Pós-Graduação em Ciência da Computação

ANA PAULA CARVALHO CAVALCANTI FURTADO

FAST:

UM FRAMEWORK PARA AUTOMAÇÃO DE TESTE

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

ANA PAULA CARVALHO CAVALCANTI FURTADO

FAST: UM FRAMEWORK PARA AUTOMAÇÃO DE TESTE

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Centro de Informática, da Universidade Federal de Pernambuco – UFPE/CIn, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.
Orientador: Prof. Sílvio Romero de Lemos Meira, Dr.

RECIFE

2017

Catalogação na fonte
Bibliotecário Jefferson Luiz Alves Nazareno CRB 4-1758

F992f Furtado, Ana Paula Carvalho Cavalcanti.
FAST: um framework para automação de teste / Ana Paula Carvalho Cavalcanti Furtado. – 2017.
179 f.: fig., tab.

Orientador: Silvio Romero de Lemos Meira.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Teste de software. 2. Automação de teste. I. Meira, Silvio Romero de Lemos (Orientador). II. Título.

005.14 CDD (22. ed.) UFPE-MEI 2017-268

Ana Paula Carvalho Cavalcanti Furtado

FAST: Um Framework para Automação de Teste

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação

Aprovado em: 13/07/2017

_____

Orientador: Prof. Dr. Silvio Romero de Lemos Meira

BANCA EXAMINADORA

_____

Prof. Dr. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

_____

Prof. Dr. Hermano Perrelli de Moura
Centro de Informática / UFPE

_____

Prof. Dr. Ivaldir Hnório de Farias Junior
SOFTEX - Recife

_____

Prof. Dr. Jones Oliveira de Albuquerque
Departamento de Estatística e Informática / UFRPE

_____

Prof. Dr. Eduardo Henrique da Silva Aranha
Departamento de Informática e Matemática Aplicada / UFRN

Dedico este trabalho a Peu e Bia.

AGRADECIMENTOS

Gostaria de iniciar agradecendo ao meu orientador, professor Silvio Meira, por ter me dado a oportunidade e honra de ser sua aluna mais uma vez. À professora Andrea Zismam que abriu as portas para a vivência de pesquisa na Open University, e que me apoiou durante essa experiência de vida tão enriquecedora para mim e toda a minha família.

Agradeço, também, aos membros da banca, professores Alexandre Vasconcelos, Hermano Perrelli, Jones Albuquerque, Eduardo Aranha e Ivaldir Junior, por me proporcionarem contribuições tão valiosas para este trabalho de doutorado.

Agradeço aos meus familiares, que estiveram juntos e apoiaram essa grande jornada que é uma pesquisa de doutorado. Ao meu pai, pela eterna orientação e esperança por um mundo melhor, ao apoio da minha mãe, principalmente cuidando dos meus filhos nos momentos essenciais durante essa caminhada. Ao meu marido, Felipe, que tanto contribuiu e me apoiou nos momentos mais difíceis. Aos meus filhos, Pedro e Bianca, que sentiram na pele a minha ausência e que um dia entenderão o esforço e dificuldade vivida nesse período! Espero não ter deixado traumas! Ao meu sogro e sogra, Paulo e Margarida, por estarem sempre ao lado apoiando e incentivando.

Aos colegas de trabalho Cesar França, que me apoiou no planejamento e execução do meu estudo de caso, e a Gilberto Cysneiros, que me apresentou a professora Andrea, para que o doutorado sanduíche pudesse ser possível e sempre esteve ao nosso lado na Inglaterra, ajudando nos momentos difíceis e compartilhando os momentos de alegria com minha família.

Ao meu grupo de pesquisa, *Test Automation Group* – TAG, criado a partir das demandas do doutorado e que permitiram um acompanhamento e relacionamento mais próximo com meus alunos, não apenas nas horas de pesquisa, mas também no compartilhamento das angústias e desafios enfrentados nessa caminhada.

Ao apoio que tive nas 2 organizações que prontamente abriram as portas e confiaram na minha pesquisa para que os estudos de caso pudessem ser realizados. Certamente esse foi o maior desafio da minha pesquisa, ver a execução em ambiente real da minha proposta. Agradeço a todos os participantes pela confiança e disponibilidade, principalmente, de tempo para que o estudo pudesse ser realizado.

Um agradecimento especial ao SOFTEX RECIFE, pela parceria e trabalho em conjunto, cuja vivência prática muito contribuiu para construção dessa pesquisa. A Raphaell Aretakis, pela brilhante capacidade de interpretar as minhas demandas e transformá-las nas figuras desta tese.

Enfim, mas não menos importante, aos meus professores da academia de ginástica, Jessica, Paulinho, Alexandre e Rodolfo, que tanto me ajudaram a cuidar da minha saúde física e mental para que eu tivesse forças para finalização desta etapa.

“*Veni, Vidi, Vici*”

(Gaius Iulius Caesar)

RESUMO

Contexto: a qualidade de um sistema ou de um produto pode ser diretamente influenciada pela qualidade do processo utilizado para desenvolver e mantê-lo. Nesse cenário, os processos de teste imaturos ou *ad-hoc* não são considerados como ambiente propício para a introdução sistemática da automação de teste, que pode ser usada como uma forma de apoiar a melhoria da qualidade do software. **Objetivo:** para a realização desta pesquisa, foi necessário, incialmente, analisar os benefícios e limitações da implantação de automação de teste de software. Além disso, analisar os fatores de insucesso da implantação de automação de teste de software nas organizações. A partir de então, propor uma estratégia para introdução sistemática de práticas de automação de teste no contexto de projeto de desenvolvimento de software. **Método:** para a concretização desta pesquisa, foi realizada uma revisão bibliográfica exploratória, para buscar a fundamentação teórica, embasamento da pesquisa e análise de trabalhos relacionados. Além disso, entrevistas empíricas foram conduzidas para coletar informações práticas sobre como as estratégias de automação de teste são introduzidas e praticadas nas organizações; e coletar experiências práticas de profissionais especialistas em automação de teste no ambiente de trabalho. Após definição da proposta, 2 estudos de caso foram executados, com intuito de avaliar a proposta. **Resultados:** proposta de uma estratégia para introdução da automação de teste consolidada por meio *do Framework for Automating Software Testing* (FAST). A proposta consiste em framework teórico que contempla uma estrutura hierárquica para a implantação de automação de teste a partir de práticas que podem ser instanciadas de acordo com as necessidades específicas e distintas de cada contexto de projeto. A partir da proposta, o FAST foi implantado e analisado em 2 contextos distintos de estudo de caso, nos quais dados quantitativos e qualitativos foram coletados. **Conclusão:** baseado na pesquisa, se pôde observar que a ausência de processos sistemáticos é um dos fatores que dificulta a introdução da automação de teste. A proposta do FAST, analisada a partir do estudo de caso, pode ser considerada como uma alternativa satisfatória para a introdução e manutenção da automação de teste no escopo do projeto de desenvolvimento de software.

Palavras-chave: Teste de Software. Automação de Teste. Melhoria de Processo de Software.

ABSTRACT

Context: The quality of a software product can be directly influenced by the quality of its development process. Therefore, immature or ad-hoc test processes are means that are unsuited for introducing systematic test automation, and should not be used to support improving the quality of software. **Objective**: In order to conduct this research, the benefits and limitations of and gaps in automating software testing had to be assessed in order to identify the best practices and to propose a strategy for systematically introducing test automation into software development processes. **Method**: To conduct this research, an exploratory bibliographical survey was undertaken so as to underpin the search by theory and the recent literature. Additionally, empirical interviews were conducted so as to gather practical information about how test automation strategies are introduced and practised in development organizations and to collect practical experiences from automated testing specialists working in the software industry. After defining the proposal, two case studies were conducted so as to analyze the proposal in a real world environment. **Results**: A Framework for Automating Software Testing – FAST, is a theoretical framework consisting of a hierarchical structure to introduce test automation into practices that can instantiated in line with the specific and distinct needs of any project. Based on the proposal, FAST was introduced into two distinct case studies, from which rich qualitative and quantitative data were collected. **Conclusion**: The findings of this research showed that the absence of systematic processes is one of the factors that hinders the introduction of test automation. Based on the results of the case studies, FAST can be considered as a satisfactory alternative that lies within the scope of introducing and maintaining test automation in software development.

Keywords: Software testing. Test automation. Software process improvement.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

AET	Automação da Execução do Teste
AP	Área de processo
AQAM	Agile Quality Assurance Model
AS	Área de Suporte
AT	Área Técnica
ATG	Automated Test Generation
CMMI	Capability Maturity Model Integration
EAP	Estrutura Analítica do Projeto
FAST	Framework for Automating Software Testing
GDF	Gestão de Ferramentas
GQM	Goal Question Metric
GUI	General User Interface
KA	Key Area
KPA	Key Process Area
MMAST	Maturity Model for Automated Software Testing
MPT.BR	Melhoria do Processo de Teste Brasileiro
PDCA	Plan Do Check Act
PO	Product Owner
SW-CMM	Software Capability Maturity Model
SWEBOK	Software Engineering Body of Knowledge
STEP	Software Testing Enhancement Paradigm
TDD	Test Driven Development
TAIM	Test Automation Improvement Model
TIM	Test Improvement Model
TMM	Test Maturity Model
TMMI	Test maturity model integration
TPI	Test Process Improvement
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Este capítulo descreve o contexto deste trabalho de doutorado, assim como a motivação que levou a realizá-lo, problemas da pesquisa e principais objetivos.

1.1 Contexto

Na indústria de software, com a crescente demanda do mercado para que soluções tecnológicas resolvam as mais diversas necessidades existentes na sociedade moderna, a **engenharia de software** aparece como uma disciplina para apoiar o processo de desenvolvimento, entrega e manutenção dos produtos de software. Engenharia de software, conforme definido pela norma IEEE 610.12 (1990, p. 67), é "(1) A aplicação de uma abordagem sistemática, disciplinada, quantificável para o desenvolvimento, operação e manutenção de software; isto é, a aplicação da engenharia de software. (2) O estudo de abordagens como em (1)".

A engenharia de software, por sua vez, é uma disciplina que contempla diversas áreas de conhecimento (*knowledge área – KA)*, conforme descreve o *Software Engineering Body of Knowledge* (SWEBOK, 2007), em que a **qualidade de software** e **teste de software** são 2 das 10 que estão presentes no guia. A ideia do SWEBOK (2007) é estabelecer uma *baseline* de conhecimento sobre a engenharia de software e fornecer uma caracterização consensualmente validada sobre os limites da disciplina.

Nesse contexto, a **qualidade de software** pode ser entendida como a "capacidade de um produto de software de satisfazer necessidades explícitas e implícitas quando utilizado sob condições especificadas" (ISO/IEC/IEEE 24.765, 2010, p. 334). A qualidade de software é uma área de interesse associada ao contexto deste trabalho, cujo conceito evoluiu a partir dos estudos de Joseph Juran (1951), Armand Feigenbaum (1961), Phillip Crosby (1979), e Edwards Deming (1982), para serem adaptados e implantados, entre outras áreas, na engenharia de software.

Melhoria de processo, por sua vez, começou a ser abordada nos anos 30 a partir dos princípios estatísticos de controle da qualidade definidos por Shewhart (1931 apud CMMI/SEI, 2010). Nesse cenário, entende-se que "a qualidade de um sistema ou de um produto é altamente influenciada pela qualidade do processo

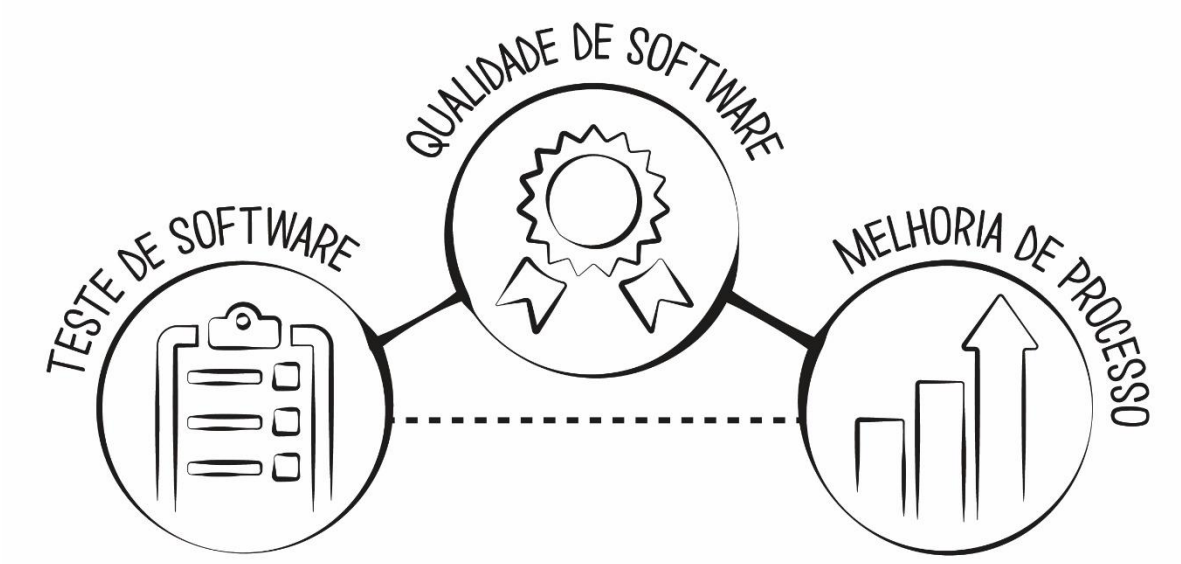
utilizado para desenvolver e manter” (CMMI/SEI, 2010, p. 5), em que a perspectiva de que a melhoria de processo é um fator necessário para o desenvolvimento de software com qualidade é apresentada para o mercado.

Neste sentido, o **teste de software** surgiu como outra vertente a ser desenvolvida quando se fala em qualidade de software, na qual o teste pode ser definido como “a dinâmica de verificação do comportamento de um programa a partir de um conjunto finito de casos de teste, adequadamente selecionados a partir de um domínio infinito de execuções” (SEWBOK, 2010, p. 5-1).

Nesse cenário, diversos **padrões** e abordagens foram definidos para contemplar o escopo da disciplina de teste de software, tais como a norma para requisitos de qualidade e teste ISO/IEC 12119 (1994), o padrão para documentação do teste de software e do sistema IEEE 829 (2008), conceitos e definições para o teste de software ISO/IEC/IEEE 29.119-1 (2013), o glossário para os termos de teste (ISTQB, 2012), para citar os mais relevantes para esse contexto.

A Figura (1) representa o contexto da qualidade de software relacionado com este trabalho, que contempla a melhoria de processo e teste de software, para se maximizar a qualidade do software desenvolvido para o mercado.

Figura 1 (1) – Perspectivas sobre qualidade de software



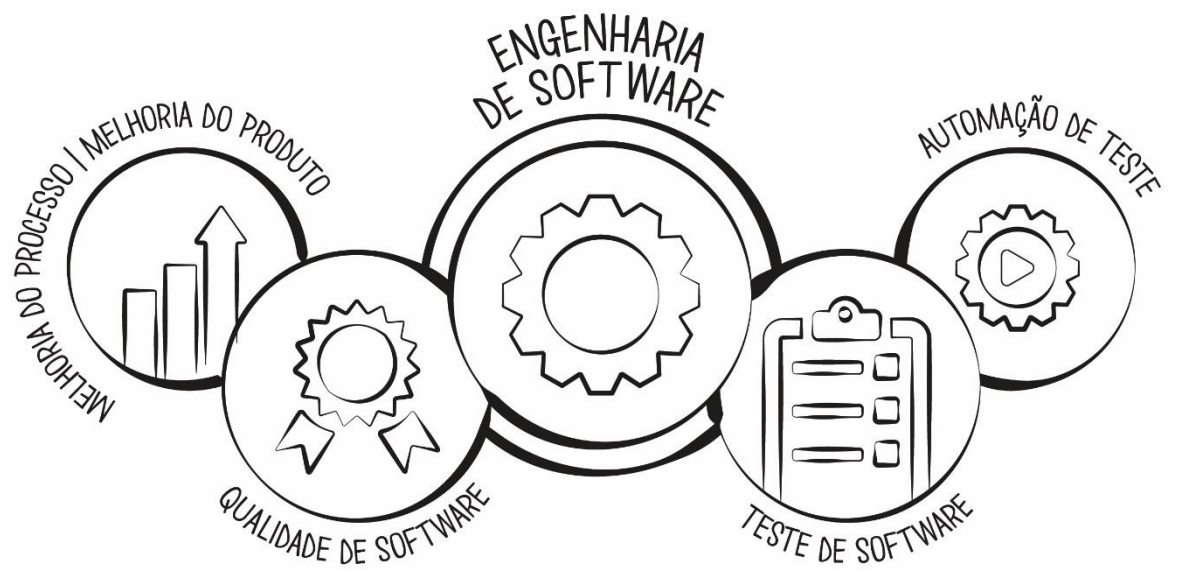
Fonte: Elaborada pela autora (2017)

No entanto, ainda sobre o contexto, a área de conhecimento de teste de software é bastante abrangente e este trabalho foca em automação do teste. Entende-se, por sua vez, que **automação de teste** é o uso de software para realização das

atividades de teste (ISTQB, 2012) e é considerada um importante tópico de interesse e vem largamente sendo estudada na literatura (BERNER et al., 2005), (RAMLER e WOLFMAIER, 2006), (KARHU et al., 2009), (RAFI et al., 2012), (WIKLUND et al., 2012) e (WIKLUND et al., 2014).

Neste trabalho, a perspectiva de qualidade é conduzida sobre a proposição de como a automação dos testes, a partir da melhoria de processo, pode auxiliar para que um software alcance a qualidade desejada. A Figura 2 (1) tem o propósito de representar a retórica sobre o contexto deste trabalho, em que engenharia de software é vista a partir de qualidade e teste de software. A qualidade, por sua vez, é compreendida por meio das práticas de melhoria de processo e de produto. Já no âmbito de testes de software, o contexto é focado em automação, conforme descrito e definido nesta seção.

Figura 2 (1) – Contexto do trabalho



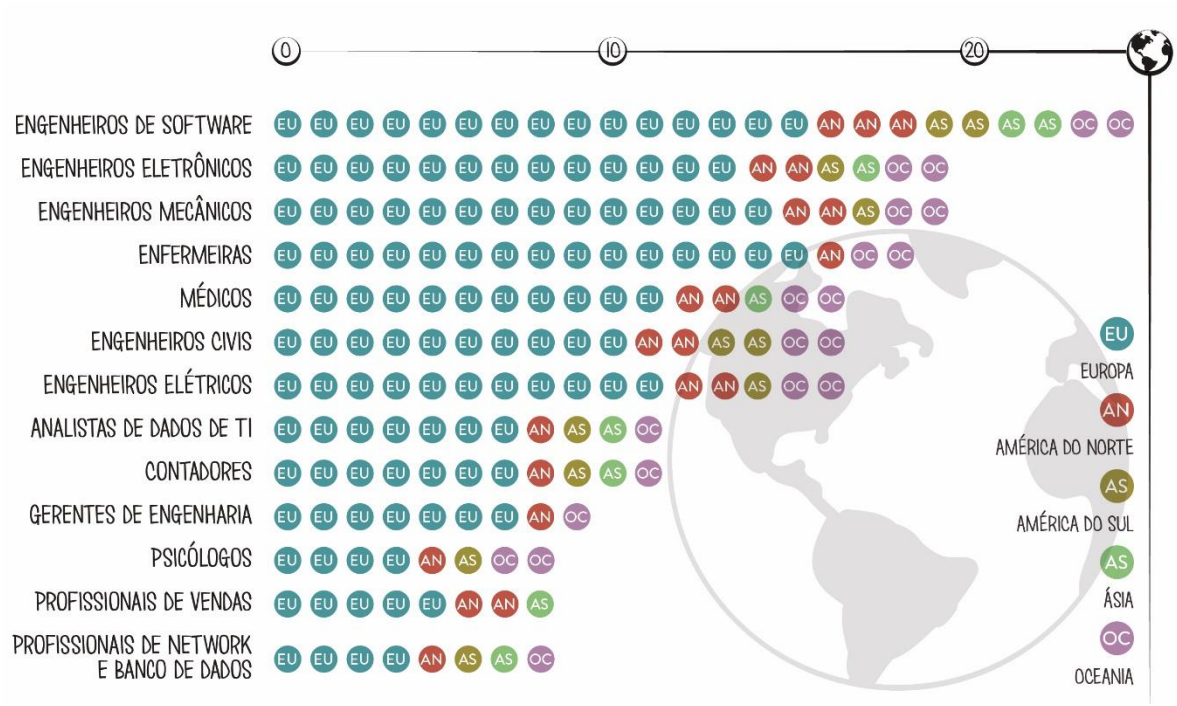
Fonte: Elaborada pela autora (2017)

1.2 Motivação

A carreira na área de computação está entre as que mais crescem em demanda na economia atual, na qual existe uma ampla gama de oportunidades no mercado de trabalho. Em um estudo realizado pela empresa de recrutamento Michael

Page[1], foi analisada a demanda por profissionais em 24 países de 5 continentes, conforme apresenta a Figura 3 (1). Neste cenário, a profissão mais demandada é a de engenheiro de software, superando a demanda em outras áreas de extrema importância, tais como engenheiros, enfermeiras e médicos.

Figura 3 (1) – Demanda das profissões



Fonte: Michael Page[2]

A partir dos dados apresentados nessa pesquisa, observa-se que a demanda por profissionais na área de engenharia de software é grande devido à crescente demanda por software, em que a qualidade com que os produtos são entregues para o mercado é um fator crucial para as empresas se manterem competitivas neste cenário.

Além da variável qualidade, busca-se também que o software seja desenvolvido ao menor custo possível. O próprio mercado impõe prazos mais

[1] Michael Page – Empresa localizada no Reino Unido, especializada em recrutar oportunidades permanentes, temporárias e contratuais, com atuação no mercado global. Disponível em http://www.michaelpage.co.uk/ acessado em março de 2017.
[2] Disponível em http://www.michaelpage.co.uk/minisite/most-in-demand-professions/ Acessado em 20 de março de 2017.

competitivos, exigindo maior agilidade e alta produtividade das equipes na utilização de seus processos (BOEHM, 2006).

Nesse contexto, teste de software pode ser compreendido como uma técnica para identificar os problemas antes que ele seja entregue para o mercado. É um efetivo instrumento para medir a sua qualidade. A prática de verificação de software pode trazer diversos benefícios, tais como (IEEE 1012, 2012, p. 4):

> (1) Facilitar a detecção e correção prévia das anomalias; (2) Apoiar os processos de ciclo de vida para assegurar a conformidade com a performance, cronograma e orçamento do projeto; (3) Fornecer uma avaliação inicial de desempenho; (4) Fornecer evidência objetiva de conformidade para apoiar um processo formal de certificação; (5) Melhorar os produtos a partir dos processos de aquisição, fornecimento, desenvolvimento e manutenção; e (6) Apoiar as atividades de melhoria de processo.

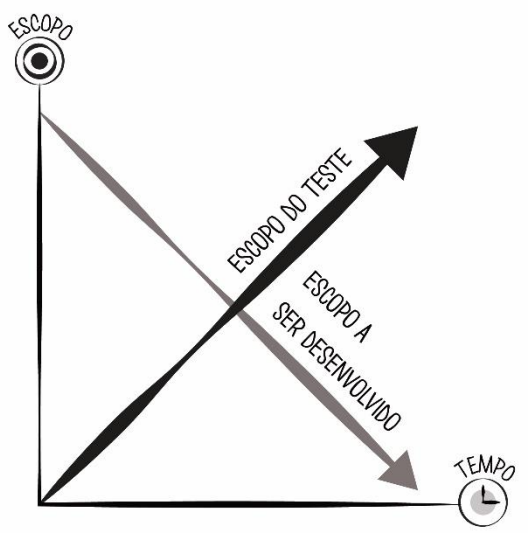
Neste cenário, o teste manual é mais custoso do que o automático, além do fato de que a automação aumenta a eficiência, especialmente para os testes de regressão (DUSTIN, 1999). Os benefícios da automação podem ser observados em longo prazo, cujo foco deve estar em aumentar a cobertura dos testes e não apenas em reduzir o custo (HAYES, 2004; DAMM e LUNDBERG, 2006; WISSINK e AMARO, 2006; WIKLUND et al., 2012). Estudos mostram que o teste contabiliza 50% ou mais do custo total do projeto (RAMLER e WOLFMAIR, 2006), pois enquanto deve ser custoso entregar um produto atrasado ao mercado, pode-se considerar catastrófica a entrega de um produto defeituoso (HAYES, 2004).

A automação do teste tem sido proposta como uma solução para reduzir os custos dos projetos (HAROLD, 2000) e está se tornando cada vez mais popular com a necessidade de melhorar a qualidade do software em meio à crescente complexidade dos sistemas (FEWSTER, 2001). Automação de teste pode ser usada para ganho de velocidade (RAFI et al., 2012), fazer com que os testes possam ser repetíveis (RAFI et al., 2012) e gerenciar mais testes em menor tempo (TAIPALE et al., 2011). Além disso, pode ser uma forma bastante eficiente de reduzir o esforço envolvido no desenvolvimento, eliminando ou minimizando tarefas repetitivas e reduzindo o risco de erros humanos (KARHU et al., 2009). Pode ser considerada uma forma efetiva de reduzir o esforço e minimizar a repetição de atividades no ciclo de vida de desenvolvimento de software (WIKLUND et al., 2014).

O custo da automação, por sua vez, é uma variável não trivial e difícil de ser obtida, pois inúmeros fatores podem influenciar no seu cálculo. Faz-se necessário analisar o retorno sobre o investimento realizado para a automação de teste. Além disso, a busca pela qualidade é limitada e deve estar integrada com a meta de redução do custo no projeto.

Nesse contexto, enquanto o escopo a ser desenvolvido em um projeto vai diminuindo ao longo do tempo, o escopo do teste aumenta, haja vista que as funcionalidades entregues vão se integrado ao que já foi produzido, gerando, assim, maior quantidade de requisitos a serem testados, conforme representado no gráfico da Figura 4 (1).

Figura 4 (1) – Escopo do teste e do software



Fonte: Elaborada pela autora (2017)

1.3 Problemática

O desenvolvimento de software é uma atividade complexa e composta por muitas variáveis, tais como escopo, custo, prazo, equipe, etc., que podem influenciar positiva ou negativamente o resultado final produzido. Observa-se, no entanto, que os problemas mais sérios não são causados por atitudes individuais, mas, sim, por fatores relacionados aos procedimentos organizacionais ou comportamento cultural, em que se exige um enfoque abrangente e de longo prazo no processo de software da organização (HUMPHREY, 1992).

No contexto da automação de teste, muitas tentativas falharam em alcançar os reais benefícios da automação de maneira duradoura (FEWSTER, 2001) e alguns problemas relacionados à adoção de automação podem ser listados (BERNER et al., 2005), tais como:

- **Escolha inadequada** ou ausência de alguns tipos de teste, o que leva à ineficiência na execução dos testes;
- **Expectativas incorretas** em relação aos benefícios de automação e o investimento a ser realizado;
- Ausência de diversificação da **estratégia de automação** a ser adotada;
- **Uso de ferramentas** de teste focadas apenas em execução de teste, onde outras potenciais áreas são esquecidas.

Na maioria dos casos, entre as razões para a falha na adoção de automação de teste está o fato de que o esforço para desenvolver e manter testes automatizados é subestimado pela falta de planejamento de projeto (RAMLER e WOLFMAIER, 2006). Além disso, poucos estudos apresentam relatórios empíricos sobre a prática de automação de teste em diferentes tipos de organizações (KARHU et al., 2009), no cenário em que estudos empíricos são cruciais para a pesquisa na área de teste de software com objetivo de comparar e melhorar as técnicas e práticas de teste (BRIAND, 2007).

Pode-se observar que existe uma **deficiência técnica** em muitas implementações de automação de execução de teste, causando problemas para o uso e manutenção dos sistemas de automação (WIKLUND et al., 2012). Entre as limitações existentes que impedem a implantação de automação de teste, pode-se citar (RAFI, et al., 2012):

- **Dificuldade de manutenção** da automação do teste como um todo, haja vista que os testes automatizados também precisam evoluir em paralelo com a evolução do software;
- Os processos e a infraestrutura para automação de teste precisam de tempo e, consequentemente, **maturidade para se adequarem** a determinado contexto; e,
- **Adoção inapropriada de estratégias de automação** leva ao seu uso inadequado, o que impede de alcançar os benefícios que a mesma pode fornecer.

Além disso, observa-se **que não existe uma definição clara a respeito de como se deve projetar, implementar e manter um sistema de automação de teste** de forma a maximizar os benefícios da automação em um determinado escopo, apesar de existir uma demanda para tal entre os desenvolvedores e testadores (WIKLUND, et al., 2012).

Ao mesmo tempo, a complexidade e o tamanho dos sistemas de testes automatizados e scripts de teste para um produto são comumente na ordem de, ou até maior, do que a complexidade e tamanho do produto a ser testado (WIKLUND et al., 2014).

Observa-se que ainda existe uma lacuna na pesquisa relacionada à automação de teste, dado a ausência de abordagens e diretrizes para auxiliar na concepção, implementação e manutenção de abordagens de automação de teste (FURTADO et al., 2016). Uma revisão multivocal da literatura[3] (GAROUSI et al., 2017) apresentou uma pesquisa sobre a avaliação da maturidade de teste e melhoria do processo de teste, constatando que os seguintes problemas são pertinentes:

- Os processos de teste de software continuam imaturos e conduzidos de forma ad-hoc;
- As práticas imaturas levam a ineficiência na detecção de defeitos;
- Cronogramas e custos constantemente excedem o que foi planejado; e,
- O teste não está sendo conduzido de forma eficiente.

Esse trabalho apresentou um estudo consistente que apoiou a justificativa da sua problemática, haja vista que o mesmo apresenta uma extensa análise da literatura existente. A abordagem apresentada neste trabalho foi um estudo da evolução das propostas para a melhoria de processo de teste existente, mediante visão histórica que contempla os trabalhos desde 1988 até 2016. Dentre os trabalhos apresentados, o autor faz referência tanto ao modelo de Melhoria do Processo de Teste Brasileiro (MPT.BR) (FURTADO et. al., 2012), cujo modelo e maturidade e vivência prática de sua implantação no mercado brasileiro foi relatado e analisado nesse artigo, como ao

[3] O termo *Multivocal Literature Review* (MLR) foi definido nos anos 90 como uma revisão sistemática que inclui a literatura formal e literatura cinza, onde a principal diferença entre uma revisão sistemática e uma MLR é as revisões sistemáticas usam apenas artigos acadêmicos que passaram por revisão por pares, enquanto a MLR considera a literatura cinza, tais como blogs, *White papers,* páginas na web (GAROUSI et. al., 2017, pp. 17).

modelo para Melhoria do Processo de Automação de Teste Brasileiro (MPTA.BR) (FURTADO et. al., 2014), cujo artigo apresenta uma versão inicial referente à pesquisa desta tese, por meio do qual sua proposta inicial era a proposição de um modelo de maturidade para automação.

Diante da problemática apresentada, a Figura 5 (1) consolida uma visão geral dos questionamentos associados a esta pesquisa, por meio dos tópicos identificados em cada esfera, os quais estão associados entre si, e, com base nesses problemas, identificados tanto na indústria de software como também nas referências estudadas, o problema deste estudo está associado com a seguinte pergunta:

Como automação de teste deve ser introduzida e mantida no processo de desenvolvimento software?

A pergunta de pesquisa deste trabalho pode ser classificada como uma pergunta de *design* (EASTERBROOK, et al., 2008), cujo objetivo é projetar melhores procedimentos e ferramentas para a realização de alguma atividade. Além disso, ela também pode ser classificada como uma pergunta de relacionamento, com interesse de analisar o relacionamento entre introdução de automação de teste software no desenvolvimento de software e o impacto para o custo e qualidade do projeto, haja vista que uma pesquisa em engenharia de software pode envolver um conjunto de perguntas de pesquisa, dado que os pesquisadores investigam problemas, como melhor resolvê-los e quais soluções funcionam melhor (EASTERBROOK, et al., 2008).

Figura 5 (1) – Visão geral dos problemas de pesquisa

Fonte: Elaborada pela autora (2017)

1.4 Objetivo

Com base no problema identificado na Seção 1.3, o **objetivo geral** desta pesquisa é propor uma estratégia para introdução sistemática de práticas de automação de teste no contexto de projeto de desenvolvimento de software.

Além disso, tem-se os seguintes **objetivos específicos**:

- Analisar os fatores de insucesso da implantação de automação de teste de software nas organizações; e
- Identificar boas práticas para introduzir automação de testes mencionadas na literatura.

1.5 Estrutura do trabalho

Além deste capítulo introdutório, este trabalho está organizado da seguinte forma:

- O Capítulo 2 descreve a metodologia de pesquisa utilizada nesta tese;
- O Capítulo 3 descreve a fundamentação teórica mediante análise do papel do teste na qualidade de software;

- O Capítulo 4 descreve e compara os trabalhos relacionados a esta pesquisa;
- O Capítulo 5 apresenta a proposta deste trabalho, por meio do FAST – Framework para Automação de Teste de Software;
- O Capítulo 6 apresenta o estudo de caso;
- O Capítulo 7 apresenta o grupo focal; e,
- O Capítulo 8 traz as conclusões e trabalhos futuros.

2 METODOLOGIA DE PESQUISA

Este capítulo apresenta o plano de pesquisa estabelecido para este trabalho, considerando o posicionamento filosófico, características da pesquisa empírica, assim como detalhamento das fases de pesquisa adotadas neste estudo.

2.1 Planejamento da pesquisa

O planejamento metodológico é essencial para a condução de pesquisas acadêmicas, cujo propósito é maximizar a apropriação dos possíveis resultados, por meio do ciclo contínuo de interações teóricas e práticas relacionadas com o tema. A **pesquisa empírica** explora, descreve, prevê e explica fenômenos naturais, sociais ou cognitivos a partir do uso de métodos científicos e experiência baseada em evidência (SJOBERG, DYBA e JORGENSEN, 2007).

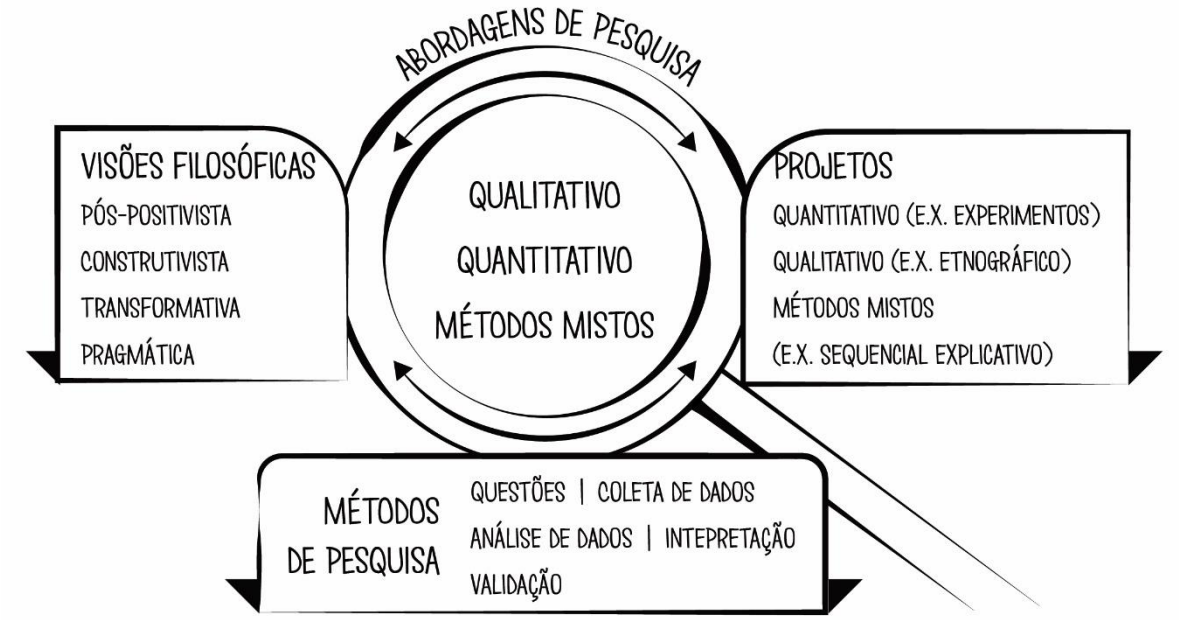
A **engenharia de software baseada em evidência** é um mecanismo que pode apoiar e melhorar o processo de tomada de decisões, cujo objetivo é integrar as melhores evidências a partir da pesquisa, das experiências práticas e valores humanos (DYBA, et al., 2005). É uma abordagem para se coletarem as melhores evidências decorrentes de experiência prática e valores humanos (KITCHENHAM; DYBA; JØRGENSEN, 2004).

No desenvolvimento de software, as atividades humanas interferem diretamente na forma com que os sistemas são construídos, e para compreender como os engenheiros de software constroem e mantêm sistemas de software, é necessário investigar não apenas as ferramentas e processos que são utilizados, mas também os processos cognitivos e sociais associados (EASTERBROOK, 2008). Faz-se necessário, portanto, o uso de método empírico, que consiste em reunir informações com base em observações e experiências sistemáticas, ao invés do uso de lógica dedutiva e matemática (SJOBERG et al., 2007), no qual o estudo empírico é um teste que compara o que acreditamos com o que observamos (PERRY et al., 2000).

Nesse cenário, a abordagem da pesquisa foi planejada com base nas propriedades do problema em questão, dadas às características do contexto e variáveis a serem consideradas para o planejamento da abordagem de pesquisa mais

apropriada para o contexto deste trabalho. Uma abordagem de pesquisa envolve pressupostos filosóficos, bem como diferentes métodos e procedimentos (CRESWELL, 2014, p. 5), ilustrados pela Figura 6 (2), que apresenta como estes elementos interagem e transformam a pesquisa em um processo prático.

Figura 6 (2) – Framework para pesquisa



Fonte: Resarch Design (CRESWELL, 2014, p. 5)

Observa-se que a abordagem de pesquisa está diretamente relacionada com o posicionamento filosófico adotado, que influencia diretamente na prática da pesquisa científica (SLIFE e WILLIAMS, 1995), cuja escolha apoia o planejamento de quais métodos devem ser seguidos para que os benefícios e desafios de cada método sejam claramente identificados e tratados ao longo da pesquisa.

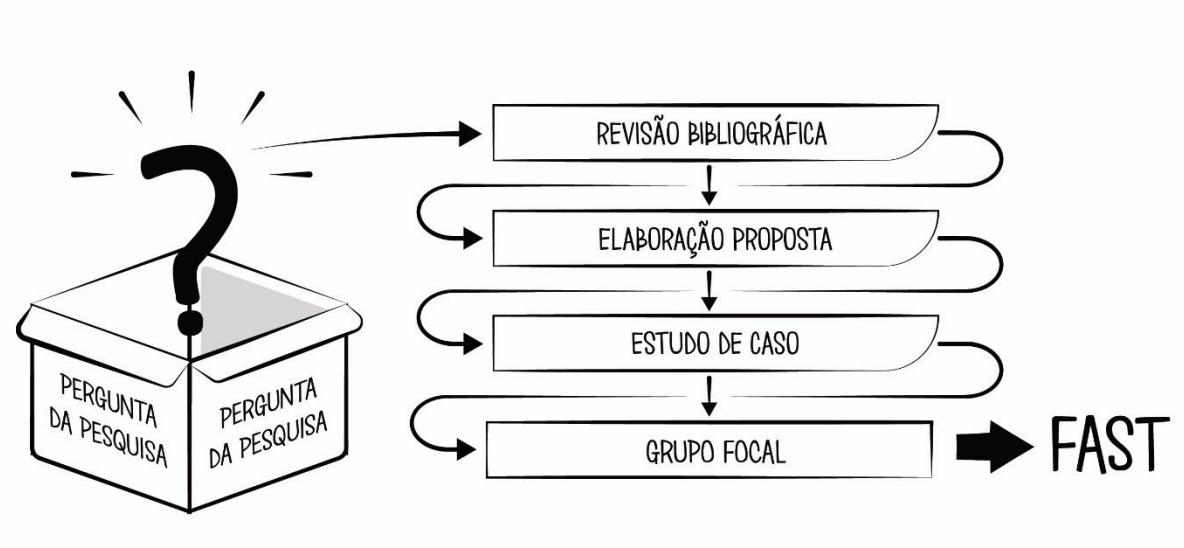
O posicionamento filosófico adotado nesse trabalho está centrado no **pensamento construtivista**, pois o conhecimento científico está associado e é construído no contexto em que o mesmo surgiu (EASTERBROOK, 2008). Construtivistas preferem métodos que **coletam dados qualitativos** sobre atividades humanas, nos quais, a partir daí, novas teorias podem surgir. O conhecimento e compreensão do mundo são construídos mediante experiência e reflexão da mesma, explorando e avaliando, continuamente refletindo sobre as experiências.

Nesse cenário, optou-se pela utilização de métodos qualitativos para avaliação da proposta deste trabalho, e por fim, com base nos pressupostos descritos e embasamento filosófico definido, o planejamento desta pesquisa será detalhado na seção seguinte.

2.2 Fases da pesquisa

O projeto de pesquisa para realizar este trabalho está dividido em 4 fases, conforme ilustra a Figura 7 (2), em que a primeira foi a realização de uma revisão bibliográfica exploratória, seguida da elaboração da proposta, na Fase 2. A Fase 3, com o propósito de avaliar a proposta, foi realizada por meio de 2 estudos de caso. Em seguida, na Fase 4, foi realizado um grupo focal.

Figura 7 (2) – Fases da metodologia de pesquisa



Fonte: Elaborada pela autora (2017)

Apesar de a estrutura metodológica estar demostrada a partir de fases sequenciais, as etapas de uma pesquisa, por diversas vezes, acontecem em paralelo, haja vista que a revisão bibliográfica é sempre revisitada e atualizada, desde o início até a finalização do trabalho. Diante dessa perspectiva geral, o detalhamento de cada fase do plano de pesquisa será descrito nas seguintes seções.

2.2.1 Revisão bibliográfica exploratória

A revisão bibliográfica é uma análise crítica, meticulosa e ampla das publicações correntes em determinada área do conhecimento. O principal objetivo deste instrumento neste trabalho é verificar textos relacionados ao assunto e conhecer a forma como o mesmo foi abordado e analisado em estudos anteriores, para poder compreender quais são as variáveis do problema em questão.

A revisão da literatura é uma importante etapa na pesquisa, pois apoia na compreensão sobre o tema e na avaliação dos problemas de pesquisa existentes, fornecendo ideias de como um pesquisador pode definir o escopo para determinada área de interesse (CRESSWELL, 2014, p. 25). Além disso, também faz a correlação entre a pesquisa e o diálogo corrente da literatura ampla, preenchendo os *gaps* e estendendo estudos anteriores (MARSHALL & ROSSMAN, 2011)

No escopo desta pesquisa, a revisão foi adotada de maneira *ad-hoc* mediante a busca nos engenhos de pesquisas disponíveis para a comunidade profissional e científica, tais como IEEExplorer, ACM, Scopus e Google Scholar. Além disso, foi realizada de modo a objetivamente analisar trabalhos anteriores, apresentando como os mesmos estão respondendo às perguntas de pesquisa deste trabalho e como os tópicos de interesse podem estar correlacionados com o tema.

Vale ressaltar que a revisão bibliográfica é essencial para o embasamento científico das questões centrais desta pesquisa, conceituando os tópicos relevantes para a construção da fundamentação teórica deste trabalho.

2.2.2 Elaboração da proposta

A elaboração da proposta da tese foi realizada com base nos conhecimentos adquiridos a partir da revisão bibliográfica, onde as referências para melhoria de processo de teste foram amplamente analisadas e comparadas entre si. No entanto, o detalhamento da estratégia para utilização das referências que formaram a base conceitual do trabalho será detalhado juntamente com a apresentação de sua proposta no Capítulo 5.

2.2.3 Estudo de caso

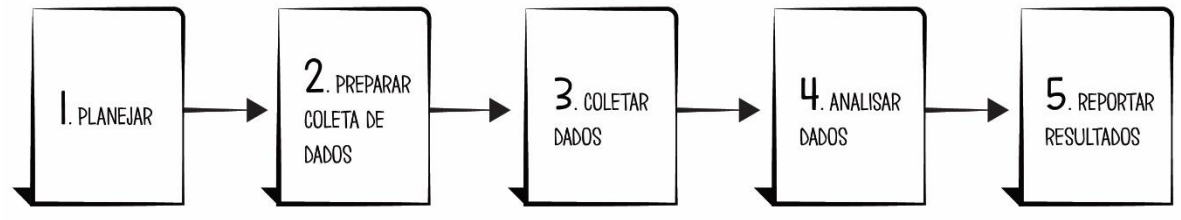
Um estudo de caso é um método de pesquisa empírico, definido como uma estratégia de pesquisa sobre a compreensão da dinâmica presente em determinado ambiente (EISENHARDT, 1989). Também pode ser descrito como um método empírico direcionado à análise de um fenômeno contemporâneo no contexto de vida real, especialmente quando os limites entre o fenômeno e o contexto não são claramente evidentes (BRATTHALL e JØRGENSEN, 2002).

Esse método foi considerado para o contexto desta pesquisa pelo fato de ela ser aplicável a contextos de pesquisas em engenharia de software, cujo objeto de estudo é um fenômeno contemporâneo difícil de ser estudado de maneira isolada e apresenta uma compreensão profunda sobre o que está sendo estudado (RUNESON e HÖST, 2008). Além disso, é aplicável para compreender, explicar ou demonstrar capacidades de uma nova técnica, método, ferramenta, processo, tecnologia ou estrutura organizacional (PERRY et al., 2004).

Vale ressaltar que o estudo de caso é predominantemente aplicável para analisar pesquisas que estão em estágios iniciais, nas quais a experiência dos atores e suas ações em um determinado contexto são extremamente valiosas no estudo do problema em questão (BENBASAT et al., 1987).

Nesse contexto, para a realização do estudo de caso, se adotou o processo apresentado por Runesson e Höst (HÖST e RUNESON, 2007; RUNESON e HÖST, 2008), conforme exporto na Figura 8 (2).

Figura 8 (2) – Passos do processo para realização do estudo de caso



Fonte: Elaborada pela autora (2017)

Apesar de a figura apresentar os passos de maneira sequencial, o processo se apresenta iterativo nos passos que contemplam a análise de dados e reportagem dos resultados. Os objetivos e descrição geral de cada passo do processo são apresentados a seguir:

1. **Planejar estudo de caso**: contempla a definição dos objetivos para que o estudo seja orientado a partir de um foco predefinido; elaboração das perguntas de pesquisa, declarando o que se precisa saber para que os objetivos sejam alcançados; definição e especificação do caso que será estudado; definição da teoria utilizada como base referencial durante o estudo; definição dos métodos de coleta de dados; e definição da origem dos dados.
2. **Preparar coleta de dados**: este passo considera a elaboração do protocolo para a coleta e análise dos dados, considerando informações sobre as fontes dos dados, método de coleta e métodos de medição dos dados.
3. **Coletar evidências**: este passo contempla a coleta dos dados que, preferencialmente, devem ser provenientes a partir de múltiplas fontes para evitar que a interpretação seja realizada apenas com uma única fonte de informação.
4. **Analisar dados**: este passo contempla a análise dos dados de maneira específica para cada tipo de informação coletada.
5. **Reportar resultados**: a elaboração do relatório deve comunicar os achados do estudo, devendo apresentar informações sobre o estudo de caso de maneira clara e com informações históricas, com dados precisos e objetivos, para que o leitor tenha certeza de que as conclusões são sensatas e isentas de viés.

Diante dos pressupostos, a aplicação do estudo de caso foi a forma utilizada para avaliar a proposta desta pesquisa, cujo detalhamento de sua execução será apresentado no Capítulo 6.

2.2.4 Grupo focal

O **grupo focal** é uma abordagem em que um grupo de pessoas se reúne para explorar conceitos e/ou problemas, constituindo uma pesquisa para a obtenção de percepções qualitativas e feedbacks de especialistas em determinados assuntos (CAPLAN, 1990). Este método conta com discussões em grupos, que são cuidadosamente planejadas, projetadas para consolidar a percepção de seus

membros, nas quais a interação do grupo é explicitamente usada para abordar um problema específico dentro de um prazo fixo, e de acordo com regras e procedimentos claramente enunciados (KONTIO et al., 2004).

Entre as propriedades desse método de pesquisa está o uso explícito da interação do grupo para produzir informações que seriam menos acessíveis sem a interação entre os membros (REED e PAYTON, 1997). Os participantes emitem opiniões pessoais, mas também podem interagir a partir da resposta de outros, e o entrevistador atua como moderador para que a entrevista permaneça focada em seus objetivos (FINCHER e PETRE, 2004).

O objetivo do grupo focal no contexto deste trabalho foi avaliar a proposta desta tese, com intuito de coletar sugestões de melhoria e, para isso, as seguintes etapas foram necessárias para sua execução, conforme descreve Kontio, et. al. (2004):

1. **Definição do problema de pesquisa**, cujo método é mais adequado para obtenção de feedback inicial sobre novos conceitos, geração de ideias, coleta ou priorização de potenciais problemas, obtenção de feedback sobre como modelos são apresentados ou documentados, dentre outros.
2. **Planejamento** do grupo focal, em que, a partir de um cronograma pré-definido, a quantidade de questões a serem abordadas deve ser limitada para que o tempo seja suficiente para que os participantes possam entender, discutir o problema, além de interagir entre si. Nesse contexto, a duração do grupo focal deve ser de até no máximo 3 horas.
3. **Seleção dos participantes**, cuja escolha dos participantes é fundamental para o sucesso do grupo focal, pois a eficácia deste método é muito sensível em relação à experiência e conhecimento dos participantes.
4. **Execução do grupo focal**, que pode ser iniciada por uma introdução onde os objetivos e as regras básicas da seção são explicados aos participantes. A partir de então, cada tópico é apresentado um após o outro, de modo que a discussão seja realizada com a condução de um moderador, onde o mesmo deve facilitar a discussão, mas não permitir que suas próprias opiniões influenciem a mesma. Durante a seção, o registro pode ser feito por meio de notas, áudio ou vídeo.

5. **Análise dos dados** e elaboração do relatório, que pode ser realizado a partir dos métodos existentes em sua análise qualitativa. Os dados quantitativos, se existirem, podem ser analisados por meio de estatísticas descritivas e outros métodos quantitativos padrão.

Com base nas etapas propostas pelo autor (KONTIO et al., 2004) e descritas acima, foi elaborada a Figura 9 (2) contemplando a visão geral das etapas do processo que foi aplicado para a realização do grupo focal.

Figura 9 (2) - Etapas do processo de realização do grupo focal

Fonte: Elaborado pela autora (2017)

A partir dessas premissas expostas relativas ao método de grupo focal, o mesmo foi realizado para avaliar a proposta desta pesquisa, cujo planejamento, execução e resultados serão apresentados e discutidos no Capítulo 7.

2.3 Considerações finais

Este capítulo apresentou a metodologia de pesquisa que foi planejada e utilizada para condução deste trabalho, de acordo com as características da pesquisa, com objetivo de maximizar os resultados obtidos em cada um dos métodos adotados. Tal metodologia foi planejada para contemplar a visão científica advinda de uma pesquisa de doutorado em conjunto com uma visão prática de como a proposta pode ser avaliada, a partir da introdução de formas qualitativa de avaliá-la. Nesse cenário, o uso dos métodos de estudo de caso e grupo focal foi essencialmente planejado para contemplar a visão prática da aplicabilidade da proposta no contexto real, assim como sua avaliação a partir de especialistas. Diante desse contexto, o capítulo seguinte irá apresentar uma visão geral da revisão bibliográfica, através da descrição do papel do teste na qualidade de software.

3 O PAPEL DO TESTE NA QUALIDADE DE SOFTWARE

Este capítulo descreve a fundamentação teórica utilizada como embasamento científico para a realização desta pesquisa, com objetivo de descrever os principais elementos conceituais e estado da arte associado ao tema central, que trata de como o teste pode apoiar a qualidade do software.

3.1 Qualidade de software

A qualidade de software e sua aplicabilidade vêm evoluindo ao longo do tempo, em que autores apresentaram diversas propostas distintas para abordar o assunto. Nesse contexto, **Joseph Juran** se destacou a partir da publicação de seu livro “Guia de Controle da Qualidade” (JURAN, 1951) e na definição da “Trilogia de Juran” (JURAN, 1989), uma abordagem de gestão da qualidade composta pelo (1) Planejamento da qualidade; (2) Controle da qualidade e (3) Melhoria da qualidade, conforme representa Figura 10 (3).

Figura 10 (3) – Trilogia de Juran

Fonte: (JURAN, 1989)

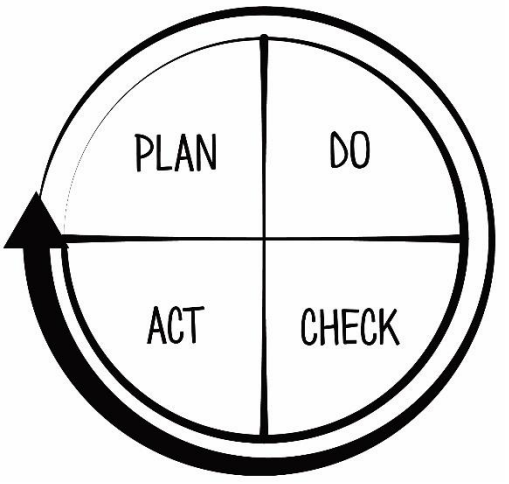
Armand V. Feigenbaum (1961) deu origem ao conceito de Controle Total da Qualidade, sistema com foco em integrar o desenvolvimento, manutenção e melhoria da qualidade, de modo a admitir a produção e serviço nos níveis mais econômicos que permitam a plena satisfação do cliente. Também trabalhou na definição dos

elementos de qualidade total para adotar o foco totalmente no cliente (interno e externo), desde a elaboração do projeto até a satisfação do cliente.

Philip Crosby (1979), autor do livro “Qualidade é de Graça”, destacou que é necessário fazer certo da primeira vez, afirmando que a implementação de melhoria da qualidade se paga por si mesma, por meio das economias da melhoria, aumento da satisfação do cliente e da melhoria da vantagem competitiva. Ele popularizou o termo "zero defeitos" para definir o objetivo de um programa de qualidade, como a eliminação de todos os defeitos e não a redução de defeitos a um nível de qualidade aceitável.

Edwards Deming avançou o estado da arte da qualidade e contribuiu com o contexto a partir do desenvolvimento da filosofia de gestão (DEMING, 1982), encapsulada em “14 pontos” que abordavam o tema. Além disso, foi o responsável pela popularização do ciclo PDCA (DEMING, 1993) (PDCA é uma sigla que do inglês para *Plan, Do, Check e Act* ou *Adjust*¸ correspondendo, em português a Planejar, Executar, Avaliar e Agir), método de gestão em quatro etapas utilizado para realizar a melhoria contínua de processos e produtos, também conhecida como a ciclo de Deming, conforme representa a Figura 11 (3).

Figura 11 (3) – Ciclo de Deming – ciclo do PDCA



Fonte: (DEMING, 1993)

O conceito e a prática da qualidade vêm evoluindo ao longo do tempo, e, de acordo com a ISO 9000 (2015), estão associados com as ações preventivas, com objetivo de eliminar a causa de potenciais não conformidades ou situações

indesejadas. É a partir dessa perspectiva que a qualidade se agrega ao teste de software, cujo intuito é antecipar as falhas do software percebidas pelos usuários, mediante implantação de melhores práticas de teste no contexto de desenvolvimento de software. Portanto, a seção 3.2 irá detalhar a visão de teste de software que foi utilizada com embasamento teórico para esta pesquisa.

3.2 Teste de software

Esta seção apresenta uma visão de teste de software relacionada ao contexto desse trabalho, e será descrita por meio de seus histórico, conceitos e processos.

3.2.1 Histórico do teste de software

A origem do teste de software coincide com a origem da programação, e sua evolução aconteceu com base nos padrões com que os programas eram testados, conforme apresentam Gelperin e Hetzel (1988) em seu artigo sobre o crescimento do teste de software. Nesse cenário, relata-se que o primeiro modelo de teste era ***debugging-oriented***, cujo foco era tirar os bugs do programa e o teste era uma das atividades para se fazer isso. Os marcos importantes dessa época estão retratados nas publicações de Allan Turing, em 1950 (TURING, 1950a), que falava em “prova de corretude”, seguido de outro artigo (TURING, 1950b), que abordava a questão sobre como é possível saber se um programa satisfaz seus requisitos.

O segundo período fala em ***demonstration-oriented***, cujo foco passa a ser em ter certeza de que o programa resolve um determinado problema, diferenciando a atividade de *debug* da atividade de teste (BAKER, 1957).

Logo em seguida, o teste passa para o período ***destruction-oriented***, em que teste era concebido como o processo destrutivo da execução de um programa com o objetivo de encontrar erros (MYERS, 1979). Nesse cenário, Dijkstra (1970) apresenta, em seu relatório técnico, notas sobre a programação estruturada e como a computação irá estabelecer os efeitos desejados, além de publicar uma de suas frases mais citadas que o teste é um processo que pode apenas demonstrar a presença de falhas, mas nunca sua ausência. Consequentemente, o teste não tem como

demonstrar a qualidade de um software, no entanto, é um instrumento para fornecer visibilidade sobre a qualidade do mesmo.

A partir dos anos 80, o teste passa a ser tratado como uma ferramenta não apenas para a detecção de erro, mas como instrumento de prevenção, caracterizado como uma atividade contínua ao longo do desenvolvimento de software (HETZEL, 1988). Foi o período caracterizado como ***evaluation-oriented****,* cujo foco passou de apenas detectar o erro para avaliar o produto durante todo o ciclo de vida do software. O modelo de ciclo de vida transfere o foco do teste do fim do projeto para seu começo. Esta mudança resulta na visão de que o teste não é uma fase sequencial que afeta somente a implementação do software, mas é uma atividade paralela que também afeta os requisitos e projeto do software (GELPERIN e HETZEL, 1988).

Nesse contexto, em 1983 foi lançado o padrão IEEE 829 (1983) para documentação do teste de software, cujo objetivo era descrever um conjunto básico de documentos de teste de software, para facilitar a comunicação e fornecer uma base referencial. Em 1986, surgiu o padrão IEEE 1012 (1986) para fornecer requisitos mínimos para o formato e conteúdo dos planos de verificação e validação no contexto de um projeto de teste. Logo em seguida, foi lançado o padrão IEEE 1008 para teste unitário (1987), cujo intuito era especificar uma abordagem padrão para teste unitário que pudesse ser utilizada como base para as práticas de engenharia de software.

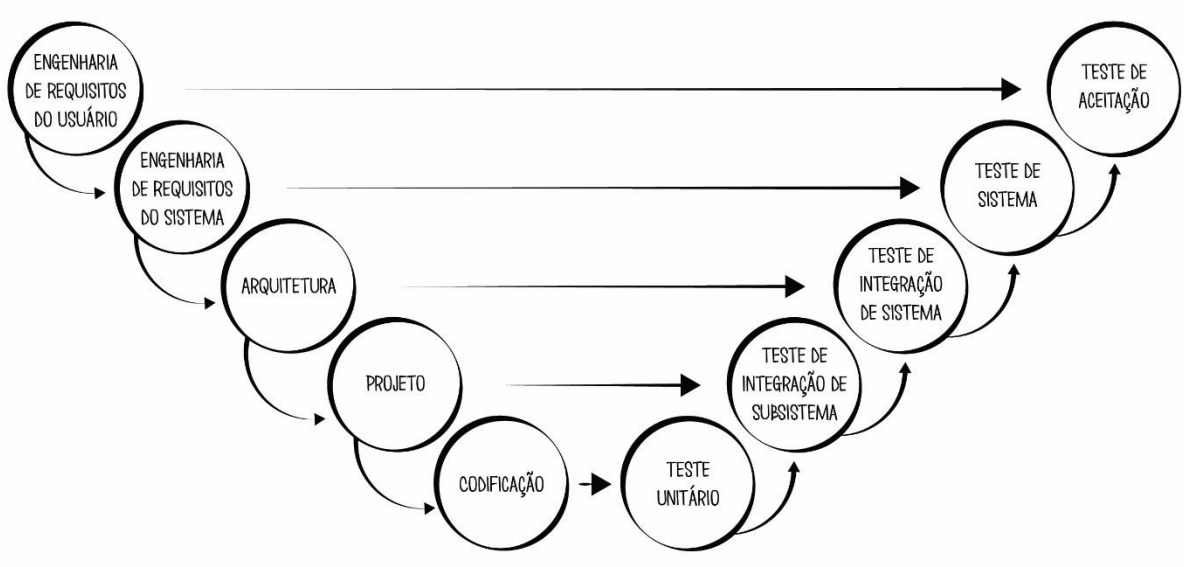
Os anos 90 evidenciaram as pesquisas em teste para software orientado a objetos, cujo paradigma introduziu novos riscos e dificuldades, aumentando, assim, a complexidade dos testes (BINDER, 2000). Ainda nessa década, o modelo V foi proposto por Forsberg e Mooz (1991), com o intuito de representar os aspectos técnicos por meio de um “V”, em que as atividades de verificação e validação são enfatizadas, de modo que o lado direito verifica ou valida as atividades do lado esquerdo, conforme apresenta a Figura 12 (3).

Já no fim dos anos 90, desenvolvimento baseado em componentes emergiu como uma abordagem para produção de software de maneira mais rápida e com menos recursos e, portanto, o teste baseado em componentes também foi listado como um dos desafios fundamentais na FOSE 2000[4].

---

[4] FOSE 2000 - Conference on The Future of Software Engineering, Limerick, Ireland — June 04 - 11, 2000. Proceedings available at: http://dl.acm.org/citation.cfm?id=336512&picked=prox

Figura 12 (3) – Visão geral do aspecto técnico do modelo V



Fonte: Adaptado de Forsberg e Mooz (1991)

Nos anos 2000, o *Test Driven Development* (TDD) (BECK, 2003) surgiu como uma das principais práticas do *Extreme Programming* (XP) (BECK, 2005), com o intuito de escrever um caso de teste automatizado antes de produzir o código que possa ser validado pelo caso de teste criado. Além disso, o estabelecimento de um processo apropriado para o teste foi considerado entre os temas de investigação fundamental em FOSE 2000 e, de fato, isto continua a ser uma pesquisa ativa hoje.

3.2.2 Conceitos relacionados ao teste de software

Barr et al. (2015) descrevem **teste de software** como uma atividade cujo propósito é estimular o sistema e observar sua resposta, na qual tanto o estímulo como a resposta possuem um valor, que podem coincidir quando o estímulo e resposta são verdadeiros. O estímulo, neste cenário, corresponde à atividade de testar o software que pode ser realizada a partir de diversas formas e técnicas.

Bertolino (2007) comenta sobre a importância do teste na engenharia de software, cujo alvo é observar a execução de um sistema para validar se ele se comporta conforme pretendido e identificar potenciais falhas. Além disso, o teste vem sendo largamente utilizado na indústria como **controle da qualidade**, fornecendo uma análise objetiva sobre o comportamento de um software.

“O trabalho de testar um software pode ser dividido entre teste automático e teste manual” (TAIPALE et al., 2011, p. 114), e a atividade de testar auxilia a garantia da qualidade por meio da coleta de informações sobre o software estudado (HARROLD, 2000). O teste de software inclui atividades que consistem em projetar os casos de teste, executar o software com os casos de teste e examinar os resultados produzidos pela execução (HARROLD, 2000).

Nesse contexto, a **automação de teste** aborda a automatização das atividades de teste de software, na qual uma das razões para o uso de teste automático ao invés do manual é que este consome mais recursos e a automação de teste aumenta a eficiência (DUSTIN et al., 1999). A decisão do que deve ser automatizado deve ser realizada o quanto antes no processo de desenvolvimento de software, com intuito de se minimizarem os problemas relacionados com esta atividade (PERSSON E YILMAZTURK, 2004).

De acordo com Bertolino (2007), a automação de teste é uma área de interesse bastante significante, cujo objetivo é melhorar o grau de automação, tanto por meio do desenvolvimento de melhores técnicas para a geração de testes automatizados, como pela automatização do processo de teste.

Ramler e Wolfmaier (2006) apresentam uma análise sobre a perspectiva econômica entre teste manual e automático, comentando que para a implantação da automação, somente os custos são avaliados e seus benefícios ignorados, em comparação aos testes manuais.

Além disso, automação de teste é uma ferramenta adequada para a melhoria da qualidade e redução de custos em longo prazo, pois os benefícios necessitam de tempo para serem observados (TAIPALE et al., 2011). Ainda neste contexto, a ISO/IEC/IEEE 29.119-1 (2013) comenta que é possível automatizar muitas das atividades descritas em seus processos, nos quais a automação requer o uso de ferramentas e está principalmente focada na execução de teste. No entanto, muitas atividades podem ser executadas por meio de ferramentas, tais como:

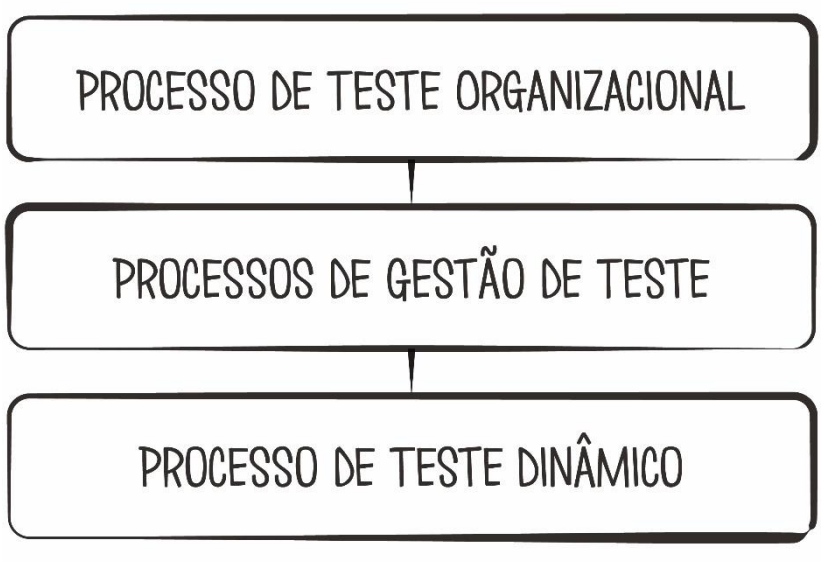
- Gestão de casos de teste;
- Monitoramento e controle do teste;
- Geração de dados de teste;
- Análise estatística;
- Geração de casos de teste;

- Monitoramento e controle do teste; e,
- Implementação e manutenção do ambiente de teste.

3.2.3 Processo de teste

Um importante aspecto para a atividade de testar o software é o **processo** utilizado para seu planejamento e implementação (HARROLD, 200). A ISO/IEC/IEEE 29.119-2 (2013) define um **modelo de processo** de teste genérico que pode ser utilizado para definir, gerenciar e implementar o processo de teste em qualquer organização ou projeto. O modelo sugere que as atividades de teste podem ser executadas, durante o ciclo de vida do software, em três grupos de processo, conforme apresentado na Figura 13 (3).

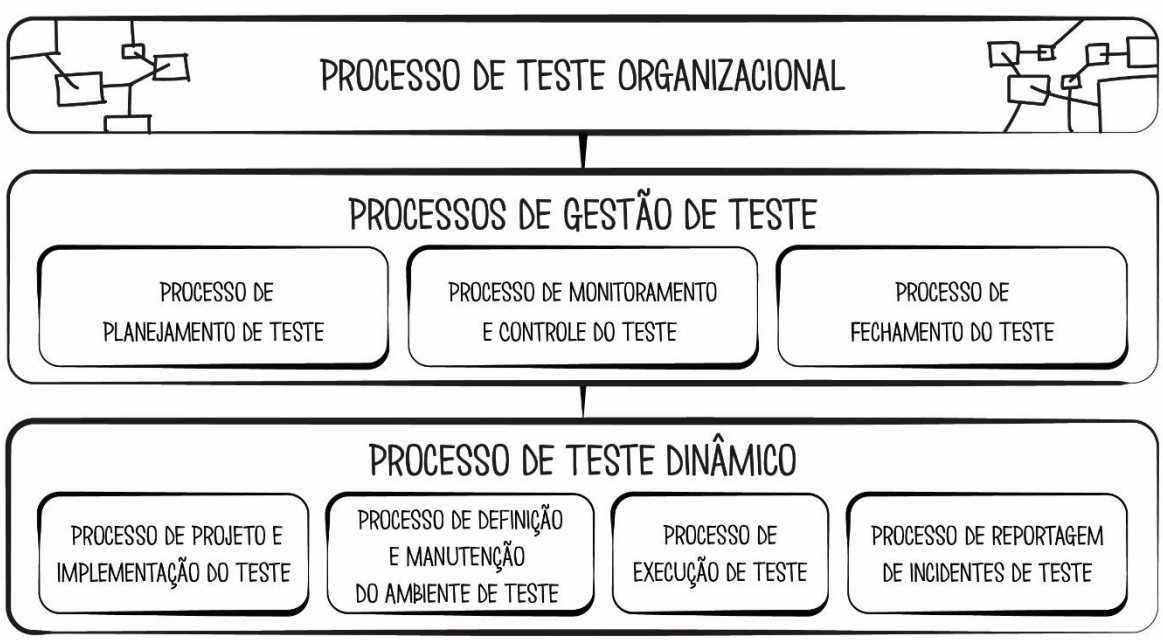
Figura 13 (3) – Multicamadas do processo de teste



Fonte: ISO/IEC/IEEE 29.119-2 (2013)

A partir das multicamadas apresentadas, o **processo de teste organizacional** tem como objetivo a definição de um processo para a criação e manutenção das especificações de teste organizacionais, como, por exemplo, as políticas, estratégias, processos e procedimentos. O **processo de gestão de teste**, por sua vez, é composto pelas ações de (1) planejamento; (2) monitoramento e controle; e (3) fechamento do teste; e o **processo de teste dinâmico** foca em (1) projeto e implementação do teste; (2) definição e manutenção do ambiente de teste; (3) execução do teste; e (4) reportagem do incidente de teste, conforme detalhado na Figura 14 (3).

Figura 14 (3) – Processos de teste distribuído por camadas



Fonte: ISO/IEC/IEEE 29.119-2 (2013)

3.3 Automação de teste de software

O teste de software deve eficientemente encontrar os defeitos no software como também deve ser eficiente para que sua execução possa ser realizada de forma rápida e com menor custo possível. Nesse cenário, a automação de teste pode diretamente apoiar a redução do esforço requerido para realização das atividades de teste ou para aumentar o escopo do teste a ser executado em determinado espaço de tempo (FEWSTER e GRAHAM, 1999).

Automação de teste realizada efetivamente apoia para que o projeto alcance a cobertura cumulativa (HAYES, 2004), que, conforme definido pelo autor, é o fato de que, ao longo do tempo, o conjunto de casos de testes automatizados permite que os mesmos sejam acumulados para que tanto os requisitos existentes como os novos possam ser testados ao longo do ciclo de vida do teste.

Fewster e Graham (1999) comentam, ainda, que teste e automação de teste são diferentes, uma vez que o primeiro se refere à habilidade de encontrar e executar os casos de teste mais adequados para determinado contexto, haja vista as inúmeras possibilidades de casos de teste existentes em um projeto. Automação de teste pode

também ser considerada uma habilidade, mas em um contexto diferente, pois a demanda, neste caso, se refere à escolha adequada e correta implementação dos testes a serem automatizados.

A automação de teste melhora a cobertura dos testes de regressão devido ao acúmulo de casos de teste automatizados ao longo do tempo, com objetivo de aprimorar a produtividade, a qualidade e a eficiência da sua execução (NAIK e TRIPATHY, 2008), haja vista que a automação permite que mesmo as mais pequenas mudanças de manutenção sejam totalmente testadas com um esforço mínimo requerido do time (FEWSTER e GRAHAM, 1999).

De acordo com a ISO/IEC/IEEE 29.119-1 (2013), é possível automatizar diversas atividades descritas no processo de Gestão de Teste e Teste Dinâmico descritos na ISO/IEC/IECC 29.119-2 (2013), e, apesar de estarem geralmente associados apenas à atividade de execução de teste, existem muitas tarefas e atividades de teste adicionais que podem ser suportadas por ferramentas baseadas em software.

3.4 Considerações finais

Este capítulo apresentou uma visão da perspectiva da qualidade no contexto de teste de software, na qual os conceitos fundamentais estabelecidos por alguns dos “gurus da qualidade” foram destacados para alinhar com a percepção de que o teste de software está diretamente associado às práticas de planejamento, controle e melhoria da qualidade, conforme apresentadas por Joseph Juran (1989) e reforçadas por Edward Deming (1993), a partir de seu PDCA. Não obstante, Armand V. Feigenbaum (1961) e Philip Crosby (1979) também destacaram a importância dos processos para a melhoria da qualidade do produto e garantia da satisfação do cliente.

A partir dessas premissas, a análise do histórico em relação ao teste de software e seus processos formaram um alicerce para o alinhamento das necessidades de qualidade e como os processos de teste apresentam os fundamentos para a definição de práticas para introdução da automação de teste. Nesse âmbito, a ISO 29.119-2 (2013) apresentou uma estrutura de organização por meio dos processos associados.

Para dar continuidade ao embasamento deste trabalho, o próximo capítulo irá apresentar os trabalhos correlacionados, detalhando os principais e aqueles que estão diretamente relacionados com esta pesquisa.

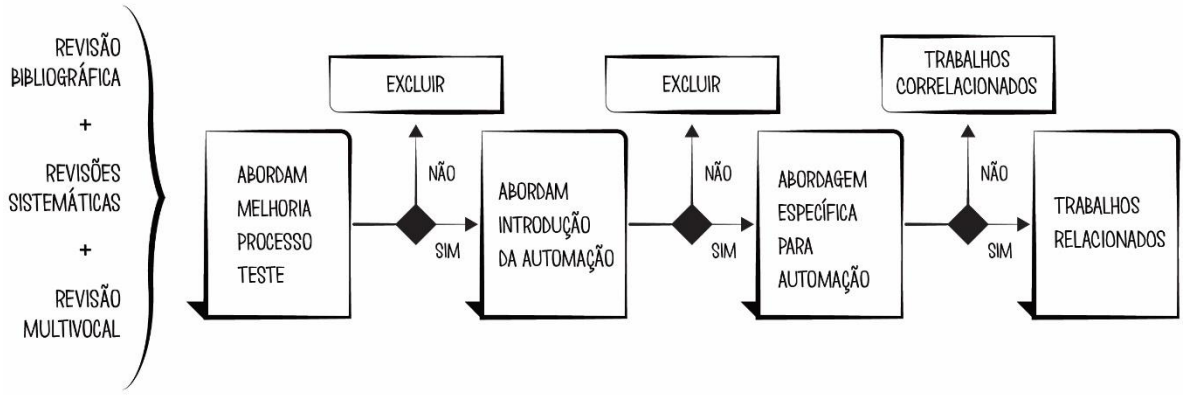
4 TRABALHOS RELACIONADOS

Este capítulo descreve os trabalhos relacionados com esta pesquisa, apresentando uma visão geral acerca dos trabalhos correlacionados e uma descrição mais ampla daqueles considerados diretamente relacionados com o contexto, mediante análise comparativa das propostas existentes e seu posicionamento com o problema desta pesquisa.

4.1 Visão geral

A análise dos trabalhos relacionados a esta pesquisa foi direcionada a partir da revisão bibliográfica, análise de revisões sistemáticas existentes (GARCIA, et al., 2014; BÖHMER e RINDERLE-MA, 2015; AFZAL et al., 2016) e da revisão multivocal (GAROUSI et. al., 2017), que abordam a melhoria do processo de teste. Os resultados alcançados foram organizados de modo que algumas abordagens foram classificadas como **trabalhos correlacionados**, sendo aqueles que apresentam sugestões para automação de teste por meio de propostas mais abrangentes de melhoria do processo de teste como um todo. Alguns outros foram considerados **trabalhos relacionados,** pois abordam especificamente processos de automação de teste, conforme representa a estratégia de seleção dos trabalhos na Figura 15 (4).

Figura 15 (4) – Estratégia de seleção dos trabalhos relacionados.



Fonte: Elaborada pela autora (2017)

A partir dessa pesquisa, foram selecionados os seguintes **trabalhos correlacionados**:

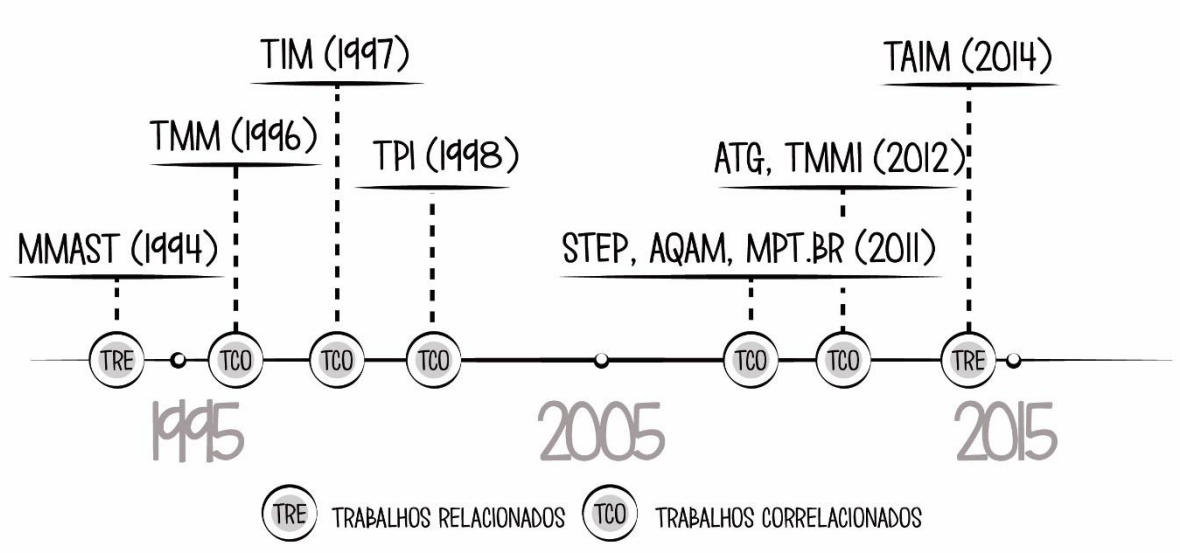
- Test Maturity Model – TMM (1996);
- Test Improvement Model – TIM (ERICSON et al., 1997);
- Test Process Improvement – TPI (KOOMEN e POL, 1998);
- Software Testing Enhancement Paradigm – STEP (CHELLADURAI, 2011);
- Agile Quality Assurance Model – AQAM (HONGYING e CHENG, 2011);
- Melhoria do Processo de Teste Brasileiro (MPT.BR) (2011) (FURTADO et al., 2012);
- Test Maturity Model Integration (TMMI) (2012); e
- Automated Test Generation – ATG (HEISKANEN et al., 2012).

Além desses, foram encontrados 2 **trabalhos relacionados**, tais como:

- Maturity Model for Automated Software Testing – MMAST (KRAUSE, 1994); e
- Test Automation Improvement Model – TAIM (ELDH et al., 2014).

Diante desse contexto, a Figura 16 (4) apresenta uma vista cronológica do surgimento desses trabalhos, que serão apresentados nas próximas Seções deste capítulo.

Figura 16 (4) – Linha do tempo com os trabalhos relacionados



Fonte: Elaborado pela autora.

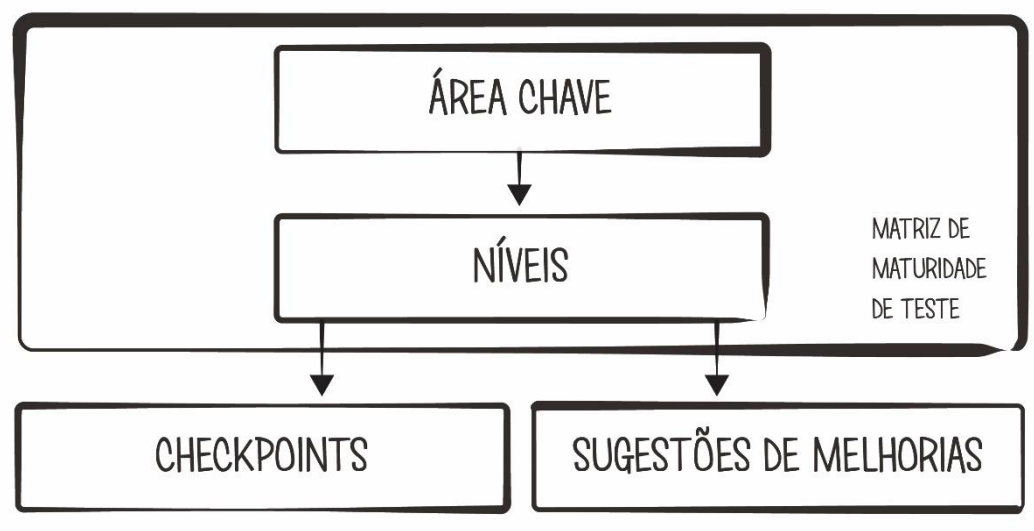
4.2 Trabalhos correlacionados

O **TMM** (1996) é um modelo composto por 5 níveis de maturidade: (1) Inicial; (2) Definição de Fases; (3) Integração; (4) Gerenciamento e Medição; e (5) Otimização, Prevenção de Defeitos e Controle de Qualidade. Em sua estrutura, cada nível contempla objetivos que são alcançados por meio de atividades, tarefas e responsabilidades. O nível 5 tem como um dos objetivos o uso de ferramentas para apoiar no planejamento e na sua execução dos testes, como também coletar e analisar os dados dos testes. No entanto, a abordagem apresentada é superficial e não contempla indicações de introdução e manutenção da automação de teste.

O **TIM** (1997) é um modelo de maturidade definido por meio de 2 componentes: um framework que contempla níveis e áreas-chave, e um procedimento de avaliação. Existem 4 níveis, tais como (1) Linha de base (*baseline*); (2) Custo efetivo; (3) Redução de riscos; e (4) Otimização; além de 5 áreas-chave: Organização, Planejamento e Acompanhamento, *Testware*, Casos de Teste e Revisão. Nesse contexto, *Testware* é a área-chave que contempla a definição da gerência de configuração dos ativos de teste e o uso de ferramentas para a realização de atividades repetíveis e não criativas. No entanto, a sistematização do uso de ferramentas para a realização das atividades de automação de teste não é detalhada.

O **TPI** (1997) é um modelo composto por 20 áreas-chave, cada uma com diferentes níveis de maturidade, que são estabelecidos a partir de uma matriz de maturidade, na qual cada nível é descrito por *checkpoints* (pontos de verificação) e sugestões de melhoria, conforme representa a Figura 17 (4). Uma das áreas-chave é denominada Automação de Teste, que descreve que a automação pode ser implementada de forma a abordar os seguintes pontos de verificação: (1) o uso de ferramentas, no nível de maturidade A; (2) gestão da automação, no nível B; e (3) otimização da automação de teste, no nível C, em que o A é o inicial e C é o mais alto. Apesar de abordar maturidade em automação de teste, observa-se negligência na avaliação do contexto da automação, na qual se trata apenas o uso de ferramentas e gestão de automação, haja vista que o escopo da automação é mais complexo do que apenas os processos apresentados.

Figura 17 (4) – Estrutura do TPI



Fonte: KOOMEN e POL (1998)

O **STEP** (2011) é um framework de melhoria de processo de teste orientado aos objetivos de negócio que está organizado por meio de: (1) Objetivos de negócio; (2) Objetivos de processo; e (3) 17 áreas de processo. A automação de teste está presente na área de processo Execução de Teste, a qual se sugere que a introdução da automação de teste seja por meio da implementação dos *scripts* de automação de teste. No entanto, não há sugestões de práticas que indiquem como os scripts de automação devam ser implementados e introduzidos no contexto do projeto.

AQAM (2011) é um modelo para a garantia da qualidade em ambiente ágil, para analisar e melhorar as capacidades desta garantia, composto por 20 áreas-chave de processo (KPAs), que contemplam guias, processos, melhores práticas, templates, customizações e níveis de maturidade. O objetivo da identificação de KPAs é avaliar objetivamente os pontos fortes e fracos das áreas importantes do processo de garantia de qualidade e, em seguida, desenvolver plano para melhoria do processo como um todo. Dentre as KPAs existentes, 9 delas estão diretamente relacionadas às atividades de teste, tais como: (1) Planejamento de teste; (2) Gestão do caso de teste; (3) Análise de defeito; (4) Reportagem de defeito; (5) Teste unitário; (6) Teste de performance; (7) Gestão do ambiente de teste; (8) Organização do teste; e (9) Automação de teste. Nesse contexto, apesar de apresentar KPAs relacionadas ao processo de teste, elas não abrangem todo os níveis de automação de teste para o contexto de um projeto, sendo restrito ao teste unitário e teste de performance. Portanto, no que se refere à introdução de automação de teste, a proposta é limitada

e, além disso, a documentação existente não apresenta descrição completa de suas práticas.

O **MPT.BR** (2011) aborda a melhoria do processo de teste, por meio de áreas de processo que contemplam as melhores práticas das atividades deste, ao longo do ciclo de vida de teste do produto. O modelo é composto por cinco níveis de maturidade, conforme apresentado na Figura 18 (4), e cada nível de maturidade é composto por área de processos.

No âmbito da automação do teste, o modelo apresenta a área de processo Automação da Execução do Teste (AET), cujo propósito é a definição e manutenção de uma estratégia para automatizar a execução do teste. Esta área de processo é composta pela seguinte lista de práticas específicas:

- AET1 – Definir objetivos do regime de automação;
- AET2 – Definir critérios para seleção de casos de teste para automação;
- AET3 – Definir um framework para automação de teste;
- AET4 – Gerenciar incidentes de teste automatizado;
- AET5 – Verificar aderência aos objetivos de automação; e,
- AET6 – Analisar retorno sobre investimento na automação.

Figura 18 (4) – Níveis de Maturidade do MPT.BR

Fonte: Elaborada pela autora (2017)

Embora as práticas específicas apresentem uma forma sistemática para a introdução de teste, as mesmas ainda são vagas no sentido de identificar o momento em que a automação deve ser realizada. Não há informação específica sobre a introdução da automação no processo de desenvolvimento de software, além de não informar quais níveis de teste podem ser automatizados. O formato escrito é genérico e abrangente, o qual pode se adequar a todo tipo de automação. No entanto, o mesmo não auxilia a escolha de onde a automação deve se iniciar e quais os benefícios que podem ser alcançados.

Existe ainda a área de processo Gestão de Ferramentas (GDF), cujo objetivo é gerenciar a identificação, análise, seleção e implantação de ferramentas na organização, composta pelas seguintes práticas específicas:

- GDF1 – Identificar necessidade de ferramentas;
- GDF2 – Selecionar ferramentas;
- GDF3 – Conduzir projeto piloto;
- GDF4 – Selecionar gurus de ferramentas;
- GDF5 – Definir estratégias de implantação de ferramentas; e,
- GDF6 – Implantar ferramentas.

Apesar de apresentar guias para o uso de ferramentas de teste e como as mesmas devem ser mantidas, a GDF não apresenta sugestões objetivas de como as ferramentas podem ser utilizadas para apoiar a automação dos testes, haja vista que a área de processo não aborda apenas ferramentas de automação. Portanto, considera-se que apesar de apresentar um guia para introdução de automação, o mesmo é genérico e não entra em detalhes sobre o processo de automação.

O **TMMI** (2012) é um modelo para a melhoria do processo de teste desenvolvido pela TMMi Foundation[5] como um guia e framework de referência e modelo complementar ao CMMI versão 1.2 (CHRISSIS et. al, 2007). O TMMI segue a versão estagiada do CMMI, e também se utiliza dos conceitos de níveis de maturidade para avaliação e melhoria do processo de teste. O foco específico do modelo é a melhoria do processo de teste, partindo desde o estágio caótico até o processo maduro e controlado mediante prevenção de defeitos.

---

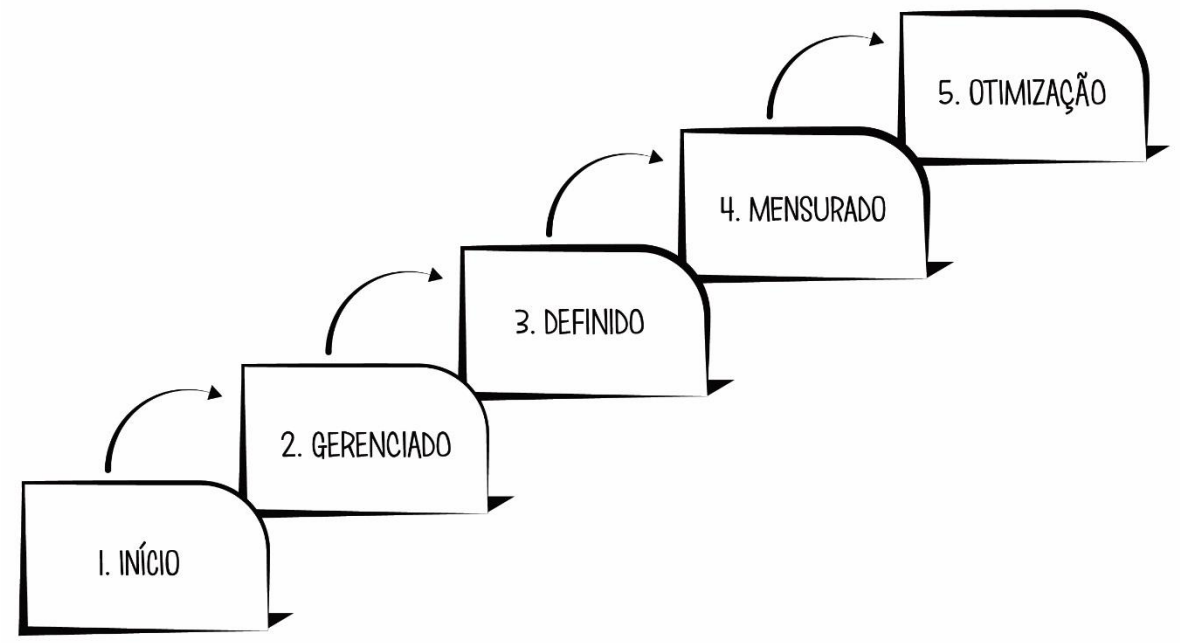
[5]TMMi Foundation é uma organização sem fins lucrativos dedicada a melhoria do processo e da prática de teste de software. http://www.tmmi.org/.

O TMMI é composto de 5 níveis de maturidade, tais como: nível 1 – Inicial; nível 2 – Gerenciado; nível 3 – Definido; nível 4 – Mensurado; e nível 5 – Otimização, conforme apresentado na Figura 19 (4). Cada nível de maturidade apresenta um conjunto de áreas de processo necessárias para se alcançar maturidade naquele nível, em que cada um é o embasamento para o nível seguinte.

Apesar de ser um modelo de maturidade especificamente para a área de teste e apresentar formas sistemáticas de introduzir a prática de teste de software no contexto do desenvolvimento de projetos, ele não apresenta área de processo dedicada especificamente a ferramentas e/ou automação de teste e não contempla sugestões sistemáticas para melhoria da automação de testes, conforme descrito no modelo:

> Observe que o TMMi não tem nenhuma área de processo dedicada a ferramentas de teste e/ou automação de teste. No contexto do TMMI, ferramentas de teste são tratadas como um recurso de apoio (práticas) e são, portanto, parte da área de processo em que eles fornecem suporte, por exemplo, a aplicação de uma ferramenta de projeto de teste é uma prática de teste de apoio dentro da área de processo de Projeto e Execução do Teste do TMMi nível 2 e aplicação de uma ferramenta de teste de desempenho é uma prática de apoio dentro da área de processo de Testes Não-funcionais no nível 3 do TMMi (TMMI, 2012, p. 10).

Figura 19 (4) – Níveis de maturidade do TMMI



Fonte: TMMI (2012)

O **ATG** é um processo que foi desenvolvido para complementar o TPI (KOOMEN e POL, 1998), cujo objetivo é a geração de testes automatizados a partir de modelos. Nesse contexto, a criação do ATG se deu a partir da introdução de 4 áreas de processo e modificações em algumas áreas existentes no TPI. Os modelos, nesse cenário, precisam ser computáveis, processados pelo computador, a partir, por exemplo, dos requisitos, projeto, código fonte, objetos de teste, etc. Caso o modelo não possa ser computável, ele deve ser convertido o quanto antes no processo de desenvolvimento de software, preferencialmente em paralelo com o projeto do software. O ATG é limitado no escopo de sua proposta, pois além de focar apenas na geração de testes baseada em modelos, o mesmo não explica como os modelos devem ser gerados para os diferentes níveis de automação de teste. Além disso, na indústria, durante a adoção de técnicas do ATG, encontraram-se algumas limitações, devido, em parte, à dificuldade de criar e manter os modelos de teste, o que pode ainda mais dificultar a adoção da abordagem.

4.3 Trabalhos relacionados

O *Maturity Model for Automated Software Testing* (**MMAST**) (KRAUSE, 1994) é um modelo que foi desenvolvido para fabricantes de equipamentos médicos computadorizados e seu propósito é definir o nível apropriado de automação em que um fabricante de equipamentos se enquadra. Ele é composto por 4 níveis de maturidade, tais como:

- Nível 1 – Automação acidental: caracterizado pelo processo ad-hoc, individualista e acidental de realizar as atividades, no qual não há documentação das informações importantes que estão restritas a peças-chave da organização. A automação de teste é realizada de maneira oportuna e não está embasada em processo e/ou planejamento.
- Nível 2 – Iniciando automação: está associado ao do uso de ferramentas de captura e repetição que reproduzem as respostas do sistema em teste. Inicia-se a documentação por meio do registro dos requisitos do software e do teste, cuja escrita fornece a base para implementação do nível 3.
- Nível 3 – Automação intencional: o foco é a realização da automação definida e planejada para o contexto de um projeto, a partir da derivação

dos requisitos e scripts do teste automático do documento de requisitos do sistema e parte do pressuposto que a equipe de teste faz parte da equipe do projeto. O modelo indica que o nível 3 é o adequado para a manufatura dos equipamentos médicos.

- Nível 4 – Automação avançada: é apresentada como uma versão aperfeiçoada do nível 3 com a inclusão da prática de gestão de defeitos pós-entrega, na qual os mesmos são capturados e enviados diretamente para os processos de correção, criação e regressão dos testes.

O modelo também apresenta um *checklist* para apoiar e identificar em qual nível o processo de teste deve ser automatizado, a partir das seguintes questões:

- Quão grandes são seus projetos de software?
- Qual a complexidade de seu produto?
- Que risco financeiro o seu produto representa para a empresa?
- Qual o risco que seu produto representa para o paciente e para o operador?

A partir do *checklist*, o nível de automação é recomendado. No entanto, apesar de ser um modelo de maturidade, o mesmo não apresenta áreas-chave ou áreas de processo e sua descrição é abstrata e não contempla aspectos em como a automação de teste pode ser introduzida de fato.

O *Test Automation Improvement Model* (**TAIM**) (ELDH et al., 2014) é um modelo para melhoria de processos baseado em medições para determinar se uma atividade é melhor do que a outra, com foco no cálculo do retorno do investimento. Além disso, ele está embasado na visão de que o processo de teste pode ser completa ou parcialmente automatizado, e fornece uma visão de como é possível a automação total do processo de teste, mediante definição de 10 áreas-chave (do inglês *Key Area* – KA):

1. Gestão de teste;
2. Requisitos de teste;
3. Especificação de teste;
4. Implementação do teste;
5. Automação do processo de teste;
6. Execução do teste;
7. Vereditos do teste;

8. Ambiente de teste;
9. Ferramentas de teste; e
10. Gestão de falhas e defeitos.

Além disso, o modelo contempla 1 área genérica (do inglês *Generic Area* – GA), que consiste em rastreabilidade, definição de medidas, análise, testware[6], padrões e qualidade da automação.

No entanto, “TAIM pode ser visto como um trabalho em progresso e como um desafio de pesquisa” (ELDH et al., 2014, p. 341), pois ainda se mostra de forma muito genérica e a evolução das KAs é apresentada como um trabalho futuro. Além disso, o modelo está baseado em medições, mas não as descreve e nem apresenta como as mesmas devem ser usadas e qual o relacionamento das mesmas com as KAs e com a GA.

Apesar de apresentar as KAs como forma de introduzir automação de teste, não está claro como as mesmas podem ser introduzidas de maneira sistemática no processo de desenvolvimento de software, haja vista que as mesmas são apresentadas de maneiras isoladas como um guia de melhores práticas. No entanto, o modelo falha em prover informações sobre como o mesmo pode ser adaptado para o contexto de desenvolvimento de software.

4.4 Considerações finais

Este capítulo apresentou os trabalhos relacionados a esta pesquisa, com intuito de descrever as abordagens de melhoria de processo de teste, como a introdução da automação é tratada nas propostas e quais as limitações existentes em relação aos problemas de pesquisa deste trabalho. As propostas apresentadas foram classificadas em trabalhos relacionados e correlacionados de acordo com o seu escopo.

A partir das informações consolidadas, foram definidos alguns critérios para analisar comparativamente os trabalhos, conforme descritos no Quadro 1 (4) e uma análise comparativa dos trabalhos foi sintetizada no Quadro 2 (4).

---

[6]Testware é definido, pelos autores, como o subconjunto do software num sistema que tem como alvo a automatização dos testes. Testware também descreve todos os artefatos utilizados para realizar um teste em todas as suas fases.

Quadro 1 (4) – Critérios para análise dos trabalhos relacionados

Critério	**Descrição do critério**
Abordagem	Indicar se a proposta é baseada em modelo de maturidade, processo, framework ou outro.
Domínio de aplicação	Especificar o domínio para o qual a proposta foi elaborada.
Escopo	Descrever o escopo da proposta, se evidencia um trabalho correlacionado, que foca em apresentar a abordagem de automação a partir de um processo de teste genérico, ou se é trabalho relacionado que apresenta, diretamente, proposta para introdução da automação.
Estratégia para introdução da automação	Descrever a proposta de introdução sistemática de automação.
Limitações	Identificar as restrições do trabalho comparado perante as perguntas desta pesquisa.

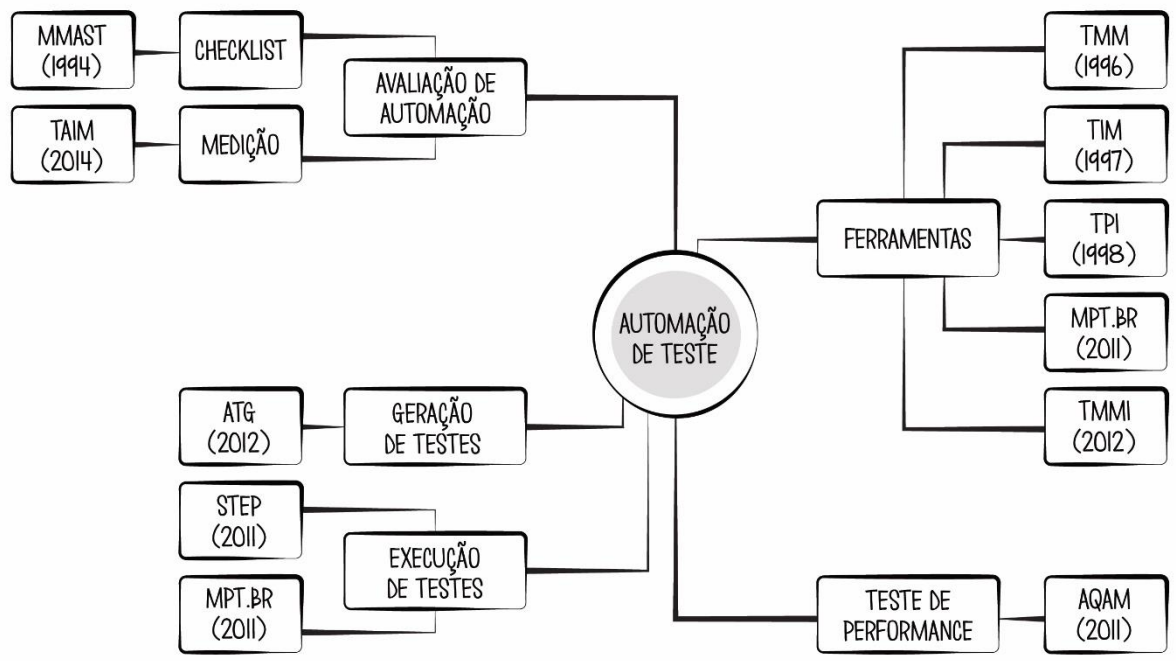
Fonte: Elaborado pela autora (2017)

Pode-se observar que a melhoria do processo de teste é uma área de pesquisa que apresentou diversos estudos/evolução ao longo do tempo. Apesar de a literatura apresentar diversas abordagens para a melhoria de processo de teste, nota-se que a automação de teste ainda é limitada em suas propostas.

As abordagens propostas são limitadas na descrição de como a introdução de automação de teste pode ser realizada de forma sistemática, e a principal barreira observada está no fato de que a automação é negligenciada no que tange ao seu escopo, sendo vista de forma atômica e não apresenta detalhes suficientemente completos para que a mesma seja introduzida e mantida no processo de desenvolvimento de software. Algumas propostas abordam a automação apenas no contexto do **uso de ferramentas** para apoiar o teste de software, tais como o TMM (1996), TIM (1997), TPI (1998), MPT.BR (2011), TMMi (2012). Em outras, a perspectiva apresentada tem o foco apenas na **execução de teste**, tais como STEP (2011) e MPT.BR (2011), como também somente na **geração de testes**, conforme proposta do ATG (2012), ou somente no **teste de performance**, como a proposta AQAM (2012). Nas propostas do MMAST (1994) e do TAIM (2014), a introdução da automação é sugerida a partir da **avaliação** para realizar um diagnóstico sobre os processos de automação de teste, cujas descrições existentes são limitadas por não tratarem de forma completa as suas formas de introdução de automação nos projetos de desenvolvimento de software.

A partir dessa análise, a Figura 20 (4) foi elaborada com intuito de sintetizar a visão das limitações existentes dos trabalhos relacionados diante do contexto dessa pesquisa. O entendimento das limitações existentes foi essencial para a elaboração da proposta, a qual será apresentada no próximo capítulo.

Figura 20 (4) – Abordagem de automação proposta pelos trabalhos relacionados



Fonte: Elaborada pela autora (2017)

Quadro 2 (4) – Análise comparativa dos trabalhos relacionados (continua)

Trabalhos	Abordagem	Domínio/Escopo	Estratégia para introdução da automação	Limitações
TMM (1996)	Modelo de maturidade	Genérico / Processo de Teste	Através do objetivo do nível 5 de maturidade, pelo uso de ferramentas para coletar e analisar os dados dos testes, como também apoiar no planejamento e na sua execução.	Não apresenta visão prática de como as ferramentas podem ser utilizadas para introduzir automação. Automação é abordada apenas através do uso de **ferramentas**.
TIM (1997)	Modelo de maturidade	Genérico / Processo de Teste	Através da área chave *testware*, que sugere o uso de gerência de configuração para os artefatos de teste além do uso de ferramentas para realização de atividades repetíveis e não criativas.	Automação é aplicada em atividades repetíveis, e não há guias para sistematizar a introdução e uso de **ferramentas** que possam apoiar a realização dessas atividades.
TPI (1998)	Modelo de maturidade	Genérico / Processo de Teste	Através da área chave automação de teste que contempla o uso de ferramentas, gestão e otimização da automação.	Abrangência do contexto da automação pelo modelo de maturidade, pois é tratada apenas a partir do uso de **ferramentas**, gestão e otimização da automação. Não apresenta guias de como as ferramentas e automação podem ser introduzidas no contexto do desenvolvimento de software.
STEP (2011)	Framework	Genérico / Orientado aos objetivos de negócio	Através da área de processo Execução de teste, onde sugere-se que a introdução da automação de teste seja através da implementação dos scripts.	Não há sugestões de práticas que indicam como os scripts de automação devem ser implementados e introduzidos no contexto do projeto. Além disso, a abordagem da automação é restrita a **execução de testes**.
AQAM (2011)	Modelo de maturidade	Genérico / Garantia da qualidade	Através de 9 KPA's relacionadas às atividades de teste, dentre elas uma está direcionada a automação de teste e outra a testes de performance.	O escopo da automação de teste está limitado a **teste de performance** e a uma KPA que aborda automação de teste de maneira genérica.
MPT.BR (2011)	Modelo de maturidade	Genérico / Processo de Teste	Através da área de processo AET, que aborda objetivos, critérios, framework e gestão de incidentes decorrentes da automação, e da área de processo GDF, que trata do uso de ferramentas para apoio às atividades de teste de software.	Não aborda a introdução da automação através de níveis de automação, e seu guia apresenta sugestões de melhores práticas com foco apenas nos processos de **automação da execução e através do uso ferramentas**.
TMMI (2012)	Modelo de maturidade	Genérico / Processo de Teste	Indica que as ferramentas de teste podem fornecer suporte aos processos e que podem apoiar no projeto e execução de testes com **apoio das ferramentas**.	Não apresenta sistemática para introdução e manutenção da automação do teste.

Quadro 2 (4) – Análise comparativa dos trabalhos relacionados (continuação)

Trabalhos	Abordagem	Domínio/Escopo	Estratégia para introdução da automação	Limitações
ATG (2012)	Modelo de maturidade	Genérico / Processo de Teste	Áreas de processo com foco na **geração de testes** automatizados a partir de modelos computáveis que possam ser processados pelo computador.	Foco na automação exclusivamente **a partir de modelos** e não há indicações de como os modelos podem ser gerados no escopo dos níveis de automação de teste.
MMAST (1994)	Modelo de maturidade	Equipamentos médicos / Processo de automação de teste	Apresenta um **checklist** que permite avaliar em que nível de automação um fabricante de equipamentos médicos se enquadra.	Modelo específico para empresas que desenvolvem equipamentos médicos, cuja descrição é abstrata e não contempla aspectos em como a automação de teste pode ser introduzida de fato.
TAIM (2014)	Modelo de maturidade	Genérico / Processo de automação de teste	Modelo de maturidade baseado em **medições** que apresenta uma visão de como é possível a automação total do processo de teste, mediante definição de 10 áreas-chave.	Apesar do modelo está baseado em medições, o mesmo não as descreve e nem apresenta como as mesmas devem ser usadas para apoiar a introdução da automação de teste.

Fonte: Elaborado pela autora (2017)

5 FAST

Este capítulo tem como objetivo apresentar a proposta deste trabalho contemplada por meio do Framework para Automação de Teste, denominado ***Framework for Automating Software Testing* – FAST**, cuja estrutura será apresentada considerando seus elementos conceituais, composição de áreas e descrição das áreas de processo.

5.1 Visão geral

De acordo com o dicionário de engenharia de software ISO/IEC/IEEE 24.765 (2010), um **framework** pode ser definido como um modelo que se refina ou especializa para fornecer partes de suas funcionalidades em determinado contexto. Esse conceito foi adaptado para o FAST com o **propósito** de estabelecer um conjunto de práticas para que a introdução da automação de teste seja consistente, estruturada e sistemática para o projeto que o implementa.

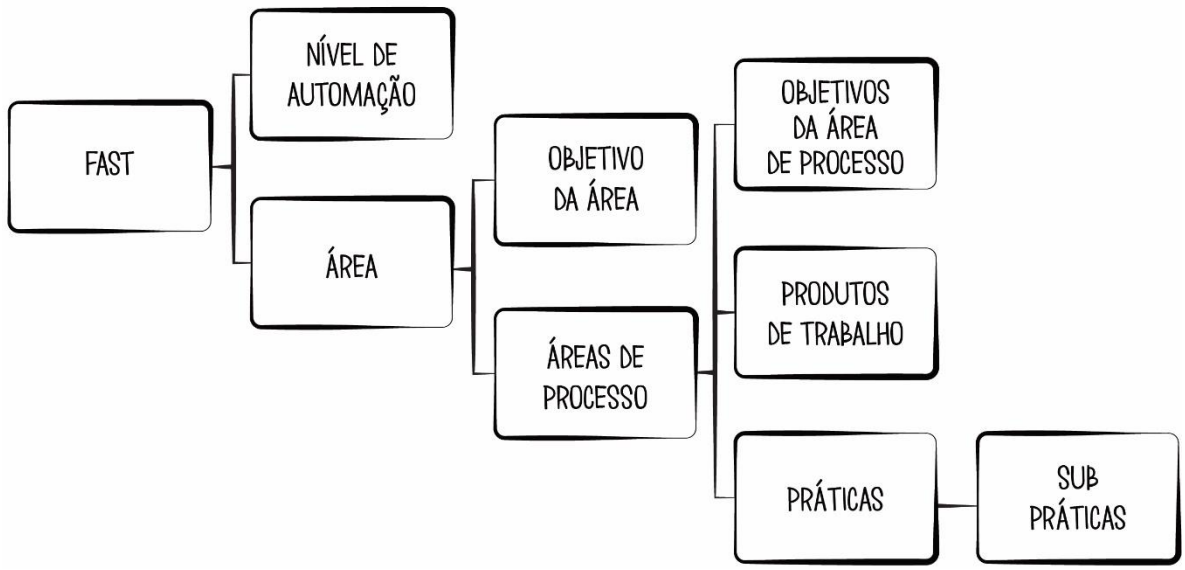
Apesar de o conceito de framework estar mais ligado ao componente técnico para a construção de sistemas, o mesmo foi adaptado para adotar a perspectiva de um **framework teórico** que contempla uma estrutura hierárquica para a implantação de automação de teste a partir de práticas que podem ser instanciadas de acordo com as necessidades específicas e distintas de cada contexto de projeto.

Este, por sua vez, difere de um modelo de maturidade pelo fato de que não há obrigatoriedade na implantação de suas áreas de processo, haja vista que o conceito de níveis de maturidade/capacidade não se aplica ao framework proposto.

A **estrutura conceitual** do FAST é composta por **elementos** que foram definidos a partir de adaptações do CMMI-DEV (SEI, 2010), conforme ilustrado na Figura 21 (5) e descrito no Quadro 3 (5).

O mesmo foi concebido considerando dois aspectos distintos, porém complementares, para compor sua proposta a partir de suas **áreas Técnica e de Suporte**, nas quais cada uma delas possui um conjunto de áreas de processo, conforme apresentado na Figura 22 (5). Nesse contexto, o framework proposto está estruturado do modo que as áreas agregam áreas de processo especificamente organizadas para manter uma semântica coerente com o objetivo de cada área.

Figura 21 (5) – Estrutura conceitual do FAST



Fonte: Elaborada pela autora (2017)

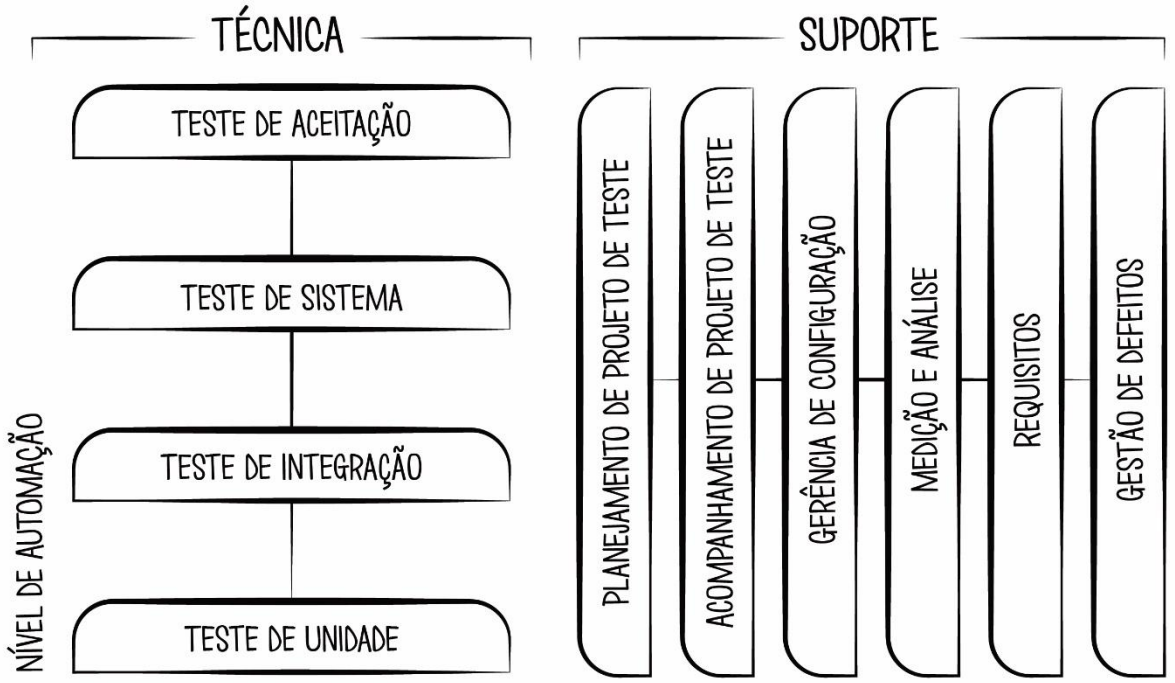
Quadro 3 (5) – Elementos conceituais do FAST

Elemento	**Descrição**
Nível de automação	Este elemento foi adaptado a partir do conceito de nível de teste, descrito como um esforço independente de teste com sua própria documentação e recursos (IEEE 829, 2008). O nível de automação de teste, por sua vez, pode ser compreendido como o escopo de atuação dos processos de automação e está associado à área técnica.
Área	Trata-se do elemento que agrega áreas de processo, considerando os aspectos práticos do modelo que dão ênfase nas principais relações existentes entre as áreas de processo. Corresponde às áreas de interesse, pelas quais o FAST está divido em duas, considerando tanto aspectos **técnicos** como de **suporte**, ambos necessários para introduzir automação em um projeto de software. Este elemento foi adaptado a partir do conceito de categoria do CMMI-DEV (2010).
Objetivo da área	Corresponde ao propósito da área ao qual está relacionado.
Área de processo	Um conjunto de práticas relacionadas a uma determinada área que, quando implementadas coletivamente, satisfazem um conjunto de objetivos considerados importantes para a melhoria daquela área (SEI, 2010).
Propósito da área de processo	Contempla a descrição dos objetivos da área de processo.
Prática	É um elemento que apresenta a descrição das atividades consideradas importantes para que o propósito da área de processo seja alcançado (SEI, 2010). Vale ressaltar que as mesmas não são descritas de maneira a demonstrar sequencialidade entre elas.
Subprática	Descrição detalhada que fornece um guia de interpretação e implementação de uma determinada prática, a partir de ideias que podem ser uteis na instanciação da prática associada. (SEI, 2010).
Produto de trabalho	Sugestões de artefatos associados a uma determinada prática, contemplando informações sobre as saídas a serem elaboradas a partir da execução da mesma.

Fonte: Elaborado pela autora (2017)

As áreas de processo, por sua vez, foram definidas por meio de objetivos, práticas com suas respectivas subpráticas e produtos de trabalho, com intuito de apresentar sugestões de como a automação pode ser introduzida no contexto de um projeto de desenvolvimento de software.

Figura 22 (5) – Áreas e áreas de processo do FAST



Fonte: Elaborada pela autora (2017)

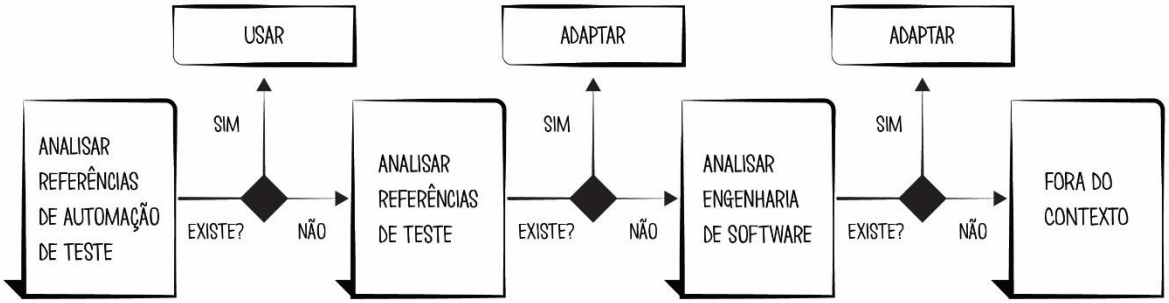
A proposta foi fundamentada de acordo com o que ilustra a Figura 22 (5), a qual representa a composição de áreas de processo para cada uma das duas áreas do framework. A forma de apresentação desta figura, em que as áreas de processo da Área Técnica aparecem na horizontal, e as da Área de Suporte estão dispostas na vertical, foi a forma encontrada para representar que existe um cruzamento entre ambas no momento da instanciação no FAST, e não uma visão isolada e independente de cada área de processo.

A base fundamental para a construção dessa proposta foi a ISO 29.119-2 (2013), que agrupa as atividades de teste em três grupos de processos – Organizacional, Gestão de Teste e Teste Dinâmico – conforme detalhamento apresentado na Seção 3.2.3. Porém, foram feitas algumas adaptações/considerações:

- Os processos organizacionais não foram contemplados no escopo da proposta deste trabalho, porque o seu foco é apresentar formas para a realização da automação de teste no âmbito de projetos e não por meio de políticas organizacionais;
- Os processos de gestão de teste foram abordados na área de suporte; e,
- Os processos de teste dinâmicos foram utilizados como base para a definição dos processos de automação, haja vista que a ISO 29.119 (2) não aborda a automação de teste, apresenta os processos de teste genéricos, sem detalhar o que se aplica ou não aos processos de automação.

A partir dessa visão, foi definida uma estratégia de construção do FAST, conforme apresentada na Figura 23 (5), para que, a partir da revisão bibliográfica, as áreas de processo, práticas e subpráticas pudessem ser construídas. A estratégia utilizada foi construir as áreas de processo do FAST a partir das referências existentes sobre automação de teste. Caso não houvesse bibliografia existente para determinado aspecto do framework, buscou-se adaptar as referências que falam de teste em geral para o contexto de automação. Ainda assim, se não houvesse referência de teste, buscou-se adaptar as referências de engenharia de software para o contexto desta proposta de melhores práticas para automação de teste.

Figura 23 (5) – Estratégia de construção do FAST



Fonte: Elaborado pela autora (2017)

Diante da estrutura apresentada, o FAST foi concebido e será descrito nas próximas Seções.

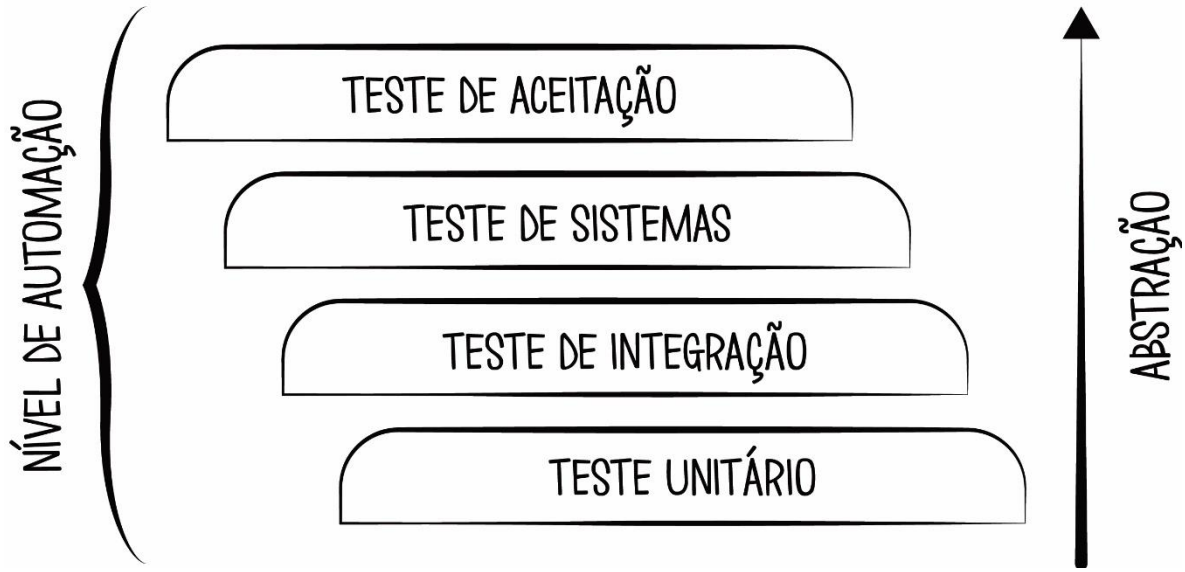
5.2 Área técnica do FAST

A **Área Técnica** foi criada para contemplar as práticas de automação de teste por meio de suas áreas de processos, de forma que elas possam ser adaptadas e implementas de acordo com o contexto no qual serão implantadas. Seu **objetivo** é estabelecer e manter mecanismos para a realização automática do teste de software, com intuito de apoiar a criação de ambientes de projetos que são eficientes para desenvolver, gerenciar e manter os testes automatizados em determinado domínio.

Além disso, essa área tem como foco apresentar aspectos práticos para realização sistemática de testes automáticos, compondo uma arquitetura baseada em áreas de processo que possam ser integradas no processo de desenvolvimento de um produto, fornecendo apoio estratégico à melhoria do processo mediante automação de teste.

A Área Técnica é composta pelas seguintes **Áreas de Processo: (1) Teste Unitário, (2) Teste de Integração, (3) Teste de Sistemas e (4) Teste de Aceitação**, em que cada uma se apresenta de maneira independente e coesa, e pode ser instanciada a partir do nível de automação requerido pelo contexto em questão, não havendo, portanto, pré-requisitos entre si. O **nível de automação** representa o nível de abstração com que a automação será realizada, conforme representado na Figura 24 (5).

Figura 24 (5) – Abstração da Área Técnica



Fonte: Elaborado pela autora.

Além disso, cada área de processo apresenta práticas que incluem projetar, implementar e executar os testes automatizados. As descrições de cada área de processo serão apresentadas nas próximas seções.

5.2.1 Teste unitário

A área de processo Teste Unitário tem o **propósito** de “determinar a corretude e completude de uma implementação, com base nos requisitos da unidade” (IEEE 1008, 1987) e é composta pelas seguintes **práticas**:

1.1. Projetar teste unitário.

1.2. Implementar teste unitário.

1.3. Executar teste unitário.

1.4. Finalizar teste unitário.

Os **produtos de trabalho** relacionados a essa área de processo são:

- Projeto do teste unitário.
- Especificação dos procedimentos de teste.
- Especificação dos casos de teste
- Dados de teste.
- Recursos de apoio ao teste.
- Configuração dos itens de teste.
- Sumário de testes.
- Log da execução de teste.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE A – ÁREA TÉCNICA: 1- TESTE UNITÁRIO.

5.2.2 Teste de integração

A área de processo Teste de Integração tem o **propósito** de avaliar a integração e relação entre os componentes de software para garantir que o sistema esteja consistente com sua arquitetura (IEEE 610-12, 1990), sendo composta pelas seguintes **práticas**:

2.1 Estabelecer técnicas de teste de integração.

2.2 Projetar teste de integração.

2.3 Implementar teste de integração.

2.4 Executar teste de integração.

2.5 Finalizar teste de integração.

Os **produtos de trabalho** relacionados com esta área de processo são:

- Projeto de teste.
- Especificação do projeto de teste.
- Scritps de teste de integração.
- Resultados da execução.
- Sumário de testes.
- Dados de teste.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos APÊNDICE B – ÁREA TÉCNICA: 2- TESTE DE INTEGRAÇÃO.

5.2.3 Teste de sistema

A área de processo Teste de Sistema tem o **propósito** de “realizar testes em um sistema completo e integrado para avaliar sua conformidade com os requisitos especificados” (IEEE 610.12, 1990, p. 74), que é composta pelas seguintes **práticas**:

3.1 Estabelecer tipos e técnicas de teste de sistema.

3.2 Projetar teste de sistema.

3.3 Implementar teste de sistema.

3.4 Executar teste de sistema.

3.5 Finalizar teste de sistema.

Os produtos de trabalho relacionados são:

- Projeto de teste.
- Especificação do projeto de teste.
- Scripts de teste de sistemas.
- Sumário de testes.
- Resultados da execução.
- Dados de teste.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE C – ÁREA TÉCNICA: 3- TESTE DE SISTEMA.

5.2.4 Teste de aceitação

A área de processo Teste de Aceitação tem o **propósito** de “conduzir testes formais para determinar se um sistema satisfaz ou não seus critérios de aceitação” (IEEE 610.12, 1990, p. 8), e é composta pelas seguintes **práticas**:

4.1 Estabelecer tipos e critérios de aceitação.

4.2 Projetar teste de aceitação.

4.3 Implementar teste de aceitação.

4.4 Executar teste de aceitação.

4.5 Finalizar teste de aceitação.

Os **produtos de trabalho** relacionados são:

- Projeto de teste.
- Especificação do projeto de teste.
- Scripts de teste de aceitação.
- Sumário de testes.
- Resultados da execução.
- Dados de teste.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE D – ÁREA TÉCNICA: 4- TESTE DE ACEITAÇÃO.

5.3 Área de suporte do FAST

A Área de Suporte foi criada com o **objetivo** de sugerir práticas que dão suporte à execução dos processos da área técnica para apoiar a implantação, execução e acompanhamento sistemáticos da automação de teste no projeto.

A Área de Suporte é composta pelas **Áreas de Processo (AP) (1) Planejamento do projeto de teste, (2) Acompanhamento do projeto de teste, (3)**

Gerência de configuração, (4) Medição e análise, (5) Requisitos, e (6) Gestão de defeitos, as quais possuem práticas que podem se relacionar entre si.

Além disso, suas APs permeiam todos os níveis de automação da Área Técnica e fornecem apoio à execução da automação a partir das áreas de processo propostas, conforme representado pela Figura 25 (5). As descrições de cada área de processo serão apresentadas nas próximas seções.

Figura 25 (5) – Transversalidade da área de Suporte

Fonte: Elaborada pela autora (2017).

5.3.1 Planejamento de projeto de teste

O **propósito** da área de processo de Planejamento do Projeto de Teste é Estabelecer o planejamento da estratégia e atividades de automação de testes no projeto (TMMI, 2012), que é composta pelas seguintes **práticas**:

1.1 Analisar riscos do produto.

1.2 Estabelecer estratégia de automação do teste.

1.3 Estabelecer estimativas.

1.4 Desenvolver o plano do projeto de automação de teste.

Os **produtos de trabalho** relacionados a essa área de processo são:

- Estrutura analítica do risco – EAR.
- Tabela de riscos do produto.
- Matriz de exposição do risco.
- Estratégia de automação de teste.
- Critérios de automação de teste.
- Estrutura analítica do projeto - EAP.
- Ciclo de vida do projeto de automação.

- Estimativas de tamanho do projeto de automação de teste.
- Estimativa de custo do projeto de automação de teste.
- Cronograma do projeto de automação de teste.
- Matriz de recursos humanos.
- Matriz de envolvimento das partes interessadas.
- Planilha de riscos do projeto.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE E – ÁREA DE SUPORTE: 1- PLANEJAMENTO DO PROJETO DE TESTE.

5.3.2 Acompanhamento de projeto de teste

O **objetivo** da área de processo Acompanhamento do Projeto de Teste é avaliar se o progresso do projeto está ocorrendo de acordo com o planejado e tomar ações corretivas, caso existam desvios significativos em relação ao planejado (TMMI, 2012).

Essa área é composta pelas seguintes **práticas**:

2.1 Monitorar o projeto com base no plano.

2.2 Gerenciar ações corretivas.

O **produto de trabalho** relacionado a essa área de processo é a planilha de acompanhamento do projeto de teste. O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE H – ÁREA DE SUPORTE: 2- ACOMPANHAMENTO DO PROJETO DE TESTE.

5.3.3 Gerência de configuração

O **objetivo** desta área de processo é estabelecer e manter a integridade do ambiente de automação e seus produtos de trabalho no contexto do projeto de automação de teste (ISO/IEC/IEEE 29.119-2, 2013). Essa área é composta pelas seguintes **práticas**:

3.1 Estabelecer ambiente de automação de teste.

3.2 Estabelecer baselines do projeto de automação de teste.

3.3 Acompanhar e controlar mudanças.

Os **produtos de trabalho** relacionados a essa área de processo são:

- Plano de gerência de configuração.
- Sistema de gerência de configuração.
- Sistema de solicitação de mudanças.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE F – ÁREA DE SUPORTE: 3- GERÊNCIA DE CONFIGURAÇÃO.

5.3.4 Medição e análise

O **objetivo** desta área de processo é desenvolver e manter mecanismos de medição para apoiar as atividades de automação de testes no projeto (TMMI, 2012). Essa área é composta pelas seguintes **práticas**:

4.1 Estabelecer mecanismos de medição e análise.

4.2 Fornecer os resultados da medição da automação de teste.

O **produto de trabalho** relacionado a essa área de processo é o plano de medição da automação de teste. O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE G– ÁREA DE SUPORTE: 4- MEDIÇÃO E ANÁLISE.

5.3.5 Requisitos

O objetivo desta área de processo é definir, analisar, estabelecer e manter os requisitos da automação de testes. As práticas associadas a esta área de processo são:

5.1 Definir requisitos da automação.

5.2 Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho.

5.3 Gerenciar mudanças de requisitos da automação.

Os **produtos de trabalho** relacionados a essa área de processo são:

- Critérios para análise dos requisitos.
- Requisitos da automação.

- Esquema conceitual de rastreabilidade.
- Matriz de rastreabilidade.
- Solicitação de mudança.
- Análise de impacto.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos no APÊNDICE I – ÁREA DE SUPORTE: 5- REQUISITOS.

5.3.6 Gestão de defeitos

O **objetivo** desta área de processo é identificar, analisar e tratar os defeitos decorrentes da automação de teste (TMMI, 2012). Além disso, deve dar apoio à investigação e documentação dos incidentes de teste reportados e ao reteste de defeitos, quando necessário (IEEE 29.119-1, 2013).

As seguintes **práticas** estão associadas a essa AP:

6.1 Estabelecer sistema de gestão de defeitos.

6.2 Analisar resultado do teste.

6.3 Estabelecer ações para eliminar a causa-raiz dos defeitos.

Os produtos de trabalho associados a essa área de processo são:

- Máquina de estados do defeito.
- Sistema de gestão de defeitos
- Registro do defeito.
- Análise da causa raiz dos defeitos.
- Ações corretivas.

O detalhamento das práticas e suas respectivas **subpráticas**, assim como os produtos de trabalho gerados em cada uma delas estão descritos APÊNDICE J – ÁREA DE SUPORTE: 6- GESTÃO DE DEFEITOS.

5.4 Considerações finais

Este capítulo apresentou o FAST, mediante descrição de sua estrutura, elementos conceituais, áreas, áreas de processo, práticas, subpráticas e produtos de trabalho relacionados, conforme resumo apresentado nos mapas disponíveis em

APÊNDICE L – MAPA DA ÁREA TÉCNICA e APÊNDICE M – MAPA DA ÁREA DE SUPORTE.

A proposta se diferencia das soluções apresentadas pelos trabalhos relacionados, pois consolida, a partir de uma única proposta, uma visão completa das atividades de automação de testes, mediante os níveis de automação, que podem ser instanciados no contexto do desenvolvimento de software. Além disso, o FAST proporciona um conjunto de práticas que podem ser instanciadas para dar suporte às atividades de automação, por meio das práticas de planejamento, acompanhamento, gerência de configuração, medição, requisitos e gestão de defeitos. Ainda, a estrutura da proposta, a partir de um framework, faz com que o seu uso seja adequado de acordo com cada contexto organizacional, no qual não se define previamente obrigatoriedade de implantação de práticas, conforme pregam os modelos de maturidade.

A originalidade do trabalho foi abordada por meio das melhores práticas existentes na literatura e estruturadas a partir de um framework, de modo que o mesmo apresenta um conjunto consistente de informações que podem apoiar a introdução da automação de teste. A estrutura apresentada é inovadora, pois a partir da análise dos trabalhos relacionados, juntamente com suas respectivas limitações descritas no Quadro 2 (4), pode-se observar que a principal contribuição desta proposta é a estrutura e consolidação das informações existentes na literatura em uma proposta que fornece apoio para introdução sistemática da automação de teste no projeto.

O apoio à introdução sistemática de automação se dá mediante sugestão de práticas para o projeto, implementação, execução e finalização dos testes automatizados para os níveis de teste unitário, integração, sistema e aceitação, nos quais o uso de ferramentas é sugerido para todas essas práticas. O FAST contempla uma proposta mais abrangente do que as dos trabalhos que abordam automação apenas por meio do uso de ferramentas, tais como TMM (1996), TIM (1997), TPI (1998), MPT (2011) e TMMI (2012).

Além disso, também aborda a geração e execução de testes automatizados, conforme proposto pelos trabalhos ATG (2012), STEP (2011) e MPT.BR (2011). Sobre a limitação do AQAM (2011), em que automação de teste é trabalhada apenas no contexto de teste de performance, o FAST trata tal teste como parte do nível de teste

de sistemas, sugerindo que sua automação seja realizada por meio de práticas de projeto, implementação, execução e finalização.

No contexto dos trabalhos MMAST (1994) e TAIM (2014), que abordam um processo para avaliar a automação de teste a partir de, respectivamente, checklists e métricas, o FAST propõe, na área de suporte, uma área de processo para abordar a definição, coleta e análise das medições decorrentes da automação.

Portanto, o capítulo a seguir apresenta o planejamento, execução e resultados coletados do estudo de caso, a partir da implantação do FAST em dois distintos casos.

6 ESTUDO DE CASO

O estudo de caso foi o método de pesquisa empírica selecionado para avaliar a implantação do FAST, e sua execução seguiu os passos apresentados no Capítulo 2, Seção 2.2.2, cujo protocolo será detalhado e dados coletados a partir de sua execução serão apresentados neste capítulo.

6.1 Planejamento

O **objetivo** de um estudo de caso é a definição do que se espera alcançar como resultado de sua execução (RUNESON et al., 2012). Neste contexto, de acordo com a classificação de Robson (2002), o propósito de um estudo de caso é de melhoria, em que, diante do contexto deste trabalho, buscam-se feedbacks do uso do FAST no ambiente real e contemporâneo de projeto de desenvolvimento de software, o qual possui um alto grau de realismo.

Nesse cenário, **perguntas de pesquisa** são declarações sobre o conhecimento que está sendo procurado, ou que se espera ser descoberto durante o estudo de caso (RUNESSON et al., 2012, pp. 30). A partir desse contexto, as perguntas de pesquisa definidas para esse estudo de caso foram as seguintes:

P1- O FAST apoiou a introdução sistemática de automação de teste no projeto?

P2- Quais foram os aspectos positivos (fatores de sucesso) do uso do FAST para introdução da automação de teste no projeto?

P3- Quais foram os aspectos negativos (fatores de insucesso) decorrentes do uso do FAST na implantação de automação de teste no projeto?

P4- Quais as limitações para introdução da automação de teste no projeto, a partir do uso do FAST?

Tais perguntas foram definidas com intuito de consolidar informações que possam servir de insumos para analisar se o FAST é um mecanismo que pode ser utilizado para responder a pergunta de pesquisa desta tese, a qual questiona: Como automação de teste deve ser introduzida e mantida no processo de desenvolvimento software?

Além disso, como parte do planejamento, considerações sobre a ética para a realização do estudo de caso também foram explicitadas mediante acordo de confidencialidade entre o pesquisador e os participantes dos casos selecionados (APÊNDICE N – ACORDO DE CONFIDENCIALIDADE) A. Todos acordaram em fazer parte deste estudo e as informações aqui fornecidas estão de acordo com os princípios de manutenção da integridade de todos os membros.

6.1.1 Caracterização dos Casos

Para a execução desse estudo empírico, 2 casos foram selecionados, baseados na disponibilidade e por conveniência (RUNESON e HÖST, 2008), os quais foram descritos com base na proposta de Baldassarre et al. (2016) e serão apresentados nas próximas seções.

6.1.1.1 Caso 1

O Caso 1 ocorreu no contexto de um projeto de desenvolvimento de um produto para gestão acadêmica, oferecido em forma de serviços (SaaS), com funcionalidades voltadas para redes de escolas públicas e privadas, por meio do gerenciamento de unidades de ensino, turmas, notas, matrícula de aluno, fechamento de ano letivo, entre outras. O produto está embasado nas seguintes tecnologias:

- PHP 7 (linguagem de desenvolvimento);
- Laravel (framework para desenvolvimento em PHP);
- Redis (estrutura de banco de dados em memória utilizado para cache);
- ElasticSearch (ferramenta de buscas para tratar grandes quantidades de dados em tempo real);
- Postgresql (banco de dados usado);
- Git (ferramenta para versionamento do projeto);
- VueJS (framework para JavaScript);
- Bootstrap (framework para CSS/HTML);
- Gulp (task runner); e
- NGinx (servidor HTTP).

A descrição dos fatores contextuais e a caracterização da equipe do Caso 1 estão, respectivamente, descritas no Quadro 4 (6) e Quadro 5 (6).

Quadro 4 (6) – Fatores contextuais do Caso 1 (C1)

Grupo	
Composição do grupo	1 *Product Owner* (PO) parcialmente alocado à equipe. 1 *Scrum Master* parcialmente alocado à equipe. 6 desenvolvedores 100% alocados na equipe.
Estilo de gestão	Gestão ágil de projetos com base na metodologia *Scrum*, com Sprints de duração mensal.
Clima da equipe	Equipe comprometida, bastante motivada e que demonstra interesse pelo projeto, equipe unida que não trabalha de forma isolada.
Dispersão geográfica	A equipe do projeto está fisicamente alocada na mesma sala de trabalho.
Processos da Equipe	
Relacionamento com o cliente	Cliente interno representado pelo PO, que é um diretor da empresa responsável pelo produto que está sendo desenvolvido. A equipe mantém um relacionamento saudável com o PO, que também é considerado um membro da equipe.
Dinâmica de comunicação	A comunicação do time é realizada com base na metodologia Scrum, onde existem as reuniões de planejamento da Sprint, reuniões de acompanhamento quinzenal, reuniões diárias e reuniões técnicas eventuais, além de reuniões de fechamento da Sprint. Comunicação direta com o PO sempre que necessário.
Processo de desenvolvimento de software	O produto está sendo desenvolvido de forma iterativa e incremental. O Scrum é a metodologia utilizada para a gestão de projetos em conjunto com práticas do MPS.BR nível F.
Cronograma de trabalho	A equipe trabalha com cronograma contemplando Sprints com 1 mês duração.
Ambiente do Projeto	
Patrocinador	Interno, em que o diretor de inovação é o PO do projeto.
Importância e criticidade do projeto	O projeto tem alta importância para organização pois está atrelado ao planejamento estratégico de evolução tecnológica e conceitual do produto atual da empresa. Considerado na mais alta escala de importância e criticidade para a organização.
Complexidade e variabilidade do projeto	O grau de complexidade é considerado como severo pela organização. Requisitos considerados estáveis, com baixa variabilidade no projeto.
Tamanho e duração do projeto	Tamanho do projeto é de 1309 pontos de função (PF). O projeto iniciou em 05/10/2015 e ainda não foi finalizado. O estudo de caso começou em 21/12/2016 e finalizou em 25/04/2017
Ambiente da Organização	
Estrutura do escritório	Sala da equipe, onde os desenvolvedores e Scrum Master trabalham em conjunto com outras equipes de projetos. Sala da diretoria, onde o PO trabalha em conjunto com os demais diretores da empresa.
Modelo de negócios	O produto está sendo desenvolvido com previsão de lançamento no mercado para agosto de 2017.
Tamanho e estrutura organizacional	27 membros em toda a empresa.

Fonte: Elaborado pela autora (2017)

Quadro 5 (6) – Caracterização dos indivíduos participantes (P) do Caso 1 (C1)
(continua)

Participante 1 (P1C1)	
Nível de experiência	18 anos de experiência prática.
Dados demográficos	Idade: 38 anos Gênero masculino. Tecnólogo em internet e redes de computadores. MBA em Gestão da Inovação. Função: Diretor de inovação na empresa. Papel no projeto: Product Owner (PO).
Interesses individuais	Possui interesse pessoal em se apropriar do conhecimento em automação de teste para que possa avaliar, sob a perspectiva de gestão de projetos, se a automação de testes é possível, viável e eficaz no projeto.
Habilidades técnicas	Desenvolvimento WEB, infraestrutura de recursos computacionais para internet. Gerência de projetos e gestão de inovação. Conhecimento e experiência prática no domínio de gestão acadêmica. Habilidades no âmbito de design centrado em usuário, front end.
Participante 2 (P2C1)	
Nível de experiência	9,5 anos de experiência prática.
Dados demográficos	Idade: 32 anos. Gênero masculino. Mestrado em Redes de Computadores. Graduado em Ciência da Computação. Função: Desenvolvedor
Interesses individuais	Interesse em pesquisas relacionadas à teste de software com intenção de entrar no programa de doutorado na área. Interesse na área
Habilidades técnicas	Java, C++, hibernate, android, php, J2ME, android
Participante 3 (P3C1)	
Nível de experiência	10 anos de experiência prática.
Dados demográficos	Idade: 31 anos. Gênero feminino. Doutorado em Ciência da Computação. Mestrado em Ciência da Computação. Graduado em Sistemas para Internet. Função: Desenvolvedora.
Interesses individuais	Não há interesses pessoais com o estudo de caso.
Habilidades técnicas	Java e PHP (framework Laravel); Banco de dados: MySql e Postgres; Testes automatizados com Cucumber, JUnit e PHPUnit; Scrum; Git, HTML, CSS, Wordpress.
Participante 4 (P4C1)	
Nível de experiência	5 anos de experiência prática.
Dados demográficos	Idade: 24 anos. Gênero feminino. Graduada em Ciência da Computação. Função: Desenvolvedora.
Interesses individuais	Interesse em aprender sobre automação para usar em projetos futuros.
Habilidades técnicas	PHP (frameworks Laravel e Cake); Banco de dados: MySql, Postgres e SQLServer; Gerenciamento de projetos; Scrum; Git; HTML, CSS, Wordpress.
Participante 5 (M5C1)	
Nível de experiência	1 ano de experiência prática.
Dados demográficos	Idade: 22 anos. Gênero masculino. Graduado em Sistemas de Informação. Função: Desenvolvedor.

Quadro 5 (6) – Caracterização dos indivíduos participantes (P) do Caso 1 (C1) (continuação)

Participante 5 (M5C1)	
Interesses individuais	Não há interesses pessoais com o estudo de caso.
Habilidades técnicas	Rails; PHP (frameworks Laravel); Banco de dados: MySql, Postgres e SQLServer; Scrum; Git; HTML, CSS, Javascript.
Participante 6 (P6C1)	
Nível de experiência	8 anos de experiência prática.
Dados demográficos	Idade: 28 anos. Gênero masculino. Graduado em Sistemas para Internet. Função: Desenvolvedor.
Interesses individuais	Interesse em aprender sobre automação para usar em projetos futuros.
Habilidades técnicas	PHP (frameworks Laravel e Cake); Banco de dados: MySql, Postgres; Scrum; Git; HTML, CSS, Wordpress, Javascript. SaaS. LesS.
Participante 7 (P7C1)	
Nível de experiência	5 anos de experiência prática.
Dados demográficos	Idade: 32 anos. Gênero masculino. Mestrado em Engenharia de Sistemas. Graduado em Ciência da Computação. Função: Desenvolvedor
Interesses individuais	Não há interesses pessoais com o estudo de caso.
Habilidades técnicas	C++, C, Java, C#, Python, Ruby, R, PHP (frameworks Laravel); Banco de dados: MySql, Postgres, SQLServer, Oracle; Scrum; Git; HTML, CSS, Flex. ElasticSearch.
Participante 8 (P8C1)	
Nível de experiência	25 anos de experiência prática.
Dados demográficos	Idade: 49 anos. Gênero masculino. Graduado em Web Design e Programação. Função: Scrum Master
Interesses individuais	Não há interesses pessoais com o estudo de caso.
Habilidades técnicas	Desenvolvimento desktop atualmente na linguagem Delphi, conhecimento em PHP, HTML, Java, Javascript. Certified ScrumMaster.

Fonte: Elaborado pela autora (2017)

6.1.1.2 Caso 2

O Caso 2 ocorreu no contexto do desenvolvimento de um sistema que centraliza as identidades de usuários de diversos sistemas da organização, desenvolvido para atender aos requisitos do protocolo OAuth 2, com o objetivo de otimizar a gestão de logins para os vários sistemas e facilitar a integração de novos sistemas. O produto está embasado nas seguintes tecnologias:

- JAVA 8 (Linguagem de programação do *back end*);
- Tomcat (Web container);

- Spring MVC (framework para desenvolvimento em JAVA);
- PostgreSQL (Banco de dados usado para dados do projeto);
- Oracle (Banco de dados usado para dados do ERP educacional);
- Rest Full Web services (Padrão de interoperabilidade de sistemas WEB);
- Angular Material (framework para desenvolvimento JavaScript para front end) e
- Gradle (gerenciador de configuração do projeto).

A descrição dos fatores contextuais e a caracterização da equipe do Caso 2 estão, respectivamente, descritas no Quadro 6 (6) e Quadro 7 (6).

Quadro 6 (6) – Fatores contextuais do Caso 2 (C2) (continua)

Grupo	
Composição do grupo	1 líder de projeto parcialmente alocado, que também atua como desenvolvedor. 1 arquiteto e PO. 1 designer. 2 desenvolvedores 1 consultor de teste alocado para acompanhar a experiência de introdução de automação para, possivelmente, replicá-la em outros projetos.
Estilo de gestão	Gestão ágil de projetos com base na metodologia *Scrum*, com Sprints de duração de 10 dias úteis.
Clima da equipe	Equipe comprometida e motivada.
Dispersão geográfica	A equipe do projeto está fisicamente alocada na mesma sala de trabalho.
Processos da Equipe	
Relacionamento com o cliente	Cliente interno da organização que é representado pelo PO, no escopo do projeto e que faz parte da equipe.
Dinâmica de comunicação	A comunicação do time é realizada com base na metodologia Scrum, onde existem as reuniões de planejamento da Sprint, reuniões de acompanhamento quinzenal, reuniões diárias e reuniões técnicas eventuais, além de reuniões de fechamento da Sprint. Comunicação direta com o PO sempre que necessária.
Processo de desenvolvimento de software	O projeto está sendo desenvolvido de forma iterativa e incremental. O Scrum é a metodologia utilizada para a gestão de projetos com práticas do *Extreme Programming* (XP).
Cronograma de trabalho	A equipe trabalha com cronograma contemplando Sprints de 10 dias úteis de duração.
Ambiente do Projeto	
Patrocinador	Interno, responsável pela comunicação da organização que, no projeto, está sendo representado pelo PO.
Importância e criticidade do projeto	Projeto com importância e criticidade alta, pois está relacionado com a segurança de informações, que impactam em aproximadamente 55 mil usuários.
Complexidade e variabilidade do projeto	Projeto com complexidade alta porque trabalha com a integração de diversos sistemas. Requisitos considerados estáveis.

Quadro 6 (6) – Fatores contextuais do Caso 2 (C2) (continuação)

Tamanho e duração do projeto	Tamanho do projeto é de 24 pontos de função (PF). O projeto iniciou em janeiro de 2017 e ainda não foi finalizado. O estudo de caso começou em 05/01/2016 e finalizou em 19/05/2017 Atualmente o projeto está na Sprint número 4, mas 2 Sprints foram abortadas ao longo do projeto até a atualidade.
Ambiente da organização	
Estrutura do escritório	Equipe do projeto trabalha no mesmo ambiente físico, inclusive o PO.
Modelo de negócios	Não se aplica, pois trata-se de uma instituição pública de ensino superior, cujo estudo de caso foi rodado no órgão responsável pela área de tecnologia da informação.
Tamanho e estrutura organizacional	98 servidores federais mais 41 bolsistas de graduação.

Fonte: Elaborado pela autora (2017)

Quadro 7 (6) – Caracterização dos indivíduos participantes (P) Caso 2 (C2) (continua)

Participante 1 (P1C2)	
Nível de experiência	14 anos de experiência prática
Dados demográficos	Idade: 33 anos. Gênero masculino. Graduado (bacharel) em sistemas da informação. Função: Coordenador de projetos. Papel no projeto: Arquiteto, desenvolvedor e PO.
Interesses individuais	Não há interesses pessoais com o estudo de caso.
Habilidades técnicas	Desenvolvedor Java, arquiteto de software, proficiente em todos os aspectos de desenvolvimento de software e análise de sistemas, incluindo a gestão e implementação de sistemas distribuídos em grande escala. Conhecimento nas principais tecnologias, tais como - Java /Java 2, J2EE, Ruby, JavaScript, SQL, DHTML, HTML, XML, JavaBeans, RMI, JSP, EJB, J, DBC, SOAP, Rest Full, Multi-threading, Java Networking, Socket Programming, Junit, Hibernate, Oracle 9i, MS SQL Server, MySQL, PostgeSQL, DB2, MongDB, Redis, Wicket, JSF, Jakarta Struts, Rails, Spring, JSF, Apache, Jakarta Tomcat, IBM WebSphere, JBoss, Thin, NGINX, Rational Rose, Jude, visual paradigm CVS, SVN, Git, Mercurial, MS Windows, Linux, Mac OSx, Andorid, IOs M3.0, Certified ScrumMaster (CSM)
Participante 2 (P2C2)	
Nível de experiência	5 anos.
Dados demográficos	Idade: 27 anos. Gênero masculino. Graduado em Análise de Sistemas. Papel no projeto: Líder de projeto e desenvolvedor
Interesses individuais	Não há.
Habilidades técnicas	Java, Bancos de dados, Angula JS
Participante 3 (P3C2)	
Nível de experiência	6 anos
Dados demográficos	Idade: 27 anos. Gênero masculino. Doutorando em Ciência da Computação. Mestre em Ciência da Computação. Graduado em Ciência da Computação. Papel no projeto: Consultor em teste de software

Quadro 7 (6) – Caracterização dos indivíduos participantes (P) Caso 2 (C2) (continuação)

Interesses individuais	Interesse em adquirir conhecimento sobre automação para poder planejar a introdução de automação nos demais projetos da organização.
Habilidades técnicas	Java, Python, Banco de dados. Certificado em teste de software (*Certified Tester Foudation Level* – CTFL). Ferramentas de teste como Selenium, Cucumber, Test Link. Experiência com ferramentas de gerência de configuração.
Participante 4 (P4C2)	
Nível de experiência	3 anos
Dados demográficos	Idade: 26 anos. Gênero masculino. Bacharel em Sistemas de Informação Papel no projeto: Desenvolvedor.
Interesses individuais	Não há.
Habilidades técnicas	Java; Angular JS; Bancos de Dados: MySQL, Postgres, Oracle
Participante 5 (P5C2)	
Nível de experiência	12 anos
Dados demográficos	Idade: 31anos. Gênero masculino. Formação: Web Design (Sistemas para Internet) Papel no projeto: Desenvolvedor.
Interesses individuais	Não há.
Habilidades técnicas	User Interface, JavaScript, Java, PostgreSQL
Participante 6 (P6C2)	
Nível de experiência	6 anos
Dados demográficos	Idade: 33 anos. Gênero masculino. Formação em Engenharia da Computação Papel no projeto: Desenvolvedor.
Interesses individuais	Não há.
Habilidades técnicas	AngularJs, JAVA, Spring boot, Git, Angular material.

Fonte: Elaborado pela autora (2017)

6.1.2 Processo de implantação do FAST

O processo de pesquisa para execução de um estudo de caso pode ser classificado como flexível (ANASTAS e MACDONALD, 1994; ROBSON, 2002), onde os parâmetros determinados no planejamento podem ser adaptados no decorrer do mesmo. A partir do pressuposto, o processo de implantação do FAST para realização do estudo de caso foi desenhado a partir das atividades apresentadas na Figura 26 (6) e descritas a seguir:

Figura 26 (6) – Atividades planejadas para implantação do FAST

Fonte: Elaborada pela autora (2017)

1- **Apresentação da proposta**: esta atividade foi realizada a partir de uma reunião com responsável técnico para apresentar o objetivo do estudo de caso e obter aceitação para a realização do mesmo.
2- **Realização do diagnóstico**: o objetivo foi avaliar se a automação e o teste eram realizados no contexto dos casos avaliados. O diagnóstico foi realizado de forma que, a partir das áreas de processo do FAST, e suas respectivas práticas, foi avaliado o grau de implementação da prática no projeto, considerando a seguinte escala:
   - **Contempla**: as práticas são realizadas de maneira sistemática no projeto.
   - **Parcialmente**: as práticas são realizadas de maneira não sistemática no projeto.
   - **Não contempla**: as práticas não são realizadas no projeto.
3- **Reunião de Kickoff**: esta atividade foi realizada para apresentar o resultado do diagnóstico e plano de trabalho do estudo de caso.
4- **Execução dos treinamentos**: foram realizados treinamentos no FAST, com o intuito de apresentar a sua estrutura, contemplando as áreas, objetivos, áreas de processo e suas práticas.
5- **Acompanhamento**: realizado por meio de reuniões para avaliar o andamento e esclarecimento de dúvidas decorrentes da implantação do FAST.
6- **Realização das entrevistas**: coleta dos dados qualitativos a partir de reuniões individuais com os participantes.
7- **Fechamento do estudo de caso**: coleta dos indicadores do estudo de caso e fechamento do projeto, mediante apresentação do feedback para os participantes.

Com base nas atividades descritas, a implantação do FAST aconteceu conforme cronograma detalhado no APÊNDICE O – CRONOGRAMA DE EXECUÇÃO DO CASO 1 e no APÊNDICE P – CRONOGRAMA DE EXECUÇÃO DO CASO 2. O diagnóstico do Caso 1 está apresentado no APÊNDICE Q – DIAGNÓSTICO DO CASO 1 e o do Caso 2 no APÊNDICE R – DIAGNÓSTICO DO CASO 2.

6.2 Preparação para a coleta de dados

Os métodos de coleta selecionados foram as Entrevistas e coleta de Métricas, que serão detalhados nas próximas seções.

6.2.1 Entrevistas

A entrevista é um processo em que o pesquisador e o participante interagem em uma conversação focada em perguntas relacionadas à pesquisa, considerado adequado para se analisarem eventos passados que não podem ser replicados (DE MARIS, 2004). De acordo com Merriam (2016, p. 84), a entrevista “é necessária quando não se consegue observar comportamento, sentimentos (...)(...) quando se está interessado em eventos passados que são impossíveis de serem replicados”.

No contexto do estudo de caso, a entrevista foi selecionada para coletar os dados qualitativos gerados a partir da introdução da sistematização da automação de teste mediante implantação do FAST no contexto dos 2 Casos selecionados. Nesse cenário, a mesma foi planejada para ser executada de maneira semiestruturada (MERRIAM, 2016), com perguntas previamente planejadas, cuja ordem pode variar de acordo com o decorrer da entrevista. O protocolo da entrevista foi organizado de acordo com o princípio do modelo funil (RUNESON e HÖST, 2008), no qual as perguntas mais genéricas foram abordadas de início e depois refinadas para as mais específicas, conforme representado na Figura 27 (6) e descrito no Quadro 8 (6).

Figura 27 (6) – Estratégia de elaboração do questionário de entrevista

Fonte: Elaborada pela autora (2017)

Quadro 8 (6) – Script da entrevista semiestruturada (continua)

PARTE	**DESCRIÇÃO**
Parte 1: Iniciação	Apresentar objetivos da entrevista e do estudo de caso. Explicar como os dados do estudo de caso serão utilizados. Apresentar termos de confidencialidade.
Parte 2: Perguntas introdutórias	Caracterização do participante, considerando os seguintes aspectos: • Nível de experiência; • Dados demográficos; • Interesses individuais; e • Habilidades técnicas.
Parte 3: Perguntas principais.	**Avaliação dos níveis de automação:** 1. O que você acha da organização e dos objetivos dos níveis de automação? 2. Eles estão adequados para introduzir automação de teste no projeto? 3. Você acha que os níveis de automação apoiaram o planejamento da estratégia de automação de teste? **Avaliação das áreas de processo da Área de Suporte:** 4. O que você acha das áreas de processo da área de suporte? (Planejamento, Acompanhamento, Gerência de Configuração, Requisitos, Medição, Gestão de Defeito) 5. Estão adequadas para apoiar a automação de teste no projeto? 6. Você sentiu falta de alguma área de processo? Precisa adicionar alguma área de processo à Área de Suporte? 7. Você sentiu falta de alguma prática? 8. Você sentiu falta da especificação e/ou descrição de quais aspectos do FAST para usá-lo? **Avaliação das áreas de processo da Área Técnica:** 9. O que você acha das áreas de processo da área de técnica? (Teste Unitário, Teste de Integração, Teste de Sistemas e Teste de Aceitação) 10. Estão adequadas para apoiar a automação de teste no projeto? 11. Você sentiu falta de alguma área de processo? Precisa adicionar alguma área de processo à Área Técnica? 12. Você sentiu falta de alguma prática? 13. Você sentiu falta da especificação e/ou descrição de quais aspectos do FAST para usá-lo? **Gerais**: 14. O FAST facilitou a introdução sistemática de automação de teste no projeto? 15. Quais aspectos positivos (fatores de sucesso) do uso do FAST para introdução da automação de testes no projeto?

Quadro 8 (6) – Script da entrevista semiestruturada (continuação)

	16. Quais aspectos negativos (fatores de insucesso) decorrentes do uso do FAST na implantação de automação de teste no projeto? 17. Quais as limitações para introdução da automação de teste no projeto a partir do uso do FAST? 18. Como você avalia a viabilidade (exequibilidade) do FAST? 19. Como você avalia a integridade do FAST? 20. Em relação à distribuição das áreas de processo em Área Técnica e Área de Suporte, você considera adequado para implantação de automação no projeto? 21. Você considera o FAST adequado para apoiar a introdução da automação de teste no seu projeto? Quais as dificuldades do uso do FAST no seu projeto?
Parte 4: Fechamento	Sumarizar, em conjunto com o entrevistado, os principais achados da entrevista para garantir o correto entendimento das respostas

Fonte: Elaborado pela autora (2017)

6.2.2 Métricas

A coleta de métricas é uma técnica focada em dados quantitativos, por meio da análise tanto de dados existentes como daqueles definidos a partir da necessidade do estudo de caso (RUNESON e HOST, 2008). Nesse contexto, o uso de métricas foi selecionado para complementar a visão qualitativa adquirida a partir das entrevistas e para melhor caracterizar os resultados alcançados em cada contexto aplicado.

Para a definição das métricas, utilizou-se o método *Goal Question Metric* – GQM (Objetivo Pergunta Métrica) (BASILI e WEISS 1984; VAN SOLINGEN e BERGHOUT, 1999), conforme descritas no Quadro 9 (6).

Quadro 9 (6) – Descrição das métricas a coletadas no estudo de caso (continua)

MÉTRICA 1: TAMANHO DO PROJETO	
Descrição:	Contagem do escopo do projeto através do uso da técnica de pontos de função.
Objetivo:	Apresentar o tamanho do escopo de atuação do estudo de caso, incluindo escopo já desenvolvido antes do início do estudo de caso (se aplicável).
Pergunta de pesquisa:	Não se aplica.
Unidade de medida:	Pontos de função (PF).
MÉTRICA 2: TAMANHO DO PLANEJAMENTO DA AUTOMAÇÃO DO PROJETO	
Descrição:	Contagem do escopo do projeto planejado para a automação através da definição da estratégia de automação de teste, separados por nível de automação.
Objetivo:	Apresentar o tamanho do projeto de automação planejado a partir do uso do FAST.

Pergunta de pesquisa:	P1. O FAST facilitou a introdução sistemática de automação de teste no projeto?

Quadro 9 (6) – Descrição das métricas a coletadas no estudo de caso (continuação)

Unidade de medida:	Pontos de função
MÉTRICA 3: TAMANHO DO REALIZADO DA AUTOMAÇÃO DO PROJETO	
Descrição:	Contagem do escopo de automação realizado no projeto, separados por níveis de automação, conforme estratégia de automação de teste.
Objetivo:	Apresentar o tamanho do projeto de automação já realizado a partir do uso do FAST.
Pergunta de pesquisa:	P1. O FAST facilitou a introdução sistemática de automação de teste no projeto?
Unidade de medida:	Pontos de função.
MÉTRICA 4: ESFORÇO DA AUTOMAÇÃO DE TESTE NO PROJETO PILOTO	
Descrição:	Contagem do esforço gasto pelos participantes no projeto.
Objetivo:	Avaliar o esforço de tempo envolvido no projeto de automação de teste.
Pergunta de pesquisa:	P1. O FAST facilitou a introdução sistemática de automação de teste no projeto?
Unidade de medida:	Horas.
MÉTRICA 5: QUANTIDADE DE DEFEITOS	
Descrição:	Apresentar a quantidade de defeitos encontrados a partir da execução da automação de teste.
Objetivo:	Avaliar a efetividade da automação de testes executada a partir da introdução do FAST.
Pergunta de pesquisa:	P1. O FAST facilitou a introdução sistemática de automação de teste no projeto?
Unidade de medida:	Número inteiro.
MÉTRICA 6: RETORNO SOBRE O INVESTIMENTO (ROI)	
Descrição:	Apresentar o retorno sobre o investimento realizado para o projeto de automação de testes.
Objetivo:	Prover informações para avaliar a viabilidade do projeto de automação de teste.
Pergunta de pesquisa:	P4. Quais as limitações para introdução da automação de teste no projeto a partir do uso do FAST?
Unidade de medida:	ROI = Porcentagem CCSA = horas CCCA = horas CA = horas
Medidas:	ROI = Retorno sobre o investimento CCSA = Custo correção sem automação CCCA = Custo correção com automação CA = Custa da automação
Fórmula:	ROI=((CCSA-CCCA)/(CA))*100

Fonte: Elaborado pela autora (2017)

6.3 Coleta dos dados

Para coletar os dados das entrevistas, os participantes foram selecionados com base em uma amostra não probabilística, considerado como o método mais

adequado para a pesquisa qualitativa, por meio de uma amostragem intencional, baseada no pressuposto de que o investigador quer compreender uma determinada visão/entendimento e, portanto, deve selecionar uma amostra a partir da qual o máximo pode ser aprendido (MERRIAN, 2016).

Nesse contexto, a seleção dos participantes foi realizada com base no critério de envolvimento do mesmo no estudo de caso, considerado importante porque está associado com os objetivos do estudo de caso de analisar a introdução do FAST no contexto de cada Caso e, quanto maior o envolvimento do participante, maiores serão as contribuições que ele pode fornecer durante a entrevista. A partir dessa definição, o Quadro 10 (6) apresenta a lista dos participantes selecionados, assim como a data e duração de cada entrevista.

Quadro 10 (6) – Lista de participantes selecionados

DATA	PARTICIPANTE	CASO	DURAÇÃO
29/03/2017	P2C1	C1	32 min
29/03/2017	P3C1	C1	44 min
18/04/2017	P1C1	C1	58 min
11/05/2017	P3C2	C2	48 min
18/05/2017	P1C2	C2	56 min

Fonte: Elaborado pela autora (2017)

Além dos dados qualitativos oriundos das entrevistas, métricas também foram coletadas, conforme detalhamento apresentado no Quadro 11 (6).

Quadro 11 (6) – Métrica coletadas do estudo de caso

MÉTRICAS	**CASO 1** 21/12/2016 a 25/04/2017	**CASO 2** 05/01/2016 a 19/05/2017
1: Tamanho do projeto	1.309 PF	24 PF
2: Tamanho do planejamento da automação	Teste unitário: 1.309 PF	Teste unitário: 12 PF Teste integração: 20 PF
3: Tamanho do realizado da automação do projeto	Teste unitário: 383 PF	Teste unitário: 7 PF Teste integração: 12 PF
4: Esforço da automação de teste no projeto	244h	180h
5: Quantidade de defeitos	0	0
6: ROI	Não coletada	Não coletada

Fonte: Elaborado pela autora (2017)

6.4 Análise dos dados

O processo de análise de dados é dinâmico e recursivo, e se torna mais extenso ao longo do andamento do estudo (MERRIAN, 2016). Os dados quantitativos, a partir da coleta das métricas, serviram para caracterizar o escopo de cada projeto do estudo de caso e serão analisados separadamente por Caso.

6.4.1 Contextualização do Caso 1

Em relação ao Caso 1, o projeto já havia iniciado 14 meses antes do início do estudo de caso e, portanto, seu tamanho de 1309 PF's (**métrica 1**) considerou todas as funcionalidades desenvolvidas desde o início até o momento da contagem, que ocorreu 2 meses após seu início. Como se tratava de um projeto em andamento, cujos requisitos estavam sendo definidos e implementados, o escopo do projeto aumentou até o final do estudo de caso, mas, devido a limitações e disponibilidade da equipe, a contagem de tamanho do projeto não pôde ser atualizada.

No que se refere à **métrica 2**, tamanho do planejamento da automação, o seu valor foi igual ao tamanho do projeto pelo fato de que a estratégia de automação planejada era de que todo o escopo do projeto deveria possuir testes unitários automatizados. Por questões estratégicas da organização, e como uma forma de introduzir gradativamente a automação de teste, o projeto priorizou apenas a automação dos testes unitários. No entanto, a estratégia de automação não excluiu o planejamento para automatizar os testes nos níveis de integração e sistemas, porém os mesmos não foram realizados durante o período em que o estudo de caso foi analisado. Apesar disso, o Caso 1 não realizou a contagem do escopo planejado para os níveis de integração e sistemas e por isso, a métrica 2 contemplou apenas o tamanho planejado para testes unitários. Tal contagem não foi realizada por ser uma atividade mais complexa, que demandava um tempo maior. A complexidade nessa contagem está principalmente associada ao teste de integração, haja vista que este se relaciona com diversas funcionalidades e está diretamente associado com a arquitetura do sistema.

Diante desse contexto, no período do estudo de caso, a automação foi introduzida apenas no teste unitário, no qual 383 PFs (**métrica 3**) foram automatizados e executados, de um total de 1309 PFs, ou seja, 29,26% dos testes unitários foram automatizados. Vale ressaltar que, apesar de o estudo de caso ter

iniciado em 21/12/2016, a automação em si não começou nesta data, haja vista que o Caso 1 investiu no planejamento, configuração do ambiente e entendimento do FAST, para que a automação pudesse ser iniciada.

A **métrica 4** trouxe o valor de 244h gastas para o projeto, incluindo os treinamentos, reuniões de acompanhamento, planejamento, projeto, implementação e execução dos testes unitários automatizados. Esse número representa 11,14% de todo o esforço gasto nesse período, cuja soma foi de 2.191h gastas por toda a equipe.

Em relação à quantidade de defeitos, **métrica 5**, o Caso 1 não reportou defeito pelo fato de que os defeitos decorrentes da automação de teste unitário não geram um registro de defeitos na ferramenta, haja vista que essa automação foi realizada pelo desenvolvedor e no mesmo momento em que o defeito foi encontrado, o mesmo foi diretamente reparado. Dessa forma, o fato de a métrica 5 estar zerada, não significa que a automação do teste unitário não gerou defeito, mas, sim, que a mesma não foi contabilizada devido à estratégia da organização em não incluir, em seu processo, o registro formal de defeitos oriundos da automação do teste unitário.

No âmbito do ROI, **métrica 6**, esta não pôde ser coletada por fazer uso de dados históricos sobre o custo da correção do defeito sem a automação de teste, dado este que a organização não tem e, portanto, não pode gerar o cálculo da mesma.

Como forma de complementar o entendimento do contexto após a realização do estudo de caso, foi feito um diagnóstico das práticas do projeto com base no FAST para poder fornecer a visão comparativa entre o cenário de automação do projeto no seu início e ao seu final. No cenário do Caso 1, pode-se observar que as áreas de processo Teste unitário, Planejamento do projeto de teste, Acompanhamento do projeto de teste e Gerência de configuração tiveram suas práticas modificadas para inclusão de atividades de automação de teste, conforme apresenta Quadro 12 (6).

Algumas práticas, apesar de terem sido planejadas, não foram realizadas, como, por exemplo, no caso da 2.2. Gerenciar Ações Corretivas, não foram formalizadas e/ou identificadas ações do projeto de automação que precisassem ser tratadas no escopo do estudo de caso. No caso das práticas de gerência de configuração, as mesmas não foram contempladas de forma isolada, as baselines do projeto de automação ficaram na mesma hierarquia das do projeto de desenvolvimento e por isso não foi tratada de forma separada. Tampouco foram

abertas solicitações de mudanças devido ao prazo do estudo de caso não ter gerado essa demanda.

Quadro 12 (6) – Diagnóstico do FAST Caso1 pós-estudo de caso

ÁREA DE PROCESSO	PRÁTICA	INCIAL	FINAL
TESTE UNITÁRIO	1.1. Projetar teste unitário.	Parcialmente	Contempla
	1.2. Implementar teste unitário.	Parcialmente	Contempla
	1.3. Executar teste unitário.	Parcialmente	Contempla
	1.4. Finalizar teste unitário.	Parcialmente	Contempla
PLANEJAMENTO DO PROJETO DE TESTE	1.1. Analisar riscos do produto.	Não Contemp.	Contempla
	1.2. Estabelecer estratégia de automação do teste.	Não Contemp.	Contempla
	1.3. Estabelecer estimativas.	Parcialmente	Contempla
	1.4. Desenvolver o plano do projeto de automação de teste.	Não Contemp.	Contempla
ACOMPANHAMENTO DO PROJETO DE TESTE	2.1. Monitorar o projeto de teste com base no plano.	Parcialmente	Contempla
	2.2. Gerenciar ações corretivas.	Não Contemp.	Não Contemp.
GERÊNCIA DE CONFIGURAÇÃO	3.1. Estabelecer ambiente de automação de teste no projeto.	Não Contemp.	Contempla
	3.1. Estabelecer as baselines do projeto de automação de teste.	Não Contemp.	Não Contemp.
	3.3. Acompanhar e controlar as mudanças.	Não Contemp.	Não Contemp.

Fonte: Elaborado pela autora (2017)

6.4.2 Contextualização do Caso 2

O projeto do Caso 2 se iniciou em conjunto com o estudo de caso e, apesar de o tamanho do projeto reportado ser de apenas 24 PFs (**métrica 1)**, o mesmo apresenta complexidade e criticidade alta, dado o seu escopo de integração de diversos sistemas.

A estratégia de automação planejada para este caso incluiu a automação de testes unitários e de integração, com o escopo de 12 PF e 20 PF, respectivamente (**métrica 2**). Desse escopo, 7 PF (**métrica 3**) foram automatizados em nível de teste unitário, ou seja, 58,33% do planejado. Para os testes de integração, 20 PF (**métrica 3)** foram automatizados, representando 60% do escopo previsto.

Foram contabilizadas 180h (**métrica 4**) gastas em atividades de planejamento, projeto, implementação e execução dos testes automatizados, e o total de horas do projeto, nesse mesmo período, foi de 900h e, portanto, a automação foi responsável por 20% do esforço do projeto.

Não foram abertos defeitos formais nesse período, por isso que a **métrica 5** está zerada, dado que o projeto não possui a formalização da gestão de defeitos. Já no contexto da **métrica 6**, similarmente ao Caso 1, os motivos foram os mesmos para não ter coletado o ROI.

Além disso, o diagnóstico realizado após a implantação do FAST mostrou que as áreas de processo de Teste unitário, Teste de integração, Planejamento do projeto de teste, Acompanhamento do projeto de teste e Gerência de configuração tiveram melhoria em suas práticas para inclusão de atividades de automação de teste, conforme apresenta o resumo no Quadro 13 (6).

Quadro 13 (6) – Diagnóstico do FAST Caso 2 pós-estudo de caso (continua)

ÁREA DE PROCESSO	PRÁTICA	INCIAL	FINAL
TESTE UNITÁRIO	1.1. Projetar teste unitário.	Parcialmente	Contempla
	1.2. Implementar teste unitário.	Parcialmente	Contempla
	1.3. Executar teste unitário.	Parcialmente	Contempla
	1.4. Finalizar teste unitário.	Parcialmente	Contempla
TESTE DE INTEGRAÇÃO	2.1. Estabelecer técnicas de teste de integração.	Não Contemp.	Contempla
	2.2. Projetar teste de integração.	Não Contemp.	Contempla
	2.3. Implementar teste de integração.	Não Contemp.	Contempla
	2.4. Executar procedimento de teste de integração.	Não Contemp.	Contempla
PLANEJAMENTO DO PROJETO DE TESTE	1.1. Analisar riscos do produto.	Não Contemp.	Contempla
	1.2. Estabelecer estratégia de automação do teste.	Não Contemp.	Contempla
	1.3. Estabelecer estimativas.	Parcialmente	Contempla
	1.4. Desenvolver o plano do projeto de automação de teste.	Não Contemp.	Contempla
ACOMPANHAMENTO DO PROJETO DE TESTE	2.1. Monitorar o projeto de teste com base no plano.	Não Contemp.	Contempla
	2.2. Gerenciar ações corretivas.	Não Contemp.	Contempla
GERÊNCIA DE CONFIGURAÇÃO	3.1. Estabelecer ambiente de automação de teste no projeto.	Não Contemp.	Contempla
	3.1. Estabelecer as baselines do projeto de automação de teste.	Não Contemp.	Contempla
	3.3. Acompanhar e controlar as mudanças.	Não Contemp.	Contempla

Fonte: Elaborado pela autora (2017)

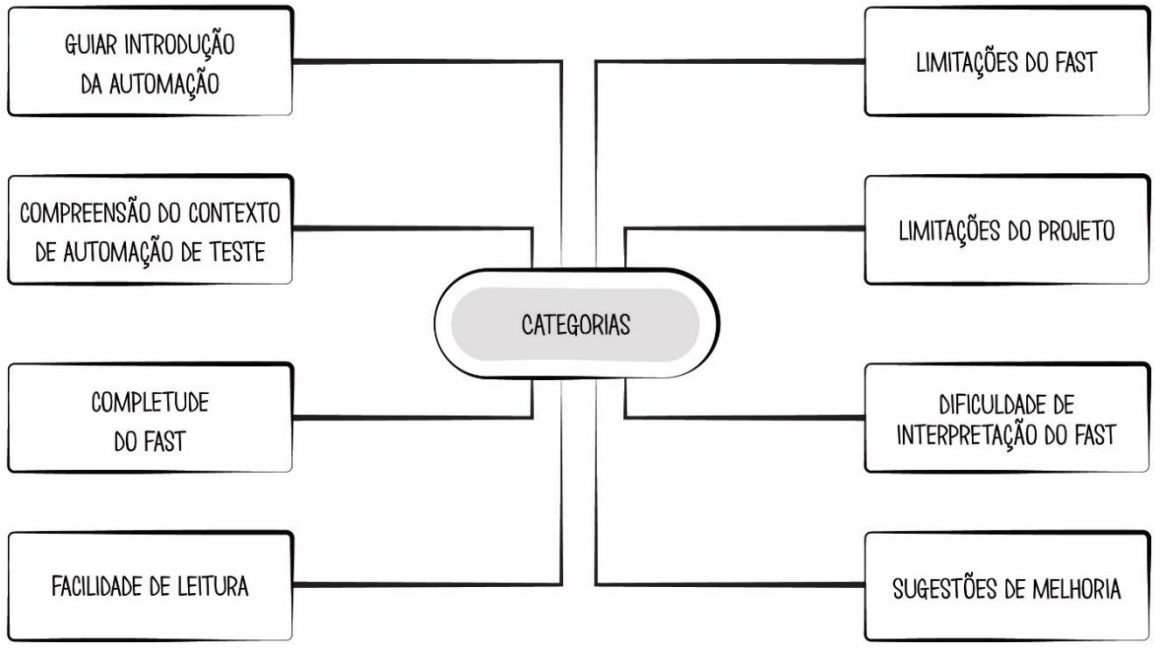
6.4.3 Análise consolidada dos dados das entrevistas

Para analisar as entrevistas, os dados coletados foram codificados e classificados, cujo desafio foi construir categorias que capturassem os padrões recorrentes que permeiam todos os dados originados das entrevistas. Para isso, os áudios foram ouvidos e transcritos na ferramenta Microsoft Excel, por meio das quais, a partir dos códigos gerados da primeira entrevista, categorias foram preestabelecidas para a classificação da entrevista seguinte. Ao se analisar a entrevista seguinte, a partir da codificação dos dados, foi feita a tentativa de encaixá-los em alguma das categorias, para avaliar se havia recorrência na informação. Caso houvesse a necessidade, novas categorias eram criadas. Esse processo de categorização se repetiu para todas as entrevistas, e ao final, uma lista de categorias foi refinada, excluindo as possíveis redundâncias, para a geração do esquema conceitual, conforme representado na Figura 28 (6).

As categorias foram definidas de acordo com os critérios propostos por Merriam (2016), considerando os seguintes aspectos:

- Devem estar alinhadas e responderem às perguntas de pesquisa;
- Devem ser exaustivas, para que haja categorias suficientes e englobar todos os dados relevantes;
- Devem ser mutuamente exclusivas;
- O nome dever ser o mais sensível possível ao contexto que está relacionado; e,
- Devem ser conceitualmente congruentes, em que cada nível de abstração deveria categorizar os conjuntos do mesmo nível.

Figura 28 (6) – Categorização dos dados das entrevistas



Fonte: Elaborada pela autora (2017)

A geração de categorias foi, inicialmente, um processo indutivo, no qual, a partir de fatos particulares, comprovados, chegou-se a uma conclusão genérica (categorias). No entanto, no decorrer do processo de categorização, ao se atingir a saturação dos dados, a partir do ponto que se observou que dados novos não foram encontrados, o método deixou de ser indutivo e passou a ser dedutivo (MARRIAN, 2016). Em contraposição ao método indutivo, o método dedutivo não produz conhecimentos novos, suas conclusões são tiradas com base nos conhecimentos já existentes e que estavam implícitos (MARRIAN, 2016).

A análise desses dados envolveu a consolidação e interpretação do que as pessoas disseram durante a entrevista, com o apoio da ferramenta Microsoft Excel, de forma a dar um sentido completo às respostas encontradas para que as perguntas de pesquisa do estudo de caso pudessem ser respondidas. Dessa forma, optou-se por explicar os achados das entrevistas em torno das categorias geradas, nas quais os trechos citados apresentam partes destacadas em negrito, para representar os códigos que foram gerados e que serão apresentadas a seguir.

6.4.3.1 Guiar a introdução da automação de teste no projeto

Os dados coletados de ambos os Casos mostraram que o FAST apoiou, de forma sistemática, a introdução da automação de teste no projeto. Na área de gestão

da automação, foi feita uma analogia entre o FAST e o PMBOK (2013), como forma de comparar os benefícios do uso deste para a gestão de projetos com benefícios da implantação daquele para a gestão da automação de testes. Além disso, os benefícios de seu uso para sistematizar as atividades de automação de teste poderiam interferir diretamente na melhoria da gestão do desenvolvimento do software como um todo. Algumas respostas que comunicaram essa análise estão presentes a seguir:

> Apoiou sim porque com FAST a gente consegue **ter um norte**, é como se a gente tivesse um projeto genérico de desenvolvimento e a gente colocasse o PMBOK, por exemplo, ele vai dar um norte para que a gente se organize. É isso que o FAST faz com a introdução da automação nas empresas, foi bastante interessante sim. P2C1

> [...] se eu fosse completamente ad-hoc em gestão de projetos, se eu usasse o FAST somente para a área de teste, eu teria uma **área de teste extremante organizada** e por contaminação ele iria organizar a minha gestão de software. P1C2

> A parte ultra positiva do seu framework é a seguinte, é poder dividir o sistema e **fazer um plano de teste** baseado. Deixa eu explicar melhor, a estratégia de teste facilita muito porque não deixa muito espaço para questionamentos. Além disso, você tem uma visão melhor do sistema, você fatia o sistema em complexidade e isso facilitou bastante. E caso a gente não esteja rodando os testes para alguns tipos de funcionalidades, **a gente sabe quais os que estão faltando e sabe onde atacar**. P1C1

Não obstante, foi observado que o FAST apoiou na definição das atividades de automação, organização do ambiente necessário para realizar a automação, como, por exemplo, a definição de tecnologias, ferramentas e estratégia de gerência de configuração e execução do projeto, conforme mostram algumas respostas apresentadas a seguir:

> [...] Como um **guia de boas práticas**, ele deu norte importantes para que eu pudesse vislumbrar o projeto de automação P1C2

> [...] Norte na **definição das atividades**, tipos de teste, como montar um cronograma, como organizar de forma geral, o ambiente. P2C1

> [...] A gente consegue planejar, antever a **organização da automação** até chegar o passo de realmente começar a implementação. P2C1

> [...] ajudou a gente a pensar de forma sistemática. Tipo, com o nível lá de testes unitários, a gente começou a pensar nas **tecnologias adequadas** para tratar nível unitário. Aí sistema, a gente já pensou no Selenium... Formas de implementar teste de sistemas. Ajudou muito. Principalmente para identificar as tecnologias que serão usadas para cada nível. P3C1

> [...] A **gerência de configuração** depois do FAST, melhorou. A questão de poder visualizar os níveis de teste baseado no projeto, depois do framework, melhorou. P1C1

> [...] sim, com certeza, sem ele a gente não teria como se **guiar**. P3C1
>
> [...] pelo que eu vi no FAST junto com a equipe, ela traz um foco para cada artefato ou estudo que a gente faz na própria engenharia de software, **organiza as coisas** que a gente já gera para o projeto, é só trazer uma seção a mais com atenção para o teste. P1C2
>
> [...] ele é bem **exequível**, não teve problemas no modelo do meu time, depois da linha do entendimento, não tive problema de execução do framework. P1C1
>
> [...] Ele é **totalmente viável**. [...] E a gente já está tendo bons resultados. P3C1

6.4.3.2 Compreensão do contexto de automação de teste no projeto.

Essa categoria foi definida para representar que, a partir do FAST, se pode ampliar a compreensão do que deve ser o contexto de um projeto de automação de teste. Essa compreensão se mostrou necessária para apoiar as atividades de planejamento, estimativa de esforço e tomada de decisão no projeto. Além disso, a partir dessa compreensão e uso, o FAST pode levar ao crescimento profissional no âmbito da engenharia de software, e consequentemente, aumento da maturidade da organização, conforme alguns depoimentos apresentados a seguir:

> [...] trouxe a reflexão para o projeto de automação... trouxe a visão **do que é um projeto de automação** associado ao desenvolvimento do projeto. P3C2
>
> [...] o teste unitário, por exemplo, as pessoas não conheciam não tinham feito, e lendo as informações a gente **consegue ter uma noção imediata bastante clara**. P2C1
>
> [...] O FAST ajudou a mostrar que a gente precisa de um determinado esforço para implantar automação... o FAST serviu como uma prova daquilo que eu estava dizendo, **serviu como subsídio**, foi muito interessante. P2C1
>
> [...] de certa forma ele **ajudou na priorização**, trouxe uma atenção para algo que a gente deveria ter pensado. P3C2
>
> [...] sim, apoiam, porque a nível técnico, a nível da engenharia de software em si, **apoiam na evolução dos desenvolvedores**, é como se tivesse aumentando nível de maturidade deles... se você não tiver desenvolvedores pensando nisso, a gente nunca vai evoluir. P1C2

6.4.3.3 Completude do framework

Outra categoria gerada foi a **completude do modelo**, como forma de mostrar que o FAST contempla todo o escopo necessário, no âmbito dos níveis de automação, na perspectiva das áreas de processo e práticas propostas. Além disso, sua adaptabilidade de acordo com contexto, como uma forma de adequar a proposta ao cenário específico, dado que se trata de um framework que pode ser adaptado de acordo com o contexto e necessidades do projeto, conforme apresentam os depoimentos.

> [...] eu não achei **nenhum gap**, parece ser sólido. P1C1
>
> [...] Ele é **completo em sim mesmo**, ele é **autocontido** e não depende de outros. P1C2
>
> [...] eu acho que por ele ser um framework ele é uma **caixa de ferramentas**, então se o implantador acha que tem coisa demais ele pode simplesmente não usar. As opções que ele te dá são justamente aquelas que você precisa para o projeto como um todo. P2C1
>
> [...] O framework é assim, eu te dou uma **coleção de técnicas**, como você executa... sinta-se livre. P1C2
>
> [...] é a **melhor configuração possível**, porque vai dos testes mais básicos aos mais avançados até a entrega do software. Então, se a gente tem que escolher um set de tipos de teste seriam justamente esses que estão presentes no framework. P2C1
>
> [...] hoje eu digo que não senti. Na realidade, eu vi que **contemplava até muita coisa**, que a gente, talvez, por exemplo, indicadores, a gente nem imaginava que pudesse usar isso agora. Acho que está completo. P3C1

6.4.3.4 Facilidade de leitura

A categoria proposta contempla a característica que tem o framework de ser compreensível da forma que ele foi proposto, em que se pode analisar que sua capacidade de compreensão está associada com a experiência profissional de quem o está lendo. Nesse sentido, caso o leitor não tenha conhecimento, experiência profissional e vivência técnica em testes de software, o mesmo terá dificuldades de compreendê-lo, conforme alguns depoimentos apresentados:

> A princípio, não vi problema algum em entender aquilo que é proposto. P3C2
>
> Dado a escrita dele, é simples de ler e de se executar. P1C2
>
> [...] eu consegui fazer esse link porque tenho experiência, o FAST me faz sentido porque eu tenho repertório. P1C2

> A forma com que ele está estruturado é justamente como se precisaria estruturar em qualquer tipo de projeto. Então, da melhor forma possível. P2C1

6.4.3.5 Limitações do FAST

A partir dos dados, essa categoria tem o objetivo de consolidar algumas limitações do FAST, dentre elas uma das relatadas foi a **ausência de boas práticas** para abordar a questão de que o framework deveria trazer sugestões de como uma determinada prática pudesse ser implementada. Abaixo estão listadas partes das entrevistas associadas a esta limitação.

> Sobre a gerencia de configuração, se tivesse uma proposta, ficaria mais fácil para saber como fazer. [...] todos **deveriam ter boas práticas** associadas. [...] isso foi o que eu senti falta. Ele diz, praticamente, o que tem que ter, mas não o como. O como foi o que que doeu mais em mim no uso. P3C1
>
> [...] trazer para cada área, **um exemplo**, associado a alguma prática. P3C2
>
> [...] são, só que **no início parece ser tudo abstrato**. Principalmente os indicadores, se já viessem com um ponta pé inicial. Algo que já vem sendo usado frequentemente. Existem alguns indicadores que são muito usados, que já é comum, já poderia estar lá, entendeu? Como dicas, sugestões, pra pessoa saber como começar. **P3C1**

Nesse contexto, apesar de um participante ter sugerido a inclusão dos indicadores mais usados, o mesmo já está presente no FAST, o que nos faz pensar se essa ausência indicada durante a entrevista é um fato ou se a mesma está relacionada à falta de leitura do material ou à maturidade do profissional e conhecimento na área de teste. Entende-se que pode haver melhorias, mas a documentação proposta já contempla sugestões de boas práticas para sua implantação.

Além disso, **ausência de ferramentas** foi apontada como uma das limitações do modelo, pois o participante tinha e expectativa de que a documentação do FAST viesse atrelada a um conjunto de ferramentas que pudesse apoiar a automação nos níveis de teste proposto, conforme apresenta um trecho da entrevista a seguir:

> [...] o FAST é massa, mas **não contempla ferramentas**. P1C1

No entanto, foi ponderado que o framework não está associado a uma tecnologia específica e que a inclusão de ferramentas em sua proposta iria limitá-lo, e

que o uso de ferramentas é uma decisão diretamente associada às tecnologias específicas do projeto e, a partir de então, o participante compreendeu e ficou de acordo com essa visão.

Outra limitação concretizada a partir das entrevistas está associada à **ausência de modelos de documentos** (*templates*) na proposta do framework, conforme apresentam os trechos das entrevistas a seguir:

> Seria interessante trazer uns **templates** e algumas informações dos templates, poucas informações básicas para a gente ter uma ideia [...] porque muitas vezes a gente vê o que está escrito, mas a gente tem que imaginar como implementar aquilo, como abstrair. Se os templates estivessem prontos, com poucas informações, seria de bastante ajuda. P2C1
>
> **Ausência de modelos**, por exemplo, plano de teste, ao meu ver tem que ter um template. P1C1

Apesar de apresentado como limitação, essa categoria trouxe uma controvérsia na qual foi apontado que a inclusão dos mesmos poderia limitar a criatividade de implantação do FAST e que essa necessidade está diretamente associada à maturidade profissional, haja vista que se o mesmo tem uma bagagem conceitual sobre o tema, a demanda por modelos pode não ser significante. Além disso, apesar de sentir a ausência dos mesmos, foi entendido que sua definição não faz parte do contexto do framework, conforme apresentam alguns trechos das entrevistas abaixo.

> [...] Eu acho que templates... Você **não deve engessar tanto**, você deve ter abertura para criar esses templates. [...] eu prefiro ter a conceituação bem feita para que eu me sinta livre para fazer meus próprios documentos e personalizar do jeito que eu quiser. [...] A necesidade de template está alinhada com a maturidade profissional. P1C2
>
> A única problemática que eu encontrei, que é uma coisa até normal, foi que como ele é abrangente, **alguns documentos**, eu precisei procurar na internet como modelos. Mas isso **não é tanta responsabilidade do framework.** P2C1

6.4.3.6 Limitações do projeto

A categoria de limitações do projeto foi definida para contemplar tópicos que dificultaram a implantação da automação de teste, mas que estão associados ao contexto do projeto. O primeiro aspecto observado foi a **imaturidade da equipe** e como isso influenciou nos resultados obtidos no projeto de automação. Além disso, a

partir da falta de conhecimento na área, observou-se que havia a necessidade de um treinamento sobre conceitos de teste, para que o framework pudesse ser mais bem compreendido, fato que está diretamente associado à imaturidade do profissional na área de conhecimento, conforme apresentam alguns trechos das entrevistas abaixo:

> [...] ficaram dúvidas porque a gente não tem costume de fazer teste. A equipe ainda não estava habituada a isso. [...] a equipe ainda **não está 100% madura** para tocar o projeto de automação. P3C1
>
> Em termos de didática, eu sugeriria uma **explanação anterior sobre o que é teste** e das áreas antes de implantar o FAST. P1C1
>
> [...] se a equipe **tivesse um conhecimento maior**, teria sido bem melhor a implantação. P3C2

A **falta de apoio da organização** foi indicada como uma das limitações para o projeto, a partir da **falta de priorização**, de **disponibilidade de tempo** e consequente **alocação de pessoas**, para que as atividades pudessem ser realizadas, tanto relativas às áreas de processo de Suporte como da área Técnica, conforme partes das entrevistas apresentadas a seguir:

> [...] ele (indicadores) é muito importante, não tem como sair [...] **cortamos por conta de tempo**. P3C1
>
> [...] a visão da organização, a visão de que **tem que ter pessoas focadas** nisso (testes). P3C1
>
> A organização tem que ter a consciência de que **tem que reservar** umas pessoas, ou 80% delas, pra trabalhar com testes. P3C1
>
> Do FAST em si não encontrei coisas negativas, ao meu ver. Coisas negativas que eu encontrei, as **dificuldades**, foi mais **relacionada com a cultura da empresa**. Por exemplo, eu começava uma atividade de teste e aparecia uma de desenvolvimento, eu tinha que parar o teste para fazer o desenvolvimento. P2C1
>
> Automação não é algo que se começa hoje e se está pronto ao final do dia. **Precisa de esforço, custo**, tanto financeiro como esforço da equipe. P2C1
>
> Sincronismo ente o projeto de automação e a equipe. O deadline do projeto **não estava incluindo o esforço necessário** para implantar o projeto de automação. P3C2
>
> [...] e a parte de testes era algo que, de certa forma vai trazer qualidade para o produto, só que ela foi feita de forma não tão sistemática como o framework propõe. Porque de certa forma **a prioridade foi baixa** comparada com outros fatores. P3C2
>
> [...] sincronismo ente o projeto de automação e a equipe. O deadline do projeto **não estava incluindo o esforço necessário** para implantar o projeto de automação. P3C2

Outra limitação está relacionada **à adequação do processo de desenvolvimento** adotado pelo projeto, que precisa ser adaptável para que o ciclo

de vida de testes possa fazer parte do mesmo, conforme apresentado na citação a seguir:

> [...] A metodologia de desenvolvimento **tem que se adequar** ao uso de teste. P1C2

6.4.3.7 Dificuldade de interpretação do FAST

Foram reportadas **dificuldades de interpretação** do framework e sua e abstração para que pudesse ser aplicado ao contexto do projeto, no qual se pôde observar que tal dificuldade está diretamente relacionada com o perfil do profissional e conhecimento técnico da área, conforme apresentam trechos abaixo:

> No começo, **ficou muito abstrato**, ficou confuso. Até porque a gente pediu alguns modelos de documentos porque a gente não sabia fazer. A gente viu o framework como é que tinha que ser, mas [...] tinha um **gap muito grande** entre o framework e a prática. P3C1
>
> [...] **overhead inicial do entendimento do framework**, isto é, muito tempo investido, foram 2 pessoas trabalhando bastante até que entendesse e assimilasse. P1C1
>
> O FAST subentendeu que a pessoa que estava lendo já tem um **conhecimento avançado** na área. P1C1
>
> [...] tive que passar um pouco desse entendimento para a equipe de desenvolvimento[...] com relação a este entendimento, níveis de teste, propósitos de cada nível, os responsáveis por cada nível, eu tive que passar um pouco do que eu domino, do que eu sabia para a equipe. Para quem não tem bagagem de teste, **talvez tivesse que ser feito um nivelamento**. P3C2

Não obstante, também foi mencionada a dificuldade de entendimento da relação entre a Área Técnica com a Área de Suporte, conforme apresentado a seguir:

> [...] não sabia como integrar relacionar a área de suporte com a cada nível de teste. P3C1
>
> [...] fazer o link entre área técnica e área de suporte não é muito fácil não. Ambas apoiam, sendo que esse link, de forma abstrata, no meu entendimento, é difícil ligar uma coisa com a outra. P1C1

6.4.3.8 Sugestões de melhoria

A entrevista consolidou dados que contemplam sugestões de melhoria ao framework proposto, como a definição de áreas de processo obrigatórias e opcionais,

assim como a inclusão de uma área de processo específica para a gestão de mudanças, conforme apresentado a seguir:

> Eu acho até que ele deveria ter **áreas opcionais e áreas obrigatórias**. Deveria ter um conjunto mínimo necessário, de certa forma exigir mais, para cobrar a equipe de desenvolvimento para pensar nisso. Se deixar tudo muito flexível, uma área pode depender de outra. P3C2
>
> [...] Uma área específica para gestão de mudanças, abordar gestão de mudanças de forma separada. P1C2

6.5 Reporte dos resultados

Com base na análise consolidada a partir do contexto dos 2 estudos de caso, pode-se avaliar como as perguntas de pesquisa seriam respondidas a partir dos dados coletados. Para isso, optou-se por relacioná-las com as categorias geradas na análise dos dados, nas quais se chegou ao relacionamento entre elas conforme proposto no Quadro 14 (6).

Quadro 14 (6) – Relacionamento entre perguntas de pesquisa e categorias (continua)

PERGUNTA DE PESQUISA	CATEGORIA
P1. O FAST apoiou a introdução sistemática de automação de teste no projeto?	Guiar a introdução da automação de teste no projeto
	Compreensão do contexto de automação de teste no projeto
P2. Quais aspectos positivos (fatores de sucesso) do uso do FAST para introdução da automação de teste no projeto?	Completude do FAST
	Facilidade de leitura
P3. Quais aspectos negativos (fatores de insucesso) decorrentes do uso do FAST na implantação de automação de teste no projeto?	Dificuldade de interpretação do FAST
	Sugestões de melhoria
P4. Quais as limitações para introdução da automação de teste no projeto a partir do uso do FAST	Limitações do FAST
	Limitações do projeto

Fonte: Elaborado pela autora (2017)

No contexto da pergunta **P1**, observou-se que o FAST apoiou na introdução sistemática da automação de teste, tanto por meio da análise quantitativa como da análise qualitativa. Os dados das métricas 3 (tamanho realizado da automação de teste no projeto) e 4 (esforço da automação de teste no projeto) mostram que houve uma introdução de práticas de automação de teste, que pode ser reforçada com os dados do diagnóstico realizado no fechamento do estudo de caso, conforme

apresentado no Quadro 12 (6) e Quadro 13 (6). Além disso, o uso do FAST trouxe benefícios na gestão da automação, definição sistemática das atividades, seleção de tecnologias, planejamento do teste, fatos que puderam ser observados a partir das entrevistas. Notou-se que o FAST introduziu automação de forma que o projeto pudesse compreender o que compõe um projeto de automação de teste e analisar um conjunto de variáveis para que sua implantação pudesse se concretizar, o que foi possível a partir das definições contidas no framework.

Em relação a **P2**, observou-se que a completude do framework foi analisada como um aspecto positivo em que o FAST foi comparado a uma caixa de ferramentas que pode ser instanciada de acordo com o contexto do projeto. Além disso, observou-se que a categoria facilidade de leitura também foi abordada como um de seus aspectos positivos. Apesar de esse ponto ser contraditório, pode-se notar que a facilidade de compreensão do mesmo está diretamente associada às habilidades de compreensão técnica sobre testes da pessoa que vai utilizá-lo. Nesse contexto, o FAST mostrou-se facilmente compreensível por profissional que tem maior experiência profissional não apenas na área de testes, mas em desenvolvimento de software por completo.

No contexto da **P3**, a dificuldade de interpretação do framework foi apontada como uma limitação do FAST para as pessoas que não têm familiaridade com teste de software ou maturidade técnica profissional, em que foi sugerido que houvesse um nivelamento inicial antes do uso do framework. Nesse cenário, a sua estrutura e compreensão de como o mesmo pode ser aplicado na prática não ficaram claras e objetivas para todos os participantes. Dentre as sugestões de melhoria consolidadas, categoria que foi associada como resposta a P3, por ser considerada como aspectos que o mesmo poderia contemplar, estão a sugestão de um escopo mínimo obrigatório a ser seguido na implantação do FAST, assim como a inclusão de uma área de processo específica para a gestão de mudanças.

No âmbito da **P4,** foram levantas limitações tanto do FAST como do projeto como fatores limitantes da introdução de automação. Dentre as limitações do FAST, foram identificados aspectos como ausência de boas práticas, ausência de ferramentas pré-definidas, assim como a ausência de templates. No contexto das boas práticas, se pôde observar que existiu a necessidade de identificar quais eram as boas práticas para que uma determinada subprática pudesse ser implementada,

ou seja, como transformar o framework em um processo exequível a um determinado contexto. Essa necessidade está diretamente associada ao contexto dos projetos e a incorporação dessa demanda ao FAST descaracterizaria seu conceito de ser um framework, pois o tornaria um processo específico para um determinado contexto. O mesmo cenário se aplica à limitação da descrição de ferramentas no contexto do framework, pois o objetivo deste é não estar limitado a um conjunto de características técnicas do contexto de um projeto. No âmbito da ausência de templates, pode-se associar essa necessidade ao perfil do profissional, visto que a definição de templates no contexto do FAST também foi apontada como uma forma de limitar a criatividade de quem o vai instanciar no projeto.

Dentre as limitações do projeto, foram identificados itens como imaturidade profissional, falta de apoio da organização, falta de priorização para as atividades de teste, indisponibilidade de tempo e falta alocação de pessoas para que as atividades possam ser realizadas. Para completar a análise desse aspecto, vale refletir sobre o percentual de esforço dedicado para automação em ambos os Casos, nos quais C1 e C2 apresentaram taxa de 11,14% e 20% de esforço dedicado à automação de teste, respectivamente. Comparando a situação anterior ao estudo de caso, em que não havia automação sistemática de teste, observa-se que houve um incremento na dedicação do time, no entanto, ainda não foi o suficiente para que toda a estratégia de automação pudesse ser contemplada. A dedicação de tempo está diretamente relacionada com o apoio da alta gestão, para que possa haver priorização das atividades no projeto para que os reais benefícios da automação possam ser observados.

Além disso, a adequação do processo existente no projeto às atividades de automação foi levantada como uma possível limitação às atividades definidas a partir do uso do FAST, cujo ciclo de vida do projeto precisa ser adaptável para que as atividades de automação possam ser incluídas.

Apesar de os dois Casos terem sido realizados em ambiente distintos, pôde-se observar que havia um alinhamento e congruência das respostas obtidas durante as entrevistas. Ou seja, os achados não variam muito apesar dos distintos contextos analisados. Em ambos os Casos, as equipes eram heterogêneas em termos de perfil e experiência, assim como do estado inicial de práticas de automação ser praticamente nulo no início do período de observação. Os dados obtidos também

mostraram que houve dificuldades para sistematizar a automação, haja vista que além da inclusão das práticas propostas também era necessária uma mudança de cultura organizacional.

Diante dos dados decorrentes do estudo de caso, algumas limitações e ameaças à validade foram levantadas e serão analisadas na Seção seguinte.

6.6 Limitações e ameaças à validade

A realização de pesquisas sobre questões do mundo real implica uma troca entre o nível de controle e o grau de realismo, cuja situação realista é muitas vezes complexa e não determinista, o que dificulta a compreensão do que está acontecendo, especialmente para estudos com propósitos explicativos (RUNESON e HÖST, 2008). Esse cenário é característico do estudo de caso, no qual as limitações são existentes e precisam de uma análise consolidada para minimizar seus impactos nos resultados obtidos.

Dentre as limitações do estudo de caso, pode-se mencionar que em nenhum dos Casos o FAST foi implantado por completo, o que limita a análise de seus resultados a partir das experiências práticas apenas das áreas de processo de Teste Unitário, Teste de Integração, Planejamento do Projeto de Teste, Acompanhamento do Projeto de Teste e Gerência de Configuração. Essa limitação, por sua vez, teve que ser aceita dentro da pesquisa, haja vista que a pesquisadora não pôde interferir no projeto para que o framework pudesse ser completamente implantado.

Além disso, não se pôde garantir que os participantes leram toda a documentação do FAST, em que se observou que o foco das respostas foi com base nas experiências práticas com escopo restrito às áreas de processo implantadas. Apesar de ter conhecimento desse fato, as perguntas, no decorrer da entrevista, não se limitaram e foram abrangentes para todo o escopo do FAST.

Outra limitação desse estudo trata do fato de que a implantação do FAST foi realizada em 2 Casos nos quais os processos de desenvolvimento de software já existiam. Apesar de sua proposta não ser limitada a esse escopo, não se pode concluir como seria a implantação do mesmo em um ambiente completamente *ad-hoc*.

De acordo com Easterbrook et al. (2008), a maior limitação dos estudos de caso é que a coleta e análise de dados estão mais abertas à interpretação e ao viés

do pesquisador, e para minimizar essa ameaça à validade, é necessária uma estrutura explícita para selecionar e coletar dados, que foram realizados por meio de um protocolo criteriosamente detalhado e seguido ao longo do estudo.

Outra ameaça levantada foi a possibilidade de viés para análise de dados em um domínio específico de desenvolvimento de software e, para que a mesma pudesse ser minimizada nesse estudo foram selecionados 2 distintos cenários, em que a ausência de determinada informação em um pode ser complementada pela existência da mesma no outro.

Não obstante, a ameaça em relação à participação do pesquisador no contexto de apoio à implantação do FAST pode ter influenciado os resultados obtidos, assim como pode ter constrangido os participantes das entrevistas na hora de fornecer suas respostas.

No entanto, de acordo com Merriam e Tisdell (2016), o que torna os estudos experimentais confiáveis é o seu cuidadoso projeto, a partir da aplicação de padrões bem desenvolvidos e aceitos pela comunidade científica, fato que pode ser observado a partir das informações detalhadamente descritas neste capítulo.

6.7 Considerações finais

Este capítulo apresentou desde o planejamento até a coleta e análise dos dados quantitativos, decorrente das métricas, e qualitativos, a partir das entrevistas, do estudo de caso em 2 distintos contextos.

A partir dos dados, foi feito um relacionamento de como os mesmos respondem as perguntas do estudo, assim como suas limitações e ameaças à validade. O próximo capítulo irá apresentar as conclusões e fechamento do trabalho.

7 GRUPO FOCAL

O objetivo deste capítulo é apresentar o planejamento, execução e análise dos resultados provenientes do grupo focal, além de ressaltar as considerações relativas ao uso método em relação à proposta desta pesquisa, destacando também suas restrições e ameaças à validade.

7.1 Planejamento

O problema de pesquisa no âmbito da realização do grupo focal está associado com o problema central deste trabalho, que seria como a automação pode ser introduzida no contexto do desenvolvimento de software. A partir dessa pergunta, o objetivo do grupo focal é avaliar o FAST, sob os aspectos de **completude, clareza e adequação** da estrutura proposta.

Para que esse objetivo fosse alcançado, planejou-se o roteiro por meio das 5 atividades descritas no Quadro 15 (7).

Quadro 15 (7)- Roteiro para realização do grupo focal (continua)

Atividade	Duração	Passos
1	10 min	Apresentação dos objetivos para realização do grupo focal; Apresentação da visão geral da proposta
2	15 min	O moderador apresentou a descrição dos **níveis de automação**, e em seguida, iniciou a seguinte discussão sobre os níveis: • A descrição dos níveis está clara? • A quantidade de nível é suficiente? • É necessário adicionar ou remover algum nível?
3	15 min	O moderador apresentou o título e o objetivo das **duas áreas**, e em seguida, iniciou a seguinte discussão sobre as áreas: • O título da área está claro? • O objetivo da área está claro? • É necessário adicionar ou remover alguma área? • A relação entre as áreas de processo está clara?
4	15 min	O moderador apresentou o título e o objetivo das **quatro áreas de processo da área técnica** e em seguida, iniciou a seguinte discussão: • O título da área de processo está claro? • O propósito da área de processo está claro? • É necessário adicionar ou remover alguma prática? • É necessário adicionar ou remover algum produto de trabalho?

Quadro 15 (7)- Roteiro para realização do grupo focal (continuação)

5	15 min	O moderador apresentou o título e o objetivo das **seis áreas de processo da área de suporte** e em seguida, iniciou a seguinte discussão: • O título da área de processo está claro? • O propósito da área de processo está claro? • É necessário adicionar ou remover alguma prática? • É necessário adicionar ou remover algum produto de trabalho?
	Tempo total:	70 min

Fonte: Elaborado pela autora (2017)

Os especialistas foram selecionados a partir do perfil e conhecimento em automação de teste, e, apesar de ter convidado 7 participantes, apenas 4 compareceram à reunião. O Quadro 16 (7) descreve o perfil dos participantes, cujos nomes foram preservados por questões de confidencialidade.

Quadro 16 (7)- Especialistas selecionados para o grupo focal (continua)

Especialista 1 (E1)	
Nível de experiência	9,5 anos de experiência prática em teste de software
Dados demográficos	Idade: 32 anos. Gênero masculino. Mestrado em Redes de Computadores. Graduado em Ciência da Computação. Função: Desenvolvedor
Interesses individuais	Interesse em pesquisas relacionadas à teste de software com intenção de entrar no programa de doutorado na área.
Habilidades técnicas	Java, C++, hibernate, android, php, J2ME, android
Especialista 2 (E2)	
Nível de experiência	12 anos de experiência práticas de engenharia de software (desenvolvimento, gerência, qualidade, devops)
Dados demográficos	Idade: 34 anos. Gênero masculino. Graduado em Ciência da Computação e Engenharia de Produção. Função: Engenheiro de Software
Interesses individuais	Interesse em pesquisas relacionadas à engenharia de software
Habilidades técnicas	Java, Ruby, PHP, Mobile, Web
Especialista 3 (E3)	
Nível de experiência	10 anos de experiência prática em teste de software.
Dados demográficos	Idade: 33 anos. Gênero feminino. Graduado Sistema de Informação. Função: Engenheiro de teste.
Interesses individuais	Interesse na área de automação de teste e teste de performance.
Habilidades técnicas	Java, JUnit, Android, JMeter, Python.

Quadro 16 (7)- Especialistas selecionados para o grupo focal (continuação)

Especialista 4 (E4)	
Nível de experiência	5 anos de experiência prática em teste de software
Dados demográficos	Idade: 37 anos. Gênero feminino. Mestrado profissional em engenharia de software. Graduado em Análise e desenvolvimento de sistemas. Função: Testadora
Interesses individuais	Interesse em pesquisas relacionadas à qualidade de software, agilidade em qualidade, automação de testes, estratégias ágeis de testes, entrega contínua, devops.
Habilidades técnicas	Java, Javascript, C++, android, Native Script, React, Cucumber, Selenium Webdriver, Docker, Jenkins

Fonte: Elaborado pela autora (2017)

7.2 Execução

O grupo focal ocorreu no dia 19 de setembro de 2017, com duração aproximada de 90 minutos, sendo registrado em áudio, mediante a permissão dos participantes, além das anotações em papel realizados pelo moderador, sempre que julgado necessário.

A seção iniciou a partir da apresentação do método grupo focal, indicando os objetivos de sua aplicação no contexto desta pesquisa para os participantes, explicitando que o objetivo deste método era coletar o maior número de informações oriundas a partir da interação entre os membros do grupo (REED e PAYTON, 1997). Nesse momento, foi apresentado o roteiro previsto para aquela sessão, conforme planejamento descrito no Quadro 15 (6). Após o início, foi solicitado aos participantes que os mesmos se sentissem à vontade para contribuir com opiniões pessoas e ideias sobre os temas abordados.

Apresentou-se a visão geral do FAST, mostrando os componentes do framework, áreas, áreas de processo.

Diante desse cenário, os dados foram coletados, analisados e consolidados a partir dos tópicos planejados, cujas análises serão descritas na próxima Seção.

7.3 Análise dos dados

Para a análise dos dados, foi realizada uma análise e comparação das discussões em torno dos tópicos apresentados ao longo do grupo focal, para

identificar as semelhanças e obter entendimento de como as diversas variáveis se comportam no contexto da pesquisa (KITZINGER, 1995). Para isso, os dados qualitativos foram analisados a partir da transcrição dos áudios, que foi realizada a partir da ferramenta Microsoft Excel.

Diante deste contexto, a seguir, serão apresentados os dados consolidados estruturados em tópicos a partir de cada uma das atividades planejadas para o grupo focal, conforme anteriormente apresentado no Quatro 15 (7).

7.3.1 Níveis de automação

Em relação aos **níveis de automação**, após apresentação de sua definição, conceitos e objetivos, todos os especialistas indicaram que sua descrição estava clara. Além disso, a quantidade de níveis de automação foi considerada adequada, sem necessidade de adição ou remoção de qualquer nível. No entanto, no decorrer da discussão, um especialista questionou sobre a aplicabilidade do nível de aceitação, haja vista que o mesmo deveria ser realizado pelo cliente. A partir desta consideração, outro especialista indicou que a realização do teste de aceitação é uma consequência dos testes de sistemas no escopo do ambiente do cliente. Algumas respostas que comunicaram essa análise estão presentes a seguir:

> [...] Os níveis refletem muito a pirâmide de teste. Para mim, nos projetos que eu já trabalhei, implementei os níveis exatamente dessa forma. E1
>
> [...] Eu acho que esses níveis **apoiam a implantação da automação**. São os níveis mais utilizados, os que mais fazem sentido. Eu trabalho, quase sempre, com esses 4 níveis. Principalmente para automação. E3
>
> [...] Eu acho que ajudam a organizar a estratégia porque a gente porque separando em níveis fica mais fácil de entender. E3
>
> [...] Eu gostei da divisão por níveis, acho que facilita o entendimento. E2

7.3.2 Áreas de processo da área de suporte

Para a análise da área de suporte, a área de processo **Requisitos** apresentou dúvidas no início da discussão, em que foi questionado se não seria melhor que essa área de processo fosse tratada como parte do planejamento do projeto de automação

de teste. Além disso, houve questionamentos sobre a diferença entre os requisitos da automação e os do projeto e como isso é abordado na proposta. Durante a própria discussão do grupo, foi compreendido que os requisitos do projeto podem servir como base para os requisitos da automação. Foi também questionado se a proposta aborda técnicas para levantamento de requisitos de automação para projetos que contemplam sistemas legados, em que se observou que a sugestão poderia ser incluída como um ponto de melhoria para o FAST. A seguir, alguns comentários relativos à área de processo de requisitos são apresentados:

> Nesse caso, a engenharia reversa seria uma das técnicas de elicitação de requisitos da automação nos sistemas legados. E3
>
> A engenharia reversa pode ser usada para entender o sistema, mesmo que seja para testá-lo novamente. E2

No escopo da área de processo de **Acompanhamento** da automação de teste, foi questionado como o framework aborda o acompanhamento do projeto de automação, para que esta prática possa ser realizada de forma objetiva. Após longo debate, foi sugerida a definição e acompanhamento de indicadores relativos à automação, cuja área de processo está presente na descrição do FAST. A fala do especialista, transcrita a seguir, mostra esse questionamento:

> Existe algum indicativo para dizer que você chegou ao objetivo proposto pelo modelo? [...] Existe alguma técnica para medir se o projeto está tendo sucesso com a automação? E1

Em relação à área de processo de **Gestão de defeitos**, foi comentado como a proposta aborda uma forma de reportar os resultados da automação, em foram discutidas as práticas e subpráticas existentes nesta área de processo. A partir da discussão e interação entre os membros, foi sugerida a inclusão, na descrição do framekwork, de uma sugestão para padrão que possa apoiar a reportagem de defeitos no projeto, conforme comentários listados a seguir:

> Você abordou analisar e reportar os resultados? Você chegou a detalhar como deve ser reportado os resultados da automação? [...] Nessa parte de reportar defeitos, a linguagem do teste de aceitação, para o cliente é importante existir um padrão de reportar os defeitos. [...] A padronização dá uma maior confiabilidade ao projeto. E3

Além disso, foi questionado sobre **a adaptação e instanciação do framework** e de seus processos em ambientes que já existem processo e/ou práticas ágeis. Também foi questionado se a proposta contemplava a definição de um processo para implantação de automação mediante uso do FAST, e após debate entre

o grupo, foi compreendida a diferença entre framework proposto e um processo de automação, o que poderia ser um trabalho futuro a ser pesquisado, conforme comentários trazidos a seguir:

> Como seria a aplicação do FAST em um ambiente ágil? E4
>
> Como é a adaptação dele em organizações que já possuem processos. Perguntou se o framework aborda isso. E4

7.3.3 Áreas do FAST

No que se refere às **Área Técnica e de Suporte**, os participantes consideram que os títulos e objetivos estavam adequados a sua proposta, e que a forma de organização ficou coerente com a estratégia proposta. Observou-se que, inicialmente, os membros tiveram dificuldade de compreender a relação entre as áreas, como também foi demonstrada uma dificuldade de compreensão da Figura 22 (5) e como as áreas de processo se relacionavam entre si.

7.3.4 Aspectos gerais

Neste contexto, questionou-se sobre quais seriam as limitações do uso do framework na prática, momento em que uma das especialistas comentou que não vê limitações no framework, pois o mesmo aborda todas as práticas para automação de teste, mas que as limitações são decorrentes das priorizações que os projetos tratam, ou seja, a introdução da automação nem sempre é realizada, porque a prática não é priorizado no escopo do projeto. Ainda neste contexto, foi mencionado que a ausência de um processo contemplando o passo a passo de como usar o framework pode ser uma limitação para o seu uso. A partir do passo a passo definido, o especialista sugeriu definir uma métrica para avaliar qual a aderência do FAST ao projeto. Apesar de não estar no processo, essa métrica é analisada quando ser faz o diagnóstico prévio a sua implantação.

> Não vejo limitação do framework, porque ele é bem abrangente. Eu vejo limitação do cliente, que pode comprometer a eficácia do framework. E4.
>
> Eu acho que o framework está completo, porque contempla todas as boas práticas que eu visualizo na prática. E3
>
> Acredito que ele dá muito apoio para introduzir automação. E2

> Eu acho que ficou faltando o passo a passo de como implantá-lo. E1
>
> Senti falta de ter um indicador para medir o quão FAST o meu projeto é. E1

Sobre a **forma de apresentação das áreas de processo**, observada na Figura 22 (5), um dos especialistas indicou que a mesma, pelo fato de contemplar as linhas, dá uma ideia de que as áreas de processo são sequenciais.

> A imagem apresentada deu a entender que os processos são sequenciais. Na figura, parece que os processos são sequenciais. E1

Ainda nesse contexto, um especialista sugeriu a inclusão de uma área de processo específica para **análise de viabilidade** do projeto de automação. Foi mencionado que o FAST pode ser utilizado como um guia para elaboração de um projeto de automação de teste.

> Usar o FAST para realizar pré-vendas de projeto de automação seria ótimo, porque muitas vezes o cliente não sabe nem o que é automação. E3

7.4 Limitações e ameaças à validade

Dentre as limitações de aplicação do grupo focal, destaca-se o fato de que participantes mais experientes podem intimidar a contribuição dos demais no momento de projetar sua opinião sobre um determinado tópico. Como forma de mitigar essa limitação, foi selecionado um grupo homogêneo, em termos de experiências em automação de teste.

Outra restrição deste método está associada à compreensão limitada, por parte dos especialistas, do que está sendo discutido, haja vista que existe um limite de tempo para realização do método. Nesse contexto, questões muito complexas podem ser mal interpretadas, o que pode levar o grupo a ter dificuldades em convergir ao tema central discutido. Como forma de reduzir essa ameaça, o moderador atuou de modo a realizar uma explicação e apresentação prévia do tópico a ser discutido, para que os especialistas pudessem ter melhor compreensão do tópico.

Uma das ameaças à validade da aplicação do grupo focal foi que nem todos os participantes conseguiram compreender a aplicabilidade do framework, haja vista que a sessão tem uma limitação de tempo e o participante precisa abstrair a compreensão para a forma de implantação prática da proposta. Nesse sentido, dado que o perfil dos participantes selecionado estava focado em profissionais com

experiência prática de automação, e não do uso e implantação de melhoria de processos no contexto de projetos, foi identificada uma limitação de sua avaliação.

Outra ameaça identificada está no fato de que a autora da proposta foi a mediadora do grupo focal, motivo pelo qual os participantes poderiam estar comedidos em seus comentários no momento de avaliação da proposta. Para minimizar esta ameaça, buscou-se a seleção de participantes que tivessem maior autonomia e independência em relação à autora e que não tivessem laços de amizade que interferissem nos comentários a serem proferidos durante a sessão.

7.5 Considerações finais

A realização do grupo focal foi conduzida com objetivo de associar os resultados obtidos do estudo de caso com a visão de especialistas na área para responder as perguntas de pesquisa desta tese. O método foi realizado com um grupo de participantes homogêneos, em termos de idade e nível de experiência profissional, contando com a presença de quatro participantes, apesar de o convite ter sido feito para seis especialistas.

A partir do objetivo do grupo focal, de avaliar o FAST sob os aspectos de **completude, clareza e adequação** da estrutura proposta, pôde-se observar que os especialistas o analisaram como uma estratégia adequada para introdução de automação de testes no projeto. Além disso, a partir das discussões geradas pela execução do grupo focal, conforme análise detalhada apresentadas nas seções anteriores, se puderam consolidar algumas sugestões de melhoria para o FAST, tais como:

a) Inclusão das práticas para realização de análise de viabilidade da automação de testes para que a mesma possa ser realizada previamente à implantação da automação em um contexto de projeto;
b) Adição de recomendações e técnicas que possam apoiar a escrita adequada dos defeitos e resultados decorrentes do processo de automação; e,
c) Definição de um processo ou guias contemplando uma estratégia para implantação da automação com o uso do FAST no contexto de um projeto.

Com base nos relatos apresentados, pôde-se constatar a relevância da pesquisa e que há indícios de sua validade, além da originalidade da proposta, conforme discutido no capítulo dos trabalhos relacionados.

8 CONCLUSÃO

O objetivo deste capítulo é apresentar as considerações finais e contribuições decorrentes dessa pesquisa assim como seus trabalhos futuros associados.

8.1 Considerações finais e contribuições

Os estudos mostram que teste de software é uma disciplina da engenharia de software que está em constante crescimento, cuja demanda de pesquisa cresce a partir da busca constante por melhores práticas, com objetivo de prover visibilidade a respeito da qualidade do produto para que o mesmo seja aprimorado.

Nesse âmbito, o teste aparece por meio de uma perspectiva para fornecer uma visão mais objetiva sobre a qualidade do produto, na qual a busca por melhores práticas para realizá-lo vem do foco de pesquisas encontradas na literatura, assim como das demandas observadas no mercado. É nesse contexto em que a introdução da automação de teste surgiu como uma demanda de sistematizar os seus processos para que os reais benefícios sejam alcançados.

Diversas abordagens de melhoria de processo de teste também surgiram na literatura, tanto como fazendo parte de modelos de maturidade para teste de software como a partir de abordagens específicas para a automação. No entanto, observou-se que nenhuma delas fornecia subsídios práticos para que a introdução de práticas de automação de teste pudesse acontecer de forma sistemática no contexto de projeto de desenvolvimento de software.

A partir desse cenário, essa pesquisa foi desenvolvida, e contou com uma metodologia baseada em entrevista empírica, desenvolvimento do FAST e sua avaliação por meio de estudo de caso e grupo focal.

O planejamento do estudo de caso foi realizado para que informações pudessem ser coletadas ao longo de sua execução. A seleção dos Casos foi realizada de modo a fornecer perfis complementares, onde o ambiente de uma empresa privada e uma instituição pública são cenários distintos e que trouxeram resultados valiosos para as demandas desta pesquisa.

A partir dos dados provenientes do estudo de caso e do grupo focal, qualitativos e quantitativos, os mesmos foram analisados e organizados a partir das seguintes categorias:

- Guiar introdução da automação;
- Compreensão do contexto de automação;
- Completude do framework;
- Facilidade de leitura;
- Limitações de FAST;
- Limitações do projeto;
- Dificuldade de interpretação do FAST; e,
- Sugestões de melhorias.

Diante dos dados provenientes do estudo de caso e grupo focal, se pôde fazer uma associação para responder à pergunta de pesquisa deste trabalho, que é:

Como a automação de teste deve ser introduzida e mantida no processo de desenvolvimento software?

Nesse contexto, a automação de teste deve ser introduzida de **maneira sistemática** no contexto do processo de desenvolvimento de software, no qual o **FAST** se mostrou como **uma estratégia viável** para que a introdução e manutenção fossem realizadas. Diante dessa visão, a principal contribuição deste trabalho é uma solução para sistematizar os processos de automação de teste, a partir de um framework que contempla aspectos técnicos e de suporte, os quais são essenciais para a introdução sistemática da automação de teste.

A partir do objetivo geral do trabalho, que foca em uma estratégia para introdução sistemática de práticas de automação de teste no contexto de projeto de desenvolvimento de software, observou-se que o FAST pode ser considerado como uma das estratégias decorrentes deste objetivo. Além disso, alguns fatores de insucesso da implantação de automação puderam ser observados, tanto a partir do conhecimento adquirido a partir da revisão da literatura, como a partir da realização do estudo de caso e grupo focal.

Este trabalho é relevante, pois responde aos problemas práticos da automação de teste, embasados tanto nas recentes pesquisas como da demanda de mercado, e é inovador porque consolidou, a partir de uma proposta única, tanto os aspectos técnicos, diretamente ligados à automação de teste, como aqueles que dão

apoio ao projeto, implementação, execução e fechamento do teste, por meio da área de suporte.

8.2 Trabalhos futuros

Como trabalho futuro, se pode destacar o detalhamento do framework, para que suas subpráticas possam ser desenvolvidas mediante boas práticas que estão diretamente relacionadas a contextos de determinadas tecnologias. Ou seja, cada área de processo pode ser desenvolvida e compreendida como um escopo de pesquisa futura.

Não obstante, a pesquisa por ferramentas e suas tecnologias também pode ser considerada como uma pesquisa futura, que irá complementar a visão do FAST para determinados domínios e, dessa forma, o framework se tona uma base de conhecimento atemporal, que pode ser incrementada e evoluída a partir do domínio de aplicação.

Além disso, a análise de viabilidade dos projetos de automação é uma pergunta de pesquisa que ainda permanece aberta, pois mesmo após a introdução do FAST nos estudos de caso, não é possível a sistematização dos critérios mínimos requeridos para que um projeto possa introduzir automação de teste.

Outra oportunidade de trabalho futuro identificada, especificamente a partir da realização do grupo focal, foi o desenvolvimento de uma pesquisa para levantamento de requisitos para automação de teste em sistemas legados, haja vista que o framework não aborda este tópico.

Não obstante, o estudo e prática sobre adaptações do FAST aos diversos contextos de projetos, assim como o uso e adequação de ferramentas que apoiem a automação e suas limitações, também podem ser levantados como um trabalho futuro decorrente desta pesquisa.

Por fim, devido a dinamicidade da ciência da computação, faz-se necessária atualização constante do FAST, para que sua proposta possa estar coerente com as diversas tendências e tecnologias constantemente lançadas no escopo da engenharia de software.

REFERÊNCIAS

AFZAL, W; ALONE, S; GLOCKSIEN, K; TORKAR, R. Software test process improvement approaches: a systematic literature review and an industrial case study. **Journal of System and Software**, Volume 111. 2016, pp. 1–33.

ANASTAS e MACDONALD. **Research design for social work and the human services**. New Yourk, Lexington, 1994.

BAKER, C. **Review of D.D**. McCracken's "Digital Computer Programming". Mathematical Tables and Other Aids to Computation 11, 60. Outubro 1957, 298-305.

BALDASSARRE, M. T; FRANÇA, C; SILVA, F. Q. B**.** What Aspects of Context Should Be Described in Case Studies About Software Teams? Preliminary Results from a Mapping Study. In **International Conference on Software Process Improvement (PROFES)**, November, 22-24, Trondheim, Norway, 2016, pp. 723–730, 2016.

BARR, E; HARMAN, M; MCMINN, P; SHAHBAZ, M; YOO, S. The oracle problem in software testing: a survey. **IEEE Transactions on Software Engineering**, Volume 41, Issue n° 5, May 2015, pp. 507-525.

BASILI, V. R, WEISS, D. M. A methodology for collecting valid software engineering data. **IEEE Transactions on Software Engineering**, Volume SE-10, Issue, 1984, pp. 728–739.

BECK, K. **Test Driven Development By Example**. Addison Wesley, 2003.

BECK, K. **Extreme programming explained**: embrace change. 2a edição, Boston, MA, Addison-Wesley Professional, 2005.

BENBASAT, I; GOLDSTEIN, D; MEAD, M. The Case Research Strategy in Information Systems. **MIS Quaterly,** Volume 11, Issue n° 3, Setembro, 1987, pp. 369-386.

BERNER, S; WEBER, R; KELLER, R. Observations and lessons learned from automated testing. In: **International Conference on Software Engineering (ICSE)**, May 15-21, St. Louis, USA, 2005, pp. 571-579.

BERTOLINO, A. **Software testing research: achievements, challenges, dreams, future of software engineering**. IEEE Computer Society, 2007, pp 85–103. doi: 10.1109/FOSE.2007.25

BINDER, R. **Testing Object-Oriented Systems Models, Patterns, and Tools**. Addison Wesley Longman, Inc., Reading, MA, 2000.

BOEHM, B. **A view of 20th and 21st century software engineering**. In: **International Conference on Software Engineering (ICSE)**, Maio, Shangai, 2006, pp. 12-29.

BÖHMER, K., RINDERLE-MA, S. A systematic literature review on process model testing: Approaches, challenges, and research directions. **Cornell University Library**, set. 2015. Disponível em https://arxiv.org/abs/1509.04076 Acesso em 01 ago. 2016.

BRATTHALL, L e JØRGENSEN, M. Can you Trust a Single Data Source Exploratory Software Engineering Case Study. **Empirical Software Engineering**, Volume 7, pp. 9–26, 2002.

BRIAND, L. A critical analysis of empirical research in software testing. In: **First International Symposium on Empirical Software Engineering and Measurement**. Madrid, Spain: IEEE Computer Society, 2007.

CAPLAN, S. **Using focus group methodology for ergonomic design**. Ergonomics, v. 33, n. 5, p. 527-33, 1990.

CHELLADURAI, P. **Watch Your STEP**, 2011 http://www.uploads.pnsqc.org/ 2011/papers/T-56 _ Chelladurai _ paper.pdf Last accessed: Feb. 2016.

CHRISSIS M; KONRAD, M; SHRUM, S. **CMMI Second Edition: guidelines for process integration and product improvement**. Addison Wesley, 2007.

CRESWELL, J. **Research design:** qualitative, quantitative and mixed methods approaches, 4th Edition, Washington D.C., 2014.

CROSBY, P. B. **Quality is free**: the art of making quality certain. New York: McGraw-Hill, 1979.

DAMM, L; LUNDBERG, L. Results from introducing component-level test automation and test-driven development. **Journal of Systems and Software**, Volume 79, no. 7, pp. 1001–1014, 2006.

DEMARRAIS, K. Qualitative interview studies: Learning through experience. In: K. de Marrais & S. D. Lapan (Eds.), **Foundations for research** (pp. 51–68). Mahwah, NJ: Erlbaum. 2004

DEMING, E. **Out of the crisis**. Cambridge, MA, MIT Center for Advanced Engineering, 1982.

DEMING, W Edward. **The New Economics**. Cambridge, Massachusetts Institute of Technology, 1993.

DIJKSTRA, E. **Notes on Structured Programming**. Technical Report 70-WSK03, Technological Univ. Eindhoven, 1970.

http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF Acesso em..1 de janeiro de 2016.

DUSTIN, E; RASHKA, J; PAUL, J. **Automated software testing:** introduction, management, and performance**.** Boston, Addison-Wesley, 1999.

DYBA, T; KITCHENHAM, B; JØRGENSEN, M. Evidence based software engineering for practitioners. **IEEE Software**, Volume 22, Issue 1, 2005, pp. 58-65.

EASTERBROOK, S; SINGER, J; STOREY M; DAMIAN, D. **Selecting empirical methods for software engineering**, Springer London, 2008, pp. 285-311.

EISENHARDT, K. Building theories from case study research. **The Academy of Management Review**, Outubro 1989.

ELDH, S; ANDERSSON, K; WIKLUND, K. Towards a Test Automation Improvement Model (TAIM). In: **IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW)**, 31 Março a 4 Abril, Cleveland, OH, 2014, pp. 337-342.

ERICSON, T; SUBOTIC, A; URSING, S. TIM **-** A Test Improvement Model. **Software Testing Verification and Reliability**, Volume 7, pp. 229-246, 1997.

FEIGENBAUM, Armand V. **Total Quality Control**. McGraw-Hill, 1991.

FEWSTER, M. e GRAHAM, D. **Software Test Automation**: Effective Use of Test Execution Tools. Addison-Wesley, New York, 1999.

FEWSTER, M. Common Mistakes in Test Automation. In: **Fall test automation conference**, Boston, 2001.

FINCHER, S, and PETRE, M. (2004). **Computer Science Education Research**. Taylor and Francis, Janeiro, 2004.

FORSBERG, K; MOOZ, H. **The Relationship of System Engineering to the Project Cycle**. Published in National Council On Systems Engineering (NCOSE) and American Society for Engineering Management (ASEM), 21–23 October, Chattanooga, TN, 1991, pp. 57-65.

FURTADO, A; GOMES, M; ANDRADE, E; FARIAS JR, I. MPT.BR: a Brazilian maturity model for testing. Published In: **Quality Software International Conference (QSIC)**, Xi'an, Shaanxi, (2012), pp. 220-229.

FURTADO, A. P; MEIRA, S; GOMES, M. Towards a Maturity Model in Software Testing Automation. Published In: **International Conference on Software Engineering Advances (ICSEA)**, Nice, França, (2014); pp. 282-285.

FURTADO, A. P; MEIRA, S; SANTOS, C; NOVAIS, T; FERREIRA, M. FAST: Framework for Automating Software Testing. Published In: **International**

Conference on Software Engineering Advances (ICSEA), Roma, Italia, (2016), pp. 91.

GARCIA, C; DÁVILA, A; PESSOA, M. Test process models: systematic literature review, In: **Software Process Improvement and Capability Determination (SPICE 2014)**, Springer, 2014, pp. 84–93.

GAROUSI, V; FELDERER, M; HACALOGLU, T. Software test maturity assessment and test process improvement: A multivocal literature review. **Information and Software Technology**. Volume 85, Maio, 2017, pp. 16-42.

GELPERIN, D; HETZEL, B. The Growth of Software Testing. **Communications of the ACM**, Volume 31, Issue 3, Junho 1988, pp. 687-695.

HAROLD, M. Testing: a roadmap. In: **The Future of Software Engineering**. Ed by Finkelstein, A., 22th International Conference on Software Engineering (ICSE), Limerick, Ireland, June 2000, pp. 61-72.

HAYES, L. Automated Testing Handbook. Software Testing Institute, Richardson, TX, Março, 2004.

HEISKANEN, H., MAUNUMAA, M., KATARA M. A test process improvement model for automated test generation, In: **Product-Focused Software Process Improvement**, Springer, 2012, pp. 17–31.

HETZEL, W. **The Complete Guide to Software Testing**, Wellesley, 1988.

HONGYING, G; CHENG. Y. A customizable agile software quality assurance model, In: **5th International Conference on New Trends in Information Science and Service Science (NISS)**, 2011, pp. 382–387.

HÖST, M; RUNESON, P. Checklists for Software Engineering Case Study Research, In **Proceedings First International Symposium on Empirical Software Engineering and Measurement**, 2007, pp 479–481.

HUMPHREY, W. **Introduction to Software Process Improvement**, Technical Report, CMU/SEI-92-TR-007, ESC-TR-92-007. Software Engineering Institute (SEI), Junho, 1992.

IEEE COMPUTER SOCIETY. **IEEE Standard for Software Test Documentation:** IEEE 829-1983.

IEEE COMPUTER SOCIETY. **IEEE Standard for Software and System Test Documentation:** IEEE 829-2008.

IEEE COMPUTER SOCIETY. **IEEE Standard for Software Unit Testing**, IEEE STD 1008-1987.

IEEE COMPUTER SOCIETY. **IEEE Standard for System and Software Verification and Validation**, IEEE STD 1012-1986.

IEEE COMPUTER SOCIETY. **IEEE Standard for System and Software Verification and Validation**, IEEE STD 1012-2004 (Revision of IEEE STD 1012-1998).

IEEE COMPUTER SOCIETY. **IEEE Standard for System and Software Verification and Validation**, IEEE STD 1012-2012 (Revision of IEEE STD 1012-2004).

IEEE COMPUTER SOCIETY. **IEEE Standard Glossary of Software Engineering Terminoly**, IEEE STD 610.12-1990.

IEEE COMPUTER SOCIETY. **IEEE Standard Taxonomy for Software Engineering Standards**: IEEE STD 1002-1987.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION/IEEE COMPUTER SOCIETY (ISO/IEC/IEEE) 24.765. **Systems and software engineering – vocabulary**, 2010.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION/IEEE COMPUTER SOCIETY (ISO/IEC/IEEE) 29.119-1. **Software and systems engineering -- Software testing -- Part 1: Concepts and definitions**, 2013.

INTERNATIONAL ELECTROTECHNICAL COMISSION/IEEE COMPUTER SOCIETY (ISO/IEC/IEEE) 29.119-2. **Software and systems engineering -- Software testing -- Part 2: Test Process**, 2013.

INTERNATIONAL ELECTROTECHNICAL COMISSION/IEEE COMPUTER SOCIETY (ISO/IEC/IEEE) 29.119-4. **Software and systems engineering -- Software testing -- Part 4: Test Techniques**, 2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION (ISO/IEC) 12119. **Information technology -- Software packages -- Quality requirements and testing**, 1994.

ISO. ISO 9000:2015: Quality management systems: Fundamentals and vocabulary. Geneva: International Organization for Standardization, 2015. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:en>. Acesso em: 02 ago. 2016.

ISTQB 2012. **Standard glossary of terms used in Software Testing**, Version 2.2, 2012

JURAN, Joseph M. **Quality Control Handbook.** New York: McGraw Hill, 1951.

JURAN, Joseph M. **The quality trilogy: a universal approach to managing for Quality.** 1989.

KARHU, K; REPO, T; TAIPALE, O; SMOLANDER, K. Empirical Observations on Software Testing Automation. In: 2009 International Conference on Software Testing Verification and Validation. **IEEE**, Apr. 2009, pp. 201–209.

KITCHENHAM B.; DYBA, T., JØRGENSEN, M. **Evidence-based Software Engineering**. International Conference on Software Engineering (ICSE'04), 2004, pp 273-281,

KITZINGER, J. The methodology of focus group: the importance of interaction between research participants. In Journal of Sociology of Health and Illness. Volume 16, Issue 1, pp. 103-121, 1995.

KONTIO, J.; LEHTOLA, L.; BRAGGE, J. Using the focus group method in software engineering: obtaining practitioner and user experiences. Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE). **IEEE**, pp. 271–280, Redondo Beach, CA., ISBN 0-7695-2165-7/04, 2004.

KOOMEN, T. e POL, M. Improvement of the test process using TPI, In: **Proc. Sogeti Nederland BV**. Nederland ,1998.

KRAUSE, M. E. A Maturity Model for Automated Software Testing. **Medical Device & Diagnostic Industry Magazine**, December 1994, Disponível em http://www.mddionline.com/article/software-maturity-model-automated-software-testing. Acesso em Janeiro de 2015.

MARSHALL, C; ROSSMAN, **G. Designing Qualitative Research**. SAGE Publications, Washington, DC, USA, 2011.

MERRIAM, S. B; TISDELL, E. J. **Qualitative Research: A guide to Design and Implementation. Jossey-Bass, 4th edition**, 2016.

MYERS, G. **The art of software testing.** John Wiley & Sons, New York, NY, USA, 1979.

_____. **Melhoria do Processo de Teste Brasileiro (MPT.BR)**. SOFTEXRECIFE, RECIFE, 2011.

NAIK, K e TRIPATHY, P. **Software Testing and Quality Assurance**. Wiley, 2008.

PERRY, D; PORTER, A; VOTTA, L. Empirical Studies of Software Engineering: A Roadmap. In: **Proceedings of the Conference on The Future of Software Engineering (ICSE '00)**, 2000, pp. 345-355.

PERRY, D; SIM, S; EASTERBROOK, S. Case Studies for Software Engineers. In: **Proceedings of the 26th International Conference on Software Engineering (ICSE'04)**, 2004, local, pp. 736-738.

PERSSON, C; YILMAZTURK, N. Establishment of automated regression testing at ABB: industrial experience report on 'Avoiding the Pitfalls'. In: The 19th

international conference on automated software engineering (ASE'04). **IEEE Computer Society**. DOI: 10.1109/ASE.2004.1342729.

PROJECT MANAGEMENT INSTITUTE (PMI). **PMBOK - A Guide to the Project Management Body of Knowledge** (PMBOK® Guide) - Fifth Edition, 2013. Newtown Square, PA. Disponível em: http://bit.ly/1GlFsa6. Acesso em 30/03/2014.

RAFI D; MOSES K; PETERSEN K; MÄNTYLÄ M. Benefits and limitations of automated software testing: systematic literature review and practitioner survey. In: **7th International Workshop on Automation of Software Test (AST)**, June, 2012, Zurich, p.p. 36-42.

RAMLER, R; WOLFMAIER, K. Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost. In: **International workshop on Automation of Software Test (AST)**, 23 May, Shanghai, China, 2006, pp. 85-91.

REE, J; PAYTON, V. R. Focus groups: issues of analysis and interpretation. In: **Journal of Advanced Nursing**, Volume 26, pp. 765-771, 1997.

ROBSON, C. **Real World Research.** Blackwell, Segunda Edição, 2002.

RUNESON, P. e HÖST, M. Guidelines for conduction and reporting case study research in software engineering. In: **Empirical Software Engineering Journal**, Volume 14, Issue 2, pp. 131-164, April 2008.

RUNESON, P; HÖST, M; RAINER, A; REGNELL, B. **Case Study Research in software engineering: Guidelines and Examples. Wiley, 2012.**

SOFTWARE ENGINEERING INSTITUTE (SEI). **CMMI for development**, version 1.3, staged representation, 2010. Pittsburgh, PA. Disponível em: www.sei.cmu.edu/reports/10tr033.pdf, CMU/SEI-2010-TR-033. Acesso em 05/06/2012.

SLIFE, B. D., WILLIAMS, R. N. **What's behind the research? Discovering hidden assumptions in the behavioral sciences**. Thousand Oaks, 1995.

SOFTEX. MPS.BR - melhoria de processo de software brasileiro - Guia de Implementação de Software – Parte 1: Nível G. Rio de Janeiro, Fevereiro, 2016.

SJOBERG, D. I. K.; DYBA, T.; JORGENSEN, M. **The future of empirical methods in software engineering research**. Proceedings of the Future of Software Engineering (FOSE '07). [S.l.]: IEEE. 2007.

SWEBOK (2007). **Guide to the Software Engineering Body of Knowledge.** Body of knowledge, Fraunhofer IESE, Robert Bosch GmbH, University of Groningen, University of Karlskrona/Ronneby, Siemens.

TAIPALE, O.; KASURINEN, J.; KARHU, K.; and SMOLANDER, K. Trade-off between automated and manual software testing. In: **International Journal of**

System Assurance Engineering and Management, June, 2011, Volume 2, Issue 2, pp. 114–125.

_____. **Test Maturity Model (TMM)**. 1996. Illinois Institute of Technology, Disponível em: http://science.iit.edu/computer-science/research/testing-maturity-model-tmm 2014.08.11

_____. **Test Maturity Model Integration (TMMI).** TMMi Foundation, Versão 1.0, 2012. Disponível em: http://www.tmmi.org/pdf/TMMi.Framework.pdf 2014.08.11. Acesso em Janeiro de 2015:

TURING, Allan. **Checking a Large Routine**. Report of a Conference on High Speed Automatic Calculating-machines, January, 1950a, pp.67-69.

TURING, Allan. **Computing Machinery and Intelligence.** Mind 59, October, 1950b, pp. 433-460.

VAN SOLINGEN, R; BERGHOUT; E. **The goal/question/metric method:** A practical guide for quality improvement of software development. McGraw-Hill, 1999.

WISSINK, T; AMARO, C. Successful Test Automation for Software Maintenance**.** In: **Software Maintenance**, 2006. ICSM'06. 22nd IEEE International Conference on. IEEE, 2006, pp. 265–266.

WIKLUND, K; SUNDMARK, D; ELDH, S; LUNDQVIST, K. Impediments for Automated Testing - An Empirical Analysis of a User Support Discussion Board. In: **IEEE International Conference on Software Testing**, Verification, and Validation, ICST 2014, March 31 -April 4, Cleveland, OH, 2014, pp 113-122.

WIKLUND, K; ELDH, S; SUNDMARK, D; LUNDQVIST, K. Technical Debt in Test Automation. In: **IEEE Fifth International Conference on Software Testing**, Verification and Validation, 17-21 April, Montreal, QC, 2012, pp. 887 – 892.

APÊNDICE A – ÁREA TÉCNICA: 1- TESTE UNITÁRIO

Área de processo:	**1- Teste unitário**
Objetivo da área de processo:	Determinar a corretude e completude de uma implementação com base nos requisitos da unidade (IEEE 1008, 1987, p. 17).
Práticas:	1.1 Projetar teste unitário. 1.2 Implementar teste unitário. 1.3 Executar teste unitário. 1.4 Finalizar teste unitário.
Referências:	ANSI/IEEE 1008 (1987). ISO/IEC/IEEE 29.119-1 (2013).

Prática:	**1.1- Projetar teste unitário**
Objetivo:	Projetar um conjunto de testes unitários que estejam de acordo com os objetivos de seus requisitos.
Subpráticas:	• **Identificar requisitos e procedimentos adicionais que precisam ser testados.** Identificar requisitos que não sejam funcionais (por exemplo, desempenho, atributos ou restrições de projeto) associados a características do software que possam ser efetivamente testadas em nível da unidade. Identifique quaisquer procedimentos operacionais associados à unidade a ser testada. • **Identificar os tipos de dados de entrada e saída.** Para cada estrutura a ser testada, identificar os tipos de dados de entrada e saída e as características, tais como taxas de chegada, formatos e intervalos de valores. Para cada característica, especifique seus intervalos válidos. • **Projetar a arquitetura do conjunto de testes.** Com base nos requisitos a serem testados, projete um conjunto de objetivos de teste hierarquicamente decompostos para que cada objetivo de nível mais baixo possa ser diretamente testado por alguns casos de teste. Selecione os casos de teste existentes que sejam apropriados para o contexto. Associar grupos de identificadores de casos de teste com os objetivos de nível mais baixo. • **Especificar os procedimentos de teste requeridos.** Documentação existente, tais como documentação de requisitos, especificação de caso de teste e informação de planejamento pode conter os procedimentos de teste unitário de maneira implícita. Nesse caso, a necessidade de especificação explícita é minimizada. Selecione procedimentos de teste existentes que podem ser modificados ou usados sem modificação. Especifique quaisquer procedimentos adicionais necessários. • **Especificar os casos de teste.** Atualizar a especificação dos casos de teste existentes ou especificar novos casos de teste.
Produtos de trabalho:	• Projeto do teste unitário. • Especificação dos procedimentos de teste. • Especificação dos casos de teste.

Prática:	1.2- Implementar teste unitário
Objetivo:	Transformar o projeto em implementação dos testes unitários.
Subpráticas:	• **Obter e verificar os dados do teste.** Obter uma cópia dos dados existentes ou gerar novos dados para serem utilizados. Incluir dados adicionais que sejam necessários para garantir sua consistência e integridade. Verificar todos os dados (incluindo os que devem ser utilizados) com base nas especificações da estrutura de dados do software. Quando a correlação entre os casos de teste e os dados não estiver claramente definida, estabelecer uma tabela para explicitar a rastreabilidade entre os mesmos. Além disso, obter os recursos adicionais do teste, conforme especificados no projeto de teste unitário. • **Obter itens de teste.** Coletar os itens de teste disponíveis, tais como manuais, procedimentos de sistemas operacionais, dados de controle e programas. Obter o software que faz interface direta com as unidades de teste. Registrar informações necessárias no sumário de testes. • **Implementar testes unitários.** Implementados os testes unitários para contemplar a especificação do projeto de teste.
Produtos de trabalho:	• Dados de teste. • Recursos de apoio ao teste. • Configuração dos itens de teste. • Sumário de testes.

Prática:	1.3- Executar teste unitário
Objetivo:	Executar os casos de teste unitário implementados de acordo com o escopo planejado.
Subpráticas:	• **Executar teste unitário.** Realizar a configuração do ambiente de teste e executar o conjunto de testes. Registrar todos os incidentes no sumário de testes. • **Determinar resultados do teste unitário**. Para cada caso de teste, determinar se a unidade passou ou falhou com base na descrição dos casos de teste e registrar, apropriadamente, os resultados no sumário de teste. Para cada falha, analisa-la e registra-la no Sumário de Teste. Atualizar os casos de teste quando necessário.
Produtos de trabalho:	• Sumário de testes. • Log da execução de teste. • Especificações de teste revisadas. • Dados de teste revisados.

Prática:	1.4- Finalizar teste unitário
Objetivo:	Analisar e registrar informações decorrentes da execução dos testes unitários.
Subpráticas:	• **Avaliar finalização normal do teste unitário.** Avaliar a necessidade de testes adicionais. Caso não seja necessário, registrar a finalização dos testes.

Prática:	1.4- Finalizar teste unitário
	• **Avaliar finalização anormal do teste unitário.** Caso critério de suspensão do caso de teste tenha sido satisfeito (por exemplo, falha grave não corrigida, execução realizada fora do tempo previsto), analisar e documentar a situação que causou a suspensão do teste no Sumário do Teste. Documentar os casos de teste que não foram executados, decorrentes dessa situação. Avaliar o esforço da execução do teste unitário. • **Incrementar o conjunto de teste unitário**. Quando testes adicionais são requeridos, incrementar o conjunto de testes através dos seguintes passos: 1) Atualizar a arquitetura dos testes; 2) Modificar a especificação dos procedimentos de teste; 3) Agregar dados de testes adicionais; 4) Registrar no Sumário de Teste; e 5) Executar os testes adicionais.
Produtos de trabalho:	• Sumário de teste. • Especificações de teste revisadas. • Dados de teste adicionais.

APÊNDICE B – ÁREA TÉCNICA: 2- TESTE DE INTEGRAÇÃO

Área de processo:	**2- Teste de Integração**
Objetivo:	Avaliar a integração e relação entre os componentes de software para garantir que o sistema esteja consistente com sua arquitetura (IEEE 610-12, 1990).
Práticas:	2.1 Estabelecer técnicas de teste de integração. 2.2 Projetar teste de integração. 2.3 Implementar teste de integração 2.4 Executar teste de integração 2.5 Finalizar teste de integração.
Referências:	ISO/IEC/IEEE 24.765 (2010) IEEE 1012 (2012) IEEE 1008 (1987) ISO/IEC/IEEE 29.119-1 (2013) ISO/IEC/IEEE 29.119-4 (2015) NAIK e TRIPATHY (2008)

Prática:	**2.1-Estabelecer técnicas de teste de integração**
Objetivo:	Selecionar os tipos e técnicas de teste de sistemas que serão aplicados no contexto do projeto de acordo com seus requisitos funcionais e não funcionais.
Subpráticas:	• **Estabelecer granularidade do teste de integração.** A granularidade do teste corresponde ao nível de detalhes em que o teste de integração pode ser realizado (NAIK e TRIPATHY, 2008) e o teste de integração pode ser realizado em diferentes granularidades. Existem diversas abordagens para se estabelecer a granularidade do teste unitário, tais como: 1) **Intrassistema**: considerado com teste de integração baixo-nível, cujo objetivo é combinar módulos para formar o sistema coeso, onde o teste de integração é realizado a partir de *builds* sucessivas da integração do código considerando toda a estrutura interna dos módulos a serem testados. 2) **Interssistema**: abordagem de teste de alto-nível que considera o teste entre interfaces de módulos independentes, onde todos os mesmos são integrados e testados de ponta a ponta, cujo objetivo é garantir que a interação entre os módulos funciona e ignora os mecanismos internos de um módulo concentrando-se unicamente nas saídas geradas em resposta a entradas selecionadas e condições de execução 3) **Pairwise**: contempla o nível intermediário de teste de integração, onde apenas dois módulos relacionados são testados por vez. • **Selecionar técnicas de projeto de teste de integração.** Técnicas de projeto de teste são atividades, conceitos, processos e padrões usados para construir o projeto de teste com o intuito de apoiar na identificação das condições para um item de teste, derivar itens de cobertura de teste correspondentes e, subsequentemente, derivar ou selecionar casos de teste (ISO/IEC/IEEE 29.119-1, 2013, pp. 8). As técnicas baseadas em estrutura são também conhecidas como técnicas caixa-branca, onde existe a visibilidade da estrutura interna do item testado. Algumas técnicas baseadas em estrutura são listadas abaixo, conforme descritas na ISO/IEC/IEEE 29.119-4 (2015): 1) Teste de declaração;

Prática:	2.1-Estabelecer técnicas de teste de integração
	2) Teste de branch; 3) Teste de decisão; 4) Teste de condição de branch; 5) Teste de combinação de condição de branch; 6) Teste de cobertura de decisão de condição modificada; e, 7) Teste de fluxo de dados. Naik e Tripathy (2008, pp. 196/164) complementam a lista de técnicas de teste de integração com as seguintes: 1) **Incremental**: a integração é conduzida de maneira incremental, onde a partir de cada ciclo de teste, as falhas são encontradas e corrigidas antes de passar para o ciclo seguinte. O número de integrações necessárias depende dos parâmetros do sistema, tais como: o Número de módulos; o Complexidade ciclomática dos módulos; o Complexidade da interface dos módulos; e, o Número de módulos que necessitam ser integrados. 2) **Top-down**: aplicável a sistemas com estrutura hierárquica onde a arquitetura do sistema deve ser utilizada como base para o projeto da automação da integração de módulos. Nesse cenário, para a realização do teste de integração, o foco começa pelos módulos pais sua integração com módulos de nível hierárquico inferior. 3) **Bottom-up**: o teste de integração se inicia com a integração dos módulos de níveis mais baixo até chegar ao topo da estrutura, onde o processo de teste de integração se finaliza quando a estrutura de nível mais elevado é testada. 4) **Sanduiche**: é uma técnica de teste de integração que mescla as abordagens top-down e bottom-up. 5) **Big bang**: abordagem em que todos os módulos são testados individualmente para depois serem integrados no sistema para que o sistema possa ser testado por completo.
Produtos de trabalho:	• Projeto de teste de integração.

Prática:	2.2- Projetar teste de integração
Objetivo:	Derivar os procedimentos e casos de teste de integração.
Subpráticas:	• **Derivar as condições do teste.** A condição de teste é um aspecto testável de um componente ou sistema, como uma função, transação, recurso, atributo de qualidade ou elemento estrutural identificado como base para o teste, e podem ser usadas para derivar itens de cobertura do teste (ISO/IEC/IEEE 29.119-1, 2013, pp. 7). As condições do teste são determinadas a partir dos critérios de automação e devem ser priorizadas com base no nível de exposição do risco, definido no planejamento do projeto, e registradas na especificação do projeto de teste. • **Derivar os itens de cobertura do teste**. Os itens de cobertura de teste devem ser derivados a partir da aplicação das técnicas de projeto de teste sobre as condições de teste para alcançar os critérios de cobertura especificados no plano de teste. Os itens devem ser priorizados de acordo com o nível de exposição do risco do produto. Registrar os itens de cobertura na especificação do projeto de teste.

Prática:	2.2- Projetar teste de integração
	• **Derivar os casos de teste**. Os casos de teste podem ser derivados a partir dos itens de cobertura. Devem-se determinar as pré-condições, por meio da seleção de valores de entrada, e resultados esperados. Os casos de teste devem ser priorizados e registrados na especificação do projeto de teste. • **Montar os conjuntos de teste.** Os casos de teste podem ser distribuídos em um ou mais conjuntos de teste, com base nas suas restrições de execução, e registrados na especificação do projeto de teste. • **Estabelecer os procedimentos de teste.** O procedimento de teste compreende uma sequência de casos de teste em ordem de execução. Os procedimentos de teste devem ser elaborados a partir da ordenação dos casos de teste de conjunto de teste, e de acordo com as dependências descritas pelas pré-condições, pós-condições e demais requisitos de teste. Os procedimentos de teste devem ser priorizados e registrados na especificação do projeto de teste.
Produtos de trabalho:	• Especificação do projeto de teste.

Prática:	2.3- Implementar teste de integração
Objetivo:	Transformar a especificação do projeto de teste de integração em script de teste automatizados.
Subpráticas:	• **Implementar scripts de teste de integração.** Os scripts devem ser implementados para contemplar a especificação do projeto de teste e devem ser desenvolvidos a partir da especificação do projeto de teste de sistemas.
Produtos de trabalho:	• Scritps de teste de integração.

Prática:	2.4. Executar teste de integração
Objetivo:	Executar os scripts de teste automatizados.
Subpráticas:	• **Executar scripts de testes de integração.** Garantir que as pré-condições do caso de teste sejam atendidas antes de inicia sua execução. Executar um ou mais scripts de teste no ambiente preparado e observar o resultado. Registrar a execução do teste e todos os incidentes no sumário de teste. • **Determinar resultados do teste.** Analisar o resultado da execução do teste automatizado e comparar o resultado previsto com o obtido após execução. Para cada caso de teste, determinar se o mesmo passou ou falhou com base em suas especificações.

Produtos de trabalho:	• Sumário de testes. • Resultados da execução.

Prática:	**2.5. Finalizar teste de integração.**
Objetivo:	Finalizar a execução do teste e determinar os resultados.
Subpráticas:	• **Avaliar finalização normal do teste de integração.** Avaliar a necessidade de testes adicionais. Caso não seja necessário, registrar a finalização dos testes. • **Avaliar finalização anormal do teste de integração.** Caso critério de suspensão do caso de teste tenha sido satisfeito (por exemplo, falha grave não corrigida, execução realizada fora do tempo previsto), analisar e documentar a situação que causou a suspensão do teste no Sumário do Teste. Documentar os casos de teste que não foram executados, decorrentes dessa situação. Avaliar o esforço da execução do teste de integração. • **Incrementar o conjunto de teste de integração**. Quando testes adicionais são requeridos, incrementar o conjunto de testes através dos seguintes passos: 1) Atualizar a arquitetura dos testes; 2) Modificar a especificação dos procedimentos de teste; 3) Agregar dados de testes adicionais; 4) Registrar no Sumário de Teste; e 5) Executar os testes adicionais. • **Executar pós-condições para os casos de teste.** Tem o objetivo de listar as atividades que devem ser realizadas a fim de garantir o fechamento apropriado da execução do teste. Essas atividades podem ser automatizadas com intuito de minimizar as falhas humanas, e compreendem nos seguintes passos: 1) Empacotar ativos de teste. 2) Descartar arquivos e registros das bases de dados de teste. 3) Restaurar o software ou hardware. 4) Arquivar dados reutilizáveis. 5) Converter formato de arquivos necessários.
Produtos de trabalho:	• Sumário de testes. • Dados de teste.

APÊNDICE C – ÁREA TÉCNICA: 3- TESTE DE SISTEMA

Área de processo:	**3- Teste de Sistema**
Objetivo:	Realizar testes em um sistema completo e integrado para avaliar sua conformidade com os requisitos especificados (IEEE 610.12, 1990, p. 74).
Práticas:	3.1 Estabelecer tipos e técnicas de teste de sistema. 3.2 Projetar teste de sistema. 3.3 Implementar teste de sistema. 3.4 Executar teste de sistema. 3.5 Finalizar teste de sistema.
Referências:	IEEE 610.12 (1990) ISTQB Standard Glossary (2012) ISO/IEC/IEEE 29.119-1 (2013) ISO/IEC/IEEE 29.119-4 (2015) IEEE 829 (2008) FEWSTER e GRAHAM (1999) HAYES (2004)

Prática:	**3.1- Estabelecer tipos e técnicas de teste de sistemas**
Objetivo:	Selecionar os tipos e técnicas de teste de sistemas que serão aplicados no contexto do projeto de acordo com seus requisitos funcionais e não funcionais.
Subpráticas:	• **Selecionar tipos de teste de sistemas.** O tipo de teste compreende um grupo de atividades de teste destinadas a testar uma determinada característica de qualidade, como por exemplo, teste funcional, teste de usabilidade, regressão, etc. (ISO 29.119-1, 2013, pp. 12). O objetivo desta prática é selecionar tipos de teste adequados de que estejam de acordo com a demanda e características do projeto. Existem diversos tipos de teste de sistemas, tais como: 1) **Teste de concorrência**: teste para determinar a estabilidade de um sistema ou aplicação através de acessos simultâneos ao mesmo código de aplicativo, módulo ou registros de banco de dados. 2) **Teste de estabilidade**: testa se o sistema se mantém funcionando de maneira satisfatória após um período de uso 3) **Teste de estresse:** tem o objetivo de avaliar como o sistema funciona sob condições extremas de uso através da opressão de seus recursos ou tirando recursos dele. 4) **Teste de GUI**: testar a interface gráfica do usuário para garantir que a mesma atende as especificações. 5) **Teste de regressão**: retestar um sistema ou componente para verificar se alguma modificação recente causou algum efeito indesejado, além de se certificar que o sistema ainda atende os requisitos. 6) **Teste de sanidade**: determinar se uma nova versão do software está funcionando bem o suficiente para permitir que a mesmas possa ir para outros estágios de teste. 7) **Teste de segurança**: teste para determinar se o sistema protege as informações e mantém as funcionalidades de acordo com o previsto. • **Selecionar técnicas para o projeto de teste de sistemas.** Técnicas de projeto de teste são atividades, conceitos, processos e padrões usados para construir o projeto de teste com o intuito de apoiar na identificação das

Prática:	3.1- Estabelecer tipos e técnicas de teste de sistemas

condições para um item de teste, derivar itens de cobertura de teste correspondentes e, subsequentemente, derivar ou selecionar casos de teste (ISO/IEC/IEEE 29.119-1, 2013, pp. 8).

As técnicas baseadas em especificação são também conhecidas como técnicas caixa-preta, onde não há visibilidade da estrutura interna do item testado. Algumas técnicas baseadas em especificação estão listadas abaixo, conforme descritas na ISO/IEC/IEEE 29.119-4 (2015):

1) Partição e equivalência;
2) Método de classificação em árvore:
3) Análise de valor limite:
4) Teste de sintaxe;
5) Técnica de projeto de teste combinatório;
6) Tabela de decisão;
7) Gráfico de causa e efeito;
8) Transição de estado;
9) Teste de cenário; e
10) Teste randômico.

A seleção dessas técnicas serve como base para projetar os casos de teste, mais especificamente para derivar as condições, os itens de cobertura e os casos de teste.

- **Selecionar técnicas de geração de dados.**

Ferramentas são capazes de serem utilizadas para automatizar parte da atividade de projetar teste, e denominadas como ferramentas de geração de dados. Tais ferramentas dependem de algoritmos para gerar os dados, e podem ser classificadas de acordo com os seguintes tipos (FEWSTER e GRAHAM, 1999, p. 19):

1) **Baseadas em código**: são ferramentas que geram entradas para os testes através da análise do código e são úteis para serem utilizadas em conjunto com ferramentas para análise de cobertura de código.
2) **Baseadas em interface**: são ferramentas que podem gerar entradas de teste a partir de interfaces bem definidas, tais com GUI ou aplicações web.
3) **Baseadas em especificação**: são ferramentas que podem gerar tanto entradas como saídas esperadas de um teste a partir de uma especificação que possa ser analisada por uma ferramenta.

O uso de ferramentas pode ser útil em alguns contextos, e a melhor forma de aplica-la é quando o contexto do que pode ou não ser feito é completamente compreendido.

- **Selecionar técnicas para geração dos scripts.**

Um script de teste é uma especificação de um procedimento de teste automatizado, composto por uma série de comandos, eventos e decisões lógicas, que são armazenadas em um arquivo de linguagem de programação (ISTQB, 2012, pp. 47) (HAYES, 2004, pp. 18). Uma linguagem de *script* é a linguagem de programação utilizada para escrever os scripts de teste para que os mesmos possam ser utilizados por uma ferramenta de execução de teste (ISTQB, 2012, p. 38).

Prática:	**3.1- Estabelecer tipos e técnicas de teste de sistemas**
	Existem diferentes técnicas para elaboração de *scprits*, tais como (FEWSTER e GRAHAM, 1999, p. 92): 1) **Script linear**: é uma técnica que faz gravação ou replicação direta das ações do teste sem nada acrescentar, que consiste em gravar as ações executadas por um usuário sobre a interface gráfica e convertê-las em scripts de teste que podem ser executadas quantas vezes for necessário. 2) **Script estruturado**: O script estruturado é similar à programação estruturada, onde algumas instruções especiais são usadas para controlar a execução do script. Essas instruções especiais podem ser estruturas de controle ou uma estrutura de chamada. 3) **Script compartilhado:** Os scripts compartilhados são usados (ou compartilhados) por mais de um caso de teste. Isso implica que é necessário que a linguagem de script permita que um script seja "chamado" por outro. A ideia é produzir script que execute alguma tarefa que tem que ser repetida para testes diferentes e, em seguida, sempre que essa tarefa tiver que ser feita, chama-se este script no ponto apropriado em cada caso de teste. 4) **Script orientado a dados**: esta técnica armazena as entradas de teste em um arquivo separado (dados) ao invés de no próprio script, onde as informações de controle ficam no script. Quando o teste é executado, a entrada de teste é lida a partir do arquivo em vez de ser tirada diretamente do script onde o mesmo pode ser usado para executar diferentes testes. 5) **Script orientado a palavra-chave (keyword)**: é uma extensão lógica à mais sofisticada técnica de orientação à dados que à combina com o desejo de ser possível a especificação de casos de teste automáticos sem ter que especificar todos os detalhes.
Produtos de trabalho:	• Projeto de teste.

Prática:	**3.2- Projetar testes de sistema**
Objetivo:	Derivar os procedimentos e casos de teste.
Subpráticas:	• **Derivar as condições do teste.** A condição de teste é um aspecto testável de um componente ou sistema, como uma função, transação, recurso, atributo de qualidade ou elemento estrutural identificado como base para o teste, e podem ser usadas para derivar itens de cobertura do teste (ISO/IEC/IEEE 29.119-1, 2013, p. 7). As condições do teste são determinadas a partir dos critérios de automação e devem ser priorizadas com base no nível de exposição do risco, definido no planejamento do projeto, e registradas na especificação do projeto de teste. • **Derivar os itens de cobertura do teste**. Os itens de cobertura de teste devem ser derivados a partir da aplicação das técnicas de projeto de teste sobre as condições de teste para alcançar os critérios de cobertura especificados no plano de teste. Os itens devem ser priorizados de acordo com o nível de exposição do risco do produto. Registrar os itens de cobertura na especificação do projeto de teste. • **Derivar os casos de teste**. Os casos de teste podem ser derivados a partir dos itens de cobertura. Devem-se determinar as pré-condições, mediante seleção de valores de entrada, e resultados

Prática:	3.2- Projetar testes de sistema
	esperados. Os casos de teste devem ser priorizados e registrados na especificação do projeto de teste. • **Montar os conjuntos de teste.** Os casos de teste podem ser distribuídos em um ou mais conjuntos de teste, com base nas suas restrições de execução, e registrados na especificação do projeto de teste. • **Estabelecer os procedimentos de teste.** O procedimento de teste compreende uma sequência de casos de teste em ordem de execução. Os procedimentos de teste devem ser elaborados a partir da ordenação dos casos de teste de conjunto de teste, e de acordo com as dependências descritas pelas pré-condições, pós-condições e demais requisitos de teste. Os procedimentos de teste devem ser priorizados e registrados na especificação do projeto de teste.
Produtos de trabalho:	• Especificação do projeto de teste.

Prática:	3.4- Implementar teste de sistemas
Objetivo:	Transformar a especificação do projeto de teste de sistemas em script de teste automatizados.
Subpráticas:	• **Implementar os scripts de testes de sistema.** Os scripts devem ser implementados para contemplar a especificação do projeto de teste e devem ser desenvolvidos com base nas técnicas para geração de scripts selecionadas.
Produtos de trabalho:	• Scripts de teste de sistemas.

Prática:	3.5- Executar teste de sistemas
Objetivo:	Executar os scripts de teste automatizados.
Subpráticas:	• **Executar scripts de testes de sistema.** Garantir que as pré-condições do caso de teste sejam atendidas antes de inicia sua execução. Executar um ou mais scripts de teste no ambiente preparado e observar o resultado. Registrar a execução do teste e todos os incidentes no sumário de teste. • **Determinar resultados do teste.** Analisar o resultado da execução do teste automatizado e comparar o resultado previsto com o obtido após execução. Para cada caso de teste, determinar se o mesmo passou ou falhou com base em suas especificações.
Produtos de trabalho:	• Sumário de testes. • Resultados da execução.

Prática:	3.- Finalizar teste de sistema
Objetivo:	Finalizar a execução do teste e determinar os resultados.
Subpráticas:	• **Avaliar finalização normal do teste de sistema.**

Prática:	3.- Finalizar teste de sistema
	Avaliar a necessidade de testes adicionais. Caso não seja necessário, registrar a finalização dos testes. • **Avaliar finalização anormal do teste de sistema.** Caso critério de suspensão do caso de teste tenha sido satisfeito (por exemplo, falha grave não corrigida, execução realizada fora do tempo previsto), analisar e documentar a situação que causou a suspensão do teste no Sumário do Teste. Documentar os casos de teste que não foram executados, decorrente, dessa situação. Avaliar o esforço da execução do teste de sistemas. • **Incrementar o conjunto de teste de sistemas**. Quando testes adicionais são requeridos, incrementar o conjunto de testes através dos seguintes passos: 1) Atualizar a arquitetura dos testes; 2) Modificar a especificação dos procedimentos de teste; 3) Agregar dados de testes adicionais; 4) Registrar no Sumário de Teste; e, 5) Executar os testes adicionais. • **Executar pós-condições para os casos de teste.** Tem o objetivo de listar as atividades que devem ser realizadas a fim de garantir o fechamento apropriado da execução do teste. Essas atividades podem ser automatizadas com intuito de minimizar as falhas humanas, e compreendem nos seguintes passos: 1) Empacotar ativos de teste. 2) Descartar arquivos e registros das bases de dados de teste. 3) Restaurar o software ou hardware. 4) Arquivar dados reutilizáveis. 5) Converter formato de arquivos necessários.
Produtos de trabalho:	• Sumário de testes. • Dados de teste.

APÊNDICE D – ÁREA TÉCNICA: 4- TESTE DE ACEITAÇÃO

Área de processo:	**4- Teste de aceitação**
Objetivo:	Conduzir testes formais para determinar se um sistema satisfaz ou não seus critérios de aceitação (IEEE 610.12, 1990, p. 8).
Práticas:	4.1 Estabelecer tipos e critérios de aceitação. 4.2 Projetar teste de aceitação. 4.3 Implementar teste de aceitação. 4.4 Executar teste de aceitação. 4.5 Finalizar teste de aceitação.
Referências:	ISO/IEC/IEEE 29.119-1 (2013) MPT.BR, (2011) NAIK e TRIPATHY (2008)

Prática:	**4.1- Estabelecer tipos e critérios de aceitação**
Objetivo:	Determinar tipos e critérios referentes ao teste de aceitação de acordo com o contexto do projeto.
Subpráticas:	• **Definir tipo de teste de aceitação.** Os tipos de teste de aceitação são determinados de acordo com o ambiente em que será executado, podendo ser classificados como (NAIK e TRIPATHY, 2008): 1) Teste de aceitação do usuário: conduzido pelo cliente para garantir que o sistema satisfaz os critérios de aceitação contratual. 2) Teste de aceitação de negócios: realizado no ambiente de desenvolvimento com objetivo de antecipar e corrigir possíveis problemas a serem encontrados no ambiente de produção. • **Definir critérios de aceitação**. O critério de aceitação deve ser, preferencialmente, mensurado e quantificável e deve ser definido de acordo com os atributos de qualidade relevantes e de importância para o domínio do projeto. Os critérios de aceitação podem ser, de acordo com (NAIK e TRIPATHY, 2008): 1) Corretude e completude funcional 2) Precisão. 3) Integridade de dados 4) Conversão de dados 5) Backup e recuperação 6) Usabilidade 7) Desempenho 8) Tempo de inicialização 9) Estresse 10) Confiabilidade e disponibilidade 11) Manutenção 12) Robustez 13) Oportunidade Timeliness 14) Confidencialidade e disponibilidade. 15) Compatibilidade e interoperabilidade. 16) Conformidade 17) Instabilidade e capacidade de atualização 18) Escalabilidade 19) Documentação

Produtos de trabalho:	• Projeto de teste.

Prática:	**4.2- Projetar teste de aceitação**
Objetivo:	Derivar os procedimentos e casos de teste de integração.
Subpráticas:	• **Derivar as condições do teste.** A condição de teste é um aspecto testável de um componente ou sistema, como uma função, transação, recurso, atributo de qualidade ou elemento estrutural identificado como base para o teste, e podem ser usadas para derivar itens de cobertura do teste (ISO/IEC/IEEE 29.119-1, 2013, pp. 7). As condições do teste são determinadas a partir dos critérios de automação e devem ser priorizadas com base no nível de exposição do risco, definido no planejamento do projeto, e registradas na especificação do projeto de teste. • **Derivar os itens de cobertura do teste**. Os itens de cobertura de teste devem ser derivados a partir da aplicação das técnicas de projeto de teste sobre as condições de teste para alcançar os critérios de cobertura especificados no plano de teste. Os itens devem ser priorizados de acordo com o nível de exposição do risco do produto. Registrar os itens de cobertura na especificação do projeto de teste. • **Derivar casos de teste**. Os casos de teste podem ser derivados a partir dos itens de cobertura. Devem-se determinar as pré-condições, mediante seleção de valores de entrada, e resultados esperados. Os casos de teste devem ser priorizados e registrados na especificação do projeto de teste. • **Montar os conjuntos de teste.** Os casos de teste podem ser distribuídos em um ou mais conjuntos de teste, com base nas suas restrições de execução, e registrados na especificação do projeto de teste. • **Estabelecer os procedimentos de teste.** O procedimento de teste compreende uma sequência de casos de teste em ordem de execução. Os procedimentos de teste devem ser elaborados a partir da ordenação dos casos de teste de conjunto de teste, e de acordo com as dependências descritas pelas pré-condições, pós-condições e demais requisitos de teste. Os procedimentos de teste devem ser priorizados e registrados na especificação do projeto de teste.
Produtos de trabalho:	• Especificação do projeto de teste.

Prática:	**4.3. Implementar teste de aceitação**
Objetivo:	Transformar a especificação do projeto de teste de sistemas em script de teste automatizados.

Prática:	4.3. Implementar teste de aceitação
Subpráticas:	• **Implementar os scripts de testes de aceitação.** Os scripts devem ser implementados para contemplar a especificação do projeto de teste e devem ser desenvolvidos com base nas técnicas para geração de scripts selecionadas.
Produtos de trabalho:	• Scripts de teste de sistemas.

Prática:	4.4- Executar teste de aceitação
Objetivo:	Executar os scripts de teste automatizados.
Subpráticas:	• **Executar scripts de testes de aceitação.** Garantir que as pré-condições do caso de teste sejam atendidas antes de inicia sua execução. Executar um ou mais scripts de teste no ambiente preparado e observar o resultado. Registrar a execução do teste e todos os incidentes no sumário de teste. • **Determinar resultados do teste.** Analisar o resultado da execução do teste automatizado e comparar o resultado previsto com o obtido após execução. Para cada caso de teste, determinar se o mesmo passou ou falhou com base em suas especificações.
Produtos de trabalho:	• Sumário de testes. • Resultados da execução.

Prática:	4.5- Finalizar teste de aceitação
Objetivo:	Finalizar a execução do teste e determinar os resultados.
Subpráticas:	• **Avaliar finalização normal do teste de aceitação.** Avaliar a necessidade de testes adicionais. Caso não seja necessário, registrar a finalização dos testes. Além disso, avaliar se o software atende aos critérios de aceitação. • **Avaliar finalização anormal do teste de aceitação.** Caso critério de suspensão do caso de teste tenha sido satisfeito (por exemplo, falha grave não corrigida, execução realizada fora do tempo previsto), analisar e documentar a situação que causou a suspensão do teste no Sumário do Teste. Documentar os casos de teste que não foram executados, decorrentes dessa situação. Avaliar o esforço da execução do teste de sistemas. • **Incrementar o conjunto de teste de aceitação**. Quando testes adicionais são requeridos, incrementar o conjunto de testes através dos seguintes passos: 1) Atualizar a arquitetura dos testes; 2) Modificar a especificação dos procedimentos de teste; 3) Agregar dados de testes adicionais; 4) Registrar no Sumário de Teste; e 5) Executar os testes adicionais.

Prática:	4.5- Finalizar teste de aceitação
Subpráticas:	• **Executar pós-condições para os casos de teste.** Tem o objetivo de listar as atividades que devem ser realizadas a fim de garantir o fechamento apropriado da execução do teste. Essas atividades podem ser automatizadas com intuito de minimizar as falhas humanas, e compreendem nos seguintes passos: 1) Empacotar ativos de teste. 2) Descartar arquivos e registros das bases de dados de teste. 3) Restaurar o software ou hardware. 4) Arquivar dados reutilizáveis. 5) Converter formato de arquivos necessários.
Produtos de trabalho:	• Sumário de testes. • Dados de teste.

APÊNDICE E – ÁREA DE SUPORTE: 1- PLANEJAMENTO DO PROJETO DE TESTE

Área de processo:	**1- Planejamento do projeto de teste**
Objetivo da área de processo:	Estabelecer o planejamento da estratégia e atividades de automação de testes no projeto.
Práticas:	1.1 Analisar riscos do produto. 1.2 Estabelecer estratégia de automação do teste. 1.3 Estabelecer estimativas. 1.4 Desenvolver o plano do projeto de automação de teste.
Referências:	TMMI (2012) PMBOK (2013)

Prática:	**1.1- Analisar riscos do produto**
Objetivo da prática:	A análise de riscos do produto visa avaliar as áreas críticas que precisam ser testadas.
Subpráticas:	• **Definir categorias dos riscos do produto.** O risco do produto é aquele diretamente relacionado ao objeto de teste (TMMI, 2012, pp. 210). As categorias dos riscos do produto são definidas para apoiar a identificação e avaliação dos mesmos no decorrer do projeto. Exemplos de categorias: 1) Riscos funcionais; 2) Riscos arquiteturais; 3) Riscos de performance; 4) Riscos não funcionais; etc. As categorias dos riscos do produto podem ser definidas a partir de uma Estrutura Analítica dos Riscos do Projeto – EAR, onde a mesma apresenta uma visão hierárquica dos riscos, ordenados por categorias e subcategorias, que identifica as diversas áreas e causas de riscos potenciais (PMBOK, 2013). • **Identificar riscos.** Os riscos do produto devem ser identificados com base nos requisitos e na perspectiva proveniente dos *stakehoolders* do projeto. Os mesmos podem ser identificados através de workshop de riscos, brainstorm, entrevistas, checklists, etc. (PMBOK, 2013). • **Analisar riscos.** Os riscos devem ser analisados com base na probabilidade de ocorrência e seu impacto. A avaliação da probabilidade pesquisa a probabilidade de cada risco específico ocorrer e a avaliação do impacto investiga as potenciais decorrências sobre um o projeto e seus parâmetros (por exemplo, cronograma, custo, qualidade ou desempenho). Os riscos podem ser priorizados com base nas suas implicações potenciais de afetar os objetivos do projeto, que podem ser analisados através de uma matriz de probabilidade e impacto, onde suas combinações são a base para a classificação de seu impacto no projeto (PMBOK, 2013).

Prática:	1.1- Analisar riscos do produto
Produtos de trabalho:	• Estrutura analítica do risco – EAR. • Tabela de riscos do produto. • Matriz de exposição do risco.

Prática:	1.2- Estabelecer estratégia de automação do teste
Objetivo:	A estratégia de automação do teste visa estabelecer o planejamento técnico da automação de teste no projeto com base na análise de riscos do produto.
Subpráticas:	• **Identificar escopo da automação do teste.** O escopo da automação deve ser definido levando em consideração a prioridade dos requisitos a partir da análise da exposição dos riscos do produto. Deve-se definir o escopo para cada nível de automação a partir da análise de riscos do produto. • **Definir estratégia de automação de teste.** A estratégia de teste identifica o que e de que forma o conjunto de teste será automatizado de acordo com o nível de automação correspondente. Para defini-la, é necessário, para cada nível de automação, associar o escopo aos tipos de teste relacionados e ferramentas que serão utilizadas. • **Definir critérios de entrada.** Definir os critérios mínimos necessários para que a automação de teste possa iniciar. Os critérios devem ser definidos de acordo com a estratégia de automação, contemplando os requisitos mínimos de cada nível de automação. Alguns exemplos de critérios de entrada: 1) Disponibilidade do relatório de automação de teste do nível anterior; 2) Disponibilidade do ambiente de execução de teste de acordo com o planejado; e, 3) Disponibilidade da documentação. • **Definir critérios de saída.** Definir os critérios mínimos necessários para que a automação de teste possa ser considerada como finalizada. Os critérios podem estar relacionados a diversas caraterísticas, tais como: 1) Percentual de testes planejados e executados; 2) Percentual de cobertura de cada item de teste; 3) Todos os riscos de alta prioridade mitigados; e, 4) Taxa de detecção de defeitos abaixo de um limiar. • **Definir critérios de suspensão do teste.** Os critérios de suspensão devem contemplar as condições necessárias para que suspender toda ou parta da automação de teste. Exemplos de critérios de suspensão: 1) Número de defeitos críticos encontrados; 2) Número de defeitos não reproduzíveis; e,

Prática:	1.2- Estabelecer estratégia de automação do teste
	3) Falhas na execução ou ambiente de teste. • **Definir critérios de reinício do teste.** Os critérios de reinício do teste especificam as atividades de teste que precisam ser repetidas para que o teste possa ser reiniciado. Alguns exemplos de critérios de reinício do teste podem ser: 1) Correção das falhas graves encontradas na rodada de testes precedente; 2) Repetir o completamente o procedimento de teste da falha encontrada; e, 3) Repetir os itens necessários de preparação do ambiente para reinício do teste.
Produtos de trabalho:	• Estratégia de automação de teste. • Critérios de automação de teste.

Prática:	1.3- Estabelecer estimativas
Objetivo:	Estimar o projeto de automação de teste.
Subpráticas:	• **Estabelecer Estrutura Analítica do Projeto – EAP.** A estrutura analítica do projeto (EAP) é uma decomposição hierárquica orientada às entregas do trabalho, sendo que cada nível descendente da EAP representa uma definição gradualmente mais detalhada da definição do trabalho do projeto, cujo objetivo é organizar e definir o escopo do trabalho (PMBOK, 2013). A definição de um EAP no projeto de automação de teste tem o objetivo de apoiar a definição do escopo das atividades de automação e, consequentemente, servir como base para a estimativa do projeto. • **Definir o ciclo de vida do teste.** O ciclo de vida de um projeto consiste nas fases do mesmo que são determinadas a partir das necessidades de gerenciamento e controle, a natureza do projeto em si e sua de aplicação (PMBOK, 2013). A definição do ciclo de vida tem o objetivo de declarar o contexto em que as atividades de automação irão acontecer no projeto. Nesse cenário, deve-se estabelecer se o ciclo de vida do projeto de automação é o mesmo do ciclo de vida do desenvolvimento do projeto, onde as atividades de automação fazer parte do desenvolvimento do projeto. Outra forma de definição de ciclo de vida se aplicada quando se trata de equipes independentes de teste, onde o ciclo de vida do projeto de automação irá contemplar o escopo apenas das atividades de automação. • **Determinar estimativas de tamanho e custo.** A partir da visão da EAP e ciclo de vida, determinar a estimativa de tamanho e esforço do projeto de automação. Para isso é necessário que as estimativas estejam baseadas em critérios objetivos. Nesse contexto, podem-se selecionar técnicas para apoiar a realização desta atividade. Realizar estimativa de esforço com base na estimativa de tamanho.
Produtos de trabalho:	• Estrutura analítica do projeto - EAP. • Ciclo de vida do projeto de automação;

Prática:	1.3- Estabelecer estimativas
	• Estimativas de tamanho do projeto de automação de teste. • Estimativa de custo do projeto de automação de teste.

Prática:	1.4- Desenvolver o plano do projeto de automação de teste
Objetivo:	Estabelecer e manter o planejamento da automação de teste como base para o monitoramento do mesmo.
Subpráticas:	• **Planejar cronograma.** Estabelecer o cronograma do projeto de automação de teste considerando os seguintes parâmetros do projeto: 1) Restrições de cronograma; 2) Duração das atividades; 3) Dependências entre as tarefas; e, 4) Marcos do projeto. • **Planejar recursos humanos.** Planejar os recursos humanos no projeto considerando: 1) Alocação dos recursos humanos no decorrer do projeto; 2) Disponibilidade e necessidades dos recursos humanos ao longo do projeto com base no cronograma; 3) Habilidades requeridas para realização do projeto; e, 4) Mecanismos para fornecer o conhecimento e habilidades necessários no projeto. • **Planejar envolvimento das partes interessadas do projeto.** Identificar as partes interessadas do projeto e planejar suas funções ao longo das atividades do projeto. • **Identificar os riscos do projeto.** Os riscos do projeto são aqueles que estão relacionados com o planejamento e acompanhamento do projeto de automação de teste. Para isso, é necessário: 1) Identificar os riscos do projeto. 2) Avaliar a probabilidade de ocorrência, que investiga a probabilidade de cada risco específico ocorrer. 3) Avaliar o impacto dos riscos que investiga o efeito potencial sobre um objetivo do projeto, como cronograma. 4) Planejar mitigação ao risco, que tem o propósito de adotar uma ação para minimizar a probabilidade e/ou o impacto de um risco ocorrer no projeto. 5) Planejar contingência, que tem o objetivo de planejar uma resposta se um risco acontecer no projeto. • **Estabelecer o plano.** Consolidar as informações para estabelecer e manter as informações para o planejamento da automação de teste no projeto.
Produtos de trabalho:	• Cronograma do projeto de automação de teste. • Matriz de recursos humanos. • Matriz de envolvimento das partes interessadas. • Planilha de riscos do projeto.

APÊNDICE F – ÁREA DE SUPORTE: 2- ACOMPANHAMENTO DO PROJETO DE TESTE

Área de processo:	**2- Acompanhamento do projeto de teste**
Objetivo da área de processo:	Avaliar se o progresso do projeto está ocorrendo de acordo com o planejado e tomar ações corretivas caso existam desvios significativos em relação ao planejado, atividades ou outros aspectos do plano de teste.
Práticas:	2.1 Monitorar o projeto com base no plano. 2.2 Gerenciar ações corretivas.
Referências:	TMMI, (2012) PMBOK (2013) ISO/IEC/IEEE 29.119-2 (2013)

Prática:	**2.1- Monitorar o projeto com base no plano**
Objetivo:	Acompanhar o progresso do projeto com relação ao que foi estabelecido no planejamento.
Subpráticas:	• **Monitorar o progresso do projeto.** As atividades de acompanhamento do projeto devem ser realizadas de maneira sistemática e devem contemplar: 1) Acompanhamento dos parâmetros do planejamento do projeto, tais como plano de trabalho, atividades, tarefas, custo, esforço e prazo. 2) Monitorar o desempenho do projeto com base no planejado. 3) Analisar o status do projeto em marcos estabelecido. 4) Identificar divergências e fatores que bloqueiam o progresso do projeto. 5) Identificar e analisar os riscos para realizar as devidas ações de mitigação ou contingência.
Produtos de trabalho:	• Planilha de acompanhamento do projeto de teste.

Prática:	**2.2- Gerenciar ações corretivas**
Objetivo:	Monitorar as ações corretivas desde a abertura até seu fechamento para garantir que os desvios do projeto sejam tratados de maneira apropriada.
Subpráticas:	• **Identificar desvios do projeto.** Esta atividade tem o foco de identificar os problemas e desvios do projeto e deve ser realizada ao longo de todo o projeto. O desvio deve ser sistematicamente registrado para que o mesmo possa ser tratado pelos responsáveis • **Registrar ações corretivas** O objetivo é realizar um registro da ação corretiva decorrente de algum problema e/ou desvio do projeto para que seja possível estabelecer ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas • **Acompanhar ação corretiva até seu fechamento.** O acompanhamento da ação corretiva é realizado até o seu fechamento para garantir a sua efetiva conclusão.

Prática:	**2.2- Gerenciar ações corretivas**
Produtos de trabalho:	• Planilha de acompanhamento do projeto de teste.

APÊNDICE G – ÁREA DE SUPORTE: 3- GERÊNCIA DE CONFIGURAÇÃO

Área de processo:	**5- Gerência de configuração**
Objetivo da área de processo:	Estabelecer e manter a integridade do ambiente de automação e seus produtos de trabalho no contexto do projeto de automação de teste (ISO 29.119-2, 2013).
Práticas:	3.1 Estabelecer ambiente de automação de teste. 3.2 Estabelecer baselines do projeto de automação de teste. 3.3 Acompanhar e controlar mudanças.
Referências:	CMMI (2010) TMMI (2012) ISO/IEC/IEEE 29.119-2 (2013)

Prática:	**3.1- Estabelecer ambiente de automação de teste.**
Objetivo:	Definir e estabelecer o ambiente necessário para os processos de automação de teste.
Subpráticas:	• **Estabelecer requisitos do ambiente:** Avaliar a estratégia e plano de teste para levantar as necessidades, dados e restrições do ambiente. Analisar e estabelecer os requisitos e projetar o ambiente, levando em consideração os seguintes itens: 1) Ferramentas. 2) Componentes de software. 3) Equipamentos de teste. 4) Base de dados de teste. • **Implementar o ambiente de automação de teste.** Implementar o ambiente de teste de acordo com seu planejamento e que esteja aderente a padrões e critérios previamente estabelecidos. Realizar testes no ambiente previamente a execução dos testes para garantir que o ambiente está apropriado para uso. Gerar dados de teste para apoiar a execução dos testes. • **Manter o ambiente de automação de teste.** O ambiente de automação de teste precisa ser mantido conforme especificações planejadas ou a partir de solicitações de mudanças para se adequar a evolução do projeto. As mudanças e adaptações precisam ser formalmente comunicadas às partes interessadas.
Produtos de trabalho:	• Plano de gerência de configuração.

Prática:	**3.2-. Estabelecer baselines do projeto de automação de teste.**
Objetivo:	Identificar e manter os itens de configuração e componentes, assim como estabelecer o sistema de gerência de configuração.
Subpráticas:	• **Identificar itens de configuração.** Identificar itens de configuração, componentes e produtos de trabalho relacionados a serem colocados sob gerência de configuração. Um item de configuração é uma entidade designada para gerenciamento de configuração, que pode consistir em vários produtos de trabalho relacionados que faze parte de uma *baseline* (CMMI, 2010, pp. 140/152). Para isso, é necessário:

Prática:	3.2-. Estabelecer baselines do projeto de automação de teste.
	1) Selecionar itens de configuração que fazem parte do projeto de automação de teste (ex. plano de teste, procedimentos, scripts, casos de teste). 2) Determinar identificadores únicos para os itens de configuração. 3) Especificar o nível de controle dos itens de configuração. O nível de controle descreve a formalidade com que um item de configuração é tratado no ambiente de gerência de configuração e sua definição se baseia nas características do projeto. • **Estabelecer o ambiente de gerenciar de configuração.** Um sistema de gerência de configuração consiste em um conjunto de subsistemas com diferentes implementações que são necessárias para o ambiente de gerência de configuração. Para isso, é necessário: 1) Estabelecer mecanismos para gerenciar múltiplos níveis de controle. 2) