma visão estratégica clara com métricas adequadas que permitam visualizar o retorno sobre o investimento [P07].

Da mesma maneira, o estudo de Ricardo & Corrêa [P51] revelou que a adoção de MPS.BR Nível D pela empresa Informática de Municípios Associados (IMA) produziu melhores resultados em termos de custo, prazo e satisfação do cliente, como também, reduziu os retrabalhos, aumentou a qualidade dos projetos entregues em torno de 42% e trouxe maior capacidade de gerenciamento de projetos. Além do mais, o estudo mostrou que iniciativas de adoção de modelos de maturidade deveria ter apoio da alta administração, bem como, ter objetivos estratégicos, táticos e operacionais claros, e largamente disseminados e alinhados entre os gestores da empresa e a equipe de desenvolvimento de software.

Igualmente, o estudo de Luzuriaga *et al.* [P31] mostrou que a adoção do modelo de referência de processos COMPETISOFT pela Área de Informática do Poder Judicial de Neuquén na Argentina contribuiu para alcançar níveis mais elevados de satisfação do cliente em um tempo relativamente curto, além disso, contribuiu para melhorar os aspectos relacionados com gerenciamento de projetos. A experiência apontou que ter processos e produtos visíveis ao cliente impacta positivamente nos resultados do projeto, o que exige maior comprometimento das pessoas envolvidas no projeto.

Boria & Bianchi [P07] e Ricardo & Corrêa [P51] concordaram que para o melhor funcionamento de modelos de maturidade é necessário estabelecer um processo de melhoria contínua, através de análises críticas e grupos de discussão com propostas de melhorias e solicitações de mudanças, como também, garantir que todas as partes interessadas estejam comprometidas com o programa de melhoria.

Boria & Bianchi [P07] e Luzuriaga *et al.* [P31] concordaram que modelos de maturidade resolvem “o que fazer”, a falta de especificidade sobre “como fazer” determinadas atividades torna a adoção de modelos de maturidade mais difícil em ambientes de governo.

O estudo de Malheiros *et al.* [P33] apontou que a instanciação de um processo de software tem sido uma tarefa desafiadora, principalmente em grandes empresas. Além disso, os autores mostraram que não se pode esperar que soluções de processos de software

possam ser bem sucedidas em grandes empresas sem estarem apoiados por um programa de melhoria de processo de software. Neste trabalho, foi descrito a experiência do Serviço Federal de Processamento de Dados (SERPRO) na institucionalização de um programa de melhoria de processo de desenvolvimento de software abrangendo várias unidades de desenvolvimento. A abordagem foi composta por elementos adaptados do RUP, *Project Management Body of Knowledge* (PMBOK) segundo as melhores práticas definidas no modelo CMM. O resultado foi que institucionalizar um programa de melhoria de desenvolvimento de software que assegurasse um nível mínimo de qualidade e um processo padrão para várias unidades foi uma tarefa difícil e exigiu uma estratégia organizacional bem definida. Os principais desafios encontrados foram:

- Identificar os procedimentos que poderiam ser estabelecidos e facilmente automatizados para promover o tratamento eficaz de melhorias, revertendo-os em otimização de processos, sem renunciar a uma análise aprofundada do seu conteúdo;
- Incorporar melhorias em novos lançamentos do processo de software, mantendo a integridade de todos os ativos de processos;
- Disseminar os esforços do programa de melhoria através de várias unidades de desenvolvimento de software de uma forma homogênea;
- Garantir a rápida absorção de mudanças e um processo padrão que atendesse as necessidades de cada unidade organizacional;
- Manter as sucessivas versões do processo respeitando as melhores práticas de gerenciamento de software e desenvolvimento, sem perder de vista a realidade de cada unidade organizacional.

3.2.3 Desenvolvimento de Software na AP e a Relação com Outras Áreas

A comunidade de ES tem visto uma mudança significativa na forma com que os projetos de software têm sido desenvolvidos nos últimos anos. Por exemplo, com a competição global no desenvolvimento de software e o crescimento exponencial do mercado, grandes empresas tem organizado equipes de maneira distribuída em escala global para tirar vantagens competitivas e reduzir custos [AuP07].

3.2.3.1 Formas de Cooperação no Desenvolvimento de Software

Assim como em muitos outros setores, a AP pode ser confrontada com a escolha entre desenvolver um produto de software inteiramente ou parcialmente com seus servidores

públicos, ou então, contratar inteiramente ou parcialmente uma empresa da indústria de software, ou até mesmo, estabelecer parcerias com instituições acadêmicas ou outras APs. Em seguida, serão apresentadas e discutidas as formas de cooperação para o desenvolvimento de software no governo encontradas neste estudo.

3.2.3.1.1 Desenvolvimento Interno

O desenvolvimento interno consiste na execução das atividades de desenvolvimento de software no ambiente da AP, geralmente executado por servidores públicos [P04] [P21]. Bei *et al.* [P04] identificou seis vantagens desta abordagem:

- Maior controle sobre o processo de desenvolvimento de software (PDS);
- Maior controle sobre os artefatos;
- Menor dependência de empresas da indústria de software;
- Redução de custos;
- Maior chance de reutilização de componentes de software;
- Maior capacidade de reação.

De acordo com Bei *et al.* [P04], para estabelecer um desenvolvimento interno, uma AP deveria cumprir com os seguintes requisitos:

- Ter um nível suficiente de experiência em PDS e Gerenciamento de Projetos (GP);
- Ter um bom conhecimento sobre ferramentas de apoio as atividades de desenvolvimento de software, bem como, ferramentas de GP, incluindo controlador de versão, gerenciamento de melhorias e defeitos, etc.;
- Garantir a disponibilidade de força de trabalho com competência adequada para atuar nas diversas atividades do desenvolvimento de software, que podem estar disponíveis internamente ou contratada como "Tempo e Recursos Materiais".
- Ter conhecimento das tecnologias utilizadas no desenvolvimento de software;
- Garantir o envolvimento dos usuários.

Além disso, este estudo revelou algumas características do desenvolvimento de software interno no setor público [P04]:

- Desenvolvedores e clientes pertencem à mesma organização, isto permite que usuários finais do software podem ser facilmente envolvidos nas atividades de especificação de requisitos e testes de aceitação;

- A interação entre a equipe de desenvolvimento e o cliente geralmente é informal, isto significa que pode ser difícil estabelecer um processo formal de solicitação de mudanças devido a questões culturais;
- Em ambientes de governo onde Tecnologia da Informação (TI) não é o *core business*, pode haver uma falta de cultura de gestão de TI;
- A reutilização de software entre APs é fortemente incentivada;
- Em muitos países, as diretrizes de adoção de software na AP requerem uma avaliação preliminar de soluções baseadas em código aberto;
- A maioria dos softwares utilizados na AP são sistemas de gerenciamento de dados e se enquadram em uma das seguintes categorias: *Enterprise Resource Planning* (ERP), Governo Eletrônico, *Business Intelligence* (BI), *Decision Support System* (DSS), *Geographic Information System* (GIS), Intranet e Portais;
- Os projetos são de diferentes tamanhos e programação, normalmente existem alguns projetos grandes e um maior número de pequenos projetos;
- Os softwares são personalizados e necessitam de melhorias e mudanças frequentes, devido à dinamicidade do ambiente organizacional e as alterações constantes de leis e regulamentos.

3.2.3.1.2 Governo e Indústria

Os estudos [P18] [P21] [P59], afirmaram que grande parte do desenvolvimento de software do governo tem sido executado quase que inteiramente por empresas contratadas da indústria. Em geral, o governo define suas necessidades, mas depende quase que inteiramente de profissionais da indústria para arquitetar, projetar, codificar, integrar e testar os sistemas desenvolvidos. Após a concepção e desenvolvimento do sistema realizado pela indústria, o governo executa a implantação do sistema e os testes necessários em seu ambiente, sendo a indústria responsável pela resolução dos problemas encontrados durante os testes [P21].

