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

# Percepções de Práticas Ágeis em Desenvolvimento de Software: Benefícios e Desafios

Alan Saulo da Costa Mazuco

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

Orientador
Prof. Dr. Rodrigo Bonifácio de Almeida

Brasília
2017

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

SAL319p

Saulo da Costa Mazuco, Alan
Percepções de Práticas Ágeis em Desenvolvimento de Software: Benefícios e Desafios / Alan Saulo da Costa Mazuco; orientador Rodrigo Bonifácio de Almeida. -- Brasília, 2017.
194 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2017.

1. Investigação das práticas ágeis de desenvolvimento de software nas empresas. 2. Estudo Terciário sobre metodologias e práticas ágeis nas empresas do Distrito Federal. 3. Pesquisa de campo sobre percepções de práticas ágeis de desenvolvimento de software. I. Bonifácio de Almeida, Rodrigo, orient. II. Título.

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

# Percepções de Práticas Ágeis em Desenvolvimento de Software: Benefícios e Desafios

Alan Saulo da Costa Mazuco

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

Prof. Dr. Rodrigo Bonifácio de Almeida (Orientador)
CIC/UnB

Prof. Dr. Edna Dias Canedo
UnB

Prof. Dr. Gustavo Henrique Lima Pinto
UFPA

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

Brasília, 04 de agosto de 2017

# Dedicatória

À minha esposa, que esteve ombreada comigo na consecução desse trabalho, me dando forças e fôlego para não esmorecer. Seu auxílio foi contumaz importante, motivando e despertando a vontade de vencer. Sua dedicação e atenção nos momentos mais críticos me inspirou os pensamentos de uma forma bem especial, levando-me a novos horizontes no despertar do caminho do saber.

# Agradecimentos

Em primeiro lugar a Deus, que é o princípio de tudo e causa primária de todas as coisas. À minha esposa e filha, que suportaram a minha ausência, fruto de horas e meses na elaboração desta obra.

Um agradecimento a todos os meus professores de mestrado que me fizeram uma nova pessoa, em especial ao meu orientador, com a sua paciência, sendo educador, compreensivo, e às vezes rígido nos momentos em que houve a necessidade, me conduzindo ao caminho certo da aprendizagem.

Agradeço também a todos os meus amigos que me deram uma força para que eu pudesse levar a cabo este projeto, pelos incentivos e pela credibilidade.

# Resumo

O presente trabalho tem por finalidade apresentar as metodologias e as práticas ágeis mais comumente utilizadas nas indústrias de software, sua evolução desde o Manifesto Ágil até os dias atuais, fatores de sucesso, possibilidades e limitações. Mostra que existem várias evidências do crescimento dessas práticas nos últimos anos e que a comunidade acadêmica vem realizando vários estudos para mostrar empiricamente que elas podem ajudar no processo de desenvolvimento de software. Apresenta uma fundamentação teórica embasada em um estudo terciário considerando vários artigos a respeito da utilização de práticas ágeis na comunidade científica. As práticas são apontadas à luz de pesquisa de campo referentes às metodologias DSDM, Crystal, FDD, Lean, Kanban, XP e Scrum, consideradas as metodologias ágeis mais atuantes hoje, sob a ótica das evidências metodológicas, casos de sucesso e principais desafios e situações. Apresenta também os resultados de uma pesquisa de campo realizada em indústrias de software no Distrito Federal e entorno, trazendo novas informações para profissionais da Engenharia de Software, em um esforço para mostrar essas tecnologias e apontar a sua evolução, bem como seus benefícios e desafios. Os resultados da pesquisa mostram que as metodologias ágeis continuam a ser de interesse para os pesquisadores no futuro e que diversos projetos, hoje em andamento nas indústrias, vêm cooperar com a investigação apresentando um estudo importante para a Engenharia de Software.

**Palavras-chave:** Metodologia ágil, Engenharia de Software, XP, Scrum

# Abstract

This study aims to present the methodologies and agile practices, their evolution since the Agile Manifesto to the present day, success factors, possibilities and limitations. It presents several evidences about the growth of agile practices that have occurred in recent years, and that the academic community has been conducting several studies that empirically show that these practices help in the software development process. It presents a theoretical framework grounded in a tertiary study considering several articles on the use of agile practices in the scientific community. The practices are identified in the light of empirical research related to methodologies, such as DSDM, Crystal, FDD, Lean, Kanban, XP and Scrum, considered the most active agile methodologies today, from the perspective of methodological evidence, case studies and key challenges and situations. Here we also present the results of a field research carried out in several software industries in Federal District - Brazil and surroundings, bringing new information for software engineering practitioners, in an effort to show these technologies and point their evolution as well as its benefits and shortcomings, and paint a frame with the picture of them on the Brazilian software industry. The results of our survey shows that agile methodologies continue to be of interest to researchers in the future and that many projects currently in progress in industries, come cooperate with the investigation presenting an important study for Software Engineering.

**Keywords:** Agile methodology, Software Engineering, XP, Scrum

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

O presente trabalho tem o propósito de investigar empiricamente as evidências, benefícios e os desafios de algumas práticas ágeis comumente empregadas na Engenharia de Software em seu contexto mais atual. Traz uma abordagem em forma de estudo terciário na evidência das metodologias que empregam essas práticas, casos de sucesso e principais desafios por que passam algumas organizações investigadas, como consequência direta desse processo. Este capítulo apresenta a contextualização da pesquisa, os objetivos do trabalho, as questões de pesquisa e a forma como o estudo está organizado.

## 1.1 Contextualização

De acordo com o Tribunal de Contas da União [149], as metodologias voltadas para o desenvolvimento de software surgiram em um contexto tecnológico muito diferente do que hoje se apresenta, baseado mais em soluções para computadores de grande porte com elevados custos de manutenção devido ao limitado acesso às tecnologias vigentes.

Hoje, esse cenário se apresenta de forma bem diferente, pois houve uma melhora significativa no aprendizado de diversas tecnologias com ofertas de mão-de-obras bastante qualificadas. De acordo com Brietzke e Rabelo [24], as organizações têm buscado melhorar os seus processos no intuito de produzir com maior qualidade, em particular o setor público que nos últimos 20 anos vem recebendo constante pressão para melhorar o seu desempenho para atender melhor a sociedade contemporânea moderna [36].

Um trabalho de revisão sistemática da literatura realizado em 2008 por Dybå e Dingsøyr [44], argumenta que o desenvolvimento ágil atrai um enorme interesse da indústria de software em todo o mundo. Nos EUA e na Europa, revela a pesquisa, 14% das empresas já começavam a usar métodos ágeis, e 49% tomando ciência sobre eles e interessadas em adotá-los. Ressalta ainda que em apenas seis anos, desde 2001, a *Agile Conference* cresceu atraindo um maior público que a maioria das outras conferências em Engenharia

de Software. De acordo com Ambler [3], dentre os vários métodos ágeis empregados pelas empresas, a Programação Extrema [16], e o Scrum [132], ainda são os mais conhecidos e adotados, sendo que Scrum entrou em maior evidência nos dias de hoje.

Atualmente, conforme ressaltam Dikert et al. [40], o panorama se mantém, onde os autores apresentam uma nova revisão sistemática mostrando uma tendência das empresas no sentido de adotar metodologias ágeis:

> "De acordo com o último levantamento, 62% das 4000 empresas entrevistadas em todo o mundo tinham mais de uma centena de pessoas na sua organização e que 43% de todos os entrevistados trabalharam em organizações de desenvolvimento, onde mais da metade das equipes eram ágeis."

Por outro lado, Ayed et al. [10] identificaram a existência de alguns estudos na literatura que relatam a adoção e adaptação de práticas ágeis, porém, a maioria deles sem a evidência de métricas quantitativas, o que dificultaria a tomada de decisão na opção pela adoção de práticas ágeis. Chama a atenção a diversidade das práticas ágeis evidenciadas nesses estudos, algumas delas defendidas por mais de uma metodologia, e os principais desafios como consequência da adoção delas, uma das motivações para a realização deste trabalho.

As metodologias ágeis de desenvolvimento de software vêm ganhando um enorme espaço, conforme apresentado em Parsons et al. [112], que relatam que em algumas circunstâncias, tais metodologias podem oferecer melhores resultados em determinados projetos.

## 1.2 O Problema

Os poucos relatórios sistemáticos e atualizados sobre as práticas ágeis de desenvolvimento de software e os consequentes obstáculos para a sua adoção em projetos corporativos podem dificultar a tomada de decisões nas organizações, no sentido de que elas explorem essa adoção. Por exemplo, no Governo Federal, em acórdão recentemente emitido pelo Tribunal de Contas da União [157], aponta a existência de diversos riscos nesse tipo de contratação, o que levou o órgão a concluir que:

> "O TCU determinou à sua Secretaria de Fiscalização de TI (Sefti) que aprofunde os estudos, inclusive com realização de fiscalizações, se forem necessárias, visando a identificar, com maior precisão, os riscos envolvidos na utilização dos métodos ágeis na contratação de desenvolvimento de software pela Administração Pública Federal, segundo o modelo atual de contratação, de maneira a orientar adequadamente os jurisdicionados do Tribunal."

Em contrapartida ao problema, em uma apresentação no 1º Seminário de Metodologia Ágil do SISP (Sistema de Administração dos Recursos de Tecnologia da Informação) [68], foi enfatizado que:

> "A adoção de métodos ágeis nas contratações públicas para desenvolvimento de software mostra-se viável, devendo-se identificar e mitigar os riscos genéricos a qualquer contratação e os específicos da utilização da metodologia em questão."

Em uma recente publicação da Revista do TCU, pelo artigo de Franco e Toledo [60], que buscou-se uma comparação das leis brasileiras com as americanas sobre o assunto, conclui-se que:

> "Atualmente, observa-se uma preocupação maior do legislador, principalmente o americano, em garantir que o software entregue atenda às necessidades requeridas e que também retorne maior ganho para a sociedade. Para isso, identifica-se no trabalho, que os pilares de sucesso para um projeto de criação de software: prazo, custo e qualidade, possuem uma melhora quando aplicada a metodologia ágil, faltando apenas o quarto pilar, escopo que, como definido dentro dessa metodologia, deve ser flexível buscando as necessidades do cliente."

Uma melhor compreensão com esses aspectos pode favorecer que uma metodologia ágil seja implantada de uma forma incremental, começando com as práticas que agregam mais valor de acordo com a percepção de empresas que já as utilizam de forma disseminada.

A implantação de projetos-piloto, conforme estudos de Melo e Ferreira [99], podem facilitar bastante essa adoção, por proporcionar uma visão bem ampla dos riscos e também dos ganhos, conforme relata o trabalho:

> "A Tabela [...] apresenta o ganho de produtividade dos pilotos ágeis (Ganho AGIL) em relação à produtividade média dos projetos tradicionais da organização. No piloto 1, houve um ganho de produtividade de 8,21%. Já no piloto 2 foi constatado um ganho de produtividade de 30,89% em relação à média da organização. O Projeto 1 era mais complexo (envolvia diversos cálculos com alta precisão) do que o Projeto 2. Além disso, foi o primeiro piloto de ágeis, o que sugere uma curva maior de aprendizado e adaptação. O projeto 2 pôde aprender com os erros do projeto 1. Por fim, a produtividade obtida pelo projeto 2 foi considerada satisfatória, entretanto a organização acredita que o ganho pode ser aumentado."

Por essas evidências, conclui-se que existe um certo temor das empresas brasileiras, especialmente as do setor público, em adotar as práticas ágeis de desenvolvimento de software, um problema que pode ser minimizado com a utilização de algumas técnicas apontadas, tais como os projetos-pilotos e equipes de estudos multidisciplinares.

## 1.3 Objetivos

### 1.3.1 Geral

Apresentar as metodologias e as práticas ágeis mais comumente adotadas nas empresas, seus fatores de sucesso e principais desafios, por meio da realização de estudo terciário e pela condução de uma pesquisa de campo, buscando essas evidências nas empresas de desenvolvimento de software.

### 1.3.2 Específicos

Para atender o objetivo geral, os seguintes objetivos específicos foram fixados:

1. Investigar as metodologias e as práticas ágeis mais evidentes na Engenharia de Software por meio de um Estudo Terciário, agrupando-as por afinidades;
2. Apontar os benefícios e os desafios das metodologias e práticas ágeis mais relatadas nas pesquisas empíricas;
3. Medir e descrever o nível de adoção e aceitação das práticas ágeis dentro das empresas com uma investigação conduzida por meio de entrevistas estruturadas;
4. Pesquisar os métodos empíricos em Engenharia de Software, em particular a condução de estudos terciários, em consonância com os resultados obtidos em entrevistas;
5. Consolidar o estudo empírico com a pesquisa de campo, mostrando os pontos fortes e fracos, problemas e relações com a produtividade.

## 1.4 Questões de Pesquisa

Os assuntos que norteiam o presente trabalho culmina em algumas indagações sobre as principais práticas ágeis utilizadas em Engenharia de Software bem como os desafios que poderiam surgir com a adoção delas. Por isto, este trabalho propõe as seguintes questões de pesquisa, a fim de fundamentar os seus estudos:

- (Q1) Que práticas ágeis estão sendo mais exploradas na literatura?
- (Q2) Qual a percepção das empresas sobre a adoção de práticas ágeis?
- (Q3) Qual a extensão da adoção de práticas ágeis nas empresas?

Para responder a primeira pergunta, realizou-se uma investigação por meio de um estudo terciário baseado nas referências apresentadas nos trabalhos de Dybå e Dingsøyr [44] e Dikert et al [40], este último, um trabalho realizado em 2016 sobre desenvolvimento de software em larga escala. Para responder a segunda e a terceira pergunta, realizou-se uma pesquisa de campo, visitas *in-loco* às principais fábricas de software sediadas no Distrito Federal, onde realizou-se entrevistas, bem como a análise de questionários enviados às empresas de software.

## 1.5 Organização do Trabalho

Este trabalho contém 4 capítulos além deste, constando de:

- **Capítulo 2**: Estudo Terciário em Engenharia de Software acerca do tema e o que dizem os principais pesquisadores sobre metodologias ágeis nas conferências, nos dias de hoje. Apresentação, em linhas gerais, de investigações realizadas sobre as metodologias ágeis mais comumente empregadas na Engenharia de Software, sob a forma de um Estudo Terciário.
- **Capítulo 3**: Apresenta a metodologia utilizada para a realização dos trabalhos e das atividades de pesquisa de campo desenvolvidas durante os estudos.
- **Capítulo 4**: Mostra a formatação, análise e discussão dos resultados obtidos na pesquisa de campo e questionários, além de discutir as ameaças à validade.
- **Capítulo 5**: Conclui-se com a análise geral dos resultados da pesquisa e trata dos trabalhos futuros.

O presente trabalho apresenta, também, dois apêndices que mostram as metodologias ágeis (Apêndice A) e as práticas ágeis mais empregadas nas empresas (Apêndice B), como forma de complementação, cuja leitura é opcional para o entendimento do estudo.

# Capítulo 2

# Pesquisa Empírica em Metodologias e Práticas Ágeis

A pesquisa empírica é a busca de dados relevantes e convenientes obtidos através da experiência, da vivência do pesquisador. Tem como objetivo chegar a novas conclusões a partir da maturidade experimental do(s) outro(s).

Assim podemos entender que a pesquisa empírica pode ser entendida como a coleta de dados a partir de fontes diretas (pessoas) que conhecem, vivenciaram ou têm conhecimentos sobre o tema, fato ou situação que causam diferenciação na abordagem e no entendimento dos mesmos, conduzindo a uma mudança, acréscimo ou alteração profunda ou relevante que não distorça, agrida ou altere o conteúdo principal mas sim, que o enriqueça e transforme em conhecimento de fácil compreensão e também sentindo-se atraído por tal.

Este capítulo apresenta as práticas ágeis sob a ótica de pesquisa empírica dentro da Engenharia de Software, fundamentando um **Estudo Terciário** realizado conforme as referências bibliográficas encontradas no trabalho de Dybå e Dingsøyr [44], no ano de 2008, e comparando o que evoluiu desde aquela época até os dias atuais, com o estudo das publicações de Dikert et al. [40], um trabalho realizado em 2016.

## 2.1 Rigor e Metodologia

### 2.1.1 Justificativa da Escolha dos Artigos

Nota-se que os dois trabalhos, Dybå e Dingsøyr (2008) e Dikert et al. (2016), possuem objetivos diferentes, enquanto o primeiro estuda o desenvolvimento de software em pequenas escalas industriais, o segundo foi centrado em sistemas de grande porte, considerados de larga escala industrial. Entretanto, não é objetivo deste trabalho estudar o desenvolvimento de software em questões de pequena ou larga escala, mas sim, o enfoque dado às práticas

ágeis adotadas pelas duas abordagens, que são as mesmas, e é nisto que este trabalho se concentra, na evidência dessas práticas ao longo dos anos que abrangem os dois estudos. Apenas para exemplificar o fenômeno, vemos os autores, em diferentes épocas, citando as mesmas práticas ágeis de desenvolvimento, algumas aparecendo mais, outras menos, conforme a evolução dos acontecimentos. De fato, no início, algumas metodologias ágeis foram relatadas como sendo de difícil implementação em projetos de larga escala, mas não impossível, conforme apresentado em Dybå e Dingsøyr [45] "XP parecia ser difícil de se introduzir em organismos complexos [...] mas os benefícios começavam a aparecer com a presença do cliente em colaboração."

Dickert et al. [40], em sua pesquisa de desenvolvimento de larga escala em 2016 observa que:

> "Era bastante comum que as organizações procurassem combinar métodos ágeis, especialmente Scrum com XP e Lean, todos em conjunto. Além disso, muitos casos mencionaram o uso de práticas XP, como Desenvolvimento Orientado a Teste e a Integração Contínua, mas sem indicar explicitamente que eram práticas do XP sendo executadas [...] pois muitas organizações consideravam que a combinação e personalização de práticas ágeis apareciam como uma evolução contínua."

Percebe-se, também, que os os problemas enfrentados pelas duas abordagens são os mesmos, quando os grupos manifestam interesse em migrar para metodologias ágeis. Veja-se outro relato em programação de larga escala de Dickert et al. [40]:

> "Durante um período de transformação, as objeções à mudança podem se tornar uma razão principal para perda de tempo e produtividade. Foram relatadas inúmeras razões para a resistência à mudança. [...] Em vários casos, a iniciativa de mudança veio da gerência e, quando apresentada de maneira ruim, tornou-se um mandato para o qual as pessoas não estavam receptivas. [...] A falta de apoio da gerência para a mudança e a falta de inclinação para mudar a cultura foram vistas como alguns dos problemas mais sérios na transformação."

E o relato de Dybå e Dingsøyr [45], em programação de pequena escala, em 2008, também não fugia muito disto:

> "O processo foi apresentado a uma equipe-piloto que trabalhou durante oito meses. [...] concluiu-se que a introdução do processo revelou-se difícil, devido a complexidade da organização. [...] Eles descobriram que os engenheiros foram motivados pelos princípios do desenvolvimento ágil, mas que o gerentes, inicialmente, tiveram medo e precisavam de ser treinados. [...] Estes tópicos têm sido negligenciados no passado por pesquisadores inspirados pelo velho paradigma, e, portanto, há um atraso de problemas de pesquisa a ser resolvido."

Mas, como citado por Dickert et al. [40], o desenvolvimento ágil evoluiu bastante de 2008 para cá, com maiores aceitações, onde já é possível uma maior percepção, veja-se:

> "[...] isto indica que parece existir um grande número de empresas que estão adotando metodologia ágil. Um tópico para pesquisas futuras seria estudar como os desafios e fatores de sucesso reconhecidos neste estudo são experimentados nas empresas: quais eles experimentam e quais eles consideram mais importantes."

E é assim que os dois trabalhos utilizados como base para esta pesquisa são úteis, sem se importar se os estudos são feitos para projetos de larga escala ou não, visto que dirigiu-se todo o esforço em colocar em evidência as práticas ágeis, e como elas estão hoje sendo relatadas nas diversas literaturas disponíveis da Engenharia de Software.

### 2.1.2 Abordagem sob a forma de Estudo Terciário

Proceder a uma revisão da literatura é uma importante parte de qualquer projeto de pesquisa que se tenha em mente. De acordo com Dyba [46], ao contrário de uma narrativa tradicional de avaliação, a *Systematic Review — SR* é um resumo conciso da melhor evidência disponível em estudos relevantes sobre um determinado tema. Os métodos são definidos com antecedência e são documentados para que outros possam avaliar criticamente e replicar a revisão.

A revisão sistemática é um estudo sobre um determinado assunto e deve seguir um plano rigoroso, com uma sequência bem definida de passos. Devido a alta meticulosidade, a vantagem da revisão sistemática é permitir que outros pesquisadores possam fazer modificações futuras desde que seguindo o mesmo protocolo.

Easterbrook et al. [47] fazem uma discussão detalhada sobre as diversas estratégias empíricas que podem ser adotadas por pesquisadores, tais como as revisões sistemáticas e os mapeamentos sistemáticos.

Sobre a metodologia de estudos terciários, Keele [80] afirma que:

> "Num domínio em que já existem várias revisões sistemáticas, uma revisão terciária é um tipo de revisão sistemática das revisões sistemáticas, utilizadas para responder a questões de pesquisas mais amplas."

Keele [80] ainda é categórico ao afirmar que "...uma revisão terciária usa exatamente a mesma metodologia de uma revisão sistemática da literatura.", e apresenta em seu trabalho intitulado *Diretrizes para a Realização de Revisões Sistemáticas da Literatura em Engenharia de Software*, um guia (protocolo) para a realização de estudos terciários, o qual esta pesquisa se baseou. Outros autores também citam a revisão terciária, como Babar e Zbang [11], que referem-se a ela como "um tipo de revisão bibliográfica de revisões sistemáticas de literatura". Kitchenham et al. [84] fazem uma menção a literatura terciária, ao afirmar que "o objetivo da revisão é avaliar literatura sistemática (que são referidas como estudos secundários), por isso este estudo é categorizado como uma revisão da literatura terciária".

Portanto, o estudo realizado neste trabalho, em sua fase de pesquisa da fundamentação teórica, consta de uma revisão sistemática sob a forma de Estudo Terciário, em cima das referências bibliográficas apresentadas por dois artigos, Dybå e Dingsøyr [44] e finalizando em Dikert et al. [40]. Várias das referências bibliográficas levaram à análise de outras referências, o que possibilitou a redação dos Apêndices A e B com maior propriedade.

Esse processo de revisão sistemática foi idealizado de acordo com o protocolo de pesquisa sugerido por Keele [80], resultando em alguns mapas mentais, conforme apresentado no Apêndice C, que resumiram todo o processo de estudo terciário seguido nesta pesquisa e ainda facilitaram a compreensão, pela visualização rápida e associações.

### 2.1.3 Leitura e classificação dos trabalhos

A metodologia e o rigor adotados otimizaram o trabalho para a seleção de textos e citações obrigatórias e constou, inicialmente, de 121 publicações referenciadas nos dois artigos que foram utilizados como base para o presente estudo terciário. A Figura 2.1 apresenta as fases deste trabalho.

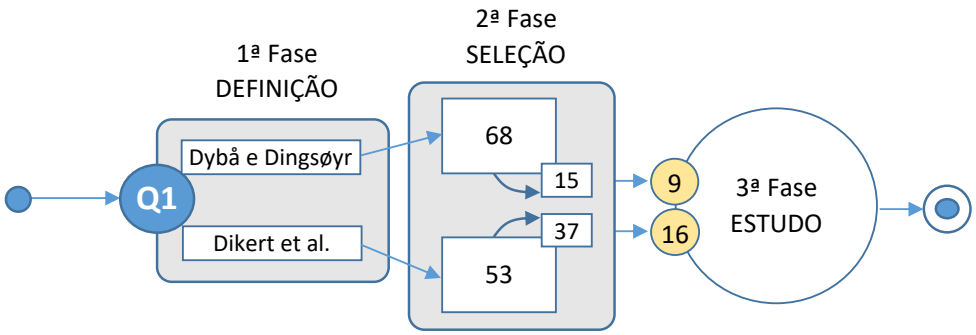



Figura 2.1: Metodologia utilizada no estudo terciário.

O início dos trabalhos foi definido na primeira questão de pesquisa — *Q1: "Que práticas ágeis estão sendo mais exploradas na literatura?"*. Para que se pudesse ter uma ideia da evolução das práticas ágeis desde o seu surgimento, buscou-se na literatura duas publicações de épocas distintas, sendo a última, a mais atual possível, ambas revisões sistemáticas da literatura sobre o assunto de metodologias ágeis de desenvolvimento de software, portanto, embasando o trabalho com estudos secundários. O motivo que levou à escolha dessas duas publicações foram:

- A primeira publicação [44], foi escolhida por atender aos objetivos de pesquisa e por se tratar de uma pesquisa publicada no ano de 2008, visto que pretendia-se realizar

uma comparação da evolução das metodologias ágeis em um intervalo não menor que oito anos, até os dias de hoje;

- A segunda [40], por ser a pesquisa mais atual possível que atendia aos objetivos da pesquisa, e por se tratar da Engenharia de Software em Larga Escala, portanto um trabalho realizado em empresas de software, que era o objetivo da pesquisa de campo.

Para a seleção dos artigos a serem estudados com maior profundidade, levou-se em consideração o que os autores mais relataram sobre as práticas ágeis, visto que o presente estudo está justamente focado nelas. Entretanto, a primeira análise das 121 publicações foi feita pela leitura e interpretação dos resumos (*abstracts*) dos trabalhos, daí resultando em uma decisão de primeira filtragem. Após essa primeira filtragem, iniciou-se o processo de síntese e análise dos dados de acordo com a primeira questão de pesquisa Q1, e uma leitura mais aprofundada de cada artigo selecionado foi feita. Para cada artigo lido, os dados de interesse da pesquisa foram coletados e em seguida consolidados, inicialmente, para uma planilha eletrônica a fim de fixar apenas as metodologias mais evidenciadas nos artigos. A esta planilha deu-se o nome de *Tabela de Investigação*, como se observa na Figura 2.2.

| **As principais metodologias ágeis investigadas são as seguintes:** | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **XP** | **XP2** | **Scrum** | **Crystal Methodologies** | **DSDM** | **Feature-driven development** | **Lean software development** |
| **FOCO** | Eficiente planejamento;<br>Pequenos lançamentos;<br>Design simples;<br>testes;<br>constante refatoração [15]. | XP2 icorpora ao XP:<br>programação em pares;<br>integração contínua;<br>Presença do cliente no local;<br>Padrões de codificação [127]. | Gestão de projetos em situações em que é difícil de planejar com antecedência;<br>Mecanismos de controle de processos empíricos ;<br>Loops de feedback constituem o elemento central;<br>Mais software que documentação;<br>Reuniões diárias com feedback [132]. | Frequentes entregas:<br>Comunicação osmótica;<br>Segurança pessoal;<br>Fácil acesso a usuários experientes;<br>Requisitos voltados para um ambiente bem técnico [28]. | Equipes pequenas;<br>Envolvimento do usuário, Capacitação da equipe;<br>Entregas frequentes;<br>Desenvolvimento iterativo e incremental;<br>Engenharia reversa;<br>Alto nível do escopo a ser fixado antes do início do projeto;<br>Testes ao longo do ciclo de vida;<br>Eficiente comunicação [144]. | MDD com ênfase no modelo de objeto inicial;<br>Design iterativo;<br>Mais adequado a sistemas críticos [110]. | Sistema Toyota de desenvolvimento;<br>Eiminação do desperdício;<br>Foco no aprendizado;<br>Decisão o mais tarde quanto possível;<br>Entregas o mais rápido possível [121]. |

Figura 2.2: Primeira investigação — apenas metodologias.

Nas leituras subsequentes, os dados foram resumidos em um editor de textos, desta vez catalogando as citações das práticas ágeis para melhor comparação em caso de futuras consultas, com a finalidade de decidir quais citações deveriam ou não participar do estudo, trabalho esse que foi feito, em um trabalho posterior, por meio do manuseio de diversos cartões impressos contendo o artigo, a referência e a citação de interesse. Os resultados só puderam ser melhor compreendidos depois que as ideias foram transportadas para os mapas mentais (Apêndice C), o que possibilitou a escrita com maior propriedade.

A Tabela 2.1 apresenta os dois artigos que foram utilizados como base para responder a primeira questão de pesquisa, bem como um relato sobre o estudo em cada um deles:

Tabela 2.1: Síntese de metodologia utilizada nos artigos base do estudo terciário.

| **Autor** | **Título/Relato** |
|---|---|
| Dybå e Dingsøyr | Empirical Studies of Agile Software Development: A Systematic Review. |
| | Selecionou-se apenas as referências a práticas ágeis. Dos 68 artigos constantes da referência bibliográfica, apenas 15 continham essa referência com informações relevantes para a pesquisa. A grande maioria das citações dirigia-se a experimentos e estudos de caso. Dos 15 selecionados, outro filtro se seguiu, para extrair os 9 finalistas, desta vez em leitura integral, pela atualidade das informações e evidências. |
| Dikert et al. | Challenges and Success Factors for Large-scale Agile Transformations: A Systematic Literature Review. |
| | O estudo sistemático de Dikert et al. [40] é um dos mais recentes na literatura ágil, publicado alguns meses antes do presente trabalho. Em relação a Dybå e Dingsøyr [44], possui menos referências bibliográficas. Em um primeiro momento, pela análise das citações textuais dos 53 artigos referenciados, extraiu-se 37 artigos, novo filtro foi aplicado, chegando aos 16 finalistas. Desta vez, um maior número de artigos foi selecionado tendo em vista a pesquisa ser a mais atual possível e por apresentar quantidade razoável de evidências, através de seus estudos em pesquisa empírica de metodologias ágeis em larga escala, que ajudariam a fundamentar melhor as pesquisas de campo, que seriam feitos a seguir em empresas de software. |

O Apêndice D condensa os trabalhos que foram selecionados pela metodologia ora

adotada. Cabe ressaltar que a leitura dessas referências também forneceu diversas outras, que permearam a escrita deste estudo, possibilitando aumentar a gama de evidências em um sentido mais amplo, característica peculiar de um estudo terciário, conforme Keele [80].

## 2.2 Metodologias Ágeis: Evidências

Dybå e Dingsøyr [44], identificaram em seus estudos que as metodologias ágeis contêm percepções tanto positivas quanto negativas, e culminaram destacando algumas lacunas na evidência sobre a aplicação de métodos ágeis para desenvolvimento de software. No que diz respeito aos métodos ágeis, apresentaram a Tabela 2.2, onde 76% dos estudos eram sobre o Extreme Programming (XP), isto em 2008.

Tabela 2.2: Metodologias ágeis em 2008, segundo Dybå e Dingsøyr [44].

| **Metodologia Ágil** | **Quantidade** | **Percentual** |
|---|---|---|
| XP | 25 | 76 |
| Geral | 5 | 15 |
| Scrum | 1 | 3 |
| Lean software development | 1 | 3 |
| Outras | 1 | 3 |
| Total | 33 | 100 |

Tais estudos foram conduzidos da seguinte forma: 73% deles foram conduzidos com profissionais iniciantes, com menos de um ano de experiência no desenvolvimento ágil, 12% eram equipes maduras, com pelos menos um ano de experiência em desenvolvimento ágil e o restante não foi possível identificar.

Os principais pontos abordados nos artigos científicos sobre metodologias ágeis até o ano de 2008 eram, basicamente:

- Entendimento sobre as diferenciações entre metodologias tradicionais e as ágeis;
- Por que e como são implantadas as metodologias ágeis;
- Estudo e integração de equipes ágeis para o desenvolvimento de software;
- Como testar aplicações enxutas na metodologia ágil;
- Como um processo ágil afeta a colaboração em uma empresa de software;
- Entendimento de como os recém-chegados às metodologias ágeis podem praticar e melhorar ao longo do tempo.

No entanto, alguns pesquisadores argumentaram que não há nada de novo sobre métodos ágeis. Hilkka et al. [72] estudaram duas organizações de desenvolvimento na Finlândia e concluíram que a metodologia XP era *"vinho velho em odres novos"*. Que o XP simplesmente formaliza vários hábitos que aparecem naturalmente, tais como o envolvimento do cliente sempre perto, ciclos curtos de liberação, desenvolvimento cíclico e respostas rápidas a pedidos de alteração; que as ferramentas e técnicas de XP já estavam sendo empregadas há mais de 10 anos, que não tinha nada de novidade. Percebe-se aí que o processo de XP emergiu como uma nova maneira de resolver problemas, embora alguns pesquisadores afirmarem não haver nada de novo.

E de lá para cá, as metodologias cresceram de adeptos. Enquanto surgem novos defensores, evangelistas do processo, surgem também opositores que afirmam "O ágil está morto!", palavras de Mathew Kern, extraídas de seu próprio site [83], que vêm causando um grande impacto.

Controvérsias à parte, Dikert et al. [40] relatam que:

> "[...] Nossa análise mostra que houveram estudos bem promissores da utilização de métodos ágeis ao redor do mundo. Embora os estudos identifiquem sérias limitações, como a insustentabilidade do local do cliente dentro do projeto por longos períodos e a dificuldade de introdução de métodos ágeis em grandes e complexos projetos, **os resultados da nossa revisão sugerem que os métodos ágeis podem melhorar a satisfação no trabalho, a produtividade e a satisfação do cliente**."

Ainda segundo eles, as evidências mais fortes e mais relevantes das práticas ágeis, são os estudos que existem sobre elas, especialmente os que relatam equipes maduras [44], o que sugere que o foco no componente humano e no social são fatores de sucesso. Especificamente, parece que um alto nível de autonomia individual tem de ser equilibrado com um nível elevado de autonomia de equipe e responsabilidade corporativa. Segundo Dybå [45], também é importante para a equipe ágil, a presença de pessoas que possuem um bom "relacionamento interpessoal de habilidades e confiança".

A evidência [45] também sugere que, em vez de abandonar de vez os princípios da gestão tradicional, as organizações devem explorar estes novos princípios, visando uma adaptação das práticas, conforme a natureza de cada projeto.

Entre os dois estudos, essas evidências também foram percebidas, em 2014 por Dingsøyr et al. [41] que relatou que "O desenvolvimento ágil tem recebido um interesse generalizado, resultando em uma mudança de padrões de pensamento..." e por Melo et al. [98], no ano de 2012, que mostrou um relativo crescimento da adoção da metologia Scrum no Brasil:

Enfim, desde os estudos de Dybå e Dingsøyr [44] e finalizando em Dikert et al. [40], razão central desta pesquisa, evidencia-se que, embora metodologias como o XP ainda sejam utilizadas em larga escala, o Scrum hoje é o mais preferido.

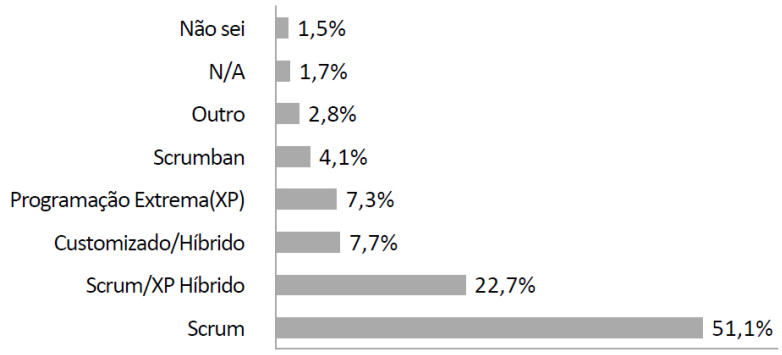



Figura 2.3: "Qual o método ágil que você mais segue?" segundo Melo et al. [98].

## 2.3 No Campo das Práticas Ágeis

Segundo Dingsøyr [41], em recente pesquisa:

> "...os grandes projetos estão adotando cada vez mais práticas de desenvolvimento ágil, e isso traz novos desafios para a investigação. O workshop sobre princípios do desenvolvimento ágil em larga escala, focado em temas centrais em grande escala, traz novas maneiras de pensar sobre a arquitetura, a coordenação inter-equipe, gestão de carteiras e dimensionamento das práticas ágeis."

Se a abordagem é tão evidente, mais estudos são necessários para esclarecer diversos pontos sobre as metodologias envolvidas, os benefícios e as limitações, bem como as percepções que a comunidade vem tendo.

### 2.3.1 Percepções, benefícios e desafios

Estudos sobre as percepções dos clientes relataram **satisfação do cliente**, exatamente pela oportunidade de interagir com as equipes [45]. Em 2008, Dybå e Dingsøyr [44] já mostravam as evidências sobre os benefícios da presença do cliente junto à equipe de desenvolvimento, principalmente na acurácia de defeito. Contudo, conforme pesquisa de Dingsøyr [41], o papel do cliente no local pode ser estressante e insustentável por longos períodos, alguns desenvolvedores relataram isto. Sobre benefícios, algumas empresas que utilizam metodologias ágeis relataram também que os **funcionários estão mais satisfeitos com seus empregos**.

Sobre a **programação em pares**, os resultados da pesquisa [45] foram controversos, onde diversos desenvolvedores consideraram como uma prática exaustiva porque requer uma maior concentração. Estudantes universitários também perceberam as práticas ágeis como relevantes para sua formação profissional, por acreditarem que elas podem melhorar a produtividade da equipe. Por outro lado, segundo o mesmo autor, a programação em

pares foi considerada a mais bem sucedida prática ágil, por permitir que novatos pudessem assimilar com mais precisão as técnicas de programação, e ainda, segundo Dikert et al. [40], existem vários estudos que atestam esta evidência.

Williams e Cockburn [71] afirmaram que os métodos ágeis funcionam melhor com **equipes pequenas**, de até 50 pessoas ou menos, quando o projeto for de larga escala, que tenham fácil acesso aos especialistas das regras de negócios, mas que, segundo Nerur et al. [103], existem desafios a alcançar com pequenas equipes, pois elas têm de cuidar para manter a **melhoria contínua** e os **testes automatizados**.

Sobre o desenvolvimento em si, percebe-se que os aspectos sobre a **arquitetura do projeto e o código que a implementa** devem ser geridos com bastante esforço, conforme ressaltam Nord et al. [106]. Segundo eles, todo o esforço deve ser dirigido para o início do projeto, no momento da concepção da arquitetura, pois tudo o mais daí pra frente será dependente dela, em termos de código.

As metodologias mais notórias como XP e Scrum, carregam em si um arcabouço enorme de práticas ágeis, pois as combinam entre si de acordo com cada projeto [77]. Ainda segundo Kaisti et al. [77], um dos principais pilares dessas metodologias é o **desenvolvimento iterativo e incremental**. As **equipes são comumente pequenas**, trabalhando em conjunto, o que ajuda na produção de um software de alta qualidade. O **feedback frequente de clientes** estreita a colaboração entre as partes envolvidas [86]. O núcleo do Scrum está em loops de **feedback frequentes** e Sprints que ocorrem em **reuniões diárias** de stand-up e mensais [131]. Outras afirmações como "...em metodologias ágeis, o desenvolvimento é **iterativo e incremental** em equipes integradas." [77], e "...requisitos para o sistema desenvolvido são armazenados no **product backlog**" [133], são cada vez mais perceptíveis.

Outra metodologia, o **Kanban**, difere das demais por possuir algumas práticas que promovem a mudança gradual e um fluxo constante ao longo de iterações do projeto [6].

Percebe-se facilmente que a adoção de práticas ágeis pode acontecer como uma revolução em uma empresa com todas as práticas, tanto antigas, quanto as mais modernas. Todas elas ressoam na capacidade de reduzir o tempo de desenvolvimento, a fim de trazer **visibilidade** aos processos. No entanto, a agilidade, como se tem percebido, não é capaz de resolver todos os problemas. A adaptação para processos ágeis pode ser vista como uma força contrária ao processo de desenvolvimento de software tradicional, e isto pode trazer vários incômodos junto às partes interessadas. Então, essas mudanças carecem de estudo razoável, com bastante atenção, pois nem sempre o próprio cliente deseja isto.

### 2.3.2 Principais desafios

Qualquer transformação organizacional que envolva inúmeras pessoas enfrentará desafios e a implantação das metodologias ágeis não foge à regra. Dikert et al. [40] afirma que o ceticismo e a desconfiança no desenvolvimento ágil em geral foram problemas em comum e descreveu como a administração, por um lado, reconhece os benefícios da agilidade, mas, por outro, se opõe à sua introdução devido a razões contratuais.

Savolainen et al. [129], descreveu que usando estórias de usuários pode-se apresentar um problema em sistemas grandes de forma bem eficiente, uma vez que somente uma simples interação com o usuário pode não ser suficiente para abstrair dele as funcionalidades necessárias ao projeto. Ainda, de acordo com ele, a alocação de trabalho para um grande número de equipes diferentes, com diferentes competências, tende a diminuir a velocidade e aumentar o papel do design e gestão.

Percebe-se, pelos estudos dessas questões, que alguns desafios enfrentados pelas equipes que decidem adotar ágil são dignos de debates, conforme discutido no restante desta seção.

#### 2.3.2.1 Quebra de Paradigmas

Desde a formulação do manifesto ágil em 2001, métodos ágeis têm transformado a prática de desenvolvimento de software em constante evolução do pensamento e forma de agir dentro das empresas, enfatizando fortemente a entrega evolutiva e o envolvimento do usuário final ativo [42]. Existe um interesse generalizado em torno do desenvolvimento ágil, resultando em uma mudança de padrões de pensamento, mas existem também fortes barreiras em sua adoção. Rajlich [125] descreve o desenvolvimento ágil como uma mudança de paradigma na engenharia de software que "traz uma série de novos temas na vanguarda da investigação em engenharia de software". Vencer paradigmas, nesse contexto, se torna um grande obstáculo para a adoção de práticas ágeis, uma vez que [125]:

> "Estes tópicos têm sido negligenciados no passado por pesquisadores inspirados pelo velho paradigma e, portanto, há um atraso de problemas de pesquisa a ser resolvido."

Segundo Heidenberg et al. [67], uma boa forma de quebrar o paradigma e de implantar as metodologias ágeis em uma empresa, é a introdução de um **projeto-piloto**, que poderia ser realizado em três etapas: a comercialização do piloto, a preparação do piloto e a execução do piloto, validando-o como um estudo de caso.

#### 2.3.2.2 O Ceticismo e as Barreiras na Implementação de Metodologia Ágil

Conforme relatado em Dikert et al. [40], o ceticismo se refere a qualquer tipo de reserva para uma nova forma de trabalho. Embora a reserva possa não ser um problema por si só, vale a pena reconhecer que a reserva é mencionada em muitos casos. Relata ainda que a tendência da equipe técnica para o ceticismo desperdiçou tempo e esforço, muitas vezes criado por equívocos, incluindo que a agência não funcionaria para produtos complexos, que as metodologias ágeis tinham de ser implementadas tal qual descritas nos livros, o que traria sobrecarga.

Dikert et al. [40] afirmam ainda que o modo como a transformação é iniciada afeta como a resistência à mudança é mostrada. Em vários casos, a iniciativa de mudança vem da gerência e, quando apresentada de maneira ruim, torna-se um empecilho à receptividade. Spayd [144] resume isto como: "As organizações não mudam simplesmente porque o patrão diz isso, pelo menos não do modo que se pretende". Mas veja-se que uma ordem vinda de cima para baixo também pode diluir a compreensão das razões para iniciar a transformação e a compreensão do desenvolvimento ágil em geral, e isto pode ser um grande desafio. Um problema relacionado com um mandato de cima para baixo é que se a gerência não definir um objetivo claro para usar ágil, os desenvolvedores podem achar que os métodos ágeis podem ser substituídos por algo mais a qualquer momento. Por outro lado, os autores também relatam a falta de apoio da gerência para a mudança e a falta de inclinação para mudar a cultura, o que foi considerado como um dos problemas mais sérios nessa transformação. Ágil traz mudanças para alguns papéis de gerenciamento, e a falta de compreensão do desenvolvimento ágil deixará os gerentes sentindo-se excluídos.

Nesse caso, reuniões para o esclarecimento do pessoal seriam preponderantes para que os esforços sejam convergidos para um ponto em comum.

#### 2.3.2.3 Falta de Conhecimento

Uma equipe de software experiente pode fazer se sair bem no treinamento, mas quando chega a hora de aplicar as técnicas ágeis na prática, a equipe pode se perder, isto porque diferentes cenários exigem diferentes habilidades. Segundo Dikert et al. [40] um desafio comum foi que a implementação de métodos ágeis se mostrou difícil, pois houve muitos exemplos de problemas causados por equívocos sobre o que é o desenvolvimento de software ágil.

Um outro mal entendido de metodologia ágil é a forma de como ela é visualizada, sob a ótica apenas das ferramentas utilizadas, suficientemente focadas nas próprias ferramentas e não nas razões por trás do seu uso, o que pode gerar uma certa frustração.

Então, um grande desafio é a necessidade do entendimento da equipe, que deve ser feita de forma planejada. É necessário que se entenda o cerne das metodologias ágeis, e muito mais que somente reuniões, é necessário introduzir também pessoal com conhecimento da metodologia nas equipes, para que ela se fortaleça, para que os líderes de células de programação possam influir de forma positiva, fazendo funcionar o que é previsto.

#### 2.3.2.4 Práticas Ágeis versus Engenharia de Requisitos

O Desenvolvimento ágil tem um relacionamento complexo com a Engenharia de Requisitos, uma vez que Beck [16], afirmou que a engenharia de requisitos tradicional e seus métodos é um grande empecilho para abraçar a causa das metodologias ágeis. A questão é que, se os requisitos mudam com tanta frequência, por que manter essa documentação? A melhor solução para isto, segundo Savolainen et al. [129], é a obtenção de um feedback do cliente a respeito dos lançamentos frequentes. Esses lançamentos são baseados nas **estórias de usuários** que já são documentadas e que promovem uma discussão com o cliente. É por isto que a comunidade ágil tem visto os requisitos tradicionais como artificiais, visto que a Engenharia de Requisitos, por si só, produziria uma grande quantidade de papel difícil de ser mantida. É uma questão interessante que se traduz em um bom desafio, mas existem algumas soluções na literatura, como Ambler [2], que apresenta várias fórmulas de adequação das duas abordagens em concomitância.

Dentre as principais causas de falha na adoção de uma metodologia, aponta-se como principais desafios:

1. A filosofia ou a cultura da empresa que conflitam com os valores e os princípios do ágil;
2. A falta de suporte gerencial para apoiar as mudanças, pois é preciso desejar as mudanças;
3. A falta de experiência ou treinamento insuficiente no novo processo. É preciso tornar-se capaz de trabalhar de maneira ágil;
4. Falta de comprometimento da própria equipe. É preciso reconhecer que há espaço para melhorias e desejá-las.

#### 2.3.2.5 Estórias de Usuários versus Especificações

Não é incomum que, após o lançamento de uma sprint, o usuário final indague “não era isto o que eu queiria”, referindo-se a uma funcionalidade implementada completamente fora das especificações. Aqui, o problema pode estar nos dois lados, pois, tanto os

desenvolvedores podem ter codificado errado, quanto o cliente que não soube se expressar adequadamente durante a coleta dos requisitos, ou o engenheiro de requisitos não captou corretamente a estória do usuário.

Sob esse aspecto, Leffingwell [88] defende que o desenvolvimento ágil tem como alvo a entrega rápida e frequente de valiosas funcionalidades visíveis ao usuário final. Uma maneira popular de descrever essas funcionalidades, que são meras necessidades dos utilizadores, é empregar o recurso **estórias de usuários**. As estórias de usuários devem descrever a funcionalidade a partir da perspectiva do usuário, mas que isto tende a trazer dois problemas básicos: A sua aplicação leva esforço significativo para serem implementadas, porque são geralmente imprecisas, onde o engenheiro gasta um esforço enorme para tentar entender e a implementação pode levar mais tempo do que um simples sprint.

As estórias de usuários podem ser divididas em requisitos menores que, segundo Maiden [93], se não forem compreendidas em sua totalidade, podem levar a problemas graves. Ainda segundo ele:

> "A coleta de requisitos do usuário exige muito esforço, e o risco de falhar ainda é muito elevado. Enquanto os consumidores ainda não tiverem a ideia do produto ou serviço (o software), será muito difícil iniciar. Neste caso, a criatividade dos engenheiros de requisitos deverá entrar em ação, na presença do usuário, para a transferência correta de suas reais necessidades para a equipe de desenvolvimento."

Segundo Sheppard [136], os requisitos, ou as estórias de usuários, são responsáveis por 37% das falhas dos projetos de desenvolvimento de software, conforme apresentado na Figura 2.4. Segundo o autor, alguns usuários têm dificuldades em externar as suas necessidades, o que acaba resultando em erros, caso a equipe não tome medidas imediatas para sanar eventuais dúvidas. Esses problemas são mostrados no gráfico como **Informação errada** (13%), **Requisitos incompletos** (12%), **Mudança de requisitos** (12%), **Falta de conhecimento técnico** (7%) e **falta de competência** (5%).

Conforme Fowler [54], "...casos de uso organizam os requisitos.", e embora a geração excessiva de documentos seja considerada contrária à filosofia ágil [16], alguns cuidados podem ser tomados durante a entrega das estórias de usuários aos desenvolvedores que as codificam, e uma prática que pode ser adotada, mas que também é um desafio, é a geração de casos de uso baseado nelas, fracionando-as em requisitos menores.

#### 2.3.2.6 Riscos

Vários riscos são apontados nas pesquisas em Engenharia de Software que podem se tornar grandes obstáculos ao desenvolvimento se não forem corretamente geridos, portanto um grande desafio para as equipes. Keil et al. [81], cita-os como riscos potenciais que merecem

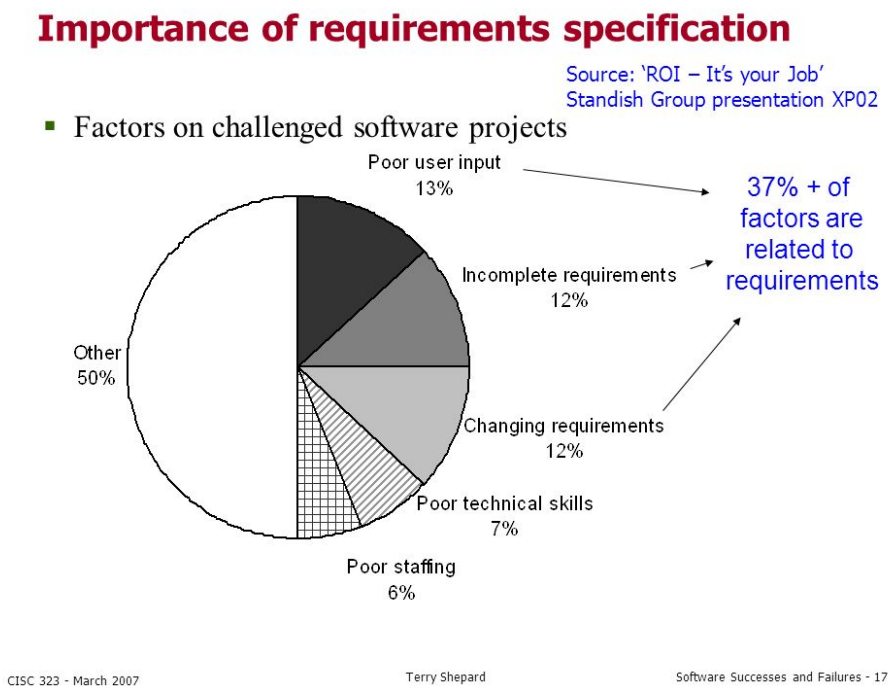



Figura 2.4: "Principais desafios em projetos de software." segundo Shepard [136].

especial atenção. Vários autores também estudaram esses riscos e a Tabela 2.3 mostra a relação deles agrupados por similaridade. Na coluna 1 são listados todos os riscos. Para as colunas 2, 3, 4 e 5, utilizou-se a seguinte convenção: a letra "S" (Sim) na coluna correspondente ao nome do autor indica que o mesmo cita o risco em seu trabalho e a letra "N" (Não) indica que o autor não cita o risco no seu trabalho.

Onde, na Tabela 2.3, convencionou-se ainda o seguinte: K = Keil; B = Boehm; A = Addison e C = CMMI.

Tabela 2.3: Comparação entre os Principais Riscos de Software.

| **Riscos** | **K** | **B** | **A** | **C** |
|---|---|---|---|---|
| Falta de compromisso da gerência sênior com o projeto | S | N | S | N |
| Falha em obter compromisso dos usuários e falta de envolvimento adequado dos usuários | S | N | S | N |
| Não-entendimento dos requisitos, falha na gerência de expectativas dos usuários e instabilidade dos requisitos | S | S | S | S |
| Escopo/objetivos não-claros e diferentes expectativas sobre o sistema entre distintos usuários | S | N | S | N |
| Falta de conhecimento/habilidade requerida do pessoal do projeto e equipe insuficiente ou inadequada | S | S | S | S |
| Introdução de novas tecnologias | S | N | S | S |
| Cronograma e orçamento não-realistas | N | S | S | S |
| Falta de uma metodologia de projeto efetiva | N | N | S | N |
| Acréscimo de funcionalidades idealizadas pela equipe sem o aval do cliente (gold plating) | N | S | S | N |
| Desenvolvimento errado das funções (implementação não atende à especificação) ou interface | N | S | S | N |
| Subcontratação (tarefas ou componentes desenvolvidos externamente) | N | S | S | S |
| Uso de recursos e desempenho do sistema inadequados | N | S | S | N |
| Projeto (desenho) inviável | N | N | N | S |

Fonte: Adaptado de Keil et al. [81].

