Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática

Colocar o Titulo do Tcc de Thais

Thais Moura de Freitas

Recife

Janeiro de 2015

Thais Moura de Freitas

COLOCA TITULO DO TCC

Orientador: Teresa Maciel

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife

Janeiro de 2015

Aos meus pais, Marcia e Abraão
Aos meus orientadores, Teresa e Giordano
Aos meus amigos

Agradecimentos

Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessãoEscrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão Escrever os agradecimentos aqui nesta sessão

Resumo

Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui...

Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui... Escrever o resumo de thais aqui...

E se quiser colocar aspas usar “assim fica dentro das aspas“

Palavras-chave: Métodos ágeis, adoção ágil, lições aprendidas

Abstract

Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles. Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles.Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles. Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles. Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles. Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles.

Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles. Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles.Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles. Escrever o resumo do tcc de thais sobre automação de testes aqui em ingles.

E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".E de novo se quiser usar "usar assim".

Keywords: Agile methods, agile adoption, lessons learned

Sumário

1 Introdução **1**

1.1 Apresentação 1

1.2 Justificativas 2

1.3 Objetivos 6

1.3.1 Objetivo Geral 6

1.3.2 Objetivo Específicos 6

1.4 Contribuições obtidas 6

1.5 Organização do trabalho 7

2 Referêncial Teórico **8**

2.1 Teste de Software 8

2.2 Tipos de Teste de software 9

2.2.1 Testes Manuais 9

2.2.2 Testes Automatizados 9

2.3 Níveis de testes de Software 9

2.3.1 Testes Funcionais 10

2.3.2 Testes Unitários 10
2.3.3 Testes de Integração 10
2.3.4 Teste de Sistema 11
2.3.5 Teste de Regressão 11
2.3.6 Testes de Aceitação 11
2.3.7 Testes Não-Funcionais 12

3 Automação de Testes **13**

3.1 Mapeamento e categorização 13
3.2 Análises das lições aprendidas 16
3.2.1 Experiência, treinamento e aprendizado 16
3.2.2 Planejamento e gerenciamento de backlog 18
3.2.3 Apoio gerencial e dos clientes 19
3.2.4 Customização e adaptabilidade 20
3.2.5 Confiança do time 22
3.2.6 Engajamento, comprometimento, disciplina e trabalho em equipe . . 23
3.2.7 Aspectos técnicos e tecnológicos 25
3.2.8 Compartilhamento de conhecimento 26
3.2.9 Velocidade de entrega e produtividade 28
3.2.10 Qualidade do produto final 29
3.2.11 Tamanho da organização 30
3.2.12 Quebra de paradigma 31

3.2.13 Comunicação remota 33

3.2.14 Cultura organizacional 34

4 Validação das Categorias de Lições Aprendidas 36

4.1 Método de validação 36

4.1.1 Definição do objetivo 36

4.1.2 Elaboração do questionário 37

4.1.3 Definição da população pesquisada 37

4.1.4 Aplicação do questionário 40

4.1.5 Análise dos resultados 41

5 Considerações Finais 47

5.1 Conclusão da pesquisa 47

5.2 Trabalhos futuros 48

Lista de Tabelas

3.1 Mapeamento de categorias de lições aprendidas e suas respectivas referências 14

3.2 Lições aprendidas agrupadas na categoria “Experiência, treinamento e aprendizado” 16

3.3 Lições aprendidas agrupadas na categoria “Planejamento e gerenciamento de backlog” 18

3.4 Lições aprendidas agrupadas na categoria “Apoio gerencial e dos cliente” . . 19

3.5 Lições aprendidas agrupadas na categoria “Customização e adaptabilidade” . 20

3.6 Lições aprendidas agrupadas na categoria “Confiança do time” 22

3.7 Lições aprendidas agrupadas na categoria “Engajamento, comprometimento, disciplina e trabalho em equipe” 23

3.8 Lições aprendidas agrupadas na categoria “Aspectos técnicos e tecnológicos” 25

3.9 Lições aprendidas agrupadas na categoria “Compartilhamento de conhecimento” 26

3.10 Lições aprendidas agrupadas na categoria “Velocidade de entrega e produtividade” 28

3.11 Lições aprendidas agrupadas na categoria “Qualidade do produto final . . . 29

3.12 Lições aprendidas agrupadas na categoria “Tamanho da organização 30

3.13 Lições aprendidas agrupadas na categoria “Quebra de paradigma” 31

3.14 Lições aprendidas agrupadas na categoria “Comunicação remota” 33

3.15 Lições aprendidas agrupadas na categoria “Cultura organizacional” 34

4.1 Resumo sobre alguns profissionais da área de TI que responderam o formulário e se identificaram 40

Lista de Figuras

1.1 Pirâmide de testes anti-padrão 4

1.2 Pirâmide ideal de testes 5

3.1 Gráfico que contabiliza a quantidade de trabalhos que referenciaram cada categoria de lições aprendidas 15

3.2 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Experiência, treinamento e aprendizado" 17

3.3 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Planejamento e gerenciamento de backlog" 19

3.4 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Apoio gerencial e dos clientes" 20

3.5 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Customização e adaptabilidade" 21

3.6 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Confiança do time" 23

3.7 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Engajamento, comprometimento, disciplina e trabalho em equipe" 24

3.8 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Aspectos técnicos e tecnológicos" 26

3.9 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Compartilhamento de conhecimento" 27

3.10 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Velocidade de entrega e produtividade" 29

3.11 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Qualidade do produto final" 30

3.12 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Tamanho da organização" 31

3.13 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Quebra de paradigma" 32

3.14 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Comunicação remota" 34

3.15 Percentual de trabalhos que referenciaram lições aprendidas da categoria "Cultura organizacional" 35

4.1 Contabilização das respostas referentes à criticidade da categoria de LAs "Experiência, treinamento e aprendizado" 41

4.2 Contabilização das respostas referentes à criticidade da categoria de LAs "Planejamento e gerenciamento de backlog" 41

4.3 Contabilização das respostas referentes à criticidade da categoria de LAs "Apoio gerencial e dos clientes" 41

4.4 Contabilização das respostas referentes à criticidade da categoria de LAs "Customização e adaptabilidade" 42

4.5 Contabilização das respostas referentes à criticidade da categoria de LAs "Confiança do time" 42

4.6 Contabilização das respostas referentes à criticidade da categoria de LAs "Engajamento, comprometimento, disciplina e trabalho em equipe" 42

4.7 Contabilização das respostas referentes à criticidade da categoria de LAs "Aspectos técnicos e tecnológicos" 43

4.8 Contabilização das respostas referentes à criticidade da categoria de LAs "Compartilhamento de conhecimento" 43

4.9 Contabilização das respostas referentes à criticidade da categoria de LAs "Velocidade de entrega e produtividade" 43

4.10 Contabilização das respostas referentes à criticidade da categoria de LAs "Qualidade do produto final" 44

4.11 Contabilização das respostas referentes à criticidade da categoria de LAs "Tamanho da organização" 44

4.12 Contabilização das respostas referentes à criticidade da categoria de LAs "Quebra de paradigma" 44

4.13 Contabilização das respostas referentes à criticidade da categoria de LAs "Comunicação remota" 45

4.14 Contabilização das respostas referentes à criticidade da categoria de LAs "Cultura organizacional" 45

4.15 Contabilização das respostas referentes às categorias de LAs consideradas mais importantes 46

Capítulo 1

Introdução

1.1 Apresentação

No desenvolvimento de software muitos métodos ágeis, como Lean, Scrum e XP é recomendado que todas as pessoas de um projeto (programadores, gerentes, equipes de homologação e até mesmo os clientes) trabalhem controlando a qualidade do produto, todos os dias e a todo momento, pois, há evidências que prevenir defeitos é mais fácil e barato do que identificar e corrigi. Vale ressaltar ainda, que os métodos ágeis não se opõem a quaisquer revisões adicionais que sejam feitas para aumentar a qualidade [38].

Uma importante etapa do ciclo de vida do desenvolvimento de software que compõe as metodologias ágeis é a fase que envolve os testes, pois a qualidade dos testes aplicados impacta diretamente no funcionamento estável das aplicações. Essa fase pode ser demorada e desgastante quando são executadas de forma repetitiva, principalmente se falhas forem encontradas através dos testes exploratórios ou ah-doc, o que levará o testador possivelmente a despender tempo na investigação e entendimento das reais causas do problema [53].

Uma das principais razões para automatizar testes é a diminuição do tempo gasto nos testes manuais [34]. Além de aumentar a eficiência de etapas repetitivas para reprodução de funcionalidades do sistema, especialmente em testes de regressão, onde os testes são executados iterativo e incremental após mudanças feitas no software [11].

1.2 Justificativas

A necessidade da entrega cada vez mais rápida de produtos de software faz com que o processo de teste necessite constantemente de rapidez e agilidade. Assim sendo, a automação de testes no ciclo de vida do desenvolvimento de software tem sido constantemente utilizada para suprir essa carência. É importante ressaltar que existe diferença entre testes e automação de testes, o primeiro termo se refere ao ato de testar, já automação é utilizar um software para imitar a interação do ser humano com a aplicação a ser testada [35].

Um fator que deve ser levado em consideração é que o tempo de execução dos testes automatizados, pois ocorre em proporções bem menores em relação ao tempo executado por um processo de teste manual, onde nem sempre o time terá tempo hábil para aplicar todos os testes planejados. Isso não exclui a possibilidade do processo de testes ser hibrido, isto é, contemplar testes manuais e automatizados. Automação de testes pode ser uma poderosa forma de testes não funcionais, por exemplo, volume, carga e regressão [39].

A automação dos testes tem como intuito a maximização da cobertura dos testes dentro do tempo disponível, para a validação e construção do software, aumentando a confiabilidade e qualidade [1]. Considerando que o esforço em atividades de testes em projetos podem ser responsável por até 50% do esforço total de desenvolvimento, automatizar o processo de testes é importante na redução e melhoria da eficácia dos testes realizados [33]. Os benefícios de tal abordagem em comparação com o teste manuais seriam: baixo custo de execução dos testes, possibilidade de replicar as sequências de teste velhas em novas versões de software (não gastando tempo com testes de regressão) e a possibilidade de realizar testes de estresse por um longo tempo de duração [1].

Ainda querendo responder a pergunta, Porque automação de testes? Segundo [12] esses seriam os principais motivos:

- Realização dos testes de regressão mais rápido para que os sistemas/aplicações possam continuar mudando ao longo do tempo, sem uma longa fase de testes no final de cada ciclo de desenvolvimento.
- Encontrar defeitos e problemas rapidamente, especialmente quando existe testes que

podem ser executadas em máquinas de desenvolvedores, e como parte do serviço de complicação em um servidor de CI.

- Certificação de que pontos de integração externos estão trabalhando da forma esperada.
- Assegurar que o usuário pode interagir com o sistema como desejado.
- Auxilia no debbuging, escrita e desenho do código fonte.
- Ajuda a especificar o desempenho da aplicação.

No geral, com o uso de testes automatizados estamos aumentado a velocidade de entrega do projeto construído e com maior garantia de qualidade [12]. O que ameniza o esforço gasto pelas empresas no processo com testes manuais e aumenta a cobertura de testes que não seriam realizados, por falta de tempo e esforço.

Muitas organizações costumam cair na mesma armadilha da aplicação da pirâmide de testes invertidas ou anti-padrão, como pode ser visto na imagem [47].

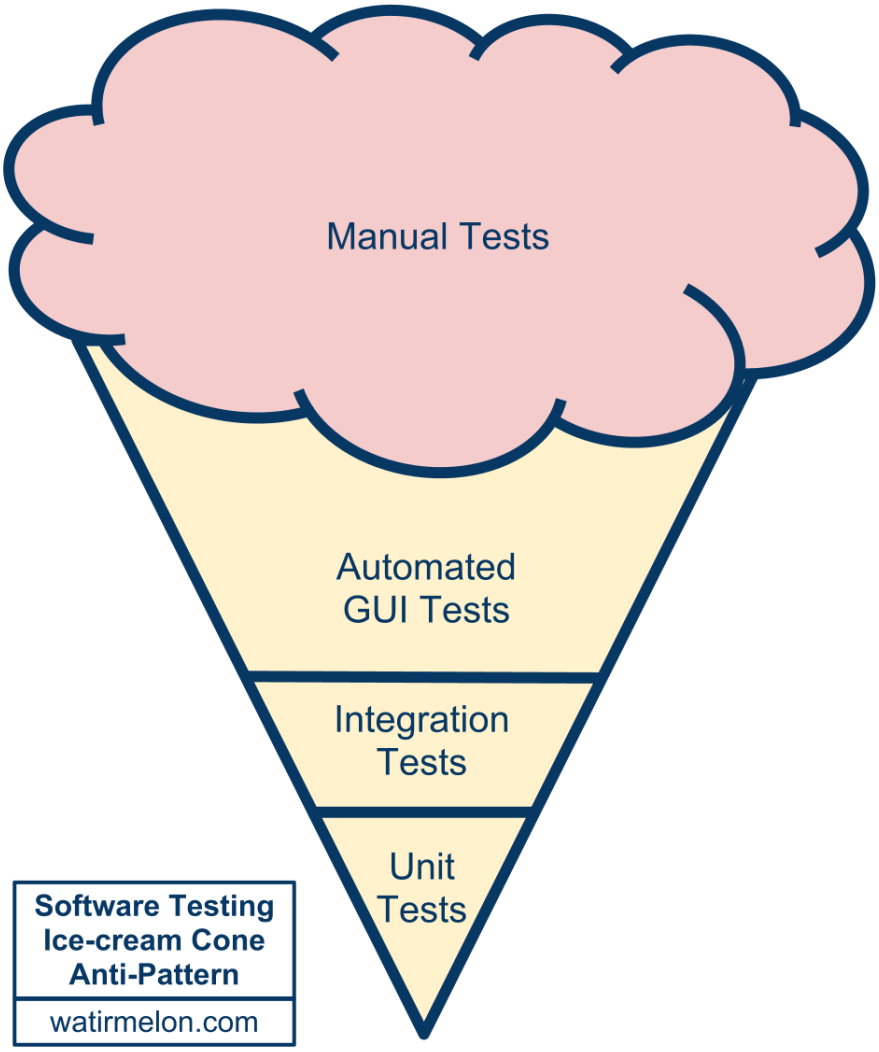


Figura 1.1: Pirâmide de testes anti-padrão

Nesta figura está representado visualmente a quantidade de diferentes tipos de testes que podem ser aplicados pelo time no decorrer do desenvolvimento do software, onde é visto que boa parte dos testes são realizados manualmente e um quantidade pequena é realizada a nível unitário.

Uma parte importante da estratégia de testes, é saber o foco de cada tipo diferente de testes e fazer com que os diferentes tipos de testes trabalhem juntos [12]. Por exemplo, realizar testes de unidade, com alguns testes de integração e um pequeno número de testes de aceitação. Com essa mistura é possível cobrir caminhos alternativos de testes no código, alcançar e

ultrapassar barreiras de testes mais rápido aplicando os testes de unidade [12]. A seguir é mostrada a pirâmide ideal para aplicação de teste automatizado, em que o foco não está apenas nos testes manuais e sim na base da pirâmide [13].

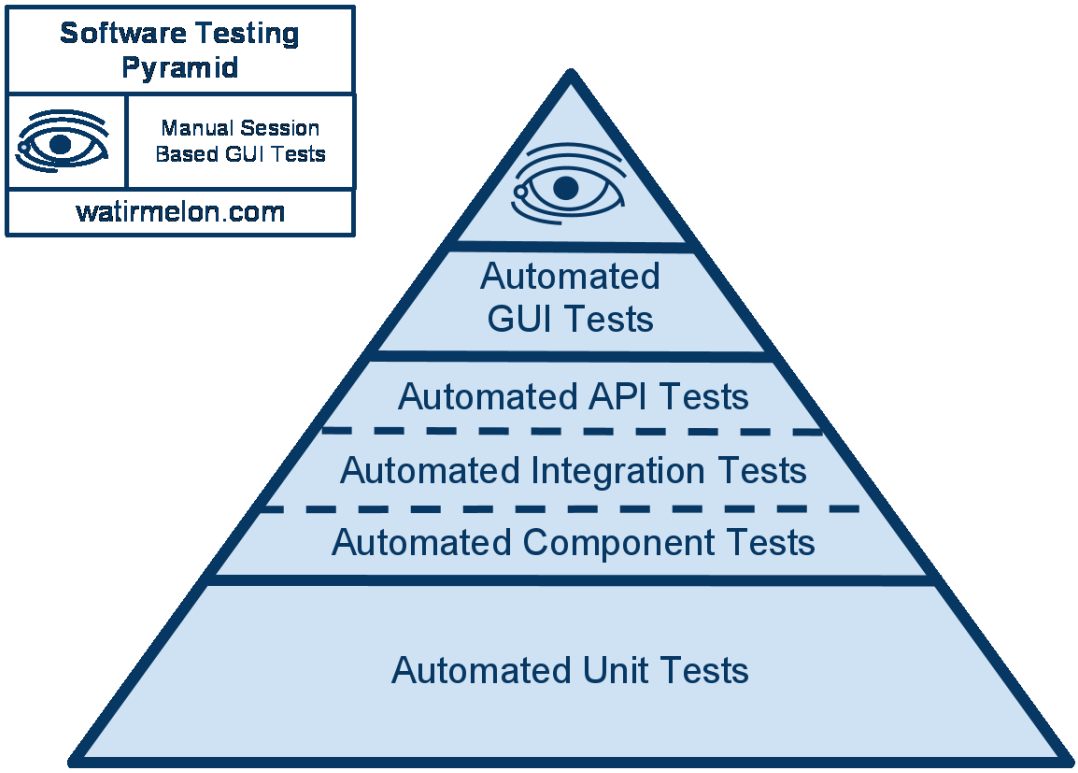


Figura 1.2: Pirâmide ideal de testes

Sendo assim, é de extrema importancia a aplição de tipos de estratégia de testes como esta, para o desenvolvimento de software, garantindo que o que está sendo implementado seja conforme o esperado e de forma mais rápida e automática, envolvendo todo o time na qualidade final e em todas as fases do processo.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é a implantação e cobertura de testes automatizados durante o desenvolvimento de uma aplicação que é desenvolvida com tecnologia de classificação de sinais de áudio para dar suporte a pessoas com deficiência auditiva ou surda.

1.3.2 Objetivo Específicos

Dentre os objetivos específicos deste trabalho, temos:

- Gerar conhecimento em torno de automação de testes, pesquisando e selecionando as melhores técnicas para esta proposta;
- Gerar conhecimento nécessário para aplicar automação dos testes da ferramenta jMIR utilizada para construção da aplicação;
- Investigar e selecionar as melhores técnicas e ferramentas aplicáveis na automação dos teste na aplicação escolhida;
- Aplicar automação de testes em todo ciclo de vida no desenvolvimento de uma aplicação;
- Aplicar automação de testes para validar a integração da aplicação com a tecnologia JMIR, mais especificamente o uso do jAudio e ACE que são os componentes principais;
- Realizar o levantamento de quantos testes foram automatizados em cada nível da pirâmide ideal de testes durante o desenvolvimento da aplicação;

1.4 Contribuições obtidas

Com esse trabalho....

1.5 Organização do trabalho

Escrever a organização do trabalho...

Capítulo 2

Referêncial Teórico

É apresentado nesse capítulo um conjunto de definições relevantes para o tema dessa proposta de trabalho. Foi realizada uma pesquisa sistemática para selecionar os principais tópicos apoiam o entendimento deste trabalho. Os detalhes estão descritos nas próximas seções.

2.1 Teste de Software

Teste de software é o processo que realiza a avaliação do sistema ou de seus componentes com o objetivo de desvendar o comportamento para garantir que os seus requisitos estão de acordo com o esperado ou não. Em outras palavras teste é execução do sistema, a fim de identificar eventuais lacunas, erros ou requisitos que não foram implementados de acordo com as necessidades requeridas [32]. Segundo [7], Teste pode ser definido como o processo de análise do software para detectar as diferenças entre condições existentes e necessárias e avaliar as características do software.

Um bom teste pode ser aquele que tem alta probabilidade de encontrar erros que ainda não foram expostos e um teste bem sucedido é aquele que revela um erro ainda não-descoberto. O teste pode ser manual, automatizado, ou ainda a combinação de ambos [44]. A redução de custo, tempo e retrabalho são proporcionais ao quão cedo o processo de testes for iniciado [32].

2.2 Tipos de Teste de software

Testes de software podem ser divididos em duas categorias, testes manuais e automatizados.

2.2.1 Testes Manuais

Este tipo de teste é feito pela execução manual do software, ou seja, sem o uso de qualquer ferramenta automatizada ou qualquer script. O testador assume o papel do usuário final para realizar os testes e identificar qualquer comportamento que não seja esperado ou revelar defeitos. Costumam usar planos de teste, casos de teste e/ou cenários para garantir a integridade dos testes. Também pode ser incluso dentro deste tipo os testes exploratórios, no qual o testador irá explorar o software com o intuito de encontrar erros[32].

2.2.2 Testes Automatizados

Testes automatizados podem ser definidos como automação de atividades de teste de software, incluindo o desenvolvimento e execução de scripts de testes, verificação de requisitos, como também a utilização de ferramentas de teste automatizadas [43].

Entre as razões para o uso da automação dos testes de software podemos citar, por exemplo, executar testes manuais é mais demorado, uso da automação de testes aumenta a eficiência processo, em um fatia particular temos os testes de regressão, onde os casos de testes são executados de forma iterativa, depois de alterações no software [43].

2.3 Níveis de testes de Software

Em [44] aponta que a realização de testes apenas quando o sistema está construído é uma abordagem ineficaz. Segundo o autor, a estratégia de testes de software de possuir uma abordagem incremental, começando com os testes de unidades, seguindo com os testes de integração, culminando com os testes do sistema final e ainda acoplando testes que se enquadrem em testes de aceitação.

2.3.1 Testes Funcionais

Este tipo de teste é baseado nas especificações do software que será testado, a aplicação será testada através do fornecimento de entradas e em seguida, os resultados serão examinados para garantir a conformidade com os requisitos de tal funcionalidade [32]. Basicamente testar se os componentes e o sistema estão feitos, uma atividade ou uma função especifica do código está coerente com o esperado. Neste nível de teste perguntas como "O usuário poderá fazer isso?"ou "Esta função em particular funciona?"podem ser validadas tipicamente através de especificações de requisitos ou funcional [52].

2.3.2 Testes Unitários

É a menor parte que pode ser testada do código que compõe o software, são de granulação fina, comportamento extremamente rápido, por exemplo, métodos de uma classe, uma classe ou classes que podem estar relacionadas. Não verificam o comportamento de unidades integradas com outros serviços ou dependências, garantem que sua função como unidade estão funcionando. Seu objetivo é isolar parte do programa e mostrar que partes individuais estão corretas em termos de requisitos e funcionalidades [12]. Existe um limite para cenários e dados que podem ser aplicados ao nível de testes de unidade, quando é nítido esse esgotamento é necessário o mistura com outras unidades do código e diferentes tipos de testes [32].

2.3.3 Testes de Integração

Neste caso, é realizada a combinação de partes da aplicação que serão agrupadas para determinar se funcionamento desse conjunto está trabalhando corretamente [32]. Alguns defeitos não podem ser encontrados a nível unitário, mas são revelados através da integração com núcleos específicos [50].

2.3.4 Teste de Sistema

Após a realização dos testes de integração, uma vezes que todos os componentes foram integrados, os testes do sistema como um todo são agora efetuados para garantir que a aplicação atende aos padrões de qualidade desejado [32]. A aplicação é testada minuciosamente para validar especificações funcionais e técnicas, é construído um ambiente o mais próximo possível do ambiente de produção, para que os testes sejam executados em um ambiente que corresponda ao ambiente no qual o sistema será implantado. O teste do sistema nos retornará a validação dos requisitos de negócio e da arquitetura como todo da aplicação.

2.3.5 Teste de Regressão

Significa testar uma aplicação após o seu código fonte ter sido modificado para validar se ainda continua funcionando devidamente. Consiste em reexecutar casos de testes existentes e verificar se alterações de código não interfere em funcionalidades que estavam trabalhando corretamente, se foram inseridos novos erros ou causa problemas em erros que já foram reparados [52]. Afim de identificar defeitos introduzidos por novas funcionalidades ou correção de defeitos.

2.3.6 Testes de Aceitação

Nível mais alto que trata da aplicação como uma caixa preta, sem dúvidas considerado um dos mais importantes entre os outros, este tipo de teste é feito na grande parte pelo time de garantia da qualidade [12]. Onde todo time verificará se o produto preenche as especificações solicitadas e satisfaz os requisitos dos clientes. O time responsável terá um conjunto de cenários previamente escritos e que serão usados no momento da validação. Durante a fase de validação outros cenários podem surgir para aumentar a cobertura dos testes. Testes de aceitação não são apenas para identificação de erros de interfaces, mas também para encontrar divergências que poderão resultar em quebra ou erros graves do sistema [32]. Para a execução de desse tipo de testes é necessário definir critérios de aceitação a partir dos requisitos do software, estabelecendo como o teste será conduzido e a partir desses critérios,

avaliar se o produto satisfaz aos requisitos [49].

2.3.7 Testes Não-Funcionais

Requisitos não funcionais são declarações que define as qualidades globais ou atributos a serem atendidos pelo sistema resultante [27]. Estes requisitos, ao contrário dos requisitos funcionais, não expressão nenhuma função a ser realizada pelo software, e sim comportamentos e restrições que este software deve satisfazer [14]. Os testes desses requisitos são normalmente executados com ajuda de ferramentas especializadas, com grande planejamento, avaliação arquitetural, aplicando técnicas avançadas [32].

Alguns dos considerados mais importantes tipos de validação não funcionais são [32]:

- Teste de Performance
- Teste de Carga
- Teste de Stress
- Teste de Segurança
- Teste de Usabilidade
- Teste de Portabilidade

Capítulo 3

Automação de Testes

Ainda que a atividade seja complexa, o teste de software nem sempre é realizado de forma sistemática devido a diversões fatores dos projetos, como tempo e recursos limitados, qualificação do time e dos envolvidos e consequência da complexidade e da rapidez da evolução dos sistemas, sendo a automação de testes uma importante medida para melhorar a eficiência dessa atividade [9].

3.1 Mapeamento e categorização

As diversas lições aprendidas reportadas pela base da literatura científica e relatos de experiência foram separadas em categorias, por similaridade, para obter uma maior visibilidade do que foi encontrado. Alguns exemplos dessas categorias são: “Apoio gerencial e dos clientes”e “Aspectos técnicos e tecnológicos”. Considerou-se a existência real de uma categoria quando pelo menos 6 trabalhos (20% do material analisado) referenciavam alguma lição aprendida que se encaixasse naquela categoria.

A tabela 3.1 apresentada abaixo relaciona as categorias criadas e, a cada categoria, estão relacionadas as referências dos artigos científicos (coluna 2) e relatos de experiência (coluna 3) que possuem lições aprendidas naquele contexto. Em seguida, a figura 3.1 é um gráfico que contabiliza a quantidade de trabalhos que referenciaram cada categoria de lições aprendidas.

Categorias	**Artigos científicos**	**Relatos de experiência**
Experiência, treinamento e aprendizado	[23], [6], [22], [40], [31], [26], [37], [3], [21]	[46], [20], [5], [15], [17], [16], [51]
Planejamento e gerenciamento de backlog	[23], [21], [6], [22], [8], [28], [37]	[16], [25], [18]
Apoio gerencial e dos clientes	[23], [40], [37], [2]	[18], [46], [5], [15], [10], [16]
Customização e adaptabilidade	[23], [6], [3], [21], [8], [36], [31], [37], [29]	[16], [25], [20], [5], [15], [48], [45], [51]
Confiança do time	[6], [3], [37], [29]	[18], [48], [16], [5]
Engajamento, comprometimento, disciplina e trabalho em equipe	[6], [3], [31], [36], [37], [29], [22], [21]	[16], [18], [46], [20], [15], [41], [5], [48]
Aspectos técnicos e tecnológicos	[6], [36], [28], [40], [31], [26], [21], [2], [8], [42], [29]	[16], [41], [46], [17]
Compartilhamento de conhecimento	[3], [40], [31], [42], [26], [30]	[19], [51], [41], [5], [15]
Velocidade de entrega e produtividade	[22], [21], [36], [40], [28], [26], [37]	[46], [41], [15], [25], [48], [16]
Qualidade do produto final	[22], [21], [8], [31], [26], [37], [28]	[18], [15], [48]
Tamanho da organização	[8], [36], [37], [28], [30]	[15]
Quebra de paradigma	[23], [6], [28], [31], [2]	[46], [5], [15], [18], [25], [48], [45]
Comunicação remota	[22], [36], [28], [42]	[20], [51], [5], [15]
Cultura organizacional	[8], [36], [37], [29], [26], [21]	[20], [5], [10], [15], [45]

Tabela 3.1: Mapeamento de categorias de lições aprendidas e suas respectivas referências

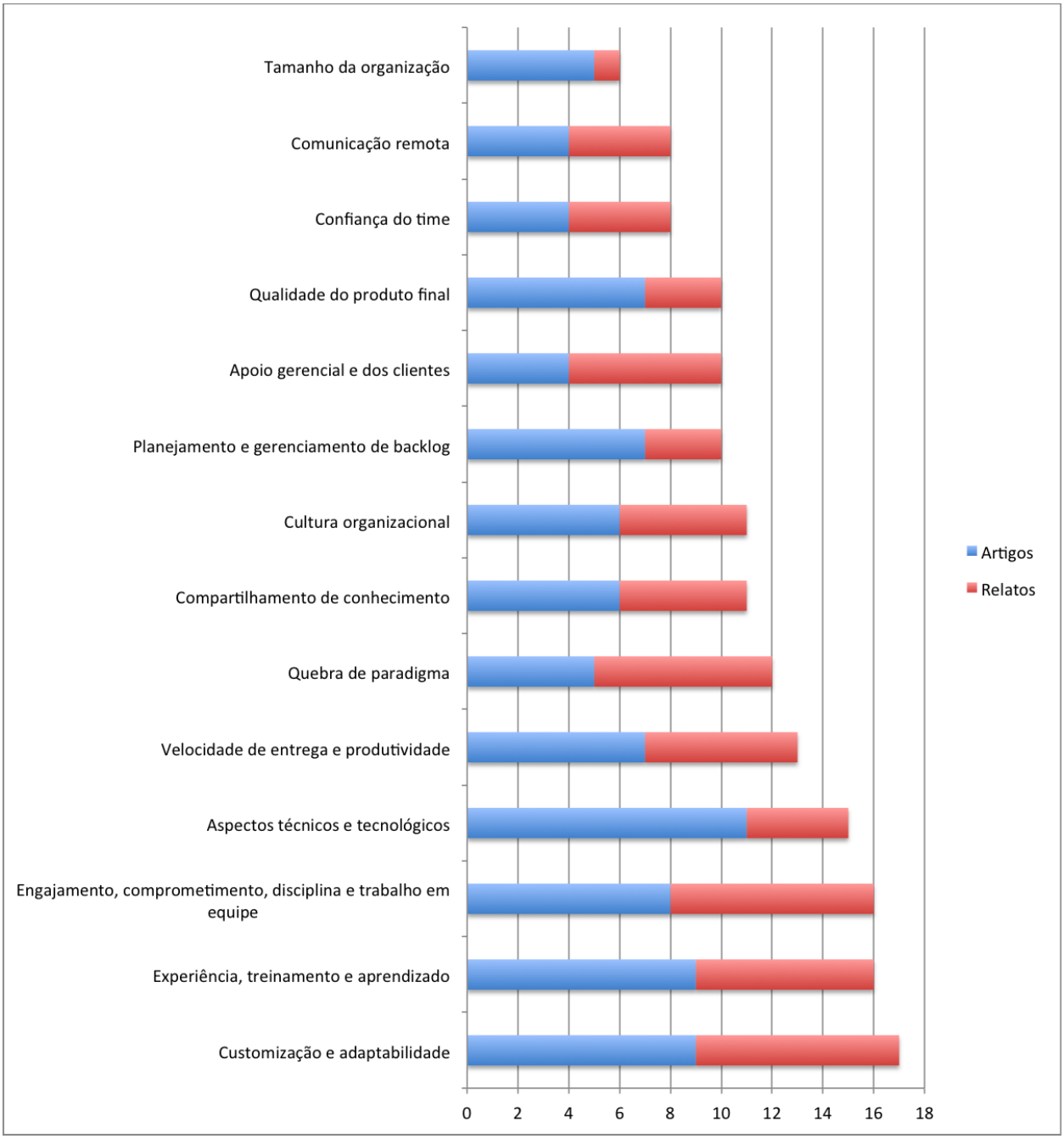


Figura 3.1: Gráfico que contabiliza a quantidade de trabalhos que referenciaram cada categoria de lições aprendidas

3.2 Análises das lições aprendidas

Esta seção analisa as categorias com as respectivas lições aprendidas coletadas. Os percentuais de trabalhos que referenciaram cada categoria de lições aprendidas estão exibidos ao final de cada subseção correspondente. As áreas hachuradas em verde correspondem aos materiais que apontaram alguma lição aprendida daquela categoria.

3.2.1 Experiência, treinamento e aprendizado

Lições aprendidas	**Referências**
É muito difícil aventurar-se em Ágil sem um Agile Coach	[23], [6], [22], [40], [31], [26], [37], [46], [20], [5], [15], [17]
Vivenciar um projeto-piloto é uma prática muito eficiente para se adquirir experiência	[23], [40], [20], [15]
É preciso prática, não apenas estudo	[23], [3], [37], [16], [51]
Trabalhar com equipes não-ágeis pode atrapalhar o andamento do projeto	[22], [15]

Tabela 3.2: Lições aprendidas agrupadas na categoria "Experiência, treinamento e aprendizado"

Por muitos anos empresas de desenvolvimento de software adotaram o modelo cascata como forma de gerenciamento de projetos. O Manifesto Ágil [24], ocorrido em 2001, modificou completamente a maneira de lidar com esse tipo de atividade.

Mudar bruscamente nunca é simples. Para que essa transição não seja tão dolorosa, uma tática muito popular entre as empresas é a contratação de um profissional experiente para servir como guia e treinar toda a equipe. Esse papel é essencial durante o processo de adoção ágil em qualquer organização [23]. Outra maneira de se adquirir experiência de forma rápida e eficaz é através de um projeto-piloto. Power utilizou essa abordagem com o intuito de entender melhor como a empresa funcionava e tentar atacar os pontos mais críticos [40].

Alguns pontos negativos também foram levantados nos trabalhos analisados. É preciso aprender com os próprios erros, não apenas realizar treinamentos [16]. E, de acordo com Green e Maciel, trabalhar com stakeholders que não são ágeis pode afetar negativamente o desempenho do time ágil, que trabalha em um ambiente menos burocrático e com um ciclo de feedback bem mais curto [22, 15].

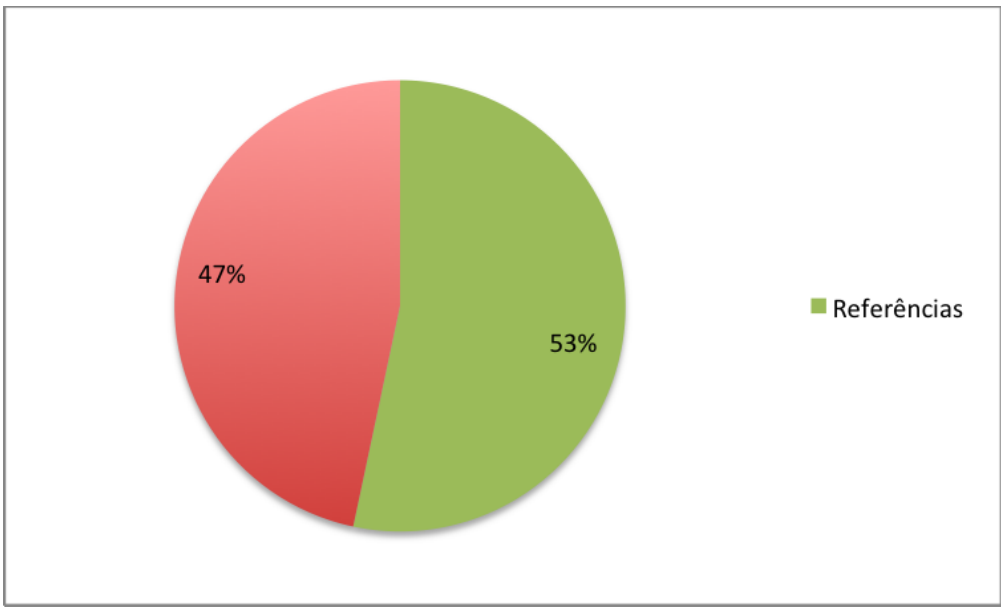


Figura 3.2: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Experiência, treinamento e aprendizado"

3.2.2 Planejamento e gerenciamento de backlog

Lições aprendidas	**Referências**
Planejar apenas quando necessário	[23], [21], [16], [25], [18]
O processo de adoção em si precisa ser bem planejado	[23]
Priorizar o backlog é uma tarefa complicada para times inexperientes	[6]
É difícil lidar com o aumento de escopo	[6]
É preciso aprender a quebrar o backlog da forma correta	[22], [25], [18]
A coleta e gerência de requisitos ocorre de forma bem diferente do habitual	[8], [28], [37], [16], [25]

Tabela 3.3: Lições aprendidas agrupadas na categoria "Planejamento e gerenciamento de backlog"

Desenvolver de forma iterativa e incremental gera um grande impacto na forma de gerenciamento de projetos. Não é necessário planejar todas as etapas do processo, é difícil prever situações e cenários muito distantes. Planejar apenas o necessário, quando necessário, é um grande desafio para muitas empresas. Fitzgerald et al. relataram que tiveram que lidar com muitos problemas relacionados à granularidade no planejamento que Ágil propõe [21].

Todavia, não podemos confundir o processo de planejamento utilizado em projetos ágeis e o planejamento para se adotar Ágil. Dado que esta é uma grande mudança de paradigma, então é preciso ter cautela. Hajjdiab e Taleb aconselharam que esse processo deve ser bem planejado e que aconteça de forma gradual [23].

Outro ponto considerado desafiador por muitas organizações é o gerenciamento de backlog. Block relatou as dificuldades ao tentar priorizá-lo e impedir que ele crescesse além do comportado pela equipe [6]. Ainda com relação ao backlog, Green lembrou que um dos princípios primários do desenvolvimento ágil de software é o foco na entrega de pequenos incrementos de valor [22]. Isto pode até parecer simples à primeira impressão, porém dividir todo um conjunto de requisitos de um projeto em pequenas fatias independentes entre si e que agregam valor ao cliente não é algo trivial.

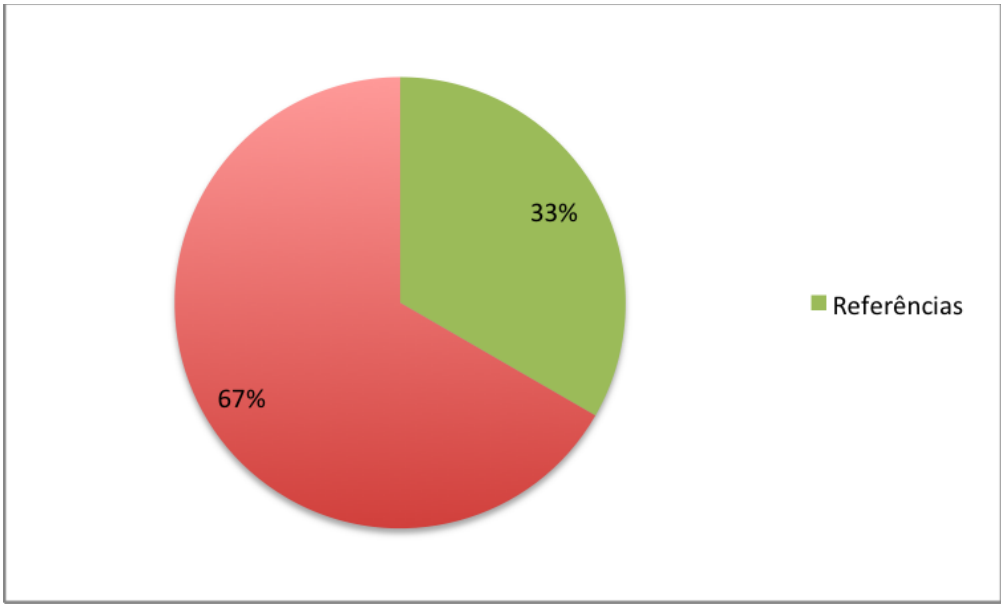


Figura 3.3: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Planejamento e gerenciamento de backlog"

3.2.3 Apoio gerencial e dos clientes

Lições aprendidas	**Referências**
Pressão por parte do alto escalão da empresa afeta negativamente no andamento do processo de adoção ágil	[23], [40], [37], [18], [46], [5], [15], [10]
O apoio do cliente é de suma importância para o sucesso do projeto	[2], [37], [18], [46], [15]
É muito difícil contornar uma situação quando o cliente pressiona o tima para que se adote práticas waterfall	[37], [16], [10]
É preciso ter liberdade ao se adotar Ágil	[16], [46], [15]

Tabela 3.4: Lições aprendidas agrupadas na categoria "Apoio gerencial e dos cliente"

Houve uma unanimidade quanto a este ponto. É preferível que todos os envolvidos em projetos de desenvolvimento de software que utilizam alguma metodologia ágil estejam alinhados com o processo. O apoio da gerência e dos clientes é fundamental para o bom desempenho

da equipe.

Contudo, em muitos casos, apenas o apoio não é o suficiente. É preciso ter liberdade para se adotar Ágil de forma bem sucedida. Muitos relatos evidenciaram problemas com esse nível de alinhamento com os princípios ágeis de liberdade e auto-organização [16, 46, 15].

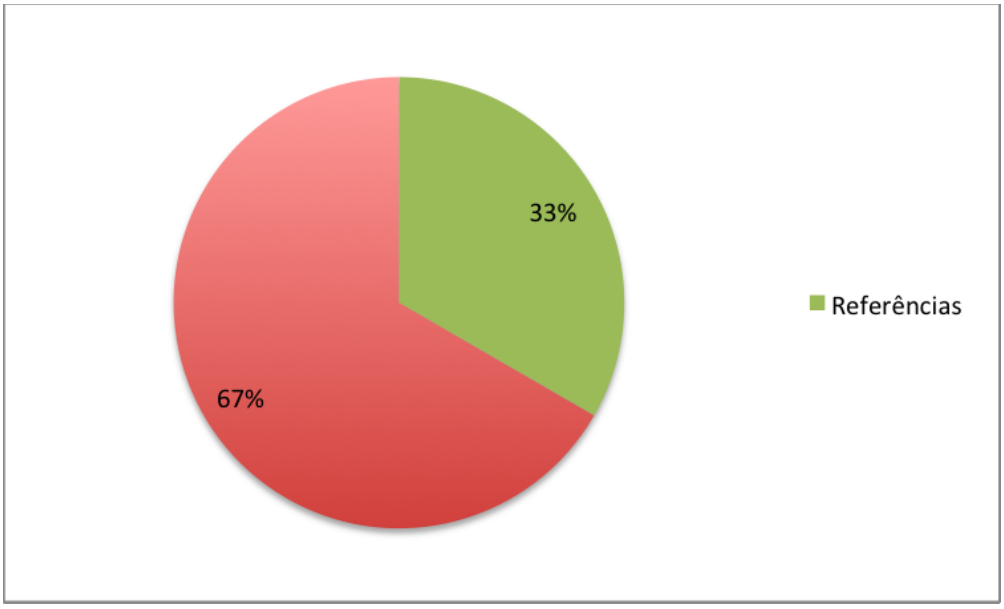


Figura 3.4: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Apoio gerencial e dos clientes"

3.2.4 Customização e adaptabilidade

Lições aprendidas	**Referências**
Ágil é customizável, não existem regras no seu processo de adoção	[23], [16], [25]
Dar suporte para adaptabilidade é um problema não-trivial	[6], [3], [21], [8], [36], [31], [37], [29], [20], [5], [15], [25], [48], [45]
Dificuldades na implantação de mudanças necessárias	[51], [5], [15], [25], [45]

Tabela 3.5: Lições aprendidas agrupadas na categoria "Customização e adaptabilidade"

Um dos principais pilares do desenvolvimento ágil é encarar mudanças como bem-vindas. Todavia, atingir um nível de maturidade de tal forma que isto ocorra naturalmente é algo para se orgulhar. Muitos trabalhos relataram problemas para dar suporte a essa adaptabilidade. Em muitas situações, times encontraram dificuldades para implantar as mudanças necessárias para obter um melhor resultado final em seus projetos. Um exemplo claro desse fator é o demonstrado por Fitzgerald et al. Segundo seus autores, ambientes regulados e métodos ágeis são frequentemente vistos como fundamentalmente incompatíveis, o que causa uma série de problemas de adaptabilidade [21].

Outro ponto relevante abordado por alguns trabalhos é o quanto Ágil pode ser customizável. Não exite um conjunto pré-definido de regras que devem ser seguidas à risca por aqueles que querem ser ágeis. Ágil é flexível. Segundo Hajjdiab e Taleb, um dos principais benefícios do desenvolvimento ágil é a sua capacidade de ser personalizado baseado na cultura e no ambiente da organização que o está adotando [23].

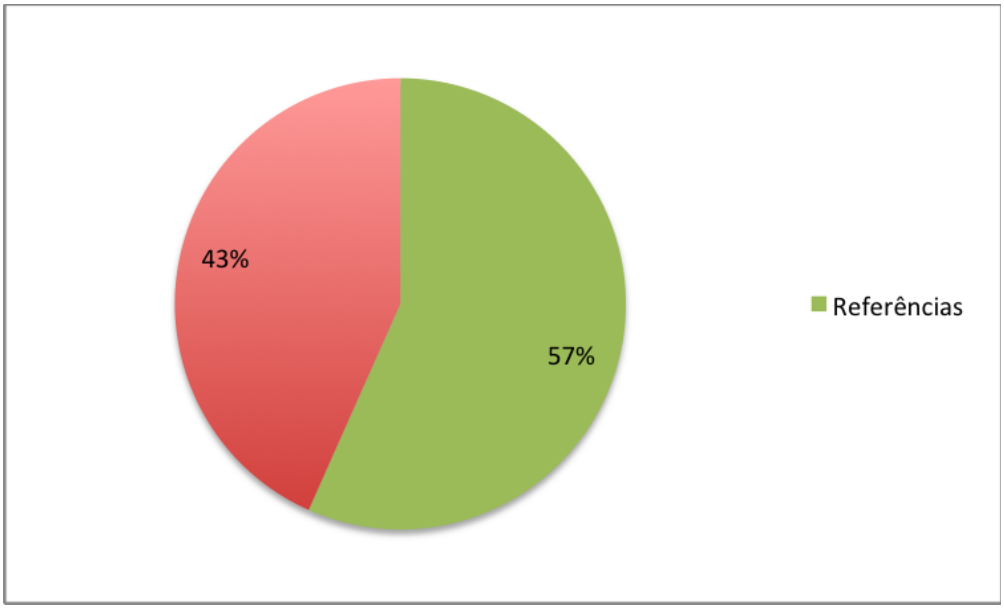


Figura 3.5: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Customização e adaptabilidade"

3.2.5 Confiança do time

Lições aprendidas	**Referências**
Entregar mais e entregar valor ajuda a manter o time confiante	[6], [3], [18]
Utilizar métodos ágeis ajuda a manter elevado o ânimo do time	[3], [37], [29], [48]
A manutenção da confiança das partes envolvidas se mostrou comprometida com o uso de metodologias ágeis	[29], [16], [5]

Tabela 3.6: Lições aprendidas agrupadas na categoria "Confiança do time"

Antes de analisar esse ponto, é preciso ter em mente que Ágil não é (nem nunca quis ser) a solução dos problemas causados por processos orientados a planejamento. Não existe solução mágica quando o assunto é desenvolvimento de software [16, 36].

Houve uma grande divergência com relação ao impacto causado por Ágil na confiança dos envolvidos. Enquanto vários trabalhos apontaram que o uso de métodos ágeis contribui com a manutenção do moral elevado do time, outros apontaram exatamente o oposto.

De acordo com Asnawi et al., o fato de Ágil proporcionar uma maior frequência de entregas de pedaços de software que agregam valor ao cliente torna o desenvolvimento do projeto mais prazeroso, menos estressante [3]. Em contrapartida, Korhonen relatou que, em certos casos, é desmotivador trabalhar em um cenário onde não há uma visão clara do que está por vir a longo prazo [29].

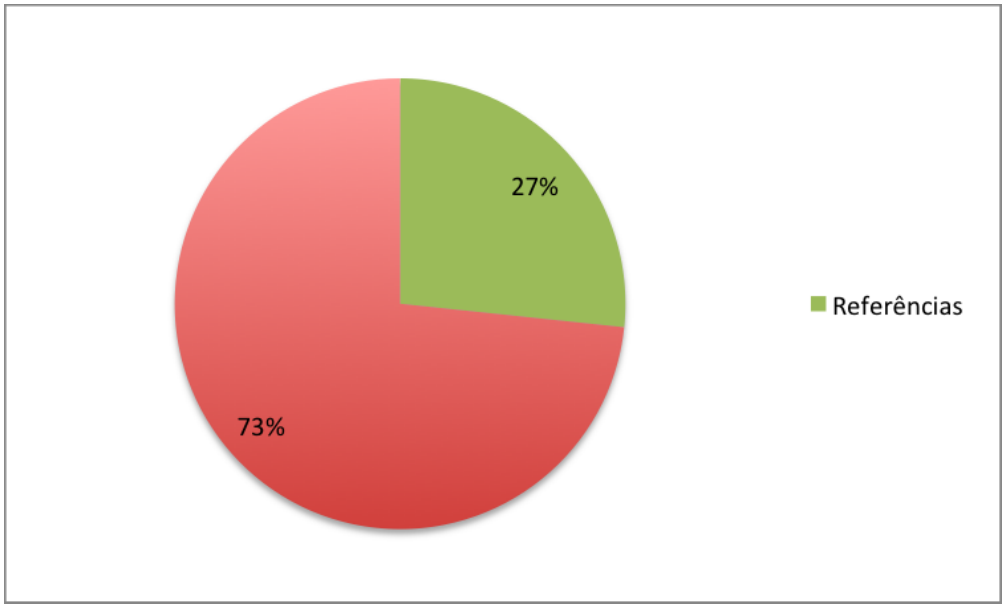


Figura 3.6: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Confiança do time"

3.2.6 Engajamento, comprometimento, disciplina e trabalho em equipe

Lições aprendidas	**Referências**
É difícil encontrar pessoas que trabalham bem em equipe	[6]
É muito importante ter o PO próximo ou, se possível, como membro ativo do time, envolvido em todas as etapas do processo	[6], [3], [31], [36], [37], [16], [18], [46], [20], [15]
Falta de ownership no projeto	[6], [29], [41]
O envolvimento dos desenvolvedores dentro do processo é extremamente importante	[3], [22], [21], [31], [36], [37], [46], [5], [15], [48]
Para projetos ágeis serem bem sucedidos, é preciso disciplina	[18]

Tabela 3.7: Lições aprendidas agrupadas na categoria "Engajamento, comprometimento, disciplina e trabalho em equipe"

Para que a construção e entrega de um software sejam bem sucedidas, é preciso esforço de diversos profissionais: desenvolvedores, analistas de qualidade, analistas de negócio, product owners, etc. Ficou bem claro que, para as empresas de desenvolvimento de software analisadas, todos os stakeholders precisam estar fortemente envolvidos em todas as etapas do processo de construção do produto. Contudo, um desafio visualizado através dos artigos e relatos foi o de manter o PO engajado. Um exemplo citado por Block é a dificuldade em manter o cliente próximo para que ele defina novas funcionalidades e priorize o backlog [6]. Situações como esta podem prejudicar consideravelmente o resultado final do projeto, causando frustração para ambos os lados.

Outro ponto pouco citado, porém relevante, é a questão da disciplina em projetos ágeis. Dado que, na teoria, deveríamos ter um ambiente menos controlado e com times auto-organizáveis. Disciplina é uma característica que precisa surgir naturalmente, visto que não existe um plano detalhado a ser seguido. Dalcin e Parzinello afirmaram que não se deve culpar o SCRUM (umas das metodologias ágeis mais populares) por sua falta de disciplina [18].

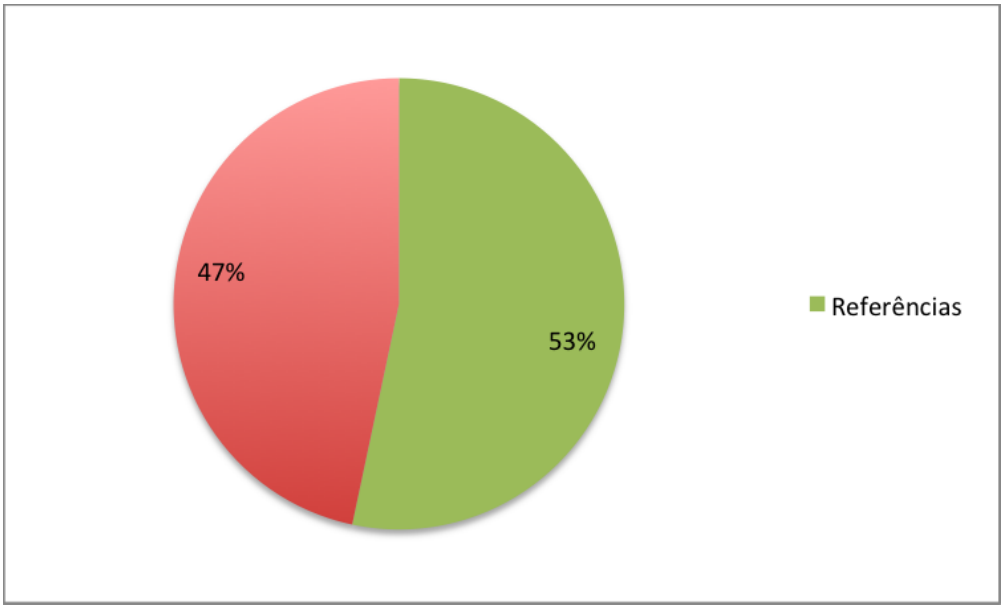


Figura 3.7: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Engajamento, comprometimento, disciplina e trabalho em equipe"

3.2.7 Aspectos técnicos e tecnológicos

Lições aprendidas	**Referências**
Utilizar Ágil em um projeto com código legado se mostrou desafiador	[6]
Integração contínua, automação de build e testes automatizados promovem ganhos muito vantajosos	[6], [36], [28], [40], [31], [26]
É necessário o suporte de boas ferramentas	[21], [36], [2]
A atividade de design arquitetural se mostrou desafiadora	[8], [42], [16]
Ágil promove uma melhor manutenibilidade do código	[8]
Infra-estruturas virtualizadas proporcionam a flexibilidade necessária para muitos projetos ágeis	[42]
Ágil promove uma pior manutenibilidade do código	[29], [41]
Em certos casos há resistência por parte do cliente para utilizar o produto construído	[46]

Tabela 3.8: Lições aprendidas agrupadas na categoria "Aspectos técnicos e tecnológicos"

Aspectos técnicos são muito peculiares, variam de projeto para projeto. Por conta disso, várias lições aprendidas referentes a esses aspectos foram coletadas. Todavia, um padrão observado é que muitas fontes fizeram alusão a uma prática relativamente comum em projetos ágeis: a implantação de um servidor de integração contínua. Pode-se usufruir de muitos benefícios a partir dessa prática, como, por exemplo, a automação de testes e deploy automático de código. Para Block, essas mudanças técnicas tiveram um papel fundamental no nível de sucesso que puderam conquistar com práticas ágeis [6].

Não planejar todos os detalhes de um projeto antes do seu início causa uma série de impactos no seu fluxo de atividades. Uma área bastante afetada por essa característica de Ágil é o design arquitetural. Diversos relatos afirmaram ter enfrentado muitos problemas com isso. Segundo Bustard et al., algumas empresas consideraram Ágil um retrocesso nesse aspecto [8].

A manutenibilidade de código foi o ponto polêmico dessa categoria. Korhonen relatou passar

por dificuldades em gerenciar muitas pessoas modificando o mesmo código simultaneamente [29], enquanto Bustard et al. mostraram que, em diversas empresas, a melhoria na qualidade e manutenibilidade do código foi um dos principais ganhos com o uso de metodologias ágeis [8].

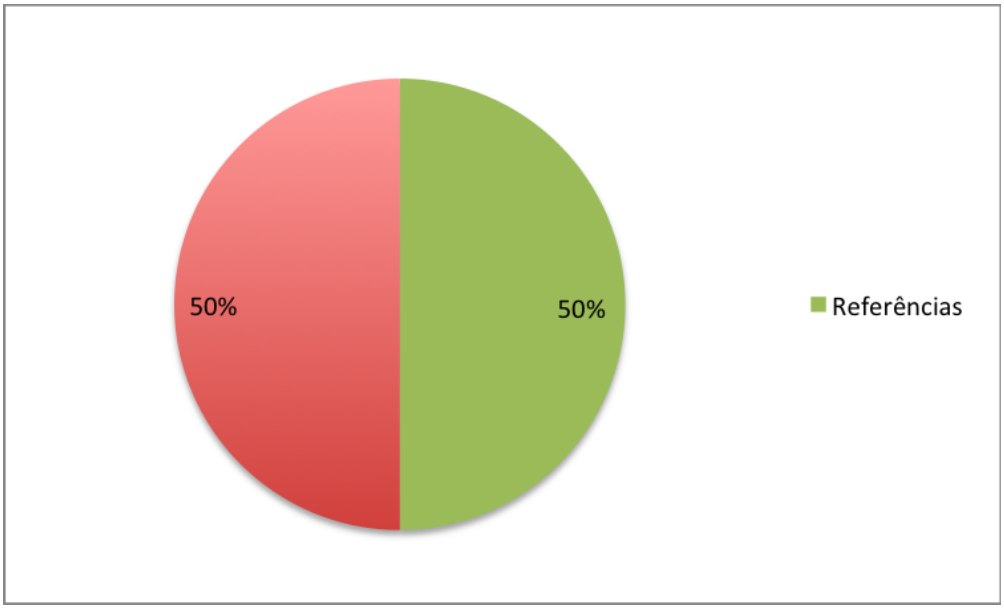


Figura 3.8: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Aspectos técnicos e tecnológicos"

3.2.8 Compartilhamento de conhecimento

Lições aprendidas	**Referências**
Compartilhar conhecimento e experiências é crucial para o sucesso de projetos ágeis	[3], [40], [31], [42], [26], [19], [51], [41], [5], [15]
O ambiente com uma cultura ágil favorece o compartilhamento de conhecimento	[30]

Tabela 3.9: Lições aprendidas agrupadas na categoria "Compartilhamento de conhecimento"

Segundo o Manifesto Ágil [24], indivíduos e interações entre eles devem ser mais valorizados que processos e ferramentas. Outro ponto citado pelo Manifesto é que o método mais eficiente e eficaz de transmitir informações para e por um time de desenvolvimento é através de uma conversa cara a cara. Lagerberg et al. perceberam que Ágil de fato proporciona um ambiente que favorece essa troca de informações, tanto entre projetos como entre pessoas de um mesmo projeto [30]. Muitos trabalhos evidenciaram a importância do compartilhamento de conhecimento para o sucesso de projetos ágeis. Nenhum se opôs a isso.

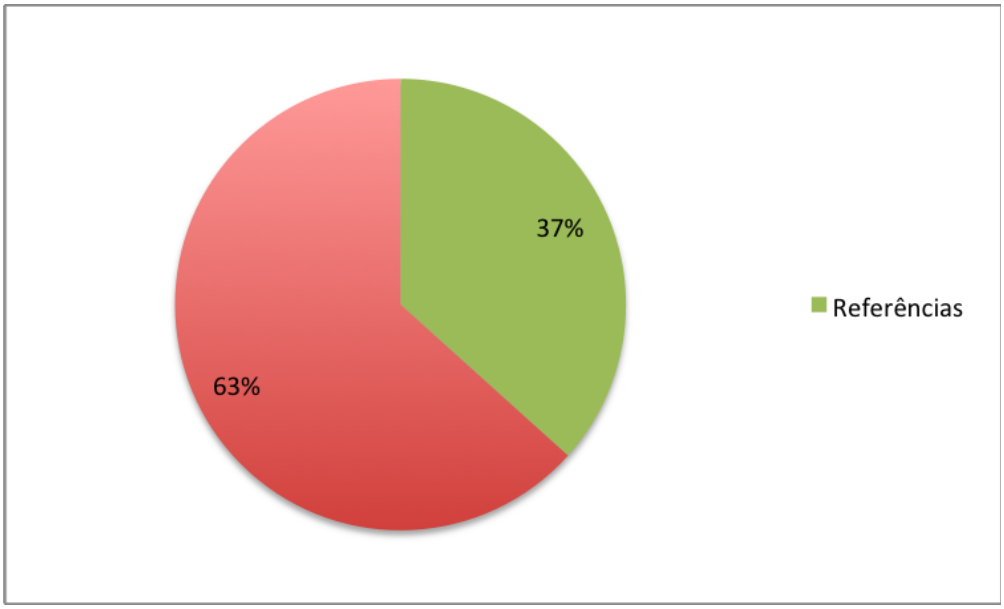


Figura 3.9: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Compartilhamento de conhecimento"

3.2.9 Velocidade de entrega e produtividade

Lições aprendidas	**Referências**
Há um aumento na velocidade/frequência de entrega com Ágil	[22], [21], [36], [40], [28], [26], [37], [46], [41], [15], [25], [48]
Defeitos são corrigidos mais rapidamente	[36], [28]
A velocidade do time é variável	[16]

Tabela 3.10: Lições aprendidas agrupadas na categoria "Velocidade de entrega e produtividade"

A maneira como projetos ágeis são estruturados (iterativos e incrementais) proporciona uma maior quantidade de entregas ao longo do tempo. Esse foi um ponto abordado em diversos trabalhos. No caso de Fitzgerald et al., as releases frequentes e a ativa sincronização com seus clientes fizeram com que solicitações pudessem ser resolvidas num tempo, em média, menor que cinco semanas, fator que contribuiu para uma maior satisfação do cliente [21]. Segundo a pesquisa feita por Claudia et al., o aumento da produtividade e o ganho na capacidade de gerenciamento de mudança de prioridades foram os benefícios mais facilmente percebidos ao se adotar Ágil [37].

Um fator bastante relevante foi apontado por Murphy et al. e Korhonen [36, 28]. Segundo estes artigos, utilizar Ágil não diminui a quantidade de defeitos encontrados no decorrer do projeto. Contudo, o tempo necessário para a correção dos mesmos diminui drasticamente. Comparando o tempo necessário para a resolução de defeitos em projetos ágeis com projetos não-ágeis, cerca de 80% dos defeitos encontrados demoraram mais a serem resolvidos em projetos não-ágeis [28].

Outra lição aprendida observada pela pesquisa foi com relação à variação de velocidade de times ágeis. É natural que, após um certo tempo, times ágeis andem a passos constantes, contudo, de acordo com Piegas e Peres, ocorrer uma variação de velocidade é comum, pois não necessariamente todas as sprints possuem a mesma tonalidade [16].

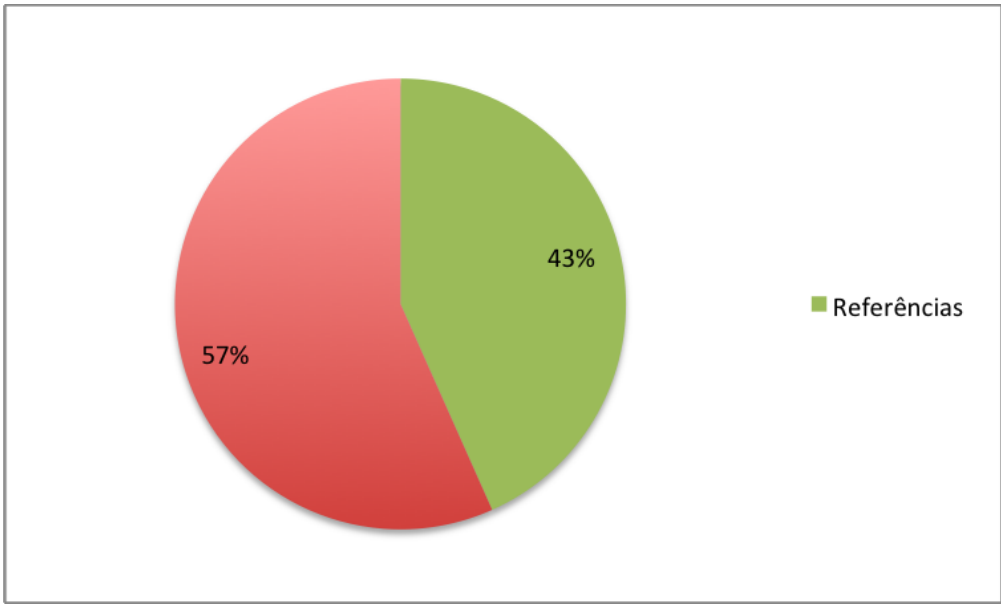


Figura 3.10: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Velocidade de entrega e produtividade"

3.2.10 Qualidade do produto final

Lições aprendidas	**Referências**
A melhoria na qualidade e valor agregado do produto é notória ao se desenvolver utilizando Ágil	[22], [21], [8], [31], [26], [37], [18], [15], [48]
Testar a aplicação durante o seu desenvolvimento reduz riscos	[28], [31], [26], [18], [48]

Tabela 3.11: Lições aprendidas agrupadas na categoria "Qualidade do produto final

O ganho de qualidade com o uso de metodologias ágeis foi outra lição aprendida bastante mencionada. Algumas práticas adotadas por Green foram fundamentais para a obtenção desse ganho: divisão do trabalho em pequenas fatias que agregam valor ao cliente, definição de critérios de aceitação para cada uma dessas pequenas fatias e criação de times multifuncionais com experiência em testes e engenharia de software. Esse ganho de qualidade não é apenas percebido pelo time, mas também pelo cliente [22].

O motivo desse ganho de qualidade, para muitos trabalhos, é decorrente da maneira de se testar aplicações em projetos ágeis. Automatizar o processo de testes, além de trazer muita segurança para o processo de desenvolvimento do software, propicia um curto ciclo de feedback.

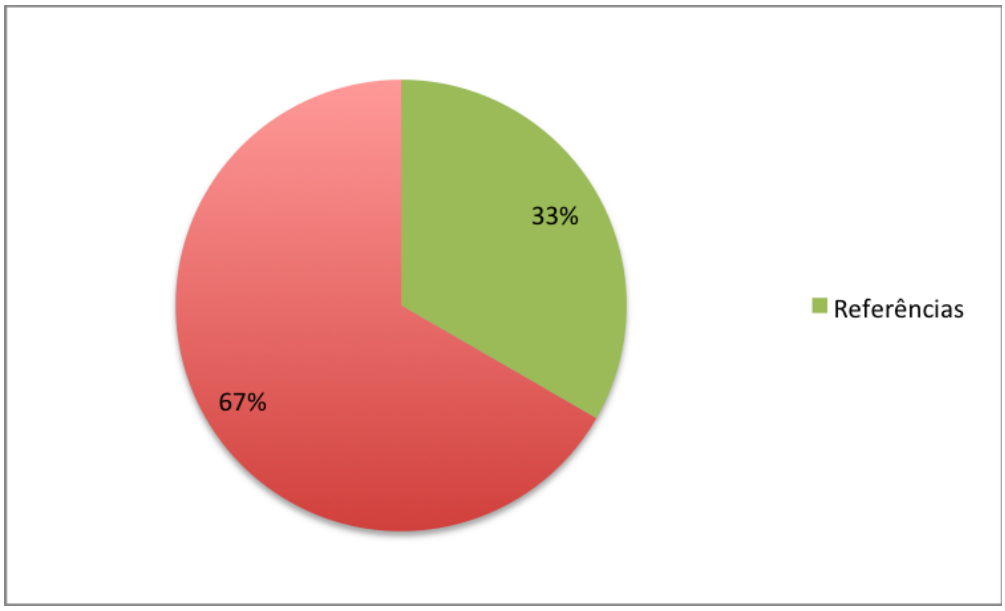


Figura 3.11: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Qualidade do produto final"

3.2.11 Tamanho da organização

Lições aprendidas	**Referências**
Pequenas empresas conseguem obter benefícios mais facilmente ao adotar Ágil	[8]
Escalar Ágil para empresas ou projetos maiores é complicado, porém não é impossível	[36], [37], [28], [15]
É possível implementar Ágil parcialmente	[30]

Tabela 3.12: Lições aprendidas agrupadas na categoria "Tamanho da organização

O Manifesto Ágil não faz menção a restrições com relação ao tamanho da empresa que deve ou não adotar Ágil. Contudo, alguns trabalhos relataram ter passado por muitas dificuldades devido ao porte da organização. Para Bustard et al., empresas menores e mais horizontais conseguem obter benefícios muito mais facilmente [8]. Vários trabalhadores consultados por Murphy et al. apontaram que escalabilidade pode ser um problema ao se tentar praticar Ágil [36].

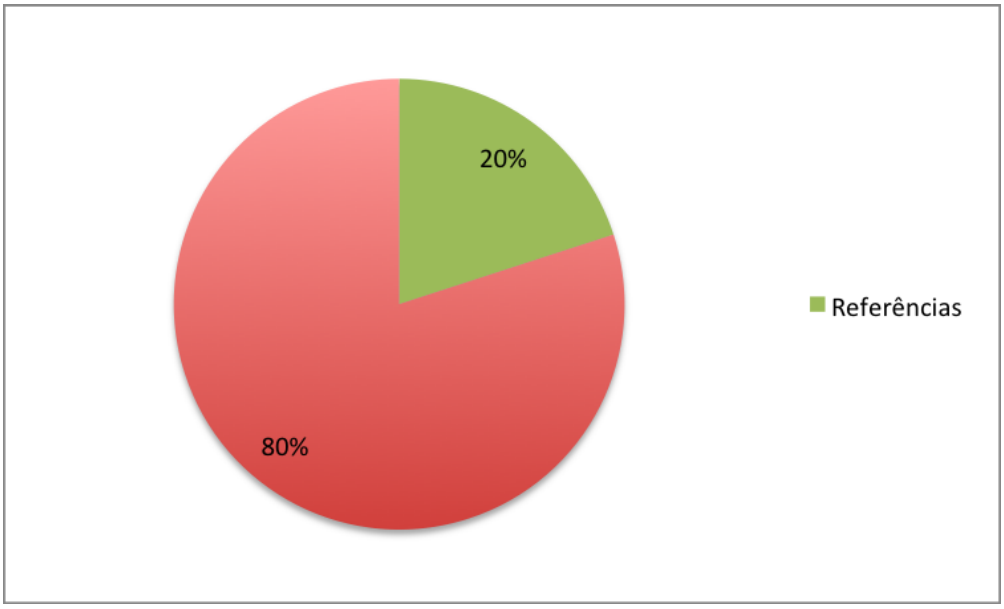


Figura 3.12: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Tamanho da organização"

3.2.12 Quebra de paradigma

Lições aprendidas	**Referências**
É difícil lidar com a quebra de paradigma para a implementação de Ágil	[23], [6], [28], [31], [2], [46], [5], [15]
A maneira de gerenciar/implementar os testes de softwares construídos em projetos ágeis é desafiadora	[28]
Colocar princípios acima de práticas	[15], [18], [25], [48], [45]

Tabela 3.13: Lições aprendidas agrupadas na categoria "Quebra de paradigma"

Muitas empresas tentam adotar Ágil com a finalidade de apenas usufruir de seus benefícios, contudo, não param para refletir que são fundamentalmente incompatíveis com muitos dos princípios ágeis [31, 5]. Fazer com que a alta gerência compreenda que é necessário modificar a essência da organização ao invés de simplesmente adotar um conjunto de práticas é algo bastante desafiador. Muitos dos trabalhos analisados mostraram ter apresentado problemas com isso. É preciso colocar os princípios acima das práticas [15].

Uma das maiores quebras de paradigma está relacionada ao processo de testes ágeis [28]. Passar a testar a aplicação durante o seu processo de desenvolvimento requer muita disciplina e capacidade técnica de implementação.

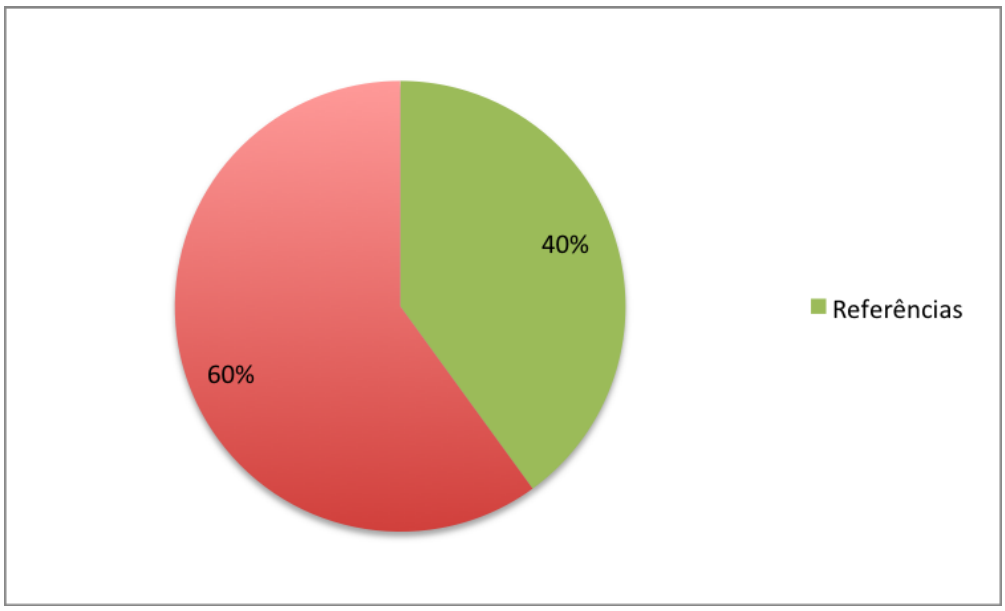


Figura 3.13: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Quebra de paradigma"

3.2.13 Comunicação remota

Lições aprendidas	**Referências**
Coordernar atividades com times distribuídos é algo bastante desafiador	[22], [36], [28], [42], [20], [51], [5], [15]
Sincronizar boards virtual e real não é uma tarefa fácil	[51]
Dificuldade ao trabalhar com equipes externas de testes	[5]

Tabela 3.14: Lições aprendidas agrupadas na categoria "Comunicação remota"

Dividir para conquistar é uma estratégia bastante antiga e que, muitas vezes, também se aplica à Ágil. Entretanto, muitas empresas se mostraram incapazes de gerenciar suas atividades diárias ao utilizar desenvolvimento distribuído de software. Atividades corriqueiras como trabalhar com equipes externas de testes [5] e sincronização de boards [51] se mostraram problemáticas.

Isso não ocorreu apenas com empresas pequenas e inexperientes. Grandes companhias também passaram por problemas dessa natureza. Uma pesquisa executada na Adobe relatou que, ao adotar uma abordagem distribuída em um de seus projetos, atividades corriqueiras como Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective se mostraram desafiadoras [22].

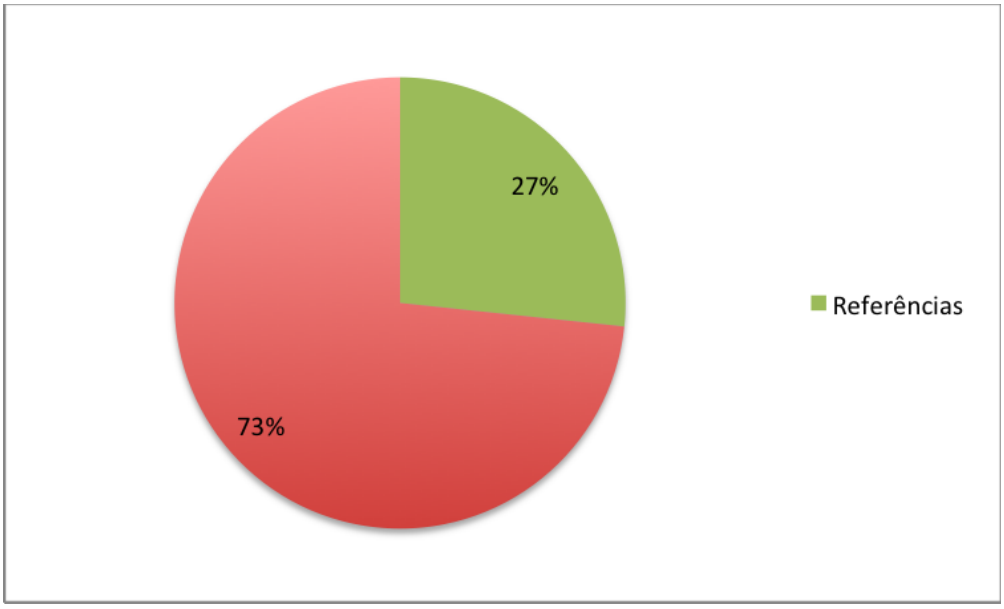


Figura 3.14: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Comunicação remota"

3.2.14 Cultura organizacional

Lições aprendidas	**Referências**
Métodos ágeis e ambientes altamente controlados são culturalmente incompatíveis	[21]
Empresas que discordam de princípios ágeis tendem a falhar caso tentem implantar tais métodos	[8], [36], [37], [29], [45], [15]
Tentar influenciar/remodelar a cultura da organização é uma tarefa desafiadora	[26], [20], [5], [45], [10], [15]

Tabela 3.15: Lições aprendidas agrupadas na categoria "Cultura organizacional"

As lições aprendidas dessa categoria estão intimamente correlacionadas com as encontradas na categoria "Quebra de paradigma", porém com um foco mais voltado para a cultura da organização que está tentando adotar Ágil. Aqui ocorreu uma unanimidade. De acordo com as experiências das empresas cujo material foi analisado, tentar influenciar ou remodelar a

cultura da organização não é um processo trivial. Nessa situação, o risco de falha do projeto é bastante elevado.

É preciso abraçar a cultura Ágil e realmente acreditar em seus princípios.

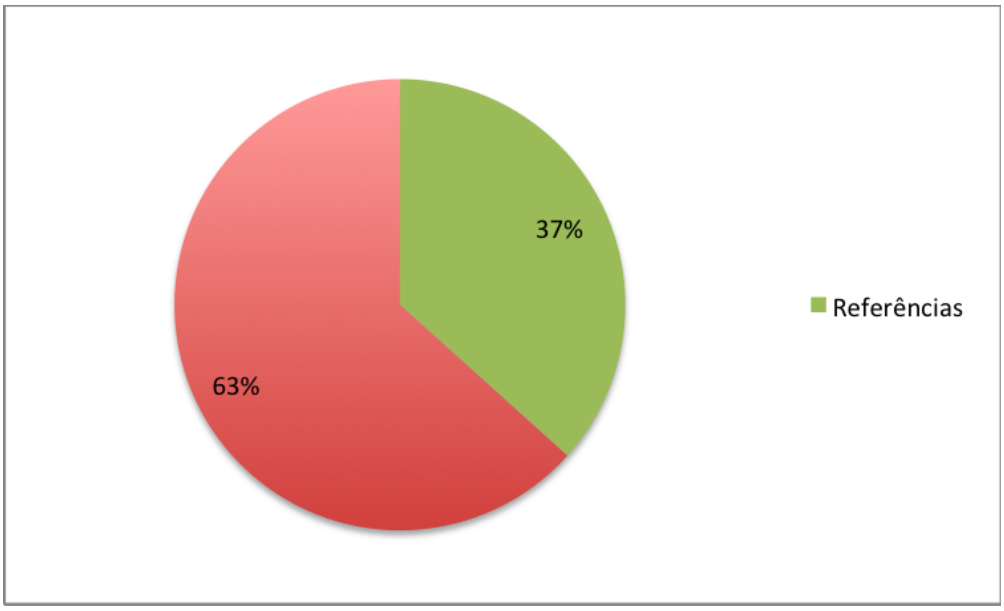


Figura 3.15: Percentual de trabalhos que referenciaram lições aprendidas da categoria "Cultura organizacional"

Capítulo 4

Validação das Categorias de Lições Aprendidas

Para se assegurar de sua relevância, as categorias de lições aprendidas mapeadas no capítulo anterior foram validadas por profissionais com experiência em adoção ágil atuantes no mercado de TI brasileiro.

4.1 Método de validação

Conforme descrito em seções anteriores, após uma revisão de literatura baseada no método de revisão sistemática de artigos publicados em revistas e conferências científicas, complementada por uma pesquisa exploratória de relatos de experiência nas principais conferências nacionais e internacionais sobre o assunto, um conjunto de lições aprendidas foi coletado. Essas lições foram agrupadas considerando suas similaridades em 14 categorias.

4.1.1 Definição do objetivo

O primeiro passo foi definir a questão a ser validada pelo questionário: "*As categorias das lições coletadas são relevantes para a comunidade?*".

4.1.2 Elaboração do questionário

Com o intuito de validar a relevância dessas categorias de lições aprendidas propostas por essa pesquisa, um questionário foi elaborado para coletar a opinião de profissionais que vivenciaram a adoção organizacional da abordagem ágil. A metodologia de survey proposta por Babbie em 1990 foi levada em consideração na elaboração do questionário [4]. São 15 questões de múltipla escolha cujo foco é a criticidade de cada categoria para que a adoção de Ágil em empresas de desenvolvimento de software ocorra com sucesso.

4.1.3 Definição da população pesquisada

Para ter um melhor controle com relação à competência e ao nível de experiência dos profissionais consultados, o questionário não foi disponibilizado publicamente. Foram considerados especialistas com reconhecimento nacional através de artigos publicados / palestras em conferências ou profissionais experientes que passaram pela adoção de metodologias ágeis em suas empresas.

Os profissionais consultados têm, em média, 14 anos de experiência com desenvolvimento de software e cerca de 8 anos com Ágil. A Tabela 4.1 apresenta um breve resumo do perfil profissional daqueles que se identificaram (isto não foi obrigatoriamente solicitado no formulário). De acordo com a tabela, pode-se observar que os perfis dos respondentes atenderam aos critérios estabelecidos.

Nome	**Resumo**
Luca Bastos	Engenheiro, empreendedor por 17 anos e desenvolvedor desde os tempos dos cartões perfurados. Organizador e palestrante no Agile Brazil 2012 e 2013. Keynoter no TWBRAway Day 2012. Palestras: CaipiraAgile 2012, TDC 2012, Qcon 2011, AgileBrazil 2011, TDC 2011, RubyConf 2010, Qcon 2010, TDC 2010 e outros. Atualmente trabalha na ThoughtWorks

Celso Martins	Atuando no mercado de TI desde 1994, teve os primeiros contatos com o desenvolvimento ainda em 1985, com 7 anos de idade. Tendo passado por grandes empresas, como Embratel, Accenture, Oi, XP Investimentos, hoje atua como consultor da Crafters Software Studio, ajudando a GVT no processo de transição para gestão ágil. Também tem se dedicado, de forma teórica e prática, ao assunto mudança organizacional nos últimos 2 anos. Especialista em Lean, Kanban, Scrum, Management 3.0, Sistemas Complexos, ToC, dispondo de uma caixa de ferramentas interessante para ajudar as empresas a realizar a mudança para a gestão moderna de uma forma suave e com o menor impacto possível. Nas horas vagas, é organizador de eventos voltados à Ágil, como o Agile Trends
Felipe Furtado	Possui mestrado em Ciências da Computação pelo Centro de Informática da UFPE na área de métricas e produtividade. Atualmente é doutorando no CIn em metodologias ágeis e modelos de maturidade, gerente de projetos do C.E.S.A.R, coordenador da pós-graduação em gestão ágil de projetos do C.E.S.A.R.edu e docente do mestrado profissional em engenharia de software. Com 16 anos na área de desenvolvimento de software, foi coordenador da garantia da qualidade de software e do SEPG (Software Engineering Process Group) onde adquiriu experiência na área de engenharia de software com ênfase em qualidade de software e gestão de projetos
Célio Santana	Doutor em Engenharia de Software pela Universidade Federal de Pernambuco (UFPE). Professor na UFPE e consultor em melhoria de processos
João Paulo Novais	Engenheiro de Software com mais de 10 anos de experiência. Analista de Negócio no SERPRO, Agile Coach e organizador de eventos da Comunidade Ágil como Agile Brazil e Agile Trends

Paulo Caroli	Agilista e protagonista da ThoughtWorks Brasil, Paulo Caroli possui mais de quinze anos de experiência em desenvolvimento de software, com passagem em várias corporações no Brasil, Índia e EUA. Em 2000, conheceu o Extreme Programming e, desde então, tem focado sua experiência em processos e práticas de Gestão e Desenvolvimento Ágil. Ingressou na ThoughtWorks em 2006 e ocupou os cargos de Agile Coach, Trainer, e Gerente de Projetos. Possui os títulos de Bacharel em Ciência da Computação e M.S. em Engenharia de Software, ambos da PUC-Rio
Marcos Garrido	Quase duas décadas de experiência em projetos de tecnologia e atuando como Product Owner em projetos ágeis desde 2008. Além disso, é professor da disciplina de Métodos Ágeis do MBA em Gestão de Projetos da PUC-RIO. Garrido é um membro ativo das comunidades Ágil e Lean, com forte interesse pelos desafios que envolvem a gestão de produtos. Participou como palestrante em um grande número de eventos nacionais e internacionais, com destaque para os Scrum Gatherings de São Paulo, Munich e Orlando, além do Agile Brazil, Ágiles e Agile Rio. Nos últimos anos, tem se envolvido profundamente com o movimento Lean Startup. Garrido é Mestre em Administração com ênfase em Marketing pela PUC-RIO e possui as certificações CSPO, CSM, CSD e CSP pela Scrum Alliance, além de ser Facilitador certificado de Management 3.0
Dairton Bassi	Possui mestrado em Engenharia de Software e bacharelado em Ciência da Computação pelo IME-USP. Co-fundador da Agile Alliance Brazil, Co-criador do Agile Brazil e Coordenador Geral do Agile Brazil 2012. Envolvido com agilidade há 10 anos, foi CTO da Dafiti e, como consultor, implantou metodologias ágeis em inúmeras empresas

Renato Willi	Tem trabalhado com desenvolvimento de software a cerca de 12 anos, sendo os últimos 9 coordenando times através de metodologias ágeis. Atualmente é associado da SEA Tecnologia, uma empresa com mais de 10 anos de estrada. Renato também trabalha como organizador e/ou palestrante em vários eventos nacionais sobre metodologias ágeis (Agile Brazil, Maré de Agilidade, QCon, TDC, etc.), além de escrever artigos para blogs e revistas do ramo e colaborar com projetos voluntários de tradução de livros técnicos

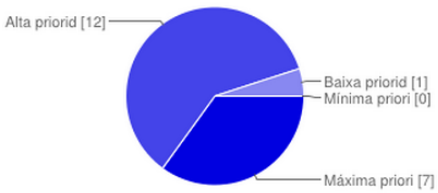
Tabela 4.1: Resumo sobre alguns profissionais da área de TI que responderam o formulário e se identificaram

4.1.4 Aplicação do questionário

Foi utilizada a plataforma Google Docs para disponibilização online do questionário. Dado que ele não estava publicamente disponível, foi necessário enviar seu link por email para aqueles especialistas considerados dentro do perfil previamente estabelecido. Foram recebidas 20 respostas dentro do prazo estipulado.

4.1.5 Análise dos resultados

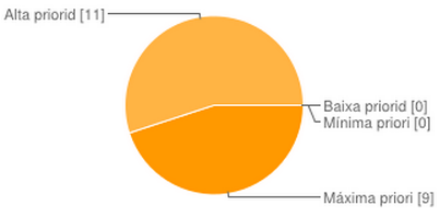
Qual nível de prioridade você daria para a experiência, treinamento e aprendizado dos envolvidos?



Máxima prioridade, pois isto é essencial para todo time Ágil	7	35%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	12	60%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	1	5%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	0	0%

Figura 4.1: Contabilização das respostas referentes à criticidade da categoria de LAs "Experiência, treinamento e aprendizado"

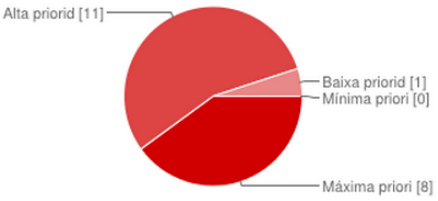
Qual nível de prioridade você daria para o planejamento e gerenciamento do backlog do projeto?



Máxima prioridade, pois isto é essencial para todo time Ágil	9	45%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	11	55%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	0	0%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	0	0%

Figura 4.2: Contabilização das respostas referentes à criticidade da categoria de LAs "Planejamento e gerenciamento de backlog"

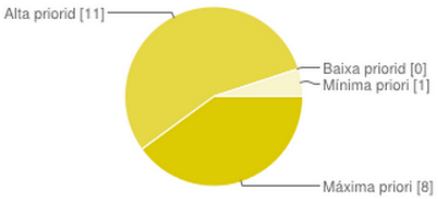
Qual nível de prioridade você daria para o apoio gerencial e dos clientes?



Máxima prioridade, pois isto é essencial para todo time Ágil	8	40%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	11	55%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	1	5%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	0	0%

Figura 4.3: Contabilização das respostas referentes à criticidade da categoria de LAs "Apoio gerencial e dos clientes"

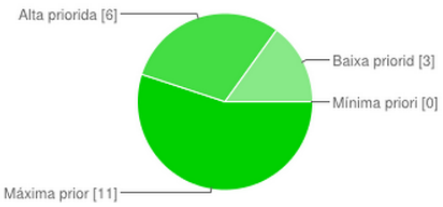
Qual nível de prioridade você daria para a capacidade de customização e adaptabilidade do projeto?



Máxima prioridade, pois isto é essencial para todo time Ágil	**8**	40%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	**11**	55%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	**0**	0%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	**1**	5%

Figura 4.4: Contabilização das respostas referentes à criticidade da categoria de LAs "Customização e adaptabilidade"

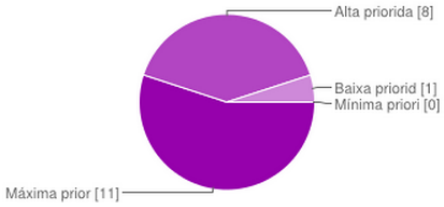
Qual nível de prioridade você daria para o nível de confiança do time?



Máxima prioridade, pois isto é essencial para todo time Ágil	**11**	55%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	**6**	30%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	**3**	15%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	**0**	0%

Figura 4.5: Contabilização das respostas referentes à criticidade da categoria de LAs "Confiança do time"

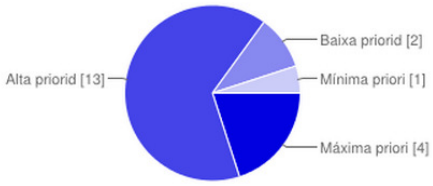
Qual nível de prioridade você daria para o nível de engajamento, comprometimento, disciplina e trabalho em equipe dos integrantes do projeto?



Máxima prioridade, pois isto é essencial para todo time Ágil	**11**	55%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	**8**	40%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	**1**	5%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	**0**	0%

Figura 4.6: Contabilização das respostas referentes à criticidade da categoria de LAs "Engajamento, comprometimento, disciplina e trabalho em equipe"

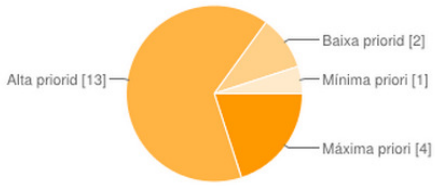
Qual nível de prioridade você daria para os aspectos técnicos e tecnológicos utilizados?



Máxima prioridade, pois isto é essencial para todo time Ágil	4	20%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	13	65%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	2	10%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	1	5%

Figura 4.7: Contabilização das respostas referentes à criticidade da categoria de LAs "Aspectos técnicos e tecnológicos"

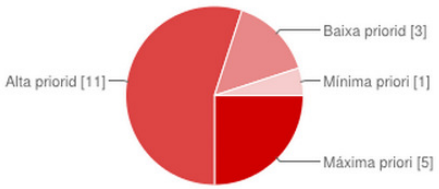
Qual nível de prioridade você daria para o hábito do compartilhamento de conhecimento dentro e fora do projeto?



Máxima prioridade, pois isto é essencial para todo time Ágil	4	20%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	13	65%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	2	10%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	1	5%

Figura 4.8: Contabilização das respostas referentes à criticidade da categoria de LAs "Compartilhamento de conhecimento"

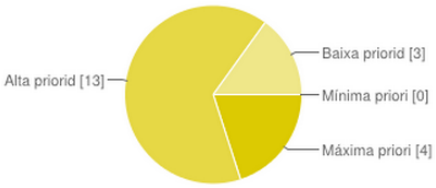
Qual nível de prioridade você daria para a velocidade de entrega e a produtividade da equipe?



Máxima prioridade, pois isto é essencial para todo time Ágil	5	25%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	11	55%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	3	15%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	1	5%

Figura 4.9: Contabilização das respostas referentes à criticidade da categoria de LAs "Velocidade de entrega e produtividade"

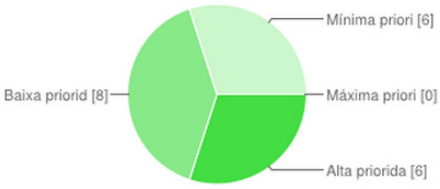
Qual nível de prioridade você daria para a busca pela melhoria na qualidade do produto final?



Máxima prioridade, pois isto é essencial para todo time Ágil	4	20%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	13	65%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	3	15%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	0	0%

Figura 4.10: Contabilização das respostas referentes à criticidade da categoria de LAs "Qualidade do produto final"

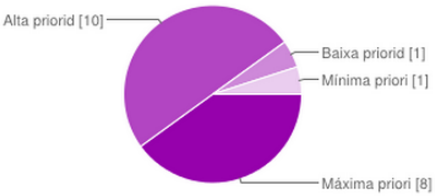
Qual nível de prioridade você daria para o tamanho da organização que está tentando adotar Ágil?



Máxima prioridade, pois isto é essencial para todo time Ágil	0	0%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	6	30%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	8	40%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	6	30%

Figura 4.11: Contabilização das respostas referentes à criticidade da categoria de LAs "Tamanho da organização"

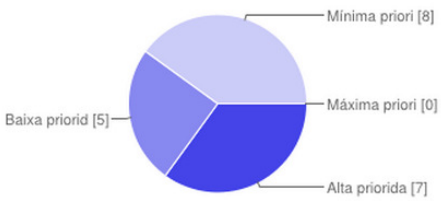
Qual nível de prioridade você daria para os esforços com a quebra de paradigma necessária para a implementação de Ágil?



Máxima prioridade, pois isto é essencial para todo time Ágil	8	40%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	10	50%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	1	5%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	1	5%

Figura 4.12: Contabilização das respostas referentes à criticidade da categoria de LAs "Quebra de paradigma"

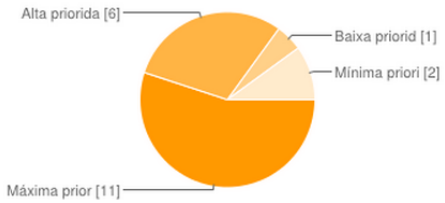
Qual nível de prioridade você daria para o esforço para se lidar com comunicação remota?



Máxima prioridade, pois isto é essencial para todo time Ágil	**0**	0%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	**7**	35%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	**5**	25%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	**8**	40%

Figura 4.13: Contabilização das respostas referentes à criticidade da categoria de LAs "Comunicação remota"

Qual nível de prioridade você daria para o alinhamento da cultura da organização com os Princípios Ágeis?



Máxima prioridade, pois isto é essencial para todo time Ágil	**11**	55%
Alta prioridade, pois isto aumenta bastante as chances de sucesso do projeto	**6**	30%
Baixa prioridade, pois isto não necessariamente trás ganhos significativos	**1**	5%
Mínima prioridade, pois times Ágeis podem trabalhar sem isto	**2**	10%

Figura 4.14: Contabilização das respostas referentes à criticidade da categoria de LAs "Cultura organizacional"

Se você tivesse que selecionar apenas 5 das categorias relacionadas anteriormente, quais você ressaltaria como as mais críticas para serem consideradas por uma organização na adoção da abordagem ágil?

Experiência, treinamento e aprendizado	11	11%
Planejamento e gerenciamento de backlog	8	8%
Apoio gerencial e dos clientes	9	9%
Customização e adaptabilidade	8	8%
Confiança do time	8	8%
Engajamento, comprometimento, disciplina e trabalho em equipe	17	17%
Aspectos técnicos e tecnológicos	7	7%
Compartilhamento de conhecimento	5	5%
Velocidade de entrega e produtividade	3	3%
Qualidade do produto final	8	8%
Tamanho da organização	1	1%
Quebra de paradigma	3	3%
Comunicação remota	0	0%
Cultura organizacional	12	12%

Figura 4.15: Contabilização das respostas referentes às categorias de LAs consideradas mais importantes

Através do questionário utilizado foi possível perceber que há uma grande relevância nas categorias de lições aprendidas propostas por essa pesquisa. Segundo seu resultado, 12 das 14 categorias foram consideradas de máxima ou alta prioridade por, no mínimo, 80% dos profissionais consultados. As categorias consideradas menos críticas foram: "Tamanho da organização"e "Comunicação remota".

Outro resultado apontado pelo questionário foi o comparativo, no quesito criticidade, entre categorias de LAs. As mais citadas pelos profissionais da área de TI foram: "Engajamento, comprometimento, disciplina e trabalho em equipe", "Cultura organizacional"e "Experiência, treinamento e aprendizado". Comprovou-se também que "Tamanho da organização"e "Comunicação remota", segundo aqueles que responderam o questionário, não devem ser priorizadas, pois foram as categorias menos referenciadas na última questão.

Capítulo 5

Considerações Finais

Nesse capítulo consta a conclusão da pesquisa, bem como possíveis trabalhos futuros.

5.1 Conclusão da pesquisa

A cada dia, mais organizações de desenvolvimento de software estão modificando seus processos internos e adotando metodologias ágeis. Esta mudança ocorre devido a uma contínua busca por melhorias na produtividade e qualidade dos softwares construídos. Entretanto, modificar processos não é uma tarefa fácil, exige esforço e dedicação das diversas partes envolvidas, pois geralmente é preciso enfrentar muita resistência durante a transição.

Apoio gerencial e dos clientes, problemas com novas tecnologias, comprometimento e disciplina, incompatibilidades entre princípios (Ágil x Organização) e falta de experiência foram alguns dos desafios mais encontrados nos relatos pesquisados. A maioria dessas dificuldades pode ser resolvida através do compartilhamento de conhecimento. Como é possível evitar cometer erros aprendendo através de relatos de experiências compartilhados por outras organizações, a proposta dessa pesquisa foi criar uma fonte com informações úteis relacionadas a processos de adoção ágil em empresas de desenvolvimento de software.

As lições aprendidas encontradas foram agrupadas em 14 categorias validadas, através de um questionário, por especialistas atuantes no mercado de TI brasileiro. Das 14 categorias, 12

foram consideradas de máxima ou alta prioridade por, no mínimo, 80% dos profissionais consultados. Dentre as mais importantes, estão: “Engajamento, comprometimento, disciplina e trabalho em equipe”, “Cultura organizacional”e “Experiência, treinamento e aprendizado”.

Um alto percentual dos especialistas consultados considerou de grande relevância a maioria das categorias de lições aprendidas propostas por essa pesquisa. Assim, como o objetivo geral desse trabalho foi gerar um conjunto de lições aprendidas relevante para a adoção de métodos ágeis por organizações de software, os resultados obtidos corroboraram com seu cumprimento.

5.2 Trabalhos futuros

A partir desse estudo, uma série de outros trabalhos podem surgir. O mais imediato dentre eles seria um conjunto de artigos a serem publicados e apresentados nas principais conferências ágeis do Brasil em 2014. Num prazo mais longo, realizar a criação de uma plataforma colaborativa open-source de relatos de experiência e lições aprendidas com adoção ágil.

Observar a contribuição das lições aprendidas coletadas por essa pesquisa para a facilitação do processo de adoção ágil por empresas de desenvolvimento de software seria também um possível trabalho futuro.

Referências Bibliográficas

[1] T. Wissink; C. Amaro. Strategies for agile software testing automation: An industrial experience. In *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on*, pages 265–266, September 2006.

[2] Adenike O. Arikpo, Iwara I.; Osofisan. Software tools and processes: A key factor in successful agile adoption. *Computing and Information Systems*, 15:14–26, Feb 2011.

[3] A.L. Asnawi, A.M. Gravell, and G.B. Wills. Factor analysis: Investigating important aspects for agile adoption in malaysia. In *AGILE India (AGILE INDIA), 2012*, pages 60–63, Feb 2012.

[4] R. Babbie. *Survey research methods*. Wadsworth Pub. Co., 1990.

[5] Ana Bastos. Adoção de práticas ágeis no desenvolvimento de software de missão crítica. Apresentado em: Agile Brazil, 2013. Disponível em: `http://www.slideshare.net/AnaHeloisaBastos/agile-brazil2013-adocaoagilecossistema`. Acessado em: 14/02/2014.

[6] M. Block. Evolving to agile: A story of agile adoption at a small saas company. In *Agile Conference (AGILE), 2011*, pages 234–239, Aug 2011.

[7] IEEE Standards Board. Ieee guide for software verification and validation plans. Technical report, The Institute of Electrical and Electronics Engineers, Inc, December 1993.

[8] D. Bustard, G. Wilkie, and D. Greer. The maturation of agile software development principles and practice: Observations on successive industrial studies in 2010 and 2012.

In *Engineering of Computer Based Systems (ECBS), 2013 20th IEEE International Conference and Workshops on the*, pages 139–146, April 2013.

[9] FANTINATO M.; CUNHA A.; DIAS S.; MIZUNO S.; E CUNHA C. Um framework reutilizável para a automação de teste funcional de software. In *SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE*, May 2004.

[10] Srinath Chandrasekharan. Agile in it services. Apresentado em: Agile Goa, 2012. Disponível em: `http://pt.slideshare.net/Srinath_cs/agile-goa-final?from_search=4`. Acessado em: 14/02/2014.

[11] E. Collins and Lucena V. F. Dias-Neto. Strategies for agile software testing automation: An industrial experience. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pages 440–445, July 2012.

[12] James Crisp. Automated testing and the test pyramid, May 2011. Disponível em: `http://jamescrisp.org/2011/05/30/automated-testing-and-the-test-pyramid/`. Acessado em: 30/11/2014.

[13] James Crisp. Yet another software testing pyramid, June 2011. Disponível em: `http://watirmelon.com/2011/06/10/yet-another-software-testing-pyramid/`. Acessado em: 30/11/2014.

[14] L. M CYSNEIROS. Integrando requisitos não funcionais ao processo de desenvolvimento de software. Master's thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio deJaneiro, 1997.

[15] Teresa Maciel e Alessandra Mendes. A incrível história de uma organização pública que acredita em agilidade. Apresentado em: Agile Brazil, 2013. Disponível em: `http://pt.slideshare.net/tmmaciel/a-incrvel-hist`. Acessado em: 14/02/2014.

[16] André Piegas e Eduardo Peres. Abolições aprendidas. Apresentado em: Agile Brazil, 2012. Disponível em: `http://www.slideshare.net/eduardomeiraperes/abolies-aprendidas`. Acessado em: 14/02/2014.

[17] Ardita Karaj e Jason Little. Evolving to agile: transforming a public sector organization. Apresentado em: Agile, 2013. Disponível em: `http://www.slideshare.net/agilecoach/agile-2013-from-stone-age-to-agile`. Acessado em: 14/02/2014.

[18] Thais Dalcin e Luiz C. Parzinello. Implantando a cultura Ágil em larga escala. Apresentado em: Agile Brazil, 2012. Disponível em: `http://www.slideshare.net/parzianello/implantando-a-cultura-gil-em-larga-escala-o-case-do-grupo-rbs`. Acessado em: 14/02/2014.

[19] Guilherme Ferreira e Rafael Valério. 4 atitudes para melhorar a agilidade de uma empresa. na prática. Apresentado em: Agile Brazil, 2013. Disponível em: `http://www.slideshare.net/rafaelvalerio311/4-atitudes-para-melhorar-a-agilidade-de-uma-empresa-na-prtica`. Acessado em: 14/02/2014.

[20] Zé Rodrigues e Vinícius Silva. Padrões e antipadrões da adoção da agilidade em governo. Apresentado em: Agile Brazil, 2013. Disponível em: `http://www.slideshare.net/Zeh/padres-a-antipadres-da-adoo-da-agilidade-agile-2013`. Acessado em: 14/02/2014.

[21] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, and D. O'Brien. Scaling agile methods to regulated environments: An industry case study. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 863–872, May 2013.

[22] P. Green. Adobe premiere pro scrum adoption: How an agile approach enabled success in a hyper-competitive landscape. In *Agile Conference (AGILE), 2012*, pages 172–178, Aug 2012.

[23] H. Hajjdiab and A.S. Taleb. Agile adoption experience: A case study in the u.a.e. In *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*, pages 31–34, July 2011.

[24] Jim Highsmith and Martin Fowler. The agile manifesto. *Software Development Magazine*, 9:29–30, 2001.

[25] Alexis Hui. Lean change: Enabling agile transformation through lean startup, kanban, and kotter. Apresentado em: Agile, 2013. Disponível em: `https://submissions.agilealliance.org/system/attachments/attachments/000/000/056/original/Agile_2013_-_Lean_Change_for_Enabling_Agile_Transformations.pdf?1385085655`. Acessado em: 14/02/2014.

[26] Eunha Kim and Seokmoon Ryoo. Agile adoption story from nhn. In *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, pages 476–481, July 2012.

[27] A. M. KIRNER, T. G.; DAVIS. Nonfunctional requirements of real-time systems. *Advances in Computers*, 42:1–37, 1996.

[28] Kirsi Korhonen. Evaluating the effect of agile methods on software defect data and defect reporting practices - a case study. In *Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology*, QUATIC '10, pages 35–43, Washington, DC, USA, 2010. IEEE Computer Society.

[29] Kirsi Korhonen. Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context. *Software Quality Journal*, 21(4):599–624, 2013.

[30] L. Lagerberg, T. Skude, P. Emanuelsson, K. Sandahl, and D. Stahl. The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two projects at ericsson. In *Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium on*, pages 348–356, Oct 2013.

[31] Mary Ann Lapham. Dod agile adoption: Necessary considerations, concerns, and changes. *CrossTalk Magazine*, pages 31–35, Jan 2012.

[32] Tutorials Point India Private Limited. Software testing-overview. Disponível em: `http://www.tutorialspoint.com/software_testing/software_testing_overview.htm`. Acessado em 30/12/2014.

[33] BUDNIK C. J.; CHAN W. K.; KAPFHAMMER G. M. Bridging the gap between the theory and practice of software test automation. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on (Volume:2)*, pages 445–446, May 2010.

[34] J. C. Maldonado and A. Vincenzi. Aspectos teóricos e empíricos de testes de cobertura de software. Technical report, Instituto de Ciencias Matemáticas e de Computação - ICMC-USP, June 1998.

[35] Wanessa Mariana and Angelica Toffano. Ferramentas free para teste de software: um estudo comparativo. Technical report, Centro Universitario de Brasilia-Uniceub, August 2012.

[36] B. Murphy, C. Bird, T. Zimmermann, L. Williams, N. Nagappan, and A. Begel. Have agile techniques been the silver bullet for software development at microsoft? In *Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium on*, pages 75–84, Oct 2013.

[37] Claudia O. Melo, Viviane Santos, Eduardo Katayama, Hugo Corbucci, Rafael Prikladnicki, Alfredo Goldman, and Fabio Kon. The evolution of agile software development in brazil. *Journal of the Brazilian Computer Society*, 19(4):523–552, 2013.

[38] Fabio Kon Paulo Cheque. A importancia dos testes automatizados. *Engenharia de Software Magazine*, 72:54–57, 2008.

[39] Bret Pettichord. Seven steps to test automation success. STAR West, San Jose, November 1999. Disponível em: `http://https://www.prismnet.com/~wazmo/papers/seven_steps`. Acessado em: 30/11/2014.

[40] K. Power. The agile office: Experience report from cisco's unified communications business unit. In *Agile Conference (AGILE), 2011*, pages 201–208, Aug 2011.

[41] Vitor Queiroz. Agile black ops - como infiltrar agile em ambiente hostil. Apresentado em: Agile Brazil, 2013. Disponível em: `http://www.slideshare.net/VitorQ/agile-black-ops`. Acessado em: 14/02/2014.

[42] Dr. Vinaya Babu Radha Shankarmani, Dr. S. S. Mantha. Inclusion of e-assist to increase agile adoption. *International Journal of Scientific and Research Publications (IJSRP)*, 2, Nov 2012.

[43] Elfriede Dustin; J. Rashka; and J. Paul. *The Art of Agile Development*. Addison-Wesley, first edition, 1999.

[44] Pressman; R.S. Software engineering: A practitioners approach. Technical report, McGraw Hill, Inc, 2002.

[45] Michael Sahota. An agile adoption and transformation survival guide. Apresentado em: Agile, 2012. Disponível em: http://pt.slideshare.net/michael.sahota/agile-2012-an-agile-adoption-and-transformation-survival-guide?from_search=10. Acessado em: 14/02/2014.

[46] Stéfano Dos Santos. Desafios do desenvolvimento Ágil para o governo. Apresentado em: Agile Brazil, 2013. Disponível em: http://www.slideshare.net/stefanohs/desafios-do-desenvolvimento-gil-para-o-governo. Acessado em: 14/02/2014.

[47] Alister Scott. Introducing the software testing ice-cream cone (anti-pattern), January 2012. Disponível em: http://watirmelon.com/2012/01/31/introducing-the-software-testing-ice-cream-cone/. Acessado em: 25/11/2014.

[48] Ahmed Sidky. Value-driven agile adoption. Apresentado em: Agile Mumbai, 2008. Disponível em: http://pt.slideshare.net/nashjain/value-driven-agile-adoption?from_search=4. Acessado em: 14/02/2014.

[49] RECIFE SOFTEX. Fundamentos do teste de software. Disponível em: http://ava.nac.softex.br/pluginfile.php/602/mod_resource/content/2/Aula%202.pdf. Acessado em 02/01/2015.

[50] Pachawan Augsornsri; Taratip Suwannasart. An integration testing coverage tool for object-oriented software. In *Information Science and Applications (ICISA), 2014 International Conference on*, pages 1–5, May 2014.

[51] Rodrigo Vieira. Agilidade das trincheiras do tribunal superior do trabalho. Apresentado em: Agile Brazil, 2013. Disponível em: http://www.slideshare.net/rcvieira/agilidade-das-trincheiras-do-tribunal-superior-do-trabalho. Acessado em: 14/02/2014.

[52] K.N. Leung Hareton; Lee White. A study of integration testing and software regression at the integration level. In *Software Maintenance, 1990, Proceedings., Conference on*, pages 290–301, November 1990.

[53] Weinan Shanxi XiangFeng Meng. Analysis of software automation test protocol. In *International Conference on Electronic and Mechanical Engineering and Information Technology*, pages 4138 – 4141, August 2011.