Além disso, várias APs têm preferido contratos de preço fixo com “escopo fixo”, porque elas acreditam ser mais fáceis de gerenciar e não exige que o governo incorra em tanto risco [P15] [P16]. Entretanto, esta estratégia tem provado não ser bem sucedida e tem resultado em uma perda significativa da competência de engenheiros de software do governo em lidar com metodologias e tecnologias modernas de desenvolvimento de software, bem como, uma perda da experiência do governo em executar de maneira consistente sua capacidade de liderança quando aspectos críticos do desenvolvimento de software são

envolvidos [P21]. Da mesma maneira, esta abordagem de aquisição tem restringido a capacidade do governo em manter sistemas críticos internamente, o qual deveria deter a competência em lidar com as metodologias e tecnologias utilizadas, assim como, a compreensão associada ao impacto em custo e cronograma [P21]. Isto significa que uma dependência da indústria para todas as atividades relacionadas com a implementação tem sido criada [P36].

Outrossim, Heil [P21] mostrou 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 [P21]. Neste trabalho, o autor argumenta esta afirmação apresentando estudos e relatórios produzidos pelas organizações *Defense Science Board* (DSB) e *Government Accountability Office* (GAO) sobre falhas de desempenho técnico de projetos ao longo de vários anos:

- Em 2002, *DSB Task Force on Defense Software* informou que apenas 16% dos projetos foram concluídos dentro do prazo e dentro do orçamento; 31% dos projetos foram cancelados; 53% dos projetos remanescentes superaram os custos em 489%, e o produto final em média incluiu apenas 61% das funcionalidades pretendidas originalmente;
- Em 2004, o GAO informou que o DoD gastou 40% de seu orçamento de desenvolvimento de software em atividades de retrabalho devido a problemas relacionados com a qualidade;
- Em 2008, o DSB informou que a maioria dos sistemas de armas do DoD estavam falhando em testes operacionais iniciais;
- Em 2009, Levin informou que desde 2006 cerca da metade dos maiores projetos de aquisição do DoD excederam *Nun-McCurdy*, e que 95 grandes projetos tiveram um crescimento médio de 26% nos custos de aquisição e sofreram um atraso médio de quase 2 anos.

Da mesma maneira, a pesquisa de Pyster [P50] apontou que o governo não deveria transferir totalmente a responsabilidade pelo desempenho de projetos de software para a indústria, mas deveria compartilhar esta responsabilidade com a indústria através da execução de processos maduros de desenvolvimento de software, incluindo gerenciamento

de valor agregado, gerenciamento de requisitos, arquitetura de sistemas e outras atividades concernentes ao desenvolvimento de software. O reflexo do enfraquecimento da capacidade do governo em atuar de maneira mais incisiva ao longo do desenvolvimento de software causou um impacto negativo no custo, cronograma e desempenho técnico de vários projetos de software do DoD e outras partes do governo dos Estados Unidos, mesmo que estes projetos foram executados por empresas com nível elevado em relação ao CMM [P50].

Uma abordagem alternativa, conhecida como *in-house*, tem se provado bem sucedida para lidar com vários dos problemas relacionados com a contratação de desenvolvimento de software em ambientes de governo relatados anteriormente. Nesta abordagem, os engenheiros de software do governo assumem literalmente a liderança do projeto e atuam em estreita colaboração com a equipe de desenvolvimento de software da indústria, participando ativamente de todo o processo de desenvolvimento [P21]. Os artefatos do produto de software (especificações de requisitos, documentos do projeto, códigos, etc.) são desenvolvidos por profissionais do governo e da indústria de maneira conjunta, dê preferência na mesma localização física (geralmente no ambiente do governo) e próxima dos clientes, seguindo melhores práticas de construção de produtos de qualidade. Desta forma, os engenheiros de desenvolvimento de software do governo têm as mesmas expectativas e necessidades em relação ao custo, cronograma e desempenho técnico, como os seus pares da indústria. Além do que, esta abordagem estende a competência de desenvolvimento interno do governo, ao considerar a participação da indústria [P21].

Um bom exemplo do uso da abordagem *in-house* foi encontrado no projeto *Sentinel* executado pelo *Federal Bureau of Investigation* (FBI) [P16], que além da utilização da configuração *in-house*, adotou métodos ágeis para lidar com problemas de tamanho, duração e isolamento de clientes. A abordagem ágil do FBI incluiu uma combinação de estrutura contratante, abrangendo "*Cost Plus*" e "Tempo e Recursos Materiais". O resultado foi que o projeto foi capaz de implementar mais funcionalidades com equipes menores, além disso, as funcionalidades implementadas foram mais aderentes as necessidades dos usuários por conta da melhor relação entre a equipe de desenvolvimento e a equipe de negócios, bem como, entre a equipe do projeto e outras partes interessadas. Os autores afirmaram que literalmente, “o sistema *Sentinel* vai mudar a forma como o FBI irá conduzir seus negócios a partir do zero. É como se o FBI tivesse construído uma *startup* de software no porão de sua sede”.

Igualmente, o projeto *One Semi-Automated Forces* (OneSAF) *Objective System* (OOS) gerenciado pela *United States Army* foi capaz de reunir na mesma instalação física

26 empresas contratadas da indústria de software, representantes dos processos de negócios/usuários, gerentes de projeto, engenheiros do governo e desenvolvedores da indústria. Tudo isso aliado com práticas ágeis de XP e métodos tradicionais baseados no *Capability Maturity Model Integration* (CMMI) foram fundamentais para o projeto atingir suas metas programáticas e técnicas, bem como, adaptar-se as mudanças de requisitos [P57]. Como resultado, o projeto foi selecionado pelo *CrossTalk* como um dos *Top 5 Quality Software Projects* do governo dos Estados Unidos em 2003. Além disto, o estudo [P55] mostrou que os projetos vencedores deste prêmio instalaram no mesmo ambiente desenvolvedores, clientes e usuários durante o desenvolvimento.

Da mesma maneira, outros exemplos bem sucedidos foram citados no estudo [P21], com ênfase para o *Naval Surface Warfare Center Dahlgren Division* (NSWCDD), onde a abordagem *in-house* tem sido utilizada com sucesso por mais de 50 anos e colecionado vários casos de sucesso, tais como, *Tactical Tomahawk Weapon Control System* (TTWCS) software, *Submarine Launched Ballistic Missile* (SLBM), *Gunslinger* e *Wolfpack*. Práticas de desenvolvimento ágil foram utilizadas com sucesso para atender necessidades de missão crítica emergentes dos projetos *Gunslinger* e *Wolfpack*.

O sucesso apresentado nos projetos *Sentinel*, OneSAF-OSS, TTWCS, SLBM, *Gunslinger* e *Wolfpack* demonstraram que o modelo *in-house* de equipe de desenvolvimento de software do governo e da indústria não é apenas conceitual, e tem produzido melhores resultados. Segundo Heil [P21] há muitos benefícios para a utilização desta abordagem de contratação de software. Ao estabelecer e manter equipes de desenvolvimento de software de forma integrada, a liderança do projeto tem a opção de aumentar a equipe de desenvolvimento de software com os engenheiros de software do próprio governo, ou então, transferir a responsabilidade do desenvolvimento da indústria para o governo. Isto pode ser feito facilmente quando os engenheiros de software do governo são parte integrante da equipe de desenvolvimento desde o início. Outrossim, se os artefatos do projeto foram construídos em um ambiente integrado de desenvolvimento, então, o processo de transferência do conhecimento da indústria para o governo pode ser relativamente fácil.

3.2.3.1.3 Governo e Academia

A importância de métodos ágeis tem aumentado no Japão [P24]. Entretanto, o ciclo *Plan-Do-Check-Act* (PDCA) requer uma visão repetida do software, que não é aceita pelo sistema de orçamento do governo japonês, por que, ciclos repetitivos geram despesas repetidas, que por sua vez, significam mau uso do dinheiro público. Uma solução para este

problema foi encontrada no estudo de Kaneda [P24]. Neste trabalho, o autor mostrou uma combinação das abordagens *Project Based Learning* (PBL) e métodos ágeis. Estas abordagens foram utilizadas pela prefeitura de Kyoto e a universidade pública de Doshisha no Japão que desenvolveram dois softwares sem a ajuda de empresas da indústria de software. O resultado desta estratégia foi que os alunos e os professores foram beneficiados com o conhecimento sobre desenvolvimento de software com métodos ágeis para o mundo real e ambos tiveram uma maior compreensão da importância da ES. Da mesma maneira, esta abordagem permitiu que o governo ajustasse os softwares desenvolvidos o quanto necessário, sem um aumento de custo. Outrossim, a execução do ciclo PDCA, importante para as atividades diárias da prefeitura de Kyoto, foi mantida.