Quando se aceita um projeto durante as fases de planejamento, a equipe também se compromete de forma intuitiva a assumir os riscos, os quais podem até aumentar de tamanho, porque os requisitos iniciais, normalmente, tendem a mudar durante o ciclo de vida do projeto, especialmente quando o cliente está lado a lado e empresas desenvolvedoras de software devem estar preparadas para essas mudanças, conforme aponta Schwaber [132]. Se não forem acompanhados adequadamente, os requisitos tendem a evoluir sem que a equipe do projeto possa notar, levando à criação de um software diferente do que o cliente esperava.

## 2.4 Considerações Finais

O estudo terciário teve a finalidade de embasar o trabalho com a fundamentação teórica necessária para a interpretação e validação dos dados obtidos na pesquisa de campo (Capítulos 3 e 4). Para que isto pudesse ser feito de forma empírica, a seguinte questão de pesquisa foi colocada: *(Q1) Que práticas ágeis estão sendo mais exploradas na literatura?*

Para respondê-la, foi utilizada a metodologia de leitura e análise das referências apresentadas nos trabalhos de Dybå e Dingsøyr [44] e Dikert et al. [40], o que representou a análise de 121 artigos, conforme apresentado no Apêndice D. Este estudo motivou a escrita dos Apêndices A e B que o complementa, cuja leitura dos dois pode ser dispensada sem o prejuízo do entendimento. A Tabela 2.4 sintetiza as práticas ágeis mais evidentes no estudo terciário (o Apêndice B faz uma análise separada de cada uma delas).

Tabela 2.4: Tabela de práticas ágeis mais evidentes no estudo terciário, agrupadas por categorias para facilitar o entendimento.

| | EVIDÊNCIA DA PRÁTICA ÁGIL |
|---|---|
| GERENCIAMENTO | Ampliar conhecimento |
| | Análise de causa inicial |
| | Comunicação |
| | Contrato de escopo negociável |
| | Design incremental |
| | Eliminação de desperdícios |
| | Folga |
| | Gerência continuada |
| | Gráfico Burndown |
| | Implantação diária |
| | Implantação incremental |
| | Integração contínua |
| | Iterativo e incremental |
| | Jogo do planejamento |
| | Melhoria contínua |
| | Metáfora |
| | Minimalismo |
| | MosCoW |
| | Otimizar o todo |
| | Pagar por uso |
| | Planning Pocker |
| | Redução de custo |
| | Plano de mitigação de riscos |
| | Processo simplificado |
| | Quadro de Tarefas |
| | Visibilidade |
| | Discussão sobre o quadro branco (lousa) |
| TESTE | Teste |
| | Teste primeiro (TDD) |
| | Testes integrados |
| | Build de 10 minutos |
| | Código e testes |

| | EVIDÊNCIA DA PRÁTICA ÁGIL |
|---|---|
| DESENVOLVIMENTO | Aprender fazendo |
| | Cliente junto |
| | Código compartilhado |
| | Desenv. por Carac. das funcionalidades |
| | Envolvimento real do cliente |
| | Estórias de usuários |
| | Modelagem em objetos de domínio |
| | Padrões de codificação |
| | Programação em pares |
| | Propriedade coletiva |
| | Prototipação |
| | Refatoração |
| | Repositório único de código |
| | Uso de templates |
| TIME | Time com poder de decisão |
| | Time completo |
| | Time-boxes |
| | Trabalho energizado |
| | Sentar junto |
| | Reunião em pé |
| | Motivação |
| | Área de trabalho informativa |
| | Equipes que encolhem |
| | Esforço do time |
| | Continuidade da equipe |
| RELEASE | Tempo de ciclo curto |
| | Retrospectiva ao término do ciclo |
| | Resultados úteis a cada duas semanas ou menos |
| | Releases curtas |
| | Entrega de 2 a 4 semanas / Entregas frequentes |
| | 40 horas semanais |
| | Adiamento de decisões ao máximo |
| | Ciclo de estação |
| | Ciclo semanal |
| | Estimativas |
| | Satisfação total do cliente |

Visto que a análise dos dois artigos base deste estudo terciário possibilitou a descoberta dessas práticas ágeis, vários outros estudos também foram utilizados para referenciá-las, realizando mais embasamento na literatura. Possibilitou um maior entendimento sobre elas, a montagem de diversos mapas mentais conforme mostrados no Apêndice C, que faz parte do presente estudo terciário. Essas imagens realizam as derivações das referências mostrando as práticas de uma maneira gráfica, para rápida visualização.

Enfim, a primeira questão de pesquisa foi satisfeita com a apresentação desses resultados, cujo teor foi comentado conforme as referências, apresentando tanto pontos fortes quanto pontos fracos.

Concluindo, um dos desafios futuros das metodologias ágeis pode ser sintetizado como *"encontrar uma maneira de vencer alguns de seus principais desafios"*. Perto ou longe de isto acontecer, elas continuam conquistando adeptos (e também opositores). Trabalhos futuros são bem vindos, pois as lacunas deixadas pela engenharia das práticas ágeis ainda poderão ser exploradas com metodologia empírica. Nesse sentido, a Engenharia de Software apresenta vários estudos relevantes e o presente trabalho de estudo terciário vem somar conhecimento na direção desse saber.

# Capítulo 3

# Pesquisa de Campo

Este capítulo apresenta a metodologia utilizada para a realização de uma pesquisa de campo que tem como objetivo responder as questões de pesquisa focando as principais ideias relacionadas com as metodologias e práticas ágeis, abordando a temática no ângulo de sua evolução, bem como as técnicas a serem utilizadas para a coleta e análise, formatação e apresentação dos resultados da pesquisa.

## 3.1 Metodologia

A metodologia utilizada em qualquer tipo de pesquisa exerce um papel fundamental, pois é por meio dela que é possível examinar e avaliar alternativas que melhor podem ser aplicadas a cada tipo de trabalho.

A metodologia, segundo Barros et al. [12], pode ser entendida como:

> "...uma ciência que se relaciona com a epistemologia. Consiste em estudar e avaliar os vários métodos disponíveis, identificando suas limitações ou não. A metodologia examina e avalia técnicas de pesquisa, e a sua geração e verificação de novos métodos que conduzem à captação e processamento das informações com o fim de resolver problemas de investigação."

Conforme ressaltam Sjoberg et al. [139], as pesquisas empíricas constantemente envolvem a aquisição de conhecimento através de métodos empíricos, com o intuito de explorar, descrever, prever e avaliar os fenômenos almejados a partir de evidências obtidas por observação sistemática ou experimento, ao invés de lógica dedutiva.

### 3.1.1 Abordagem sob as Formas Qualitativa e Quantitativa

A investigação proposta neste trabalho tem como estratégia metodológica a abordagem qualitativa e também a quantitativa. Ela é **qualitativa**, pelos resultados de um processo de reflexão e análise da realidade para a compreensão detalhada do objeto em estudo.

Para Oliveira [108]:

> "Conceitua-se abordagem qualitativa ou pesquisa qualitativa como sendo um processo de reflexão e de análise da realidade, por meio da utilização de métodos e de técnicas cuja finalidade é a compreensão detalhada do objeto em estudo quanto ao seu contexto histórico ou sua estruturação."

Ainda, Para Martins [151]:

> "Uma das principais características da pesquisa qualitativa é a predominância da descrição, das pessoas, das situações, dos acontecimentos, das reações, e inclusive transcrições de alguns relatos. [...] É muito importante em uma pesquisa do tipo qualitativa que o pesquisador capture a perspectiva dos participantes envolvidos no estudo. Assim, ao considerar vários pontos de vista, o pesquisador deverá ser capaz de entender melhor o dinamismo entre os elementos da pesquisa."

Esta pesquisa também é do tipo **quantitativa**, pela apresentação da formatação dos dados obtidos em métricas que evidenciam a realidade com que se deparam os fenômenos pesquisados.

Segundo Martins [151], "...uma pesquisa quantitativa tem como principal característica permitir uma abordagem focalizada, de forma pontual e estruturada, utilizando-se de dados que são quantitativos." A coleta desses dados quantitativos se realiza por meio da obtenção de algumas respostas estruturadas a perguntas fechadas que admitem respostas previamente definidas. Neste caso, as técnicas de análise são dedutivas, partindo do geral para o particular e orientadas pelos resultados, que normalmente são generalizáveis, mas os resultados são imutáveis porque são extraídos das respostas oriundas de questionários aplicados.

Para Barros et al. [12], o processo de formatação dos dados de uma pesquisa pode ser resumida como:

> "Antes de se iniciar a fase de interpretação dos resultados, é necessário que se examine os dados, submetendo-os a uma análise crítica, observando falhas, distorções e erros. Uma vez selecionados os dados passíveis de análise e interpretação, os passos seguintes são: a classificação, a codificação e a tabulação."

Para o estudo deste trabalho, a formatação dos dados coletados foi feita com o auxílio de tabelas e gráficos, para as questões fechadas (quantitativo), e por meio da descrição e transcrição de relatos nas questões abertas (qualitativo), da seguinte forma: Após a aplicação dos questionários, por meio de entrevista face-a-face, telefone ou e-mail, os dados foram analisados, selecionados e interpretados de duas maneiras:

- As perguntas fechadas foram analisadas com o auxílio de tabelas e gráficos e
- as perguntas abertas, foram analisadas utilizando-se a descrição de relatos, obtidas por meio da gravação das entrevistas e das reações dos entrevistados.

### 3.1.2 Pesquisa Descritiva e Exploratória

A partir dos objetivos do estudo, classificou-se a presente pesquisa como sendo do tipo descritiva e exploratória. Abaixo, descreve-se os principais pressupostos de cada uma delas:

#### 3.1.2.1 Pesquisa Descritiva

A Pesquisa Descritiva tem como meta buscar a solução de problemas melhorando as práticas existentes. É descritiva porque visa descrever, analisar com que frequência ocorre, suas causas e suas relações e conexões com outros fenômenos. Neste tipo de *survey* a hipótese não é casual, mas tem o propósito de verificar se a percepção dos fatos está ou não de acordo com a realidade.

Sobre isto, Barros et al. [12] destacam que "...nesse tipo de pesquisa, não há a interferência do pesquisador, pois ele descreve o objeto de pesquisa. Ele descobre a frequência com que o fenômeno ocorre, sua natureza, características, causas, relações e conexões com outros fenômenos."

Com relação à pesquisa descritiva, Cervo [27] diz ainda:

> "A pesquisa descritiva observa, registra, analisa e correlaciona fatos ou fenômenos sem manipulá-los. Procura descobrir, com maior precisão, a frequência com que o fenômeno ocorre, sua relação e conexão com outros fenômenos, sua natureza e suas características. [...] Os dados por ocorrerem em seu hábitat natural, precisam ser coletados e registrados ordenadamente para seu estudo propriamente dito."

Nesta pesquisa evidencia-se o tipo descritivo porque procurou-se descobrir a frequência com que as práticas ágeis aparecem hoje nas empresas de software, sua relação e conexões entre elas e as metodologias em que aparecem, por meio de sua natureza e características, descrevendo puramente o fenômeno.

A análise e as descrições objetivas, delineadas a partir de entrevistas com especialistas em um determinado domínio do conhecimento, contemplam esse método, perfeitamente alinhado com os propósitos desta pesquisa, pois ela busca identificar quais situações, eventos, atitudes ou opiniões estão manifestas em uma população, além de descrever a distribuição de algum fenômeno na população ou entre os subgrupos de população ou, ainda, fazer uma comparação entre essas distribuições.

#### 3.1.2.2 Pesquisa Exploratória

Um estudo é exploratório quando a pesquisa apresenta o objeto de estudo e, a partir dos dados obtidos, estrutura o cenário para sua análise. Conforme atesta Severino [135], "A

pesquisa exploratória busca apenas o levantamento de informações sobre um determinado objeto, delimitando assim um campo de trabalho...".

De acordo com Cervo [27]:

> "A pesquisa exploratória não requer a criação de hipóteses a serem testadas, restringindo-se a definir objetivos e buscar informações sobre determinados assuntos de estudo. Tais estudos têm por objetivo familiarizar-se com o fenômeno ou obter uma nova percepção dele e descobrir novas ideias."

Percebe-se, assim, que a pesquisa exploratória é importante pois, por meio dela, levanta-se todas as informações acerca do objeto em estudo, para deduções e aferições futuras. Possui o seu foco a identificação dos conceitos iniciais sobre um tópico, dar ênfase na determinação de quais conceitos devem ser medidos e como eles devem ser medidos e buscar descobrir novas possibilidades e dimensões da população de interesse.

## 3.2 Survey

Para a consecução do trabalho, adotou-se os padrões de pesquisa comumente empregados por pesquisadores em Engenharia de Software, conforme apresentado na Seção 3.1. O *survey* pode ser caracterizado como a coleta de informações sobre as características, ações ou opiniões de um grupo de pessoas, representado por uma amostra, por meio de um instrumento de pesquisa, **geralmente um questionário**, conforme aponta Fowler Jr. [59]. Para a execução das tarefas de *survey* existem diversas ferramentas web, tais como Google Forms [61], Lime Survey [49] e Survey Monkey [147], entre outras. Tais recursos fornecem suporte a diversas atividades envolvidas em uma pesquisa deste tipo, como criar questionários de pesquisa e executar análises de dados.

Para utilizar a técnica explanada em Linaker et al. [91] em seu trabalho intitulado *Diretrizes para a Realização de Pesquisas em Engenharia de Software*, os seguintes passos foram seguidos, ao longo do estudo, em consonância com o trabalho citado:

1. **Definição dos objetivos**: Os objetivos foram estabelecidos a partir das 2ª e 3ª questões de pesquisas (Q2 e Q3):
   (Q2) Qual a percepção das empresas sobre a adoção de práticas ágeis?
   (Q3) Qual a extensão da adoção de práticas ágeis nas empresas?

2. **Concepção do Protocolo de Survey**: Técnica que norteou a aplicação do *survey* — conjunto de regras, ambientes e comportamentos na qual se enquadrou a pesquisa. Foi conduzido um projeto-piloto relativo ao *survey* antes da realização das entrevistas propriamente ditas.

3. **Execução**: O trabalho exercido pelo pesquisador em campo constou de visitas às empresas de software na região do Distrito Federal e entorno, onde foram realizadas as entrevistas. Outros dados foram coletados por e-mail utilizando-se o mesmo questionário;

4. **Análise dos resultados**: Formatação, apresentação e exame dos dados coletados na pesquisa de campo, por meio de análise qualitativa e quantitativa.

### 3.2.1 O Protocolo de *Survey* adotado

Segundo Linaker et al. [91], o protocolo é o mecanismo que norteia a execução do *survey*. O estabelecimento de um sólido e consistente protocolo é de extrema importância para o sucesso de qualquer pesquisa. Para a construção desse protocolo as definições abaixo foram levadas em consideração, para o caso deste estudo, em que aborda-se as metodologias e práticas ágeis:

1. **Caracterização do tipo de survey**: Conforme Lakatos e Marconi [123], uma pesquisa científica deve ser desenvolvida como um processo de investigação em que se interessa descobrir as relações existentes entre os aspectos que envolvem os fatos, fenômenos, situações ou coisas. Em uma pesquisa científica procura-se utilizar o método científico que, de uma maneira geral, consiste em realizar as seguintes etapas para a resolução de um problema: – definição e delimitação de um problema de pesquisa; – observações, coleta de dados e de informações e – análise e interpretação dos resultados. A pesquisa desenvolvida neste trabalho, como vista na Seção 3.1, foi definida com o cunho descritivo e exploratório.

2. **Definição da unidade de análise**: indivíduos, grupo de pessoas, setor de uma determinada organização, toda a organização ou grupo de organizações. Para o caso deste estudo, foi definido que as unidades de análise seriam pertencentes ao Distrito Federal e entorno, cujos dados seriam obtidos por meio de entrevistas semiestruturadas (face-a-face e por telefone) e também por e-mail.

3. **Definição do tamanho da amostra**: Segundo Linaker et al. [91], existem dois tipos de amostras: A probabilística e a não-probabilística.

   Nas amostras probabilísticas todas as unidades de uma amostragem devem ter a mesma probabilidade de serem selecionadas de forma aleatória.

   As amostragens não-probabilísticas estão relacionadas com todas as abordagens de amostragens em que a aleatoriedade não pode ser observada em selecionar as unidades. A literatura especializada apresenta 4 principais projetos de amostragem não probabilística:

- **Amostragem Acidental**: O único critério para a seleção de cada unidade é a conveniência. É um projeto comum em pesquisas. Os pesquisadores recrutam indivíduos de suas conexões pessoais.
- **Amostragem por Quotas**: Uma base de amostragem pode ser composta por subconjuntos mutuamente exclusivos. Por exemplo, um projeto de pesquisa poderia estabelecer que 20 empresas serão pesquisadas e limitar o convite de pesquisa para apenas dez empregados de cada empresa.
- **Amostragem de Julgamento**: Destina-se a reduzir o viés da amostragem acidental, uma vez que há razões claras para a seleção de cada unidade. Inclui práticas, como o uso da opinião dos especialistas sobre a seleção de unidades.
- **Bola de Neve**: Estende-se a amostragem acidental, tipicamente selecionando sementes de indivíduos para indicar indivíduos (segundo nível) de ser recrutado pelos pesquisadores.

Nesta pesquisa, optou-se por utilizar a técnica do tipo Acidental e Bola de Neve. **Acidental**, porque os dois primeiros participantes foram escolhidos entre indivíduos de conexão pessoal com o entrevistador e **Bola de Neve** porque os dois indivíduos que iniciaram a pesquisa com seus depoimentos, indicaram outros dois, e esses quatro indicaram outros oito, e assim por diante, até que se atingisse a meta fixada na pesquisa, que era de, não menos que 30 empresas (número definido no início da pesquisa) de desenvolvimento de software sendo analisadas entre os participantes, o que seria suficiente para a realização de uma análise tendo os objetivos propostos.

4. **Definição e formatação do questionário**: Algumas questões, como a definição dos elementos que iriam compor o questionário conforme as duas últimas questões de pesquisa, e os elementos que serviriam de base para a coleta das informações, foram levadas em consideração na hora de elaborar o questionário.

   Optou-se pela ferramenta Google Forms para a distribuição por e-mail. O Google Forms é uma plataforma web que fornece esse tipo de serviço em modalidade gratuita, possuindo as seguintes características, adequadas a este trabalho:

   - variados tipos de perguntas;
   - Página, pergunta e lógica de ramificação;
   - Questionários que podem ser enviados por meio de aparelhos móveis, pela web ou via mídias sociais.

   O uso de Google Forms neste trabalho possibilitou agilizar o tempo de coleta e a análise dos dados, bem como a manipulação de documentos, o que facilitou bastante

a substituição dos instrumentos em papel por meio de um formulário online que permitiu a coleta das respostas de forma organizada, poupando tempo e melhorando as condições das análises, quando a coleta ocorreu de forma online. Uma outra realidade é que os arquivos puderam ser acessados em qualquer lugar e horário, além ainda de ser gratuito e não requerer conhecimentos aprofundados para sua utilização.

O mesmo questionário foi utilizado nos três tipos de coletas adotados pelo pesquisador (face-a-face, telefone e e-mail), e foi refinado após as três primeiras entrevistas do piloto. O piloto foi rodado com a realização das 3 entrevistas iniciais. A sua formatação final, após a aplicação do *pilot-survey*, pode ser visto no Apêndice E.

### 3.2.2 Implementação

Segundo Linaker et al. [91], em inquéritos por questionário, os dados são sempre coletados de indivíduos, que são considerados as unidades de observação, o objeto principal, que são necessariamente os indivíduos inquiridos.

Sobre a técnica, o autor afirma ainda que:

> "Um instrumento de pesquisa é normalmente um questionário que é muito importante e requer considerações especiais. As perguntas representam o principal objetivo ou meta da investigação que está a ser realizado. Os resultados e conclusões de um inquérito dependem diretamente da qualidade do questionário utilizado."

O autor classifica os questionários em dois tipos, a saber: **Auto-administrado** e **Administrado por Entrevistador**.

Os questionários auto-administrados são guiados pelos participantes e eles são executados por e-mail, sem a ajuda dos investigadores. Portanto, este tipo de questionário exige alguma informação adicional no início do questionário. São menos onerosos e fácil de administrar, no entanto, este tipo geralmente resulta em baixa taxa de respostas porque os entrevistados são menos motivados a responder. Para amenizar o problema pode-se projetar o questionário escrevendo as apresentações necessárias com informações sobre a importância dos resultados da pesquisa para os inquiridos.

Os questionários administrados pelo entrevistador admitem duas modalidades: A modalidade **face-a-face** e a **por telefone**. Segundo Linaker et al. [91], ao usá-los, a população-alvo pode ser acessada facilmente, independentemente da sua formação ou especialização. Nelas, o pesquisador pode ajudar a esclarecer algumas perguntas ambíguas, sendo a taxa de resposta maior do que no questionário da web. Ao utilizar este tipo de questionário é necessário tomar cuidado com as interpretações das respostas.

A metodologia utilizada na presente pesquisa alinhou essas técnicas mencionadas com a aplicação de um questionário comum a três atividades, conforme abaixo (o mesmo questionário foi utilizado como guia):

1. **Entrevista face-a-face**: O pesquisador visitou as fábricas de software para a obtenção das entrevistas, sendo que cada participante fora indicado por participantes anteriores (Bola de Neve). Todas as entrevistas foram gravadas com a permissão do entrevistado;
2. **Entrevista por telefone**: Algumas entrevistas foram feitas por telefone, quando houve incompatibilidade de tempo entre as partes envolvidas. Também foram gravadas com a permissão do entrevistado;
3. **Coleta por e-mail**: Foram enviados a alguns participantes um link para a realização da coleta de dados online, via Google Forms. O tipo de amostragem utilizada aqui foi a Acidental;

A Tabela 3.1 mostra as quantidades de pessoas envolvidas na coleta dos dados, os participantes nas três modalidades, por amostragem Acidental e Bola de Neve, e por e-mail:

Tabela 3.1: Técnica aplicada na realização de *survey*.

| **Processo** | **Quantidade** | **Observação** |
|---|---|---|
| Entrevista face-a-face | 14 | Gravadas |
| Entrevista por telefone | 5 | Gravadas |
| Coleta por e-mail | 17 | Google Forms |
| TOTAL | 36 | |

## 3.3 Condução dos Trabalhos

O planejamento do presente trabalho foi materializado em uma sequência de atividades que foi empreendida pelo pesquisador.

### 3.3.1 Sequência das Atividades

As seguintes atividades foram desenvolvidas:

1. **Atividade 1 - Elaboração de questionário**: O questionário, objeto do *survey* descrito na Seção 3.2, foi desenvolvido para emprego nas entrevistas face-a-face, e também para disponibilização na web, utilizando a ferramenta **Google Forms**.

2. **Atividade 2 - Execução do estudo-piloto**: A execução de *pilot-survey*, conforme previsto em Linaker et al. [91], foi um dos pontos cruciais para a pesquisa, que foram validadas as questões previstas no questionário por meio de um "ensaio". Para a realização deste piloto foram escolhidas três empresas de software para figurar na amostra da análise inicial dos dados. Os três participantes foram escolhidos dentre os possuidores de certificação em metodologias ágeis, para que as perguntas pudessem ser ajustadas de forma coerente após a realização das três entrevistas do piloto.

3. **Atividade 3 - Revisão do questionário**: A revisão foi feita conforme os resultados apontados no *pilot-survey*, onde algumas perguntas sofreram ajustes conforme o que se segue:

    - Algumas questões abertas foram modificadas para questões fechadas, a fim de permitir análise quantitativa e coerência. Por exemplo: uma das questões abertas era "Qual o seu cargo na empresa?", que foi modificada para uma questão de múltipla escolha, onde o entrevistado deveria marcar apenas uma opção dentre as oferecidas. Uma outra questão que foi modificada pelo mesmo motivo foi a que perguntava "Quanto tempo de experiência em projetos ágeis?".
    - O questionário teve de ser enxugado, pois a primeira entrevista do *pilot-survey* levou cerca de uma hora para ser finalizada. Some-se a isto outros minutos gastos com a recepção do entrevistador e as conversas iniciais, o tempo ficou bastante dispendioso. Foi notório a percepção do entrevistador de que os participantes demonstraram, durante o *pilot-survey*, algum grau de cansaço, pois as últimas perguntas evidenciaram isto. Desta forma algumas questões foram eliminadas, e o critério para a eliminação foi a retirada daquelas que apresentavam alguma redundância. Por exemplo: A pergunta "Sobre princípios ágeis, cite quais você conhece" foi eliminada porque uma outra pergunta similar já a respondia "Na sua equipe, é possível perceber algum princípio ágil sendo empregado? Qual (is)?". Uma outra pergunta eliminada foi "Quais são as práticas ágeis mais utilizadas?" que já era respondida em outro lugar.
    - Ao observar o Apêndice E, a questão de número 12 apresentava 60 práticas ágeis para o entrevistado sinalizar se elas eram consideradas sem importância, pouco importante, importante, muito importante ou extremamente importante. Essa foi a experiência mais cansativa da entrevista. Após o *pilot-survey*, essas práticas foram reduzidas para 40. A Seção 6.10, Ameaças à Validade, apresenta uma análise particular a respeito disto.

4. **Atividade 4 - Visitar empresas-alvo e conduzir entrevistas**: As empresas consideradas alvos da pesquisa foram visitadas, localmente, como parte da atividade

de pesquisa de campo prevista para este estudo. As indicações vieram dos próprios entrevistados por meio da técnica *Bola de Neve* já mencionada.

5. **Atividade 5 - Compilar relatórios**: Esta atividade compreendeu o estudo e a classificação, bem como a análise dos resultados obtidos no *survey*. Esses resultados foram formatados e organizados em tabelas e gráficos.

6. **Atividade 6 - Ajustes finais**: De posse dos relatórios, gravações de áudio e outras documentações acumuladas durante os trabalhos de pesquisa, foram realizados os ajustes finais e a formatação do trabalho.

# Capítulo 4

# Resultados e Discussão

A mensuração final de qualquer trabalho é um dos meios pelos quais os dados são apresentados para uma melhor compreensão do fenômeno. Por isto, a mensuração está presente nas ciências e a real consistência do estudo empírico quantitativo somente pode ser viável se utilizadas práticas de mensuração. Desta forma, a teoria da medição em várias ciências, nela inclusa a Engenharia de Software, tem avançado bastante, por meio de análise mais apurada acerca de variáveis concretas e até de variáveis abstratas, tais como tensão, satisfação, lealdade, etc.

A pesquisa de campo realizada no presente trabalho teve por finalidade responder à segunda e terceira questões de pesquisa, (Q2) Qual a percepção das empresas sobre a adoção de práticas ágeis? e (Q3) Qual a extensão da adoção de práticas ágeis nas empresas? Para respondê-las, foram realizadas entrevistas face-a-face, por telefone e por e-mail.

A coleta de dados foi realizada durante o período de 01 de março a 10 de maio de 2017, através de entrevistas estruturadas e coletas por e-mail, cujo roteiro encontra-se evidenciado no Apêndice E. O roteiro sofreu ligeira modificação após as três primeiras entrevistas, consideradas o projeto-piloto do *survey*, a fim de melhorar o tempo gasto nas seções com os participantes. A pesquisa contou com 36 participantes. Todas as perguntas do questionário foram analisadas e discutidas neste trabalho. A formatação dos resultados foi o trabalho final, onde os áudios obtidos durante as entrevistas foram resumidos em um texto, com a numeração dos entrevistados utilizando-se os códigos P18 a P36, que correspondiam aos 19 participantes por meio de entrevistas face-a-face e por telefone. Em seguida, o texto foi tabulado em algumas planilhas eletrônicas para possibilitar agrupamentos e análises, e nessas planilhas foram também inseridas as 17 entrevistas obtidas por e-mail. A partir daí, os dados foram tabulados, tanto qualitativamente, pela análise, apresentação e apreciação das narrativas das respostas, quanto quantitativamente, onde utilizou-se o Programa R [124] para a plotagem dos gráficos. Os *scripts* utilizados para a realização das plotagens estão materializados no Apêndice F. Durante as plotagens,

algumas técnicas foram empregadas para a melhor compreensão e adaptação aos tipos de perguntas utilizadas no questionário. Nesse proceder, gráficos de barras, de colunas e também uma escala do tipo Likert [90] foram empregados para visualizar e comentar os resultados.

## 4.1 O Perfil dos Participantes

O perfil dos participantes pode ser analisado pela apreciação das respostas obtidas nas perguntas 2, 3 e 4 do questionário, conforme apresentado na Tabela 4.1. A pergunta número 1 solicitava o e-mail do participante que, por razões éticas, deixou de figurar neste trabalho.

A pergunta 2 solicitava que o participante escolhesse apenas uma das alternativas, entre "Supervisor", "Administrador", "Gerente", "Arquiteto", "Analista", "Testador", "Suporte", "Desenvolvedor", "Webdesigner" e ainda dando a opção para que ele escrevesse outro tipo de cargo ocupado dentro da empresa, caso nenhuma das opções se enquadrasse em seu perfil. Alguns participantes perguntaram se poderiam assinalar mais de uma resposta e quando isto ocorria, foi solicitado para que eles optassem, dentre duas ou três respostas previamente selecionadas, qual a que mais poderia corresponder à sua realidade.

A pergunta número 3 tratava-se do nível de formação dos participantes, onde os mesmos puderam escolher entre as opções de "Doutorado", "Mestrado", "Graduado" e "Nível Médio". Nesse aspecto, ressalta-se que as escolhas dos participantes foram feitas pelas próprias indicações entre eles (técnica Bola de Neve) [91]. A amostragem final definida para a apuração dos resultados apresentou, em quase toda a sua totalidade, o nível de Graduação, apenas quatro possuíam o nível Mestrado.

A pergunta de número 4 solicitava ao participante que ele indicasse o tempo de experiência em desenvolvimento de software em geral — o tempo aqui englobava não só as experiências obtidas em metodologias ágeis, mas também as tradicionais.

A Tabela 4.1 apresenta um resumo extraído dos resultados dessas três perguntas. P# indica o número do participante, EMPRESA (EMP), o código da empresa do participante, TIPO mostra se a entrevista foi realizada por e-mail, face-a-face ou telefone, GARGO mostra a função do participante, GRADUAÇÃO (GRAD) o seu nível de escolaridade e EXPERIÊNCIA (EXP) a faixa do tempo de experiência em desenvolvimento de software (anos). O resultado da pergunta número 5, METODOLOGIA (MET), na 6ª coluna, é analisada na próxima seção.

Tabela 4.1: Perfil dos Participantes.

| P# | EMP(**) | TIPO | CARGO | GRAD. | EXP.(*) | MET. |
|---|---|---|---|---|---|---|
| P1 | A | email | Gerente | Graduado | 1 e 2 | NA |
| P2 | B | email | Gerente | Graduado | 1 e 2 | LEAN |
| P3 | C | email | Gerente | Graduado | 9 e 12 | FDD |
| P4 | D | email | Gerente | Graduado | 6 e 8 | NA |
| P5 | E | email | Gerente | Graduado | 1 e 2 | NA |
| P6 | B | email | Gerente | Graduado | 6 e 8 | SCRUM |
| P7 | F | email | Gerente | Graduado | 1 e 2 | FDD |
| P8 | G | email | Administrador | Graduado | 1 e 2 | SCRUM |
| P9 | H | email | Gerente | Graduado | 6 e 8 | NA |
| P10 | I | email | Administrador | Graduado | 17 e 20 | SCRUM |
| P11 | J | email | Desenvolvedor | Graduado | 6 e 8 | KANBAN |
| P12 | K | email | Gerente | Mestrado | 13 e 16 | XP |
| P13 | L | email | Administrador | Mestrado | 3 e 5 | SCRUM |
| P14 | B | email | Analista | Graduado | 6 e 8 | NA |
| P15 | M | email | Suporte | Graduado | 1 e 2 | NA |
| P16 | N | email | Desenvolvedor | Graduado | 3 e 5 | XP |
| P17 | E | email | Analista | Graduado | 1 e 2 | NA |
| P18 | G | Face-a-face | Analista | Graduado | 1 e 2 | SCRUM |
| P19 | O | Face-a-face | Desenvolvedor | Graduado | 6 e 8 | SCRUM |
| P20 | D | Face-a-face | Gerente | Graduado | 1 e 2 | SCRUM |
| P21 | B | Face-a-face | Desenvolvedor | Graduado | 6 e 8 | XP |
| P22 | C | Face-a-face | Gerente | Graduado | 17 e 20 | XP |
| P23 | P | Face-a-face | Gerente | Graduado | 13 e 16 | SCRUM |
| P24 | A | Face-a-face | Desenvolvedor | Graduado | 9 e 12 | KANBAN |
| P25 | N | Face-a-face | Desenvolvedor | Graduado | 6 e 8 | SCRUM |
| P26 | Q | Face-a-face | Arquiteto | Graduado | 9 e 12 | SCRUM |
| P27 | L | Face-a-face | Analista | Graduado | Mais de 20 | KANBAN |
| P28 | 0 | Face-a-face | Analista | Graduado | 6 e 8 | SCRUM |
| P29 | Q | Face-a-face | Desenvolvedor | Mestrado | 9 e 12 | XP |
| P30 | C | Face-a-face | Analista | Mestrado | 6 e 8 | SCRUM |
| P31 | R | Face-a-face | Administrador | Graduado | Mais de 20 | SCRUM |
| P32 | M | Telefone | Administrador | Graduado | Mais de 20 | XP |
| P33 | S | Telefone | Desenvolvedor | Graduado | 13 e 16 | NA |
| P34 | K | Telefone | Gerente | Graduado | 9 e 12 | SCRUM |
| P35 | O | Telefone | Analista | Graduado | 3 e 5 | SCRUM |
| P36 | O | Telefone | Arquiteto | Graduado | 6 e 8 | SCRUM |

(*) Tempo de experiência em Desenvolvimento de Software, sem levar em conta as metologias ágeis.

(**) Empresas onde trabalham os participantes da pesquisa. As empresas possuem um número aproximado de funcionários conforme a Tabela 4.2:

Tabela 4.2: Empresas participantes.

| EMPRESA | NÚMERO APROXIMADO DE FUNCIONÁRIOS |
|---|---|
| A | 150 |
| B | 500 |
| C | 1000 |
| D | 700 |
| E | 50 |
| F | 70 |
| G | 20 |
| H | 50 |
| I | 80 |
| J | 70 |
| K | 250 |
| L | 50 |
| M | 70 |
| N | 70 |
| O | 1000 |
| P | 200 |
| Q | 70 |
| R | 750 |
| 5 | 200 |

### 4.1.1 Apreciação Sintética das Áreas de Atuação

Após alimentar o programa R com os dados da questão número 2, conforme apresentado na Tabela 4.1 até a terceira coluna, uma matriz de sumarização (%) abaixo representada foi obtida, seguida do gráfico correspondente mostrado na Figura 4.1.

```
1 === sumario (resultados em porcentagens)===
2 36.111111 & 22.222222 & 19.444444 & 13.888889 & 5.555556 & 2.777778
```

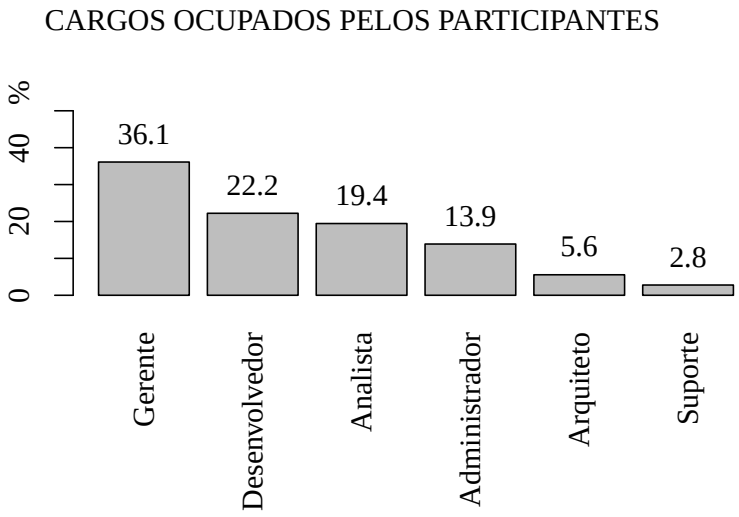



Figura 4.1: Apreciação sintética das áreas de atuação dos participantes da pesquisa..

É notório o volume dos participantes que ocuparam cargos de gerente de projeto em suas respectivas empresas, com um percentual em torno de 36%. Cabe ressaltar que um dos participantes que ocupava o cargo de gerente, o P20, também relatou ser o *Scrum Master*, responsável pelo treinamento das equipes ágeis, exercendo o papel de orientar os participantes a não saírem do programa estabelecido. Segundo o participante P20:

> "...minha empresa possui cerca de 500 funcionários, trabalhando em diversas frentes, de forma destacada junto aos vários clientes que possuímos, e meu papel, em meu grupo, é fazer com que as metodologias ágeis funcionem sem interrupção, de forma a impulsioná-los à resolução de seus problemas com maior agilidade."

Em segundo lugar, com 22% do universo, apareceram os desenvolvedores, que puderam dar sua contribuição, seguido dos analistas (19%), administradores (13,8%), arquitetos (5%), cada qual com sua visão pessoal sobre a temática. Em menor escala o pessoal de suporte (2,7%), normalmente um profissional que realiza a alavancagem dos trabalhos, realizando o monitoramento do andamento dos processos, como por exemplo, nas empresas B e C, onde existe a presença do Scrum Master com essa função.

### 4.1.2 O Nível de Escolaridade

O nível de escolaridade dos participantes foi obtido por meio da questão número três, a quarta coluna da Tabela 4.1 e sua análise dispensa o uso de ferramentas (R), pois apenas quatro deles (11,1%) possuíam o nível mestrado completo e os 32 restantes (88,9%), com o nível de graduação. Esses dados foram suficientes para se estimar o potencial do material humano à disposição da análise.

### 4.1.3 Apreciação Sintética da Experiência dos Participantes

A quinta coluna da Tabela 4.1 mostra as respostas referentes à quarta questão do questionário. Abaixo, a matriz referente à sumarização desses dados obtidos no programa R, seguida do gráfico correspondente, Figura 4.2.

```
1  === sumario (resultados em porcentagens)===
2  Entre 1 e 2 Anos    25.000000
3  Entre 3 e 5 Anos    8.333333
4  Entre 6 e 8 Anos    27.777778
5  Entre 9 e 12 Anos   13.888889
6  Entre 13 e 16 Anos  11.111111
7  Entre 17 e 20 Anos  5.555556
8  Mais de 20 Anos     8.333333
```

Percebe-se que cerca de 20% dos participantes possuem um tempo de experiência em desenvolvimento de software situado entre 6 e 8 anos, ocupando a primeira posição no

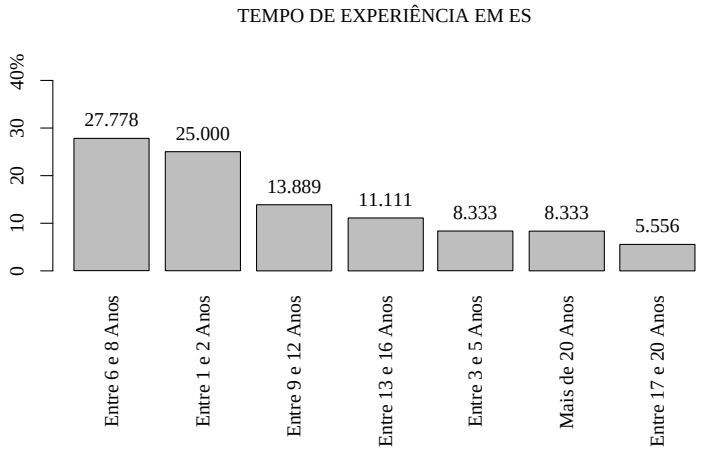



Figura 4.2: Apreciação sintética do tempo de experiência dos participantes da pesquisa.

ranking desta análise. A segunda posição foi ocupada pelos participantes que possuíam 1 e 2 anos de experiência, porém, boa parte deles obtida por participantes que enviaram suas respostas pelo e-mail. A experiência de mais de 20 anos ocupou a quinta posição, com cerca de 8% dos participantes. Não houve a possibilidade de escolha dos indivíduos para a pesquisa, haja vista ela ser de forma aleatória, sendo os participantes indicados pelos seus antecessores (técnica Bola de Neve), embora tenha havido a recomendação de que isto fosse feito.

### 4.1.4 Avaliação Final do Perfil da Amostra

No geral, o perfil dos participantes pode ser traçado como integrados e adaptados às metodologias de desenvolvimento de software, com maioria quase absoluta de profissionais de nível 3º grau e boa média de experiência em desenvolvimento, sendo bem adaptados aos cargos que ocupam em suas empresas.

## 4.2 Evolução da Adoção das Metodologias Ágeis

As metodologias ágeis é tema recorrente em várias conferências da Engenharia de Software pelo mundo. Embora se tenha passado quase 20 anos depois de seu surgimento, ainda continuam estudos e não faltam pesquisadores que se aprofundam no tema. Assim sendo, nada mais que oportuno a apresentação de dados que possam refletir a real situação de como elas se encontram hoje, seus principais obstáculos e suas fórmulas de sucesso e também seus fracassos.

### 4.2.1 Cenários

Neste trabalho, a evolução das metodologias ágeis está sendo medida em relação aos últimos nove anos, desde a pesquisa realizada por Dybå e Dingsøyr [44] e Dikert et al. [40], este último, um trabalho realizado em 2016 e chegando até os dias atuais. Para que isto pudesse ser realizado empiricamente, os cenários inicial e final foram comparados com as pesquisas daquela época (2008), conferindo os dados em 2016 e também por meio da questão número 5 do questionário da pesquisa de campo (2017). Entretanto, um cenário intermediário também se fez necessário, onde levou-se em consideração os dados apresentados por Melo et al. [98] em maio de 2012, que mostrou um relativo crescimento da adoção de metodologias ágeis, especificamente Scrum. Os dados podem ser vistos na Tabela 4.3

### 4.2.2 Formatação e Análise

Reportando-se à Tabela 4.1, a legenda "NA" representa a informação de que o participante nunca utilizou metodologia ágil em sua empresa, até o presente momento em sua vida profissional — mas que utiliza Metodologias Tradicionais, na atualidade. Para uma melhor compreensão, em um primeiro momento, os dados foram formatados em dois grupos distintos, sendo o primeiro com os dados sobre as metodologias que mais se sobressaíram, SCRUM e XP, colocando-as lado a lado para comparação conforme a Tabela 4.3, e o segundo grupo, mostrado na Tabela 4.4, com as outras metologias citadas e também as respostas de quem nunca as utilizaram, mas reportando apenas os estudos referentes à pesquisa de campo deste trabalho, a qual rotulou-se de Mazuco(2017).

(a) Dybå x Melo x Mazuco

A Tabela 4.3 evidencia as colunas M, Dybå/2008, Melo/2012 e Mazuco/2017, com os seguintes elementos: A coluna M mostra os dados que representam os resultados das três pesquisas estudadas, constando de **XP**, **SCRUM** e **Outras**. Cabe ressaltar que a variável "Outras" representa os resultados diferentes de Scrum e XP, podendo ser uma outra metodologia ágil ou ainda as metodologias tradicionais; A coluna Dybå/2008 formata e apresenta os resultados de sua pesquisa realizada em 2008 [44]; A coluna Melo/2012 formata e apresenta os dados da pesquisa empreendida por Melo em 2012 [98] e a coluna Mazuco/2017 sumariza e apresenta os dados constantes da Tabela 4.1, especificamente os dados da coluna METODOLOGIA, portanto os dados da pesquisa de campo deste trabalho.

Tabela 4.3: Evolução das metodologias Scrum e XP, em comparação com outras metodologias desde 2008.

| **M** | **Dybå/2008** | **Melo/2012** | **Mazuco/2017** |
|---|---|---|---|
| XP | 76% | 7.3% | 16.7% |
| SCRUM | 3% | 51.1% | 44.4% |
| Outras | 21% | 41.6% | 38.9% |

Para esses dados, um script rodando no programa R (ver o Apêndice F) obteve o gráfico mostrado na Figura 4.3. Evidencia-se que em 2008, com o estudo de Dybå, a metodologia XP tinha a preferência dos desenvolvedores de software, um resultado esperado para a época, pois a metodologia estava bem consolidada entre os praticantes, bem aliados aos pressupostos de Kent Beck [15] e seus adeptos. Portanto, em 2008 a curva do gráfico acentuado em XP justificava-se pela sua forte presença, em detrimento de Scrum de Schwaber e Sutherland [133], que apenas despontava no cenário, ocupando uma terceira posição, ainda bem abaixo das outras metodologias juntas.

Em 2012, com os estudos de Melo et al. [98], o panorama se inverte, onde houve uma queda na adoção da metodologia XP, desta vez dando lugar à metodologia Scrum que passou a ocupar 51% da preferência. Entretanto, é digno de menção que outras metodologias passassem a ocupar o lugar que era quase exclusivo de XP e essa tendência vem a se consolidar no presente estudo, conforme apresentado na Figura 4.3, apontado por Mazuco(2017) com um aumento considerável (38,9%). Percebe-se que com o passar do tempo as metodologias ágeis apresentaram uma tendência a se tornar homogêneas, embora Scrum nos dias de hoje ainda ocupe a primeira posição do ranking com 44,4% da preferência.

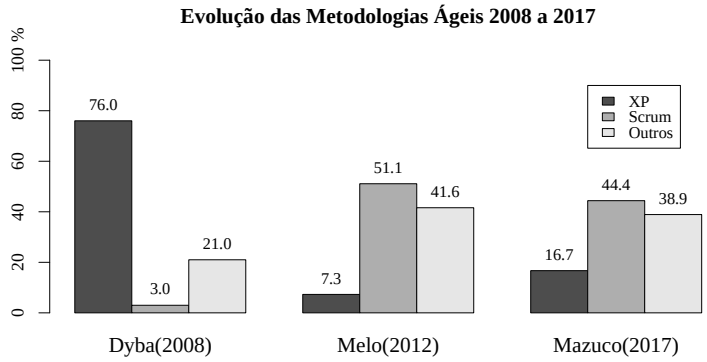



Figura 4.3: Comparação entre Dybå, Melo e Mazuco.

Chamou a atenção o considerável aumento da adoção de outros tipos de metodologias que não Scrum e XP. É notório que esse fenômeno também foi observado nos três cenários descritos. Em 2008, 21% das preferências era destinado a essas metodologias, e em 4 anos cresceram o dobro. Dybå rotulou essas metodologias de Lean, que entre elas era a que mais se sobressaía, e as demais de "Genéricas". Melo especificou bem as outras metodologias diferentes de Scrum e XP denominando-as, ora como um misto entre as duas, ora como tipos customizados entre elas e até uma variante do Scrum denominada Scrumban. Essas, juntas, ocupavam 41,6% da preferência. Em 2017 há uma pequena diminuição na ocorrência dessas outras metodologias, portanto mantendo essa tendência, fato que mereceu uma análise em separado.

(b) Outras Metodologias (Mazuco/2017)

Neste estudo, o intuito era descobrir, dos 38,9% que responderam "Outras", quais eram as metodologias utilizadas, mesmo quem não adotassem metodologias ágeis, ou seja: o que representa os dados compilados em "Outras". Esses dados foram transportados para uma outra tabela, conforme apresentado na Tabela 4.4, que sumariza os resultados diferentes de SCRUM e XP da pesquisa de campo deste trabalho, esses dados também são mostrados na quinta coluna da Tabela 4.1, onde a variável **M** identifica o fato ocorrido ("Metodologias Tradicionais", "Lean", "FDD"e "Kanban" e a variável **Mazuco/2017**, a sumarização dos resultados.

Tabela 4.4: Outros resultados evidenciados em Mazuco/2017.

| **M** | **Mazuco/2017** |
|---|---|
| Metodologias Tradicionais | 57.1% |
| Lean | 7.1% |
| FDD | 14.3% |
| Kanban | 21.5% |

Neste cenário, pouco mais da metade dos participantes que não adotavam Scrum ou XP (38,9% da pesquisa de Mazuco/2017) informaram que utilizavam as Metodologias Tradicionais, com um percentual de ocorrência de 57,1%. Dentre as metodologias Lean, FDD e Kanban. Neste cenário, dentre as metodologias ágeis, Kanban é a que mais se sobressaiu, ocupando a segunda posição com 21,5%, ficando FDD e Lean nas posições subsequentes com 14,3% e 7,1%. O gráfico da Figura 4.4 mostra esse panorama.

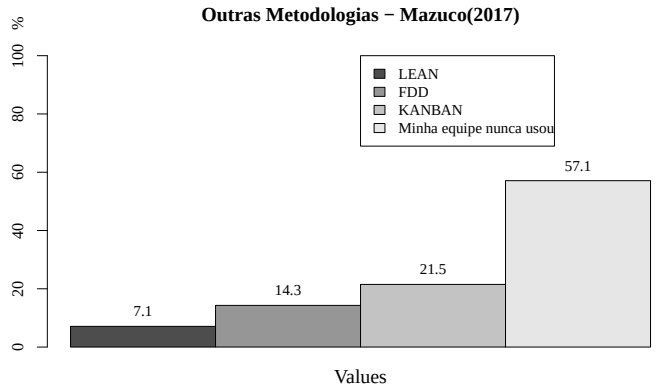



Figura 4.4: Outras metodologias em Mazuco(2017).

## 4.3 Níveis de Adoção das Práticas Ágeis

A pergunta número 6 do questionário indagava aos participantes que eles respondessem quais das práticas ágeis previamente enumeradas eles mais utilizavam no seu dia-a-dia na empresa onde trabalhavam, reportando-se aos dias de hoje. A Tabela 4.5 apresenta as respostas de cada participante.

Tabela 4.5: Quadro de escolhas das práticas ágeis.

| P# | PR1 | PR2 | PR3 | PR4 | PR5 | PR6 | PR7 | PR8 | PR9 | PR10 | PR11 | PR12 | PR13 | PR14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | | | | | | x | | | | | | | | |
| P2 | | x | | | | | | x | | | x | x | | |
| P3 | | | x | | | | x | x | | | | | | x |
| P4 | | x | | | | | x | | | x | | | | x |
| P5 | x | | x | | x | | | | | x | | | | |
| P6 | x | | | | | | | x | | x | x | | | |
| P7 | x | | | | x | x | | | | | | x | | |
| P8 | | | | | x | x | x | | | | | | | x |
| P9 | | | x | | | x | | | | x | | | x | |
| P10 | x | | | | | | x | | | x | | x | | |
| P11 | | | | x | x | | x | x | | | | | | |
| P12 | | | x | | | x | | x | | | | | | |
| P13 | x | x | x | x | x | x | x | x | x | | | | x | x |
| P14 | | x | x | | | | | | | | | | | |
| P15 | | | | | | | | | x | | x | | | |
| P16 | x | | | x | x | | | | | x | x | | | |
| P17 | | | | | | | x | | | | | | | |
| P18 | | | | | x | x | x | | | | | | | |
| P19 | | | | x | | x | | x | | | x | | | |
| P20 | | x | | x | x | x | x | | x | x | x | | | x |
| P21 | | | x | | | x | x | x | | | x | | x | x |
| P22 | x | | x | x | x | | x | x | x | x | x | x | x | x |
| P23 | | | | x | | x | | | | | x | | x | x |
| P24 | | x | x | x | | | x | x | | x | | | | |
| P25 | | | x | | x | x | x | | | | | | | |
| P26 | | | x | x | | x | | | x | | | | x | |
| P27 | | x | x | x | x | | | | x | | x | | | |
| P28 | | | | | x | x | x | x | x | | x | | x | x |
| P29 | | | x | | | | x | | | x | x | | x | |
| P30 | x | x | x | x | x | x | x | x | x | | x | | | x |
| P31 | | | | x | x | x | x | | x | | x | | | |
| P32 | | x | x | x | x | x | x | x | x | x | x | | | x |
| P33 | x | x | x | | x | | x | x | | | x | | x | |
| P34 | | | x | x | x | x | x | | x | | x | | | x |
| P35 | | x | x | | x | x | | x | x | | x | | | x |
| P36 | | x | x | | x | x | x | x | x | | x | x | | x |
| Σ | 9 | 12 | 19 | 14 | 19 | 20 | 21 | 16 | 13 | 11 | 19 | 5 | 9 | 14 |

LEGENDA:
P# Participante
PR1 Design Incremental
PR2 Implantação Diária
PR3 Integração Contínua
PR4 Discussão sobre Qd Branco
PR5 Sentar Junto
PR6 Reunião em Pé
PR7 Cliente Junto
PR8 Código Compartilhado
PR9 Estórias de Usuários
PR10 Programação em Pares
PR11 Prototipação
PR12 TDD
PR13 Releases Curtas
PR14 Entrega de 2 a 4 Semanas

Σ Contagem das ocorrências

Na elaboração desta questão, foram escolhidas 14 práticas ágeis de interesse desta pesquisa, isto porque essas são as práticas ágeis estabelecidas em XP, XP2 e Scrum de maior importância para figurarem como opções, em que o participante deveria marcar as que ele estava utilizando e ainda tecesse algum comentário sobre elas. Poucos participantes comentaram, a maioria se limitou a assinalar as que eles mais adotavam.

