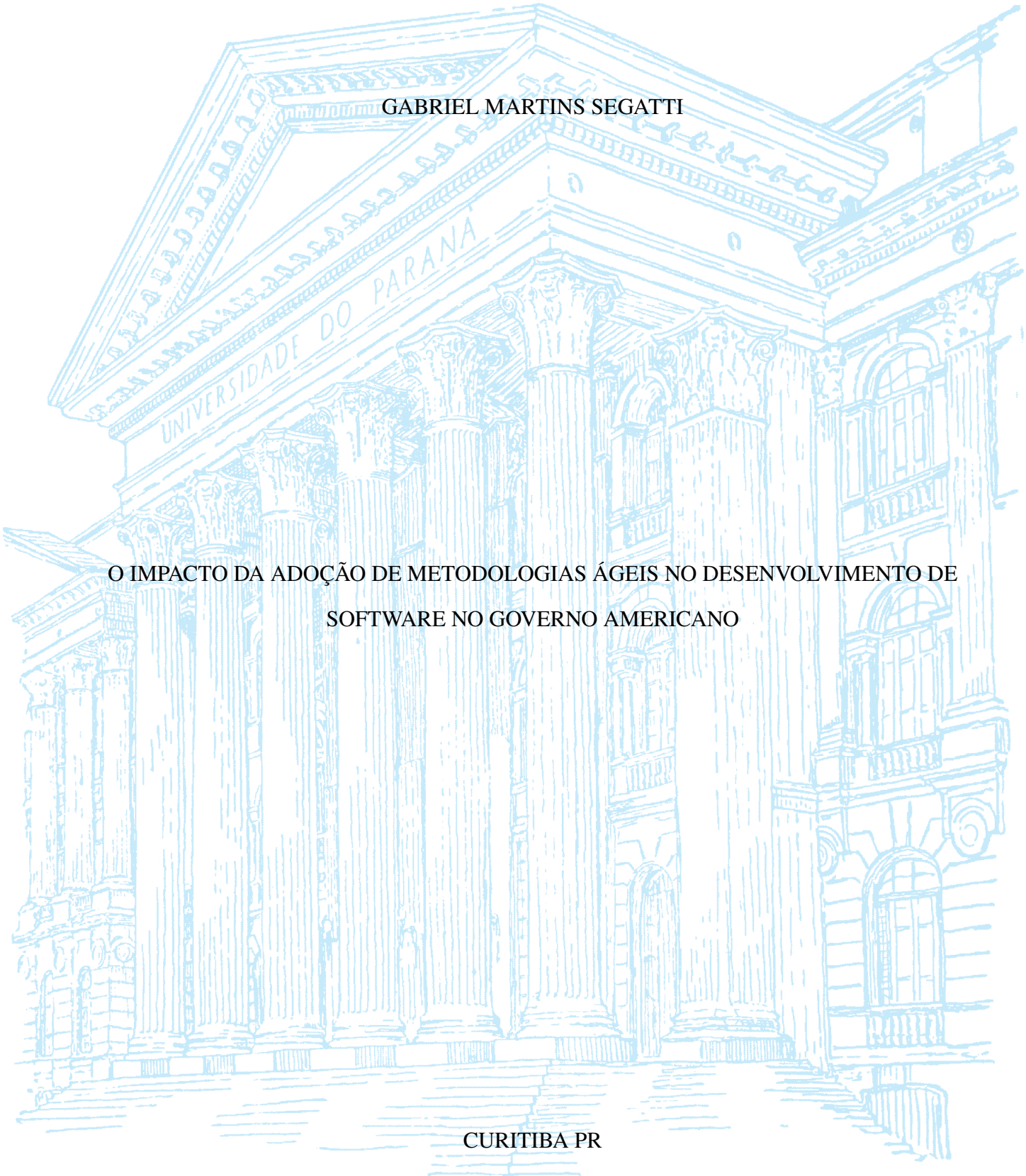
UNIVERSIDADE FEDERAL DO PARANÁ

GABRIEL MARTINS SEGATTI



O IMPACTO DA ADOÇÃO DE METODOLOGIAS ÁGEIS NO DESENVOLVIMENTO DE SOFTWARE NO GOVERNO AMERICANO

CURITIBA PR

2022

GABRIEL MARTINS SEGATTI

O IMPACTO DA ADOÇÃO DE METODOLOGIAS ÁGEIS NO DESENVOLVIMENTO DE SOFTWARE NO GOVERNO AMERICANO

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Armando Luiz Nicolini Delgado.

CURITIBA PR

2022

Dedico este trabalho primeiramente à minha avó, Maria das Dores Martins cujo carinho e atenção permitiram que eu atingisse os feitos que cumpro. Adicionalmente meus tios, Jackeline e Zé Roberto, cuja benevolência permitiu que eu desse continuidade à minha graduação. Por fim, meus pais, pelo apoio incessante, muitas vezes irreconhecido.

AGRADECIMENTOS

Agradeço a todos que colaboraram, direta ou indiretamente, para a construção deste trabalho. Em especial, agradeço veementemente ao Professor Armando Nicolini Delgado pela confiança, apoio e suporte na construção deste trabalho. Agradeço também a Professora Elenice Novak pelas orientações iniciais que levaram este trabalho na direção que aqui se encontra.

RESUMO

Este trabalho aborda como ocorre à transição, assim como algumas das consequências, da adoção de métodos ágeis em uma organização. Para tanto, reuniu-se dados de 15 departamentos do ramo executivo americano que deixaram de utilizar o modelo cascata em detrimento de algum método ágil. Com isso, foi-se gerado gráficos que expõem métricas acerca do rendimento destes departamentos, assim como a maneira que estes trabalham. Assim, espera-se que este trabalho seja uma amostra do processo de troca destas metodologias em desenvolvimento de *software*. Adicionalmente, espera-se concluir proposições que justifiquem a escolha de uma metologia, seja ágil ou cascata.

Palavras-chave: Metodologia. Cascata. Ágil.

ABSTRACT

This work addresses how it occurs the transition, as well as the consequences, of the adoption of agile methods within an organization. Therefore, data from 15 departments from the executive branch of the american government, which stopped utilizing the waterfall methodology, was gathered. With that, charts exposing the metrics regarding the performance of those departments, as well as the manners work was approached, were generated. That way, it is hoped that this article serves as a sample of the process of exchanging between such methodologies in software development. Furthermore, it is expected to conclude propositions that justify the choice of a metholodology, agile or waterfall.

Keywords: Methodology. Waterfall. Agile.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O termo "engenharia de *software*" tem origens incertas, embora muitos acreditem que foi-se cunhado pela primeira vez no ano de 1968, em uma conferência da OTAN [1] dedicada ao tópico em questão (Booch, 2018). O motivo para tal conferência, era a inabilidade de se cumprir com os requisitos de entrega de produtos de software, tanto em questão de tempo, como em funcionalidade (Wirth, 2008). Esta inabilidade era generalizada e afetava até mesmo companhias de grande porte, a tal ponto de colocá-las em situação financeira de risco (Wirth, 2008).

Conforme a necessidade de se obter soluções tecnológicas se alastrou, novas técnicas surgiram com o tempo, visando combater os problemas oriundos da produção de *software* ao redor do planeta. Algumas destas soluções remetiam ao aspecto técnico de programar, apresentando novas linguagens ou paradigmas de programação. Outras, propunham novas maneiras de se organizar o trabalho a ser feito. Daí, surgem dois principais métodos de organização de trabalho a serem escolhidos, sendo eles o método Cascata e o Ágil (Dima e Maassen, 2018).

BASILI (1985) afirma que um dos maiores problemas sobre desenvolvimento de *software* é a escolha apropriada de métodos e ferramentas para a construção e manutenção de projetos, assim como a avaliação do quão adequado é o produto e o processo que o gera. Diante essa afirmação, a maneira como o trabalho a ser realizado é abordado é muito importante, visto sua relação direta com a qualidade do resultado.

Dado este cenário, o trabalho aqui apresentado busca apresentar e caracterizar ambos métodos, assim como demonstrar sua utilização e transição, do modelo cascata para o Ágil, dentro do ramo executivo do governo americano. Com isto, visamos demonstrar as consequências desta troca, comparar o desempenho de trabalho sob estas metodologias e prover um ponto de referência na discussão da utilização destas técnicas.

Isto será feito apresentando o rendimento de diversos órgãos governamentais dos Estados Unidos, antes, durante, e depois de alternarem entre o método cascata e alguma metodologia ágil. Para isso, foram compilados dados de toda década de 2010, limpando-os e apresentando-os visualmente, servindo como uma fonte de dados relevantes sobre estas questões.

Este trabalho é composto por mais 5 capítulos. Inicialmente discutimos o que é uma metodologia dentro do contexto de produção de *software*. Depois, abordamos o que são metodologias do tipo cascata e ágil. Por fim, no capítulo 5 discutimos às origens e características dos dados que compõem informações dispostas no capítulo 6, onde se registra parte do processo e consequências da adoção de métodos ágeis em detrimento do método cascata.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Analisar como se dá a adoção de métodos ágeis no governo federal americano, e quais os impactos em como recursos e tarefas são utilizados e abordados, quando comparado às práticas previamente utilizadas durante o uso do modelo cascata.

[1] (Randell, 1996)

1.1.2 Objetivos Específicos

- Explicitar alguns dos processos de desenvolvimento de software usualmente utilizados no desenvolvimento de aplicações.
- Ilustrar a transição entre métodos discrepantes de construção de *software*, dentro das organizações relevantes.
- Demonstrar as consequências das trocas de abordagens para a realização de tarefas.
- Comparar o desempenho dos órgãos federais americanos perante os diferentes métodos de desenvolvimento.

2 METODOLOGIAS

O desenvolvimento de *software* envolve a combinação de programação, verificação, teste, e depuração de um sistema (BOURQUE et al., 2004). O Processo de Desenvolvimento de *Software* envolve a definição, implementação, avaliação, medição, gestão, mudança e melhoria da execução do desenvolvimento de *software* em si (BOURQUE et al., 2004). As diferentes maneiras com que este processo pode ocorrer formam, enfim, o que referiremos como Metodologia.

A importância de uma metodologia está concentrada no fato que o Processo de Desenvolvimento de *Software* é um *Wicked Problem* (DEGRACE e STAHL, 1990). *Wicked Problem* (ou Problema Perverso) é caracterizado como algo de difícil ou impossível solução, devido ao fato de possuir requisitos incompletos, contraditórios e mutáveis. É um problema não fixo, onde não se há uma única solução (BRIGGS, 2007).

Algumas das principais características dos Problemas Perversos, apresentadas tanto por CONKLIN (2003) quanto por RITTEL e WEBBER (1973) são:

- Não existe formulação definitiva para o problema.
- Não possuem tempo de parada.
- Não existe solução certa ou errada por completo, somente soluções melhores ou piores.
- Toda instância deste tipo de problema é única.

Logo, dada à complexidade inerente a esta classe de problemas, a maneira de abordá-los (pelo uso de metodologias de *software*), torna-se crucial, podendo produzir um resultado satisfatório ou não. Diante a existência das mais diversas metodologias utilizadas no contexto de tecnologia da informação, os próximos capítulos visam fornecer o embasamento teórico daquelas que serão referenciadas com maior frequência neste artigo. Mais especificamente, descreveremos o que é a Metodologia do tipo Cascata, assim como àquelas denominadas Ágeis. Existem diversos métodos além destes, mas não os abordaremos aqui. Por fim, buscamos enxergar como se dá a substituição do uso de uma metodologia por outra no governo americano.

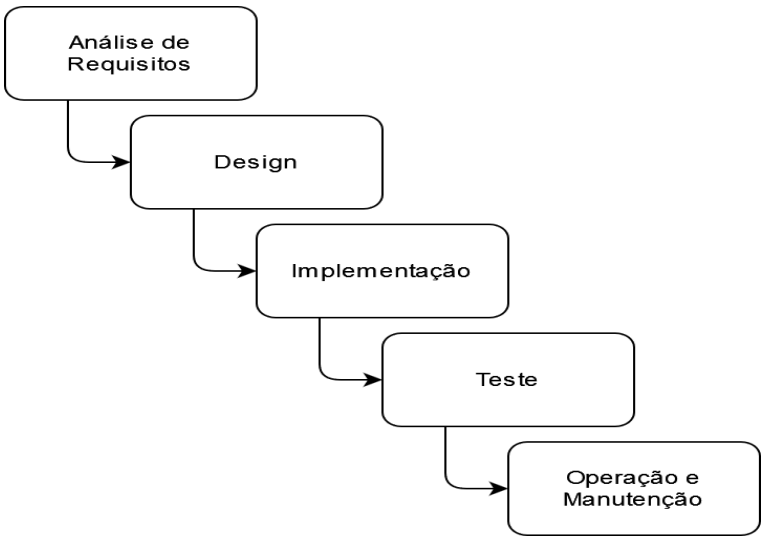
3 METODOLOGIA CASCATA

Formalmente introduzido em 1970, a Metodologia Cascata aborda o desenvolvimento de *software* como um fluxo sequencial de fases, terminando uma etapa por completo antes de avançar para a próxima (ROYCE, 1970).

Uma vez que tal avanço ocorre, não se tem a oportunidade de retornar a um estágio previamente concluído, devido ao fato que de cada fase completada é bem definida e contém objetivos completamente discrepantes das demais, não existindo sobreposição entre tais atividades (ADENOWO e ADENOWO, 2013). Esta série de características, que resultam em uma certa rigidez, são originárias das estratégias de manufatura de hardware muito presentes na década de 70, coincidentes com o mesmo período de surgimento do Modelo Cascata (McCORMICK, 2012). Devido à proximidade entre *hardware* e *software*, herdou-se então tais práticas no modelo em questão.

As etapas que constituem o modelo cascata são (FOWLER e HIGHSMITH, 2001): Análise de Requisitos, *Design*, Codificação e Testes. PRESSMAN (2005), por sua vez, as identifica como: Iniciação do Projeto e Análise de Requisitos, Planejamento, Modelagem (*design*), Construção (codificação e testes) e Distribuição (entrega e suporte). Visando encontrar um meio termo, temos: Análise de Requisitos, *Design*, Implementação, Teste e Manutenção e Suporte (ADENOWO e ADENOWO, 2013).

Figura 3.1: Fases do Modelo Cascata.



Adaptado de ADENOWO e ADENOWO, 2013.

Análise de Requisitos: trata-se de listar, de maneira detalhada, os requisitos que o programa em questão deve satisfazer, com o objetivo de se obter um documento de especificação de requisitos ao final de tal fase. Desta forma cria-se um entendimento do que é necessário para realizar o *design*, assim como quais são os objetivos que o projeto deve cumprir. Dentre as atividades realizadas estão: determinar o escopo do problema, identificar os usuários do sistema, entrevistá-los, dentre outros (FOWLER e HIGHSMITH, 2001).

***Design*:** delimitada como a atividade entre a especificação de requisitos e a codificação de uma aplicação, a etapa de *Design* tem como objetivo visualizar e definir as possíveis soluções

para os objetivos citados na documentação de especificação de requisitos. Trata-se de definir, por exemplo, o fluxo da aplicação, os algoritmos, padrões e *frameworks* a serem utilizados.

Implementação: a etapa destinada a tradução dos resultados das fases anteriores a uma (ou mais) linguagem de programação.

Teste: realiza-se todos os testes necessários para assegurar o funcionamento correto dos resultados pertinentes ao passo anterior. Testes de unidades, componentes, integrações e outros são realizados aqui. Checa-se se o *software* foi programado de acordo com às especificações obtidas, design planejado e corretude do código.

Manutenção e Suporte: uma vez que o produto desenvolvido encontra-se pronto, dentro dos requerimentos iniciais, esta fase torna-se necessária para cumprir com outros requisitos que podem vir a surgir durante o ciclo de vida do *software*.

Finalmente, os casos em que se recomenda o uso da metodologia cascata são (STOICA et al., 2013):

- Os requisitos já encontram-se bem definidos e claros.
- Não existem requisitos ambíguos (ou a existência destes é mínima).
- A tecnologia a ser utilizada já é bem compreendida pelos times de desenvolvimento.
- O projeto tem curta duração.

3.1 CARACTERÍSTICAS DO MODELO CASCATA

A metodologia Cascata foi a primeira formalmente publicada e dominou o cenário de desenvolvimento de *software* até o inicio da década de 90 (SOARES, 2004). Em 1995 o STANDISH GROUP apresentou dados que apontavam ineficácias em tal modelo. Tomando como base mais de 8000 projetos, somente 16,2% destes foram entregues respeitando as estimativas iniciais de tempo e custo. Um terço dos mesmos foram cancelados. Cerca de metade foram entregues, porém, ou com prazos maiores, custos maiores ou funcionalidades insuficientes. Dentre os projetos cancelados, as médias de custo e atraso eram de 189% e 222% respectivamente. O motivo pelo qual as falhas eram tão sobressalentes aos casos de sucesso estavam correlatos ao Modelo Clássico de desenvolvimento (SOARES, 2004). Sob tal metodologia, até mesmo organizações com amplos recursos documentaram atrasos de 500% acima do prazo inicialmente estimado e recolhimento de produtos, mostrando que até mesmo grandes corporações encontravam mesmos problemas (CUSUMANO e SMITH, 1995).

As subseções seguintes tem como intuito explicitar quais características do Modelo Cascata justificam tais estatísticas.

3.1.1 Especificação Prévia e Única

No clássico artigo "*No Silver Bullet: Essence and Accidents of Software Engineering*", BROOKS (1987) afirma que a especificação completa de um *software* dentro de uma única etapa é impossível. No entanto, é exatamente isto que ocorre no Modelo Cascata.

O problema está no fato de que os próprios clientes não sabem o que querem. Isto é, não foi-se pensado de maneira profunda suficiente o produto a ser desenvolvido e o problema

em questão (BROOKS, 1987). Adicionalmente, o cliente não tem contato algum com alguma versão ainda em desenvolvimento do *software* a ser entregue (visto que a etapa de levantamento de requisitos ocorre antes de todas demais fases do Modelo Cascata), o que o impede (o cliente) de validar se a codificação sendo realizada é condizente com seus ideais de usabilidade e funcionalidade. Pesquisas realizadas sob demanda da NASA (*National Aeronautics and Space Administration*) apontam que isto é demasiadamente prejudicial visto que requisitos sofrem o maior número de mudanças durante a fase de Teste (LUTZ, 1993). Logo, perde-se a chance de adquirir mais conhecimento sobre o problema e prover requisitos com maior precisão e corretude.

Se os requisitos usados como direção para a construção do produto não estão corretos, então, obviamente, os resultados não serão condizentes com as expectativas. BROOKS (1987) também afirma que retificar a cadeia de erros criada por requisitos inexatos é de extrema dificuldade e causa as maiores ineficiências do *software* resultante.

3.1.2 Linearidade e Completude Única de Etapas

Como dito anteriormente, o Modelo Cascata consiste em realizar etapa por etapa, sequencialmente, até que no final tenha-se em mãos o produto almejado. Ainda no artigo "*No Silver Bullet: Essence and Accidents of Software Engineering*" (BROOKS, 1987, p.14) afirma-se que "[...] as estruturas conceituais que construímos atualmente são demasiadamente complicadas para serem especificadas de forma previamente acurada e complexas demais para serem construídas de forma impecável, então, nós devemos tomar abordagens completamente diferentes.". Ou seja, o autor afirma não só o ponto explicado na subseção anterior (3.1.1), mas vai além, ao explicitar que, as chances de fazer o *design* e construir-se um *software* que funcione da maneira mais próxima ao ideal em uma única longa tentativa, são baixas.

Este tipo de prática impede que sejam oferecidos *feedbacks* (ou sugestões) ao desenvolvimento sendo efetuado. Logo, permite que o projeto seja continuamente dirigido na direção de grandes partes de códigos inutilizáveis e interfaces inutilizáveis, que não cumprem com as especificações obtidas como observado por BOEHM (1988).

BROOKS defende, portanto, que pequenas partes do *software* sejam entregues continuamente, e, adicionalmente, observa que esta prática leva a resultados melhores que o contrário.

4 METODOLOGIAS ÁGEIS

A origem do uso de metodologias ágeis ainda é aberta. Muitas vezes, referenciado como sua pedra fundacional, é o Manifesto Ágil (2001), que, embora seja responsável por cunhar o nome "Ágil", apareceu depois da criação de métodos já assim caracterizados (FOWLER e HIGHSMITH, 2001).

A utilização de métodos iterativos e incrementais pode ser traçada até meados do final da década de 50, em laboratórios da IBM, por colegas de John Von Neumann: (LARMAN e BASILI, 2003):

> ”Até onde consigo me lembrar, todos nós achávamos que cascatear um projeto enorme era estúpido, ou, ao menos, ignorante perante à realidade. O que a definição do Modelo Cascata fez para nós, foi levar a realização de que estávamos fazendo outra coisa, algo sem nome, exceto pela caracterização de ser desenvolvimento de *software*”. (Gerald M. Weinberg, tradução nossa).

ZELKOWITZ (2009), apresenta em seu livro "*Emerging Technologies*", publicações do setor privado japonês, no ano de 1986, repletas de similaridades ao que hoje definimos como ágil. A revisão deste método, publicada na revista *Harvard Business Review* afirmava que "a abordagem velha e sequencial simplesmente não é suficiente". Três anos depois, um grupo de gerentes da IBM (*International Business Machines Corporation*) também publicaram um artigo que apresentava uma nova forma de trabalhar, onde afirmaram que "o envolvimento do usuário no desenvolvimento do produto resulta em cumprir com requisitos e menor ciclo de desenvolvimento[...]", uma ação contrária as características da seção anterior. Observa-se que o descontentamento com as práticas rígidas do modelo Cascata fez que times de desenvolvimento mudassem a forma de abordar seu trabalho, mesmo que ainda na época não existisse um termo que definisse tal mudança.

COLLIER (2012) define metodologias ágeis como o conjunto de abordagens nas quais requisitos e soluções evoluem em conjunto, a partir de um esforço dos times de desenvolvimento e usuários. COCKBURN e HIGHSMITH (2001) afirmaram que Ágil é "[...] a entrega prematura de valor. Isto implica na entrega regular e com antecedência de *software* funcional, com foco em comunicação entre os times e interação com os usuários." Por fim, LAPHAM et al. (2010), provê a seguinte definição:

> ”Uma abordagem iterativa e incremental para o desenvolvimento de *software*, performada em um ambiente altamente colaborativo, por times auto-organizados, com "somente o necessário" de cerimônias, produzindo *software* de alta qualidade de maneira efetiva, do ponto de vista financeiro e temporal, em congruência com as necessidades voláteis dos *stakeholders*.”

4.1 CARACTERÍSTICAS DE METODOLOGIAS ÁGEIS

Conforme grupos de desenvolvimento percebiam deficiências no Modelo Cascata, estes criavam seus próprios processos para construção de *software* que abordavam tais falhas conforme suas necessidades. Estas adaptações refletiam-se nas características que cada abordagem apresentava. MILLER (2001) afirma que, por fim, de maneira geral, estas diferentes implementações convergiam para um ponto em específico: diminuir o tempo de desenvolvimento de um projeto. Para tanto, o autor definiu algumas principais características relacionadas a processos ágeis.

4.1.1 Iteratividade

MILLER (2001) também afirma que processos ágeis reconhecem que erros necessariamente precedem acertos dentro de um projeto. Para tanto, divide-se a progressão do mesmo em pequenos ciclos que duram semanas. Neste intervalo de tempo, se dá a ocorrência de diversas etapas do modelo tradicional de desenvolvimento, dentre elas: análise, projeto, implementação, teste e, possivelmente, lançamento do que foi produzido (PALMQUIST et al., 2013).

PALMQUIST et al. (2013) atestam que estas iterações permanecem ocorrendo até que um destes pontos ocorra:

- Todas funcionalidades desejadas pelo cliente foram entregues.
- Esgotamento da verba para o projeto.
- O prazo de entrega é alcançado.

4.1.2 Limitações

As iterações mencionadas não ocorrem de maneira irrestrita. Estas possuem longevidade delimitada entre uma a seis semanas, geralmente (MILLER, 2001). Uma iteração só possui todas atividades do projeto se este está próximo de seu término, caso contrário, repete-se o modelo Cascata. No entanto, deve-se, idealmente, cumprir para com todos objetivos demarcados ao início da mesma, de modo a se obter alguma funcionalidade (BECK, 1999). Logo, a quantidade de trabalho a ser realizado deve ser factível perante a habilidade do time e a longevidade da iteração.

Desta forma, minimiza-se o risco das ações tomadas devido a maior adaptabilidade do produto, que passa por diversas etapas do desenvolvimento de *software* incansavelmente (MORAN, 2014).

4.1.3 Incrementabilidade

Devido a natureza de como ocorre a divisão de trabalho, particionada em pequenos ciclos de desenvolvimento, metodologias ágeis não buscam construir um produto inteiro em uma única passada. Desta forma, partes cruciais do sistema podem sem desenvolvidas em paralelo, ou em tempos diferentes, ou em razões de esforço diferentes. Desta forma, a cada funcionalidade concluída e testada, ocorre sua integração com o sistema, isto é, de maneira incremental.

4.1.4 Adaptabilidade

A cada iteração assume-se que a possibilidade de surgir novas tarefas antes não planejadas (como um *bug* crítico recém encontrado a ser analisado). Da mesma forma PALMQUIST et al. (2013) afirmam que, apesar de requisitos serem providos de maneira prévia ao time de desenvolvimento, estes não são fixos, podendo acarretar na geração de novas tarefas. Adicionalmente, a coleta de sugestões acerca de melhorias deve ser constante. Para tanto, o processo ágil utilizado em questão deve ser maleável o suficiente para descartar e acrescentar novas atividades em movimento.

4.1.5 Colaborativo e Orientado à Pessoas

MILLER (2001) atesta que comunicação é essencial em qualquer processo de desenvolvimento de *software*. Adicionalmente, diz que, em metologias ágeis, devido a característica de

incrementabilidade, entender como o sistema interage com o problema a ser resolvido requer extensa comunicação entre os times de desenvolvimento e as pessoas que atribuem requisitos à aplicação. Para tanto, COCKBURN e HIGHSMITH (2001) explicitam a necessidade de comunicação como instrumento de redução do tempo e corretude de desenvolvimento. Adicionalmente, todo método ágil deve incluir um representante das partes nele interessadas (denominado *Product Owner*), para que se mantenha congruência entre o desenvolvimento do projeto e as expectativas dos clientes. Visto que trata-se de um processo inerentemente iterativo e passível de sofrer mudanças, a comunicação é a chave para que estas alterações sejam, em primeiro lugar, iniciadas, e depois atendidas.

Estas características refletem os princípios filosóficos estabelecidos no Manifesto Ágil, (FOWLER e HIGHSMITH, 2001):

- Indivíduos e interações são mais valiosos que processos e ferramentas.
- *Software* funcional tem mais valor que documentações compreensivas.
- Colaboração por parte dos clientes é mais valiada que negociações de contratos.
- Responder à mudanças é mais crucial que seguir um plano.

5 DADOS DA PESQUISA

As seções a seguir buscam apresentar aspectos e origens dos dados cujos objetivos do trabalho estão alicerçados, levando em consideração as definições apresentadas anteriormente. Adiante, traremos incessantemente dados relativos aos departamentos executivos do governo americano. A escolha de utilizar o governo americano, neste caso, muito se dá pela dificuldade de encontrar dados públicos que permitam analisar a forma como se aborda o desenvolvimento de software em organizações largas. Dados de outros governos foram pesquisados para a realização deste trabalho, no entanto, não se encontrou informações mais detalhadas que as aqui apresentadas.

5.1 CONTEXTO

Em 2012, a OMB, *Office of Management and Budget* (ou Escritório de Gerenciamento e Orçamento), responsável por averiguar as verbas e qualidade dos demais programas governamentais americanos, emitiu oficialmente uma orientação às organizações públicas americanas sob sua jurisdição: para que estas passassem a adotar metodologias ágeis e práticas iterativas, e, portanto, se afastarem de projetos que envolvessem desenvolvimentos lineares longínquos. Logo, obtemos um cenário onde se é possível averiguar a transição e as consequências de adoção de metodologias ágeis. Abordamos, portanto, nesta e na próxima seção, os dados relativos à década de 2010, visto que conseguimos ver as práticas de trabalho antes, durante e depois da orientação mencionada.

5.2 FONTE DE DADOS

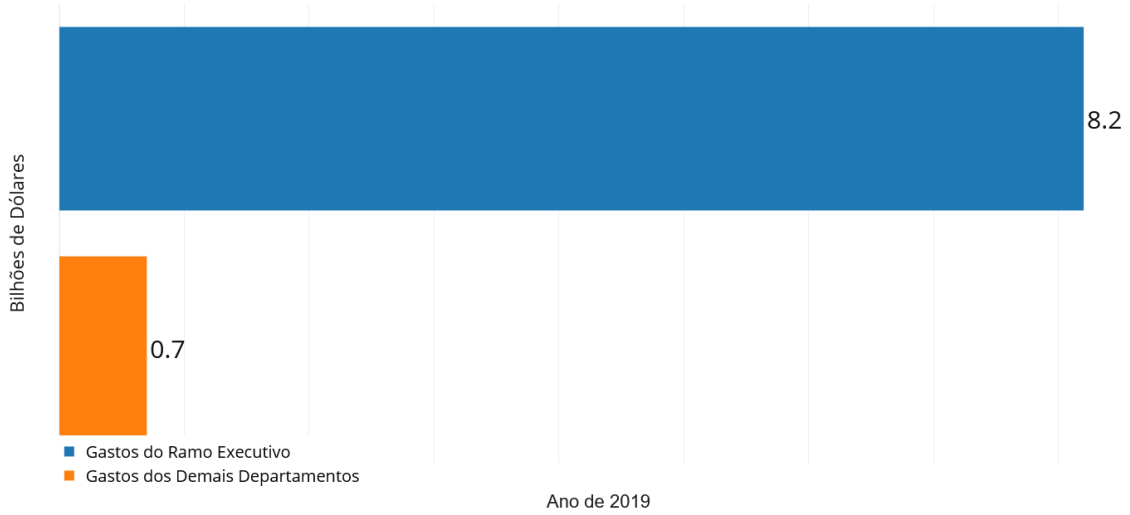
Fundado no ano de 2009, o *IT Dashboard*[1] é um portal com intuito de permitir, tanto à população como aos órgãos públicos, acesso aos investimentos e progressos em tecnologia da informação realizados pelo governo federal americano. Desta forma, é possível que as partes interessadas possam tomar ação de forma rápida e eficaz, visto que, além dos dados disponibilizados, se possui, também, os contatos necessários para engajar os times responsáveis pelos projetos.

Os dados disponibilizados contam com informações gerais e detalhes de mais de sete mil projetos em tecnologia. Tais dados são atualizados regularmente pelo *Chief Information Officer* (Oficial Chefe de Informação) respectivo de cada agência, apontados no portal. Existem vinte e seis agências listadas no portal, das quais abordaremos quinze neste trabalho, todas pertencentes ao ramo executivo do governo americano. Desta forma, espera-se haver uma congruência no comportamento destas agências (já que, ultimamente, todas respondem à mesma liderança), tornando possível apresentar e justificar disparidades nos dados apresentados com maior clareza. Os órgãos executivos são os que possuem as maiores verbas para a área de tecnologia da informação, trazendo consigo maiores números de projetos, gerando amostras mais representativas (figura 5.1).

Adicionalmente, para cada uma das 26 agências citadas, o *IT Dashboard* disponibiliza 12 tabelas com dados que retratam seus projetos de tecnologia, tanto no aspecto legal, financeiro, contratual ou gerencial. Neste trabalho, nos atemos a utilizar 2 tabelas destas 12: uma que aborda

[1] https://itdashboard.gov/

Figura 5.1: Gastos em TI no Ano Fiscal de 2019 (em bilhões)



Fonte: O Autor (2022)

primordialmente o andamento dos projetos, e a outra que retrata a progressão das atividades de um projeto (ambos termos serão definidos em maior detalhe na próxima seção).

5.3 CARACTERIZAÇÃO DOS DADOS

No total analisaremos mais de 3000 projetos (e as tarefas a eles relacionadas) cujo início se deu entre primeiro de janeiro de 2010 e trinta e um de dezembro de 2019.

A tabela de projetos, como dito anteriormente, carrega consigo informações relativas ao estado de um projeto. Estes dados são utilizados para gerar as imagens expostas no capítulo a seguir. Alguns destes dados são exibidos na tabela 5.1.

Tabela 5.1: Alguns dos Campos Presentes na Tabela de Projetos.

Campo	**Descrição**
ID Único do Projeto	Um número especificado por uma agência que identifica de maneira única um projeto.
Nome	Nome utilizado por uma agência para se referir a um projeto.
Data de Início	Data de início para projetos em andamento e data esperada de início para projetos ainda por começar.
Data de Completude	Análogo à data de início.
Projeto de *Software*	Indica se o projeto se trata primariamente em programar uma aplicação utilizando alguma linguagem de programação.
Metodologia de Ciclo de Vida	Metodologia utilizada durante o ciclo de vida do projeto, podendo ser: "Modelo Cascata", "Ágil"ou "Não primordialmente um projeto de desenvolvimento de *Software*".
Release a cada 6 meses	Indica se um projeto lança uma nova versão em produção ao menos a cada 6 meses.

O Escritório de Gerenciamento e Orçamento (OMB), define um projeto como:

> ”[...] Um esforço temporário tomado visando completar um produto ou serviço com um início e fim bem definidos, assim como objetivos específicos que, quando atingidos, impliquem em completude. Projetos são iniciados para o desenvolvimento, modernização, melhoramento, aposentamento ou manutenção de um ativo tecnológico. Estes projetos são compostos por tarefas chamadas Atividades.” (OMB, 13 de junho de 2020, tradução nossa).

Uma Atividade por sua vez é uma tarefa cuja conclusão torna o fim do projeto mais próximo, independente de seu grau de complexidade (OMB, 2020). Para lidar com isto, uma

atividade pode possuir outras atividades filhas que possuem um escopo de ação com maior especificidade. Algumas das 16 ações categorizadas são, por exemplo, recolher requisitos, prototipar, desenvolver (no sentido de programar) e realizar testes em aceitação com o usuário. Na tabela 5.2, encontra-se uma amostra de alguns dos dados disponíveis na tabela de atividades, assim como seus significados. Tais informações também são utilizadas para mostrar os dados expostos no capítulo a seguir:

Tabela 5.2: Alguns dos Campos Presentes na Tabela de Atividades

Campo	Descrição
Nome da Atividade	Nome utilizado pela agência para se referir a uma atividade.
Descrição da Atividade	Descreve o resultado do trabalho empenhado.
Data Planejada de Início	Data esperada em que uma atividade se iniciará.
Data de Início Real	Data em que uma atividade começou a ser realizada de fato.
Data de Completude Planejada	Data esperada de entrega de uma atividade.
Data de Completude Real	Data em que uma atividade foi entregue de fato.
Contribuidores Técnicos	Quantidade de pessoas que contribuíram no desenvolvimento técnico.
Contribuidores Gerais	Quantidade de pessoas que contribuíram em demais áreas não técnicas.

A OMB define alguns termos comumente utilizados neste trabalho, sendo estes:

Release: é o lançamento ou atualização de um produto entregue ao cliente, cada projeto contém várias *releases* dentro de si.

Sprint: uma sequência distinta de atividades planejada de maneira prévia, cujo critério de avaliação, se contemplado, pode resultar numa *release*.

Contribuidores Técnicos: quantidade de pessoas que contribuem diretamente à construção do *software* (como engenheiros de *software*).

Contribuidores Gerais: outros funcionários que contribuem diretamente ao projeto, usuários que realizam testes, gerentes de projetos e outros.

5.4 USO DOS DADOS

Na próxima seção apresentaremos métricas construídas com base nos dados descritos anteriormente. Para tanto filtramos projetos com as seguintes características:

Data de Início: Entre 1 de janeiro de 2010 e 31 de dezembro de 2019.

Data de Completude: Projetos com valores não nulos, ou seja, completos.

Projeto de *Software*: Projetos que indicam ser primariamente correlatos a desenvolvimento.

Metodologia do Ciclo de Vida: Projetos que utilizam tanto Cascata ou Métodos Ágeis.

Agência: Uma das 15 agências do ramo executivo dentre as 26 disponíveis.

A partir desta filtragem, agregamos os projetos em relação aos anos em que estes estão em desenvolvimento, ou seja, um projeto ativo nos anos de 2010 e 2011 estará presente em dados relativos a ambos os anos, mas não aos demais.

6 DE METODOLOGIA CASCATA PARA ÁGIL NO GOVERNO AMERICANO

Como foi orientado pela OMB, projetos executados pelo governo deveriam então se afastar dos métodos tradicionais de desenvolvimento de *software*. Visamos nesta seção, analisar se isto foi de fato realizado, isto é, se os projetos carregados na última década espelham os princípios ágeis citados anteriormente, ao invés de simplesmente categorizados como ágeis por pressão governamental. Concorrentemente, buscamos explicitar as consequências de seguir tal diretriz.

As métricas a seguir são geradas com o propósito de explicitar a transição causada pela diretriz mencionada. Portanto mantemos um enfoque em gráficos que evidenciam, tanto a transição citada, quanto as consequências de tal situação. Adicionalmente, a quantidade de dados exposta publicamente é limitada, o que nos impede de abordar um número demasiadamente alto de métricas diferentes.

6.1 IMPLEMENTAÇÃO DE MÉTODOS ÁGEIS

A figura 6.1 mostra as metodologias usadas no desenvolvimento de *software* durante a década de 2010. É possível perceber que, inicialmente, grande parte dos projetos sequer eram categorizados. Esta tendência se mantém até o ano de 2013, um ano após a OMB lançar a orientação mencionada anteriormente. De 2014 em diante projetos ágeis passam a ser a maioria, desaparecendo quase em totalidade os projetos não categorizados.

Estes projetos (sem categorização) são possíveis frutos da transição entre metodologias utilizadas. No decorrer das próximas páginas, as estatísticas que levam em conta o fruto do trabalho realizado durante estes anos, mostram que a capacidade de trabalho dos times encontrava-se limitada, confirmando a possibilidade de uma transição entre formas de trabalhar. Uma vez que esta falta de categorização desaparece, os departamentos passam a apresentar rendimentos consideravelmente maiores. Isto é retratado com maiores detalhes na seção 6.2 adiante.

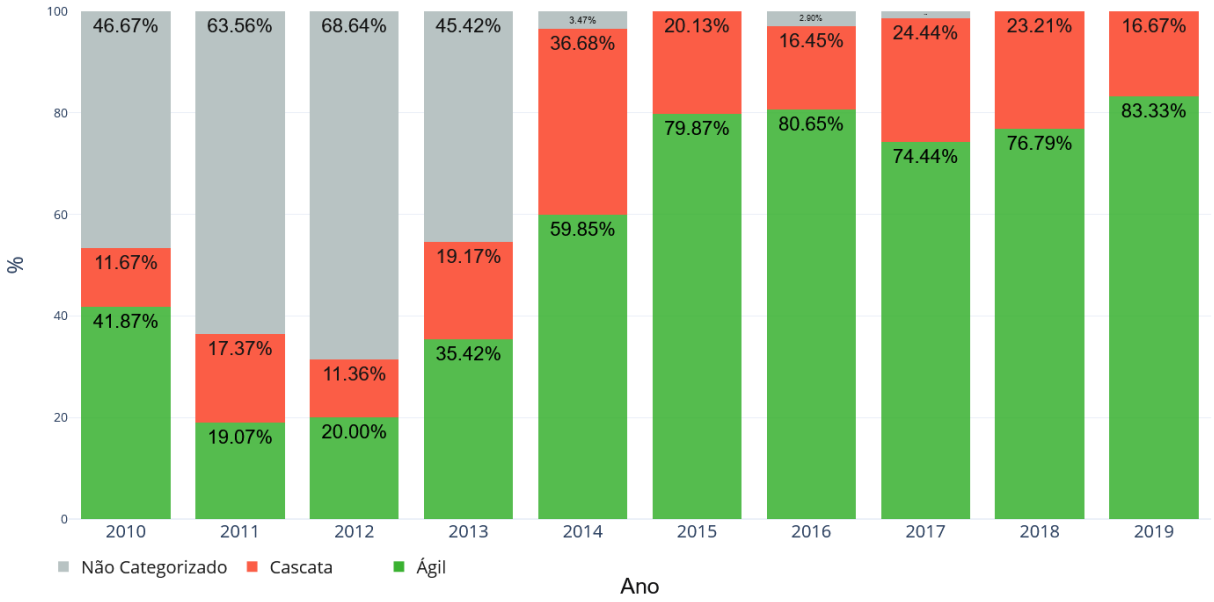
Em um estudo realizado pela *Hewlett Packard* com mais de 600 correspondentes, espalhados por 475 empresas, foi-se atestado que grande parte destas organizações passaram a adotar metodologias ágeis com intensidade no início da última década, sendo que até 2008 sua taxa de uso havia aumentado em pequenos passos, incrementalmente (Hewlett Packard, 2017).

O gráfico da figura 6.2 mostra em maior detalhe a linha do tempo de adoção de metodologias ágeis baseado no estudo acima. Percebe-se que do ano 2004 até 2009 temos uma variação de apenas 13%, ou seja, um aumento de quase 2% ao ano. No entanto, de 2009 para 2010 temos um aumento de 20% na adoção de tais práticas, mais que o acumulado de todos anos passados. Em 4 anos, de 2010 a 2014, temos um salto de 63%.

O comportamento dos departamentos do ramo executivo, apresentado na figura 6.1, comparado às 475 empresas estudadas pela Hewlett Packard, pode ser dito similar. No mesmo intervalo de tempo, entre 2010 e 2014, temos um salto de 40% no uso de alguma metodologia considerada ágil, e, se comparamos aos anos com menores percentuais de adoção, 2011 ou 2012, o aumento do uso destes métodos sobe em aproximadamente 60% também.

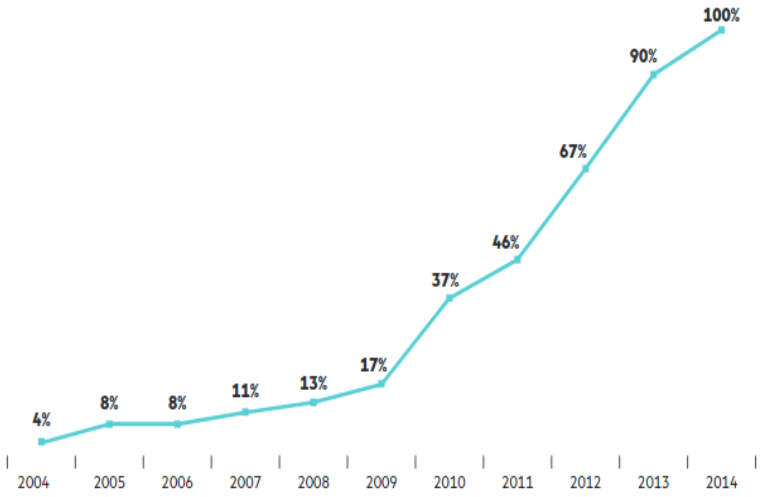
Os números correspondentes ao uso do modelo cascata apresentam similaridades, embora em menor nível. Ao entrevistar 601 correspondentes, no ano de 2017, em relação a qual forma de se abordar trabalho era usado em projetos em suas companhias, somente 9% relataram o uso de desenvolvimento incremental ou algo similar (figura 6.3). Os departamentos

Figura 6.1: Metodologias Utilizadas a Cada Ano



Fonte: O Autor (2022)

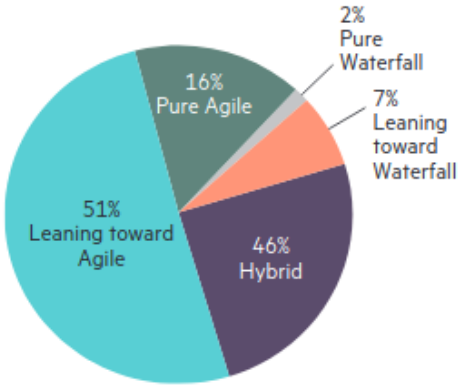
Figura 6.2: Porcentagem das Empresas que Adotaram Metodologias Ágeis



Fonte: Hewlett Packard (2017)

estudados neste trabalho mantém quase 20% de seus projetos utilizando modelo cascata durante os 5 últimos anos da década anterior (figura 6.1), quase o dobro relativo ao mencionado. De qualquer forma, é possível enxergar a diminuição expressiva em seu uso.

Figura 6.3: Principais Metodologias Utilizadas no Ano de 2017



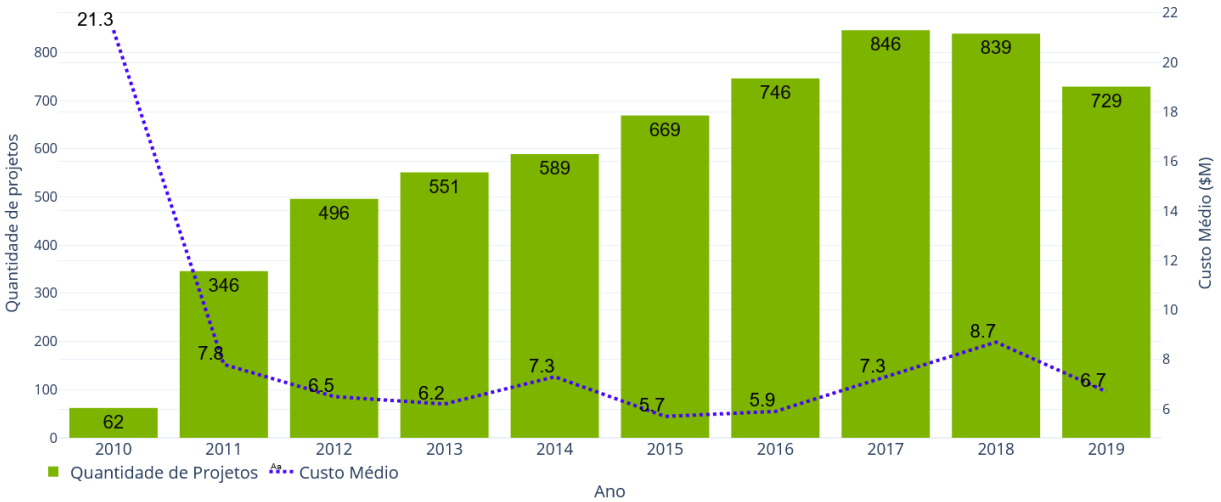
Fonte: Hewlett Packard (2017)

6.1.1 Quantidade e Recursos dos Projetos

É importante observar o número de projetos sobre os quais as estatísticas anteriores são regidas. Embora no primeiro ano temos quase 40% de projetos caracterizados como ágeis trata-se de um total de 62 trabalhos iniciados em 2010.

Algo a ser caracterizado como um indicador de transição para uma produção de *software* orientada por métodos ágeis é que, no ano seguinte, temos 346 projetos em execução. Ou seja, migra-se de projetos com escopo maior para partições menores a serem trabalhadas. Pode-se confirmar isto pelo fato de que o custo médio de execução destes diminuiu em uma escala de três vezes (figura 6.4). Unindo a maior quantidade de projetos existentes ao menor escopo destes, os departamentos do governo americano se colocam em posição de cumprir com as características da iteratividade e adaptabilidade, por exemplo, descritas no capítulo 4. É mais simples iterar pelas etapas de desenvolvimento e adaptar o que necessita ser feito, se a quantidade de atividades a ser feita é menor. Adicionalmente, projetos menores possuem mais chances de obter melhor execução quando comparado a projetos grandes (DYBÅ e DINGSØYR, 2008).

Figura 6.4: Custos Médio (em Milhões) e Quantidade de Projetos em Execução

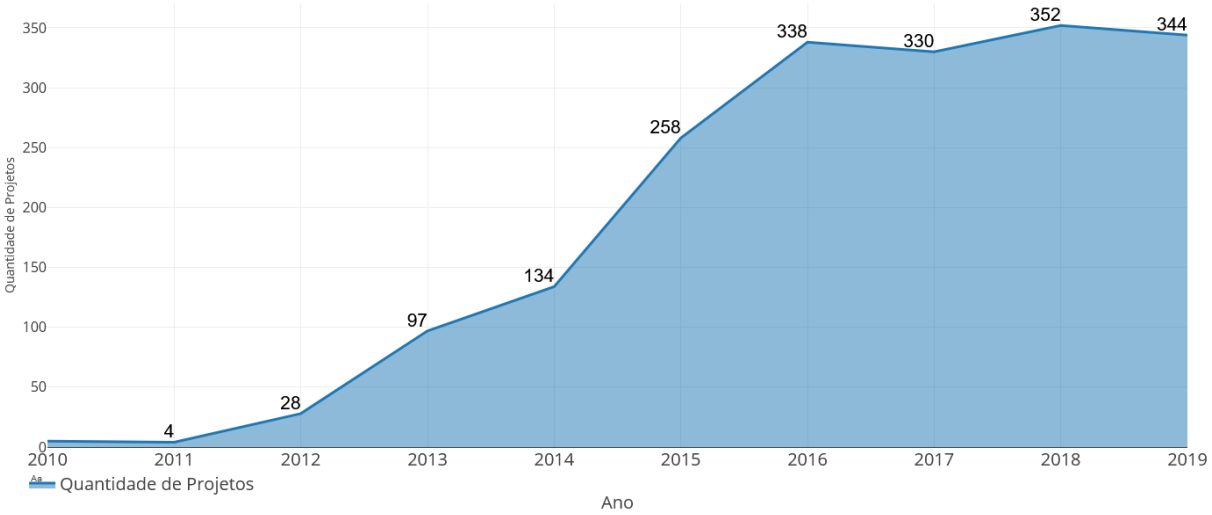


Fonte: O Autor (2020)

6.1.2 Entregas de Projetos

Com a quebra de projetos em partes menores, torna-se possível a entrega de funcionalidades do sistema em um menor intervalo de tempo. O gráfico da figura 6.5 mostra a quantidade de projetos movidos para produção (ou seja, entregues para uso do cliente) ao longo dos anos.

Figura 6.5: Quantidade de Projetos Entregues Anualmente



Fonte: O Autor (2020)

Percebe-se que de 2010 para 2012 o número de projetos entregues anualmente aumenta em 4 vezes. Posteriormente, de 2012 para 2015, temos um aumento aproximado de 9 vezes. Ou seja, em um período de 5 anos a quantidade de valor entregue tornou-se 36 vezes maior, mesmo estando em uma época de transição da categorização dos projetos (como notado na seção 6.1).

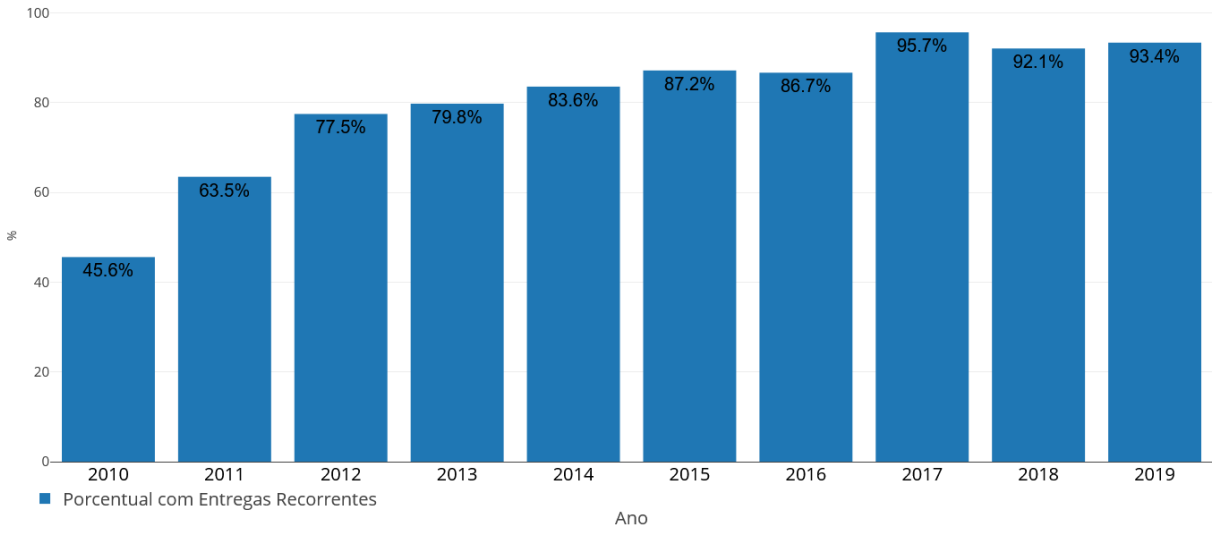
Após este momento de transição, o número de entregas anual atinge o patamar próximo de uma entrega por dia em média, um expressivo aumento quando comparado aos resultados do início da década. MAHANTI (2006) explica que trocar a maneira de se trabalhar afeta negativamente a capacidade de produção das partes envolvidas, devido a curva de aprendizado. Isto pode explicar o motivo do lento crescimento do número de projetos entregues, que indica a existência de um período de adaptação.

6.1.3 Incrementabilidade e Iteratividade

Na seção anterior verificou-se que a entrega de valor passou a ser mais constante. No entanto, foi abordada somente a entrega final e conclusiva de um projeto. No contexto de metodologias ágeis é imprescindível a tentativa de entrega prematura de valor, visando a coleta de sugestões e experiências de usuários, resultando em um produto melhor alinhado com expectativas existentes (COCKBURN e HIGHSMITH, 2001). Para isto, foi medido para quantos projetos em execução, dado um determinado momento do ano, haviam *releases* constantes, ou seja, com funcionalidades prontas e intervalos máximos de 6 meses (figura 6.6).

Percebe-se que no primeiro ano o percentual de projetos ágeis é similar a quantidade de projetos com entregas de funcionalidades constantes, cerca de 40%. Mas, vale a pena relembrar que somente 62 projetos foram iniciados naquele ano (figura 6.4). No ano de 2018 somente 4% dos projetos em execução não possuem entregas recorrentes. Considerando que aproximadamente 23% dos projetos em execução deste ano foram classificados como Cascata (conforme a figura

Figura 6.6: Projetos com *Releases* Recorrentes

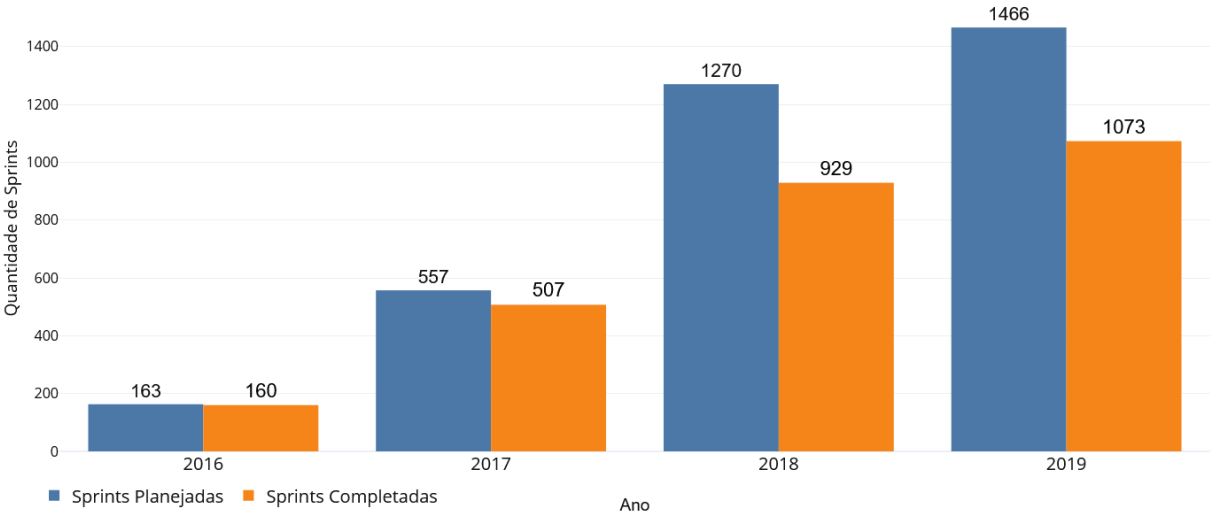


Fonte: O Autor (2020)

6.1), podemos ponderar se até mesmo os projetos não categorizados como ágeis passam a herdar princípios desta metodologia.

Tardiamente, em 2016, alguns dos projetos em andamento começaram a reportar a quantidade de iterações (ou *sprints*) iniciadas e terminadas (figura 6.7). Embora poucos anos tenham sido reportados, a tendência de alta do gráfico também indica uma adoção generalizada de times à modelos ágeis, sustentando a ideia de que as percentagens relativas as categorizações de projetos (figura 6.1) não são somente números a serem apresentados à gerência, e, sim, uma confirmação do que ocorre na prática.

Figura 6.7: Número de *Sprints*



Fonte: O Autor (2020)

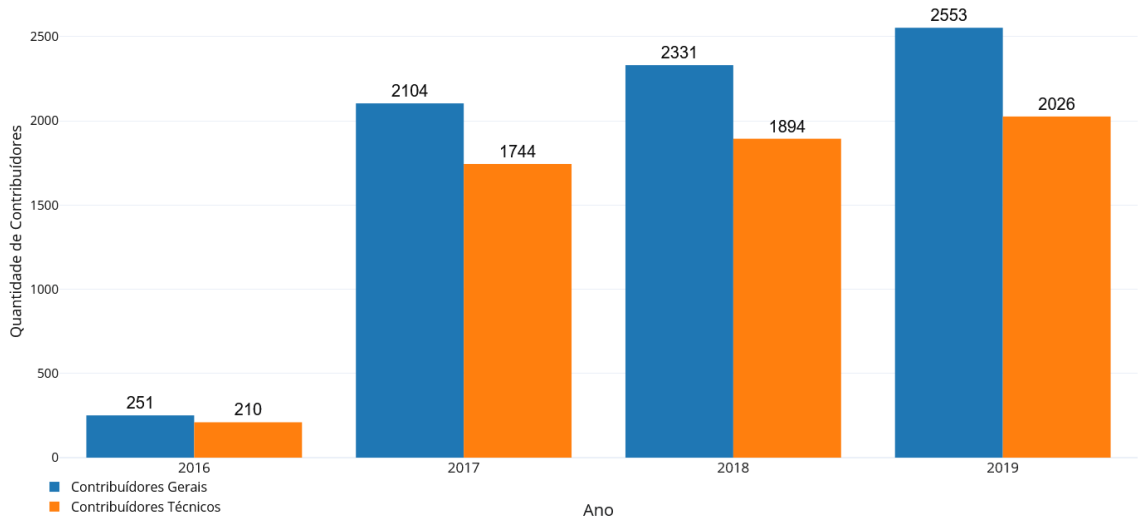
Uma *sprint* completa acarreta em ter todas atividades à ela relacionadas, completas. Portanto, as barras em azul representam iterações onde ao menos uma atividade não foi terminada de maneira integral.

PLANDEK (2021) afirma que times altamente funcionais utilizando *SCRUM* (um tipo específico de método ágil), obtêm 85% ou mais de suas *sprints* completadas. Na figura 6.7 temos um percentual de completude das *sprints* de 98%, 91%, 73% e 73% nos anos apresentados. Conseguimos, então, enxergar que o percentual de iterações completadas deixa de atingir um patamar ideal conforme o número de *sprints* realizadas anualmente cresce. Mas, se obtermos a média ao longo de todos os anos, obtemos um valor de 83% de completude, bem próximo do que é considerado ideal.

6.1.4 Colaboratividade

Junto à quantidade de iterações realizadas por projeto, em 2016, iniciou-se o hábito de reportar o número pessoas participantes de um projeto, sejam estas contribuidoras técnicas ou não. Esta distinção ocorre da seguinte maneira: os contribuidores técnicos são classificados como aqueles que contribuem com a realização de atividades com complexidade tecnológica como codificação ou projeto (seja criar uma logo, ou o esquema de um banco de dados, por exemplo). Os demais são os que realizam atividades não inerentemente relacionadas à algum desenvolvimento técnico, sendo estes, gerentes de projeto, ou usuários realizando testes, por exemplo.

Figura 6.8: Número de Contribuidores



Fonte: O Autor (2020)

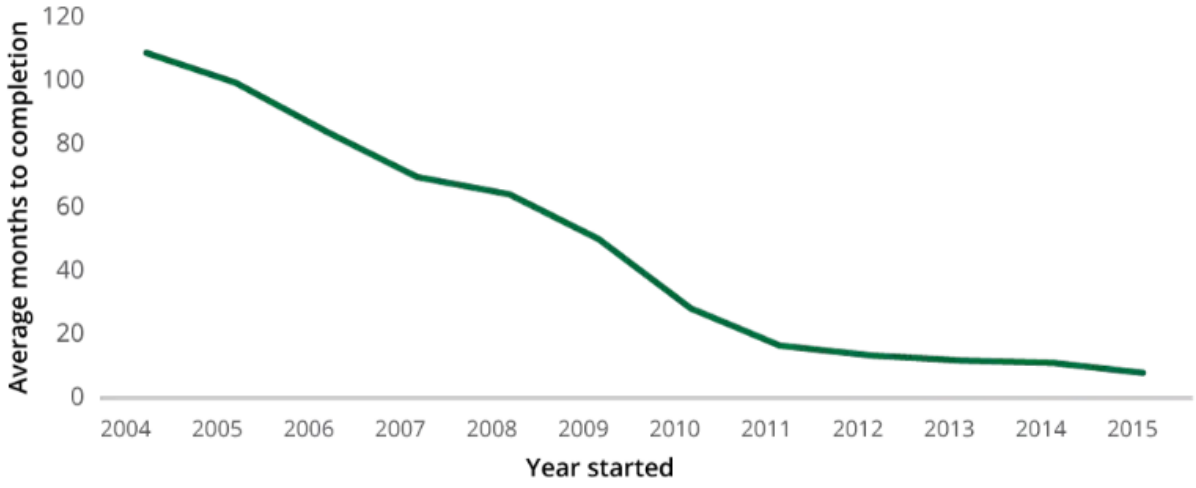
É possível enxergar na figura 6.8 que o número de contribuidores não técnicos constantemente excedeu seu grupo complementar em todas ocasiões nos últimos anos. Também, cada vez mais a diferença entre estas quantidades torna-se maior, conforme os projetos passam a reportá-los. Isto indica, possivelmente, um alto grau de envolvimento de usuários e partes interessadas, crucial para a geração de sugestões significativas, que, por sua vez, habilita a adaptabilidade do projeto para um produto que atenda às necessidades existentes. Logo, vemos conformidade com a subseção 4.1.5, onde se é afirmado a importância do envolvimento das partes interessadas no desenvolvimento de um projeto de *software*.

6.2 TEMPO DE ENTREGA

Como mencionado, uma das falhas do modelo Cascata para o desenvolvimento de *software* é o consequente atraso em entregas devido às características desta metologia, que levam o projeto na direção de ser composto por grandes blocos de código, complexos, altamente acoplados, difíceis e custosos de realizar-se operações (BROOKS, 1987).

Um estudo realizado pelos consultores VIECHNICKI e KELKAR, embora com algumas agências públicas norte americanas a mais que aquelas pertencentes ao escopo deste trabalho, serve de parâmetro para comparar os resultados do inicio dos anos 2000 com o final dos anos 2010. No ano de 2004, antes de "Ágil" ser reconhecido como uma possível opção no contexto das agências aqui estudadas, demorava-se, em média, 108 meses para entregar um projeto devidamente finalizado (VIECHNICKI e KELKAR, 2017). Pode-se observar, na figura 6.9, uma melhora constante, até que ao final da década, em 2009, temos por volta de 50 meses para realizar uma entrega.

Figura 6.9: Tempos Médios de Entrega no Governo Americano (2004-2015)



Fonte: VIECHNICKI e KELKAR (2017)

Também foi avaliado a quantidade de projetos categorizados como Ágeis, neste período de tempo. VIECHNICKI e KELKAR apontam que entre 2002 e 2009, este valor nunca chegou a 30% (figura 6.10).

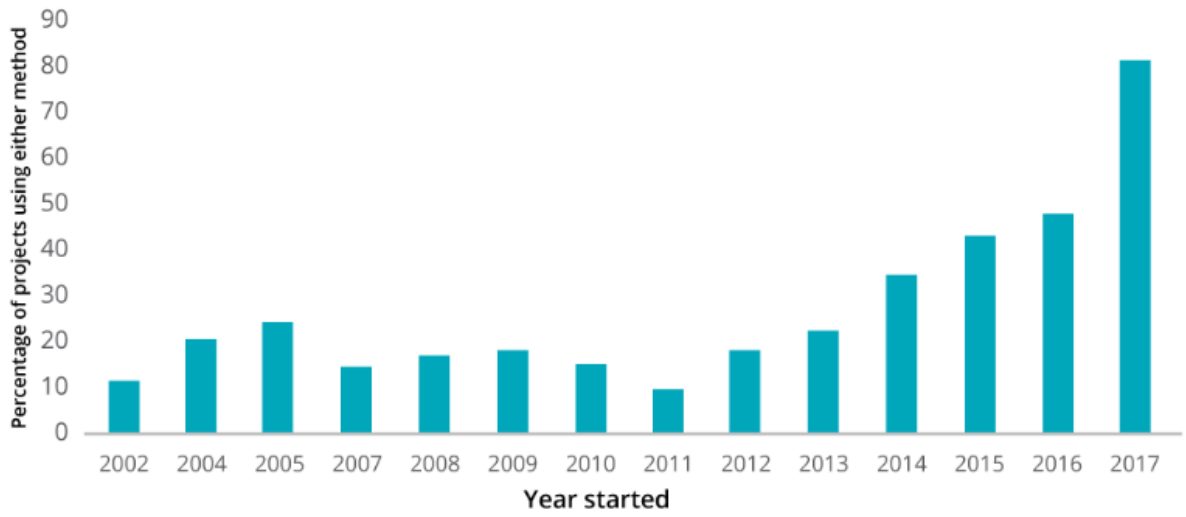
Os autores afirmam também, que a diminuição no tempo de entregas pode ter sido ocasionado devido a uma natural migração para lidar com projetos menores ao longo do tempo, já que métodos ágeis não eram ainda muito difundidos neste período.

Na década seguinte (2010-2019) ocorre um rearranjo radical do tempo de entrega durante os anos para os departamentos aqui estudados. No primeiro ano da década, têm-se um resultado condizente com os gráficos anteriores, cerca de 41 meses para a entrega ser realizada.

No ano seguinte, 2011, o que vemos na figura 6.11 é uma antecipada adequação as orientações da OMB, com o tempo de entrega caindo para aproximados 18 meses. Este valor foi o teto durante o resto da década, e, conforme os anos se passaram e a adoção de metodologias ágeis aumentou, o intervalo foi diminuindo.

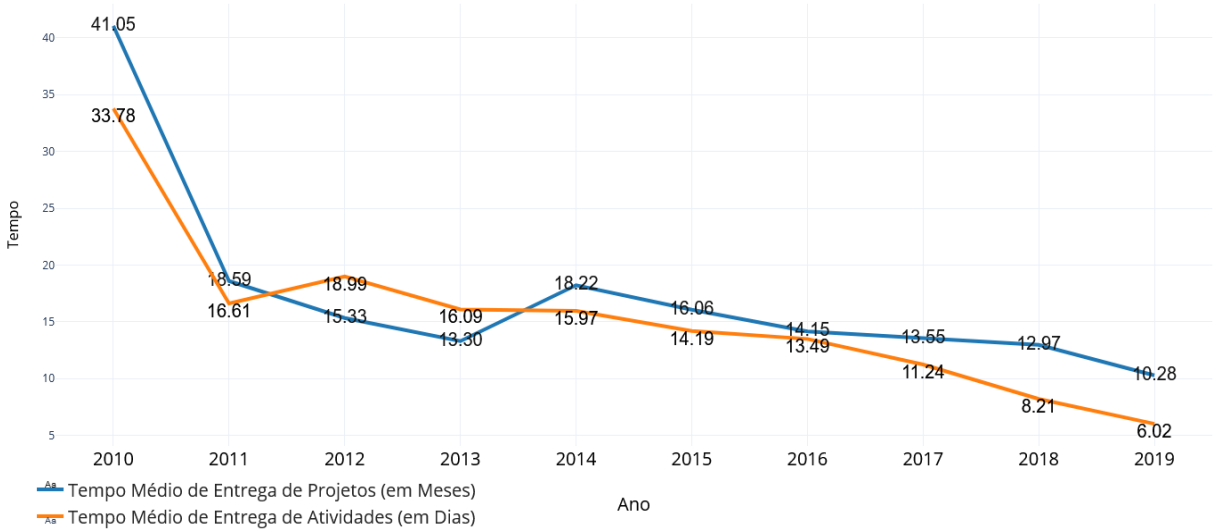
Este comportamento é possível graças ao atenuamento do tempo para resolução das atividades atreladas aos projetos. Uma atividade, como foi citado, é definida como uma tarefa que leva a resolução do projeto (OMB, 2020), sendo que todo projeto deve ter ao menos uma

Figura 6.10: Projetos Ágeis no Governo Americano (2002-2017)



Fonte: VIECHNICKI e KELKAR (2017)

Figura 6.11: Tempo Médio para Entrega de Projetos (meses) e Atividades (dias)



Fonte: O Autor

atividade dentro de seu escopo, e que uma atividade pode ter consigo várias outras atividades em sua composição.

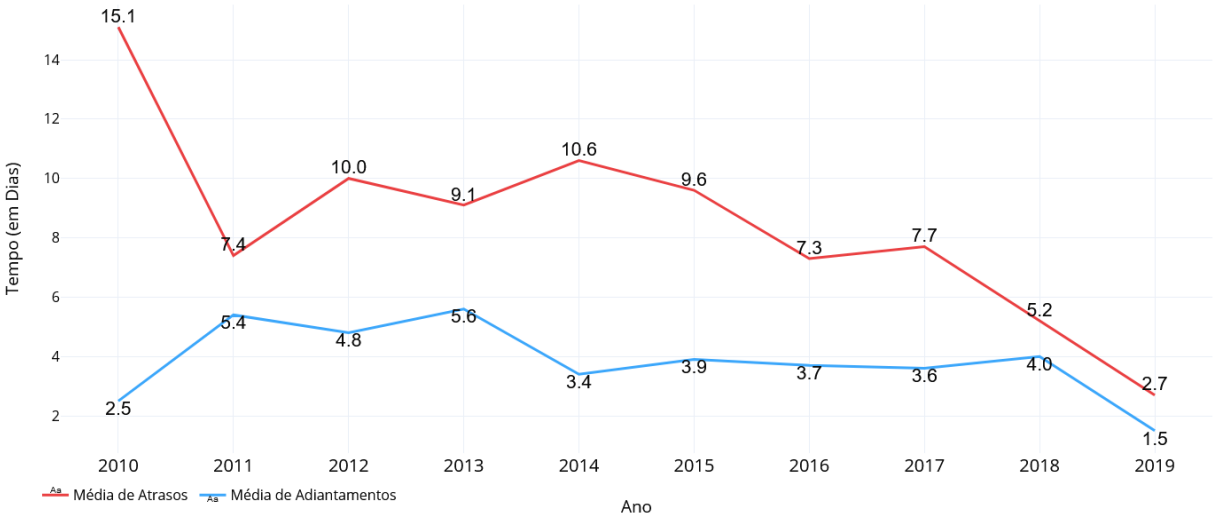
De maneira análoga ao tempo de entrega dos projetos, no início da década de 2010 temos um valor quase duas vezes maior que o próximo valor máximo do gráfico. Nos anos seguintes, o tempo médio de conclusão de atividades passa a diminuir durante todos os anos.

6.2.1 Adiantamentos e Atrasos

Por parte de tarefas classificadas como atividades, foi verificado que o tempo de entrega destas diminui constantemente durante todos anos aqui mencionados. Partindo deste resultado, pode-se imaginar que o tempo de atraso sofrido pelas atividades tornou-se menor. O tempo médio de atraso é calculado pela diferença, em dias, da data planejada de entrega, contra a data

em que a entrega da atividade foi realizada, para valores positivos. Na figura 6.12 podemos ver que sim, de fato, o atraso diminuiu ao longo dos anos, inclusive de maneira similar ao tempo de entrega de projetos e atividades, com um pico no ano de 2010 seguido de uma tendência de diminuição:

Figura 6.12: Média de Atrasos e Adiantamentos na Entrega de Atividades



Fonte: O Autor

Já o tempo médio de adiantamento é calculado da mesma forma que o de atraso, embora só levando em conta a média dos valores negativos. Conforme a passagem dos anos e a maior adoção de metodologias ágeis dentre os departamentos, as atividades passam a ser entregues em dias mais próximos à data planejada inicialmente. Ou seja, as atividades passam a ser entregues em datas menos precoces.

Como observado na seção que trata das limitações das metodologia ágeis (seção 4.1.2), busca-se abordar o trabalho a ser realizado com base na capacidade do time, na complexidade das tarefas e no tempo disponível. Logo, a menor frequência de entregas prévias ao prazo estimado é, na verdade, um sinal de que há, de fato, um bom controle da distribuição de trabalho através de contribuidores nas organizações estudadas.

6.3 ESTIMATIVAS DE CUSTO

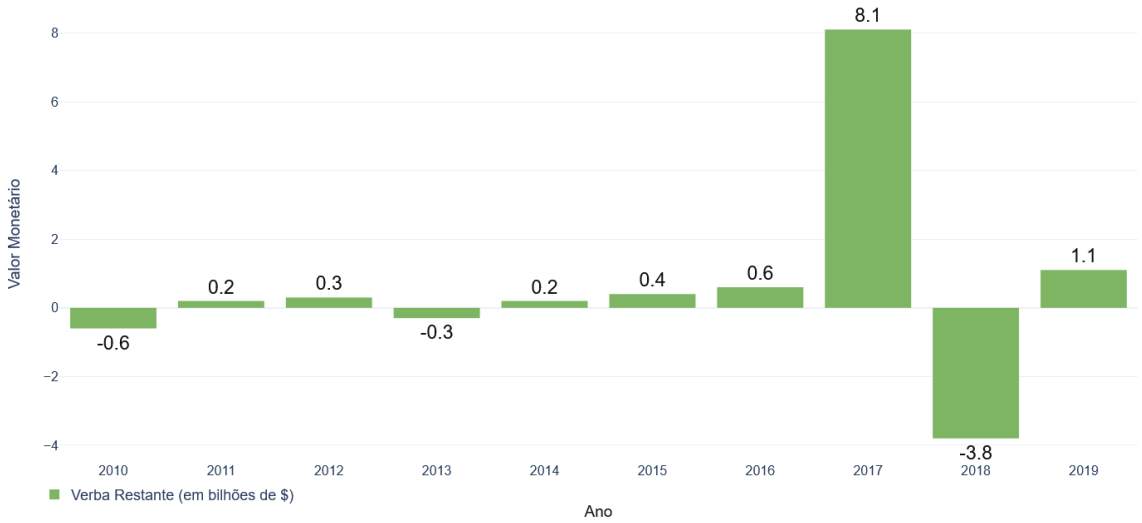
Como mostrado na figura 6.4, o custo médio de projetos diminuiu em um fator aproximado de 3 vezes entre o valor máximo e mínimo. No entanto, pode-se observar que grande parte desta diminuição de gastos é relacionada ao aumento do número de projetos. Obviamente, com um número muito maior de possíveis recipientes de verba governamental, o dinheiro distribuído entre os demais projetos é feito em quantias menores.

Como visto na subseção 3.1 produtos desenvolvidos sobre o modelo cascata apresentavam estimativas ineficientes de tempo e custos de entregas. Anteriormente, mostramos gráficos que indicam uma possível melhora nas estimativas de tempo (figura 6.12). Como sabemos, praticamente a cada ano que passa, os projetos passam a ser, em média, entregues em prazos mais próximos ao que foi inicialmente imaginado.

Visando abordar a eficiência em planejamento de gastos, a figura 6.13 mostra o quão efetiva as estimativas de gasto foram ao longo dos anos, onde cada coluna corresponde à diferença entre o total de gastos planejados e o total de gastos realizados para um projeto iniciado em

um determinado ano. Aqui, valores positivos representam um excedente de verba, já valores negativos representam gastos maiores que a verba disponível.

Figura 6.13: Variação do Uso de Verba



Fonte: O Autor

No primeiro ano os 15 departamentos americanos extrapolam a verba estimada em 0.6 bilhões de dólares, o que, com base nos demais valores dos próximos anos, parece completamente aceitável, visto que durante vários anos valores similares foram poupados, como 2016. O que temos é uma quebra de consistência durante 2017 e 2018, um dos picos de adoção de métodos ágeis com base na figura 6.1. Os demais anos possuam variações similares em percentual de projetos ágeis, logo, nenhuma alteração neste percentual é suficiente para apontá-la como possível causa desta discrepância de gastos.

Apesar de 2017 apresentar gastos muito menores que inicialmente planejado, métodos ágeis buscam realizar estimativas condizentes com o gasto realizado. Ou seja, o ideal é que a verba restante seja o mais próximo de 0 o possível. Então, embora de um ponto de vista financeiro gastos menores sejam bons, perante a visão de um gerente de projetos isto indica uma incapacidade de se estimar dados corretamente, um ponto negativo que deve ser corrigido. O mesmo raciocínio é aplicado para 2018.

É difícil afirmar o quão eficiente o uso de metodologias ágeis é para estimar o uso de recursos financeiros, devido as grandes variações de valores apresentados nos anos de 2017 e 2018 (figura 6.13). Podemos observar, no entanto, que o uso destes métodos acarretou em menores custos no ciclo de desenvolvimento de *software*. A figura 6.1 mostra que a partir de 2014, métodos ágeis tornam-se o método predominante utilizado nos departamentos estudados. Ao observamos a figura 6.13, enxergamos que, de 2014 em diante, os gastos esperados são menores em todos anos, exceto 2018.

7 CONCLUSÃO

O trabalho aqui apresentado demonstrou as implicações da adoção de metodologias ágeis em detrimento ao uso do modelo cascata, provendo dados que evidenciam como se dá este processo assim como algumas de suas consequências.

Para tanto, foi definido o que é uma metodologia no contexto de desenvolvimento de *software*, o que é uma metodologia do tipo cascata, ágil, e as características pertinentes de cada, cumprindo o primeiro objetivo específico aqui listado.

Posteriormente, explicamos a origem e características dos dados que servem para ilustrar a transição do modelo cascata para algum método ágil em diversos departamentos do ramo executivo do governo americano. No início do capítulo 6 é demonstrado que, de fato, o modelo cascata passa a ser menos utilizado que as demais opções já em 2011.

Com esta troca de abordagens, é mostrado adiante, no capítulo 6, como a maneira de se trabalhar muda nos departamentos estudados. Projetos passam a ter tamanhos e custos menores, possibilitando a execução e entrega concorrente destes. Adicionalmente, colaboração passa a ter maior enfoque, com o número de contribuidores aumentando a cada ano.

Por fim, os departamentos passaram a apresentar rendimentos melhores após a transição para métodos ágeis. O tempo médio de entrega de projetos e atividades diminuiu em, aproximadamente, 4 vezes. As estimativas de completude de tarefas também passam a ser muito mais precisas. Tratando-se de custos, temos um excedente de verba no somatório de todos anos em que ágil é a metodologia predominante.

Logo, por meio dos resultados apresentados sobre a migração entre o modelo cascata para métodos ágeis, é possível concluirmos que tal processo muda a forma com que os departamentos trabalham, que, por sua vez, gera resultados positivos. Ademais, o uso de métodos ágeis levou o governo americano a criação de novas métricas, o que podem resultar em melhor capacidade de vistoria de seu desempenho. Este trabalho apresenta limitações acerca do quão essencial a disponibilidade de verba é para a implementação adequada da troca de metodologias. Contudo, consegue-se enxergar que o uso de ágil levou à economia de gastos, algo não observado historicamente com o uso do método cascata.

Para trabalhos futuros recomenda-se o estudo do rendimento de organizações não aplicantes de alguma metodologia específica em detrimento de àquelas que usam ágil. Adicionalmente, é sugerido o estudo quantitativo sobre a preferência de contribuidores técnicos acerca da metodologia que estes utilizam, ágil ou não.

REFERÊNCIAS

ADENOWO, A. A. e ADENOWO, B. A. (2013). Software engineering methodologies: a review of the waterfall model and object-oriented approach. *International Journal of Scientific & Engineering Research*, 4(7):427–434.

BASILI, V. (1985). Quantitative evaluation of software methodology. Relatório Técnico TR-1519, University of Maryland, College Park, Maryland.

BECK, K. (1999). Embracing change with extreme programming. *Computer*, 32(10):70–77.

BOEHM, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5):61–72.

Booch, G. (2018). The history of software engineering. *IEEE Software*, 35(5):108–114.

BOURQUE, P., DURPUIS, R., ABRAN, A., MOORE, J. W. e TRIPP, L. (2004). *Guide to the Software Engineering Body of Knowledge*. IEEE.

BRIGGS, L. (2007). Tackling wicked problems: A public policy perspective. *Australian Public Service Commission*, 1:1–37.

BROOKS, F. (1987). No silver bullet: Essence and accidents of software engineering. *Computer*, 20(4):10–19.

COCKBURN, A. e HIGHSMITH, J. (2001). Agile software development: the business of innovation. *Computer*, 34(9):120–127.

COLLIER, K. (2012). *Agile analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing*. Addison-Wesley.

CONKLIN, J. (2003). *Dialogue Mapping: Building Shared Understanding of Wicked Problems*. Wiley.

CUSUMANO, M. A. e SMITH, S. A. (1995). Stanley a. beyond the waterfall: Software development at microsoft. Relatório Técnico 3844-BPS-95, Massachusetts Institute of Technology (MIT) Sloan School of Management, [S.l.].

DEGRACE, P. e STAHL, L. (1990). *Wicked problems, Righteous Solutions*. Yourdon Press.

Dima, A. M. e Maassen, M. A. (2018). From waterfall to agile software: Development models in the it sector, 2006 to 2018. impacts on company management. *Journal of International Studies*, 11(2):315–326.

DYBÅ, T. e DINGSØYR, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50:833–859.

FOWLER, M. e HIGHSMITH, J. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

Hewlett Packard (2017). Agile is the new normal. Relatório Técnico 4AA5-7619ENW, Hewlett Packard Enterprise Development LP., [S.l.].

LAPHAM, M. A., WILLIAMS, R., HAMMONS, C., BURTON, D. e SCHENKER, A. (2010). Considerations for using agile in dod acquisition. Relatório Técnico CMU/SEI-2010-TN-002, Carnegie Mellon University, [S.l.].

LARMAN, C. e BASILI, V. (2003). Iterative and incremental development: A brief history. *IEEE Computer*, 36:47–56. Cover Feature.

LUTZ, R. R. (1993). Analyzing software requirements errors in safety-critical, embedded systems. Em *Proceedings of the IEEE International Symposium on Requirements Engineering*, páginas 126–133, Pasadena - California.

MAHANTI, A. (2006). Challenges in enterprise adoption of agile methods - a survey. *Journal of Computing and Information technology*, 14(3):197–206.

McCORMICK, M. (2012). eAgile Project Life Cycle. `http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf`. Acesso em: 02 de Jul. 2022.

MILLER, G. (2001). The characteristics of agile software processes. *Technology of Object-Oriented Languages, International Conference on*, 1:0385–0387.

MORAN, A. (2014). *Agile Risk Management*. Springer.

OMB (2020). FY 2020 IT Budget – Capital Planning Guidance. `https://www.whitehouse.gov/wp-content/uploads/2018/06/fy-2020-it-budget-guidance.pdf`. Acesso em: 06 de Jul. 2022.

PALMQUIST, M. S., LAPHAM, A., MILLER, S., CHICK, T. e OZKAYA, I. (2013). Parallel worlds: Agile and waterfall differences and similarities. Relatório Técnico CMU/SEI-2013-TN-021, Carnegie Mellon University, [S.l.].

PRESSMAN, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave macmillan.

Randell, B. (1996). The 1968/69 nato software engineering reports. *History of software engineering*, 37.

RITTEL, H. e WEBBER, M. (1973). Dilemmas in a general theory of planning. *Policy sciences*, 4:155–169.

ROYCE, W. (1970). Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESCON*, páginas 1–9.

SOARES, M. (2004). Metodologias Ágeis: Extreme programming e scrum para o desenvolvimento de software. *Revista Eletrônica de Sistemas de Informação*, 3(1):64–76.

STANDISH GROUP (1995). The Chaos Report. `https://www.csus.edu/indiv/r/rengstorffj/obe152-spring02/articles/standishchaos.pdf`. Acesso em: 02 de Jul. 2022.

STOICA, M., MARINELA, M. e GHILIC-MICU, B. (2013). Software development: Agile vs. traditional. *Informatica Economicã*, 17(4):64–76.

VIECHNICKI, P. e KELKAR, M. (2017). Agile by the numbers: a data analysis of agile development in the us federal government. *Agile in Government: A Playbook from the Deloitte Center for Government Insights, Deloitte*, páginas 42–47.

Wirth, N. (2008). A brief history of software engineering. *IEEE Annals of the History of Computing*, 30(3):32–39.

ZELKOWITZ, M. (2009). *Advances in Computers: Emerging Technologies*. Academic Press.