Em março de 2008 foi iniciado um projeto de parceria entre a Empresa de Processamento de Dados do Estado do Pará (PRODEPA) e a Universidade Federal do Pará (UFPA) a partir de um acordo de cooperação técnico–científica apoiada pela Secretaria de Desenvolvimento, Ciência e Tecnologia (SEDECT) do Estado do Pará. A parceria permitiu que a PRODEPA e a UFPA atuassem conjuntamente na definição e implantação de um processo de testes de software para a PRODEPA. Dois projetos piloto foram executados com o processo e os resultados preliminares apontaram para um significativo ganho na qualidade do produto final [P39]. Além disso, a PRODEPA reconheceu que a parceria com a academia resultou na eficácia do processo.

No Brasil, uma parceria envolvendo o Instituto Tecnológico de Aeronáutica (ITA) e reais necessidades do Instituto de Aeronáutica e Espaço (IAE), assim como, do Instituto Nacional de Pesquisas Espaciais (INPE) foi estabelecida para o desenvolvimento de aplicações aeroespaciais [P29]. O resultado da parceria contribui para aumentar a experiências dos profissionais envolvidos.

3.2.3.1.4 Governo Colaborativo

Desde 2004 a *Swedish Association of Municipalities for Joint Development of eServices* (Sambruk) tem atuado com o desenvolvimento colaborativo de serviços eletrônicos para municípios associados. Embora que estabelecer um projeto que compreenda as necessidades de vários municípios, leve mais tempo e resulte em atrasos nas fases iniciais, um grupo de municípios que atuam colaborativamente tem tido um maior poder de barganha, em comparação com municípios que agem individualmente. Além, disto, os custos são compartilhados entre os municípios associados [P44]. O estudo de Olsson & Rönnbäck [P44] revelou que a estratégia de desenvolvimento colaborativo entre municípios,

que reúne várias competências e diferentes experiências, tem oferecido uma abordagem mais ampla para enfrentar os desafios de desenvolvimento de serviços eletrônicos mais adequados às necessidades dos cidadãos. Esta abordagem tem fornecido um cenário maior de utilização de serviços de governo que são necessários para a criação do projeto conceitual do software. Porém, o resultado tem dependido muito do envolvimento frequente dos usuários, o que coloca uma pressão extra sobre os indivíduos que talvez não sejam aliviados de seus trabalhos diários para participar do projeto de desenvolvimento.

O desenvolvimento colaborativo no governo também tem acontecido entre diferentes instituições pertencentes à mesma empresa pública [P58]. Neste modelo, uma equipe de desenvolvimento é formada por desenvolvedores distribuídos em várias instituições. Unphon *et al.* [P58] citou que do ponto de vista dos desenvolvedores, uma questão crucial para o êxito do desenvolvimento colaborativo tem sido a possibilidade de capturar as semelhanças entre as organizações e a partir delas construir partes comuns, bem como, a perspectiva de integrar facilmente as especificidades de cada organização ao sistema geral. A principal ação executada neste estudo alocou os desenvolvedores perto das organizações dos usuários para facilitar uma implantação local do software, tal como, modificou o papel dos desenvolvedores, que além de desenvolvedores atuavam como usuários avançados e administradores do sistema em suas organizações. No que tange ao gerenciamento de requisitos, neste estudo os critérios de priorização de requisitos adotados foram: (1) o requisito foi altamente necessário em várias organizações, e (2) o requisito foi solicitado por uma organização que alocou mais recursos no projeto, por exemplo, tempo.

4 LIMITAÇÕES DA REVISÃO SISTEMÁTICA

A RSL descrita neste trabalho contribuiu para identificar estudos dentro do domínio pesquisado. Contudo, como qualquer outro método, foram encontradas algumas limitações.

Em primeiro lugar, a escolha das bases de dados para consulta procurou abranger o máximo das principais alternativas científicas disponíveis para execução de uma RSL. Entretanto, é possível que outras fontes de publicações científicas não utilizadas neste estudo também contenham artigos sobre desenvolvimento de software na AP. Portanto, não é possível garantir a cobertura total de artigos científicos sobre este assunto. Além disso, outros tipos de publicações (tais como: publicações governamentais, livros, teses, dissertações, etc.) disponibilizadas em outras fontes de dados (tais como: sítios Web da AP) não foram incluídas por não fazerem parte do escopo deste estudo. Porém, acredita-se que a inclusão de artigos científicos disponíveis em bases de dados conhecidas internacionalmente, assim como, os resultados apresentados forneceram uma boa indicação do "estado da arte" e o "estado da prática" do desenvolvimento de software na AP.

Em segundo lugar, no que diz respeito à utilização do software Start 2.0, não foi possível exportar e importar, de uma só vez, todos os artigos recuperados em algumas bases de dados para a ferramenta. Esta limitação foi encontrada nos mecanismos de busca do SpringerLink, ACM e BASE que não oferecem suporte adequado para o intercâmbio de dados em lote com outros sistemas. O caso mais crítico foi o da ACM que teve a estratégia de busca reduzida a um conjunto restrito de campos: título, autoria, palavras-chave e resumo. Porém, a adoção da ferramenta Start 2.0 foi positiva, e o seu uso poderá ser mais eficiente quando os mecanismos de busca oferecerem recursos mais sofisticados de integração com outros sistemas. Da mesma maneira, a combinação da ferramenta Start 2.0 com o software Microsoft Excel® foi importante para a execução da análise quantitativa dos resultados desta RSL.

Em terceiro lugar, o texto completo dos artigos foi obtido através dos serviços mantidos pelas bibliotecas da Empresa Brasileira de Pesquisa Agropecuária (Embrapa), Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) e Universidade Estadual de Campinas (Unicamp), sendo que vinte e oito artigos que poderiam estar relacionados com o tema não foram analisados por não estarem disponíveis “gratuitamente” nessas bibliotecas.

Por fim, 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.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foram apresentadas evidências científicas sobre o uso de modelos, processos e métodos de ES para o desenvolvimento de software na AP. Muitas constatações apareceram ao longo do texto, porém alguns aspectos merecem destaque:

5.1 Benefícios dos Modelos Adaptativos de Processos

Os estudos mostraram que modelos prescritivos de desenvolvimento de software são ineficientes para a natureza cada vez mais volátil e dinâmica dos ambientes de governo atuais [P05] [P34] [P38]. O principal argumento é que eles 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 de desenvolvimento de software onde os requisitos são estáveis [P05]. 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 [P05] [Som11].

Neste sentido, a adoção de modelos adaptativos tem melhorado a execução de vários projetos de governo [P13] [P16] [P23] [P36] [P40] [P44] [P45] [P57] [P59] [P60]. O processo de desenvolvimento de software tem sido mais aberto para o cliente através de uma comunicação direta entre o cliente e os desenvolvedores [P05] [P13] [P15] [P16] [P23] [P45] [P57]. Uma maior aproximação dos desenvolvedores aos clientes, aliado a uma priorização dos itens com maior valor agregado, bem como, a uma entrega rápida de software tem resultado em uma maior satisfação dos usuários [P23] [P59]. O *feedback* do cliente ao longo do desenvolvimento tem sido útil para melhorar o desenvolvimento das funcionalidades requeridas pelos usuários e agregar mais valor ao negócio, assim como, tem garantido o alinhamento do software aos objetivos de negócio [P23]. Entretanto, esta abordagem tem requerido um compromisso profundo e preparação, tanto do cliente como dos desenvolvedores para alcançar o seu pleno potencial [P44] [P45].

Igualmente, a implementação de novas funcionalidades tem se tornado algo previsível e transparente para o cliente. Novos recursos têm sido adicionados com maior qualidade, de maneira mais rápida, sem retrabalhos, com menos defeito e mais aderente às necessidades do cliente [P15] [P16]. Uma melhor comunicação entre os membros da equipe de desenvolvimento, bem como, entre desenvolvedores e usuários têm sido relatadas [P59]. Um efeito positivo no aprendizado de novas tecnologias, na satisfação do cliente, na moral da equipe, na qualidade do código e na produtividade das equipes estudadas tem sido

descritas [P13] [P36] [P45] [P60], como também, eles têm permitido uma maior facilidade de ajustes no escopo e direção do projeto [P44].

5.2 Desafios dos Modelos Adaptativos de Processo e Oportunidades de Pesquisa

Em alguns casos, a imagem bastante otimista no nível teórico dos modelos adaptativos de processo pode ser contraposta por uma realidade prática dominada por desafios, dificuldades e problemas concretos. Na sequência, os principais desafios e oportunidades de pesquisa relacionados a adoção de modelos adaptativos de processo identificados neste estudo são apresentados.

5.2.1 Relacionados com a Cultura Organizacional

Evidências sugerem que nem todos os ambientes de governo têm evoluído com o mesmo entusiasmo dos modelos adaptativos [P05] [P13] [P15] [P19] [P36]. Assim, a cultura inerente de uma organização pública nem sempre coincide com práticas de trabalho flexíveis e colaborativas, bem como, com clima de confiança e tomada rápida de decisões necessárias para que o desenvolvimento adaptativo aconteça com sucesso [P05].

Na mesma direção, grande parte dos “ruídos” sobre a utilização de modelos, processos e métodos de ES em ambientes de governo, está relacionada com a maneira de posicionar estas abordagens em um contexto maior, caracterizado como um contexto burocrático, hierárquico e incerto. Desta maneira, há muitas outras atividades necessárias para atender plenamente as necessidades de negócio do governo, incluindo atividades relacionadas com a cultura organizacional, formação de competências e trabalho colaborativo entre desenvolvedores, usuários, gestores do processo de negócio e profissionais de desenvolvimento de software da indústria, da academia e de outras APs.

Diante disso, estudos empíricos podem ser executados para compreender os ambientes de governo, tipicamente burocráticos (no que diz respeito aos regulamentos fixos, rigidez do modelo de gestão e estrutura organizacional) e extrair lições importantes para entender e ampliar o que se sabe sobre o uso de modelos adaptativos na AP em benefício do desenvolvimento de software no setor público.

5.2.2 Relacionados com o 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). Quando isso aconteceu, as melhores soluções de software foram criadas, inclusive com uma maior aceitação na

organização pública e na sociedade [P05] [P23]. 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 [P05] [P44].

5.2.3 Relacionados com a Manutenção de Software e Escalabilidade Organizacional

Grande parte das evidências encontradas apresentam a adoção de modelos adaptativos de processo no âmbito de novos projetos na AP (capacidade para novos projetos). Pouco se sabe sobre a adoção de modelos adaptativos em toda AP (capacidade organizacional), bem como, pouco se sabe sobre a viabilidade do uso de modelos adaptativos em projetos de manutenção de software. Pelo contrário, escalar modelos adaptativos de processo para toda AP e adotá-los em projetos de manutenção têm sido um desafio.

5.2.4 Relacionados com o Conhecimento e a Experiência dos Servidores Públicos

Há uma necessidade de mais estudos sobre o conhecimento e a experiência dos servidores públicos para atuar com métodos mais modernos e atuais de desenvolvimento de software, principalmente com métodos ágeis. Por exemplo, há várias características sobre métodos ágeis que são novas e desconhecidas para muitos membros da equipe [P36] [P45]. Em algumas situações, o conhecimento técnico e as habilidades da equipe de desenvolvimento não foram suficientes para realizar as tarefas atribuídas [P15] [P19]. Para solucionar este problema, algumas APs ajudaram elaborando e viabilizando um programa de treinamento, selecionando instrutores externos para ministrar o treinamento ou ministrando-o eles próprios [P13] [P19] [P15] [P36] [P57] [P59]. Ou então, algumas APs aceleraram o aprendizado por meio do uso de *coaching* interno [P36] ou contratação de *coaching* externo [P19].

No mesmo sentido, métodos ágeis têm exigido uma mentalidade muito diferente do que as pessoas estão acostumadas [P13]. Hajjdiab *et al.* [P19] concordou com esta opinião e acrescentou que a experiência de abordagens baseadas em modelos prescritivos é completamente diferente do que realizar reuniões diárias, trabalhar com *timebox*, entrega de pequenos incrementos de software, bem como, manter histórias de usuário e *backlogs* de produto. Por exemplo, ao invés de completar uma tarefa de especificação de requisitos em duas semanas, agora uma iteração inteira é de apenas duas semanas de duração [P13]. Isto significa que um ciclo completo de especificação de requisitos, codificação e testes deve

ser concluído dentro destas duas semanas. Projetos estudados por Melo & Ferreira [P36] relataram ter enfrentado dificuldades em convencer os clientes a realizar publicações parciais do sistema em produção, mesmo que agregassem valor ao negócio. Em geral, inserir novas práticas de trabalho, alterar o gerenciamento de projetos e hábitos de programação tem sido difícil [P19] [P59].

5.2.5 Relacionados com a Seleção do Projeto Piloto e Apoio Gerencial

A seleção do projeto certo como piloto pode ser desafiadora, nem todo projeto é adequado para ser o primeiro. O projeto piloto ideal depende da confluência de fatores como tamanho, duração, importância do projeto e engajamento do patrocinador ligado ao produto [Coh11]. Por exemplo, no estudo [P36] foram escolhidos sistemas que não eram críticos sob o ponto de vista de negócio e nem possuíam restrições de prazos severos. A escolha de projetos de grande porte é considerada um risco [P13]. Por outro lado, a falta de um projeto piloto tem sido relatada como um dos fatores de fracasso na adoção de métodos ágeis no governo [P19]. Além disso, a adoção de novos métodos de desenvolvimento requer o apoio da alta administração [P05] [P13] [P19] [P36]. Um suporte gerencial tem permitido que o aprendizado ocorra independentemente de qualquer problema ou falha [P36].

5.2.6 Relacionados com a Elaboração de Contratos Adaptativos

Entende-se que o sucesso apresentado nos projetos *Sentinel* [P16] e *OneSAF-OSS* [P57] demonstra que o modelo *in-house* de equipe de desenvolvimento de software do governo e da indústria juntamente com modelos adaptativos não é apenas conceitual, mas tem produzido melhores resultados práticos. Isto cria uma oportunidade para pesquisadores compreenderem e explorarem esta área a partir de uma perspectiva técnica, incluindo informações sobre diferentes modelos de contrato, tais como "*Cost Plus*", "Tempo e Recursos Materiais" e “Escopo Negociável”, assim como, informações sobre o ambiente de execução do projeto, onde desenvolvedores da indústria e do governo, clientes e usuários finais se instalam no mesmo ambiente e compartilham as mesmas expectativas.

Há uma necessidade de verificar a viabilidade da aplicação da abordagem *in-house* no cenário de desenvolvimento de software no setor público brasileiro. Essa abordagem complementa as estratégias de mitigação de riscos recomendadas pelo Tribunal de Contas da União [TCU13] ao considerar que o governo não deveria transferir totalmente a responsabilidade pelo desempenho de projetos de software para a indústria, mas deveria compartilhar esta responsabilidade com a indústria através da execução de processos de

desenvolvimento de software, incluindo a participação de servidores públicos em todas as etapas do desenvolvimento de software.

5.2.7 Relacionados com o Desenvolvimento Cooperativo

Projetos adaptativos dependem de decisões com base na confiança mútua, sendo bem adequados para projetos internos. Porém, contratar o desenvolvimento adaptativo quando a solução está sob a liderança de empresas contratadas externamente parece não proporcionar benefícios reais [P16]; 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 [P15] [P21] [P36]. O problema é agravado quando faltam incentivos para estabelecer projetos com base na confiança mútua, quando licitações públicas são criadas com base em abordagens prescritivas e quando há falta de conhecimento para criar e estabelecer contratos adaptativos. Potencializar a cooperação entre o governo e as empresas contratadas da indústria de software no desenvolvimento de software é um desafio para ser explorado futuramente. Além disso, as APs têm estabelecido parcerias com outras APs, bem como, com a academia para o desenvolvimento de software no setor público. Estas novas formas de cooperação para o desenvolvimento de software na AP criam oportunidades de estudo para entender as razões e os benefícios da adoção destas abordagens, as quais se opõem a opção convencional de contratação de empresas da indústria de software.

5.3 Trabalhos Futuros

Finalmente, uma vez que modelos adaptativos não têm sido testados e explorados suficientemente no setor público, 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 ambientes de governo para realizarem estudos empíricos para entender como estes modelos são executados na prática e que efeitos eles geram, incluindo suas vantagens e desvantagens. Além disso, várias questões de pesquisa emergem e precisam ser melhor investigadas:

- Será que os servidores públicos são abertos ao uso de modelos adaptativos e dispostos a participar ativamente em todo o processo de desenvolvimento?

- Os profissionais de TI da AP se sentem suficientemente apoiados e confiantes em “desafiar” os líderes de departamento e líderes políticos para executar o desenvolvimento de software de maneira diferente de modo a entregar serviços melhores e com maior valor agregado?
- Será que a equipe do governo tem a competência adequada para trabalhar em estreita colaboração com a equipe de desenvolvimento de software das empresas contratadas, e não apenas monitorarem e verificarem os resultados finais das tarefas realizadas?
- Será que modelos adaptativos podem ser escalados para ambientes organizacionais hierárquicos e burocráticos, típicos de governo?
- Será que existe uma imagem bastante otimista no nível teórico contra uma realidade prática dominada por desafios, dificuldades e problemas concretos?
- O que tem sido bom para os governos de outros países pode ser bom para o governo brasileiro também?
- Como o governo pode diminuir a burocracia e aumentar a agilidade, ao mesmo tempo, sem incorrer em tantos riscos no desenvolvimento de software?

Esta e outras perguntas serão exploradas na continuidade desta pesquisa, que tem como objetivo entender e ampliar o que se sabe sobre a adoção de métodos ágeis no setor público em benefício do desenvolvimento de software na AP no Brasil.

REFERÊNCIAS BIBLIOGRÁFICAS

[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.

[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.

[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.

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

[BoF14] Bourque, P.; Fairley, R. E. (Ed.) “SWEBOK V3.0: Guide to the Software Engineering Body of Knowledge”, IEEE Computer Society, 2014, Não paginado. Capturado em http://www.computer.org/portal/web/swebok/swebokv3, Maio 2014.

[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.

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

[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.

[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

[KiC07] Kitchenham, B.; Chartes, S. “Guidelines for performing systematic literature reviews in software engineering (version 2.3)”. Technical report, Software Engineering Group, School of Computer Science and Mathematics, Keele University and University of Durham, 2007, 57p.

[NAO12] UK NAO. “Governance for Agile delivery”, National Audit Office, 2012, 35p. Capturado em: http://goo.gl/YHc3Et, Dezembro 2014.

[Oat06] Oates, B. J. “Researching Information Systems and Computing”. London: SAGE Publications Ltd, 2006, 360p.

[Pre11] Pressman, R. S. “Engenharia de Software: Uma abordagem profissional”. Tradução de Ariovaldo Griesi. Porto Alegre: AMGH, 2011, 7.ed., 780p.

[Som11] Sommerville, I. “Engenharia de Software”. Tradução de Kalinka Oliveira e Ivan Bosnic. São Paulo: Pearson, 2011, 9.ed., 544p.

[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.

[TCU13] 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.

[WMR06] Wieringa, R.; Maiden, N.; Rolland, C. “Requirements engineering paper classification and evaluation criteria: a proposal and a discussion”. *Journal of Requirements Engineering*, vol. 11-1, Mar 2006, pp. 102- 107. DOI: 10.1007/s00766-005-0021-6

[WRH+12] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. “Experimentation in Software Engineering”. Springer, 2012, 259p. DOI: 10.1007/978-3-642-29044-2

[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: *Proceedings* Congresso Brasileiro de Software (CBSoft’10), 2010, 1p.

APÊNDICE A

CONJUNTO FINAL DE ARTIGOS

Tabela 17 - Conjunto final de artigos para análise em profundidade.

ID	Título	BD	Autoria	Ano	Fonte	Tipo
[P01]	Making agile development work in a government contracting environment-measuring velocity with earned value	IEEE	Alleman, G. B.; Henderson, M.; Seggelke, R.	2003	Agile Development Conference - ADC	Conference
[P02]	Verification and validation of operational software: a process and methodology critique	Wiley	Arthur, J. D.; Gröner, M. K.	2004	Software Process: Improvement and Practice	Journal
[P03]	Risk and risk management in software projects: A reassessment	Scopus	Bannerman, P. L.	2008	Journal of Systems and Software	Journal
[P04]	Processes for software development within the public administration	Scopus	Bei, A.; Lancia, M.; Lombardi, F.; Puccinelli, R.	2009	Workshop on Software Quality - WoSQ	Workshop
[P05]	Agile development in a bureaucratic arena - A case study experience	Scopus	Berger, H.	2007	International Journal of Information Management	Journal
[P06]	The utility of rapid application development in large-scale, complex projects	Scopus	Berger, H.; Beynon-Davies, P.	2009	Information Systems Journal	Journal
[P07]	Software process improvement under duress: experiences of SPI in a town hall in Argentina	IEEE	Boria, J. L.; Bianchi, A. J.	1999	Portland International Conference on Management of Engineering and Technology - PICMET	Conference
[P08]	An examination of software maintenance practices in a U.S. government organization	Wiley	Brown, A. W.; Christie, A. M.; Dart, S. A.	1995	Journal of Software Maintenance: Research and Practice	Journal
[P09]	Managing the dynamics of mutual adaptation of technology and organisation in information systems development projects	Wiley	Bygstad, B.	2005	Software Process: Improvement and Practice	Journal
[P10]	Driving usability into the public administration: The Italian experience	Scopus	Catarci, T.; Matarazzo, G.; Raiss, G.	2002	International Journal of Human Computer Studies	Journal
[P11]	Managing risk in software development projects: A case study	Scopus	Dey, P. K.; Kinch, J.; Ogunlana, S. O.	2007	Industrial Management and Data Systems	Journal
[P12]	Independent validation of software safety requirements for systems of systems	IEEE	Driskell, S. B.; Murphy, J.; Michael, J. B. and Shing, M.-T.	2010	International Conference on System of Systems Engineering - SoSE	Conference

ID	Título	BD	Autoria	Ano	Fonte	Tipo
[P13]	Agile metrics at the Israeli Air Force	Scopus	Dubinsky, Y.; Talby, D.; Hazzan, O.; Keren, A.	2005	AGILE Conference	Conference
[P14]	Software product assurance at the Jet Propulsion Laboratory	Science Direct	Fairley, R. E.; Bush, M.	1990	Information and Software Technology	Journal
[P15]	A case study: Introducing eXtreme programming in a US government system development project	Scopus	Fruhling, A.; McDonald, P.; Dunbar, C.	2008	Hawaii International Conference on System Sciences - HICSS	Conference
[P16]	The FBI gets agile	Scopus	Fulgham, C.; Johnson, J.; Crandall, M.; Jackson, L.; Burrows, N.	2011	IT Professional	Journal
[P17]	Do project management tools and outcomes differ in organizations of varying size and sector?	Scopus	Furumo, K.; Pearson, J. M.; Martin, N. L.	2006	Interdisciplinary Journal of Information, Knowledge, and Management	Journal
[P18]	Pessimism, computer failure, and information systems development in the public sector	Scopus	Goldfinch, S.	2007	Public Administration Review	Journal
[P19]	An industrial case study for Scrum adoption	Scopus	Hajjdiab, H; Taleb, A. S.; Ali, J.	2012	Journal of Software	Journal
[P20]	A bayesian belief network cost estimation model that incorporates cultural and project leadership factors	Scopus	Hamdan, K; Bibi, S.; Angelis, L.; Stamelos, I.	2009	IEEE Symposium on Industrial Electronics and Applications - ISIEA	Conference
[P21]	Addressing the Challenges of Software Growth and Rapidly Evolving Software Technologies	Wiley	Heil, J. W.	2010	American Society of Naval Engineers	Journal
[P22]	Why can't we manage large projects?	Scopus	Humphrey, W. S.	2010	CrossTalk	Journal
[P23]	A Case Study on the Adoption of Measurable Agile Software Development Process	BASE	Iliev, M.; Krasteva, I.; Ilieva, S.	2009	International Conference on Software, Services & Semantic Technologies - S3T	Conference
[P24]	Agile software development under university-government cooperation	Scopus	Kaneda, S.	2006	World Multi-Conference on Systemics, Cybernetics and Informatics - WMSCI	Conference
[P25]	Lessons for software development ecosystems: South Korea's e-government open source initiative	Scopus	Kim, S. L.; Teo, T. S. H.	2013	MIS Quarterly Executive	Journal
[P26]	Comparing private and public sector on information systems development and maintenance efficiency	Scopus	Krogstie, J.	2012	IFIP International Conference on Electronic Government - EGOV	Conference

ID	Título	BD	Autoria	Ano	Fonte	Tipo
[P27]	A survey on the application of the V-Modell XT in German government agencies	Scopus	Kuhrmann, M.; Lange, C.; Schnackenburg, A.	2011	European System, Software & Service Process Improvement & Innovation - EuroSPI	Conference
[P28]	Application of the V-Modell XT - Report from a pilot project	Scopus	Kuhrmann, M.; Niebuhr, D.; Rausch, A.	2006	International Software Process Workshop - SPW	Workshop
[P29]	Testing critical software: A case study for an aerospace application	Scopus	Loubach, D. S.; Nobre, J. C. S.; Da Cunha, A. M.; Dias, L. A. V.; Nascimento, M. R.	2006	Digital Avionics Systems Conference - DASC	Conference
[P30]	Can the case for CASE technology be advanced by Process Improvement?	Scopus	Love, M.; Siddiqi, J.	1998	Software Quality Journal	Journal
[P31]	Setting SPI practices in Latin America: An exploratory case study in the justice area	Scopus	Luzuriaga, J. M.; Martínez, R.; Cechich, A.	2008	International Conference on Theory and Practice of Electronic Governance - ICEGOV	Conference
[P32]	Supporting the Distributed German Government with POLITeam	Springer	M. Sohlenkamp, P. Mambrey, W. Prinz, L. Fuchs, A. Syri, U. Pankoke-Babatz, K. Klöckner, S. Kolvenbach	2000	Multimedia Tools and Applications	Journal
[P33]	Continuous process improvement at a large software organization	Scopus	Malheiros, V.; Paim, F. R.; Mendonça, M.	2009	Software Process Improvement and Practice	Journal
[P34]	From art form to engineering discipline? A history of us military software development standards, 1974-1998	Scopus	McDonald, C.	2010	IEEE Annals of the History of Computing	Journal
[P35]	Lessons learned using agile methods on large defense contracts	Scopus	McMahon, P. E.	2006	CrossTalk	Journal
[P36]	Adoção de métodos ágeis em uma Instituição Pública de grande porte - um estudo de caso	WBMA	Melo, C. de O.; Ferreira, G. R. M.	2010	Workshop Brasileiro de Métodos Ágeis - WBMA	Workshop
[P37]	Is Agile the Answer? The Case of UK Universal Credit	Springer	Michaelson, R.	2013	International Working Conference on Transfer and Diffusion of IT - TDIT	Conference
[P38]	Managing information system development in bureaucracies	Scopus	Middleton, P.	1999	Information and Software Technology	Journal
[P39]	Uma Experiência de Implantação de Processo de Testes - Caso PRODEPA	BDBComp	Monteiro, R. W.; Sizo, A.; Vinicius, T; Guimarães, L. F.; Martins, C. R. de L.; Sales, E.; Reis, C. A. L.	2010	Simpósio Brasileiro de Qualidade de Software - SBQS	Conference
[P40]	Evolving to a "lighter" software process: a case study	IEEE	Moore, R. J.	2001	Annual NASA Goddard Software Engineering Workshop	Workshop

ID	Título	BD	Autoria	Ano	Fonte	Tipo
[P41]	Factors affecting duration and effort estimation errors in software development projects	Scopus	Morgenshtern, O.; Raz, T.; Dvir, D.	2007	Information and Software Technology	Journal
[P42]	Web Application Development: Java, .Net and Lamp at the Same Time	Springer	Navón, J.; Bustos, P.	2005	International Conference on Web Engineering - ICWE	Conference
[P43]	Requirements Engineering: A Case of Developing and Managing Quality Software Systems in the Public Sector	Springer	Martin, N.; Gregor, S.	2005	Engineering and Managing Software Requirements	Journal
[P44]	Collaborative development of public information systems: A case study of "Sambruk" e-services development	Scopus	Olsson, C.-O.; Rönnbäck, A. Ö.	2010	eChallenges Conference	Conference
[P45]	Extreme Programming by example	BDPA	Pedroso Júnior, M.; Visoli, M. C.; Antunes, J. F. G.	2002	International Conference on Extreme Programming and Agile Processes in Software Engineering	Conference
[P46]	Practical experiences of model-based development: Case studies from the Swedish Armed Forces	Wiley	Pilemalm, S.; Hallberg, N; Sparf, M.; Niclason, T.	2012	Systems Engineering	Journal
[P47]	Adopting iterative development: The perceived business value	Scopus	Pinheiro, C.; Maurer, F.; Sillito, J.	2008	International Conference on Agile Processes in Software Engineering and Extreme Programming	Conference
[P48]	Improving responsiveness, bug detection, and delays in a bureaucratic setting: A longitudinal empirical IID adoption case study	Scopus	Pinheiro, C.; Maurer, F.; Sillito, J.	2010	International Conference on Agile Processes in Software Engineering and Extreme Programming	Conference
[P49]	Assessment of Maintenance Maturity in IT Departments of Public Entities: Two Case Studies	Springer	Polo, M.; Piattini, M.; Ruiz, F.; Jiménez, M.	2001	International Conference on Product-Focused Software Process Improvement - PROFES	Conference
[P50]	What Beyond CMMI Is Needed to Help Assure Program and Project Success?	Springer	Pyster, A.	2006	International Software Process Workshop - SPW	Workshop
[P51]	MPS.BR Nível D – A Experiência em Implantar o Modelo na Área de Governo Municipal	BDBComp	Ricardo, M. F. C.; Corrêa, A. S.	2011	Workshop Anual do MPS - WAMPS	Workshop
[P52]	Metrics in software process assessment, quality assurance and risk assessment	Scopus	Rosenberg, L. H; Sheppard, S. B.	1994	International Software Metrics Symposium - METRICS	Conference
[P53]	Cost profile of a highly assured, secure operating system	ACM	Smith, R. E.	2001	ACM Transactions on Information and System Security - TISSEC	Journal

ID	Título	BD	Autoria	Ano	Fonte	Tipo
[P54]	Increasing testing productivity and software quality: A comparison of software testing methodologies within NASA	Springer	Sova, D. W.; Smidts, C.	1996	Empirical Software Engineering	Journal
[P55]	Winning projects exemplify success for developers and acquirers	Scopus	Starrett, E.	2004	CrossTalk	Journal
[P56]	E-business in the regulation of medicines in Serbia	Scopus	Stojadinovic, T.; Radonjic, V.; Radenkovic, B.	2010	Drug Information Journal	Journal
[P57]	Army simulation program balances agile and traditional methods with success	Scopus	Surdu, J.; Parsons, D. J.	2006	CrossTalk	Journal
[P58]	Taking care of cooperation when evolving socially embedded systems: The PloneMeeting case	ACM	Unphon, H.; Dittrich, Y.; Hubaux, A.	2009	International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE	Workshop
[P59]	Staying agile in government software projects	Scopus	Upender, B.	2005	AGILE Conference	Conference
[P60]	Exploring XP for scientific research	Scopus	Wood, W. A.; Kleb, W. L.	2003	IEEE Software	Journal
[P61]	Information systems development project performance in the 21st century	ACM	Wright, M. K.; Capps III, C. J.	2010	ACM SIGSOFT Software Engineering Notes	Journal
[P62]	Supporting decision making in risk management through an evidence-based information systems project risk checklist	Scopus	Zhou, L.; Vasconcelos, A.; Nunes, M.	2008	Information Management and Computer Security	Journal

APÊNDICE B

INFORMAÇÕES RELACIONADAS COM A ORGANIZAÇÃO DA PESQUISA

Tabela 18 - Informações relacionadas com a organização da pesquisa.