Obteve-se a plotagem do gráfico mostrado na Figura 4.5.

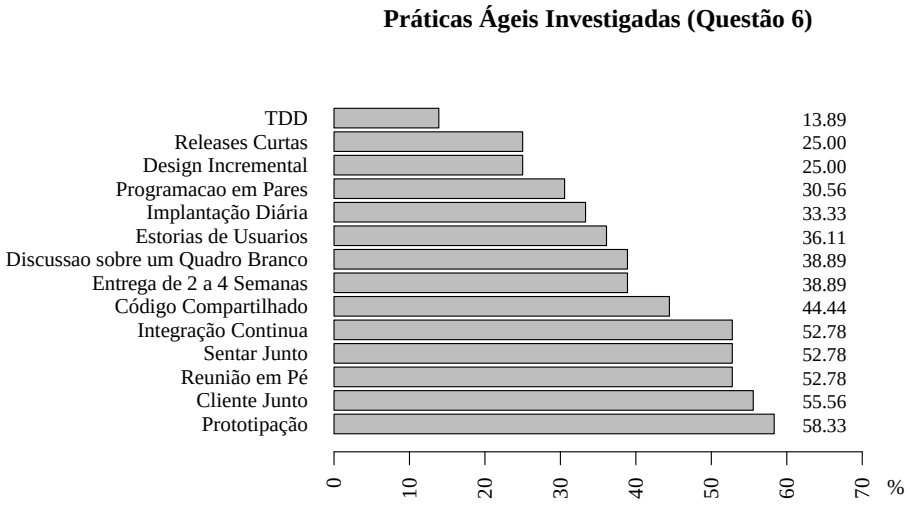



Figura 4.5: Níveis de Adoção de Práticas Ágeis por Participante.

### 4.3.1 Avaliação das Práticas de Menor Aceitação

As práticas de menor adoção na preferência das empresas, nelas o TDD, Releases Curtas, Design Incremental, Programação em Pares, Implantação Diária e Estórias de Usuários, embora figurassem entre as menos adotadas, mostraram resultados significativos, refletindo um quadro positivo para a adoção das 14 práticas investigadas.

**(a) Test Driven Develpment (TDD)**: Segundo a evidência, a prática menos adotada foi a Test Driven Development, com 13,89% de ocorrência. O fato dela ser a menos adotada pelas empresas não a exclui totalmente, porque 13,89% é um número considerável. Não obstante, o participante número 30 (P30), que foi um dos que sinalizou a adoção dessa prática, mencionou que "...nos últimos dois anos nossa empresa passou a utilizar constantemente a prática do *Test Driven Development*", um indício de que,

embora tenha sido a menos adotada, é bem utilizada nas empresas. Um dos motivos que a levaram para o último lugar na colocação do ranking das quatorze, e embora alguns participantes da pesquisa possuíssem pouco tempo de experiência, convém ressaltar o que afirmam Mazuco e Canedo [97]:

> "...alguns desenvolvedores profissionais estão bem acostumados com o processo habitual de desenvolvimento, o de primeiro implementar o código e depois realizar os testes. Mas essa prática vem diminuindo à medida que novos estudos introduzem uma nova maneira de pensar e de agir. O TDD cumpre bem esse papel, em um cenário onde a fábrica de software adote metodologias ágeis."

**(b) Releases Curtas e Design Incremental**: Essas duas, um pouco além de TDD, com 25% de adoção, ocupando os últimos lugares, também são bem adotadas pelas empresas. Um indício de que a prática Releases Curtas foi pouco ranqueada é que havia a presença de uma outra prática similar como opção para ser assinalada, a Entrega de 2 a 4 Semanas, o que pode ter influenciado no momento da escolha. Sobre a prática de Design Incremental, P30 assinalou que "...procuramos colocar sugestões nas funcionalidades e assim ajudando a melhorar cada vez mais o design.", visto que a prática do Design Incremental inclui previsão de futuro, Boehn [19] assevera:

> "Design incremental, em vez de desenvolver um projeto antecipado detalhado antes da implementação, investe-se no design do sistema todos os dias conforme as experiências do passado. A viabilidade e a prudência do projeto antecipado têm mudado nosso ambiente de negócios..."

Em outras palavras, é comum ver aquilo que poderia ser útil no futuro, deixar para resolver no futuro quando houver a certeza da necessidade.

**(c) Programação em Pares**: A programação em Pares tem sido uma das primeiras práticas ágeis adotadas por empresas em processo de migração. Em nosso estudo ocupou o 11º lugar. Os indícios desse ranqueamento, 30,56% da preferência, pode ser explicado por várias formas, e vários autores têm relatado isto, como Williams et al. [162]:

> "A indústria de software vem praticando a programação em pares (com dois programadores funcionando lado a lado em um computador com o mesmo problema) com grande sucesso há anos, mas as pessoas que não o tentaram muitas vezes rejeitam a ideia como um desperdício de recursos."

P30, em uma de suas respostas, disse que "...usamos a programação em pares, mas ela ainda é um complicador para nós [...] ter de empenhar dois programadores em um único código é dispendioso, embora saibamos que haja um bom aprendizado..."

**(d) Implantação Diária e Estórias de Usuários**: Essas duas práticas, com 33,33% e 36,11% respectivamente, também compondo a segunda metade do ranking, foram comentadas pelos participantes como utilizadas, mas de forma parcial. P30 citou a primeira como "...realizamos a implantação diária, mas nem sempre diária" e que "...são os analistas de requisitos que elaboram as estórias de usuários.", e P34 relatou ainda que "...usamos bastante as estórias de usuários, inclusive utilizamos uma ferramenta chamada Tuleap".

A prática de estórias de usuários ocupa a melhor posição na lista das práticas menos adotadas porque ela é, por vezes, considerada como fundamental para as indústrias no processo de desenvolvimento de software, no entanto, algumas empresas utilizam o modelo tradicional de levantar requisitos, o que é considerado por muitos como uma tarefa bastante difícil.

### 4.3.2 Avaliação das Práticas de Média Aceitação

As práticas de **Discussão sobre um Quadro Branco** e **Entrega de 2 a 4 Semanas** ocuparam, cada uma, o centro da escala de adoção, com 38,89% da preferência. Foram denominadas médias apenas para serem referenciadas neste estudo de que ocuparam mais ou menos o centro das preferências entre as quatorze avaliadas.

**(a) Discussão Sobre um Quadro Branco**: P30 referiu-se à prática de Discussão sobre um Quadro Branco como "...usamos bastante *postit*, com o quadro dividido em colunas." e P25 respondeu que "...o pessoal vai a frente do quadro e com o auxílio de um marcador de textos expõe as suas ideias.". Para Beck [17], um dos principais problemas na gerência de projetos é a comunicação e Schwaber e Sutherland [133] pregam que essa comunicação deve ser feita face-a-face, sendo o uso do quadro branco fundamental para auxiliar nas discussões. Por isto muitas empresas não dispensam essa prática, entretanto, apoiando a evidência, outras não a fazem uso, como P19 em sua entrevista:

> "Usamos mais uma conversa em pé de 10 minutos, pois achamos que longas discussões sobre o quadro branco onera o tempo da equipe e é pouco produtiva. Mas é comum os arquitetos utilizarem... fazemos reuniões sentados, às vezes, mas não temos a chance de usar o quadro branco, embora me pareça uma boa ideia."

**(b) Entrega de 2 a 4 Semanas**: Em termos, a prática de entregar o software funcionando entre 2 e 4 semanas compõe o terceiro princípio do Manifesto Ágil [17] que aborda que a frequência da entrega de software funcional tem de ser alta. Isto significa que a equipe precisa ter em mente que após um curto período de tempo terá de apresentar

o software ao cliente. Pela evidência, cerca de metade das empresas cumprem ou adotam esse princípio. Mas cabe ressaltar que algumas delas adotam um tempo menor que este, como relata P32 em sua entrevista "...estamos praticando a entrega de 2 a 3 semanas, que para nós está sendo mais viável." Isto pode ser justificado, quando a questão da pressão financeira estar exercendo forte influência no planejamento das entregas.

### 4.3.3 Avaliação das Práticas de Maior Aceitação

Curioso notar que das práticas de média aceitação para as de maior aceitação há um salto considerável de quase 10%. Isto reforça o fato de que as práticas da primeira metade são bem mais adotadas, com um percentual quase discrepante. Assim, as práticas Código Compartilhado, Integração Contínua, Sentar Junto, Reunião em Pé, Cliente Junto e Prototipação foram as que apresentaram o maior índice de adoção nas empresas de acordo com as respostas coletadas.

**(a) Código Compartilhado**: A prática de compartilhar o código é adotada por 44,44% das empresas investigadas. O pessoal sempre se refere a ela como um dos fatores chave na produção de código, pois agiliza-a. A questão desta prática estar entre as mais preferidas deve-se ao fato de que ela, por vezes, é utilizada em qualquer metodologia de desenvolvimento, ágil ou tradicional. Mas alguns participantes, como P1, P4, P5, P7-P10, P14-P18, P20, P23, P25-P27, P29, P31 e P34 não a mencionaram em seus relatos.

Para Braithwaite [21], uma das razões dessa evidência, pode ser observada nas situações em que os membros da equipe podem se ajudar mutuamente:

> "Membros da equipe amplamente separados precisam manter uma identidade comum como técnica de solucionar problemas. Eles precisam compartilhar direitos e responsabilidades em relação aos trabalhos, seus e dos outros. Portanto, todos os membros da equipe, em todos os lugares, usam uma única base de código compartilhada."

Com o surgimento do conceito, várias ferramentas foram criadas para otimizar o trabalho das equipes de desenvolvimento de software, fazendo com que essa prática tivesse forte aceitação na comunidade.

**(b) Integração Contínua, Sentar Junto e Reunião em Pé**: Essas três práticas ocuparam a terceira posição no ranking das práticas ágeis mais populares, com 52,78% de adoção. Um dos indícios dessas práticas figurarem o topo, pode ser encontrado nas afirmações de Fowler[53]:

> "A Integração Contínua é uma prática de desenvolvimento de software em que os participantes de um time integram seu trabalho de forma frequente, pelo menos uma vez por dia. Cada integração é verificada por uma *build* automatizada para detectar erros o mais rápido possível. Muitas equipes consideram que essa abordagem leva a uma significante redução nos problemas de integração, permitindo que um time desenvolva software coeso bem mais rapidamente."

Apoiando essa evidência, P30 relatou que "Existe um líder com dois desenvolvedores responsáveis por integrar as funcionalidades que vão sendo implementadas..."

Sobre as reuniões em pé, P36 revelou que "'...nossas reuniões em pé não ultrapassam 15 minutos", e sobre a prática de sentar junto, que arremete mais para uma reunião técnica, Hamed et al. [65], afirma que:

> "A equipe de trabalho deve sentar-se em um lugar aberto e dentro de um contato entre eles. Além disso, o espaço de trabalho informativo exige que o sistema defina gráficos em torno do espaço de trabalho que ilustram o progresso do projeto e compartilha as informações entre eles."

A prática de sentar junto fortalece o time, uma vez que todos os participantes podem falar sobre o que está realizando no projeto, expondo suas ideias e realizando a comunicação de forma mais humana. Sobre isto, P32 relatou que "...nós realizamos uma reunião sentada todos os dias", evidenciando que a prática acontece diariamente.

**(c) Cliente Junto**: A prática de Cliente Junto figurou em segundo lugar na pesquisa de adoção com 55,56% de ocorrência. Um indício que pode apoiar essa evidência, é visto em Atkinson [9] onde o autor afirma que:

> "Ciclos de desenvolvimento promovem e facilitam a velocidade de implementação e o *feedback* regular leva a uma melhoria contínua em termos de aprendizagem e compreensão, e o cliente tem a oportunidade de priorizar os recursos que agregam maior valor a intervalos regulares."

A satisfação total do cliente está intimamente ligada com essa prática e pode haver uma pressão financeira por parte da diretoria para agradar o cliente, fazendo com que ele se sinta satisfeito com o produto que recebe. Mas não é só isto, pois que o cliente junto aos desenvolvedores favorece que o software seja confeccionado de forma a corresponder às suas expectativas, seus anseios. Apoiando esta evidência, Mann e Maurer [94] conduziram um estudo de caso onde, em entrevistas aplicadas à clientes de uma empresa, relataram o seguinte:

> "[...] Vários clientes mencionaram que eles estavam muito mais envolvidos no processo do que antes, '...no início do Scrum o processo exigiu que estivéssemos mais envolvidos na revisão diária e nas discussões. Isso nos levou a ser mais conscientes das nossas responsabilizados no processo para quaisquer mudanças e preocupações que pudessem ser consideradas.' "

Nesta pesquisa, P30 relatou que "As equipes são deslocadas para junto do cliente — na empresa do cliente, desenvolvendo junto com o cliente." e P32 "...com clientes maiores, conseguimos que a nossa empresa colocasse um funcionário à disposição.", são relatos que fornecem alguns indícios que apoiam a evidência da adoção desta prática, fazendo com que ela atingisse o patamar de segundo lugar, portanto entre o rol das mais preferidas.

**(d) Prototipação**: A prática Prototipação atingiu o primeiro lugar na pesquisa com 58,33% de adoção, portanto não muito distante do segundo e terceiro lugares. Tão próximas que, a priori, podem ser consideradas no mesmo patamar. O que faz com que essa prática atinja em cheio a preferência pelos desenvolvedores é a simples forma com que as ideias possam sair do papel e passam a ganhar vida, modelando o que apenas se pensava sobre uma regra de negócio ou uma dada funcionalidade. Sobre isto, P23 relatou que "... a prática de prototipação é fundamental para nós", e para P28, a prototipação "...deve ser feita da forma mais simples possível". Isto porque ela é uma das formas mais eficientes de transpor as ideias em um protótipo que possa ser apresentado ao cliente, e assim receber o seu *feedback*, ao propor uma solução adequada ao seu problema, aumentando a sua real percepção de valor.

Mostrando a importância da prototipação para a Engenharia de Software, Tomayko [153] afirma que:

> "Desenvolvedores que usam modelos de ciclo de vida baseados em protótipos estão familiarizados com o cliente que se apaixona pelo protótipo, fazendo com que se evite que a documentação insuficiente possa caracterizar o produto. [...] Os métodos ágeis propiciam rápido desenvolvimento de um protótipo que responde a uma pergunta simples sobre requisitos de conteúdo, desta forma, o cliente é convidado a dividir a responsabilidade da obtenção dos requisitos corretos."

Outro indício de apoio a evidência é a realização de reuniões de testes de aceitação, a mostra do protótipo, onde a equipe (e a indústria como um todo) tem a chance de apresentar um trabalho que se aproxime da realidade, uma forma interessante de contaminar o cliente e ganhar a sua aprovação. Sobre isto, veja o que dizem Beaudouin et. al [14]:

> "Eles incentivam a comunicação, ajudando designers, engenheiros, gerentes, desenvolvedores de software, clientes, e usuários para discutir opções e interagir

> uns com os outros. Eles também permitem uma avaliação antecipada porque podem ser testados de várias formas, incluindo estudos de usabilidade tradicionais e comentários dos usuários, ao longo do processo de design."

Nos dias atuais, nada é de se estranhar que a prototipação lidere esse resultado, pois que esta prática está muito em voga [48]. A técnica está compreendida dentro de um modelo denominado *Minimum Viable Product* (MVP), ou Produto Minimamente Viável, que consta de um conjunto de procedimentos para a obtenção de um protótipo funcional — técnica amplamente usada e difundida [150] por empresas como Facebook, Apple e Dropbox.

Sobre essa técnica que vem despontando na frente, Nguyen-Duc et. al [105] afirma que:

> "É essencial para as *startups* experimentarem rapidamente ideias empresariais construindo protótipos tangíveis e coletando *feedback* dos usuários sobre eles. Como a prototipagem é uma parte inevitável da aprendizagem de software em estágio inicial, o quão rápido elas podem aprender depende de quão rápido elas podem prototipar. Apesar da importância, há uma falta de pesquisa sobre prototipagem em *startups* de software."

## 4.4 Níveis de Percepção das Práticas Ágeis

Enquanto a seção 4.3 analisava os níveis de adoção das práticas ágeis, esta Seção analisa os níveis de percepção delas, considerando que elas já estivessem adotadas e integradas na empresa. Desta vez, foi colocado à disposição dos participantes uma relação de 40 princípios e práticas ágeis, extraídos de diversas metodologias (XP, Scrum, Lean, DSDM, etc.), sem se preocupar em definir o que era princípio ou o que era prática, o que muitas vezes se misturavam. Os participantes deveriam mensurar o nível de percepção de cada prática adotada e ainda tecer um breve comentário sobre ela. Durante a realização do *Pilot Survey*, nas três primeiras entrevistas, a relação constava de 60 práticas e as entrevistas se tornaram morosas e cansativas. Isto foi corrigido com a eliminação de 20 delas, processo descrito na Seção "Ameaças à Validade".

### 4.4.1 Tabulação

A pergunta número 12 do questionário, cujo teor era "Quais dessas Práticas Ágeis são ou foram aplicadas em sua equipe de desenvolvimento?", solicitava que o participante indicasse o nível de sua percepção quanto a importância da aceitação dessas práticas (as práticas foram colocadas mais abaixo para que o entrevistado as assinalasse), enquanto

participante ativo do projeto, conforme proposto em uma escala Likert [90] de 5 posições. A escala Likert foi utilizada, não só porque ela tem sido um modelo de referência entre os pesquisadores, quando o foco é mensurar atitudes comportamentais [76], mas porque ela desenvolve um conjunto de afirmações relacionadas à definição do fenômeno pesquisado, para as quais os participantes emitem o seu grau de concordância. No caso da presente pesquisa, deveriam escalonar o nível de percepção (comportamental) sobre a aceitação das práticas ágeis sendo executadas em seu ambiente de trabalho.

Os graus de dimensão Likert referente à pergunta número 12 do questionário podem ser visualizados na Tabela 4.6.

Tabela 4.6: Dimensões da aceitação das práticas ágeis adotadas nas empresas.

| **Dimensão** | **Descrição** |
|---|---|
| 1-Sem importância | Indica que a prática ágil não foi percebida pelo entrevistado. |
| 2-Pouco importante | Indica que a prática ágil foi pouco percebida e que a mesma não contribuiu muito para o sucesso do projeto e que, por vezes, foi ofuscada. |
| 3-Importante | Indica que a prática foi percebida pelo participante de forma positiva e disseminada, contribuindo razoavelmente para a melhoria do software. |
| 4-Muito importante | Indica que a prática foi bastante percebida, contribuindo muito para a melhoria dos processos, a diminuição de custos e a redução de defeitos. |
| 5-Extremamente importante | Indica que a prática foi extremamente percebida, contribuindo de forma preponderante para o sucesso do projeto. |

As respostas da questão número 12 foram tabuladas em uma planilha de Excel, cujos dados serviram de base para o script número 5 (Apêndice F) ao Programa R, conforme podemos ver na Figura 4.6.

### 4.4.2 Níveis de Percepção

O gráfico da Figura 4.6 apresenta os níveis de percepção das 40 práticas avaliadas, e assim relacionando-as de forma crescente de graus de percepção (aceitação), de baixo para cima, analisando o lado direito (cor *cyan*).

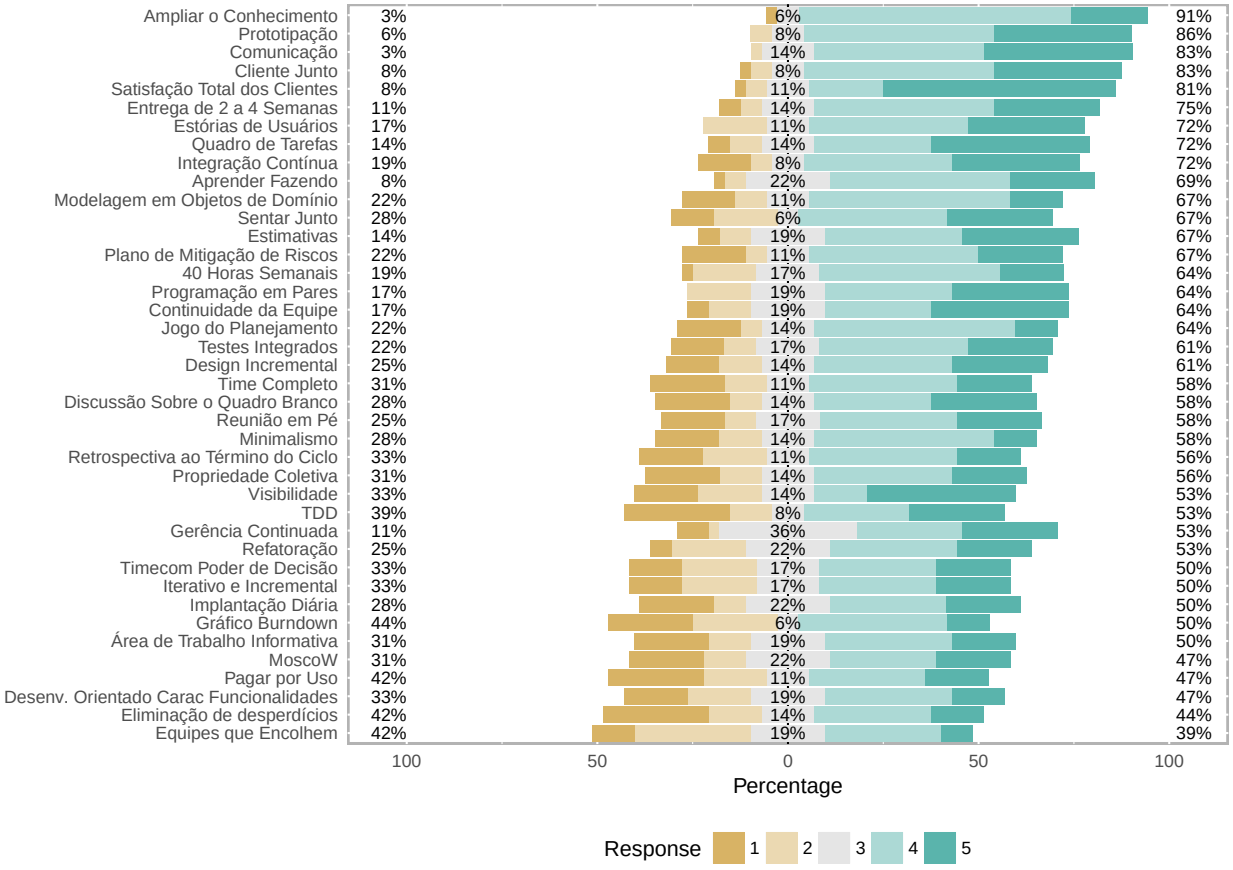



Figura 4.6: Graus de percepção de práticas ágeis nas empresas.

Um gráfico Likert separa esses graus em **graus inferiores**, neste caso, os graus 1 e 2, o **grau mediano**, que é o 3, e os **graus superiores** que são o 4 e o 5. As três partes são mensuradas em porcentagem. Quando queremos analisar o nível de aceitação, analisamos os graus superiores do lado direito, e quando queremos analisar o nível de rejeição, analisamos os graus inferiores, do lado esquerdo. A classificação do menor para o maior do lado esquerdo ocorre inversamente proporcional ao lado direito, ou seja, de cima para baixo. O nível mediano pode ser analisado no gráfico observando-se, neste caso, a cor cinza — aí não há um ordenamento claramente definido.

O gráfico mostra que a prática Ampliar o Conhecimento liderou a pesquisa com uma taxa de 91% de aceitação, contra 3% de rejeição, sendo que 6% dos participantes se declararam neutros. Sobre isto, P35 opinou que "...a minha empresa arca com todos os custos na realização de cursos para a obtenção de certificação". Isto é notório, uma vez que as empresas necessitam apresentar profissionais certificados em seus quadros no momento das contratações. Entretanto, conforme diz Wlodarkiewicz [163], o fato desta prática ser bem percebida pelo pessoal não reside apenas nisto, conforme relata:

> "A incerteza e a volatilidade do ambiente em que as empresas modernas precisam operar exigem que os conceitos e métodos de gerenciamento sejam continuamente monitorados e ajustados para que o alto nível de competitividade e atratividade do mercado possa ser mantido. É possível atingir esse estado aproveitando ao máximo os conhecimentos adquiridos. [...] A empresa que aspira o sucesso precisa moldar sua estrutura interna de uma forma que lhe permita alcançar o nível de uma empresa baseada em um conhecimento de alto nível de agilidade. Uma empresa assim é marcada por alta sensibilidade e habilidade em aproveitar as oportunidades que ela encontra, que decorrem da configuração interna de recursos que conduz a esse comportamento."

Um outro motivo é porque essa prática engloba várias outras práticas ágeis (TDD, Cliente Junto, etc). Desta forma é que essa percepção ocorre de maneira disseminada em todos os níveis operacionais, desde a diretoria eté o funcionário menos graduado.

A Prototipação é a segunda prática mais percebida, com 86% de aceitação, 6% de rejeição e 8% neutro. Convém ressaltar que na seção anterior essa prática liderava o ranking das mais adotadas pelas empresas. P20 referenciou-a como:

> "...em minha equipe, a arte de fazer protótipos se tornou, praticamente, trabalhos de elaboração de rascunhos. Sentamos com o cliente e esboçamos no papel as funcionalidades. Se ele achar que está legal, levamos o rascunho diretamente ao desenvolvedor para a codificação."

Entretanto, P36, que deu uma menção neutra, relatou que a prática de Prototipação "...é crucial, mas está sendo feita hoje de forma errada, na minha opinião. O cliente pede coisas adicionais que exigem a colocação de pontos de função, onde não deveriam."

Pela percepção do gráfico, o maior índice de neutralidade ficou com a prática Gerência Continuada, com 36% de ocorrência, entretanto, no geral, ela obteve uma taxa de aceitação maior (53%) que a de rejeição (11%). Nenhum dos participantes deram opiniões adicionais sobre ela.

As duas práticas que lideraram os índices menos citada, pelo fato de estarem sendo pouco percebidas nas empresas, foram Equipes que Encolhem e Eliminação de Desperdícios, cada uma com 42%. Mas é importante ressaltar que, embora possuam o maior índice de rejeição, suas taxas de aceitação também são consideradas boas. Entretanto, a prática Equipes que Encolhem possui uma taxa de rejeição maior que a de aceitação e foi isto que a fez ocupar o primeiro lugar das práticas mais rejeitadas. P23 referenciou a ela como "...percebi que tínhamos pessoas que não necessitavam estar ali. Comecei a fazer revisão em pares com analistas...". Esta prática [15] prega que a equipe vai diminuindo ao longo do projeto e que as pessoas que o deixam, passam a emprestar os seus conhecimentos em outros projetos, potencializando-os.

Por outro lado, mas não contrariando o resultado desta pesquisa, e também para elucidar porque ela é bem percebida com uma taxa de 39% de ocorrência, Dove e Schindel [43] relataram que há uma variação considerável nas equipes Scrum, por meio de contratações e designações, segundo eles:

> "O pessoal da equipe, trabalhou com os outros mestres em Scrum para coordenar e compartilhar recursos, com a utilização de desenvolvedores, [...] testadores e empreiteiros, que operavam em um ambiente sem barreiras, com a separação de funções baseada em status primário e secundário. [...] Essa abordagem melhorou a capacidade do programa para atrair e manter os melhores talentos, [...] para trazer um novo funcionário de forma rápida."

De uma forma geral, as 40 práticas submetidas à avaliação pelos participantes possuíram em seus diferentes níveis, mais aceitações do que rejeições, embora todas elas possuíssem as duas dimensões, em maior ou menor grau, indicando que há forte evidência das práticas de metodologias ágeis sendo empregadas nas empresas e que elas estão sendo bem aceitas, tanto por parte de funcionários, alto ou baixo escalão, quanto de seus clientes.

## 4.5 Percepção de Princípios Ágeis

A questão número 7 do questionário perguntou "Na sua equipe, é possível perceber algum princípio ágil sendo empregado, quais?". Como as práticas ágeis são regidas por alguns princípios fundamentais, constituindo a raiz ou a base dessas estruturas, trivial que a pesquisa também procurasse saber o quão profundo estaria o nível de entendimento dos participantes acerca desses princípios. A questão colocava 12 princípios à disposição dos participantes, que foram numerados de I1 a I12, para que eles escolhessem quais os que mais empregavam e que realizassem uma breve descrição sobre os escolhidos. Os resultados, mais uma vez, foram tabulados em uma planilha de Excel e em seguida levados ao Programa R, cuja plotagem é mostrada na Figura 4.7.

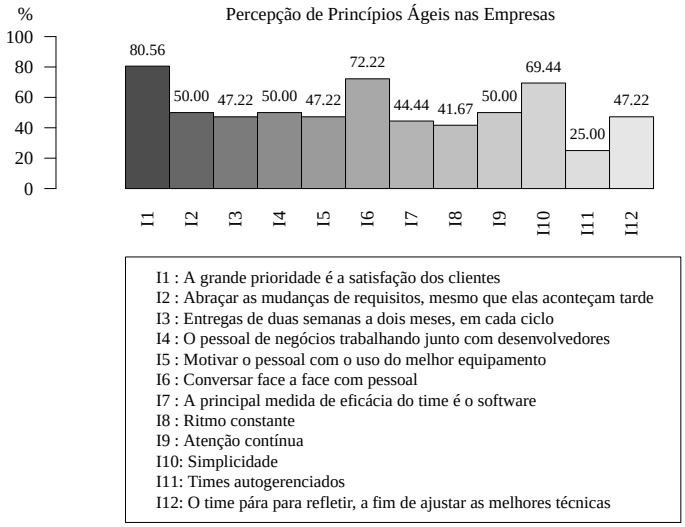



Figura 4.7: Percepção de Princípios Ágeis nas empresas.

Observando o gráfico, vemos imediatamente três princípios que são mais percebidos, conforme as respostas dos participantes: Em primeiro, aparece “A grande prioridade é a satisfação dos clientes”, em segundo, “Conversar face a face com o pessoal” e em terceiro “Simplicidade”, com percentuais de 80,5%, 72,2% e 69,4%, respectivamente. O princípio menos percebido foi “Times autogerenciados”. Sobre essas três, P23 declarou que “...temos levado a satisfação dos clientes muito a sério, pois isto faz a diferença. Temos diminuído bastante o escopo de entrega, visando exatamente isto aqui.”. P20 referiu-se às conversas face a face como “...bastante comunicação entre o pessoal, sem precisar de autorização do Scrum Master. Isto propicia resolver problemas de forma mais rápida.” e sobre esse mesmo princípio, P28 relatou que “sim, evitamos sempre usar e-mail — para ser mais rápido.”. Três participantes falaram sobre o princípio “Simplicidade”. P20 se referiu a ele como “transmissão de forma clara e objetiva, retirando as dúvidas.”, P28 “sempre existe uma conversa com a equipe para encontrar a solução mais rápida e simples.” e P32 “quanto mais simples mais facilita o entendimento do cliente.”

Em relação ao princípio “Times autogerenciados”, que foi o menos percebido, pode-se afirmar que o fenômeno depreende de alguma centralização de poder por parte da gerência, o que contraria os pressupostos da metodologia ágil, mas não totalmente, posto que o percentual de 25% de ocorrência ainda é um número que representa bem, um pouco afastado da média, mas ainda assim representativo.

No geral, pode-se dizer que em relação aos princípios que regem as metodologias ágeis investigadas na pesquisa, a média fica entre 40 e 50% de percepção na visão dos entrevistados, dentro das empresas às quais pertencem.

## 4.6 Produtividade

As perguntas de número 8 e 9 solicitavam aos participantes que eles informassem se houve ou não algum aumento de produtividade quando passaram a utilizar metodologias ágeis em seus projetos. Se positivo, deveriam indicar qual, ou quais, as práticas que mais agregaram valor. A Figura 4.8 mostra esse resultado:

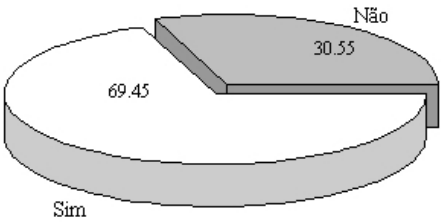



Figura 4.8: Percepção de produtividade com práticas ágeis.

Foi observado que 69,45% dos participantes perceberam que houve um aumento na produtividade da empresa quando passaram a utilizar as práticas ágeis em seus projetos, contra 30,55% que não perceberam. Quando respondido "Sim", o participante deveria indicar quais práticas que ele se lembrava terem agregado mais produtividade. P8, embora com pouca experiência em desenvolvimento, relatou que "...ajudou isto a conversa face-a-face com o pessoal", P10 relatou que "aumentou a produtividade porque houve conversas face-a-face com todas as pessoas do time". Outros participantes responderam de forma rápida, como P11 que relatou "Estar junto ao cliente", P13 "Entregas constantes" e P36 afirmando que "Prototipação é a prática que mais tem agregado valor aos nossos trabalhos", enquanto outros, ficaram alguns minutos relatando as suas experiências, como P32 que afirmou que "...é difícil precisar qual a prática ágil que melhor contribuiu. No geral, a metodologia ágil contribuiu de forma geral.", e P20, com a seguinte contribuição:

> "[...] O cliente se sentiu mais confiante e passou a financiar mais o projeto, quando experimentou o software funcionando, sentindo de perto, o que antes era apenas uma promessa de software."

...ou ainda P28:

> "A conversa face-a-face possibilitou melhorar o planejamento. Foi expressiva a melhoria, pois as metodologias tradicionais possuíam um tempo muito grande de planejamento. Agora, o meu planejamento é menor, e assim possibilitando o lançamento do produto de forma mais rápida."

O participante P36 relatou que:

"Quando utilizávamos a metodologia tradicional, conseguíamos faturar média de 100 a 150 Pontos de Função mensal por desenvolvedor. Quando mudamos para as metodologias ágeis, esse faturamento subiu para 300 a 350. Isto aumentou bastante a satisfação real do cliente e também a nossa, em nossa visão. A entrega foi mais consistente. Melhorou bastante, pois não ficamos mais 6 meses trabalhando em um projeto sem ver o software funcionando."

## 4.7 Problemas

As perguntas de número 10 e 11 serviram para extrair dos participantes se houveram ou não problemas durante a implantação de práticas ágeis e, se sim, quais foram as práticas que mais influíram para essa negatividade. A Figura 4.9 mostra esse resultado:

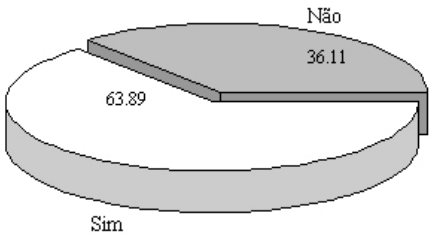



Figura 4.9: Percepção de problemas na adoção de práticas ágeis.

Observa-se que 63,89% dos participantes declararam que sim, houveram problemas, contra 36,11%. As respostas desta pergunta também foram multivariadas, sendo expostas nas Tabelas 4.7 a 4.8, sendo que alguns participantes não opinaram.

Tabela 4.7: Percepção de problemas na adoção de práticas ágeis por participante.

| **P#** | **Descrição** |
|---|---|
| P1 | "Problemas com falta de comunicação." |
| P2 | "Houveram problemas com quebra do paradigma, as pessoas começaram a não se importar com os clientes deixando eles em segundo plano." |
| P3 | "O problema foi com a falta de conhecimento das pessoas." |
| P4 | "A entrega de documentos não pôde ser ágil, pois houveram diversos problemas." |
| P5 | "Houveram problemas com a quebra do paradigma, pois as pessoas mais velhas custaram a absorver os conhecimentos, sendo assim, os resultados desejados não puderam ser alcançados plenamente, durante a mudança." |
| P6 | "Houveram problemas com a quebra de paradigmas." |
| P7 | "Quebra de paradigmas." |

Tabela 4.8: Percepção de problemas na adoção de práticas ágeis por participante — continuação.

| P# | Descrição |
|---|---|
| P9 | "Houve problemas com a quebra de paradigmas, as pessoas mais antigas demoraram para absorver os novos conhecimentos, pois resistiram muito em acreditar que os novos métodos fossem capazes de melhorar a produtividade." |
| P12 | "Aceitação e mudança nos processos de trabalho." |
| P13 | "Definição de ponto e inclusão dos testes na sprint." |
| P18 | "Quebrar o paradigma. Apesar de que abraçar as mudanças de requisitos tenha sido o grande diferencial para nós, ela também foi o principal obstáculo, pois foi difícil mudar a mentalidade do pessoal para a adoção dessas práticas ágeis." |
| P19 | "Na cabeça do programador, quanto mais ele desenvolve, maior tem de ser o seu salário. Se a produtividade é maior, ele pensa que está produzindo mais e pleiteia aumento de salário." |
| P20 | "O impacto é grande devido a mudança de mente ser muito grande. Sempre tem uma resistência. Em minha função, o que mais faço é tentar mudar a mentalidade da equipe, ajustando-a à metodologia ágil. No início eles não acreditam que as entregas de 2 a 4 semanas, vão funcionar. Acham que não vai ter documentação (mas na verdade possui sim uma documentação mínima). O controle de gerência é normal, mas dentro da metodologia ágil, e isto o pessoal no início sente dificuldade em entender e tende a impor barreiras por temor de não dar certo. Muitas vezes, o gerente do projeto acha que está utilizando metodologia ágil, mas na verdade está seguindo uma metodologia tradicional (RUP, por exemplo). Ele está fazendo ponto de controle todos os dias, pedindo relatórios todos os dias, e aí temos de intervir e informar que isto não funciona mais assim, para fazer ele enxergar as novas práticas que vai agregar valor. Essa foi para a nossa empresa a maior barreira." |
| P21 | "Em um cliente anterior tivemos problemas na entrega da release (que era de 2 semanas), pois tivemos a urgência em antecipar para 1 semana e isto não conseguimos fazer de imediato. Tivemos de aumentar a equipe para fazer face à demanda — uma reunião em uma manhã e depois dispor de 2 a 3 dias para confeccionar a release. Depois de um mês praticando assim, voltamos à situação inicial (2 semanas), pois não foi possível confeccionar uma release completa em apenas 1 semana." |
| P22 | "Mudanças sem controle que afetaram outras entregas. Apesar de se apregoar que as mudanças são bem vindas, começaram a realizar mudanças que impactaram em entregas que deixaram de ser entregues." |
| P23 | "Como tenho uma estimativa para a execução de uma determinada tarefa, e tenho também um escopo que pode ser aumentado ou diminuído, se o cliente está junto isto, esse escopo pode aumentar bastante. Acaba sendo difícil gerenciar essas estimativas quando o cliente interfere muito." |
| P24 | "Falta de clareza no momento de uma manutenção." |
| P25 | "A programação em pares não funcionou na nossa empresa, de forma adequada." |
| P28 | "A equipe necessita ter conhecimento profundo na metodologia adotada, para dar certo o projeto. Uma equipe de juniores não vai dar certo na metodologia ágil. É preciso colocar desenvolvedores maduros." |
| P30 | "O planejamento inicial foi ineficaz, causando retrabalhos." |
| P31 | "A quebra de paradigmas com os desenvolvedores mais antigos." |
| P32 | "O cliente junto não funcionou. Em um dos projetos, os clientes não eram muito comprometidos. Depois que tivemos essa percepção, em alguns projetos, elaboramos uma cartilha (agenda) com algumas responsabilidades do cliente, e aí ele passou a seguir a cartilha, o que propiciou uma maior integração. A Programação em pares, também não agregou muito em nossa experiência." |
| P34 | "O pessoal começou a confundir os conceitos e deixou de produzir a documentação mínima prevista." |
| P36 | "A falta de conhecimento da área técnica prejudicou um pouco o andamento. Duas semanas foi um tempo insuficiente para absorver a gama de conhecimentos para uma nova sprint." |

Percebe-se uma boa incidência sobre a influência da alta gerência em manter as práticas mais tradicionais, onde a "quebra de paradigmas" é mais relatada (P2, P5, P6, P7, P9, P12, P18, P20 e P31). Se o cliente junto é considerado uma das práticas mais aclamadas (veja seções anteriores), também ela pode ser um problema nos momentos iniciais da mudança para ágil, conforme relataram três participantes (P2, P10 e principalmente, P23). A falta de conhecimento técnico também foi frequente (P3, P28, P34 e P36). Outros relataram problemas de entregas (P4 e P21) decorrentes de má gestão.

É evidente que, se as práticas ágeis podem aumentar a produtividade (Figura 4.8), também a sua adoção, mesmo que em momentos iniciais, também pode trazer inúmeros problemas (Tabelas 4.7 a 4.8). Cabe a gerência a melhor gestão no sentido de aparar todas as arestas, providenciando a prática do melhor conhecimento a respeito do assunto, através do estudo sobre as metodologias e assim, diminuindo os problemas e os riscos.

## 4.8 Pontos Fortes e Fracos

A questão número 13 solicitava que o entrevistado apontasse pelo menos um ponto forte e um ponto fraco, durante a implantação e operação com práticas ágeis dentro de sua empresa. A Figura 4.10 mostra o resultado dessas duas questões, sendo colocados lado a lado para comparação.

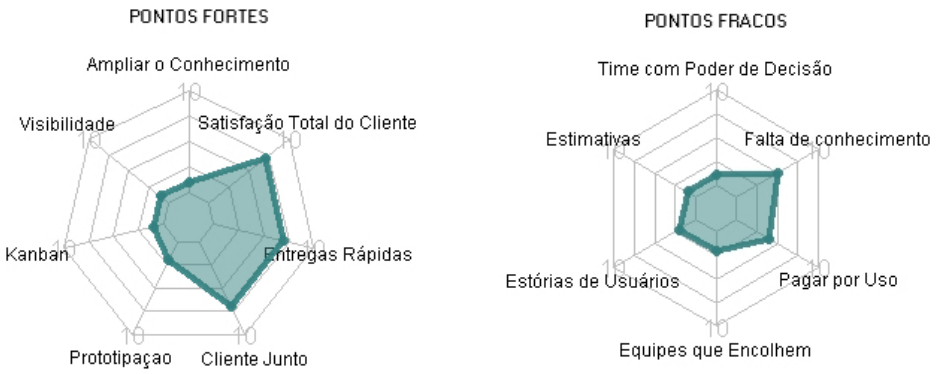



Figura 4.10: Percepção dos pontos fortes e fracos do emprego de práticas ágeis.

O gráfico de radar foi o mais indicado para uma rápida visualização. Na Figura, os dois são colocados lado a lado e cada um deles contendo uma escala de 0 a 10. Esta escala representa o número de ocorrência em cada posição, e não porcentagens. Imediatamente já é possível observar que há um maior número de ocorrências de pontos fortes que pontos fracos, apenas pela comparação do tamanho entre os dois polígonos, embora essa diferença seja sutil.

Sete pontos fortes foram destacados pelos participantes, sendo eles, Visibilidade, Kanban, Prototipação, Ampliar o Conhecimento, Cliente Junto, Entregas Rápidas e Satisfação Total do Cliente. Esses sete pontos foram os que mais apareceram na pesquisa. P21 referiu-se às entregas rápidas como "o cliente não precisa esperar 6 meses para ver o resultado", e P23 opinou sobre a satisfação dos clientes como "a satisfação do cliente faz a diferença. No momento em que comecei a utilizar métodos ágeis na empresa, as entregas foram feitas realmente, o que foi o diferencial.", P36 relatou que "...trabalhar junto com o cliente é o ponto mais forte para mim, pois a todo instante o cliente pode observar o que a equipe está fazendo e, também, acompanhar o processo como um todo."

Seis pontos fracos apareceram com mais evidência, sendo eles, Equipes que Encolhem, Estórias de Usuários, Estimativas, Time com Poder de Decisão, Pagar por Uso e Falta de Conhecimento. Este último, liderou o rol dos pontos fracos com 5 ocorrências. O participante P18 relatou que "...a diminuição da equipe ao longo do desenvolvimento (equipes que encolhem) — é bom para a empresa pois auxilia outros projetos, mas é ruim para o projeto." P22 afirmou que "...existe uma falta de pessoal habilitado e a demora no tempo em capacitar as pessoas. Não há preocupação com custos. Permissão de realizar mudanças sem controle. Exige times altamente treinados para poder gerenciar as metas." Referindo-se à falta de conhecimento do pessoal de sua equipe e também a Time com Poder de Decisão, P34 relatou que "as pessoas se confundem muito e acham que somente por usar Scrum as coisas vão fluir naturalmente, mas a influência do gerente é que pode fazer toda a diferença", P24 também falou sobre isto como "...dificuldade em compreender algumas metodologias. Falta de conhecimento das pessoas.". Em outra direção, P21 afirmou que "se não tiver um sistema que gerencia as estórias, fica muito difícil — desde o início da estória até os *bugs* gerados depois.". Sobre este aspecto, em recente trabalho de Kamei et al. [78], a prática de Estórias de Usuários também foi relatada como de difícil de ser trabalhada: "[...] O uso de pontos de estórias (Estórias de Usuários) não funciona bem porque elas são subjetivas. [...] Creio que estimar o esforço em horas seja mais preciso e é difícil estimar em pontos quando a tecnologia é desconhecida."

De modo geral, foram identificados pontos fortes e pontos fracos ao se utilizar práticas ágeis de desenvolvimento de software dentro das empresas. As ocorrências registradas nos dois gráficos da Figura 4.10 foram contadas conforme as respostas da pergunta número 13 do questionário da pesquisa de campo, entretanto, várias outras ocorrências foram relatadas, mas em número inferior que não mereceram destaque. Alguns participantes citaram "...não visualizei um ponto forte, ou um fraco", outros simplesmente nem opinaram. Resultados assim não influenciaram na contagem do resultado, uma vez que eles não foram mensurados em graus de porcentagem.

## 4.9 Principais Desafios

Para encerrar a entrevista, a última pergunta do questionário foi "Você poderia nos dizer quais foram seus principais desafios, enquanto trabalha/trabalhava com práticas ágeis?". Dos 36 participantes, 6 não quiseram opinar (16,7%) e o restante responderam, quase sempre, um mesmo padrão. Essas respostas estão materializadas na Tabela 4.9.

Pela análise das respostas, ficou evidente que um dos grandes desafios de uma empresa em implantar metodologia ágil é a "Quebra de Paradigmas". Isto já era um conhecimento antigo, como explanado em [15], [38], [34], [78] e tantos outros trabalhos, mas não da forma como apresentada nesta pesquisa, onde os dados são mensurados de forma quantitativa e com um alvo bem específico. Desta forma, o presente trabalho de exploração de campo vem, mais uma vez, reforçar esta evidência pela análise e exposição dos dados, conforme as transcrições de áudio evidenciadas na Tabela 4.9.

O gráfico da Figura 4.11 apresenta uma sumarização da contagem dos desafios, desde que os mesmos apresentassem pelo menos duas ocorrências.

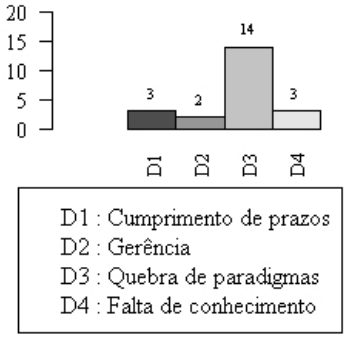



Figura 4.11: Maiores desafios na implantação de metodologias ágeis..

Para obter os números de ocorrências, as respostas foram agrupadas por similaridade, como por exemplo: onde aparecia a palavra "paradigma" ou algumas expressões do tipo "foi difícil convencer as pessoas", ou ainda "motivar quanto a mudança" e "resistência à mudança", atribuiu-se a elas o desafio de quebra de paradigmas. As respostas que apresentaram apenas uma ocorrência, como por exemplo "foi difícil a implantação", ou ainda expressões como "transportar as estórias para a prototipação", foram descartadas na avaliação, por serem isoladas.

O resultado final apresenta os principais desafios na implantação de metodologias ágeis em: primeiro, a quebra de paradigmas, segundo, o cumprimento de prazos e a falta de conhecimento e, terceiro, a gerência.

Tabela 4.9: Maiores desafios na implantação de metodologias ágeis.

| P# | Descrição |
|---|---|
| P2 | “Problemas com a gerência.” |
| P4 | “Foi difícil convencer as pessoas a entender e nem todos estavam preparados.” |
| P5 | “Quebra de paradigmas.” |
| P6 | “Foi difícil a implantação.” |
| P7 | “Problemas com a gerência, não se sabia como repassar as ideias para os participantes.” |
| P8 | “Quebra de paradigmas.” |
| P9 | “Dificuldade em aderir os novos conceitos.” |
| P10 | “Quebra do paradigma, pois foi muito difícil convencer as pessoas a utilizar a metodologia.” |
| P11 | “Motivar a equipe quanto a mudança em instituições públicas.” |
| P12 | “Resistência na mudança e controlar o trabalho a ser realizado.” |
| P13 | “Implantar testes automatizados e colocar as releases em produção.” |
| P15 | “Resistência na mudanças.” |
| P18 | “Fazer com que o cliente acredite na venda do software como ferramenta de melhoria para a sua instituição.” |
| P19 | “A falta de conhecimento técnico foi o maior desafio quando nos deparamos com um tipo de conhecimento ainda não praticado.” |
| P20 | “Se não for executado de forma correta e coerente, o ágil não vai funcionar. Exemplo: em uma reunião de planejamento, se eu não possuir um planejamento eficiente, posso cair em descrédito.” |
| P21 | “Quebra de paradigmas. Na minha empresa [...], nosso maior desafio foi sincronizar a equipe. Alguns são resistentes em aceitar novas metodologias. Várias pessoas não se adaptaram, pois estavam há muito tempo na metodologia antiga.” |
| P23 | “Entregar o software com qualidade — sempre teremos uma entrega com métodos ágeis, mas o que diferencia é a qualidade.” |
| P24 | “O principal desafio é o cumprimento de demandas e tarefas em um curto prazo.” |
| P25 | “Foi a quebra do paradigma. Fazer com que as pessoas que trabalham há muito tempo com isto aceitassem as metodologias ágeis como uma forma de trabalho. esse foi nosso principal desafio, mas, com muito custo e trabalho árduo, conseguimos mostrar ao pessoal mais antigo as vantagens desses novos métodos.” |
| P26 | “Entender como funciona a metodologia, sem dúvida foi a mudança da forma de agir — quebrar o paradigma.” |
| P27 | “Cumprir prazos está sendo nosso maior desafio.” |
| P28 | “Cultura das pessoas — quebra do paradigma. Encarar o desafio com a gerência, meio desconfiada.” |
| P29 | “Aceitação da equipe — Quebrar o paradigma. Muita gente acha que não vai dar certo, porque acham que tem de possuir mais documentação, etc.” |
| P30 | “O desenvolvimento de forma incremental, adaptando as novas funcionalidades, foi um desafio.” |
| P31 | “A quebra do paradigma.” |
| P32 | “Dificuldade em quebrar o paradigma.” |
| P34 | “A implantação inicial foi o maior desafio, pois faltou maturidade da equipe, faltou conhecimento para abraçar a causa com maior ênfase.” |
| P35 | “Transportar as estórias para a prototipação.” |
| P36 | “A comunicação para mim foi o meu maior desafio.” |

## 4.10 Ameaças à Validade

Esta análise pode ser feita de acordo com as metodologias empregadas e de algumas características peculiares, como o grau de consistência e o rigor utilizado nas pesquisas de campo. Apoia-se nisto, o fato de que o trabalho da aplicação do questionário, que serviu de prospecção dos dados, foi feito segundo as regras da metodologia científica adaptadas à Engenharia de Software.

### 4.10.1 Validação

Pela utilização do projeto-piloto de *survey*, as perguntas puderam ser validadas. Assim, descobriu-se que as entrevistas, durante o piloto (as três primeiras entrevistas), estavam bem cansativas, forçando a intervenção do entrevistador nos momentos em que as respostas dos participantes começavam a divagar. Com os ajustes, as entrevistas fluíram melhor. Os ajustes constaram da eliminação de 20 práticas ou princípios ágeis, julgadas dispensáveis, isto apenas na pergunta número 12, que fazia com que os entrevistados apontassem o grau de adoção de cada prática e ainda tecessem um breve comentário sobre elas. Influiu nesse julgamento o fato de que essas 20 práticas eliminadas foram consideradas pouco expressivas ou ainda que elas se repetiam entre uma metodologia e outra, como por exemplo, Implantação Incremental com Integração Contínua, Redução de Custo com Eliminação de Desperdícios, Planning Poker com Jogo do Planejamento, Uso de Templates com Padrões de Codificação, Código Compartilhado com Repositório Único de Código, entre outras.

### 4.10.2 Qualidade da Amostra

A variabilidade de cargos evidenciada na pesquisa foi providencial para o estudo, pois permitiu que várias facetas do assunto pudessem ser esclarecidas por meio da troca de palavras entre o entrevistador e o entrevistado.

(a) PONTOS DE VISTA

Dessa forma, foi possível observar os participantes relatando o seu ponto de vista sobre determinado assunto. Por exemplo, o participante P30, um analista em sua equipe, descreveu a prática Integração Contínua como:

> "Na equipe onde participo como analista de sistemas, existe um líder que coordena dois desenvolvedores que são os responsáveis por integrar todas as funcionalidades

> que vão sendo implementadas. Mas existem muitas conversas com o cliente para que esta integração possa ocorrer de fato."

...já o participante P29, um desenvolvedor, e sobre a mesma pergunta, limitou-se apenas a responder que "...a integração contínua é importante para o desenvolvimento", sem tecer maiores comentários. É possível que o desenvolvedor não consiga enxergar a dimensão total e abrangente desse tipo de prática ágil, mas o analista e o gerente a percebe sob outro ângulo. Por isto foi importante o entendimento dessas dimensões para a pesquisa e assim possibilitando a tabulação e agrupamento dos dados com maior exatidão, uma vez analisados cada áudio com cuidado.

(b) PROFISSIONAIS QUALIFICADOS

O nível de escolaridade dos participantes, especialmente dos desenvolvedores e dos gerentes de projetos, também foi considerado importante para se diminuir o grau das ameaças e incertezas da pesquisa, uma vez que lidou-se com profissionais qualificados, bem como o tempo de experiência dos mesmos.

(c) SIGILO E ÉTICA

Outro fato importante para a redução das ameaças da pesquisa, relacionado aos participantes, foi a questão do "segredo profissional". No início de cada entrevista, ao participante que se dispôs a falar, foi informado sobre o sigilo dos dados e que nem o seu nome, nem o da sua empresa, seriam revelados, característica que estabeleceria a ética e a transparência entre o entrevistador e os entrevistados.

# Capítulo 5

# Conclusão

Produzir software com qualidade, por vezes, torna-se atividade bastante trabalhosa, onde as equipes se expressam com liberdade e criatividade para a consecução de seus objetivos. Nesse contexto, o presente trabalho colocou algumas questões sobre práticas ágeis, traçando um panorama desde o seu surgimento até os dias mais atuais. A metodologia usada nas pesquisas propiciou a realização de algumas definições sobre o ciclo de vida dos projetos, principalmente nas áreas do conhecimento da Engenharia de Software de que tratam as metodologias ágeis.

O Capítulo 2 apresentou um estudo terciário sobre as práticas ágeis, tomando-se como base um artigo escrito por Dybå e Dingsøyr [44] e outro escrito por Dikert et al [40], este último, um trabalho realizado em 2016 sobre desenvolvimento de software em larga escala. O Capítulo 3 descreveu as metodologias utilizadas para a realização das pesquisas de campo empreendidas neste trabalho e o Capítulo 4 a formatação, tabulação e apresentação dos resultados utilizando-se abordagens sob as formas quantitativa e qualitativa.

## 5.1 Sobre o Estudo Terciário

Foi realizado um estudo terciário sobre as metodologias e suas práticas ágeis cuja finalidade foi a de embasar este trabalho com a fundamentação teórica necessária. Verificou-se as práticas ágeis mais adotadas no mundo hoje e como elas aparecem nos estudos empíricos da comunidade científica.

Visto que o mercado reclama cada vez mais de dificuldades típicas enfrentadas por uma equipe de desenvolvimento de software, as metodologias ágeis chegaram para tentar resolver diversos desses problemas através de práticas que favorecem o produto final e os problemas que os clientes desejam resolver. Entretanto, embora as metodologias ágeis tenham chegado com a proposta de resolver esses problemas, vários pesquisadores

engrossaram outras filas formando uma frente antagônica, com discursos acalorados contra a filosofia ágil. O fato é que, com discursos acalorados ou não, as metodologias ágeis ganharam espaço nas fábricas de software causando impacto na comunidade científica.

### 5.1.1 Verificando a Primeira Questão de Pesquisa

O estudo terciário visou a responder a primeira questão de pesquisa:

*(Q1) Que práticas ágeis estão sendo mais exploradas na literatura?*

Sobre este aspecto foi possível relacionar algumas práticas ágeis que as empresas mais utilizam, pela evidência da pesquisa empírica colocada à mostra pelo estudo terciário. Alguns mapas mentais (ver Apêndice C) foram utilizados para compreender a situação dessas práticas, que foram agrupadas em uma tabela por similaridades.

### 5.1.2 Embasamento Teórico para a Pesquisa de Campo

O estudo terciário forneceu diversas pistas para a elaboração de um questionário futuro, que seria utilizado em pesquisa de campo. Assim, por meio dele, foi possível identificar as evidências e também as controvérsias em torno do tema, percepções sobre satisfação de funcionários e clientes e limitações. As metodologias mais notórias foram Scrum e XP, mostrando que a adaptação para esses processos pode trazer vários benefícios, mas também vários incômodos junto às partes interessadas, pois existem antagonismos.

## 5.2 Sobre a Pesquisa de Campo

A pesquisa de campo, que consistiu de entrevistas realizadas por e-mail, face-a-face e telefone, que contou com 36 participantes, teve por objetivo comprovar pela prática as evidências apresentadas no estudo terciário e a responder as duas últimas questões de pesquisa:

- *(Q2) Qual a percepção das empresas sobre a adoção de práticas ágeis?*
- *(Q3) Qual a extensão da adoção de práticas ágeis nas empresas?*

### 5.2.1 Verificando a Segunda e a Terceira Questões de Pesquisa

Para responder a essas duas questões, algumas perguntas do questionário tiveram como alvo a descoberta da adoção e também da percepção de forma gradual dessas práticas.

Além disso, o estudo também mostrou uma comparação da evolução de métodos ágeis no tempo, referentes aos anos de 2008, 2012 e 2017.

(a) SEGUNDA QUESTÃO (Q2)

Os dados referentes às práticas ágeis foram tabulados e apresentados sinteticamente, por meio de gráficos (abordagem quantitativa), analisando o nível de adoção das práticas. Realizou também um estudo qualitativo das respostas dos participantes, por meio de análise das transcrições de áudio, aqui também analisando as extensões e as circunstancias (Q3) com que as evidências foram apresentadas. Desta forma, a questão (Q2) foi satisfeita, na medida em que pôde-se constatar os fenômenos-alvo por meio dos números apresentados.

(b) TERCEIRA QUESTÃO (Q3)

Uma escala Likert foi utilizada para demonstrar o grau de aceitação, portanto evidenciando a extensão com que essas práticas são adotadas nas empresas. Entretanto, a terceira questão de pesquisa também foi verificada com a análise qualitativa de todas as outras perguntas do questionário, avaliando não só as extensões, mas também as circunstancias com que as práticas ágeis foram introduzidas nas empresas.

## 5.3 Estratégias

Se os métodos ágeis são uma alternativa à gestão tradicional de projetos, eles hoje podem ser aplicados em qualquer tipo de projeto, inclusive os que não se arremetem ao software propriamente dito. Os métodos ágeis vem ajudando muitas equipes a encarar as imprevisibilidades através de entregas incrementais e ciclos iterativos, fazendo com que eles passassem a ser mais uma opção junto aos métodos tradicionais.

Outrossim, importa-se que a adoção ou não de metodologia ágil irá depender bastante do tipo de projeto e do contexto onde ele está inserido, porque existem diferenças de um projeto para outro, de uma organização para outra. Alguns afirmam que o ágil, por si só, não vai resolver os problemas, mas sim, evidenciar suas fraquezas para que se possa traçar um panorama e identificar prováveis causas de insucesso. Aí percebe-se uma estratégia, onde se poderia apontar com mais propriedade as brechas e as lacunas para a entrada das práticas ágeis. Cabe então a equipe atuar de forma pró-ativa e desejar as mudanças para que os benefícios da metodologia ágil possam trazer as respostas certas.

## 5.4 Trabalhos Futuros

Este trabalho não encerra a questão das práticas ágeis, pois ainda existem lacunas a serem exploradas, mesmo após quase 20 anos de debates. Isto é evidente hoje, com as publicações recentes, trabalhos atuais de pesquisadores sendo publicados a todo momento [43], [105] e [73], apenas para citar alguns, e todos publicados no ano de 2017. Em relação a este trabalho, os dados encontram-se em poder do pesquisador e estão disponíveis para realizar outros tipos de prospecções.

Sobre o futuro das práticas ágeis, basta simplesmente olhar para o seu próprio conceito e entender o que está por vir. Por que será que empresas de sucesso as utilizam? Toyota e Google estão tirando o máximo proveito da adoção desses princípios, pois eles oferecem a habilidade de projetar a mentalidade em várias áreas, não só a do software.

Não há como negar essa evidência, pois o mundo passa por diversas transformações e é preciso adaptar os novos conceitos à nova filosofia de vida que ora se desponta. Quem estiver mais bem preparado vai sair-se na frente e o conhecimento das metodologias ágeis é mais uma ferramenta do desenvolvedor de software, capaz de lhe mostrar um caminho que deve ser seguido nas frentes a que se depara. No mínimo, é preciso "abrir mentes", pois os estudos estão aí, todos embasados em experiências realizadas conforme as metodologias empíricas, fazendo com que se possa refletir com mais sobriedade.

# Referências


[1] J Ågerfalk, Brian Fitzgerald, e Old Petunias In. Flexible and distributed software processes: old petunias in new bowls. In *Communications of the ACM*. Citeseer, 2006. 87

[2] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002. 18

[3] Scott W Ambler e Mark Lines. *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press, 2012. 2

[4] Ann Anderson, Ralph Beattie, Kent Beck, David Bryant, Marie DeArment, Martin Fowler, Margaret Fronczak, Rich Garzaniti, Dennis Gore, Brian Hacker, et al. Chrysler goes to"extremes. *Distributed Computing*, 1(10):25–28, 1998. 87

[5] David J Anderson. *Agile management for software engineering: Applying the theory of constraints for business results*. Prentice Hall Professional, 2003. 95

[6] David J Anderson. *Kanban: successful evolutionary change for your technology business*. Blue Hole Press, 2010. 15, 101, 102, 104

[7] Cynthia Andres e Kent Beck. Extreme programming explained: Embrace change. *Reading: Addison-Wesley Professional*, 2004. 109

[8] Mikio Aoyama. Web-based agile software development. *IEEE software*, 15(6):56–65, 1998. 87

[9] Susan Atkinson. Why the traditional contract for software development is flawed. *Computer and Telecommunications Law Review*, 16(7):179, 2010. 48

[10] Hajer Ayed, Naji Habra, e Benoît Vanderose. Am-quick: a measurement-based framework for agile methods customisation. In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*, pages 71–80. IEEE, 2013. 2

[11] Muhammad Ali Babar e He Zhang. Systematic literature reviews in software engineering: Preliminary results from interviews with researchers. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, pages 346–355. IEEE, 2009. 8

[12] Aidil Jesus da Silveira Barros e Neide Aparecida de Souza Lehfeld. Fundamentos de metodologia científica. *São Paulo*, 2, 2007. 24, 25, 26

[13] Hubert Baumeister e Martin Wirsing. Applying test-first programming and iterative development in building an e-business application. In *International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet, SSGRR 2002, L'Aquila, Italy*, 2002. 136

[14] Michel Beaudouin-Lafon e Wendy Mackay. Prototyping tools and techniques. *Human Computer Interaction-Development Process*, pages 122–142, 2003. 49

[15] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999. 41, 53, 61, 105, 106, 108, 121, 132, 133

[16] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000. 2, 18, 19, 87, 88, 130

[17] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001. 46, 83, 85

[18] Barry Boehm. Get ready for agile methods, with care. *Computer*, 35(1):64–69, 2002. 87

[19] Barry Boehm. A survey of agile development methodologies. *Laurie Williams*, 2007. 45, 119

[20] Adeyemi Bolaji. *A cross-disciplinary systematic literature review on Kanban*. Tese (Doutorado), Tesis de maestría Obtenida en herkules. oulu. fi/thesis/nbnfioulu-201502111073. pdf, 2015. 101

[21] Keith Braithwaite e Tim Joyce. Xp expanded: Distributed extreme programming. In *International conference on Extreme Programming and Agile Processes in Software Engineering*, pages 180–188. Springer, 2005. 47

[22] Exército Brasileiro. Sistema de gestão de desempenho - sgd. `http://daprom.dgp.eb.mil.br/index.php/2017-01-25-11-35-04`, 2014. [Online; accessed 13-april-2017]. 134

[23] M. Bravo. Programação extrema. `http://www.agilcoop.org.br/poster`, 2016. [Online; accessed 20-August-2016]. xiii, 111

[24] Josiane Brietzke e Abraham Rabelo. Resistance factors in software processes improvement. *CLEI Electronic Journal*, 9, 2006. 1

[25] P. Brodzinski. Lean, agile and kanban. `http://www.brodzinski.com`, 2016. [Online; accessed 11-August-2016]. xiii, 103, 104

[26] John Millar Carroll e Society for Technical Communication. *Minimalism beyond the Nurnberg funnel*. MIT Press, 1998. 122

[27] Amado Cervo e Pedro A Bervian. *Metodologia científica*. México, 1990. 26, 27

[28] Alistair Cockburn. *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004. xiii, 88, 93, 94

[29] Alistair Cockburn e Laurie Williams. The costs and benefits of pair programming. *Extreme programming examined*, pages 223–247, 2000. 132, 133

[30] David Cohen, Mikael Lindvall, e Patricia Costa. An introduction to agile methods. *Advances in computers*, 62:1–66, 2004. 85, 86

[31] Kieran Conboy, Sharon Coyle, Xiaofeng Wang, e Minna Pikkarainen. People over process: key people challenges in agile development. *Ieee Software*, 99(1):47–57, 2010. 127

[32] Larry L Constantine. *Constantine on peopleware*. Prentice Hall, 1995. 133

[33] Philip B Crosby. *Quality without tears*. New American Library, 1985. 142

[34] James A Crowder e Valerie Bernard. New evm constructs to support agile development projects: Human performance and evms. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, page 78. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016. 61

[35] FABIO CRUZ. *Scrum e PMBOK unidos no Gerenciamento de Projetos*. Brasport, 2013. 86, 123

[36] Monica Rottmann de Biazzi, Antonio Rafael Namur Muscat, e Jorge Luiz de Biazzi. Process management in the public sector: A brazilian case study. pages 2898–2904, 2009. 1

[37] Jeff De Luca, Peter Coad, e E Lefebyre. Java modeling in color with uml: Enterprise components and process, 1999. 95

[38] William Chaves de Souza Carvalho, Pedro Frosi Rosa, Michel dos Santos Soares, Marco Antonio Teixeira da Cunha Jr, e Luiz Carlos Buiatte. A comparative analysis of the agile and traditional software development processes productivity. In *Computer Science Society (SCCC), 2011 30th International Conference of the Chilean*, pages 74–82. IEEE, 2011. 61

[39] Esther Derby, Diana Larsen, e Ken Schwaber. Agile retrospectives: Making good teams great. 2006. 137

[40] Kim Dikert, Maria Paasivaara, e Casper Lassenius. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 2016. 2, 5, 6, 7, 9, 10, 11, 13, 15, 16, 17, 22, 40, 65

[41] Torgeir Dingsøyr e Nils Brede Moe. Towards principles of large-scale agile development. In *International Conference on Agile Software Development*, pages 1–8. Springer, 2014. 13, 14

[42] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, e Nils Brede Moe. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6):1213–1221, 2012. 16

[43] Rick Dove, W Schindel, e M Kenney. Case study: agile se process for centralized sos sustainment at northrop grumman. In *Proceedings International Symposium. International Council on Systems Engineering. Adelaide, Australia*, pages 17–20, 2017. 54, 68

[44] Tore Dybå e Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9):833–859, 2008. xii, xv, 1, 5, 6, 9, 11, 12, 13, 14, 22, 40, 65, 88

[45] Tore Dyba e Torgeir Dingsoyr. What do we know about agile software development? *IEEE software*, 26(5):6–9, 2009. 7, 13, 14

[46] Tore Dybå, Torgeir Dingsøyr, e Geir Kjetil Hanssen. Applying systematic reviews to diverse study types: An experience report. In *ESEM*, volume 7, pages 225–234, 2007. 8

[47] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, e Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008. 8

[48] Endeavor. Po guia prático para o seu mvp – minimum viable product. `https://endeavor.org.br/mvp/`, 2017. [Online; accessed 08-jun-2017]. 50

[49] Nicole C Engard. Limesurvey http://limesurvey. org: Visited: Summer 2009. 2009. 27

[50] John Erickson, Kalle Lyytinen, e Keng Siau. Agile modeling, agile software development, and extreme programming: the state of research. *Journal of database Management*, 16(4):88, 2005. 87

[51] Nicholas D Evans. *Business innovation and disruptive technology: Harnessing the power of breakthrough technology... for competitive advantage*. FT Press, 2003. xiii, 118

[52] Aline Cristine Fadel e Henrique da Mota Silveira. Metodologias ágeis no contexto de desenvolvimento de software: Xp, scrum e lean. *Monografia do Curso de Mestrado FT-027-Gestão de Projetos e Qualidade da Faculdade de Tecnologia–UNICAMP*, 2010. 98, 101

[53] M. Fowler. Principlesofxp. `http://http://martinfowler.com/bliki/PrinciplesOfXP.html`, 2003. [Online; accessed 13-August-2016]. 47, 107

[54] M. Fowler. What is the difference between a usecase and xp's userstory? `https://www.martinfowler.com/bliki/UseCasesAndStories.html`, 2003. [Online; accessed 13-april-2017]. 19, 131

[55] M. Fowler. C3. http://www.martinfowler.com/bliki/C3.html, 2004. [Online; accessed 11-August-2016]. 105

[56] Martin Fowler e Kent Beck. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999. 135

[57] Martin Fowler e Matthew Foemmel. Continuous integration. *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, page 122, 2006. 120

[58] Martin Fowler e Jim Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001. 138, 139, 142

[59] Floyd J Fowler Jr. *Survey research methods*. Sage publications, 2013. 27

[60] Carlos Alberto Castilho Franco e Rodrigo de Toledo. Terceirização do desenvolvimento de software no brasil e nos eua. *Revista do TCU*, (128):20–29, 2013. 3

[61] Edward F Gehringer. Daily course evaluation with google forms. In *ASEE, American Society for Engineering Education Annual Conference & Exposition*, 2010. 27

[62] Tom Gilb. *Competitive engineering: a handbook for systems engineering, requirements engineering, and software engineering using Planguage*. Butterworth-Heinemann, 2005. 88

[63] Fred Grossman, Joe Bergin, David Leip, Susan Merritt, e Olly Gotel. One xp experience: introducing agile (xp) software development into a culture that is willing but not ready. In *Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, pages 242–254. IBM Press, 2004. 127

[64] Richard Ford' Guy Boy e Ronda Henning. Product maturity, security and software engineering. *Advances in Human Factors, Ergonomics, and Safety in Manufacturing and Service Industries*, page 331, 2010. 129

[65] Amani Mahdi Mohammed Hamed e Hisham Abushama. Popular agile approaches in software development: Review and analysis. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, pages 160–166. IEEE, 2013. 48

[66] Marc Hamilton. *Software development: building reliable systems*. Prentice Hall Professional, 1999. 99

[67] Jeanette Heidenberg, Mari Matinlassi, Minna Pikkarainen, Piia Hirkman, e Jari Partanen. Systematic piloting of agile methods in the large: two cases in embedded systems development. In *International Conference on Product Focused Software Process Improvement*, pages 47–61. Springer, 2010. 16

[68] C. A. M Hernandes. 1º seminário de metodologia Ágil do sisp - levantamento sobre métodos Ágeis. http://www.sisp.gov.br/guiaagil/wiki/download/file/Apresentacao_I_Seminario_SISP%20-TCU.pdf, 2014. [Online; accessed 30-October-2016]. 3

[69] James A Highsmith. *Agile software development ecosystems*, volume 13. Addison-Wesley Professional, 2002. xii, 90, 91, 94, 99, 101

[70] Jim Highsmith. Retiring lifecycle dinosaurs a look at adaptive software development, an alternative to traditional, process-centric software management methods. *Software testing and quality engineering*, 2:22–30, 2000. 90

[71] Jim Highsmith e Alistair Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001. 15

[72] Merisalo-Rantanen Hilkka, Tuunanen Tuure, e Matti Rossi. Is extreme programming just old wine in new bottles: A comparison of two cases. *Journal of Database Management*, 16(4):41, 2005. 13

[73] Rashina Hoda e James Noble. Becoming agile: a grounded theory of agile transitions in practice. In *Proceedings of the 39th International Conference on Software Engineering*, pages 141–151. IEEE Press, 2017. 68

[74] Trainer Peter Hundermark. An unofficial set of tips and insights into how to implement scrum well. 123

[75] CMMI Institute. A guide to scrum and cmmi: Improving agile performance with cmmi. `http://cmmiinstitute.com/cmmi-and-agile`, 2017. [Online; accessed 04-April-2017]. 87

[76] Severino Domingos da Silva JÚNIOR e Francisco Costa. Mensuração e escalas de verificação: uma análise comparativa das escalas de likert e phrase completion. *PMKT–Revista Brasileira de Pesquisas de Marketing, Opinião e Mídia*, 15:1–16, 2014. 51

[77] Matti Kaisti, Ville Rantala, Tapio Mujunen, Sami Hyrynsalmi, Kaisa Könnölä, Tuomas Mäkilä, e Teijo Lehtonen. Agile methods for embedded systems development-a literature review and a mapping study. *EURASIP Journal on Embedded Systems*, 2013(1):1–16, 2013. xii, 15, 88

[78] Fernando Kamei, Gustavo Pinto, Bruno Cartaxo, e Alexandre Vasconcelos. On the benefits/limitations of agile so ware development: An interview study with brazilian companies. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 154–159. ACM, 2017. 60, 61

[79] Sam Kaner. *Facilitator's guide to participatory decision-making*. John Wiley & Sons, 2014. xiii, 126, 127

[80] Staffs Keele. Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. sn, 2007. 8, 9, 12

[81] Mark Keil, Paul E Cule, Kalle Lyytinen, e Roy C Schmidt. A framework for identifying software project risks. *Communications of the ACM*, 41(11):76–83, 1998. 19, 21

[82] Tom Kendrick. *Identifying and managing project risk: essential tools for failure-proofing your project*. AMACOM Div American Mgmt Assn, 2015. 123

[83] M. Kern. Agile is dead. `https://www.linkedin.com/pulse/agile-dead-matthew-kern`, 2016. [Online; accessed 25-August-2016]. 13

[84] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, e Stephen Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009. 8

[85] knowledgeblob.com. What is agile? `http://knowledgeblob.com/agile/what-is-agile/`, 2016. [Online; accessed 16-August-2016]. xii, 90

[86] Craig Larman. *Agile and iterative development: a manager's guide*. Addison-Wesley Professional, 2004. 15

[87] Craig Larman e Victor R Basili. Iterative and incremental development: A brief history ieee computer, 2003. 87

[88] Dean Leffingwell. *Scaling software agility: best practices for large enterprises*. Pearson Education, 2007. 19

[89] Alexis Leon e Alan S Koch. *Agile software development evaluating the methods for your organization*. Artech House, Inc., 2004. 87

[90] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932. 35, 51

[91] Johan Linaker, Sardar Muhammad Sulaman, Martin Höst, e Rafael Maiani de Mello. Guidelines for conducting surveys in software engineering v. 1.1. 2015. 27, 28, 30, 32, 35

[92] Nebulon Pty. Ltd. Página do feature-driven development. `http://www.nebulon.com`, 2016. [Online; accessed 10-August-2016]. 96, 98

[93] Neil Maiden. User requirements and system requirements. *IEEE Software*, 25(2):90–91, 2008. 19

[94] Chris Mann e Frank Maurer. A case study on the impact of scrum on overtime and customer satisfaction. In *Agile Conference, 2005. Proceedings*, pages 70–79. IEEE, 2005. 48

[95] Michele Marchesi. The new xp. *Disponıvel em: www. agilexp. org/downloads/TheNewXP. pdf. Acessado em*, 30(05), 2007. 120, 123

[96] Likoebe M Maruping, Xiaojun Zhang, e Viswanath Venkatesh. Role of collective ownership and coding standards in coordinating expertise in software project teams. *European Journal of Information Systems*, 18(4):355–371, 2009. 134

[97] Alan SC Mazuco e Edna Dias Canedo. Reports with tdd and mock objects: an improvement in unit tests. *ICSEA 2016*, page 85, 2016. xiv, 45, 136, 137, 138

[98] C Melo, Viviane A Santos, Hugo Corbucci, Eduardo Katayama, Alfredo Goldman, e Fabio Kon. Métodos ágeis no brasil: estado da prática em times e organizações. *São Paulo: Departamento de Ciência da Computação do IME/USP*, 2012. xii, 13, 14, 40, 41, 86, 89

[99] Claudia de O Melo e Gisele RM Ferreira. Adoção de métodos ágeis em uma instituição pública de grande porte-um estudo de caso. In *Workshop Brasileiro de Métodos Ágeis (Agile Brasil 2010). Centro de Eventos da PUCRS (CEPUC), Porto Alegre*, 2010. 3

[100] Peter Middleton. Lean software development: two case studies. *Software Quality Journal*, 9(4):241–252, 2001. 99

[101] John Miltenburg e Jacob Wijngaard. Designing and phasing in just-in-time production systems. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 29(1):115–131, 1991. 102

[102] Deepti Mishra, Alok Mishra, e Sofiya Ostrovska. Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation. *Information and software Technology*, 54(10):1067–1078, 2012. 125, 126

[103] Sridhar Nerur, RadhaKanta Mahapatra, e George Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78, 2005. 15, 87

[104] Dan Schilling Nguyen et al. Success factors for building and managing high performance agile software development teams. *International Journal of Computer (IJC)*, 20(1):51–82, 2016. 129

[105] Anh Nguyen-Duc, Xiaofeng Wang, e Pekka Abrahamsson. What influences the speed of prototyping? an empirical investigation of twenty software startups. In *International Conference on Agile Software Development*, pages 20–36. Springer, 2017. 50, 68

[106] Robert L Nord, Ipek Ozkaya, e Philippe Kruchten. Agile in distress: Architecture to the rescue. In *International Conference on Agile Software Development*, pages 43–57. Springer, 2014. 15

[107] Taiichi Ohno. *Toyota production system: beyond large-scale production*. crc Press, 1988. xiii, 119

[108] Maria Marly de Oliveira. Como fazer pesquisa qualitativa. In *Como fazer pesquisa qualitativa*. Vozes, 2013. 25

[109] Aneta Ozieranska, Dorota Kuchta, Agnieszka Skomra, e Pawel Rola. The critical factors of scrum implementation in it project-the case study. *Journal of Economics & Management*, 25:79, 2016. 129

[110] Frauke Paetsch, Armin Eberlein, e Frank Maurer. Requirements engineering and agile software development. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 308–313. IEEE, 2003. 134

[111] Steve R Palmer e Mac Felsing. *A practical guide to feature-driven development*. Pearson Education, 2001. xiii, 88, 95, 97, 130

[112] David Parsons, Hokyoung Ryu, e Ramesh Lal. The impact of methods and techniques on outcomes from agile software development projects. In *IFIP International Working Conference on Organizational Dynamics of Technology-Based Innovation*, pages 235–249. Springer, 2007. 2

[113] Jeff Patton. Presentation: Embrace uncertainty by jeff patton. `https://www.infoq.com/news/2008/12/Uncertainty-Jeff-Patton`, 2008. [Online; accessed 11-April-2017]. xiii, 120, 121

[114] Mark C Paulk. *The capability maturity model: Guidelines for improving the software process*. Addison-Wesley Professional, 1995. 85

[115] J. Perakis. We can live on mars. `http://wecanliveonmars.com/wp/?tag=scrum-2`, 2015. [Online; accessed 20-August-2016]. xiii, 113

[116] Putu Adi Guna Permana. Scrum method implementation in a software development project management. *International Journal of Advanced Computer Science and Applications*, 6(9):198–204, 2015. xiii, 119, 120

[117] Andrew Pham e Phuong-Van Pham. Scrum em ação. *São Paulo: Novatec*, 2012. 82

[118] Minna Pikkarainen, Jukka Haikara, Outi Salo, Pekka Abrahamsson, e Jari Still. The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3):303–337, 2008. 124

[119] GUIDE PMBOK. Um guia do conhecimento em gerenciamento de projetos. *Quarta Edição*, 2013. 123

[120] M. Poppendieck e M. A. Cusumano. Lean software development: A tutorial. *IEEE Software*, 29(5):26–32, Sept 2012. 117

[121] Mary Poppendieck e Tom Poppendieck. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley, 2003. xiii, 88, 102, 140

[122] Mary B Poppendieck e Tom Poppendieck. *The Lean Mindset: Ask the Right Questions*. Pearson Education, 2013. 99

[123] Cleber Cristiano Prodanov e Ernani Cesar de Freitas. *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico-2ª Edição*. Editora Feevale, 2013. 28

[124] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. 34

[125] Vaclav Rajlich. Changing the paradigm of software engineering. *Communications of the ACM*, 49(8):67–70, 2006. 16

[126] Linda Rising e Norman S Janoff. The scrum software development process for small teams. *IEEE software*, 17(4):26–32, 2000. 123

[127] Evan Robinson. Why crunch mode doesn't work: 6 lessons. *IGDA Retrieved Feb*, 17:2009, 2005. 139

[128] Stefan Roock. extreme frameworking. xiii, 110

[129] Juha Savolainen, Juha Kuusela, e Asko Vilavaara. Transition to agile development-rediscovery of important requirements engineering practices. In *2010 18th IEEE International Requirements Engineering Conference*, pages 289–294. IEEE, 2010. 16, 18

[130] K Schwaber e Mike Beedle. gilè software development with scrum. 2002. 88

[131] Ken Schwaber. Scrum development process. In *Business Object Design and Implementation*, pages 117–134. Springer, 1997. 15

[132] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004. 2, 21

[133] Ken Schwaber e Jeff Sutherland. The scrum guide. *Scrum Alliance*, 2011. 15, 41, 46, 111, 113

[134] Scott Sehlhorst. To estimate or not to estimate - that is the question. `http://www.techwell.com/2013/05/estimate-or-not-estimate-question`, 2013. [Online; accessed 14-april-2017]. 140

[135] Antônio Joaquim Severino. *Metodologia do trabalho científico*. Cortez editora, 2014. 26

[136] Terry Shepard. Cisc 323 software successes and failures. `http://http://slideplayer.com/slide/9488510/`, 2007. [Online; accessed 29-april-2017]. xii, 19, 20

[137] James Shore et al. *The art of agile development*. "O'Reilly Media, Inc.", 2007. 117

[138] Indeed Site. Job trends. `http://www.indeed.com/jobtrends/q-Scrum-q-XP.html`, 2016. [Online; accessed 04-April-2017]. xii, 89

[139] Dag IK Sjoberg, Tore Dyba, e Magne Jorgensen. The future of empirical methods in software engineering research. In *2007 Future of Software Engineering*, pages 358–378. IEEE Computer Society, 2007. 24

[140] Nigel Slack, Stuart Chambers, e Robert Johnston. *Operations and process management: principles and practice for strategic impact*. Pearson Education, 2009. xiv, 142, 143

[141] Stanley F Slater e John C Narver. Market orientation and the learning organization. *The Journal of marketing*, pages 63–74, 1995. 130

[142] Hubert Smits e Guy Pshigoda. Implementing scrum in a distributed software development organization. In *Agile Conference (AGILE), 2007*, pages 371–375. IEEE, 2007. 140

[143] Vitor Estevao Silva Souza e Ricardo de Almeida Falbo. An agile approach for web systems engineering. In *Proceedings of the 11th Brazilian Symposium on Multimedia and the web*, pages 1–3. ACM, 2005. 132

[144] Michael K Spayd. Evolving agile in the enterprise: implementing xp on a grand scale. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 60–70. IEEE, 2003. 17

[145] Jennifer Stapleton. *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press, 1997. xii, 91, 92

[146] Jennifer Stapleton. *DSDM: Business focused development*. Pearson Education, 2003. 87, 92

[147] Emily Symonds. A practical application of surveymonkey as a remote usability-testing tool. *Library Hi Tech*, 29(3):436–445, 2011. 27

[148] David Talby, Arie Keren, Orit Hazzan, e Yael Dubinsky. Agile software testing in a large-scale project. *IEEE software*, 23(4):30–37, 2006. 128, 129

[149] TCU/Brasil. Levantamento de auditoria. conhecimento acerca da utilização de metodologias Ágeis nas contratações de software pela administração pública federal, 2013. 1

[150] Techifide. Pmvp for startups. `https://www.techifide.com/mvp/`, 2017. [Online; accessed 08-jun-2017]. 50

[151] Carlos Renato Theóphilo e Gilberto de Andrade Martins. Metodologia da investigação científica para ciências sociais aplicadas. *São Paulo: Atlas*, 2:104–119, 2009. 25

[152] In Todd. The cone of uncertainty. xiv, 141, 142

[153] James E Tomayko. Engineering of unstable requirements using agile methods. In *International Workshop on Time-Constrained Requirements Engineering (TCRE'02). Essen*, 2002. 49

[154] Trung Hieu Tran, Nhat Duy Duong, et al. Adoption of agile software development in vietnam. 2014. xiii, 124, 125

[155] Salvador Trujillo, Don Batory, e Oscar Diaz. Feature oriented model driven development: A case study for portlets. In *Proceedings of the 29th international conference on Software Engineering*, pages 44–53. IEEE Computer Society, 2007. 130

[156] Dan Turk, Robert France, e Bernhard Rumpe. Limitations of agile software processes. *arXiv preprint arXiv:1409.6600*, 2014. 124

[157] Tribunal de Contas. União. Levantamento sobre aplicação de metodologias ágeis em desenvolvimento de software. `http://portal.tcu.gov.br/lumis/portal/file/fileDownload.jsp?fileId=8A8182A24F0A728E014F0B2ECB8A00CC`, 2013. [Online; accessed 30-October-2016]. 2

[158] Betteke van Ruler. Agile public relations planning: The reflective communication scrum. *Public Relations Review*, 41(2):187–194, 2015. 117

[159] Yaqian Wang e Tony Huzzard. The impact of lean thinking on organizational learning. *Lund University, Sweden*, pages 1–19, 2011. 130

[160] Kelly Waters. Prioritization using moscow. *Agile Planning*, 12, 2009. xiii, 122

[161] Laurie Williams. Case study retrospective: Kent beck's xp versions 1 and 2. *Ponencia en "Annual Research Review*, 2005. xiii, 107

[162] Laurie Williams, Robert R Kessler, Ward Cunningham, e Ron Jeffries. Strengthening the case for pair programming. *IEEE software*, 17(4):19–25, 2000. 45

[163] Hanna Wlodarkiewicz-Klimek. Agility of knowledge-based organizations. In *Advances in Ergonomics of Manufacturing: Managing the Enterprise of the Future*, pages 375–384. Springer, 2016. 52

# Apêndice A

# Metodologias Ágeis de Desenvolvimento de Software

Este apêndice descreve as metodologias ágeis estudadas no presente trabalho, como foram concebidas, seus autores, principais óbices e fatores de sucesso que possibilitaram a adoção em larga escala, em todo o mundo.

## A.1 Histórico

Ao longo do tempo, desde quando os cientistas optaram por resolver problemas com o uso da máquina computacional, a busca pela rapidez e qualidade sempre foi a tônica, desde quando os primeiros cálculos feitos pelo ENIAC, uma máquina de 30 toneladas na época, considerada um prodígio, Figura A.1. A procura por ganhos sustentáveis de produtividade e de qualidade no desenvolvimento de software com o uso de diversos modelos, formas de organização do trabalho e abordagens cada vez mais inovadoras, tornou-se uma evidência na Engenharia de Software. Isto veio a colaborar na construção de novos pensamentos na forma de desenvolver software, uma questão que merece ser estudada.

Sob esse contexto, é interessante o que aponta Phan [117]:

> "Como as pessoas e as empresas tendem a abraçar as mudanças lentamente, só o fato de o movimento ágil, em especial o Scrum, terem decolado e gerado tanta excitação já é um bom sinal, e que independentemente da razão, o método ágil e o Scrum trazem mudanças, e estas são sempre difíceis, a menos que sejam gerenciadas apropriadamente. Mas com um bom gerenciamento, tais dificuldades podem ser transformadas em oportunidades."

### A.1.1 Marco Um: o Manifesto Ágil

O Manifesto Ágil foi um acontecimento ocorrido entre os anos de 2000 e 2001, em Oregon e posteriormente em Utah, em uma segunda reunião. Após essa segunda reunião, surgiu um documento de mesmo nome que passou a nortear a comunidade científica a respeito de metodologias ágeis de desenvolvimento de software em contraposição às metodologias tradicionais. Na primeira reunião, um grupo de líderes da comunidade do Extreme Programming (XP) discutiram as várias questões que envolviam o processo de desenvolvimento. Na ocasião, foi debatida a relação entre XP e processos semelhantes conhecidos como métodos leves, uma denominação aos novos métodos de desenvolvimento de software que começariam a surgir naquela época. Os métodos leves eram considerados como uma espécie de reação contrária aos métodos julgados pesados, estes denominados tradicionais, que pautavam muito mais para a documentação e a formalização exagerada, quase sempre influenciados pelo modelo em cascata de desenvolvimento.

Eles concluíram que o XP era um método bem específico que se destacava entre as outras metodologias leves e que, mesmo assim, haveria um espaço comum para todas

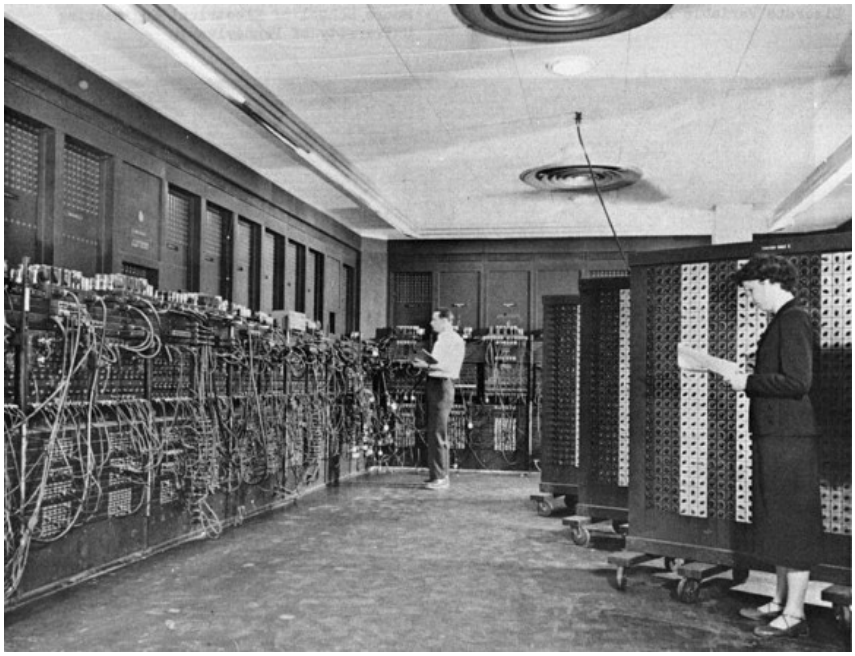


Figura A.1: O primeiro computador: a capacidade de processamento era de 5.000 operações por segundo.

elas. Em fevereiro de 2001, nas montanhas de Utah, foi realizada uma segunda reunião, impulsionando de vez as metodologias ágeis. Ao longo da reunião, várias deliberações foram postas em debate, surgindo um consenso comum entre os participantes: a reunião, por si só, necessitava ser divulgada para alcançar um patamar mais elevado. Por fim, optaram por redigir um documento formal que serviria como um "grito de guerra" aos novos processos de desenvolvimento de software que praticamente emergiam em um novo cenário. Entretanto, o termo "leve" não era representativo o bastante para o movimento, era necessário algo mais, que chamasse a atenção da comunidade, algo que realmente significasse aquilo que eles estavam propondo. Assim, "métodos leves" deixou de ser uma opção válida para dar lugar a "métodos ágeis", que melhor captaria a essência da proposta.

### A.1.2 Valores e Princípios

No final da reunião, o Manifesto Ágil se configurou como uma ferramenta que não rejeita os processos com a sua documentação diversa ou o planejamento em si, mas elenca a importância entre os indivíduos e interações e, principalmente, o software funcionando, com a presença do cliente, possibilitando respostas mais rápidas a mudanças.

Os dezessete participantes do movimento, autointitulados *anarquistas* [17], Figura A.2, apresentaram um documento chamado Manifesto Ágil, que continha, basicamente, 4 valores e 12 princípios, conforme vemos nas Tabelas A.1 a A.2:

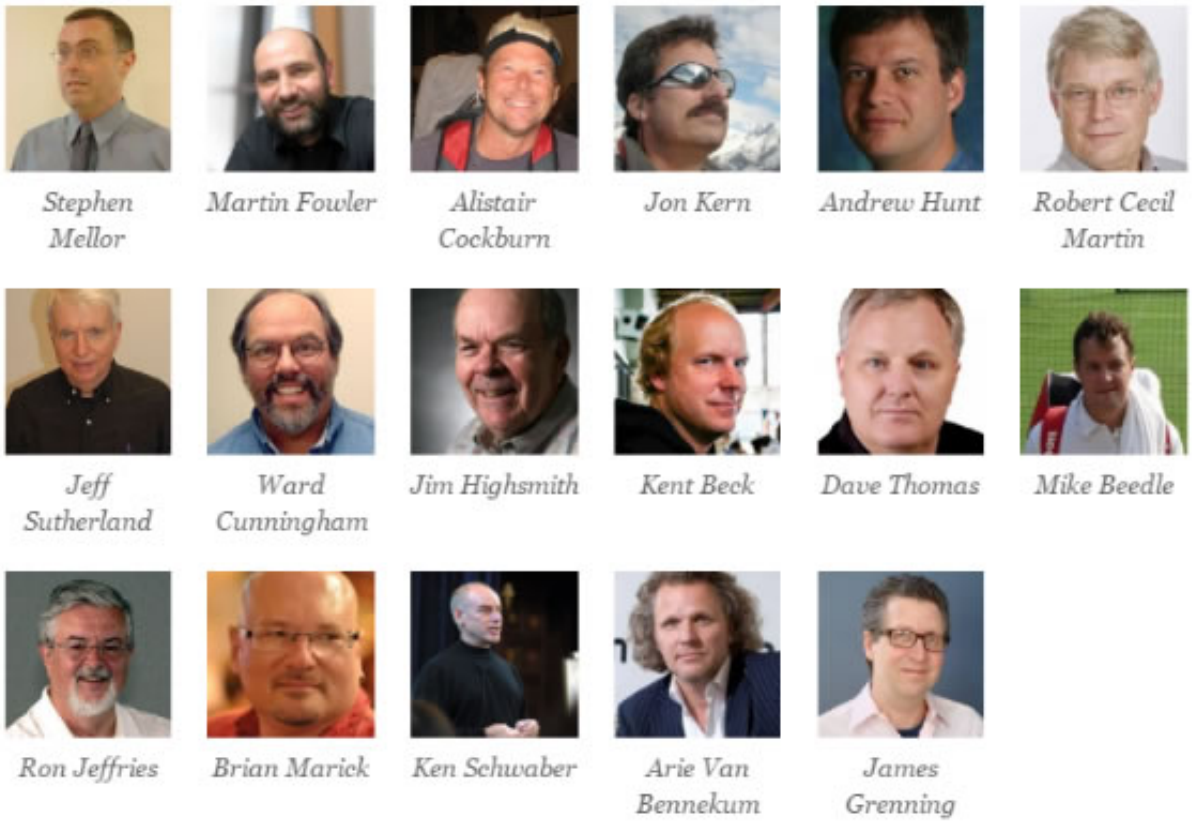



Figura A.2: Os protagonistas do Manifesto Ágil: Autodenominados de anarquistas.

Tabela A.1: Os quatro valores do Manifesto Ágil.

| # | **Valores** |
|---|---|
| 1 | **Indivíduos e interação entre eles** mais que processos e ferramentas. |
| 2 | **Software em funcionamento** mais que documentação abrangente. |
| 3 | **Colaboração com o cliente** mais que negociação de contratos. |
| 4 | **Responder a mudanças** mais que seguir um plano. |

Tabela A.2: Os doze princípios do Manifesto Ágil.

| # | **Princípios** |
|---|---|
| 1 | Priorizar sempre o cliente, por meio de entregas contínuas e adiantadas. |
| 2 | Aceitar as mudanças de requisitos, mesmo que elas aconteçam ao fim de uma determinada release. Isto irá propiciar que o cliente obtenha mais satisfação e vantagens competitivas. |
| 3 | Entregas frequentes de software funcionando, em poucas semanas. |
| 4 | Interação entre os desenvolvedores e o pessoal ligado às regras de negócios, trabalhando juntos, lado a lado. |
| 5 | A motivação do pessoal deve ser buscada, invariavelmente, dando a eles o suporte necessário. |
| 6 | Priorizar a conversa face-a-face, para transmitir as informações necessárias ao desenvolvimento. |
| 7 | A medida do sucesso é o software funcionando em pouco espaço de tempo. |
| 8 | Desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente. |
| 9 | A atenção redobrada e a excelência técnica, aliada ao bom design, aumenta a agilidade. |
| 10 | Simplicidade. |
| 11 | Times energizados e auto-organizados. |
| 12 | Ao fim de cada release, a equipe senta junto para refletir sobre o trabalho realizado, para a reflexão sobre onde acertou, onde errou, e assim poder melhorar suas ações. |

De fato, Beck et al. [17] afirmam que o Movimento Ágil não é anti-metodologia, que na verdade, muitos deles, os autores, querem restaurar a credibilidade e restabelecer um equilíbrio. Que abraçam a modelagem, mas não apenas para arquivar alguns diagramas em um repositório corporativo empoeirado. Que também abraçam a documentação, mas não centenas de páginas que não podem ser mantidas e papéis que raramente serão utilizados. Que reconhecem o planejamento, mas reconhecem seus limites em um ambiente turbulento. Que consideram aqueles que os julgam de "hackers" como ignorantes, tanto de metodologias quanto da definição do termo "hacker".

### A.1.3 Consolidação

Cohen et al., em revisão publicada em 2004 [30], enfatizam a história do desenvolvimento ágil mostrando algumas de suas raízes e, em particular, discutem as relações entre o desenvolvimento ágil e o *Capability Maturity Modelo (CMM)* [114]. Eles descrevem o

estado da arte no que diz respeito aos principais métodos ágeis e suas características, além de descreverem os resultados online de uma interessante discussão entre 18 praticantes, muitos dos quais envolvidos na definição de diferentes metodologias de desenvolvimento ágil. Eles discutiram questões tais como a introdução e gerenciamento de projetos em desenvolvimento ágil. Relataram também experiências atuais e pesquisas em torno de sete estudos de caso de desenvolvimento ágil. Os autores afirmaram que os métodos ágeis seriam consolidados em um futuro próximo, assim como métodos orientados a objetos foram consolidados no passado. Além disso, eles não acreditavam que os métodos ágeis governariam os métodos tradicionais, em vez disso, eles acreditavam que ágil e métodos tradicionais teriam uma relação simbiótica, em que fatores como o número de pessoas trabalhando em um projeto, domínio de aplicação, criticidade e inovação, iriam determinar qual o processo a ser adotado, mostrando uma postura bem mais coerente.

Em 2017, o Manifesto Ágil completa 16 anos de pleno funcionamento e, podemos dizer, consegue se manter como uma das abordagens de desenvolvimento de software mais utilizadas, isto ao redor do mundo. Conforme Melo et al. [98], empresas e pesquisadores procuram compreender seu estágio de adoção, benefícios e limitações.

Hoje, pode-se dizer que as práticas ágeis no Brasil se consolidam bem, pois mesmo após 16 anos de sua adoção ainda mantém-se forte, o que é um indício de sua importância e relevância para a Engenharia de Software, de que realmente elas vieram para colaborar com as indústrias, naquilo que for aplicável.

## A.2 As Metodologias Ágeis, Hoje

O que popularizou a utilização de métodos ágeis ao redor do mundo foi a rapidez com que colocam os softwares funcionando em tempo mínimo, elevando a satisfação do cliente. As recompensas para quem opta por metodologias ágeis prometem ser grandes, não à toa, empresas como Google, Yahoo, Microsoft e IBM as utilizam.

Conforme vemos em Cruz [35], o ágil não é melhor que o tradicional, e muito menos o inverso dele. Sendo assim, o que se busca nessa abordagem é a união das duas vertentes para trazer mais força às equipes de gerenciamento de projeto e mostrar que a crença de que só um deles resolveria todos os problemas pode ser um grande risco.

Há muito o que se considerar quando analisamos mais profundamente as metodologias ágeis, enquanto existem enxurradas de elogios, também existem tantas outras críticas destrutivas. Vejamos o que dizem Cohen et al. [30], a respeito disto:

> "Os Métodos Ágeis estão a criar um verdadeiro alarido na comunidade. Enquanto algumas pessoas os consideram a melhor coisa que aconteceu com o desenvolvimento de software nos últimos anos, outras os vêem como uma reação à engenharia de software, consideradas como práticas nocivas, como *hackers*."

...e existem vários pesquisadores que se aprofundam no estudo dessa dicotomia, como vemos em [89], onde há apontamentos das benéfices e óbices, tanto de um lado quanto de outro:

> "Os Métodos ágeis têm tomado o desenvolvimento de software pela tempestade, mas uma controvérsia crescente sobre a sua verdadeira eficácia agora coloca os defensores contra detratores em uma batalha aquecida de reivindicações."

Recentemente (janeiro/2017) o Instituto CMMI publicou um documento intitulado "A Guide to Scrum and CMMI: Improving Agile Performance with CMMI.", conforme relatado em [75], o que vem causando alguma manifestação de agilistas que não poupam seus esforços em criticar o modelo antigo, mesmo que já o tenha consolidado na Engenharia de Software, mas também colaborando e alinhando os pensamentos, colocando as duas abordagens frente a frente, possibilitando assim que cada um faça o seu juízo.

Se tivéssemos que definir Metodologia Ágil em toda a sua essência, e em poucas palavras, começaríamos por Ericksson et al. [50]:

> "Agilidade significa despir o máximo de peso, comumente associado com metodologia tradicional de desenvolvimento de software, o quanto possível, para promover a rápida resposta às mudanças ambientais, mudanças nos requisitos de usuário e acelerar prazos do projeto."

...e terminaríamos dizendo que "métodos ágeis constituem um conjunto de práticas de desenvolvimento que foram criadas por profissionais experientes..." [1], e que "...esses métodos podem ser vistos como uma reação ao extenso planejamento tradicional, que enfatiza uma engenharia baseada em racionalidade e em abordagem." [103]. Diríamos ainda que "...os problemas são completamente determináveis e que soluções ótimas e previsíveis existem para cada tipo deles." [18], porquanto, o uso ou não de metodologias ágeis irá depender da natureza do projeto e do contexto onde ele se insere.

### A.2.1 O Campo das Metodologias Ágeis

A metodologia ágil desponta em um artigo que descreve a história do desenvolvimento de software iterativo e incremental, onde Larman et al. [87] apontam o Método Dinâmico de Desenvolvimento de Sistemas (DSDM) [146], como o primeiro método ágil na história, seguida da Programação Extrema (XP) [16], originária do projeto Chrysler C3 em 1996 [4]. A palavra "ágil" não foi uma exclusividade do Manifesto Ágil ocorrido em 2001/2002, pois ela, na verdade, foi utilizada em combinação com processos de software pela primeira vez em 1998 por Aoyama: "A pressão para ser a primeira indústria a comercializar um novo produto tem acelerado o processo de desenvolvimento. Modelos de processos de hoje exigem agilidade [...]" [8]. vários outros métodos se seguiram, incluindo a

família das metodologias Cristal [28], EVO [62], Feature-Driven Development [111], Lean Desenvolvimento [121] e Scrum [130]. Em 2004, uma nova versão de XP apareceu [16], o que se convencionou chamar XP2.

No ano de 2008, Dybå e Dingsøyr [44] realizaram uma revisão sistemática onde apontava as principais metodologias ágeis em evidência na época. O Scrum, de Ken Schwaber, timidamente começava a ganhar o interesse da comunidade, enquanto o XP ainda era a metodologia predileta, conforme vemos no gráfico da Figura A.3:

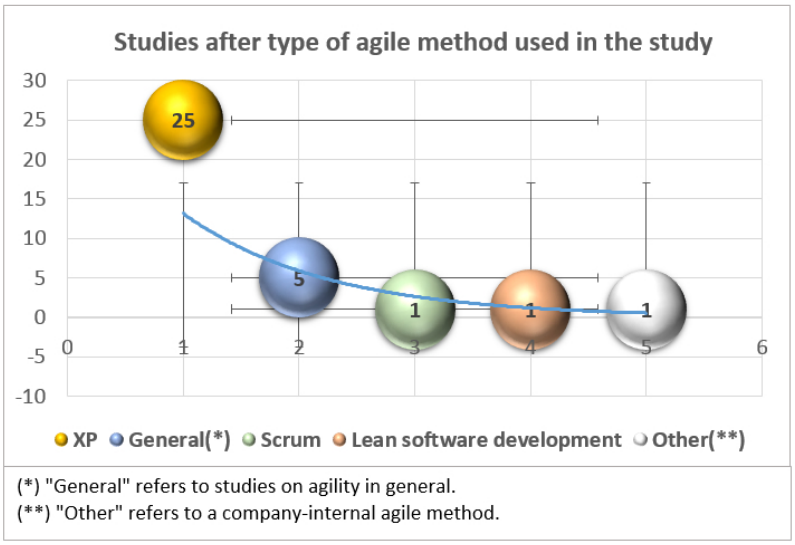



Figura A.3: Principais metodologias ágeis em 2008, adaptado de Dybå [44].