ID	Tipo da pesquisa	Método de pesquisa	Método de coleta de dados	Tipo de dados analisado	Método de análise de dados
[P01]	Evaluation research	Estudo de caso	Não definido (ND)	ND	ND
[P02]	Evaluation research	Experimento	ND	Ambos	Estatístico
[P03]	Evaluation research	Múltiplos estudos de caso	Entrevista	Ambos	Estatístico Análise de conteúdo
[P04]	Evaluation research	Múltiplos estudos de caso	ND	Quantitativo	Estatístico
[P05]	Evaluation research	Estudo de caso Etnografia	Entrevista Observação	Qualitativo	Análise de conteúdo *Grounded theory*
[P06]	Evaluation research	Estudo de caso Etnografia	Entrevista Observação Análise de documentos	Qualitativo	Análise de conteúdo
[P07]	Evaluation research	Estudo de caso	ND	Ambos	Estatístico Análise de conteúdo
[P08]	Evaluation research	*Survey*	Entrevista	Qualitativo	Análise de conteúdo
[P09]	Evaluation research	Estudo de caso	Entrevista Observação Análise de documentos	Qualitativo	Análise de conteúdo
[P10]	Evaluation research	Estudo de caso	Entrevista	Qualitativo	Análise de conteúdo
[P11]	Evaluation research	Estudo de caso	Entrevista	Ambos	ND
[P12]	Evaluation research	Estudo de caso	ND	ND	ND
[P13]	Evaluation research	Estudo de caso	Entrevista Observação Questionário Análise de bases de dados	Ambos	Estatístico Análise de conteúdo
[P14]	Evaluation research	ND	ND	Qualitativo	ND
[P15]	Evaluation research	Estudo de caso	Entrevista Observação Questionário Análise de documentos	Ambos	Estatístico Análise de conteúdo
[P16]	Evaluation research	Estudo de caso	ND	Qualitativo	ND

ID	Tipo da pesquisa	Método de pesquisa	Método de coleta de dados	Tipo de dados analisado	Método de análise de dados
[P17]	Evaluation research	*Survey*	Entrevista Questionário	Quantitativo	Estatístico
[P18]	Opinion paper	Múltiplos estudos de caso	Análise de documentos	Qualitativo	Análise de conteúdo
[P19]	Evaluation research	Estudo de caso	Entrevista	Qualitativo	ND
[P20]	Evaluation research	*Survey*	Entrevista Análise de documentos Análise de bases de dados	Quantitativo	Estatístico
[P21]	Solution proposal	Múltiplos estudos de caso	Análise de documentos	Qualitativo	Análise de conteúdo
[P22]	Evaluation research	Estudo de caso	ND	Qualitativo	Análise de conteúdo
[P23]	Evaluation research	Estudo de caso	Observação Análise de bases de dados	Ambos	Estatístico Análise de conteúdo
[P24]	Evaluation research	Estudo de caso	ND	Qualitativo	ND
[P25]	Evaluation research	Estudo de caso	ND	ND	ND
[P26]	Evaluation research	*Survey*	Questionário	Quantitativo	Estatístico
[P27]	Evaluation research	*Survey*	Entrevista Questionário	Ambos	Estatístico Análise de conteúdo
[P28]	Evaluation research	Estudo de caso	ND	Quantitativo	Estatístico
[P29]	Evaluation research	Experimento	ND	ND	ND
[P30]	Evaluation research	ND	ND	Qualitativo	ND
[P31]	Evaluation research	Estudo de caso	Entrevista Observação Questionário Análise de documentos	Ambos	Estatístico Análise de conteúdo
[P32]	Evaluation research	Estudo de caso	ND	ND	ND
[P33]	Evaluation research	*Survey*	Questionário	Qualitativo	Análise de conteúdo
[P34]	Opinion paper	Revisão da literatura	Análise de documentos	Qualitativo	Análise de conteúdo
[P35]	Opinion paper	ND	ND	Qualitativo	ND
[P36]	Evaluation research	Estudo de caso	Entrevista Observação Questionário Análise de bases de dados	Ambos	Estatístico Análise de conteúdo
[P37]	Opinion paper	Estudo de caso	ND	ND	ND
[P38]	Evaluation research	Múltiplos estudos de caso *Survey*	Entrevista Observação Questionário	Qualitativo	Análise de conteúdo

ID	Tipo da pesquisa	Método de pesquisa	Método de coleta de dados	Tipo de dados analisado	Método de análise de dados
[P39]	Evaluation research	Estudo de caso	Entrevista Análise de documentos	Quantitativo	Estatístico
[P40]	Evaluation research	Estudo de caso	ND	ND	ND
[P41]	Evaluation research	*Survey*	Questionário Análise de bases de dados	Quantitativo	Estatístico
[P42]	Evaluation research	Experimento	ND	Qualitativo	ND
[P43]	Evaluation research	Estudo de caso	ND	Qualitativo	ND
[P44]	Evaluation research	Múltiplos estudos de caso	Entrevista	Qualitativo	Análise de conteúdo
[P45]	Evaluation research	Estudo de caso	ND	Qualitativo	ND
[P46]	Evaluation research	Múltiplos estudos de caso	Entrevista	Qualitativo	Análise de conteúdo
[P47]	Evaluation research	*Survey*	Entrevista Observação	Qualitativo	Análise de conteúdo
[P48]	Evaluation research	Estudo de caso	Entrevista Análise de bases dedados	Ambos	Estatístico
[P49]	Evaluation research	*Survey*	Entrevista Questionário	Qualitativo	Análise de conteúdo
[P50]	Solution proposal	*Survey*	Entrevista	Qualitativo	Análise de conteúdo
[P51]	Evaluation research	Estudo de caso	ND	ND	ND
[P52]	Evaluation research	*Survey*	Análise de bases de dados	Qualitativo	Análise de conteúdo
[P53]	Evaluation research	Estudo de caso	Análise de bases de dados	Quantitativo	Estatístico
[P54]	Evaluation research	Múltiplos estudos de caso	Entrevista Análise de bases de dados	Quantitativo	Estatístico
[P55]	Opinion paper	*Survey*	Entrevista	Qualitativo	Análise de conteúdo
[P56]	Evaluation research	Estudo de caso	ND	ND	ND
[P57]	Evaluation research	Estudo de caso	Entrevista	Qualitativo	Análise de conteúdo
[P58]	Evaluation research	*Grounded theory*	Entrevista Observação Análise de documentos *Workshop*	Ambos	Estatístico *Grounded theory*
[P59]	Evaluation research	Estudo de caso	ND	Qualitativo	Análise de conteúdo
[P60]	Evaluation research	Experimento	ND	Qualitativo	ND
[P61]	Evaluation research	*Survey*	Questionário	Quantitativo	Estatístico
[P62]	Opinion paper	Múltiplos estudos de caso	Análise de documentos	Qualitativo	Análise de conteúdo

APÊNDICE C

INFORMAÇÕES RELACIONADAS COM O CONTEÚDO DA PESQUISA

Tabela 19 - Informações relacionadas com o conteúdo da pesquisa.

ID	Administração Pública	País	Área do SWEBOK	Modelo, Processo, Método de ES	Nível de experiência	Quem executou o desenvolvimento do software?
[P01]	Rocky Flats Environmental Technology Site	Estados Unidos	Software Engineering Management (SEM)	Extreme Programming	Não definido (ND)	Indústria
[P02]	NASA Langley Research Center	Estados Unidos	Software Testing (ST)	Software Engineering Evaluation System - SEES	Iniciante	Governo e Academia
[P03]	Várias agências de governo da Austrália	Austrália	SEM	Não se aplica (NA)	NA	NA
[P04]	Consiglio Nazionale delle Ricerche (CNR) - Italian National Research Council	Itália	Software Engineering Models and Methods (SEMM)	Agile Unified Process	Experiente	Governo
[P05]	UK Regional Government Departament	Reino Unido	SEMM	Métodos Ágeis	Experiente	ND
[P06]	CAP Management - CAPM	Reino Unido	SEMM	Rapid Application Development	Iniciante	Indústria
[P07]	Dirección General de Informática de Rosario	Argentina	Software Engineering Process (SEP)	CMM	Iniciante	Governo
[P08]	Uma organização do governo dos Estados Unidos	Estados Unidos	Software Maintenance (SM)	Computer-Aided Software Engineering	Experiente	Governo e Indústria
[P09]	Norwegian Office of the Auditor General	Noruega	SEMM	Microsoft Solutions Framework	Iniciante	Governo
[P10]	Administração Pública da Itália	Itália	Software Design (SD)	Waterfall	ND	Indústria
[P11]	Town and Country Planning Office (TCPO)	Barbados	SEM	NA	NA	NA
[P12]	National Aeronautics and Space Administration - NASA - Estados Unidos	Estados Unidos	ST	IV&V - Verificação e Validação Independente	Iniciante	Governo
[P13]	Israeli Air Force	Israel	SEM	Extreme Programming	Iniciante	Governo

ID	Administração Pública	País	Área do SWEBOK	Modelo, Processo, Método de ES	Nível de experiência	Quem executou o desenvolvimento do software?
[P14]	Jet Propulsion Laboratory - NASA	Estados Unidos	Software Quality (SQ)	NA	NA	NA
[P15]	United States Strategic Command (USSTRATCOM)	Estados Unidos	SEMM	Extreme Programming	Iniciante	Indústria
[P16]	Federal Bureau of Investigation (FBI)	Estados Unidos	SEMM	Scrum	Iniciante	Governo e Indústria
[P17]	Vários gerentes de organizações do setor público	Estados Unidos	SEM	NA	NA	NA
[P18]	Setor Público	NA	Software Engineering Economics (SEE)	NA	NA	NA
[P19]	Um departamento de Telecomunicações e TI do governo dos Emirados Árabes Unidos	Emirados Árabes Unidos	SEMM	Scrum	Iniciante	Governo
[P20]	Mais de 20 organizações governamentais dos Emirados Árabes Unidos	Emirados Árabes Unidos	SEM	NA	NA	NA
[P21]	Department of Defense and Navy (DoD/Navy)	Estados Unidos	SEM	NA	NA	NA
[P22]	Naval Oceanographic Office (NAVO)	Estados Unidos	SEMM	Team Software Process - TSP	Experiente	Governo
[P23]	Uma Administração Pública da Bulgária	Bulgária	SEMM	Métodos Ágeis	Iniciante	Governo
[P24]	Prefeitura de Kyoto	Japão	SEMM	Métodos Ágeis	Iniciante	Academia
[P25]	National Information Society Agency (NIA)	Coréia do Sul	SEMM	OSS Development Models	Iniciante	Governo e Indústria
[P26]	Várias organizações públicas da Noruega	Noruega	SEM	NA	NA	NA
[P27]	Federal Office of Administration	Alemanha	SEMM	V-Modell XT	NA	NA
[P28]	German Department of the Interior	Alemanha	SEMM	V-Modell XT	Iniciante	Governo
[P29]	Instituto de Aeronáutica e Espaço (IAE) Instituto Nacional de Pesquisas Espaciais (INPE)	Brasil	ST	Rational Unified Process	Iniciante	Academia

ID	Administração Pública	País	Área do SWEBOK	Modelo, Processo, Método de ES	Nível de experiência	Quem executou o desenvolvimento do software?
[P30]	Information Systems Department (ISD)	Reino Unido	SEP	Computer-Aided Software Engineering	ND	Governo
[P31]	Área de Informática do Poder Judicial de Neuquén	Argentina	SEP	CompetiSoft	Iniciante	Governo
[P32]	German National Research Center for Information Technology (GMD)	Alemanha	SEMM	Evolucionário	ND	Governo e Academia e Indústria
[P33]	Serviço Federal de Processamento de Dados (SERPRO)	Brasil	SEP	Processo SERPO de Desenvolvimento de Soluções - PSDS	NA	NA
[P34]	US Department of Defense (DoD)	Estados Unidos	SEMM	Waterfall	NA	NA
[P35]	Government Defense	Estados Unidos	SEM	Métodos Ágeis	Experiente	ND
[P36]	Banco Central do Brasil	Brasil	SEMM	Scrum e Extreme Programming	Iniciante	Governo
[P37]	Her Majesty's Revenue and Customs (HMRC)	Reino Unido	SEMM	Métodos Ágeis	Iniciante	ND
[P38]	Várias organizações públicas do Reino Unido	Reino Unido	SEMM	SSADM	NA	NA
[P39]	Empresa de Processamento de Dados do Estado do Pará (PRODEPA)	Brasil	ST	ND	Iniciante	Governo
[P40]	Uma organização do governo dos Estados Unidos	Estados Unidos	SEMM	Extreme Programming	Iniciante	Governo
[P41]	Departamento de TI de uma grande organização pública do governo de Israel	Israel	SEM	NA	NA	NA
[P42]	Uma organização governamental do Chile	Chile	SEMM	Unified Software Development Process	Iniciante	Governo
[P43]	Australian Bureau of Statistics (ABS)	Austrália	Software Requirements (SR)	ABS Software Development Process	Experiente	Governo
[P44]	Swedish Association of Municipalities for Joint Development of eServices (Sambruk)	Suécia	SEMM	Scrum	ND	Governo Colaborativo

ID	Administração Pública	País	Área do SWEBOK	Modelo, Processo, Método de ES	Nível de experiência	Quem executou o desenvolvimento do software?
[P45]	Embrapa Informática Agropecuária	Brasil	SEMM	Extreme Programming	Iniciante	Governo
[P46]	Swedish Armed Forces	Suécia	SEMM	MODAF	Iniciante	Governo e Indústria
[P47]	Uma grande agência de petróleo e gás do governo do Canadá	Canadá	SEMM	Rational Unified Process	Iniciante	Governo
[P48]	Uma agência do governo do Canadá	Canadá	SEMM	Rational Unified Process	Experiente	Governo
[P49]	Provincial Center of Informatics (CENPRI) Departamento Público	Espanha	SM	ND	ND	Governo
[P50]	Departamento de Defesa dos Estados Unidos - DoD	Estados Unidos	SEM	NA	NA	NA
[P51]	Informática de Municípios Associados S/A (IMA)	Brasil	SEP	MPS.BR	Iniciante	Governo
[P52]	National Aeronautics and Space Administration (NASA)	Estados Unidos	SQ	NA	NA	NA
[P53]	US National Computer Security Center	Estados Unidos	SQ	Waterfall	Experiente	ND
[P54]	Software Engineering Laboratory (SEL) National Aeronautics and Space Administration's (NASA) Goddard Space Flight Center (GSFC) Flight e Dynamics Division (FDD)	Estados Unidos	ST	Waterfall	Experiente	ND
[P55]	Governo dos Estados Unidos	Estados Unidos	SEM	NA	NA	NA
[P56]	Uma agência de Medicina do Governo da Sérvia	Sérvia	SEMM	Rational Unified Process	ND	ND
[P57]	United States Army	Estados Unidos	SEMM	Métodos Ágeis e Métodos Tradicionais	Experiente	Governo e Indústria
[P58]	Uma agência do governo da Bélgica	Bélgica	SEM	Iterativo e Incremental	Iniciante	Governo Colaborativo
[P59]	National Institutes of Health (NIH)	Estados Unidos	SEMM	Scrum e Extreme Programming	Iniciante	Indústria
[P60]	NASA Langley Research Center	Estados Unidos	SEMM	Extreme Programming	Iniciante	Governo

ID	Administração Pública	País	Área do SWEBOK	Modelo, Processo, Método de ES	Nível de experiência	Quem executou o desenvolvimento do software?
[P61]	Setor Público	ND	SEM	NA	NA	NA
[P62]	Vários projetos do setor público do Reino Unido, Estados Unidos e Nova Zelândia	Reino Unido, Estados Unidos e Nova Zelândia	SEM	NA	NA	NA