Cinco anos depois, Kaisti et al. [77], apresentavam uma outra revisão sistemática com novos números. A metodologia XP ainda era predominante na comunidade ágil:

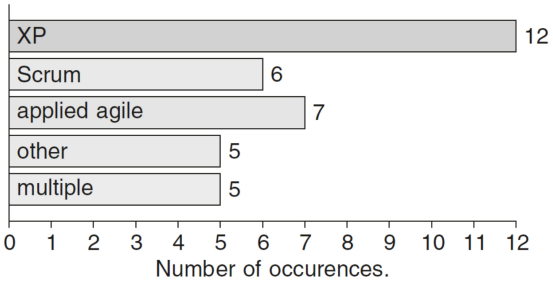



Figura A.4: Número de ocorrências de metodologias ágeis em 2013, segundo estudos de Kaisti et al. [77].

Segundo Indeed [138], Um site americano especialista em estatística de ofertas de empregos por meio de indexação de palavras-chave que mais crescem em determinados períodos de tempo, o Scrum é que hoje se configura como a metodologia que mais possui oferta de emprego. Ao realizar uma pesquisa usando as palavras-chave "Scrum" e "XP", um gráfico fornece uma imagem clara sobre o rápido crescimento da procura por profissionais Scrum. A busca é feita em diversos sistemas operacionais móveis, mídias sociais, e uma variedade de áreas em tecnologia da informação.

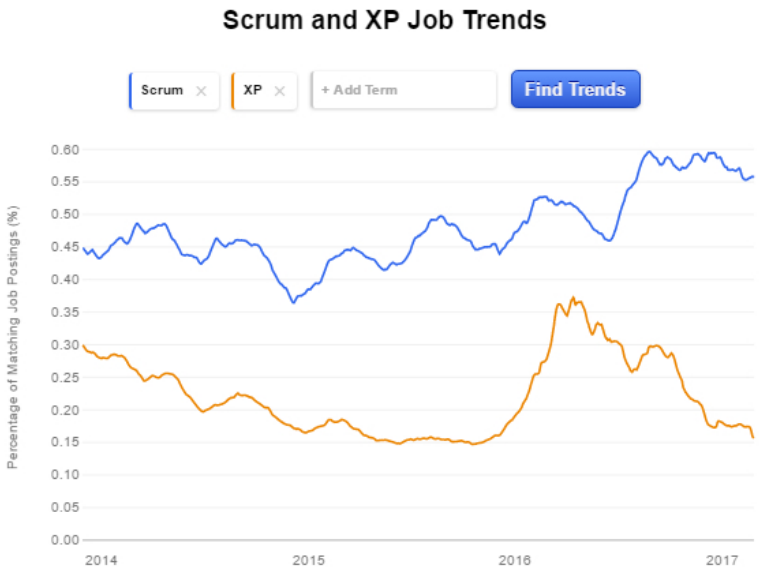



Figura A.5: A procura por profissionais de Scrum hoje é maior que os de XP. [138].

Segundo Melo et al. [98] em estudo recente no Brasil, esta tendência também se configura, porém de uma forma bem mais acentuada com o Scrum na dianteira. Mas é importante verificar aqui uma tendência que vem se consolidando nos últimos anos, como a combinação das metodologias XP e Scrum, conforme podemos ver na Figura A.6:

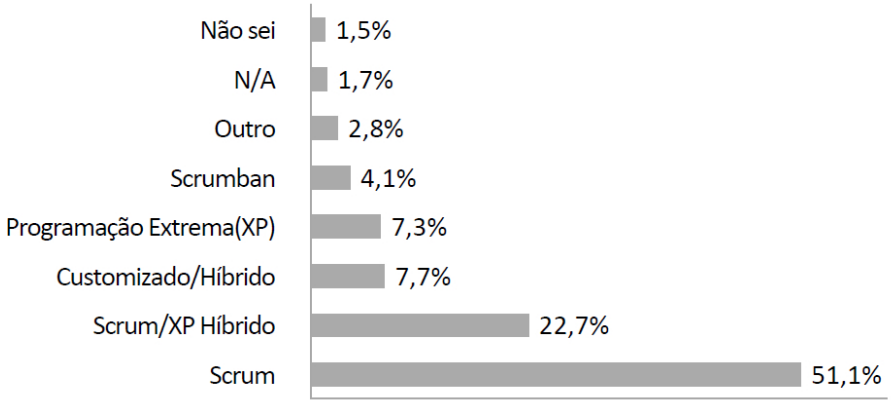



Figura A.6: Principais tendências de métodos ágeis no Brasil segundo Melo et al. [98]. [138].

A seguir, mostramos um apanhado sobre as metodologias ágeis mais evidentes na Engenharia de Software Mundial e atual, o clássico "guarda-chuva", como gostam de representar os entusiastas, diante do presente estudo terciário. No próximo capítulo, as práticas comumente empregadas por essas metodologias, serão analisadas dentro de um contexto de pesquisa empírica.

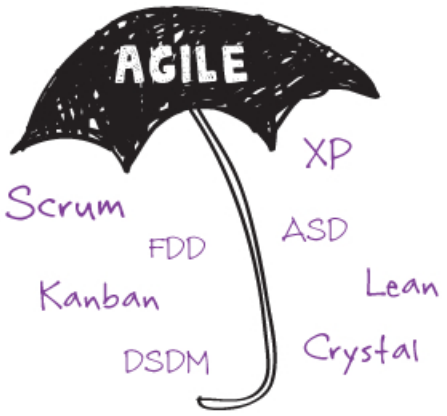



Figura A.7: O clássico "guarda-chuva" das principais metodologias ágeis [85].

### A.2.2 Adaptive Software Development (ASD)

O Adaptive Software Development (ASD) incorpora mais conceitos da teoria de sistemas adaptativos complexos. Nesse contexto, propõe trocar o ciclo de desenvolvimento, antes baseado em **planejar**, **projetar** e **construir**, por um outro com as fases de **especular**, **colaborar** e **aprender**. Isto é importante porque os ciclos são diferentes, senão vejamos: o primeiro consideraria a estabilidade no ambiente de negócios, e o segundo focaria mais em ambientes de incertezas e de grandes mudanças [70] – aqui considerando uma visão comum a todos os métodos ágeis. Dentro desse contexto, a figura abaixo, adaptada de Highsmith [69], ilustra bem as três fases do método:

**Especular**: Na fase de Especular, é realizada a iniciação do projeto com todo o rigor do planejamento. Nesta fase, são realizadas várias sessões, as nominadas Joint Application Development (JAD), onde são definidos os objetivos, os requisitos, os problemas, os riscos e as estimativas, finalizando com o escopo de projeto.

**Colaborar**: Aqui é realizada a implementação do sistema. Os desenvolvedores devem tentar promover ao máximo a comunicação e a colaboração entre as pessoas e os times. Práticas como discussões sobre quadro branco, conversas pessoais (em pé), ou até aplicação

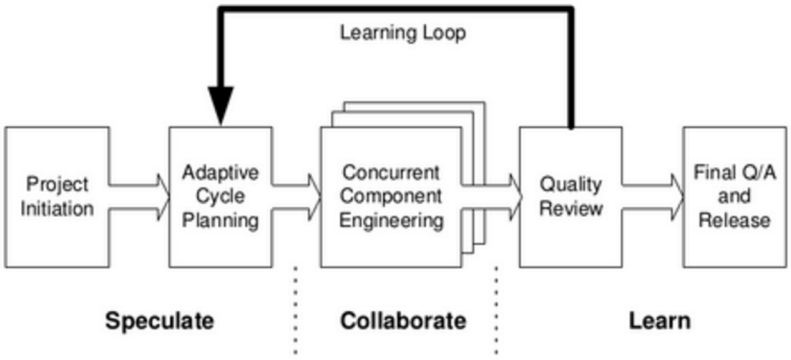



Figura A.8: Ciclo de vida da metodolgia ASD [69].

de outras práticas consagradas do XP, como a programação em pares.

**Aprender**: Nesta fase ocorrem as revisões da qualidade, onde todos os trabalhos dos participantes são avaliados. Ao final, as informações obtidas são retro-alimentadas, permitindo que novos planejamentos sejam feitos de forma mais adequada a *posteriori*. Segundo Highsmith [69], avalia-se também o desempenho das iterações, nos seguintes aspectos:

- Qualidade resultante sob a ótica do cliente;
- Qualidade técnica;
- Análise das práticas adotadas;
- O status do projeto.

### A.2.3 Dynamic Systems Development Method (DSDM)

O Dynamic Systems Development Method (DSDM) é uma reformulação dos métodos RAD empreendida por um consórcio de companhias que fornecem serviços e treinamentos em engenharia de software. Consta de mais uma ferramenta dentro do guarda-chuva das metodologias ágeis, onde os proponentes sugerem a formação do produto por meio da realização de um protótipo inicial que vai se evoluindo sempre com a presença do cliente dentro da fábrica [145].

O processo é divido em cinco fases onde, em uma fase de pré-projeto, tem-se o objetivo de definir a sua viabilidade, observando os diversos aspectos gerenciais, como financiamento das atividades e um plano para o estudo dessa viabilidade, a fim de se definir se a metologia DSDM é a mais adequada. Nas fases subsequentes, são observados

os processos que serão afetados e suas necessidades [145], onde haverá a definição do escopo. A Figura A.9 mostra as fases de execução de um projeto usando a metodologia.

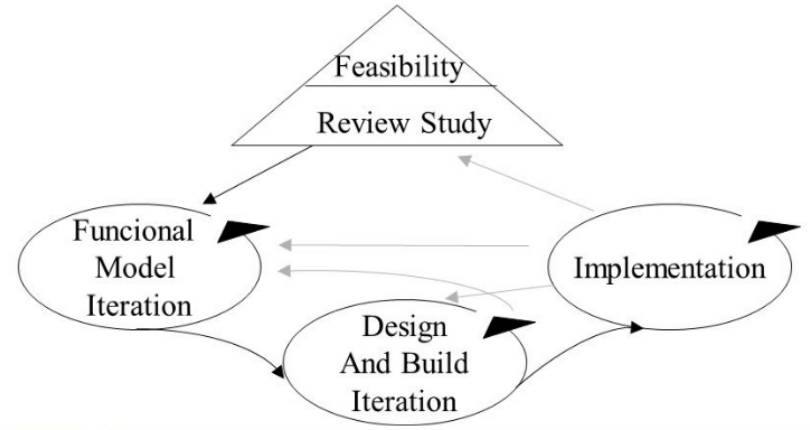



Figura A.9: Fases da metodologia DSDM, conforme Stapleton [145].

De acordo com Stapleton [146] o DSDM está embasado em nove princípios:

- Envolvimento ativo do usuário;
- O time tem o poder para tomar decisões;
- Foco na entrega frequente;
- Regra de negócio é o critério essencial para a aceitação das entregas.
- Desenvolvimento iterativo e incremental;
- Todas as mudanças são passíveis de serem revertidas (refatoração);
- Requisitos alinhados em alto nível;
- O teste é integrado por todo o ciclo de vida;
- Abordagem colaborativa e cooperativa entre as partes envolvidas.

Além desses princípios, Stapleton [145] cita outras técnicas utilizadas pelo método, enumeradas abaixo:

- **Time-boxes**: Fixação de datas de entregas;
- **MoSCoW**: Prioridade para os requisitos principais;
- **Modelagem**: Prover melhor entendimento, sem burocracias;
- **Prototipação**: Facilitar a compreensào do cliente;

- **Teste**: Sistematizada e contínua;
- **Gerência de configuração**: Garantir a continuidade das entregas.

o framework da metodologia DSDM, hoje, é mantido pela DSDM Consortium, visto em www.dsdm.org. Este grupo é mundial e conta com diversas empresas que são membros do grupo que mantém o DSDM, responsáveis por realização de treinamentos e consultorias, além de emissão de certificações. É importante frisar que o DSDM possui uma metodologia bem diferente das outras, exceto Scrum, onde apresenta algumas simillaridades, com processos de modelagem, concepção do tempo, e outras implementações de forma bem flexível, mas mantendo prazos definidos.

### A.2.4 Crystal Methodologies

A metodologia Crystal, na realidade, compõe-se de um grupo de metodologias ágeis de desenvolvimento de software elencadas dentro de uma mesma família, onde o foco principal é a gestão de pessoas. O grupo foi criado por Alistair Cockburn, um pesquisador com grande experiência em gestão de projetos. Cockburn começou sua carreira no campo da tecnologia de comunicação e desenvolvimento, na IBM, em 1984. A partir de 1992 assumiu um papel consultivo na empresa, em particular os métodos de desenvolvimento de software orientado a objetos. Em seus estudos pesquisas, o pesquisador sentiu a necessidade de uma metodologia para o desenvolvimento de software de uma forma mais otimizada, explorando as características especificas de cada membro da equipe.

Segundo Cockburn [28], os membros que compõem as equipes têm um conjunto diferente de talentos e habilidades, ou seja, é fundamental utilizar um processo exclusivo feito sob medida para eles.

As metodologias Crystal são consideradas e descritas como metodologias leves e foram criadas para atender às equipes de diferentes tamanhos que necessitem de estratégias para resolver problemas diversos. Estabelecem princípios para cada formato de projeto tendo como base a sua complexibilidade. Ainda segundo Cockburn, isso é proposital, pois a ideia é que cada organização defina as tarefas que lhes forem mais suscetíveis ao projeto. Por convenção, a família de metodologias Crystal é dividida em cores, como a seguir:

- Crystal Clear;
- Crystal Yellow;
- Crystal Orange;
- Crystal Red;

- Crystal Maroon;
- Crystal Diamond;
- Crystal Sapphire.

Esses membros da família cristal são regidos por alguns princípios. Cockburn descobriu que quanto mais desses princípios fossem inseridos em um projeto, as chances de obtenção de sucesso mais se elevariam. A Figura A.10 mostra essa abordagem, conforme as cores dos membros se tornem mais escuras, tem-se um maior peso dos métodos, o que é necessário devido à complexidade dos projetos. O peso é representado pela quantidade de artefatos e a rigidez da gerência, itens que são absorvidos entre os elementos definidos para cada método: papéis, habilidades, times, técnicas, atividades, processos, artefatos, produtos de trabalho, padrões, ferramentas, personalidades, qualidade e valores da equipe [69]. Atribuem-se pesos para cada um desses itens.

| | | Crystal Methodologies | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Clear | Yellow | Orange | Red | Maroon | Diamond | Saphire |
| Criticality of the Project | Life (L) | L6 | L20 | L40 | L80 | L200 | L300 | L500 |
| | Essential Money (E) | E6 | E20 | E40 | E80 | E200 | E300 | E500 |
| | Discretionary Money (D) | D6 | D20 | D40 | D80 | D200 | D300 | D500 |
| | Comfort (C) | C6 | C20 | C40 | C80 | C200 | C300 | C500 |
| | | 1 to 6 | 7 to 20 | 21 to 40 | 41 to 80 | 81 to 200 | 201 to 300 | 301 to 500 |
| | | Number of People involved in the Project | | | | | | |

Figura A.10: Distribuição dos métodos da família Crystal a partir de duas dimensões. Adaptado de Cockburn [28].

Pode ser usada em conjunto com XP ou Scrum e destaca a importância das seguintes práticas de desenvolvimento:

- Comunicação iterativa;
- Jogo do planejamento;
- Modificações constantes (refatoração);
- Desenvolvimento incremental e iterativo, com incrementos de até 4 meses;
- Reflexão antes e depois de cada incremento (feedback).

Em sua pesquisa, Cockburn ainda definiu o comportamento das pessoas em equipes, onde chegou-se à conclusão de que a comunicação frente a frente eliminaria dúvidas com maior rapidez, através de discussões em grupo, mas que, ao mesmo tempo, as pessoas tendem a perder o desempenho ao longo do tempo. Esforços devem ser canalizados para contornar esse problema em específico.

### A.2.5 Feature-driven development (FDD)

Feature Driven Development, criado por Peter Coad, Jeff De Luca e Stephen Palmer, conforme visto em [111], é uma metodologia ágil para gerenciamento e desenvolvimento de software que combina as principais vantagens das metodologias ágeis com técnicas baseadas em modelos, em uma única abordagem completa para a Engenharia de Software. Ela é totalmente orientada a objetos, o que a torna bem eclética, e por isto conquista os três principais públicos envolvidos na construção do software, conforme de Luca et al. [37]: clientes, gerentes e desenvolvedores.

Em 2002, Stephen Palmer, gerente de desenvolvimento do projeto em Singapura, e John Mac Felsing, arquiteto senior na TogetherSoft, publicaram o livro "A Practical Guide to Feature Driven Development" [111], com a versão completa, atualizada e comentada da metodologia. Em 2003, David Anderson, que foi o especialista em interface com o usuário no projeto de Cingapura, publicou um marco na literatura ágil, "Agile Management for Software Engineering: Using the Theory of Constraints for Business Results" [5], onde oferece uma análise profunda sobre a FDD, entre outras metodologias, além de material inédito.

A FDD chama a atenção pelas seguintes características, que a torna bem peculiar:

- Modelagem em objetos de domínio;
- Desenvolvimento orientado por características das funcionalidades;
- Integração regular (build);
- Resultados úteis a cada duas semanas ou menos;
- Blocos bem pequenos de funcionalidades valorizadas pelo cliente, chamadas de "Features";
- Planejamento detalhado e guia para medição;
- Rastreabilidade e relatórios com incrível precisão;
- Monitoramento detalhado dentro do projeto, com resumos de alto nível para clientes e gerentes, tudo em termos de negócio;

- Fornece uma forma de saber, dentro dos primeiros 10% de um projeto, se o plano e a estimativa são sólidos;
- Reportar a visibilidade do projeto (todos sabem sobre a evolução do sistema).

Com relação às outras metodologias de desenvolvimento de software, situa-se numa posição intermediária entre as abordagens mais prescritivas, como o Processo Unificado, e as abordagens Ágeis (XP, Scrum, Crystal, etc.).

#### A.2.5.1 A Técnica

Conforme vemos no site oficial do FDD [92], para que seja possível gerenciar de forma correta um projeto usando FDD, existe a definição de alguns papéis centrais responsáveis por assumir a gerência, algo bem simples: O **gerente**, o **arquiteto-chefe**, o **dono das classes** e o **programador-chefe**. Esse último é também um desenvolvedor que cuida da implementação das regras de negócio e deve ser conhecedor profundo delas. Ele é o criador do time de métodos ágeis e que conduzirá o desenvolvimento através da verificação de quais são as classes que serão usadas, recrutando seus respectivos "donos". Com isso, podem existir, dependendo do tamanho da equipe, diversos times de desenvolvimento, o que pode fazer com que os desenvolvedores trabalhem em mais de um time, com recomendação de, no máximo, três, para cada iteração. O programador-chefe pode também acumular a função de "dono" de algumas classes. O papel do "dono" das classes é o de manter a documentação em dia e em ordem.

#### A.2.5.2 O Processo

A modelagem é feita para cada funcionalidade associada com a troca de informações entre o usuário da ponta da linha e o sistema. Assim, poder-se-á ter uma visão mais clara e específica do comportamento de cada requisito, suas pormenoridades, e checando sempre com o cliente, se é aquilo mesmo que ele solicitou — o lema é cliente satisfeito. A modelagem, nesse caso, quase sempre é feita pelo programador-chefe e a elaboração do código massivo, dita "pesada", pelos demais programadores do time.

Abaixo, as etapas desse processo:

1. Desenvolver um modelo abrangente;
2. Construir a lista de funcionalidades;
3. Planejar por funcionalidades;
4. Detalhar cada funcionalidade e

5. Construir por funcionalidades.

Dessas etapas, as três primeiras pertencem à parte de planejamento do processo e as duas últimas, ao desenvolvimento, que são consideradas a parte iterativa. Os trabalhos de modelagem de negócios estão contidos dentro da parte iterativa, conforme podemos observar na Figura A.11.

#### A.2.5.2.1 (1) Desenvolver um modelo abrangente:

A primeira etapa do processo consiste na criação do modelo de domínio, com a concepção das classes de domínio, representando as entidades abstraídas das regras de negócios, com todas as relações apuradas e com um nível de detalhamento apenas suficiente para dar forma ao sistema de um modo geral. A finalidade desta etapa, segundo Palmer e Felsing [111], é diminuir um possível ou provável refatoramento no futuro, uma vez que gera um arcabouço para a construção do sistema nas etapas subsequentes, mantendo assim uma integridade conceitual. Essa integridade conceitual tem de ser cuidadosamente estudada para evitar problemas futuros.

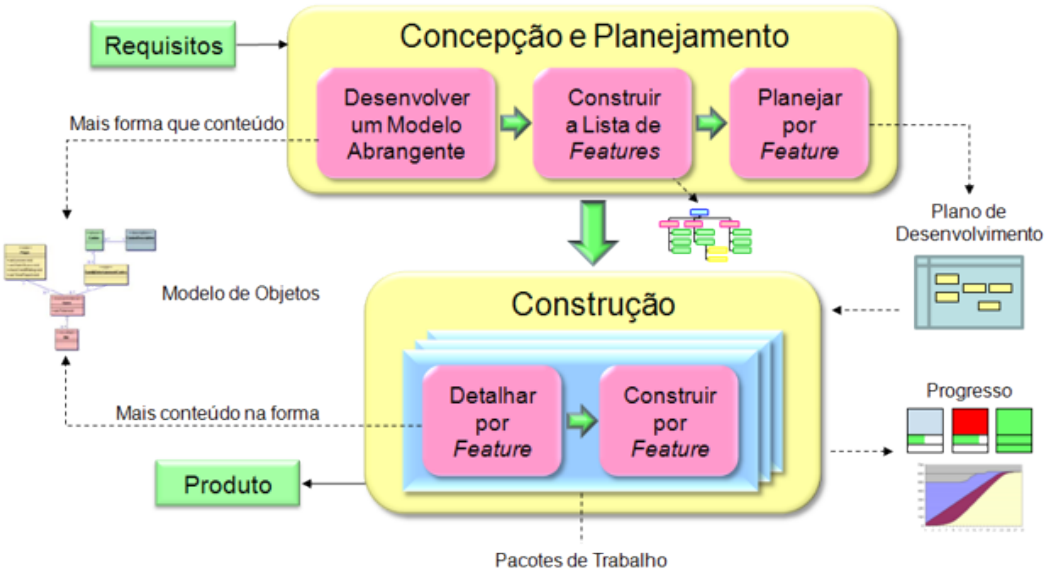



Figura A.11: Etapas de um processo usando a metodologia FDD, adaptado de Palmer e Felsing [111], Fonte: http://heptagon.com.br/fdd-estrutura.

#### A.2.5.2.2 (2) Construir a lista de funcionalidades:

Na segunda etapa constrói-se a lista de funcionalidades previstas para todo o sistema (as features), agrupando-as e priorizando-as com a atribuição de pesos em cada uma delas. Essa lista pode ser obtida através da transformação de métodos criados anteriormente

no modelo de domínio, removendo ou adicionando detalhes abstraídos de requisitos. Ela deverá corresponder adequadamente as expectativas do cliente. Um ponto importante é que essas funcionalidades devem ser breves, havendo sempre a preocupação com o prazo estipulado para a entrega, não podendo ultrapassar, em geral, duas semanas – o que pode obrigar a decomposição de uma funcionalidade, caso ela seja muito extensa.

#### A.2.5.2.3 (3) Planejar por funcionalidades:

Com a lista das funcionalidades em mãos, a terceira etapa se caracteriza pelo intenso planejamento — como transformar as ideias abstraídas em funcionalidades reais, com a elaboração de um plano voltado especificamente aos desenvolvedores. Esse plano deve considerar as relações entre as características de cada funcionalidade, a carga de trabalho que a equipe pode suportar e a complexidade delas, bem como a avaliação da tecnologia a ser utilizada para a implementação, em especial a camada de apresentação [92].

#### A.2.5.2.4 (4) Detalhar cada funcionalidade:

Após a terceira etapa, é iniciada a fase de construção do software propriamente dita, abrangendo mais duas etapas. Na etapa de detalhamento, são desenvolvidos os diagramas de sequência, procurando detalhar o máximo a implementação, desta vez levando-se em conta a linguagem de programação que é importante. Agindo assim, o programador-chefe tem condições de refinar bem o modelo a ser codificado. É nesta etapa que o "dono" das classes inicia a escrita da documentação da funcionalidade que está sendo implementada. O cliente junto poderá participar dessa atividade.

#### A.2.5.2.5 (5) Construir por funcionalidades:

Nesta etapa, tem-se início a construção do software, os programadores devem implementar o que foi discutido na etapa anterior (métodos, classes, etc) e criar e executar os testes de unidade.

### A.2.6 Lean software development

A metodologia Lean surgiu como inspiração ao sistema Toyota de produção, que foi uma técnica de produção em massa criada no Japão logo após a Segunda Guerra mundial na fábrica da empresa automobilística Toyota. Nessa época, de acordo com Fadel e Silveira [52], a indústria japonesa possuía uma produtividade muito baixa e sofria com a falta de recursos, época em que os empresários tiveram de se superar para vencer as barreiras que lhes eram impostas.

O Lean Development (LD) é um método baseado na visão da produção Lean da área de manufatura e processos, cujo principal objetivo e tônica dominante é a eliminação de desperdícios e a diminuição drástica dos gastos com a produção [69]. Segundo a metodoogia, sempre que um problema é encontrado no processo de fabricação Toyota, qualquer trabalhador na linha poderia parar o processo e todos os funcionários poderiam ajudar a corrigir o problema antes de recomeçar o ciclo produtivo. Os trabalhadores eram encorajados a encontrar a causa raiz dos problemas e corrigi-los no local. Os trabalhadores de linha também eram incentivados a fazer melhorias em sua estação sem ter que pedir permissão da gerência. Este processo foi chamado de Melhoria Contínua.

Alguns dos benefícios de melhoramentos contínuos são:

- As melhorias são baseadas em pequenas alterações, em vez de mudanças radicais;
- Pequenas melhorias são menos propensas a exigir grande investimento de capital do que as grandes mudanças no processo;
- Ajuda a incentivar os trabalhadores em seu local de trabalho e pode ainda reforçar o trabalho em equipe, melhorando assim a motivação.

Essas são as ideias que foram transportadas para a Engenharia de Software. Essas propostas seguem a mesma linha, onde não há a preocupação em pregar práticas de desenvolvimento ágeis exatamente como as outras metodologias fazem, pois que, seu foco é o ponto de vista estratégico, direcionado mais ao nível gerencial da organização do que ao nível do desenvolvimento em si. Como vemos em Poppendieck e Poppendieck [122], o foco da metodologia Lean é a observação dos problemas da mudança de requisitos sob uma nova perspectiva: as decisões do cliente devem ser adiadas ao máximo, deixando para que elas sejam feitas quando houver um maior conhecimento do assunto. E quando as decisões forem feitas, a implementação deve ser rápida, evitando que as condições externas afetem uma funcionalidade antes dela ser entregue. E adiantam:

> "Por ser uma forma de produção passada para a área de Engenharia de Software, é necessário que ocorra uma adaptação adequada de seus princípios com a preocupação de manter a filosofia para esse outro ambiente de produção."

Esse trabalho [122] foi influenciado pela literatura da engenharia industrial, de forma parcial, sob a alegação de que a técnica de Lean pode melhorar significativamente a produtividade, reduzir o tempo de trabalho, e também, como atestam Hamilton [66] e Middleton [100], reduzir os defeitos por realização de consistentes observações.

Poppendieck e Poppendieck [122] desenvolveram sete princípios que são vistos como o impulso do software produzico sob a ótica de Lean, listados abaixo:

1. Eliminar o desperdício
2. Fortalecer o time
3. Entregas rápidas
4. Otimizar o todo
5. Construir qualidade
6. Adiar decisões
7. Amplificar conhecimento

Abaixo, algumas práticas claramente identificadas em seu trabalho:

- Não entregar antes do prazo previsto, para dar tempo para que as decisões sejam tomadas com o máximo de conhecimento sobre a iteração do momento;
- Satisfação total do cliente;
- Todos do time devem ter um conhecimento amplo do software sendo construído;
- A satisfação do cliente é a maior prioridade;
- Sempre provenha o melhor valor para o dinheiro;
- Sucesso depende na participação ativa do cliente;
- Todo projeto LD é um esforço do time;
- Tudo pode ser mudado;
- Complete, não construa;
- 80% da solução hoje é melhor do que 100% da solução amanhã;
- Minimalismo é essencial;
- As necessidades determinam a tecnologia;
- O crescimento do produto é o crescimento de características, e não de tamanho.

#### A.2.6.1 O Processo

O desenvolvimento de um projeto utilizando a metodologia Lean é feito em três etapas, a saber: o **início**, o **estado de crescimento regular (steady-state)** e a **transição e renovação**.

#### A.2.6.1.1 (1) Início:

Nesta etapa é realizada a análise de valor, o estudo do negócio e a viabilidade do projeto, conforme [69], também a análise de riscos e outras etapas de escopo gerencial inicial.

#### A.2.6.1.2 (2) Estado de crescimento regular (steady-state):

Na etapa de estado de crescimento regular o objetivo é realizar o desenvolvimento do sistema de uma forma iterativa e incremental em períodos fechados (time-boxes) de 90 dias no máximo. Cada iteração realiza um pouco das fases tradicionais de desenvolvimento (análise, projeto, implementação e testes), integrando continuamente o software. A grande diferença em relação aos outros métodos é a utilização em larga escala de templates para aumentar a velocidade de produção e sua reutilização.

#### A.2.6.1.3 (3) Transiçãao e renovação:

Nesta etapa é realizada a transição e a entrega, focando a documentação, treinamento e a evolução do sistema como um todo.

Segundo Fadel e Silveira [52], a metodologia Lean utiliza técnicas de produção arrojada para agendar o trabalho e são dotadas de mecanismos com sinalizações locais, os quais ajudam os outros desenvolvedores a identificarem o trabalho que precisa ser realizado, à semelhança do Kanban. Esta técnica correspondente à entrega de versões refinadas e incrementais do software em intervalos de tempo regulares, controladas por meio de sinalização local. A Figura A.12 ilustra o quadro de cartões de funcionalidades do processo Lean, destacando bem alguma similaridade com Kanban.

As colunas são as etapas do desenvolvimento, onde há um cartão afixado, representando o objetivo daquela determinada iteração. Abaixo do cartão são colocados outros cartões que definem os requisitos a serem implementados para a funcionalidade que está sendo construída. O nivelamento da produção pode ser feito através da quantidade de trabalho que precisa ser executado durante a implementação dos requisitos.

### A.2.7 Kanban

Kanban é um termo de origem japonesa que significa, literalmente, “cartão” ou “sinalização”. É um conceito relacionado com a utilização de cartões do tipo *post-it* e outros, para indicar o andamento dos fluxos de produção em empresas de fabricação em série.

Segundo Bolaji e Ademi [20], o Kanban:

> "[...] é cada vez mais a prática do Lean evidente em desenvolvimento de software. Kanban foi introduzido para o desenvolvimento de software por Anderson [6], que

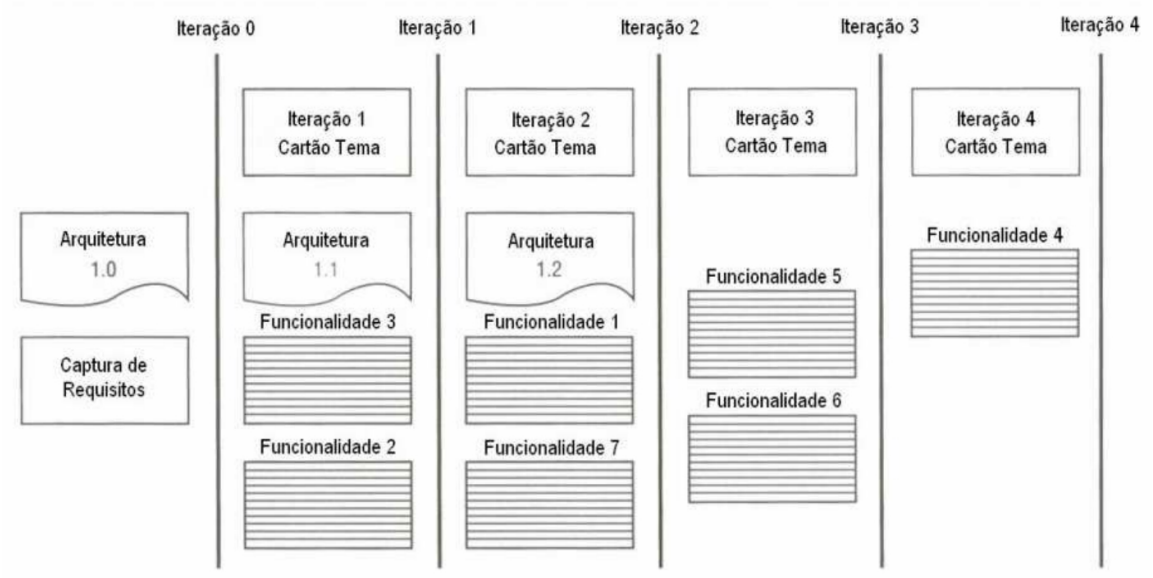



Figura A.12: Quadro de funcionalidades do processo Lean, conforme Poppendieck e Poppendieck [121]: Similaridade com Kanban.

> queria proteger a sua equipe de afluxo de tarefas de gestão para alcançar contínuo ritmo de desenvolvimento e também para ampliar a agilidade com pouca ou nenhuma resistência à mudança. Aplicando Kanban ao seu processo de desenvolvimento de software ajudou na obtenção de todos estes objetivos. Kanban, subsequentemente, atraiu muitas atenções tanto de pesquisadores quanto de profissionais da área."

O método Kanban foi inicialmente aplicado em empresas japonesas de fabricação em série e está estreitamente ligado ao conceito de *just in time*. A empresa japonesa de automóveis Toyota foi a responsável pela introdução desse método devido a necessidade de manter um eficaz funcionamento do sistema de produção em série. Em setembro de 2006, o então diretor sênior de engenharia da Corbis, empresa fundada por Bill Gates, David J. Anderson, seu trabalho pode ser visto em [6], decide projetar um sistema Kanban que substituiria a então abordagem existente para atualização de aplicativos. Os resultados preliminares do uso do Kanban na Corbis foi apresentado nas conferências "Lean New Product Development" e "Agile 2007". Desde então, o Kanban vem ganhando respeito na comunidade de desenvolvimento de software e mais empresas passaram a adotá-lo.

Segundo Miltenburg e Wijngaard [101], em termos de benefícios, em análise de estudos primários sobre a metodologia, o potencial do Kanban no desenvolvimento de software é consistente, embora alguns estudos primários também revelem que Kanban impeça ou reduza o excesso de produção.

### A.2.7.1 O Processo

A Figura A.13 ilustra os processos de desenvolvimento de software utilizando um quadro de Kanban. Os processos de desenvolvimento de software são guiados com a utilização desse quadro que é dividido em colunas, de tal modo que cada coluna representa uma fase separada no processo de desenvolvimento.

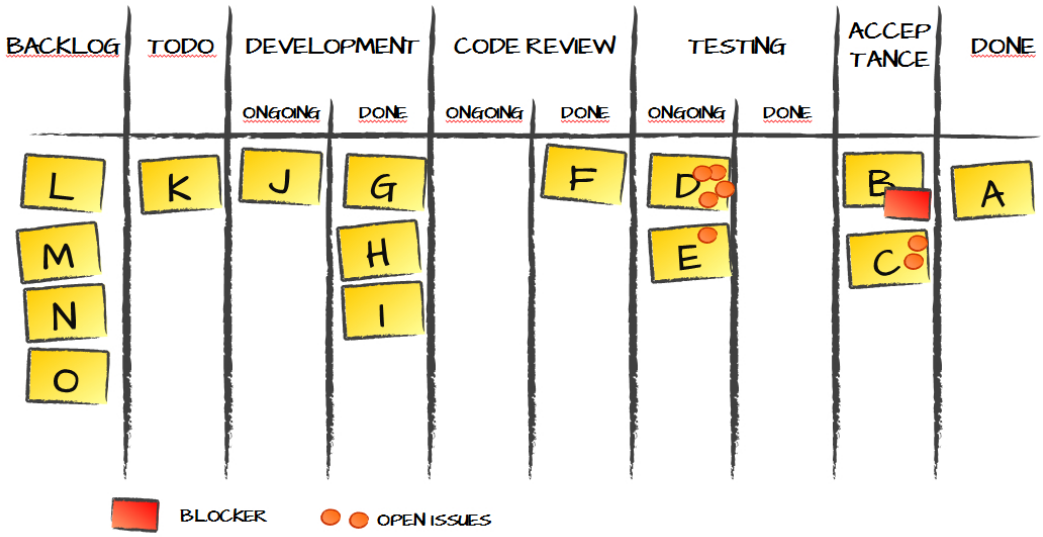



Figura A.13: Exemplo de quadro Kanban, conforme Brodzinski [25].

A técnica ajuda a assimilar e controlar o progresso das tarefas de um projeto de forma visual. Para tanto, utiliza-se um quadro branco (mas pode ser um software) com alguns pequenos papéis coloridos e colados sobre ele que representam as tarefas, ao termino de cada tarefa o papel é puxado para a etapa seguinte até que a mesma seja finalizada. Ao olhar para um quadro Kanban é fácil enxergar como o trabalho de cada membro do time flui, permitindo não só comunicar o status, mas também dar e receber *feedbacks*, conseguindo levar informações que, normalmente, seriam escritas de maneira lenta e pouco objetiva.

Se tomamos um desenvolvedor como exemplo, o processo poderia ter a aparência da Figura A.13. Uma vez terminada a codificação de um novo item de trabalho, o desenvolvedor manteria o olhar sobre o quadro, pois pode acontecer de o gerente bloquear um produto por algum motivo técnico, ou um bloqueador poderia estar esperando por respostas do cliente, ou algo atrelado à tecnologia, etc. Uma breve conversa com a equipe poderia revelar a natureza desses bloqueadores. Ao mesmo tempo, poderia ser um bom momento para lembrar o cliente sobre os bloqueadores, já que eles poderiam impactar na entrega.

Não obstante, segundo Brodzinski [25], a técnica é bastante simples, o suficiente para escrever uma orientação a cada um dos participantes:

> "Sempre que você terminou o trabalho em um item, primeiro verifique se existem bloqueadores. Se houver, tente removê-los, se não, olhe para todos os itens mais próximos à coluna *feito*, normalmente a parte mais à direita do quadro. Afixe ali seu papel (tarefa). Inicie um novo item apenas quando não há literalmente nada que você possa fazer com itens em curso."

Segundo Anderson [6], o Kanban possui apenas cinco princípios:

1. Visualizar o fluxo de trabalho (Workflow);
2. Limitar a quantidade de trabalho em andamento (WIP);
3. Gerenciar e medir o fluxo;
4. Tornar as políticas do processo explícitas;
5. Usar modelos para reconhecer oportunidades de melhoria.

#### A.2.7.2 Práticas comumente associadas

A metodologia Kanban pode ser aplicada em conjunto com qualquer outra metodologia ágil e, portanto, herdando suas práticas, mas é possível identificar algumas que mais se evidenciam:

- Estórias de usuários;
- Uso de templates;
- Tempo de ciclo curto;
- Gestão de mudanças;
- Processo simplificado;
- Visibilidade;
- Redução de desperdício;
- Redução de custo;
- Eliminação de atividades que não agregam valor para a equipe;
- Motivação.

### A.2.8 Extreme Programming (XP)

O Extreme Programming (XP) surgiu como uma tentativa para solucionar os problemas causados pelos ciclos de desenvolvimento longos implementados pelas metodologias mais tradicionais. Foi resultante da experiência realizada no Projeto C3 Payroll na empresa Chrysler americana que, segundo Fowler [55]...

> "[...] era o nome abreviado do projeto Chrysler Comprehensive Compensation, um projeto da folha de pagamento na Chrysler que desde então tornou-se famoso por dar origem ao Extreme Programming."

Esse projeto, que fracassou várias vezes utilizando metodologias mais tradicionais, foi uma tentativa de substituir os diversos sistemas legados COBOL na folha de pagamento da empresa. O projeto começou com um trabalho de desenvolvimento em Smalltalk, em 1995, mas não foi capaz de chegar a um estado estável e foi reiniciado sob a liderança do Kent Beck em 1996, que foi o primeiro a usar uma metodologia que reunia diversas práticas que, mais tarde, ficou conhecida como Extreme Programming. Iniciava-se ali a consagração do termo "práticas ágeis" que já pertencia ao jargão da comunidade, mas que desta vez despontava com bastante ênfase em todo o mundo, nos jornais e congressos da Engenharia de Software. Após o sucesso nesse projeto, XP começou a despontar no meio acadêmico e empresarial e se tornou alvo de inúmeras citações positivas, além de ser adotado em diversas empresas de software do mundo inteiro e apoiado por grandes nomes da orientação a objeto como Ron Jeffries, Martin Fowler, Grady Booch, entre outros.

#### A.2.8.1 O Processo

(1) PAPÉIS E RESPONSABILIDADES

Existem diferentes papéis sugeridos pelo XP ao longo de um projeto que leva o nome de sua metodologia. De acordo com Beck [15], esses papéis assumem as seguintes formas:

- **Programador:** São os responsáveis por escrever os programas e os testes unitários, mantendo o software o mais simples e conciso quanto possível.
- **Cliente:** Escreve as estórias e sugere os testes funcionais, além de decidir se o requisito foi ou não satisfeito. O cliente também define a prioridade de implementação de cada requisito.
- **Testador:** Auxilia o cliente a escrever os testes funcionais, além de executá-los na presença do cliente. É o responsável por manter todos os testes atualizados.
- **Monitor:** O feedback do projeto é de sua responsabilidade. Tem a responsabilidade de acompanhar a conformidade das estimativas feitas pela equipe de desenvolvimento

(por exemplo, estimativas de esforço) e fornece comentários de quanto acuradas elas estão, para poder melhorar futuras estimativas. Ele também acompanha o progresso de cada iteração e avalia se o objetivo é viável dentro das limitações de tempo e recursos, ou se alguma mudança é necessária no processo.

- **Treinador:** É o responsável por todo o processo de desenvolvimento. Deve ser dotado de conhecimento aprofundado da metodologia XP, pois é o membro da equipe que tem a responsabilidade de guiar os demais envolvidos no projeto a executar o processo de forma adequada.
- **Consultor:** Membro externo com conhecimento técnico específico necessário para o projeto, normalmente sobre tecnologia. Deve estar sempre em condições de sanar dúvidas dos desenvolvedores.
- **Chefe:** Responsável pelas tomadas de decisões e por conduzir o direcionamento dos trabalhos, gerenciamento de conflitos e comunicação com as partes envolvidas. Deve envidar todo o esforço a fim de diagnosticar problemas e ter em mãos situações capazes de solucioná-los.

(2) VALORES

O XP é composto essencialmente por cinco valores, alguns princípios e várias práticas. Os valores representam a essência do que gostamos ou não a respeito de algo. Cuidar para que os valores sejam explícitos é importante porque, sem eles, as práticas rapidamente perdem o sentido e tornam-se atividades sem sentido ou direção. Os valores trazem propósito às práticas. Conforme as próprias palavras de Beck [15], os cinco valores são:

1. **Comunicação:** Muitos dos problemas que ocorrem no decorrer do projeto podem ser relacionados com problemas de comunicação entre a equipe ou entre a equipe do projeto e o próprio cliente. Uma pessoa pode deixar de comunicar um fato importante à outra pessoa, um programador pode deixar de levantar uma questão importante ao cliente etc. O XP mantém o fluxo de comunicação através de algumas práticas que não podem ser realizadas sem comunicação. Exemplos dessas práticas são: testes de unidade, programação em pares e estimativa do esforço de cada tarefa.
2. **Simplicidade:** Deve-se sempre selecionar a alternativa mais simples que possa funcionar. O XP se baseia no fato de que é bem mais barato fazer algo que possua simplicidade, no início, e depois alterá-lo conforme as necessidades forem surgindo. Tentar prever as necessidades futuras com alta complexidade não é considerado uma boa prática, pois tais necessidades podem nem vir a ser implementadas, portanto completamente desnecessário no estágio atual considerado.

3. **Feedback:** Todo problema é evidenciado o mais cedo possível para que possa ser corrigido com a maior brevidade. Toda a oportunidade de melhoria do projeto tem de ser descoberta o mais cedo possível para que se possa ser incorporada de forma mais rápida ao produto que está sendo construído.

4. **Coragem:** É preciso coragem para apontar um problema no projeto, para solicitar ajuda quando necessário, para simplificar o código que já está funcionando (podendo introduzir novos defeitos), comunicar ao cliente que não será possível implementar um requisito no prazo estimado e, até mesmo, para fazer alterações no processo de desenvolvimento.

5. **Respeito** [1] Todos têm sua importância dentro da equipe e devem ser respeitados e valorizados, o que mantém o trabalho energizado.

(3) PRINCÍPIOS

Já os princípios, são direcionados a domínios específicos e foram criados para servir de ponte entre os valores e as práticas, Figura A.14.


Figura A.14: Valores, princípios e práticas do XP: os princípios servem como "ponte" entre os valores e as práticas, conforme Williams [161].

Segundo Fowler [53], os princípios não são tão conhecidos ou debatidos, pois não se têm falado muito neles, até mesmo por Kent Beck. Foram discutidos os valores e as práticas tão debatidos nos estágios de formação de profissionais de XP que os princípios ficaram um pouco ofuscados. Na verdade, um dos maiores problemas com o XP e com qualquer método ágil, é a aplicação correta da receita de como se faz para realizar as adaptações necessárias para a adoção de XP, adequando as práticas às condições locais. Os princípios ajudam a fornecer algumas orientações sobre quais pedaços de adaptação irão funcionar melhor e quais deles irão contra os valores pregados pelo XP.

Os Valores são abstratos demais para guiar práticas em contextos específicos, e os princípios surgem para assumir formas de comunicação entre eles. Inicialmente, Kent Beck propôs dez princípios que nortearam as práticas dos trabalhos, a saber:

[1] Este valor foi incorporado pelo advento do XP2, que foi uma melhoria do método.

- Passos de bebê;
- Fluxo;
- Oportunidade;
- Melhoria;
- Reflexão;
- Benefício mútuo;
- Responsabilidade aceita;
- Falha;
- Redundância;
- Qualidade.

(4) PRÁTICAS

As práticas podem ser consideradas como um conjunto de atividades que as equipes têm como base para a evolução do software em construção. Consagradas, tem-se uma confiança muito grande nelas, onde os pontos fracos de uma se compensa nos pontos fortes de outra. Se aplicadas em conjunto, o desenvolvimento de software funciona melhor.

As práticas do XP ganharam tanta notoriedade que a maioria delas se incorporaram a outras metodologias que já existiam na época de sua criação e esse comportamento perdura até os dias de hoje. Kent Beck [15], o autor de "Extreme Programming Explained", definiu 12 práticas de Extreme Programming como se segue:

- Jogo do Planejamento;
- Releases curtas;
- Metáfora;
- Projeto simples;
- Teste;
- Refatoração;
- Programação em pares;
- Propriedade coletiva;
- Integração contínua;
- 40 horas semanais;
- Cliente junto;

- Padrões de codificação.

Com a reformulação da metodologia, no ano de 2004, por Andres e Beck [7], essas práticas foram extendidas ganhando uma nova roupagem. As práticas de XP2, comumente chamadas "Evolucionárias", agora foram divididas em duas grandes categorias, a saber:

**Práticas Primarias:** São práticas que podem ser adotadas imediatamente de forma segura para melhorar o esforço de desenvolvimento de software. Qual delas adotar primeiro dependerá inteiramente do ambiente e das oportunidades de melhoria. Algumas pessoas precisam de planejamento porque não sabem o que precisa ser feito. Outros precisam de práticas relacionadas à melhoria de qualidade porque estão criando defeitos demais para serem capazes de ver o que está acontecendo.

**Práticas Corolárias:** Segundo Andres e Beck [7], As práticas corolárias são perigosas de serem implementadas antes de se adotar as práticas primárias:

> "Se você começar a implantar o software diariamente, por exemplo, sem baixar a taxa de defeitos para algo muito próximo de zero (com programação em par, integração contínua e desenvolvimento orientado a testes), você terá um desastre nas mãos."

A Tabela A.3 mostra as novas práticas de XP, as primárias e a Tabela A.4 as corolárias e a Figura A.15 mostra como essas práticas estão integradas de uma forma gráfica, na metodologia XP.

Tabela A.3: Práticas primárias de XP2.

| # | **Práticas** |
|---|---|
| 1 | Sentar junto |
| 2 | Time completo |
| 3 | Área de trabalho informativa |
| 4 | Trabalho energizado |
| 5 | Programação em pares |
| 6 | Estórias de usuários |
| 7 | Ciclo semanal |
| 8 | Ciclo de estação |
| 9 | Folga |
| 10 | Build de 10 minutos |
| 11 | Integração contínua |
| 12 | Teste primeiro |
| 13 | Design incremental |

Tabela A.4: Práticas corolárias de XP2.

| # | **Práticas** |
|---|---|
| 1 | Análise de causa inicial |
| 2 | Repositório único de código |
| 3 | Código compartilhado |
| 4 | Código e testes |
| 5 | Continuidade da equipe |
| 6 | Contrato de escopo negociável |
| 7 | Envolvimento real do cliente |
| 8 | Esquipes que encolhem |
| 9 | Implantação diária |
| 10 | Implantação incremental |
| 11 | Pagar por uso |

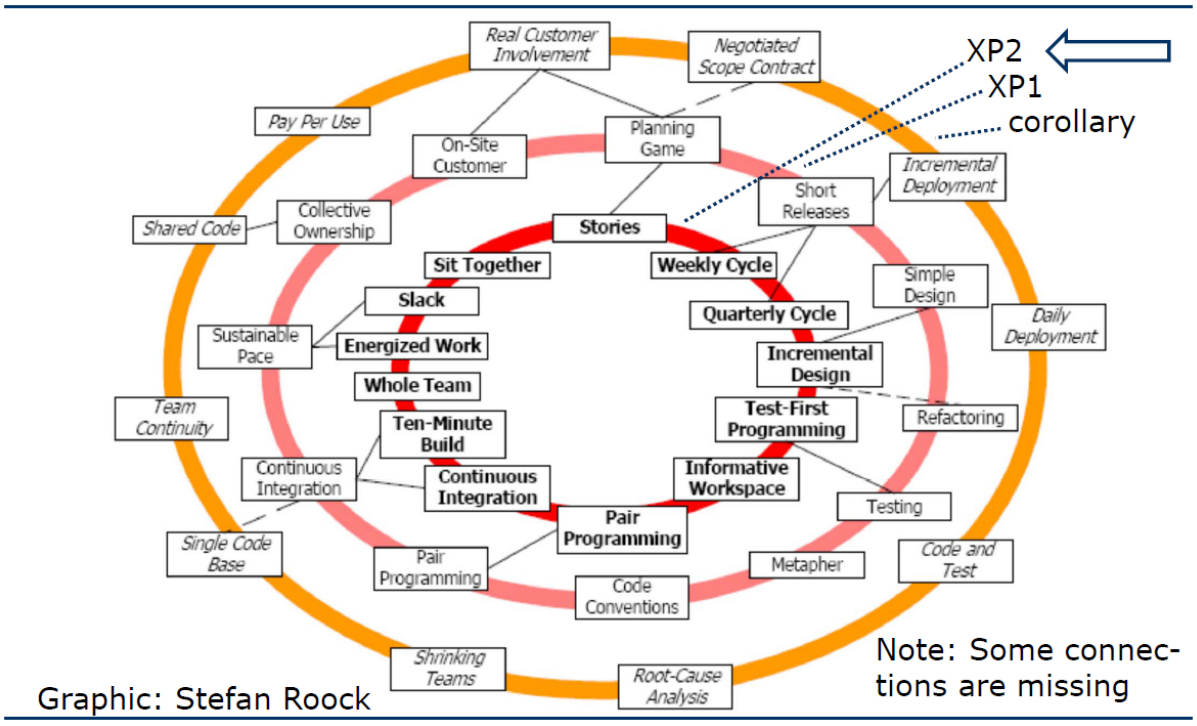



Figura A.15: Integração das novas (XP2) e consagradas (XP1) práticas XP, conforme Roock [128].

A Figura A.16 nos mostra as interligações entre valores, princípios e práticas do XP.

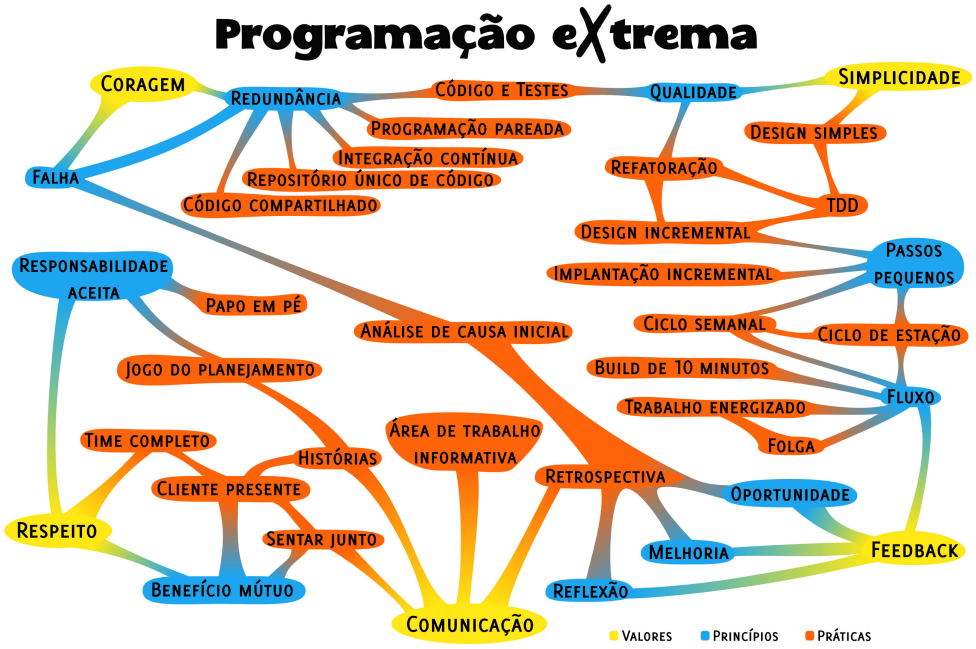



Figura A.16: Valores, princípios e práticas do XP conforme Bravo [23].

### A.2.9 Scrum

O Scrum é um processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software que é mais utilizado para trabalhos complexos nos quais é impossível predizer tudo o que irá ocorrer.

Conforme atestam Schwaber e Sutherland [133], o Scrum é "leve, simples de entender e extremamente difícil de se dominar", e afirmam:

> "Scrum é um framework estrutural que está sendo utilizado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990. O Scrum não é um processo ou uma técnica para construir produtos, em vez disto, é um framework dentro do qual você pode empregar vários processos ou técnicas. O Scrum deixa claro a eficácia relativa das práticas de gerenciamento e desenvolvimento de produtos, de modo que você possa melhorá-las."

#### A.2.9.1 O Processo

(1) PAPÉIS E RESPONSABILIDADES

São 3 os papéis principais de um projeto envolvendo Scrum: o Product Owner, o Scrum Team, e o Scrum Master:

**Product Owner:**

- Definir os requisitos do produto, decidir a data de release e o que deve conter nela;

- Retorno financeiro (ROI) do produto;
- Priorizar os requisitos de acordo com o seu valor de mercado. Pode mudar os requisitos e prioridades a cada Sprint.
- Aceitar ou rejeitar o resultado de cada Sprint.

**Scrum Master:**

- Garantir que o time esteja totalmente funcional e produtivo;
- Facilitar a colaboração entre as funções e áreas e eliminar os impedimentos do time;
- Proteger o time de interferências externas;
- Garantir a fluidez do projeto, participando das reuniões diárias, revisão da Sprint, e planejamento.

**Scrum Team:**

- Selecionar, entre os itens priorizados, os que irão ser executados durante a Sprint. Tem todo o direito de realizar o que quiser dentro da Sprint, com coerência. Equipe formada por 5 a 9 membros.

(2) COMO FUNCIONA

A metodologia divide os projetos em releases e sprints curtos. Cada sprint ocupa apenas a funcionalidade necessária e suficiente que é priorizada pelo cliente e que será entregue no tempo certo, de modo que no final de cada Sprint tem-se um produto de trabalho que pode ser liberado.

Os desenvolvedores e os testadores também participam da história escrita do usuário, que são os itens do backlog. Isso dá uma visão inicial do comportamento do produto para todos na equipe e também ajuda a chegar aos critérios de aceitação no início do próprio sprint.

#### A.2.9.1.1 Sprints:

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de Sprints. O Sprint representa um tempo definido dentro do qual um conjunto de atividades deve ser executado. Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que no Scrum são chamadas de Sprints e geralmente duram de 2 a 4 semanas. O trabalho a ser realizado na Sprint é planejado na reunião de planejamento da Sprint. Este plano é criado com o trabalho colaborativo de todo o Time Scrum. Reunião de planejamento da Sprint possui um time-box com no máximo

oito horas para uma Sprint de um mês de duração. Para Sprints menores, este evento é usualmente menor. O Scrum Master garante que o evento ocorra e que os participantes entendam seu propósito. O Scrum Master ensina o Time Scrum a manter-se dentro dos limites do time-box

#### A.2.9.1.2 Product Backlog:

As funcionalidades a serem implementadas no projeto são mantidas em uma lista conhecida como Product Backlog. No início de cada Sprint, faz-se um Sprint Planning Meeting (uma reunião de planejamento), na qual o Product Owner (quem representa os envolvidos) prioriza todos os itens do Product Backlog e a equipe seleciona as funcionalidades que ela será capaz de implementar durante o Sprint que se inicia. As funcionalidades atribuídas a um Sprint são transferidas do Product Backlog ao Sprint Backlog. O Backlog do Produto é dinâmico, mudando constantemente para identificar o que o produto necessita para ser mais apropriado, competitivo e útil. A Figura A.17 ilustra o processo.

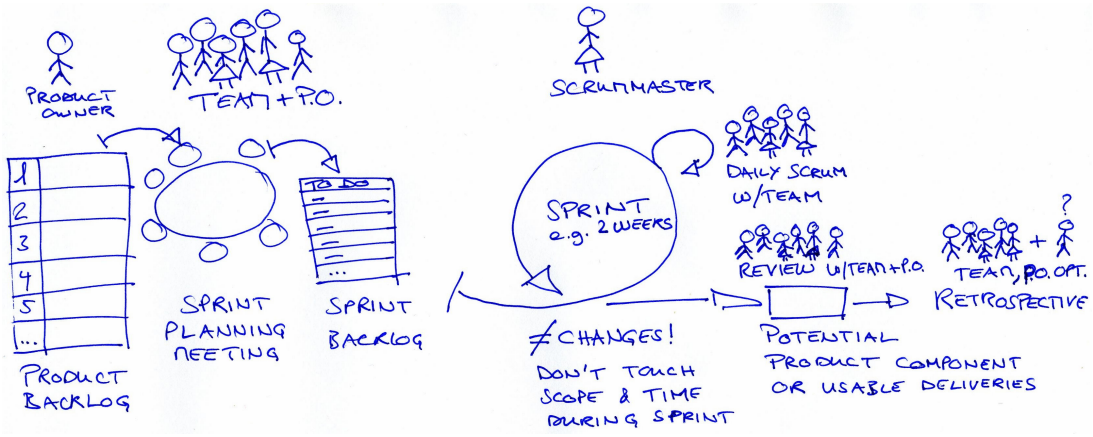



Figura A.17: No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de Sprints [115].

(3) PRÁTICAS

As práticas ágeis utilizadas na metodologia Scrum não estão textualmente declaradas no Guia Definitivo de Schwaber e Sutherland [133], mas entendemos que elas estão presentes pela evidência dos trabalhos realizados no dia-a-dia. Podemos ainda notar que a maioria das práticas utilizadas pelas outras metodologias, normalmente, também fazem parte do rol de práticas do Scrum de uma maneira complementar, algo que está descrito no guia de uma forma diferente como uma prática a ser adotada, mas sem uma especificação clara de como fazer. Um bom exemplo disto é o Quadro Kanban, que antes de ser uma prática, é também uma metodologia, e quase que obrigatória em Scrum.

Abaixo, as principais práticas de Scrum, específicas de Scrum, lembrando que a metodologia utiliza várias práticas de outras metodologias, realizando as adaptações que cada projeto requer.

1. Quadro de Tarefas (Kanban);
2. Gráfico Burndown;
3. Estimativas
4. Planning Poker;
5. Plano de mitigação de riscos;
6. Entrega de 2 a 4 semanas.

# Apêndice B

# Práticas Ágeis de Desenvolvimento de Software

Este apêndice descreve as principais práticas ágeis de desenvolvimento de software, sem contudo se preocupar a quais metodologias elas possam pertencer, focando, também, a evidência delas em referências bibliográficas de pesquisas empíricas e agrupando-as em cinco categorias por afinidades.

## B.1 Agrupamento das Práticas Ágeis

As práticas sugeridas para comporem este trabalho foram agrupadas por similaridade, independente se estivessem ou não configuradas como valores ou princípios de determinadas metodologias. As práticas foram agrupadas, em um momento inicial, com 60 itens. Ao rodar o piloto de *survey* em pesquisa de campo, chegou-se a conclusão de que algumas práticas estariam redundantes entre si por significarem a mesma coisa, mas em metodologias diferentes. As redundâncias [1] foram eliminadas e a tabela ficou com a seguinte configuração:

GERENCIAMENTO: 1 - Ampliar o conhecimento, 2 - Comunicação, 3 - Design incremental, 4 - Eliminação de desperdícios, 5 - Gráfico Burndown, 6 - Implantação diária, 7 - Integração contínua, 8 - Iterativo e incremental, 9 - Jogo do planejamento, 10 - Minimalismo, 11 - MosCow, 12 - Pagar por uso, 13 - Plano de mitigação de riscos, 14 - Quadro de tarefas, 15 - visibilidade, 16 - Discussão sobre o quadro branco (lousa).

TIME: 1 - Time com poder de decisão, 2 - Time completo 3 - Sentar junto, 4 - Reunião em pé, 5 - Área de trabalho informativa, 6 - Equipes que encolhem, 7 - Continuidade da equipe.

DESENVOLVIMENTO: 1 - Aprender fazendo, 2 - Cliente junto, 3 - Desenvolvimento orientado por características das funcionalidades, 4 - Estórias de usuários, 5 - Modelagem em objetos de domínio, 6 - Programação em pares, 7 - Propriedade coletiva, 8 - Prototipação, 9 - Refatoração.

TESTE: 1 - Teste primeiro (TDD), 2 - Testes integrados.

RELEASE: 1 - Retrospectiva ao término do ciclo, 2 - Entrega de 2 a 4 semanas, 3 - 40 horas semanais, 4 - Adiamento de decisões ao máximo, 5 - Estimativas 6 - Satisfação total do cliente.

[1] Os detalhes dessa abordagem estão retratados na seção que trata das ameaças à validade desta pesquisa.

## B.2 Práticas de Gerenciamento

### B.2.1 Ampliar o Conhecimento

Segundo Poppendieck e Cusumano [120], o conhecimento não deve ficar restrito à apenas um pequeno time de desenvolvedores. Ele pode sim, nascer ali em um pequeno time, e imediatamente ser propagado para todo o universo, para que o próprio conhecimento se melhore ao longo do tempo. Segundo eles, os métodos e suas práticas ágeis geralmente esperam que o projeto ocorra fora da equipe de desenvolvimento ou que ocorra em incrementos muito pequenos dentro da equipe. É por causa disto que as práticas ágeis, revelam-se insuficientes para resolver determinados problemas de desenvolvimento. Assim sendo, a técnica de ampliar o conhecimento se resume em disseminar os conhecimentos adquiridos, por meio de cursos e treinamentos ou por meio de publicação, desenvolvendo o conhecimento útil em suas áreas de especialização, e aumentando o número de pessoas que aplicam esse conhecimento, melhorando a maneira de pensar sobre um determinado assunto.

### B.2.2 Comunicação

A prática de comunicação entre os membros de uma equipe de desenvolvimento, segundo Betteke [158], consiste em integrar todos os membros do time por meio de conversas e/ou comunicação formal a fim de deixar cada participante ciente de todas as atividades que ocorrem a cada momento. Segundo ele, as simples questões relativas ao planejamento, ainda que detalhadas, não são suficientes para se atingir o sucesso e, pior ainda, que esse modelo fornece uma ilusão indesejável sobre o controle. É por isso que essa abordagem não pode ser mais considerada como adequada em uma sociedade digitalizada, que deve ser muito mais complexa do que a noção enferrujada de comunicação bidirecional apenas com públicos relevantes. Finaliza mostrando que a comunicação deve ser feita de forma disseminada, com todos os atores envolvidos no processo, em reuniões com o pessoal.

### B.2.3 Design Incremental

Conforme visto em Shore et al. [137], na metodologia de Programação Extrema em sua segunda versão (XP2), a cada semana, exige-se de seus programadores a conclusão de quatro a dez estórias de usuários, o que pode se tornar um desafio. Em outras palavras, os programadores devem ser capazes de produzir um código que agregue valor ao cliente, começando do zero a partir da primeira semana. segundo eles, isto só pode ser conseguido

se a codificação for executada em pequenos passos (passos de bebê), e a cada ciclo incrementando o design.

A solução para o dilema vem com a prática "Design Incremental", também chamada de "Design Evolutivo", que permite a construção de infra-estrutura técnica, como modelos de domínio e estruturas de persistência, de forma incremental, em pequenos pedaços, à medida que divulga as estórias. Não importa o nível de design que o desenvolvedor vê em determinado momento, o design tende a melhorar aos poucos. Normalmente, ele implementará o código no projeto existente por vários ciclos, fazendo pequenas alterações à medida que for avançando. Em seguida, novas ideias vão se incorporando para uma nova abordagem de design, exigindo uma série de refatorações para apoiá-la.

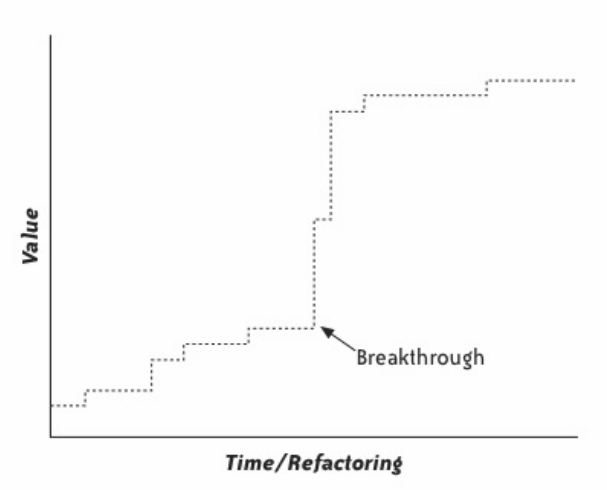



Figura B.1: Segundo Evans [51], os avanços podem ser obtidos por meio de pequenos passos, de forma incremental.

Evans [51] chama isso de um avanço (ver Figura B.1), mostrando que as descobertas acontecem em todos os níveis do projeto e que os avanços são o resultado de *insights* importantes que irão agregar melhorias substanciais ao design.

### B.2.4 Eliminação de Desperdícios

Os desperdícios podem ser considerados como a consequência direta de implementação de modos de operar ineficientes ou mal planejados. Os primeiros a realizar um mapeamento completo em uma linha de produção foram os japoneses e a ferramenta que mais aplica esses conceitos é a Lean Manufacturing. Os desperdícios podem assumir várias formas, podendo ser encontrados ao longo do processamento de um produto qualquer, seja em entradas ou em saídas desnecessárias, onerando o custo final do produto. Podem ainda ser observados na forma de material, estoque, equipamento, infraestrutura, utilidades,

documentos e até movimentos dentre outras atividades que simplesmente não agregam valor algum. Segundo Ohno [107], considerado o criador do Sistema Toyota de Produção e o pai do sistema Kanban, os desperdícios podem ser agrupados em sete categorias, como evidenciado na Figura B.2.

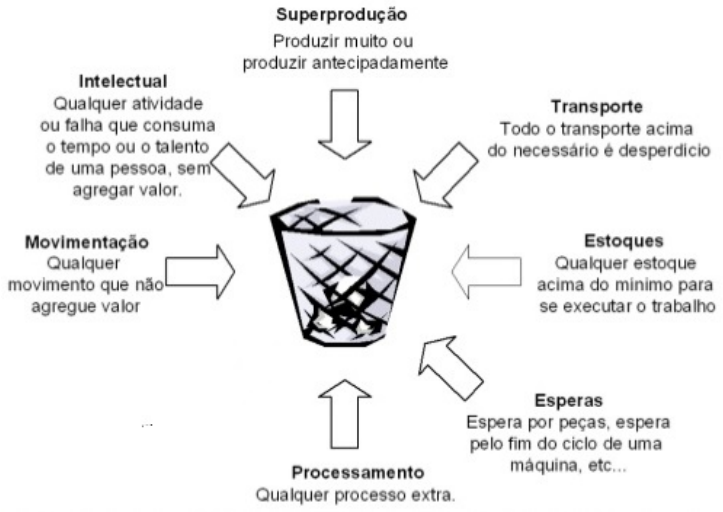



Figura B.2: os sete desperdícios segundo Ohno [107], adaptado de https:// pt.slideshare.net/ LuizFelipeCherem/ aula-pcp-lean-parte-ii-unoesc-so-miguel-do-oeste (acessado em 10 abril 2017).

Esse mapeamento em sete tipologias distintas se baseia nos diferentes tipos de resíduos inerentes ao fluxo de valor. A utilização das ferramentas que combatem um problema específico, ou a combinação de várias delas é, por conseguinte, impulsionada por esses tipos de resíduos a serem removidos.

### B.2.5 Gráfico Burndown

Segundo Permana et al. [116], o gráfico de Burndown é um gráfico que mostra quanto tempo um desenvolvedor leva para concluir uma tarefa naquele dia, em relação ao tempo total do projeto, ou seja, após cada dia o gráfico apresenta a porção de trabalho finalizada pelo desenvolvedor em comparação com o trabalho total planejado. Na Figura B.3 existe um esforço pendente e uma estimativa do esforço, e nele pode-se observar que a média do esforço pendente está abaixo do esforço estimado, o que significa que o progresso de trabalho é mais rápido que a estimativa e que aquele desenvolvedor está com alguma folga.

### B.2.6 Implantação Diária

Frequentemente citada pelos autores como uma das práticas corolárias do XP, isto é, uma afirmação deduzida de uma verdade já demonstrada, segundo Boehm [19], a implantação

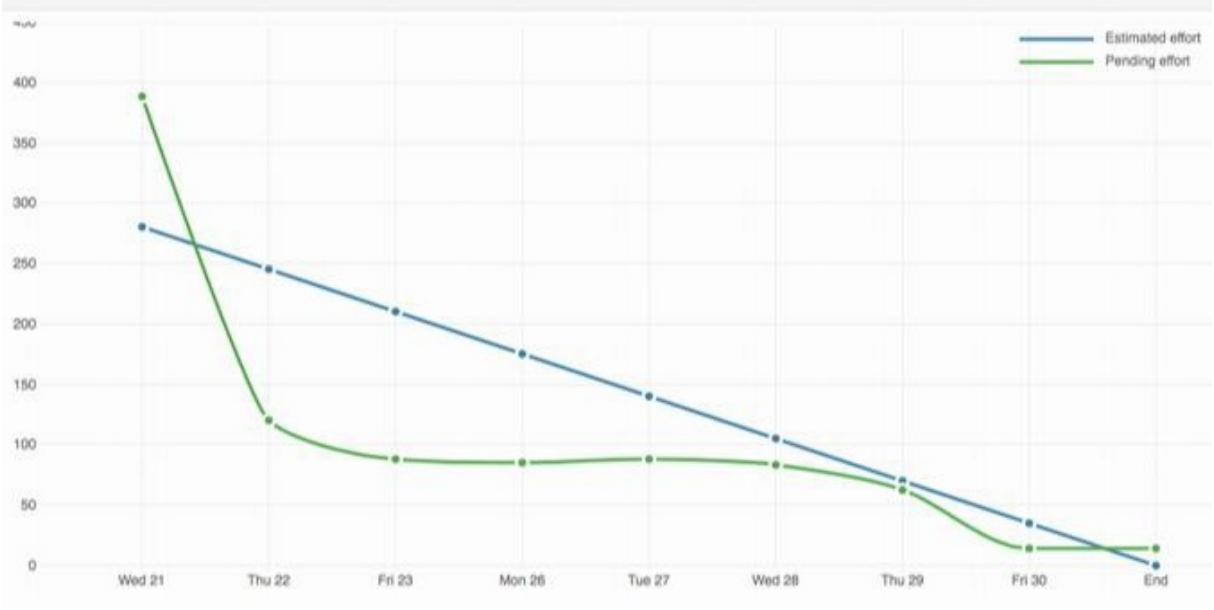



Figura B.3: O Gráfico Burndown segundo Permana et al. [116].

diária é a técnica de colocar um código novo em produção, especificamente a cada noite, durante o ciclo de desenvolvimento do projeto. Ainda conforme Marchesi [95], não se pode admitir uma diferença muito grande de código em relação ao que está em produção com o que está dentro da máquina do desenvolvedor, que isto pode ser considerado uma má prática por oferecer um risco para o projeto.

### B.2.7 Integração Contínua

Integração contínua, conforme afirma Fowler [57], é uma prática de desenvolvimento de software onde os membros de uma equipe integram o seu trabalho com frequência, pelo menos diariamente — levando a múltiplas integrações por dia. Cada integração é verificada por uma compilação automatizada, incluindo o teste, por meio de softwares de versionamento. Muitas equipes acham que essa abordagem ajuda a diminuir vários problemas, agregando valor mais rapidamente.

### B.2.8 Iterativo e Incremental

Para ilustrar de forma mais clara o que vem a ser o desenvolvimento iterativo e o desenvolvimento incremental (os dois podem ou não ser excludentes), apresenta-se a seguir a clássica exposição de Patton [113], onde ele descreve os dois processos utilizando uma obra de arte, conforme vemos na Figura B.4.

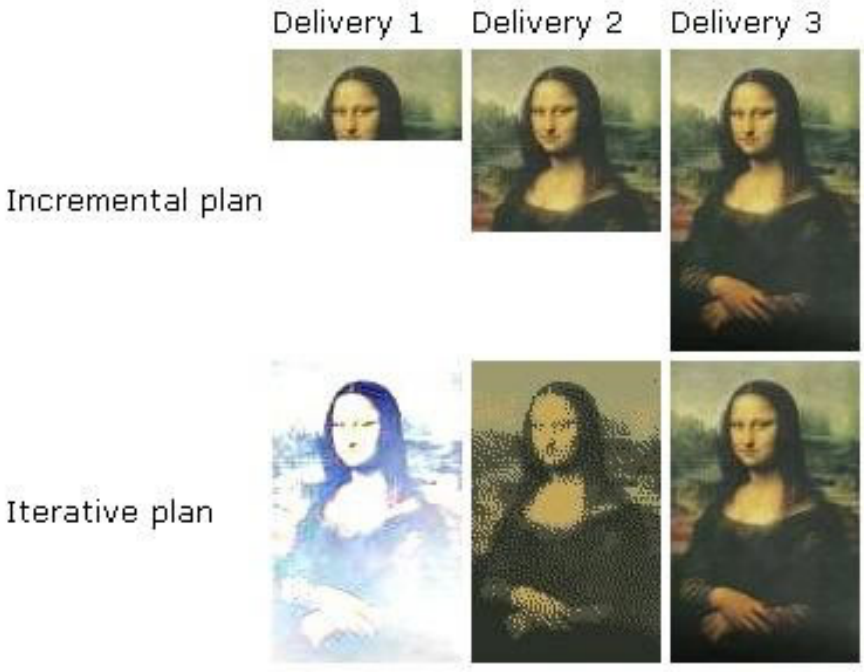



Figura B.4: Comparando os desenvolvimentos Iterativo e Incremental, segundo Patton [113].

Uma das primeiras práticas incompreendidas a se tratar em desenvolvimento ágil é a iteração, diz Patton. *Iterate* não é incremento. *Increment* requer ter uma ideia completa sobre o produto final, como na imagem, já o *Iterate* permite construir uma versão bruta inicial, validá-la, e em seguida, lentamente, construir a qualidade.

A principal diferença entre a técnica iterativa e a incremental é a percepção do cliente. Na incremental, o cliente conhece os mínimos detalhes, desde o início do projeto, e na iterativa esses detalhes vão surgindo ao longo do desenvolvimento, mas já se tem uma visão geral do projeto. As duas podem levar ao resultado final, embora a técnica iterativa possa levar à absorção de mudanças com menor impacto.

### B.2.9 Jogo do Planejamento

Após cada iteração, o cliente senta junto com a equipe e realiza deliberações acerca da nova iteração, Bech [15] chamou isto de Jogo do Planejamento, onde os clientes decidem escopo e calendário de novos lançamentos baseados nas estimativas fornecidas pelos programadores. Os programadores implementam apenas a funcionalidade exigida pela estória na iteração em questão. Bech explica bem isto realizando uma comparação com uma pessoa que vai ao mercado fazer compras com apenas 100 dólares no bolso: dispondo apenas desse valor, ele tem de ficar atento às suas prioridades, decidindo o que comprar. Os preços dos produtos são as estimativas, e o orçamento é calculado por

meio da produção da equipe em termos de estórias estimadas já entregues por unidade de tempo. O cliente pode encher um carrinho inteiro, ele escolhe um conjunto de estórias. No final os programadores calculam o acabamento e a data de entrega. O jogo pode continuar, se o cliente ainda não estiver satisfeito, ou até que se chegue a um contento.

### B.2.10 Minimalismo

Segundo Carroll [26], o minimalismo é uma abordagem de instrução e documentação orientada para a tarefa, que enfatiza a importância de atividades e experiências realistas para a aprendizagem efetiva e a busca de informações. Essa abordagem foi definida por John Carroll em 1990, focando as ideias de aprendizagem desde o início de um projeto e a maneira mais fácil de se fazer alguma coisa na percepção do aprendiz, também aproveitando os erros obtidos como fonte dessa aprendizagem.

### B.2.11 MoSCoW

Conforme vemos em Waters [160], MoSCoW é uma técnica bastante simples de se classificar os recursos (ou estórias de usuários) em ordem de prioridade — uma maneira de ajudar as equipes a entender rapidamente o que é essencial para o lançamento de um produto e o que não é. A origem da palavra pode ser vista na Figura B.5. Por esta técnica, a regra é separar primeiro as estórias de acordo com a prioridade estipulada, sendo que os itens "Tem que ter" (Must have) e "Deveria ter" (Should have) devem ser os primeiros da lista, e o "Poderia ter" (Could have) e o "Gostaria de ter" (Would have) devem ficar para o final dela. O último item da lista é facultativo, podendo ser deixado para uma próxima release.

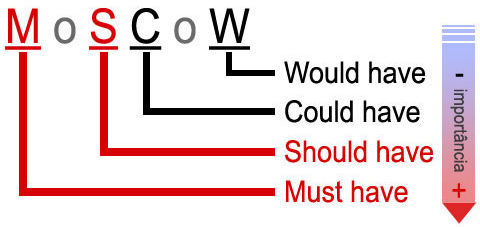



Figura B.5: Boa técnica de priorização, segundo Waters [160], adaptado de http:// www.fabiocruz.com.br/ outros/ sprint-planning-sp1/ (acessado em 10 abril 2017).

Waters [160] nos diz que é uma boa ideia certificar-se de que um projeto tem e deve ter um número saudável de requisitos (ou as estórias de usuários). Isso ajuda a prover o projeto com alguma flexibilidade se as coisas começarem a demorar mais tempo do que o esperado. Assim sendo, se um projeto tiver somente "Must Haves", o escopo não

pode ser variado. Portanto, os custos e os prazos não devem ser realmente fixados sem antes contingenciar ("Wold Have") essas prioridades. Estas são efetivamente tarefas de alongamento, recursos que serão incluídos, se possível, mas a data de lançamento não será movida para acomodá-los se não houver tempo suficiente para concluí-los.

### B.2.12 Pagar por Uso

Pagar por Uso é uma prática corolária do XP, onde o cliente paga para poder utilizar a nova funcionalidade incrementada na release do software em tela. Segundo Marchesi [95], ligar o fluxo de dinheiro diretamente ao desenvolvimento de software fornece informações precisas, antenadas com a melhoria, o cliente paga geralmente por cada liberação do software, mas isto pode criar um conflito entre o fornecedor e o cliente, que desejariam menos lançamentos com mais funcionalidades, ou vice-versa, conforme o ponto de vista.

### B.2.13 Plano de Mitigação de Riscos

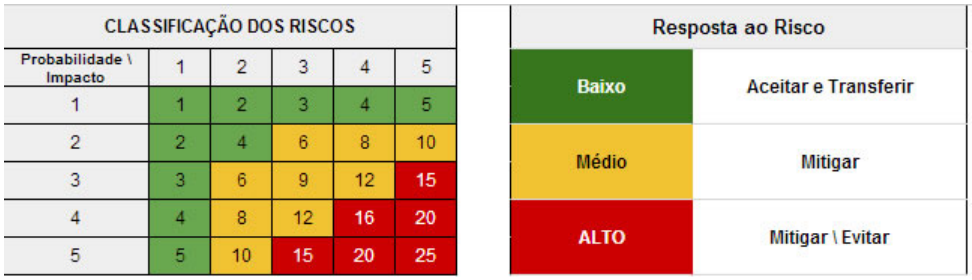
Segundo Kendrick [82]:

> "[...] em projetos, um risco pode ser considerado como qualquer evento indesejável que está associado com o trabalho realizado e é o resultado de dois fatores: os impactos esperados de um evento e a probabilidade de que o evento venha a ocorrer".

Aceitar e Transferir o gerenciamento de riscos aparece com mais relevância no PMBOK [119], mas é frequentemente associado às metodologias ágeis [35], especialmente Scrum, onde problemas podem ocorrer se não houver uma gerência eficaz [126]. Sob esse contexto, um Plano de Mitigação de Riscos se faz necessário para prover ao projeto alternativas para detectar, analisar e corrigir os riscos, previamente estabelecidos em estimativas. Opcionalmente o gerente do projeto confecciona um quadro com os riscos identificados, categorizando-os em três partes: "Aceitar e Transferir", "Mitigar" e "Mitigar/Evitar", conforme vemos na Figura B.6. À medida que os riscos são identificados ao longo do projeto, eles são relacionados no quadro de acordo com o modo de gerenciamento escolhido para os mesmos. É importante que esse quadro esteja visível à todas as partes interessadas, ampliando a visibilidade do projeto.

### B.2.14 Quadro de Tarefas

O quadro de tarefas é praticamente utilizado por todas as metodologias ágeis e algumas vezes não e textualmente citado. O Scrum, por exemplo, o executa largamente. Veja que o quadro de tarefas também pode ser considerado como uma metodologia à parte, também denominada Kanban, conforme atesta Hundermark [74]:



| CLASSIFICAÇÃO DOS RISCOS | | | | | |
|---|---|---|---|---|---|
| Probabilidade \ Impacto | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 2 | 4 | 6 | 8 | 10 |
| 3 | 3 | 6 | 9 | 12 | 15 |
| 4 | 4 | 8 | 12 | 16 | 20 |
| 5 | 5 | 10 | 15 | 20 | 25 |

| Resposta ao Risco | |
|---|---|
| Baixo | Aceitar e Transferir |
| Médio | Mitigar |
| ALTO | Mitigar \ Evitar |

Figura B.6: O PMBOK apresenta algumas técnicas de mitigação de riscos, adaptado de https://lucasblancob.wordpress.com/ (acessado em 11 abril 2017).

> "O quadro de tarefas é um exemplo de um kanban, uma palavra japonesa significando sinal ou visível sinal. Diz a toda a equipe e a qualquer pessoa o trabalho que planejaram ou o Sprint e seu status atual."

Conforme atestam Pikkarainen et al. [118], embora o uso de Scrum e algumas práticas de XP facilite a comunicação organizacional e de equipe das dependências entre os recursos do produto e as tarefas de trabalho, o uso de práticas ágeis requer que a equipe e a organização usem práticas adicionais planejadas para garantir a eficiência da comunicação externa entre todos os atores do desenvolvimento de software. E é assim que nesse contexto surge o quadro de tarefas, muito utilizado para ajudar na coordenação da equipe do projeto, onde o processo é simplificado e utiliza-se bastante as estórias de usuários, templates, visibilidade e até motivações.

A Figura B.7 mostra como um quadro desses pode ser confeccionado, na prática do dia-a-dia.

### B.2.15 Visibilidade

A visibilidade de um projeto pode ser definida como a capacidade que um projeto tem de ser exposto, de forma bem ampla, em relação aos seus processos, atividades e escopo. Isto proporciona uma maior participação de todos os atores envolvidos, e faz com que eles se engajem mais no trabalho, proporcionando transparência, liderança, eficiência e espírito de corpo da equipe. Segundo Turk et al. [156], a visibilidade de um projeto pode ser alcançada por meio de entregas, de incrementos e da exposição de algumas métricas, fazendo reuniões constantes e transmitindo a evolução.

Segundo Tran et al. [154], a visibilidade no início do projeto é menos participativa, principalmente por parte dos patrocinadores, aumentando com as entregas evolutivas. A Figura B.8 permite compreender a diferença entre processos ágeis e processos tradicionais de desenvolvimento, onde a linha contínua representa a visibilidade em projetos ágeis ao longo do tempo de desenvolvimento e a linha pontilhada a visibilidade em projetos


Figura B.7: Quadro de tarefas — tudo à mão-livre, https://www.flickr.com/ photos/ improveit/ 1674682711 (acessado em 11 abril 2017).

com metodologias tradicionais, onde a visibilidade é mais evidente no início e no final do projeto.

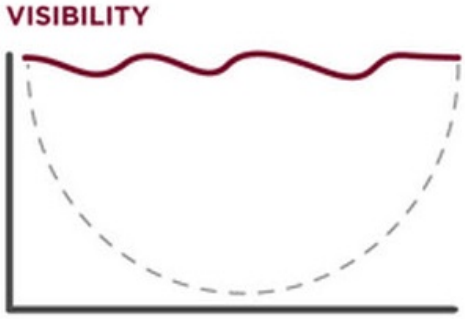



Figura B.8: A prática de visibilidade em métodos ágeis e processos tradicionais, segundo Tran et al. [154].

### B.2.16 Discussão sobre o Quadro Branco (Lousa)

Comunicação, colaboração e coordenação são os principais facilitadores do desenvolvimento de software [102], mas em um ambiente de trabalho, o seu espaço físico é que desempenha o

papel preponderante para o sucesso na comunicação eficaz e na colaboração e coordenação entre as pessoas enquanto desenvolve software. Segundo Mishra et al. [102]:

> "Nossa análise também revela que as barreiras de vidro de meia altura são muito eficazes durante as atividades de resolução de problemas das pessoas enquanto trabalham juntos como uma equipe. De fato, tal ambiente fisicamente aberto parece melhorar a comunicação, coordenação e colaboração."

E ainda ressaltam que "[...] o quadro branco era apagável e era usado para discussão ou elaboração de qualquer estória".

Percebe-se que um importante papel para o exercício dessas atividades é a discussão de determinados assuntos sobre um quadro branco, onde cada participante pode empunhar um marcador de texto e expressar sua opinião.

## B.3 Práticas de Time

### B.3.1 Time com Poder de Decisão

Um time com poder de decisão é aquele capaz de se reunir rapidamente e tomar uma decisão sobre algum problema que necessite rápida reparação. Segundo Kaner [79], o fator essencial para o real comprometimento do time é o envolvimento e engajamento de todos, cientes de cada questão em pauta, e o assessoramento de um facilitador, que também pode assumir o papel de um treinador do time. As decisões a serem tomadas devem levar em consideração as opiniões de todos os membros, mesmo as conflitantes, mantendo a velocidade sem o comprometimento do sucesso. A representação da Figura B.9 foi utilizada por Kaner para ilustrar a abordagem intitulada como "O Diamante da Tomada de Decisão Participativa", onde afirma que "o diamante é uma representação esquemática de diferentes estágios no tempo, em que o time tem de se mover para procurar e encontrar soluções satisfatórias."

**Business as Usual**: Algumas soluções triviais são propostas, mas o time não se arrisca muito, até que ideias inspiradoras surjam e o time evolua para a próxima zona. O facilitador pode impulsionar esse avanço, caso não apareça alguma proposta inovadora.

**Divergent Zone**: As percepções são diferentes, pois as pessoas se tornam mais descontraídas, ou nervosas e curiosas, etc. O facilitador precisa ajudar o time a expressar os diferentes pontos de vista, e técnicas como *brainstorming* podem ser utilizadas para que as pessoas possam expressar suas ideias.

**Groan Zone**: Conflitos podem surgir com o desentendimento entre os diferentes pontos de vista, o que pode trazer um certo desconforto. Nessa zona, o facilitador terá de trabalhar bastante para dirimir todas as dúvidas, assegurando que o time passe para

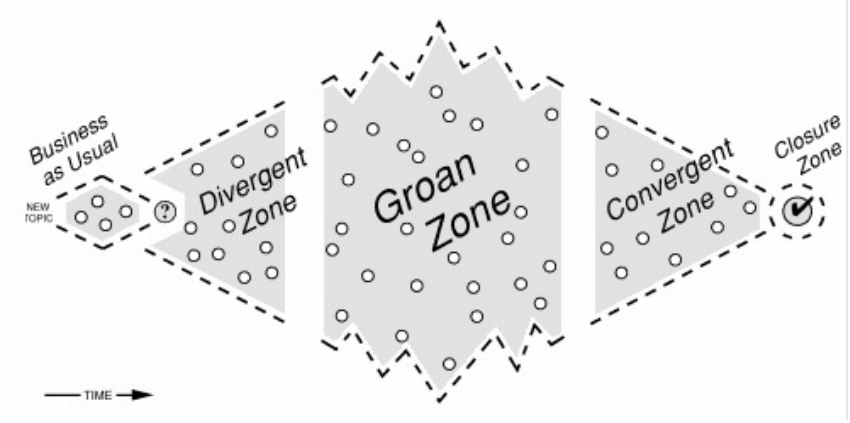



Figura B.9: Kaner [79] afirma que os times com poder de decisão são treinados e fortalecidos com um modelo conhecido como "O Diamante da Tomada de Decisão Participativa".

o próximo estágio, solucionando todos os problemas. Quando se atinge um entendimento comum, a próxima fase acontece naturalmente.

**Convergent Zone**: Todos os participantes do time já possuem um consenso e as discussões ocorrem mais traquilamente. Há um certo entusiasmo e o comprometimento é maior.

**Closure Zone**: Finalmente tem-se o subsídio para a tomada de decisão final. O facilitador tem de orientar o time para essa tomada de decisão sem que haja desvios, baseando-se em uma escala de aceitação que deve ser colocada à mostra. Uma boa ideia neste estágio é a utilização de testes de aceitação.

### B.3.2 Time Completo

Por Time Completo, entende-se que a equipe de desenvolvedores deve possuir um portfólio completo de pessoas capacitadas em todas as áreas necessárias ao andamento do projeto. Segundo Conboy et al. [31], "...em um ambiente ágil é importante que os desenvolvedores competentes possuam uma ampla gama de habilidades em oposição a ser perito em apenas uma área.", segundo ele:

> "Para ser um desenvolvedor ágil de sucesso, você precisa ser um codificador, um testador, um arquiteto, um cliente, um especialista em garantia de qualidade e uma infinidade de outras coisas relacionadas ao software."

Complementando a informação sobre a necessidade de um time completo, Grossman et al. [63] afirmam que:

> "A Programação ágil faz sentido como uma metodologia de desenvolvimento em um ambiente multidisciplinar que inclui equipes diversificadas, e talvez distribuídas,

> que requeiram uma estreita coordenação com habilidades multidisciplinares, como a arquitetura da informação, visual, XML, Java e outros."

Uma equipe ágil para ser considerada multidisciplinar deve ser compreendida de:

- Analistas de sistemas;
- Especialistas em testes;
- Especialistas em bancos de dados;
- Programadores especializados;
- Administradores de redes;
- Especialistas em design;
- Especialistas em requisitos;
- Especialistas em construção de relatórios;
- Outros...

### B.3.3 Sentar Junto

Para o fortalecimento da equipe, os participantes devem realizar reuniões com as pessoas sentadas. Conforme dizem Talbet et al. [148], as equipes devem trabalhar em espaços amplos, possibilitando boa comunicação. Dizem ainda:

> "As práticas de *sentar junto* exigem que todos, incluindo os testadores, sentem-se para conversar, no local de trabalho, e que todos participem no planejamento, nos jogos e reuniões, para que as pessoas conheçam uns aos outros e as respectivas áreas de conhecimento. Dado isto, os membros da equipe geralmente podem identificar a pessoa certa para um dado problema."

### B.3.4 Reunião em Pé

As equipes ágeis devem promover reuniões em pé para a transmissão de novos assuntos sobre o projeto e sua evolução, bem como das metas a serem atingidas naquele dia em específico. Sobre isto Talbet et al. [148] dizem que "...as equipes têm realizado uma reunião em pé todas as manhãs, desta forma praticando a integração contínua e alcançando uma a três compilações de integração em um dia médio de desenvolvimento."

### B.3.5 Área de trabalho Informativa

A técnica consiste em transformar o local de trabalho num ambiente que reflita o projeto em si, seja utilizando a prática de Kanban, estórias sobre um quadro branco, gráficos, *post-its* afixados sobre um quadro, etc. A ideia é fazer com que cada pessoa que passear pelo local tenha, por meio de simples observação, o conhecimento sobre a evolução do projeto. Segundo Talbet et al. [148], os membros da equipe podem promover seu local de trabalho usando cartões de estórias na parede, que descrevem os detalhes das diferentes tarefas e seu status, o que ajuda na comunicação do projeto (veja-se Kanban e Quadro de Tarefas).

### B.3.6 Equipes que Encolhem

A prática diz respeito à troca de participantes da equipe, para que esses sejam deslocados para outros projetos a fim de melhorá-los. Isto ocorre quando há uma evolução das capacidades da equipe, com um time muito bom em mãos dá-se ao luxo de remover alguns de seus participantes, retirando-os de seu local de trabalho e alocando-os em outras equipes, para melhorar as capacidades dessas novas equipes que se formam. Nguyen et al. [104], acreditam que o movimento de participantes entre equipes que se encolhem com esta prática ajudam a melhorar o software sendo produzido e também a aumentar as capacidades dos envolvidos.

### B.3.7 Continuidade da Equipe

Segundo Ozieranska et al. [109], a continuidade do trabalho em equipe refere-se a assegurar que o projeto permaneça com as mesmas pessoas trabalhando juntas, podendo haver rotação de membros, conforme o caso. A frequência de surgimento de problemas e o tempo que se leva para resolvê-los pode estar diretamente relacionada a esse aspecto. E ainda, segundo Guy et al. [64], "a continuidade da organização da equipe durante todo o processo deve ser preservada", isto tem de ser levado em consideração, quando o que se busca é a produtividade.

## B.4 Práticas de Desenvolvimento

### B.4.1 Aprender Fazendo

A prática de aprender fazendo consiste em colocar a equipe para trabalhar de forma que ela vá assimilando conceitos novos à medida que se depara com novas metodologias e novas

formas de pensamento. Os japoneses foram pioneiros nessa forma de aprendizagem e, conforme atesta Wang e Huzzard [159], trata-se de um tipo de aprendizagem organizacional que está comumente alinhado com a competitividade, produtividade e inovação tecnológica de mercado, e está bem identificada como um tipo de vantagem competitiva sustentável [141].

### B.4.2 Cliente Junto

Um dos grandes valores em práticas ágeis é a comunicação, onde as conversas presenciais são priorizadas, em reuniões do pessoal. Por causa disto, o cliente presente em pelo menos uma vez por semana é crucial para o desenvolvimento. É notório que essa prática não significa que o cliente tenha que estar o tempo inteiro junto à fábrica, e sim visitá-la para que ele possa dialogar com os desenvolvedores do projeto com a maior frequência possível e assim tirar suas dúvidas. A prática do cliente junto permite que os desenvolvedores entendam melhor o que se espera do produto e, por outro lado, permite uma melhor compreensão por parte do cliente, que se sente mais satisfeito com as respostas. A consequência disto é o software em seu produto final com as exatas especificações daquilo que o cliente realmente esperava.

Entretanto, segundo Beck et al. [16], "a prática do cliente no local é muitas vezes difícil de se implementar devido a restrições organizacionais ou de tempo". Daí, é necessário um esforço do time em compreender os anseios do cliente e saber planejar e dialogar, para que não haja constrangimentos.

### B.4.3 Desenvolvimento Orientado por Características das Funcionalidades

A prática do desenvolvimento de software orientada por funcionalidades é tão vasta que, assim como no Kanban, existem algumas metodologias que a retratam em separado, veja-se a metodologia Feature Driven Development — FDD [111] e Feature Oriented Model Driven Development — FOMDD [155], entretanto, conceitualmente é largamente praticada em combinação com Scrum e outras metodologias. Se o FDD elenca uma série de técnicas para alinhar os requisitos do sistema com as funcionalidades que ele deve ter, o Scrum, embora com poucas diferenças, busca também esse alinhamento. O segredo é gerar código executável baseado na modelagem e na composição das características de domínio de um objeto cujo resultado é o programa funcionando com suas funcionalidades bem próximas da realidade. A prática se confunde um pouco com a Modelagem em Objetos de Domínio, mas é bem mais ampla, considerando uma série de outros fatores.

### B.4.4 Estórias de Usuários

Estórias de usuários nada mais são que os artefatos que foram gerados durante a etapa de documentação de uma determinada funcionalidade do sistema. As estórias, na prática, são comumente empregadas para formalizar um Caso de Uso, como afirma Fowler [54]:

> "Casos de uso e estórias de usuários são semelhantes e ambas são formas de se organizar os requisitos. Os casos de uso organizam requisitos para formar uma narrativa de como os usuários se relacionam e usam um sistema. Assim, eles se concentram nos objetivos do usuário e como eles interagem com o sistema para satisfazer suas metas."

Entretanto, enquanto Casos de Uso descrevem ações de interação conforme uma narrativa normalmente impessoal entre o usuário e o sistema, com dados técnicos que foram polidos e convertidos para o desenvolvedor poder realizar o seu trabalho, as estórias de usuários se baseiam mais nos objetivos e em como o sistema os alcança, do jeito que foram contadas pelo cliente. Desta forma, as estórias de usuários podem fracionar os requisitos facilitando o esforço dos analistas e desenvolvedores, pois são descrições mais curtas e mais simples segundo o ponto de vista do cliente [54]. Dizem que se não há espaço suficiente para escrevê-las em um cartão, deve-se refiná-la (repensá-la), dividindo-as em frações menores.

As estórias de usuários devem ser construídas sob as perspectivas do ator, da ação e da funcionalidade, conforme abaixo:

- **Ator** – É o usuário (cliente), o principal interessado na funcionalidade retratada. É recomendado anotar o nome e o telefone do usuário para depois ficar mais fácil identificá-lo em uma possível retirada de dúvidas.
- **Ação** – É o que o ator deseja realizar no sistema.
- **Funcionalidade** – É o resultado esperado após o usuário ter executado a ação. Também pode descrever as nuances do que se deseja e as justificativas do porquê tem de ser assim.

### B.4.5 Modelagem em Objetos de Domínio

A modelagem em objetos de domínio consiste em mapear os objetos que compõem o domínio da empresa. Essa atividade é normalmente executada após a realização de um estudo detalhado sobre o escopo do sistema e a visão de negócios. É extremamente importante a criação do modelo, pois ele servirá de guia para a implementação, entretanto, requer muita habilidade e não é tarefa trivial.

Um modelo de domínio é um documento que simplifica os principais aspectos do funcionamento de uma empresa, com seus produtos, suas atividades e suas operações ao longo de sua existência. Constantemente é bem específico ao *modus operandi* da organização que se mapeia e culmina em um vocabulário próprio que a identifica no universo. Um modelo de domínio descreve as informações de interesse do negócio da empresa, que são categorizadas em **entidades de domínio** com dados e comportamentos associados e interligados entre si. Exemplos de modelagem de domínio incluem modelos entidade-relacionamento tradicionais e modelos de objetos que podem ser traduzidos em classes de linguagem de programação computacional, como a Java, por exemplo.

As classes de domínio são utilizadas para definir as estruturas de dados, mapeadas para um banco de dados relacional, Souza e Almeida Falbo [143] relatam como funcionam essas classes, na prática:

> "As classes de domínio representam os conceitos de domínio referentes aos negócios identificados e modelados em diagramas de classes durante a análise de requisitos e refinados durante o projeto. Normalmente, essas classes são muito simples, lembrando meras estruturas de dados na maioria das vezes (e, portanto, comumente chamadas de *objetos mudos*). A inteligência que está faltando nesses objetos é definida para as classes de aplicativo, que mapeia para codificar o que foi definido como casos de uso na fase de especificação de requisito. Desta forma, cabe às classes de aplicativo criar, recuperar, atualizar e excluir objetos de domínio da mídia persistente, de acordo com as descrições de casos de uso."

A tarefa de mapeamento, como relatado, pode ser realizada por meio de ferramentas disponíveis na linguagem pré-definida, ou ainda ser codificada por um bom programador, e pode ser representada como na Figura B.10, que ilustra bem esse processo.

### B.4.6 Programação em Pares

A programação em pares, conforme afirmam Cockburn e Williams [29], ocorre quando dois programadores trabalham colaborativamente no mesmo algoritmo, projeto ou tarefa de programação, sentados lado a lado ao computador. Esta prática foi citada várias vezes nas últimas décadas como uma forma positiva de desenvolver software [15].

No entanto, tem-se observado a existência de algumas controvérsias, pois a despeito da economia de recursos, dois estariam fazendo o trabalho de um, e alguns gerentes relutariam em desperdiçar mais um programador para a tarefa [29]. Outras vezes, programadores experientes são relutantes em programar com outra pessoa, pois dizem que seu código é "pessoal", ou que outra pessoa só poderia atrasá-los, e ainda outros dizem que trabalhar com um parceiro traria dificuldades em coordenar horários de trabalho.

Ao mesmo tempo, é certa a evidência de que, em alguns casos, a programação em pares traz benefícios, veja-se:

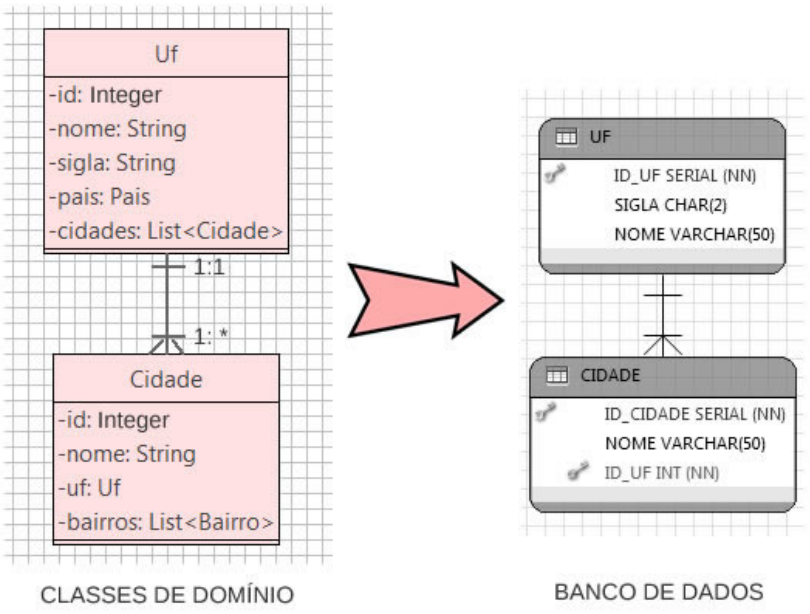



Figura B.10: Mapeamento das classes de domínio para o banco de dados; Fonte: Elaborada pelo autor.

- Vários programadores bem respeitados preferem trabalhar em pares, tornando essa prática o seu preferido estilo de programação [32] [15];
- Programadores em pares experientes descrevem que a prática é "mais de duas vezes mais rápido" [15];
- Evidências qualitativas sugerem que o design é melhor, resultando em código mais simples e mais fácil de se entender [29];
- Mesmo os novatos podem contribuir para o aprendizado de um perito em programação, de acordo com entrevistas [29].

Cokcburn, em seu trabalho detalhado sobre custos e benefícios da programação em pares [29], conclui afirmando que:

> "Os benefícios significativos da programação em pares são:
>
> - Muitos erros de digitação são plotados (revisão contínua);
> - O resultado final dos defeitos é estatisticamente inferior;
> - Os projetos são melhores e o código é mais enxuto;
> - A equipe resolve problemas mais rapidamente (revezamento);
> - As pessoas aprendem muito mais, sobre desenvolvimento de software;
> - O projeto acaba com várias pessoas compreendendo cada parte do sistema;
> - As pessoas aprendem a trabalhar juntas e a falar mais frequentemente juntas, dando melhores informações sobre o fluxo e dinâmica da equipe;
> - As pessoas desfrutam mais do seu trabalho."

### B.4.7 Propriedade Coletiva

O código-fonte do projeto não possui um proprietário em específico. Todos os membros da equipe são donos do código, o que significa que qualquer um pode adicionar valor. Mas é importante salientar que cada código adicionado deve ser precedido de testes e que, acima de tudo, haja um versionamento do arquivo sendo criado, excluído ou modificado. Isto pode ser conseguido facilmente com a adoção de ferramentas de versionamento disponíveis na comunidade.

Uma grande vantagem do código coletivo é a sua manutenção por outras pessoas. Assim sendo, se um participante deixar o projeto, outros podem dar continuidade.

Segundo Maruping et al. [96], foi constatado, por meio de pesquisa científica, que "[...] a propriedade coletiva e os padrões de codificação têm um papel importante na melhoria da qualidade técnica do projeto de software". E hoje sabemos que é quase impossível confeccionar software sem o uso dessa prática, que mantém o código em um único local à disposição de todos os programadores.

### B.4.8 Prototipação

Um protótipo pode ser considerado uma versão inicial do sistema que se disponibiliza logo no início do processo de desenvolvimento. Segundo Paetsch et al. [110], os protótipos são frequentemente utilizados para obter e validar os requisitos do sistema e sua principal finalidade é facilitar o entendimento dos requisitos. Os desenvolvedores sentam-se sobre um papel (ou podem utilizar o quadro de lousa) e começam a desenhar a funcionalidade como foi descrita no caso de uso. Nesta tarefa, é importante que todos expressem sua opinião sobre como ficaria uma determinada tela e de como o usuário poderia interagir com ela.

A prototipação é uma prática aliada às metodologias ágeis porque está em perfeita consonância com elas, porque agiliza o processo de desenvolvimento e porque pode ser utilizada para validação junto ao cliente. A Figura B.11 e a Figura B.12, gentilmente cedidas pela equipe de desenvolvedores do projeto Gestão do Desempenho do Exército Brasileiro [22], evidenciam como eles lidaram com a prototipação, primeiro realizando um mapeamento do processo (Figura B.11) e depois o protótipo final da tela (Figura B.12). Tais rascunhos foram feitos à mão livre com lápis, borracha e papel, sem o uso de ferramenta de apoio. Posteriormente sim, os desenhos foram refinados com as ferramentas **Bisage** e **Papel & caneta** (Figura B.12).

Segundo Paetsch et al. [110], os protótipos podem ser baseados em papel, como relatado, ou automatizado, onde é requisitado o trabalho de um *webdesigner*, com todas as telas desenhadas em HTML e com automatização apenas da camada view.

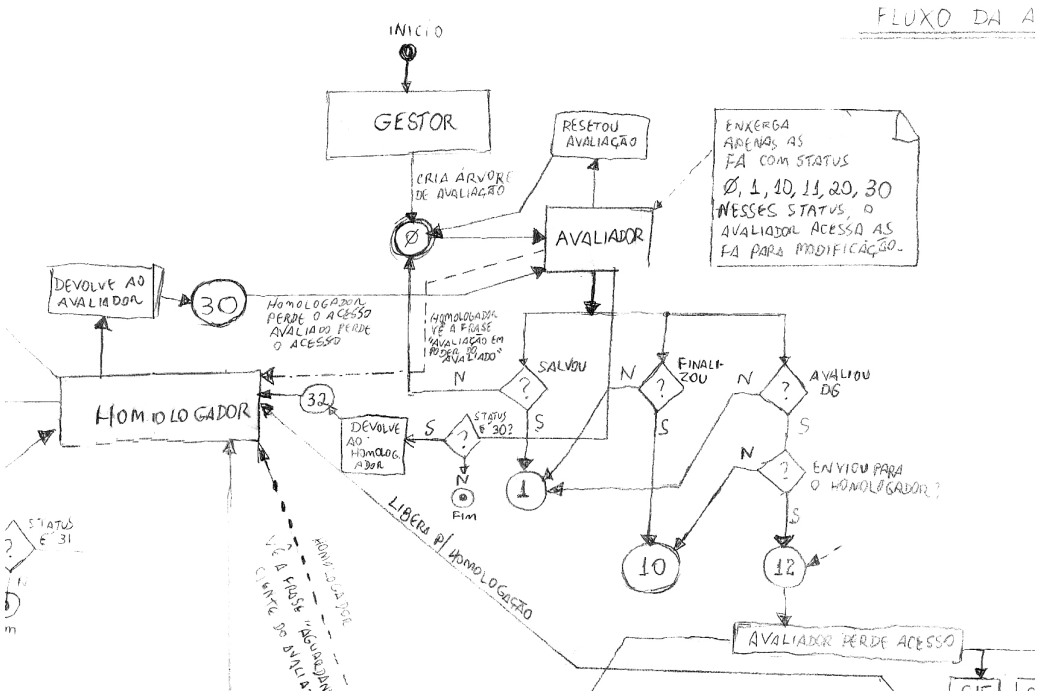



Figura B.11: Mapeamento de processo executado à mão livre, que serviu de ponto de partida para a prototipação. Fonte: Exército Brasileiro.

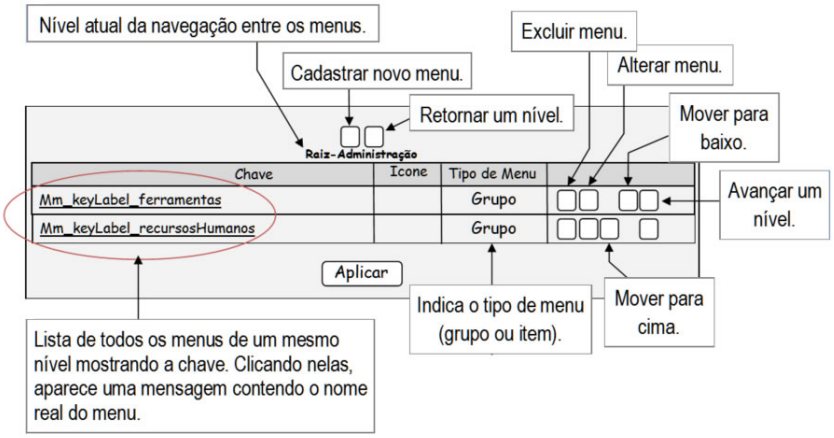



Figura B.12: Em um momento posterior, os desenhos foram refinados com ferramentas de apoio. Fonte: Exército Brasileiro.

### B.4.9 Refatoração

Segundo Fowler, em seu trabalho *Refactoring: Improving the Design of Existing Code* [56]:

> "Refatoração é a arte de alterar um software de modo que o comportamento externo do mesmo não mude, mas que a sua estrutura interna seja melhorada. É uma forma disciplinada de se aperfeiçoar um código diminuindo a chance de

> falhas. Em essência, quando se usa refatoração, o projeto melhora com a melhoria do código."

Na prática, o desenvolvedor escreve o código, modifica ele várias vezes, escreve os testes para ver se ele vai funcionar ao contento e atinge um grau de satisfação consigo mesmo, quando a funcionalidade estiver de acordo com os requisitos. Esse processo de escrever e reescrever produz muito "lixo" dentro do código e, fatalmente, o programador terá de refatorá-lo para deixá-lo mais limpo e inteligível. Nesta prática, é importante que seja adicionado comentários sobre o propósito do código e como ele funciona, variáveis utilizadas e o porquê delas. Esses comentários devem ser adicionados depois que o código foi rigorosamente testado e sabe-se que o mesmo não vá se modificar muito. Os comentários ajudarão outros desenvolvedores quando os mesmos forem revisitar o código que, com um simples olhar sobre ele, entenderão como funciona, facilitando a manutenção.

## B.5 Práticas de Teste

### B.5.1 Teste Primeiro (TDD)

*Test Driven Development — TDD*, ou ainda, Desenvolvimento Dirigido por Testes é uma técnica de programação onde o principal objetivo é escrever código funcional limpo a partir de um teste que tenha falhado. Com o efeito colateral, obtém-se um código fonte bem testado [13].

Segundo Mazuco e Canedo [97], ao utilizar Test Driven Development:

> "...inicialmente, escreva um teste para a regra de negócios do seu código principal que vá falhar (o ideal é que esse código principal ainda nem tenha sido implementado), em seguida, faça-o passar da maneira mais simples possível e, por fim, refatore o código fazendo-o passar. Esse ciclo é conhecido como **Ciclo Vermelho-Verde Refatora**"

A Figura B.13 ilustra bem a técnica utilizando ferramenta IDE para facilitar o processo...

### B.5.2 Testes Integrados

Os testes integrados são utilizados para verificar a consistência de uma determinada funcionalidade que incorpore um ou mais módulos, combinando-os para ver se vai ou não ocorrer alguma falha. Eles devem acontecer sempre após a realização dos testes unitários (cada módulo deve ser testado separadamente). O propósito é a verificação dos requisitos funcionais e de confiabilidade da modelagem do sistema, pois com eles é possível detectar os erros de interface entre os componentes do programa.

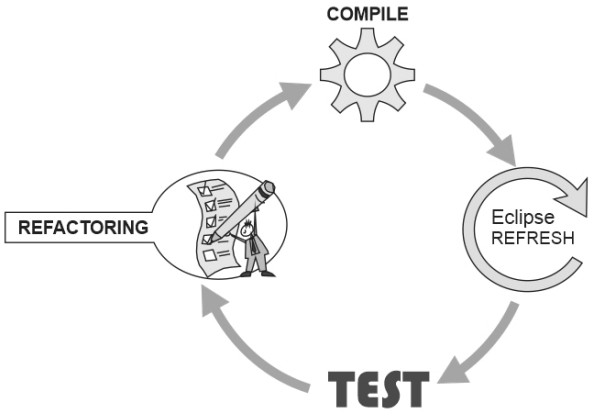



Figura B.13: O ciclo de TDD, segundo Mazuco e Canedo [97].

## B.6 Práticas de Release

### B.6.1 Retrospectiva ao Término do Ciclo

Ao término de cada ciclo, a equipe deve dar uma parada para refletir sobre onde errou e também sobre os acertos que tiveram. Segundo Derby et al. [39], "...na retrospectiva, o líder apoia a equipe para discutir os eventos e para entender os fatos e os sentimentos durante a iteração", portanto, além das discussões normais sobre o projeto, tais como técnicas utilizadas que deram certo e as que não deram, os participantes podem também comentar o lado afetivo, se discordam ou não sobre alguma atividade que se passou ou que ainda vai ocorrer, pois, "as retrospectivas permitem o aprendizado do time com um todo, agindo como catalizador de mudanças e impulsionador de novas ações e atitudes" [39].

A Figura B.14 é bem explicativa e ilustra uma reunião de retrospectiva, sobre aquilo que o gerente tem de prestar mais atenção e fazer executar durante a reunião. Pode ser adaptada para qualquer metodologia de desenvolvimento ágil:

Onde:

1. **Set the Stage**: Ao falar sobre as bases, reveja o objetivo e a agenda. Deve-se criar um ambiente de participação através de *check-in* e do estabelecimento de acordos de trabalho.

2. **Gather Data**: Revise as informações objetivas e subjetivas para criar uma imagem compartilhada. Referencie a pessoa responsável por cada atividade. Quando o grupo vê a iteração de muitos pontos de vista, eles têm uma visão maior.

3. **Generate Insights**: Dê um passo atrás e olhe para a foto que a equipe criou. Use atividades que ajudem as pessoas a pensar juntos.

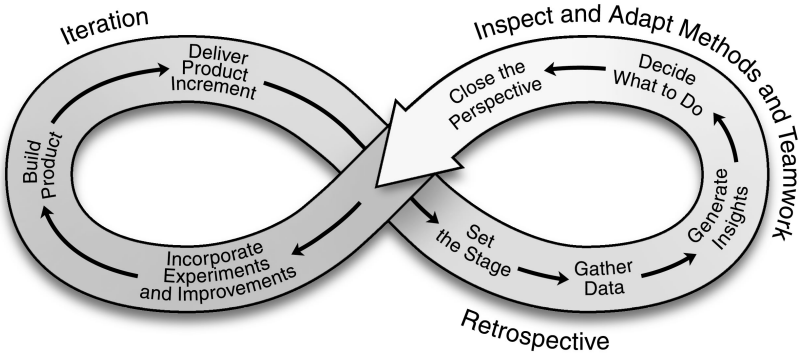



Figura B.14: O ciclo ágil sob o foco da retrospectiva, segundo Derby et al. [97].

4. **Decide What to Do**: Decida o que fazer. Priorize as percepções da equipe e escolha algumas melhorias ou experimentos que podem fazer a diferença para a equipe.

5. **Close the Retrospective**: Resumir como a equipe irá acompanhar os planos e compromissos. Agradeça à equipe e membros por seu trabalho duro.

### B.6.2 Entrega de 2 a 4 Semanas

Segundo Fowler e Highsmith [58], precisamente o 3º princípio do Manifesto Ágil, "entregar o software funcionando de forma frequente, de duas semanas a dois meses, de preferência à uma escala de tempo mais curta", é a máxima prática ágil que se pode realizar em um projeto. Isto significa que os membros da equipe precisam trabalhar tendo em foco o software funcionando em pouco espaço de tempo, o que exigirá algum esforço do time.

Entregar o software funcionando é uma das formas que o pessoal tem de apresentar o trabalho e receber um retorno sobre ele, do que foi realizado nas últimas semanas. Assim sendo, a entrega em uma menor escala de tempo admissível, permite uma melhor avaliação do mesmo, tal que o produto final seja o esperado pelo cliente.

O Scrum é talvez uma das metodologias ágeis que mais leva a sério esta prática, a cada período de desenvolvimento, o que eles chamam de *Sprint*, Figura B.15.

O recomendado é que esse período seja bem curto, algo em torno de 2 a 4 semanas no máximo, fazendo com que a frequência de entrega de software seja alta. Isto possibilita elevar o alto astral do pessoal e também os ganhos da fábrica, bem como a satisfação total do cliente, que espera receber o software funcionando o mais rápido possível.

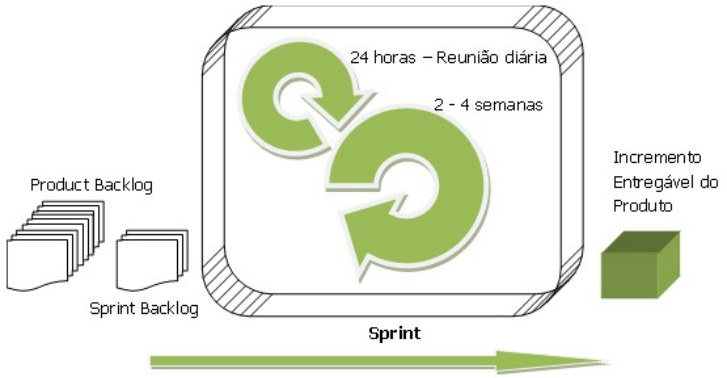



Figura B.15: Entregas de 2 a 4 semanas, adaptado de http://blog.myscrumhalf.com/ 2012/ 02/o-que-e-sprint-%E2%80%93-faq-scrum/, acessado em 14-abril-2017.

### B.6.3 40 horas Semanais

de acordo com o 8º princípio do Manifesto Ágil [58] "...patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente", devemos ser capazes de trabalhar em uma fábrica de software mantendo uma carga-horária de 8 horas por dia, de segunda a sexta-feira.

Em um trabalho sobre indústria de jogos, Robinson [127] afirma que trabalhar mais do que 40 horas por semana aumenta os custos e diminui a qualidade do trabalho produzido. Segundo ele:

> "Programadores cansados trabalham mal. Ocorre uma diminuição natural da qualidade de vida das pessoas que compõem o time. [...] depois de certo tempo, as pessoas passam a produzir negativamente."

Conforme o autor, essa visão pode ser mais complexa do que se pensa, e pode ser representada pela seguinte função:

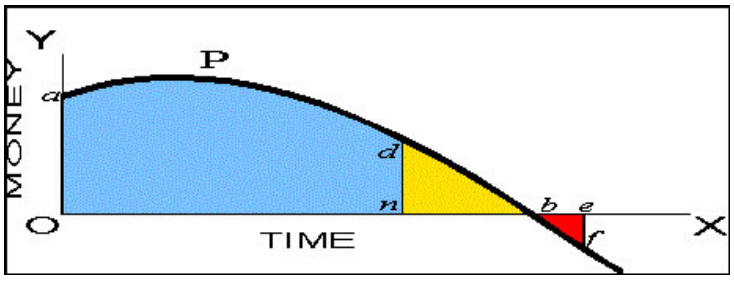



Figura B.16: O produto de trabalho de um homem em relação a horas trabalhadas pode ser medido conforme a equação O = P(t1, t2, t3 ...tn).

Onde **O** é a produção total e **P()** representa as mudanças na produtividade horária que ocorrem ao longo dos tempos (t1 — tn) — a produtividade do trabalhador por

unidade de tempo em um determinado número de horas trabalhadas por dia. P() pode variar de trabalhador para trabalhador, porque alguns produzem mais do que outros. P() também poderá variar por hora, porque os seres humanos não são máquinas e não fazem exatamente a mesma quantidade de trabalho iguais. Finalmente, P() variará de acordo com a história recente do trabalhador, porque as pessoas não trabalham tão bem na manhã depois de terem trabalhado até tarde da noite do dia anterior.

Percebe-se que há um ponto, **b** — podendo representar a oitava hora de trabalho, onde trabalhar mais horas não cria mais valor. De fato, após **b**, cada hora trabalhada a mais produz valor negativo, tanto na quantidade quanto na qualidade do trabalho realizado no final de um determinado dia. Eventualmente essa fadiga diária poderá agravar a produção do dia seguinte e com isto havendo uma queda na produtividade.

### B.6.4 Adiamento de Decisões ao Máximo

De acordo com Smits [142], "as decisões sobre como e onde implementar um recurso em um conjunto de componentes deve ser adiada até o último momento possível." Isto porque o processo de decisão deve ser baseado nos processos de aprendizagem, pois o amadurecimento da equipe traz confiança nas tomadas de decisões.

Uma das principais maneiras de se fazer isto é diminuir, de alguma forma, as incertezas e assim retardando as decisões, até que elas possam ser feitas com base em acontecimentos mais firmes, previsíveis e conhecidos. Segundo a metodologia Lean [121], as decisões tardias tendem a ser as mais corretas porque as melhores decisões são baseadas em fatos, e não em suposições. Uma boa estratégia para adiar decisões ao máximo é usar a capacidade da equipe e as práticas que permitam abraçar as mudanças tardiamente.

### B.6.5 Estimativas

Estimativas, antes de serem meras "previsões" sobre a realização de um trabalho em si, representam o comprometimento de alguém da equipe, da empresa, onde o nome pode significar muito. É também sinônimo de certeza de que algo será entregue no prazo estipulado.

Muitas vezes, a equipe estima as dificuldades e o tempo necessário para realizar o trabalho, mas se sente incapaz de se comprometer com prazos fixos. Sehlhorst [134], descreve a estimativa como a "reflexão da distribuição de probabilidades e não números específicos." A estimativa pode carregar muito risco, pois elas podem estar erradas, por isto são caras. Entretanto, são carregadas de valor e, se rigorosamente cumpridas, trazem satisfação para os donos e para os clientes, permitindo investimentos eficientes no esforço da equipe, obtendo o maior lucro possível.

De acordo com Todd [152], as incertezas sobre as estimativas podem ser estudadas utilizando um gráfico denominado "O Cone da Incerteza". Segundo o autor, no início de um projeto, não são claros os detalhes específicos da natureza do software a ser construído, detalhes de requisitos, detalhes da solução, planos de projeto, pessoal e outras variáveis. A variabilidade nesses fatores contribui para a variabilidade das estimativas do projeto. À medida que essas fontes de variabilidade são investigadas e fixadas, a variabilidade no projeto diminui, e assim a variabilidade nas estimativas do projeto também pode diminuir.

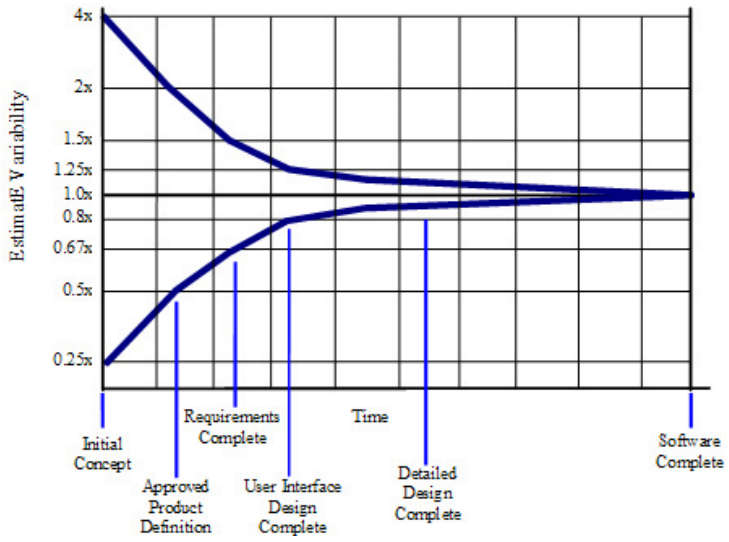



Figura B.17: Estimativas de um projeto, mesmo que bem planejado e bem controlado pelo gerente, segundo Todd [152].

Na Figura B.17 o eixo vertical contém o grau de erro que foi encontrado nas estimativas criadas em vários pontos do projeto, e o eixo horizontal o tempo de execução do projeto. As duas curvas juntas mensuram o grau (tamanho) do erro da estimativa, quanto mais as duas curvas se aproximam, mais diminuem-se o grau das incertezas. Veja que as estimativas criadas muito cedo estão sujeitas a um alto grau de erro. No exemplo da figura, as estimativas criadas no momento do conceito inicial podem ser imprecisas por um fator de 4x no lado alto ou 4x no lado baixo (também expresso como 0,25x, que é apenas 1 dividido por 4). Significa dizer que a equipe trabalha com um grau de incerteza 4 vezes para mais ou 4 vezes para menos.

Por outro lado, em um projeto mal controlado, podemos ter estimativas como as mostradas na Figura B.18. Aqui, mostra-se o que acontece quando o projeto não é conduzido de forma a reduzir a variabilidade — a incerteza não gera um cone, mas sim uma nuvem que persiste até o fim do projeto, as estimativas não convergem.

Nas metodologias ágeis preza-se a transparência e assim a existência de riscos são sempre presumidas. Eles existirão e serão forte elementos para a tomada de decisão. Lidar

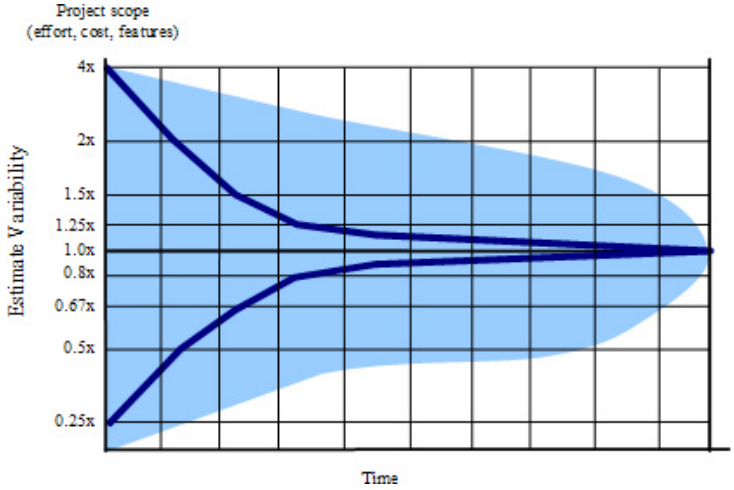



Figura B.18: Estimativas de um projeto, onde não haja um controle rigoroso do processo de desenvolvimento, segundo Todd [152].

com incertezas requer espertize e o engajamento do pessoal, alinhado ao conhecimento da tecnologia ajudam a vencer nesse processo. Com o tempo e a a evolução da maturidade, as estimativas tendem a melhorar.

### B.6.6 Satisfação Total do Cliente

A satisfação total do cliente, na verdade, antes de ser uma prática, é o 1º princípio do Manifesto Ágil [58], onde afirma-se que "A nossa maior prioridade é a satisfação do cliente através de entregas contínuas e adiantadas de software com maior valor agregado". Essa satisfação pode ser alcançada quando se entrega ao cliente aquilo que ele realmente deseja, ou seja, o software que venha trazer soluções e benefícios para a sua empresa. Mas apenas isso não basta, pois o software tem de ser entregue de forma continuada, com a finalidade de possibilitar a validação do seu produto ao longo do desenvolvimento, e no final iniciando ou aumentando fluxo de retorno dos investimentos.

E se qualidade traz satisfação total dos clientes, Crosby [33], diz que "qualidade é conformidade com as especificações" e, segundo Slack e Stuart, dois grandes estudiosos da gestão da produção e operações [140], dizem que a qualidade gira em torno de 5 objetivos que devem ser atingidos, como evidenciado na Figura B.19.

Hoje em dia, em um ambiente globalizado e competitivo, a busca pela qualidade não é um diferencial, e sim uma obrigação de toda a empresa para garantir a sua própria sobrevivência. baseado nisto, busca-se a satisfação do cliente pela qualidade do produto final que é oferecido, mas antes, levando-se em conta os seguintes aspectos:

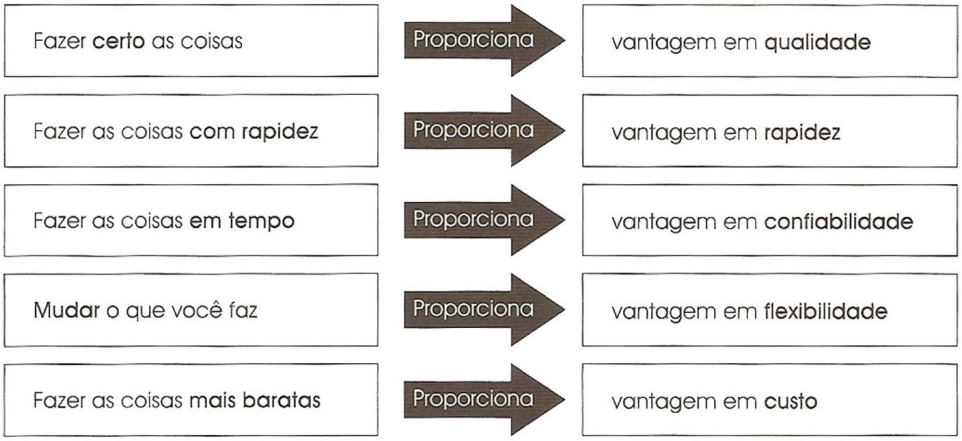



Figura B.19: Os cinco objetivos de desempenho de um processo produtivo, segundo Slack e Stuart [140].

- Os clientes é que são os personagens que decidem se um determinado produto atende ou não as especificações. E antes de atender as especificações, devem atender as suas expectativas;
- A satisfação do cliente sempre é influenciada pelo que a concorrência oferece, pois o cliente pode ser considerado como um alvo móvel, cujas expectativas crescem à medida que se ampliam suas opções;
- A satisfação total do cliente pode ser obtida por meio de um conjunto de atividades ao longo do tempo e não apenas pela ocasião da venda de um único produto. Estas atividades podem incluir o projeto, a fabricação, a própria venda, o atendimento e a assistência técnica, entre outros.

Ninguém melhor que o próprio cliente para saber se o software em desenvolvimento agregará ou não valor ao seu negócio e, considerando que o quanto mais rápido forem as entregas, em pedaços, menores serão os custos, não se deve aguardar o fim do seu desenvolvimento para entregar o software por inteiro. Planejar as entregas de forma contínua é uma boa estratégia que permite ao cliente avaliar a qualidade do produto com maior rapidez, mesmo que essas pequenas entregas não tragam todas as funcionalidades esperadas, mas já contando com um conjunto suficiente delas para trazer benefícios de forma antecipada.

# Apêndice C

# Mind Map Dybå e Dingsøyr e Dickert et al. - Resumo de Estudo Terciário

**DE SOUZA (UNB - 2016)**

Utilização de Projet-Piloto em Gestão Pública.

Cuidado com fanatismos sobre metodologias ágeis.

Entrega incremental

Para manter a agilidade, EVITAR O USO DE CRONOGRAMA.

É necessário um equilíbrio entre a conclusão de novas estórias de usuários e manter a estabilidade em geral do aplicativo.

Reduzir as dependências entre as equipes.

Coragem

Métodos ágeis podem trabalhar com métodos antigos, em paralelo.

Gestores devem ser notificados para resistirem à tendência de mando e controle, permitindo espaço para a auto-organização, a fim de realizar a mudança na mentalidade das pessoas envolvidas.

Planejar para LONGO PRAZO é um anti-prática ágil.

Testes automatizados como tônica - TDD.

(1) Que desafios foram relatados na adoção de práticas ágeis em larga escala?
(2) Que fatores de sucesso têm sido relatados na adoção de práticas ágeis em larga escala?

Refinamento dos requisitos são considerados desafiadores e são imprescindíveis em Scrum. Cuidado com a ambiguidade dos requisitos.

PMO (Escritório de Projetos) pode se tornar um dos gargalos da metodologia ágil. Deve-se ter o cuidado na abordagem devido a alta burocracia.

Um dos maiores desafios de transformação é a dificuldade em coordenar o trabalho de várias equipes ágeis. A coordenação pode ser obstruída por equipes independentes que não respeitam o contexto mais amplo. Quando muitas equipes implementam ágil sem uma orientação consistente, atritos e fragmentações podem surgir.

Alinhar os recursos humanos.

**DISCKERT ET AL. (2016)**

**COCKBURN E HIGHSMITH (2001)**

**DINGSØYR E NILS (2014)**

**RONKAINEN E ABRAHANSSON (2002)**

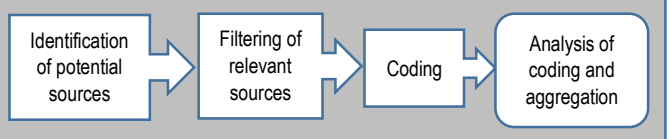
**KAISTY ET AL. (2013)**

**DYBA (2008)**

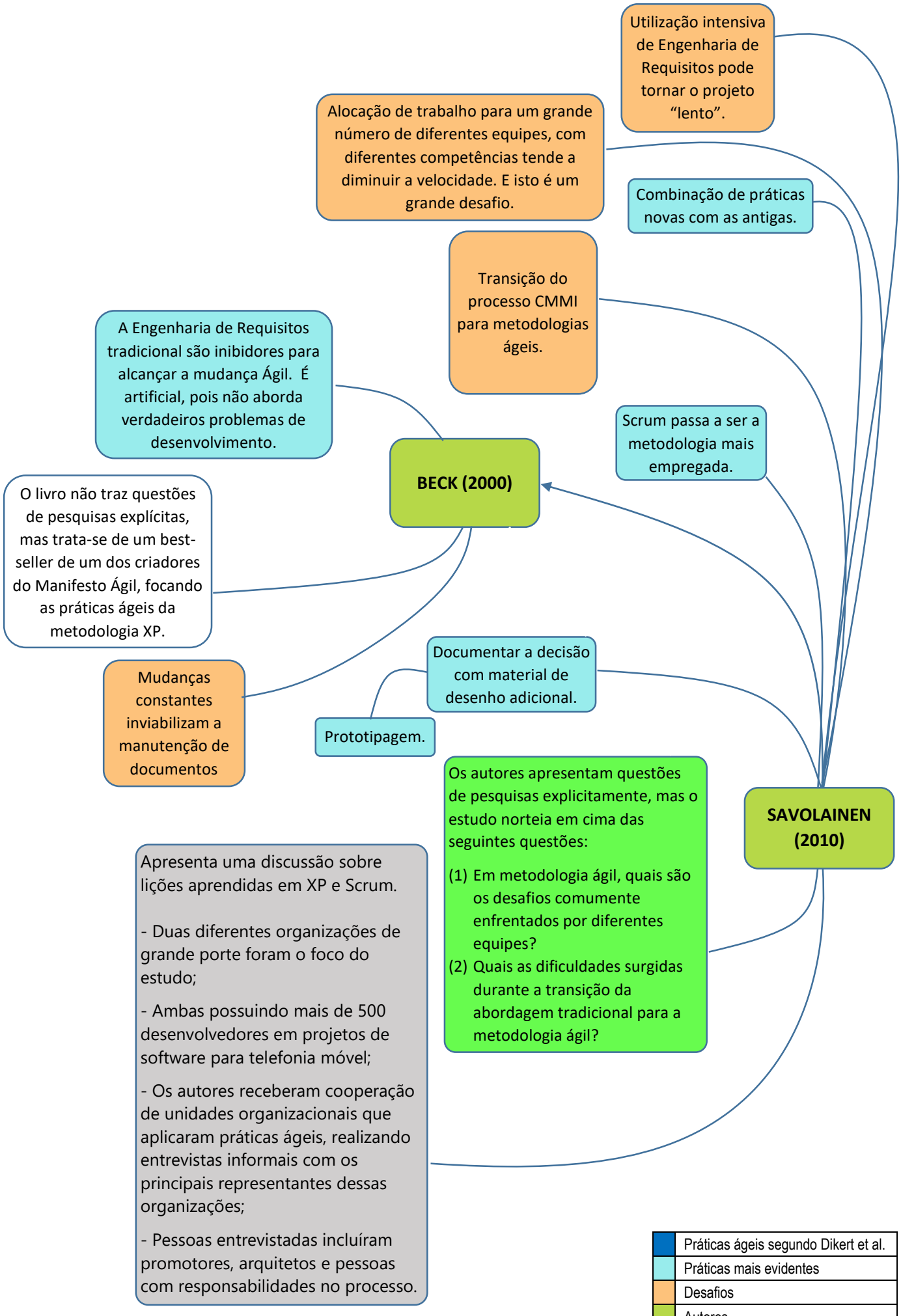
**SAVOLAINEN (2010)**

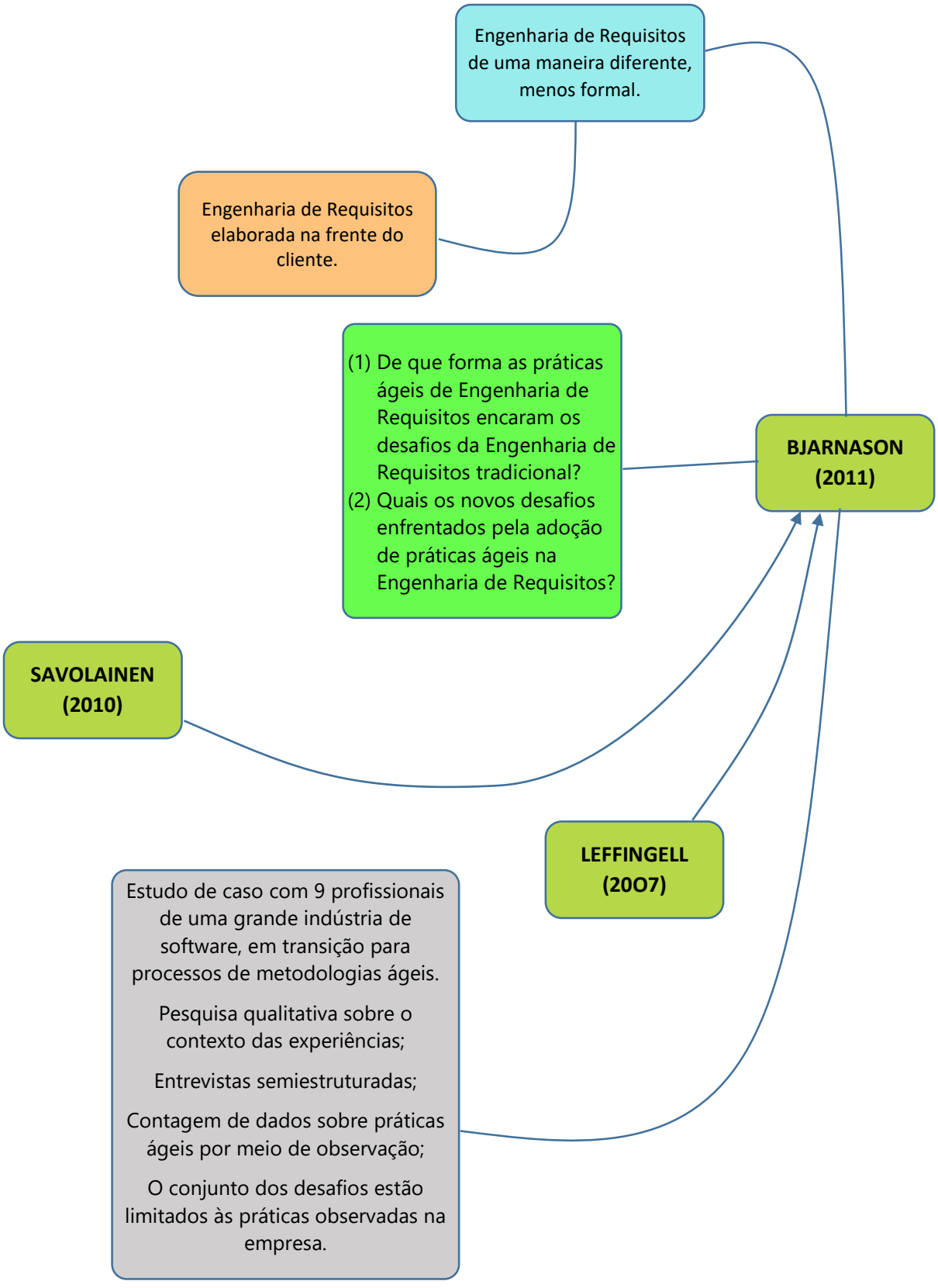
**BJARNASON (2011)**

Revisão sistemática da literatura. O processo de pesquisa consistiu em quarto etapas principais, representadas na figura ao lado.

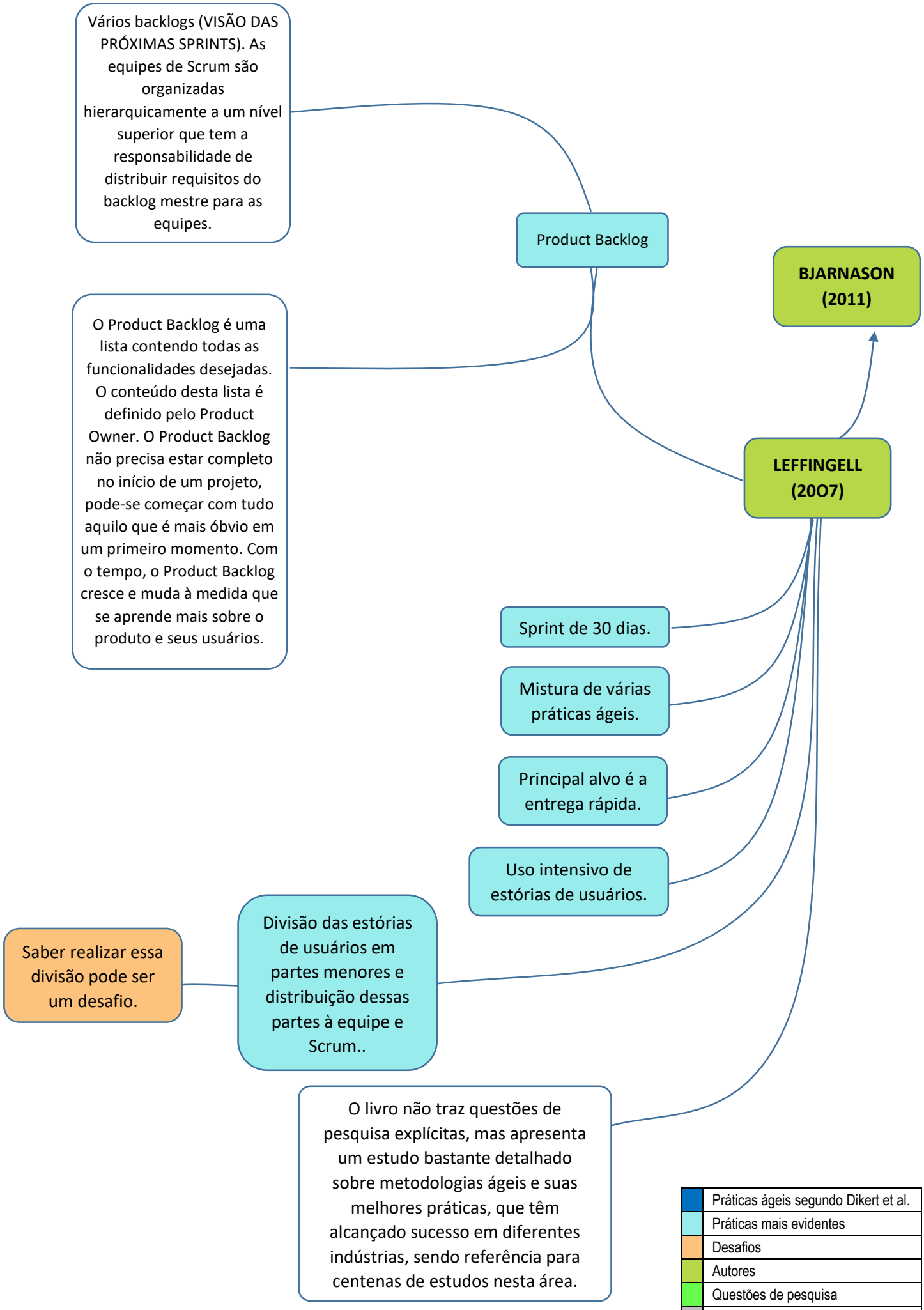


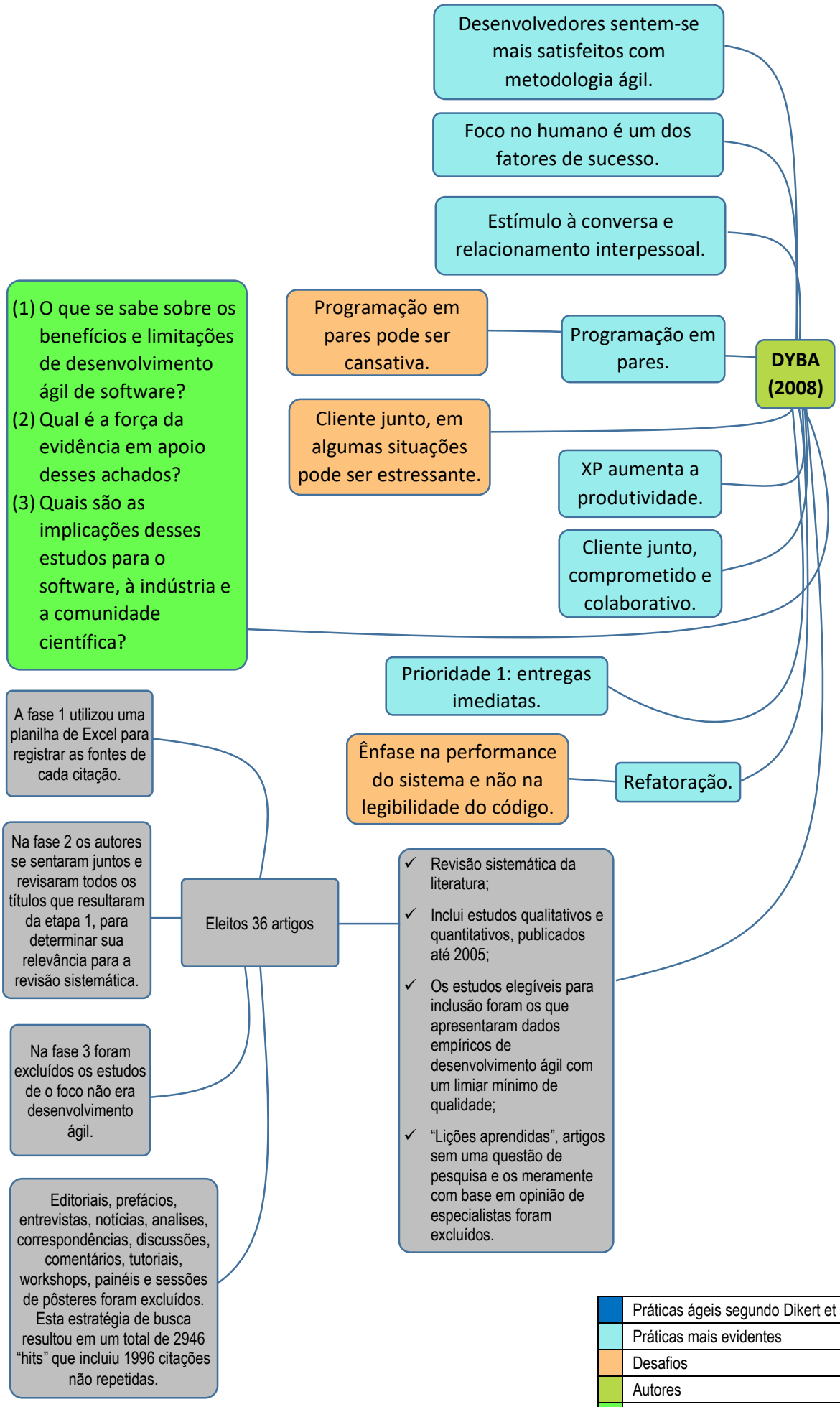

| |
|---|
| Práticas ágeis segundo Dikert et al. |
| Práticas mais evidentes |
| Desafios |
| Autores |
| Questões de pesquisa |
| Metodologias |
| Notas Explicativas |

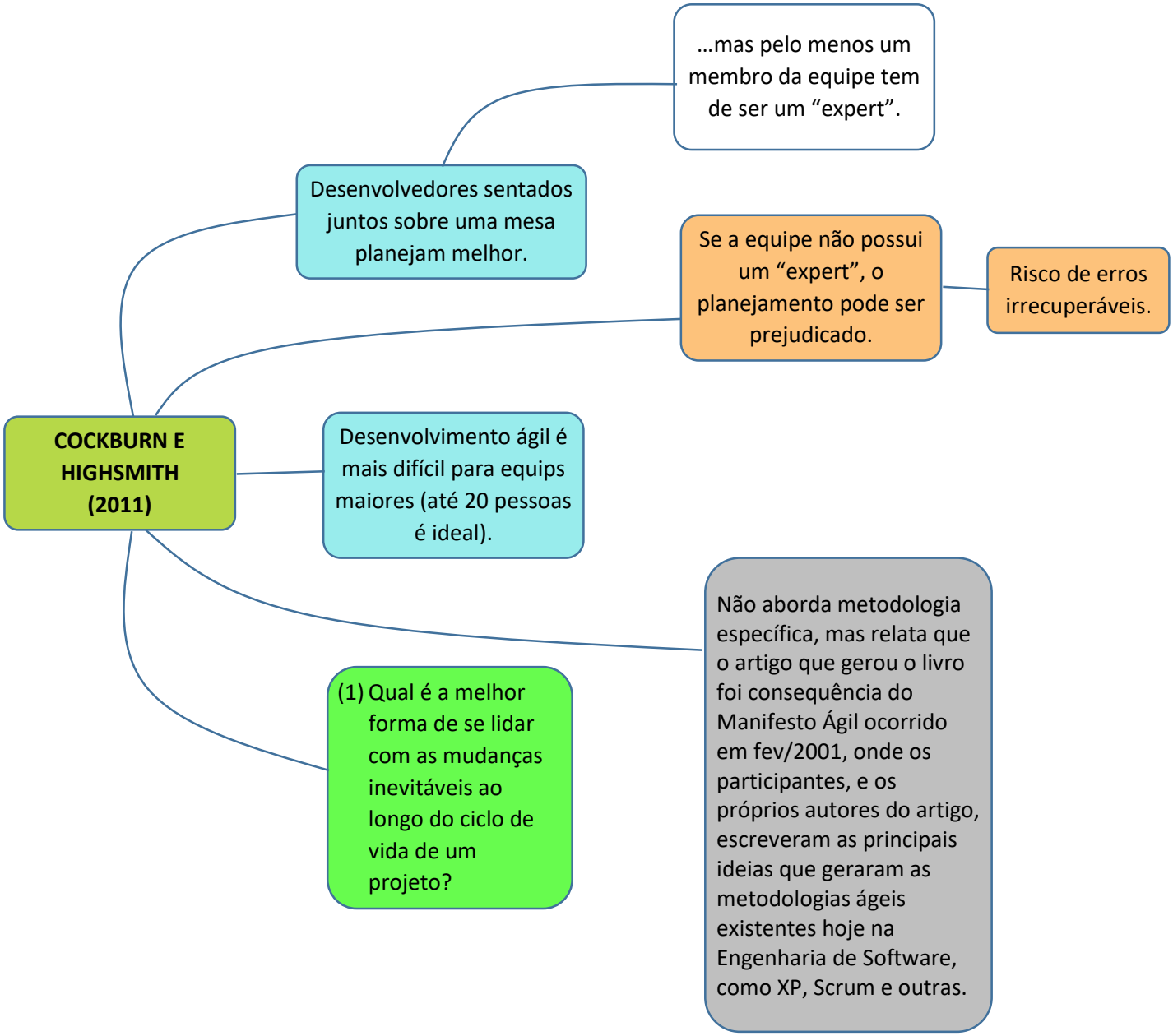

Utilização intensiva de Engenharia de Requisitos pode tornar o projeto "lento".
Alocação de trabalho para um grande número de diferentes equipes, com diferentes competências tende a diminuir a velocidade. E isto é um grande desafio.
Combinação de práticas novas com as antigas.
Transição do processo CMMI para metodologias ágeis.
A Engenharia de Requisitos tradicional são inibidores para alcançar a mudança Ágil. É artificial, pois não aborda verdadeiros problemas de desenvolvimento.
Scrum passa a ser a metodologia mais empregada.
BECK (2000)
O livro não traz questões de pesquisas explícitas, mas trata-se de um best-seller de um dos criadores do Manifesto Ágil, focando as práticas ágeis da metodologia XP.
Documentar a decisão com material de desenho adicional.
Mudanças constantes inviabilizam a manutenção de documentos
Prototipagem.
Os autores apresentam questões de pesquisas explicitamente, mas o estudo norteia em cima das seguintes questões:
(1) Em metodologia ágil, quais são os desafios comumente enfrentados por diferentes equipes?
(2) Quais as dificuldades surgidas durante a transição da abordagem tradicional para a metodologia ágil?
SAVOLAINEN (2010)
Apresenta uma discussão sobre lições aprendidas em XP e Scrum.
- Duas diferentes organizações de grande porte foram o foco do estudo;
- Ambas possuindo mais de 500 desenvolvedores em projetos de software para telefonia móvel;
- Os autores receberam cooperação de unidades organizacionais que aplicaram práticas ágeis, realizando entrevistas informais com os principais representantes dessas organizações;
- Pessoas entrevistadas incluíram promotores, arquitetos e pessoas com responsabilidades no processo.
Práticas ágeis segundo Dikert et al.
Práticas mais evidentes
Desafios
Autores
Questões de pesquisa
Metodologias
Notas Explicativas



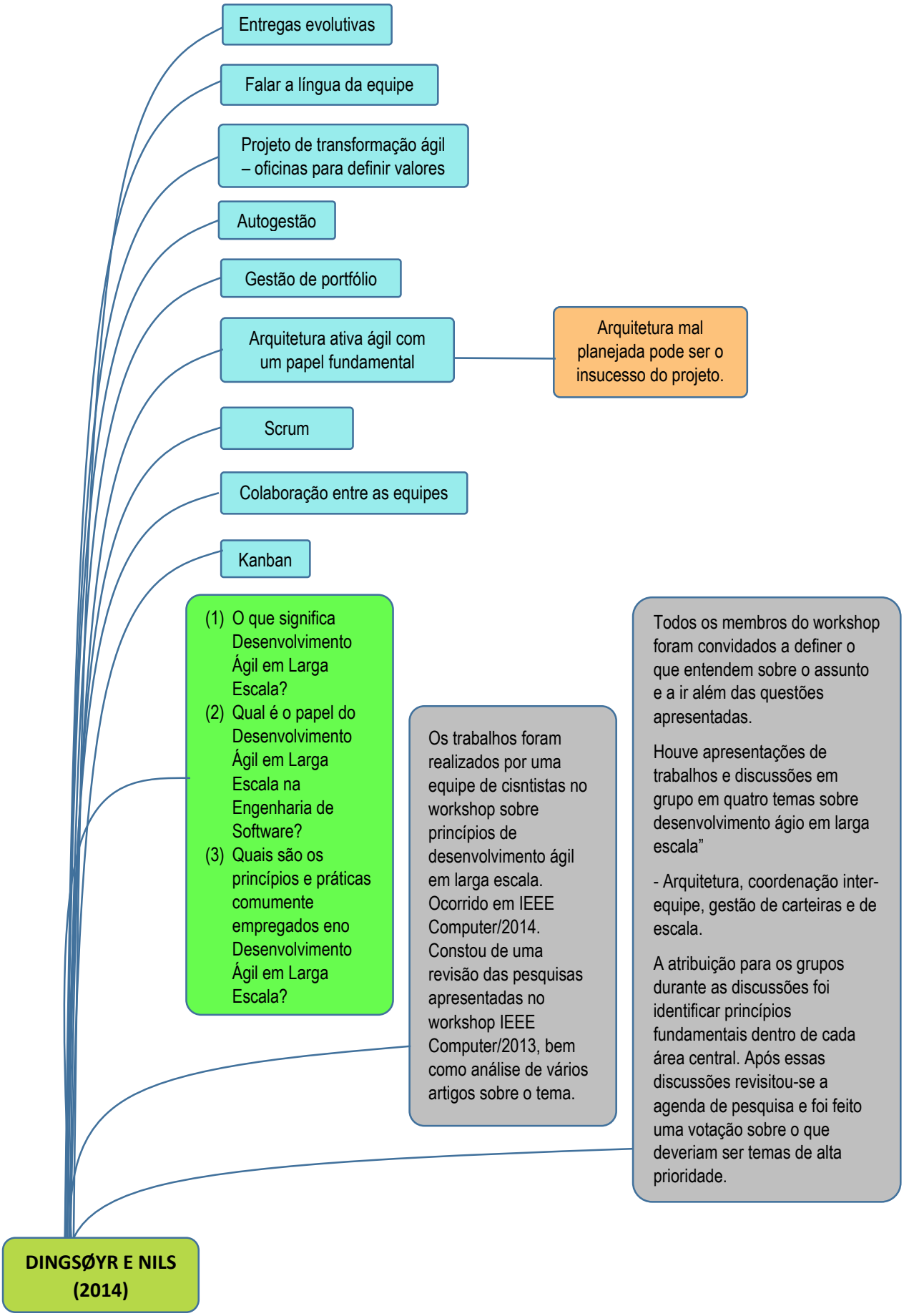

| | |
|---|---|
| | Práticas ágeis segundo Dikert et al. |
| | Práticas mais evidentes |
| | Desafios |
| | Autores |
| | Questões de pesquisa |
| | Metodologias |
| | Notas Explicativas |


Vários backlogs (VISÃO DAS PRÓXIMAS SPRINTS). As equipes de Scrum são organizadas hierarquicamente a um nível superior que tem a responsabilidade de distribuir requisitos do backlog mestre para as equipes.
Product Backlog
BJARNASON (2011)
O Product Backlog é uma lista contendo todas as funcionalidades desejadas. O conteúdo desta lista é definido pelo Product Owner. O Product Backlog não precisa estar completo no início de um projeto, pode-se começar com tudo aquilo que é mais óbvio em um primeiro momento. Com o tempo, o Product Backlog cresce e muda à medida que se aprende mais sobre o produto e seus usuários.
LEFFINGELL (20O7)
Sprint de 30 dias.
Mistura de várias práticas ágeis.
Principal alvo é a entrega rápida.
Uso intensivo de estórias de usuários.
Saber realizar essa divisão pode ser um desafio.
Divisão das estórias de usuários em partes menores e distribuição dessas partes à equipe e Scrum..
O livro não traz questões de pesquisa explícitas, mas apresenta um estudo bastante detalhado sobre metodologias ágeis e suas melhores práticas, que têm alcançado sucesso em diferentes indústrias, sendo referência para centenas de estudos nesta área.
Práticas ágeis segundo Dikert et al.
Práticas mais evidentes
Desafios
Autores
Questões de pesquisa
Metodologias
Notas Explicativas


Desenvolvedores sentem-se mais satisfeitos com metodologia ágil.
Foco no humano é um dos fatores de sucesso.
Estímulo à conversa e relacionamento interpessoal.
(1) O que se sabe sobre os benefícios e limitações de desenvolvimento ágil de software?
(2) Qual é a força da evidência em apoio desses achados?
(3) Quais são as implicações desses estudos para o software, à indústria e a comunidade científica?
Programação em pares pode ser cansativa.
Programação em pares.
DYBA (2008)
Cliente junto, em algumas situações pode ser estressante.
XP aumenta a produtividade.
Cliente junto, comprometido e colaborativo.
Prioridade 1: entregas imediatas.
Ênfase na performance do sistema e não na legibilidade do código.
Refatoração.
A fase 1 utilizou uma planilha de Excel para registrar as fontes de cada citação.
Na fase 2 os autores se sentaram juntos e revisaram todos os títulos que resultaram da etapa 1, para determinar sua relevância para a revisão sistemática.
Eleitos 36 artigos
✓ Revisão sistemática da literatura;
✓ Inclui estudos qualitativos e quantitativos, publicados até 2005;
✓ Os estudos elegíveis para inclusão foram os que apresentaram dados empíricos de desenvolvimento ágil com um limiar mínimo de qualidade;
✓ "Lições aprendidas", artigos sem uma questão de pesquisa e os meramente com base em opinião de especialistas foram excluídos.
Na fase 3 foram excluídos os estudos de o foco não era desenvolvimento ágil.
Editoriais, prefácios, entrevistas, notícias, analises, correspondências, discussões, comentários, tutoriais, workshops, painéis e sessões de pôsteres foram excluídos. Esta estratégia de busca resultou em um total de 2946 "hits" que incluiu 1996 citações não repetidas.
Práticas ágeis segundo Dikert et al.
Práticas mais evidentes
Desafios
Autores
Questões de pesquisa
Metodologias
Notas Explicativas




| | |
|---|---|
| | Práticas ágeis segundo Dikert et al. |
| | Práticas mais evidentes |
| | Desafios |
| | Autores |
| | Questões de pesquisa |
| | Metodologias |
| | Notas Explicativas |

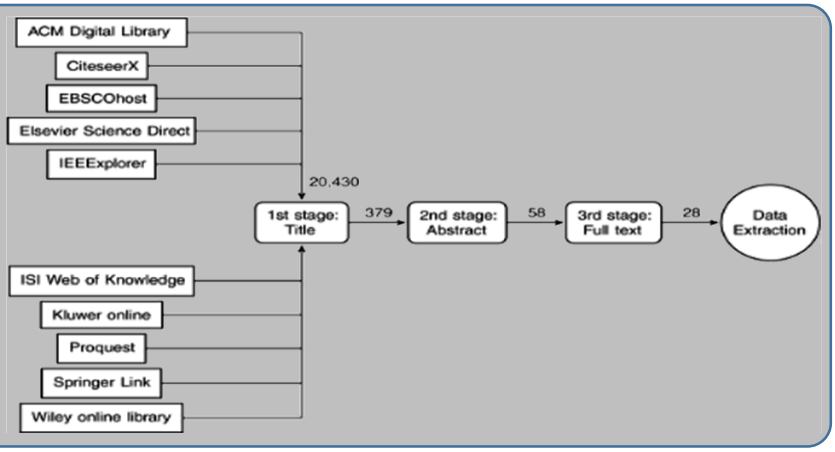

Entregas evolutivas
Falar a língua da equipe
Projeto de transformação ágil – oficinas para definir valores
Autogestão
Gestão de portfólio
Arquitetura ativa ágil com um papel fundamental
Arquitetura mal planejada pode ser o insucesso do projeto.
Scrum
Colaboração entre as equipes
Kanban
(1) O que significa Desenvolvimento Ágil em Larga Escala?
(2) Qual é o papel do Desenvolvimento Ágil em Larga Escala na Engenharia de Software?
(3) Quais são os princípios e práticas comumente empregados eno Desenvolvimento Ágil em Larga Escala?
Os trabalhos foram realizados por uma equipe de cisntistas no workshop sobre princípios de desenvolvimento ágil em larga escala. Ocorrido em IEEE Computer/2014. Constou de uma revisão das pesquisas apresentadas no workshop IEEE Computer/2013, bem como análise de vários artigos sobre o tema.
Todos os membros do workshop foram convidados a definer o que entendem sobre o assunto e a ir além das questões apresentadas.
Houve apresentações de trabalhos e discussões em grupo em quatro temas sobre desenvolvimento ágio em larga escala”
- Arquitetura, coordenação inter-equipe, gestão de carteiras e de escala.
A atribuição para os grupos durante as discussões foi identificar princípios fundamentais dentro de cada área central. Após essas discussões revisitou-se a agenda de pesquisa e foi feito uma votação sobre o que deveriam ser temas de alta prioridade.
DINGSØYR E NILS (2014)
Práticas ágeis segundo Dikert et al.
Práticas mais evidentes
Desafios
Autores
Questões de pesquisa
Metodologias
Notas Explicativas

Revisão da literatura e mapeamento.

A pesquisa foi dividida em três fases. Na primeira, os motores de busca foram divididos entre três autores e os estudos foram incluídos com base no título dos artigos.

A seleção foi feita a partir do campo de desenvolvimento ágil de sistemas embarcados.

De 20.430 visitas, só foram selecionados 379 para a segunda fase.



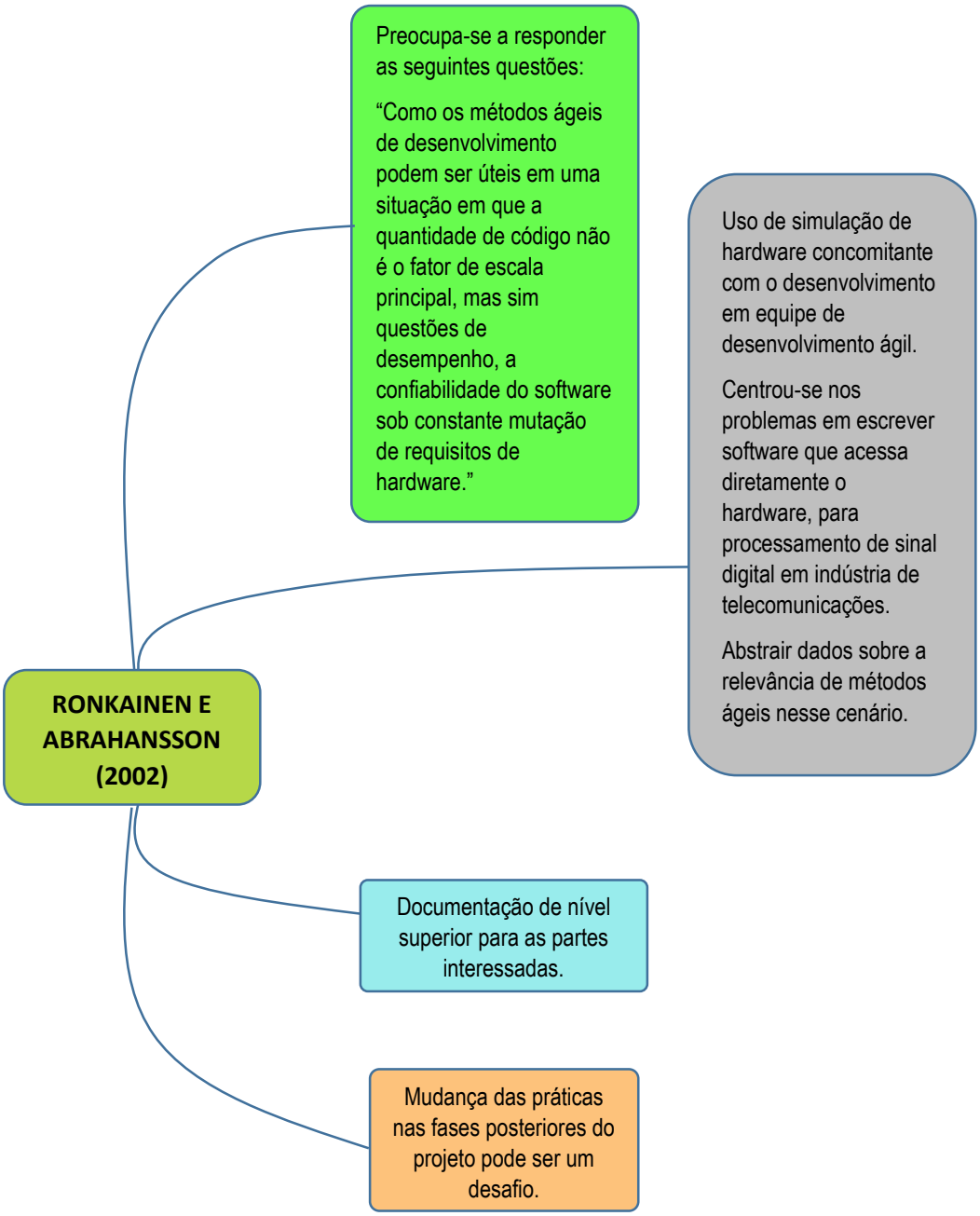

(1) O que se sabe atualmente sobre métodos ágeis em sistemas embarcados e desenvolvimento de software embarcado?
(2) São os métodos ágeis adequados para sistemas embarcados?
(3) Que tipo de evidência existe para estas conclusões?

**KAISTY ET AL. (2013)**

- Mais software e menos documentação
- Presença de desenvolvedores de nível senior.
- XP e Scrum são utilizados juntos.
- Customizar um projeto (uma arquitetura) e não criar um novo partindo do zero.
  - Determinar a funcionalidade que é partilhada entre projetos é um desafio.
- Indivíduos e interações sobre processos e ferramentas.
- Projeto piloto em três etapas:
  - Vender a ideia;
  - Preparação;
  - Execução.
- Monitoramento de defeitos
  - Ambiente de testes
- TDD
  - Problemático, devido a certas limitações.
- Desenvolvimento iterativo
- Simulação
  - Ambiente de homologação
- Estórias de usuários.
- Pesquisa sobre desenvolvimento de código aberto.
- Ciclos curtos de desenvolvimento.

**RONKAINEN E ABRAHANSSON (2002)**

| | |
|---|---|
| | Práticas ágeis segundo Dikert et al. |
| | Práticas mais evidentes |
| | Desafios |
| | Autores |
| | Questões de pesquisa |
| | Metodologias |
| | Notas Explicativas |




| | |
|---|---|
| | Práticas ágeis segundo Dikert et al. |
| | Práticas mais evidentes |
| | Desafios |
| | Autores |
| | Questões de pesquisa |
| | Metodologias |
| | Notas Explicativas |

# Apêndice D

# Seleção das Referências - Metodologia do Estudo Terciário

**ESTUDO TERCIÁRIO: SELEÇÃO DE ARTIGOS**
**CROQUI DA METODOLOGIA UTILIZADA**
**(Tore Dyba˚ , Torgeir Dingsøyr)**

1

| **1ª Fase: DEFINIÇÃO**<br>**68 Referências Iniciais** | **2ª Fase: SELEÇÃO**<br>**15 Referências selecionas**<br><br>**3ª Fase: ESTUDO ( * )**<br>**9 Referências finais** |
|---|---|
| Empirical studies of agile software development: A systematic review<br>(Tore Dyba˚, Torgeir Dingsøyr) | P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile software development methods: review and analysis, VTT Technical report, 2002. |
| | P. Abrahamsson, J. Warsta, M.T. Siponen, J. Ronkainen, New directions on agile methods: a comparative analysis, in: Proceedings of the 25th International Conference on Software Engineering (ICSE'03), IEEE Press, 2003. |
| | K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000, ISBN 0-201-61641-6. |
| | K. Beck, Extreme Programming Explained: Embrace Chage, second ed., Addison-Wesley, 2004, ISBN 978-0321278654. |
| | B. Boehm, Get ready for agile methods, with care, IEEE Computer 35 (1) (2002) 64–69. |
| | A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley, 2004, ISBN 0-201-69947-8. |
| | K. Conboy, B. Fitzgerald, Toward a conceptual framework of agile methods: a study of agility in different disciplines, in: Proceedings of XP/Agile Universe, Springer Verlag, 2004. |
| | G. Melnik, F. Maurer, A cross-program investigation of student's perceptions of agile methods, in: International Conference on Software Engineering (ICSE), St. Louis, MI, USA, 2005. |
| | T. Dyba˚, Improvisation in small software organizations, IEEE Software 17 (5) (2000) 82–87. |
| | R. Baskerville, B. Ramesh, L. Levine, J. Pries-Heje, S. Slaughter, Is internet-speed software development different? IEEE Software 20(6) (2003) 70–77. |
| | M. Poppendieck, T. Poppendieck, Lean Software Development – An Agile Toolkit for Software Development Managers, Addison-Wesley, Boston, 2003, ISBN 0-321-15078-3. |
| | K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, 2001. |
| | P. A ˚ gerfalk, B. Fitzgerald, Flexible and distributed software processes: old petunias in new bowls? Communications of the ACM 49 (10) (2006) 27–34. |
| | G. Melnik, F. Maurer, Perceptions of agile practices: a student survey, in: Proceedings, eXtreme Programming/Agile Universe 2002, Lecture Notes in Computer Science, vol. 2418, Springer Verlag, 2002, pp. 241–250. |
| | S.R. Palmer, J.M. Felsing, A Practical Guide to Feature-driven Development, Prentice Hall, Upper Saddle River, NJ, 2002, ISBN 0-13-067615-2. |

Selecionados para o estudo da 3ª Fase.

( * ) Ao aprofundar o estudo, foram extraídas novas citações (citações de citações), que desencadeou novos estudos não mostrados na tabela, mas referenciados no estudo terciário e nos mapas mentais.

**ESTUDO TERCIÁRIO: SELEÇÃO DE ARTIGOS**
**CROQUI DA METODOLOGIA UTILIZADA**
**(Dikert et al.)** 2

| **1ª Fase: DEFINIÇÃO 53 Referências** | **2ª Fase: SELEÇÃO 37 Referências selecionadas**<br><br>**3ª Fase: ESTUDO ( * ) 16 Referências finais** |
|---|---|
| Challenges and success factors for large-scale agile transformations: A systematic literature review (Dikert et al.) | Beck, K. , 1999. Embracing change with extreme programming. |
| | Bjarnason, E. , Wnuk, K. , Regnell, B. , 2011. A case study on benefits and side-effects of agile practices in large-scale. |
| | Boehm, B. , 2002. Get ready for agile methods, with care. |
| | Cockburn, A., Highsmith, J., 2001. Agile software development, the people factor. |
| | Dingsøyr, T., Fægri, T., Itkonen, J., 2014. What is large in large-scale? a taxonomy of scale for agile software delopment. |
| | Dingsøyr, T., Moe, N., 2014. Towards principles of large-scale agile development. |
| | Hodgkins, P. , Hohmann, L. , 2007. Agile program management: Lessons learned from the verisign managed security services team. |
| | Fowler, M., Highsmith, J., The agile manifesto (2001). http://www.agilemanifesto. org/ . |
| | Koehnemann, H. , Coats, M. , 2009. Experiences applying agile practices to large systems. |
| | Kaisti, M. , Rantala, V. , Mujunen, T. , Hyrynsalmi, S. , Konnola, K. , Makila, T. , Lehto- nen, T. , 2013. Agile methods for embedded systems development - a literature review and a mapping study. |
| | Kitchenham, B.A. , 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical.. |
| | Miller, J. , Haddad, H. , 2012. Challenges faced while simultaneously implementing cmmi and scrum: a case study in the tax preparation software industry. |
| | Nerur, S. , Mahapatra, R. , Mangalaraj, G. , 2005. Challenges of migrating to agile methodologies. |
| | Paasivaara, M., Behm, B., Lassenius, C., Hallikainen, M., 2014. Towards rapid releases in large-scale xaas development at ericsson: A case study. |
| | Paasivaara, M., Lassenius, C., Heikkila, V., Dikert, K., Engblom, C., 2013. Integrating global sites into the lean and agile transformation at ericsson. |
| | Schwaber, K. , Beedle, M. , 2002. Agile Software Development with Scrum. |
| | Scott, J. , Johnson, R. , McCullough, M. , 2008. Executing agile in a structured organization: government. |
| | Williams, L. , Cockburn, A. , 2003. Agile software development: it's about feedback and change. |
| | Nerur, S. , Mahapatra, R. , Mangalaraj, G. , 2005. Challenges of migrating to agile methodologies. |
| | Paasivaara, M. , Durasiewicz, S. , Lassenius, C. ,2008. Using scrum in a globally distributed project: a case study. |
| | Petersen, K. , Wohlin, C. , 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices. Emp. |
| | Dybå, T. , Dingsøyr, T. , 2009. What do we know about agile software development? |
| | Senapathi, M. , Srinivasan, A. , 2013. Sustained agile usage: a systematic literature review. |
| | Giudice, D.L. , Kisker, H. , Angel, N. , 2014. How Can You Scale Your Agile Adoption? |
| | Dybå, T. , Dingsøyr, T. , 2008. Empirical studies of agile software development: a systematic review. . |
| | Chow, T., Cao, D.-B., 2008. A survey study of critical success factors in agile software projects. J. Syst. Softw. |
| | Lyon, R. , Evans, M. , 2008. Scaling up –pushing scrum out of its comfort zone. |
| | Highsmith, J. , Cockburn, A. , 2001. Agile software development: the business of innovation. |
| | Lindvall, M. , Muthig, D. , Dagnino, A. , Wallin, C. , Stupperich, M. , Kiefer, D. , May, J. , Kahkonen, T. , 2004. Agile software development in large organizations. |
| | Fowler, M. , 20 0 0. Put your process on a diet. |
| | Cruzes, D. , Dyba, T. , 2011. Recommended steps for thematic synthesis in software engineering. |
| | Elshamy, A. , Elssamadisy, A. , 2006. Divide after you conquer: an agile software development practice for large projects. |
| | Cohn, M. , Ford, D. , 2003. Introducing an agile process to an organization [software development].. |
| | Cloke, G. , 2007. Get your agile freak on! agile adoption at yahoo! music. |
| | Berger, H. , Beynon-Davies, P. , 2009. The utility of rapid application development in large-scale, complex projects. Inf. Syst. J. 19 (6), 549–570 . |
| | Boehm, B. , Turner, R. , 2005. Management challenges to implementing agile pro- cesses in traditional development organizations. Softw. IEEE 22 (5), 30–39 . |
| | Cloke, G. , 2007. Get your agile freak on! agile adoption at yahoo! music. In: Agile Conference (AGILE), 2007, pp. 240–248 . |

Selecionados para o estudo da 3ª Fase.

( * ) Ao aprofundar o estudo, foram extraídas novas citações (citações de citações), que desencadeou novos estudos não mostrados na tabela, mas referenciados no estudo terciário e nos mapas mentais.

# Apêndice E

# Questionário de Pesquisa de Campo - Versão Final

# Evidência de Práticas Ágeis nas Empresas de Desenvolvimento de Software

*Obrigatório

1. **Endereço de e-mail** *

_____

2. **Qual a sua área de atuação na empresa?** *
*Marcar apenas uma oval.*

- ◯ Supervisor
- ◯ Administrador
- ◯ Gerente
- ◯ Arquiteto
- ◯ Analista
- ◯ Testador
- ◯ Suporte
- ◯ Desenvolvedor
- ◯ Webdesigner
- ◯ Outro: _____

3. **Qual o nível da sua formação?** *
*Marcar apenas uma oval.*

- ◯ Pós-Doutorado
- ◯ Doutorado
- ◯ mestrado
- ◯ graduado
- ◯ nível médio

4. **Quanto tempo de experiência em projetos de desenvolvimento de software em geral?** *
*Marcar apenas uma oval.*

- ◯ Entre 1 e 2 anos
- ◯ Entre 3 e 5 anos
- ◯ Entre 5 e 8 anos
- ◯ Entre 8 e 12 anos
- ◯ Entre 13 e 16 anos
- ◯ Entre 17 e 20 anos
- ◯ Mais de 20 anos

5. **Qual metodologia ágil sua equipe mais adotou (ou adota) em processos de desenvolvimento?** *

*Marcar apenas uma oval.*

- ◯ XP
- ◯ SCRUM
- ◯ KANBAN
- ◯ LEAN
- ◯ FDD
- ◯ ASD
- ◯ Minha Equipe nunca trabalhou com metodologias ágeis
- ◯ Outro: _____

6. **Abaixo, relacionamos algumas das principais práticas ágeis de desenvolvimento de software. Quais delas você/sua equipe mais utilizam ou já utilizaram?** *

*Marque todas que se aplicam.*

- ☐ Design Incremental
- ☐ Implantação Diária
- ☐ Integração Contínua
- ☐ Discussão sobre o quadro branco (Kanban)
- ☐ Sentar Junto
- ☐ Reunião em Pé
- ☐ Cliente Junto
- ☐ Código Compartilhado
- ☐ Estórias de Usuários
- ☐ Programação em Pares
- ☐ Prototipação
- ☐ TTD
- ☐ Releases Curtas
- ☐ Entregas de 2 a 4 semanas
- ☐ Outro: _____

7. **Na sua equipe, é possível perceber algum princípio ágil sendo empregado? Qual (is)?** *
*Marque todas que se aplicam.*

- [ ] A grande prioridade é a satisfação dos clientes, por meio de entregas rápidas e contínuas.
- [ ] Abraçar as mudanças de requisitos, mesmo que elas aconteçam tarde.
- [ ] Freqüência das entregas de duas semanas a dois meses, em cada ciclo.
- [ ] O pessoal da área de negócios deve trabalhar juntos com os desenvolvedores.
- [ ] Motivar sempre o pessoal, com o uso do melhor equipamento e da melhor ferramenta.
- [ ] Conversar cara a cara com pessoal.
- [ ] A principal medida de eficácia do time é o software funcionando.
- [ ] Ritmo constante.
- [ ] Atenção contínua.
- [ ] Simplicidade.
- [ ] Times auto-gerenciados.
- [ ] O time pára, vez por outra, para refletir sobre o trabalho, a fim de ajustar as melhores técnicas.
- [ ] Outro: _____

8. **Houve um aumento de produtividade após a implantação de alguma prática ágil? Ouve uma redução do custo de produção?** *
*Marcar apenas uma oval.*

- ◯ Sim
- ◯ Não

9. **Em relação à pergunta anterior, se "Sim", qual foi a prática ágil que mais agregou produtividade?**

_____

10. **Ocorreram problemas em projetos durante a implantação da metodologia ágil na empresa?** *
*Marcar apenas uma oval.*

- ◯ Sim
- ◯ Não

11. **Em relação à pergunta anterior, se a resposta foi "Sim", o Sr. poderia nos relatar quais foram os problemas, de forma sucinta?**

_____

_____

_____

_____

_____

12. **Quais dessas Práticas Ágeis são ou foram aplicadas em sua equipe de desenvolvimento? Indique a sua percepção dessas práticas, enquanto participante ativo do projeto, conforme as assertivas abaixo:**

| Dimensão | Descrição |
|---|---|
| 1-Sem importância | Indica que a prática ágil não foi percebida pelo entrevistado. |
| 2-Pouco importante | Indica que a prática ágil foi pouco percebida e que a mesma não contribuiu muito para o sucesso do projeto e que, por vezes, foi ofuscada. |
| 3-Importante | Indica que a prática foi percebida pelo participante de forma positiva e disseminada, contribuindo razoavelmente para a melhoria do software. |
| 4-Muito importante | Indica que a prática foi bastante percebida, contribuindo muito para a melhoria dos processos, a diminuição de custos e a redução de defeitos. |
| 5-Extremamente importante | Indica que a prática foi extremamente percebida, contribuindo de forma preponderante para o sucesso do projeto. |

*Marcar apenas uma oval por linha.*

| | Sem Importância | Pouco Importante | Importante | Muito Importante | Extremamente Importante |
|---|---|---|---|---|---|
| Ampliar conhecimento | ◯ | ◯ | ◯ | ◯ | ◯ |
| Comunicação | ◯ | ◯ | ◯ | ◯ | ◯ |
| Design incremental | ◯ | ◯ | ◯ | ◯ | ◯ |
| Eliminação de desperdícios | ◯ | ◯ | ◯ | ◯ | ◯ |
| Gerência continuada | ◯ | ◯ | ◯ | ◯ | ◯ |
| Gráfico Burndown | ◯ | ◯ | ◯ | ◯ | ◯ |
| Implantação diária | ◯ | ◯ | ◯ | ◯ | ◯ |
| Integração contínua | ◯ | ◯ | ◯ | ◯ | ◯ |
| Iterativo e incremental | ◯ | ◯ | ◯ | ◯ | ◯ |
| Jogo do planejamento | ◯ | ◯ | ◯ | ◯ | ◯ |
| Minimalismo | ◯ | ◯ | ◯ | ◯ | ◯ |
| MosCoW | ◯ | ◯ | ◯ | ◯ | ◯ |
| Pagar por uso | ◯ | ◯ | ◯ | ◯ | ◯ |
| Plano de mitigação de riscos | ◯ | ◯ | ◯ | ◯ | ◯ |
| Quadro de tarefas | ◯ | ◯ | ◯ | ◯ | ◯ |
| Visibilidade | ◯ | ◯ | ◯ | ◯ | ◯ |
| Discussão sobre o quadro branco | ◯ | ◯ | ◯ | ◯ | ◯ |
| Time com poder de decisão | ◯ | ◯ | ◯ | ◯ | ◯ |
| Time completo | ◯ | ◯ | ◯ | ◯ | ◯ |
| Sentar junto | ◯ | ◯ | ◯ | ◯ | ◯ |
| Reunião em pé | ◯ | ◯ | ◯ | ◯ | ◯ |
| Área de trabalho informativa | ◯ | ◯ | ◯ | ◯ | ◯ |
| Equipes que encolhem | ◯ | ◯ | ◯ | ◯ | ◯ |
| Continuidade da equipe | ◯ | ◯ | ◯ | ◯ | ◯ |
| Aprender fazendo | ◯ | ◯ | ◯ | ◯ | ◯ |
| Cliente Junto | ◯ | ◯ | ◯ | ◯ | ◯ |

| | Sem Importância | Pouco Importante | Importante | Muito Importante | Extremamente Importante |
|---|---|---|---|---|---|
| Desenvolvimento orientado por características das funcionalidades | | | | | |
| Estórias de usuários | | | | | |
| Modelagem em objetos de domínio | | | | | |
| Programação em pares | | | | | |
| Propriedade coletiva | | | | | |
| Prototipação | | | | | |
| Refatoração | | | | | |
| TDD | | | | | |
| Testes Integrados | | | | | |
| Retrospectiva ao término do ciclo | | | | | |
| Entrega de 2 a 4 semanas | | | | | |
| 40 horas semanais | | | | | |
| Estimativas | | | | | |
| Satisfação total dos cliente | | | | | |

13. **Quais são os pontos fortes e pontos fracos de se utilizar metodologia ágil como método de desenvolvimento na visão da empresa, pela sua perspectiva?**

14. **Você poderia nos dizer quais foram seus principais desafios, enquanto trabalha/trabalhava com práticas ágeis?**

# Apêndice F

# Programa R: Scripts de plotagens

```
#********
#SCRIPT 1
#********
#-----------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 2
# 2 - Qual a sua area de Atuacao na empresa?
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
#-----------------------------------------
# Orientador: Dr. Rodrigo Bonifácio

#==============================
#Instalando pacotes necessarios
install.packages('descr')
#==============================

#Disponibilizando ferramentas
require("descr")
require("plyr")
require("mdatools")
require("tools")
require("stats")
require("graphics")
require("utils")
require("datasets")
require("methods")
require("base")
require("reshape")
#-----------------------------------------

# Ler os dados do arquivo - localizar o arquivo q2_barplot.txt
items <- read.table(file.choose(),header=TRUE)

# Sumarizando os dados
sumario <- factor(rep(items$Q2),levels=c("Administrador","Analista", "Arquiteto",
"Desenvolvedor", "Gerente", "Suporte"))

# Realizando a contagem da porcentagem de cada tipo de participante (P)
# e ordenando por quantidades de cada tipo
quantOrd <- sort(table(sumario),decreasing=TRUE)
quantOrdPorc <- prop.table(quantOrd) * 100
quantOrdPorc

#Formatacao do plot
par(cex=0.8, family="serif") #.............tamanho da fonte
par(bg="white") #.........................cor do fundo
par(mar=c(10,5,5,5)) #...................margens(botom,left,top,right)

# Criacao do plot
barplot(quantOrdPorc, las=3)

# titulo eixo y
mtext(iconv("%", to="UTF-8", from="latin1"), side=2, line=1, at=38,cex=0.7,
family="serif")

# titulo do grafico (deve ou nao ser suprimido?)
# side:3=top,4=bottom,2=left,3=right
# line:posicionamento horizontal
# at:posicionamento vertical
mtext(iconv("CARGOS OCUPADOS PELOS PARTICIPANTES", to="UTF-8", from="latin1"),
side=3, line=2, at=3,cex=0.8, family="serif")

#removendo objetos
rm(items) rm(quantOrd) rm(quantOrdPorc) rm(sumario)
#---
#FIM
#---
```

```
#********
#SCRIPT 2
#********
#-----------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 4
# 4 - Quanto tempo de experiencia em projetos de software em geral?
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-----------------------------------------

#Disponibilizando ferramentas
require("descr")
require("plyr")
require("mdatools")
require("tools")
require("stats")
require("graphics")
require("utils")
require("datasets")
require("methods")
require("base")
require("reshape")
#-----------------------------------------

# Ler os dados do arquivo - localizar o arquivo q4_barplot.txt
items <- read.table(file.choose(),header=TRUE)

# Sumarizando os dados
sumario <- factor(rep(items$Q4),levels=c("Entre 1 e 2 Anos","Entre 3 e 5 Anos",
"Entre 6 e 8 Anos", "Entre 9 e 12 Anos", "Entre 13 e 16 Anos", "Entre 17 e 20
Anos", "Mais de 20 Anos"))

# Realizando a contagem da porcentagem de cada tipo de participante (P)
# e ordenando por quantidades de cada tipo
quantPorc <- prop.table(table(sumario)) * 100
quantPorc

#Formatacao do plot
par(cex=0.8, family="serif") #.............tamanho da fonte
par(bg="white") #.........................cor do fundo
par(mar=c(10,5,5,5)) #...................margens(botom,left,top,right)

# Criacao do plot
mp <- barplot(quantPorc, las=3, ylim = c(0, 40))

# Draw the bar values above the bars
text(mp, quantPorc, labels = format(quantPorc, 7, digits = 4), pos = 3, cex = .55)

# titulo eixo y
mtext(iconv("%", to="UTF-8", from="latin1"), side=2, line=1, at=28,cex=0.7,
family="serif")

# titulo do grafico (deve ou nao ser suprimido?)
# side:3=top,4=bottom,2=left,3=right
# line:posicionamento horizontal
# at:posicionamento vertical
mtext(iconv("TEMPO DE EXPERIÊNCIA EM ES", to="UTF-8", from="latin1"), side=3,
line=2, at=3,cex=0.8, family="serif")

#removendo objetos
rm(items)
rm(quantPorc)
rm(sumario)
#---
#FIM
#---
```

```
#********
#SCRIPT 3
#********

#-----------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 5
# 5 - Qual metodologia ágil sua equipe mais
#     adotou (ou adota) em processos de
#     desenvolvimento?
#     (Em contraposicao com DYBA e MELO)
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-----------------------------------------

#Disponibilizando ferramentas
library(reshape2)
library(ggplot2)
#-----------------------------------------

# Os dados da pesquisa - confrontando Dyva(2008), Melo(2012) e Mazuco(2017)

#   M        Dyva(2008)    Melo(2012)  Mazuco(2017)
#   XP              76           7.3          16.7
#   Scrum            3          51,1          44,4
#   Outros          21          41.6          38.9

# Os dados de "Outros" em Mazuco(2017) serah calculado em outro script

# Passando os dados da pesquisa para o R
x <- c(76,  7.3, 16.7) #XP
y <- c( 3, 51.1, 44.4) #Scrum
z <- c(21, 41.6, 38.9) #Outros

# Criando as 3 linhas da matriz com x,y,z
height <- rbind(x, y, z)

par(cex=0.8, family="serif") #.............tamanho da fonte

# Plotando
mp <- barplot(height, beside = TRUE, ylim = c(0, 100), main="Evolução das
Metodologias Ágeis 2008 a 2017", names.arg=c("Dyba(2008)", "Melo(2012)",
"Mazuco(2017)"), cex.names=0.7)

# Colocando labels em cada coluna
text(mp, height, labels = format(height, 4), pos = 3, cex = .75)

#Adicionando as legendas
legend(10, 90, c("XP","Scrum","Outros"), fill=gray.colors(3), cex = .75)

#removendo objetos
rm(height)
rm(mp)
rm(x)
rm(y)
rm(z)

#---
#FIM
#---
```

```
#********
#SCRIPT 4
#********
#-------------------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 5
# 5 - Qual metodologia ágil sua equipe mais
#     adotou (ou adota) em processos de
#     desenvolvimento?
#     (Somente "Outros" -
#      outros resultados diferentes de XP r Scrum)
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-------------------------------------------------

# Os "Outros"dados da pesquisa - questao 5
#   M                           Values
#   LEAN                           7.1
#   FDD                           14.3
#   KANBAN                        21.5
#   Minha equipe nunca usou       57.1

# Passando os dados da pesquisa para o R
x <- c(7.1)  #LEAN
y <- c(14.3) #FDD
z <- c(21.5) #KANBAN
w <- c(57.1) #Minha equipe nunca usou

# Criando as 4 linhas da matriz com x,y,z,w
height <- rbind(x, y, z, w)

par(cex=0.8, family="serif") #.............tamanho da fonte

# Plotando
mp <- barplot(height, beside = TRUE, ylim = c(0, 100), main="Outras Metodologias -
Mazuco(2017)", names.arg=c("Values"), cex.names=0.7)

# Colocando o label na coluna
text(mp, height, labels = format(height, 4), pos = 3, cex = .75)

#Adicionando as legendas
legend(3, 100, c("LEAN","FDD","KANBAN", "Minha equipe nunca usou"),
fill=gray.colors(4), cex = .75)

#removendo objetos
rm(height)
rm(mp)
rm(x)
rm(y)
rm(z)
rm(w)

#---
#FIM
#---
```

```
#********
#SCRIPT 5
#********

#-----------------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 12 tabulada
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-----------------------------------------------
#==============================
#Instalando pacotes necessarios
install.packages('likert')
install.packages('mdatools')
install.packages('tools')
install.packages('reshape')
#==============================

#Disponibilizando ferramentas
require("likert")
require("plyr")
require("mdatools")
require("tools")
require("stats")
require("graphics")
require("utils")
require("datasets")
require("methods")
require("base")
require("reshape")
#-----------------------------------------

# Ler os dados do arquivo - localizar o arquivo q12_likert.txt
items <- read.table(file.choose(),header=TRUE)

#Obtendo apenas as colunas necessarias
dados <- items[,substr(names(items), 1,2) == 'PR']
dados1<-dados[,1:40, drop=FALSE]

head(dados1) #Checando os dados
ncol(dados1) #Checando o numero de colunas
nrow(dados1) #Checando o numero de linhas

#Renomeando as colunas (Praticas Ageis)
temp<-rename(dados1, c(
  PR1= 'Ampliar o Conhecimento',
  PR2= 'Comunicação',
  PR3= 'Design Incremental',
  PR4= 'Eliminação de desperdícios',
  PR5= 'Gerência Continuada',
  PR6= 'Gráfico Burndown',
  PR7= 'Implantação Diária',
  PR8= 'Integração Contínua',
  PR9= 'Iterativo e Incremental',
  PR10='Jogo do Planejamento',
  PR11='Minimalismo',
  PR12='MoscoW',
  PR13='Pagar por Uso',
  PR14='Plano de Mitigação de Riscos',
  PR15='Quadro de Tarefas',
  PR16='Visibilidade',
  PR17='Discussão Sobre o Quadro Branco',
  PR18='Timecom Poder de Decisão',
  PR19='Time Completo',
  PR20='Sentar Junto',
  PR21='Reunião em Pé',
  PR22='Área de Trabalho Informativa',
```

```
  PR23='Equipes que Encolhem',
  PR24='Continuidade da Equipe',
  PR25='Aprender Fazendo',
  PR26='Cliente Junto',
  PR27='Desenv. Orientado Carac Funcionalidades',
  PR28='Estórias de Usuários',
  PR29='Modelagem em Objetos de Domínio',
  PR30='Programação em Pares',
  PR31='Propriedade Coletiva',
  PR32='Prototipação',
  PR33='Refatoração',
  PR34='TDD',
  PR35='Testes Integrados',
  PR36='Retrospectiva ao Término do Ciclo',
  PR37='Entrega de 2 a 4 Semanas',
  PR38='40 Horas Semanais',
  PR39='Estimativas',
  PR40='Satisfação Total dos Clientes'
))

#Manipulacao dos dados para a leitura likert
dadosToPlot<-likert(temp,nlevels = 5)

#Exibindo os dados do likert
summary(dadosToPlot, ordered = TRUE)

plot(dadosToPlot)

#removendo objetos
rm(dados)
rm(dados1)
rm(items)
rm(temp)
rm(dadosToPlot)

#---
#FIM
#---
```

```
#********
#SCRIPT 6
#********

#-------------------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 6
# 6 - Abaixo, relacionamos algumas práticas ágeis de
#     desenvolvimento de software. Quais delas você/sua
#     equipe mais utilizam hoje?
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-------------------------------------------------

# Praticas que as industrias utilizam (quantidades)
# Universo de 36 participantes (P)
# Dados extraidos da questao 6
# PR1 PR2 PR3 PR4 PR5 PR6 PR7 PR8 PR9 PR10  PR11  PR12  pr13  pr14
#   9  12    19    14    19    20    21    16    13     11    19     5     9
        14

# Passando os dados da pesquisa e transformando-os em porcentagem
PR1   <- c((9*100)/36)  #Design Incremental
PR2   <- c((12*100)/36) #Implantação Diária
PR3   <- c((19*100)/36) #Integração Continua
PR4   <- c((14*100)/36) #Discussao sobre um Quadro Branco
PR5   <- c((19*100)/36) #Sentar Junto
PR6   <- c((20*100)/36) #Reunião em Pé
PR7   <- c((21*100)/36) #Cliente Junto
PR8   <- c((16*100)/36) #Código Compartilhado
PR9   <- c((13*100)/36) #Estorias de Usuarios
PR10  <- c((11*100)/36) #Programacao em Pares
PR11  <- c((19*100)/36) #Prototipação
PR12  <- c((5*100)/36)  #TDD
PR13  <- c((9*100)/36)  #Releases Curtas
PR14  <- c((14*100)/36) #Entrega de 2 a 4 Semanas


# Criando as linhas da matriz
height <- rbind(PR1, PR2, PR3, PR4, PR5, PR6, PR7, PR8, PR9, PR10, PR11, PR12,
PR13, PR14)
h <- sort(height, decreasing = TRUE)
par(cex=0.85, family="serif", par(las=2)) #.............tamanho da fonte
par(mar=c(5,12,4,2)) # increase y-axis margin.

# Plotando - colocando o label ordenado
mp <- barplot(h,
              horiz=TRUE,
              beside = TRUE,
              xlim = c(0, 70),
              ylim = c(0, 18),
              main="Práticas Ágeis Investigadas (Questão 6)",
              names.arg=c(
                "Prototipação",
                "Cliente Junto",
                "Reunião em Pé",
                "Sentar Junto",
                "Integração Continua",
                "Código Compartilhado",
                "Entrega de 2 a 4 Semanas",
                "Discussao sobre um Quadro Branco",
                "Estorias de Usuarios",
                "Implantação Diária",
                "Programacao em Pares",
                "Design Incremental",
                "Releases Curtas",
                "TDD"
```

```
              ), cex.names=0.85)

# Colocando o label na coluna
text(x=65,mp, offset = -0.30, h, labels = format(h, 3, digits = 4), pos = 3, cex =
.75)

# titulo do grafico (deve ou nao ser suprimido?)
mtext(iconv("Porcenteagem"), side=2, line=-10, at=-5,cex=0.85, family="serif")


#removendo objetos
rm(height)
rm(mp)
rm(PR1)
rm(PR2)
rm(PR3)
rm(PR4)
rm(PR5)
rm(PR6)
rm(PR7)
rm(PR8)
rm(PR9)
rm(PR10)
rm(PR11)
rm(PR12)
rm(PR13)
rm(PR14)

#---
#FIM
#---
```

```
#********
#SCRIPT 7
#********

#-------------------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 7
# 7 - Na sua equipe, é possível perceber algum princípio
#     ágil sendo empregado, quais?
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-------------------------------------------------

# Praticas que as industrias utilizam (quantidades)
# Universo de 36 participantes (P)
# Dados extraidos da questao 7
# I1 I2 I3 I4 I5 I6 I7 I8 I9 I10  I11  I12
# 29 18 17 18 17 26 16 15 18  25   9     17

# Passando os dados da pesquisa e transformando-os em porcentagem
I11  <- c(( 9*100)/36) #Times autogerenciados
I8   <- c((15*100)/36) #Ritmo constante
I7   <- c((16*100)/36) #A principal medida de eficácia do time é o software
I3   <- c((17*100)/36) #Freqüência das entregas de duas semanas a dois meses, em
cada ciclo
I5   <- c((17*100)/36) #Motivar sempre o pessoal, com o uso do melhor equipamento e
da melhor ferramenta
I12  <- c((17*100)/36) #O time pára, vez por outra, para refletir sobre o trabalho,
a fim de ajustar as melhores técnicas
I2   <- c((18*100)/36) #Abraçar as mudanças de requisitos, mesmo que elas aconteçam
tarde
I4   <- c((18*100)/36) #O pessoal da área de negócios deve trabalhar juntos com os
desenvolvedores
I9   <- c((18*100)/36) #Atenção contínua
I10  <- c((25*100)/36) #Simplicidade
I6   <- c((26*100)/36) #Conversar face a face com pessoal
I1   <- c((29*100)/36) #A grande prioridade é a satisfação dos clientes, por meio
de entregas rápidas e contínuas


# Criando as linhas da matriz
height <- rbind(I1, I2, I3, I4, I5, I6, I7, I8, I9, I10, I11, I12)
h <- height #sort(height, decreasing = TRUE)
par(cex=0.85, family="serif", par(las=2)) #.............tamanho da fonte
par(mar=c(18,5,4,2)) # increase y-axis margin.

# Plotando - colocando o label ordenado
mp <- barplot(h,
              horiz=FALSE,
              beside = TRUE,
              xlim = c(0, 14.5),
              ylim = c(0, 100),
              names.arg=c(
                "I1",
                "I2",
                "I3",
                "I4",
                "I5",
                "I6",
                "I7",
                "I8",
                "I9",
                "I10",
                "I11",
                "I12"
              ), cex.names=0.85)
```

```
# Colocando labels em cada coluna
text(mp, height, labels = format(height, 4, digits = 4), pos = 3, cex = .75)

#Adicionando as legendas
legend(1,-30,c("I1 : A grande prioridade é a satisfação dos clientes",
                "I2 : Abraçar as mudanças de requisitos, mesmo que elas aconteçam
tarde",
                "I3 : Entregas de duas semanas a dois meses, em cada ciclo",
                "I4 : O pessoal de negócios trabalhando junto com
desenvolvedores",
                "I5 : Motivar o pessoal com o uso do melhor equipamento",
                "I6 : Conversar face a face com pessoal",
                "I7 : A principal medida de eficácia do time é o software",
                "I8 : Ritmo constante",
                "I9 : Atenção contínua",
                "I10: Simplicidade",
                "I11: Times autogerenciados",
                "I12: O time pára para refletir, a fim de ajustar as melhores
técnicas"), xpd = TRUE, cex = 1.15)

#removendo objetos
rm(height)
rm(mp)
rm(I1)
rm(I2)
rm(I3)
rm(I4)
rm(I5)
rm(I6)
rm(I7)
rm(I8)
rm(I9)
rm(I10)
rm(I11)
rm(I12)
rm(I13)
rm(I14)

#---
#FIM
#---
```

```
#********
#SCRIPT 8
#********
#-----------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 2
# 8 - Houve um aumento de produtividade após
#     a implantação de alguma prática ágil?
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-----------------------------------------

#==============================
#Instalando pacotes necessarios
install.packages('plotrix')
#==============================

#Disponibilizando ferramentas
library(plotrix)

par(family="serif", par(las=2)) #.............tamanho da fonte
slices <- c(30.5, 69.45)
lbls <- c("Não", "Sim")
pie3D(col = c("gray","white"), slices,labels=lbls,explode=0.05, labelcex=0.85)
#---
#FIM
#---
```

```
#********
#SCRIPT 9
#********
#-----------------------------------------
# PERCEPCAO E PADOCAO DE RATICAS AGEIS
# plot baseado nos dados da Questao 10
# 10 - Ocorreram problemas em projetos durante
#     a implantação da metodologia ágil na
#     empresa?
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-----------------------------------------

#==============================
#Instalando pacotes necessarios
install.packages('plotrix')
#==============================

#Disponibilizando ferramentas
library(plotrix)

par(family="serif", par(las=2)) #.............tamanho da fonte
sim <- 63.89
nao <- 36.11

slices <- c(nao, sim)
lbls <- c("Não", "Sim")
pie <- pie3D(col = c("gray","white"), slices,labels=lbls,explode=0.05,
labelcex=0.85)

# Colocando labels
mtext(iconv(nao, to="UTF-8", from="latin1"), side=4, line=-5, at=0.3,cex=0.8,
family="serif")
mtext(iconv(sim, to="UTF-8", from="latin1"), side=4, line=-12, at=0.1,cex=0.8,
family="serif")

#removendo objetos
rm(lbls)
rm(nao)
rm(sim)
rm(pie)
rm(slices)
#---
#FIM
#---
```

```
#*********
#SCRIPT 10
#*********
#-----------------------------------------------
# PERCEPCAO E ADOCAO PRATICAS AGEIS
# plot baseado nos dados da Questao 10
# 13 - Ponto forte em utilizar metodologia ágil?
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-----------------------------------------------

#==============================
#Instalando pacotes necessarios
install.packages('fmsb')
#==============================

# Library
library(fmsb)

# Create data
A <- matrix(c(1, 7, 2, 7, 7, 1, 1), nrow=1, ncol=7, byrow = TRUE)
data=as.data.frame(A)
colnames(data)=c("PR1" , "PR2" , "PR3" , "PR4" , "PR5", "PR6" , "PR7")

# To use the fmsb package, I have to add 2 lines to the dataframe: the max and min
of each topic to show on the plot!
data=rbind(rep(10,5) , rep(0,5) , data)

# The default radar chart proposed by the library:
radarchart(data)

# Custom the radarChart !
radarchart( data  , axistype=2 ,

            #custom polygon
            pcol=rgb(0.2,0.5,0.5,0.9) , pfcol=rgb(0.2,0.5,0.5,0.5) , plwd=4 ,

            #custom the grid
            cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,10,5),
cglwd=0.8,

            #custom labels
            vlcex=0.7
)

#removendo objetos
rm(A)
rm(data)
#---
#FIM
#---
```

```
#*********
#SCRIPT 11
#*********
#----------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 10
# 13 - Ponto fraco em utilizar metodologia ágil?
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#----------------------------------------

#==============================
#Instalando pacotes necessarios
install.packages('fmsb')
#==============================

# Library
library(fmsb)
par(mar=c(9,2,3,5)) #.....................margens(botom,left,top,right)
# Create data
A <- matrix(c(1, 1, 2, 2, 4, 5), nrow=1, ncol=6, byrow = TRUE)
data=as.data.frame(A)
colnames(data)=c("Time com Poder de Decisão" , "Estimativas" , "Estórias de
Usuários" , "Equipes que Encolhem" , "Pagar por Uso", "Falta de conhecimento")

# To use the fmsb package, I have to add 2 lines to the dataframe: the max and min
of each topic to show on the plot!
data=rbind(rep(10,5) , rep(0,5) , data)

# The default radar chart proposed by the library:
radarchart(data)

# Custom the radarChart !
radarchart( data  , axistype=2 ,

            #custom polygon
            pcol=rgb(0.2,0.5,0.5,0.9) , pfcol=rgb(0.2,0.5,0.5,0.5) , plwd=4 ,

            #custom the grid
            cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,5,10),
cglwd=0.55,

            #custom labels
            vlcex=0.75
)

#removendo objetos
rm(A)
rm(data)
#---
#FIM
#---
```

```
#*********
#SCRIPT 12
#*********

#-------------------------------------------------
# PERCEPCAO E ADOCAO DE PRATICAS AGEIS
# plot baseado nos dados da Questao 7
# 14 - Você poderia nos dizer quais foram seus
#      principais desafios, enquanto
#      trabalha/trabalhava com práticas ágeis?
# Pesquisa de Mestrado UNB
# Autor Alan Saulo Costa Mazuco
# Orientador: Dr. Rodrigo Bonifácio
#-------------------------------------------------

# desafios (quantidades)
# Dados extraidos da questao 14 apos interpretacao
# D1 D2 D3 D4
#  3  2 14  3

# Passando os dados da pesquisa
D1  <- c(3)  #cumprimento de prazos
D2  <- c(2)  #gerência
D3  <- c(14) #paradigmaS
D4  <- c(3)  #falta de conhecimento

# Criando as linhas da matriz
height <- rbind(D1, D2, D3, D4)
h <- height #sort(height, decreasing = TRUE)
par(cex=0.85, family="serif", par(las=2)) #..............tamanho da fonte
par(mar=c(18,5,4,2)) # increase y-axis margin.

# Plotando - colocando o label ordenado
mp <- barplot(h,
              horiz=FALSE,
              beside = TRUE,
              xlim = c(0, 14.5),
              ylim = c(0, 20),
              names.arg=c(
                "D1",
                "D2",
                "D3",
                "D4"
              ), cex.names=0.85)

# Colocando labels em cada coluna
text(mp, height, labels = format(height, 4, digits = 4), pos = 3, cex = .75)

#Adicionando as legendas
legend(-1.3,-10,c(  "D1 : Cumprimento de prazos",
                    "D2 : Gerência",
                    "D3 : Quebra de paradigmas",
                    "D4 : Falta de conhecimento"), xpd = TRUE, cex = 1.15)

#removendo objetos
rm(height)
rm(mp)
rm(D1)
rm(D2)
rm(D3)
rm(D4)

#---
#FIM
#---
```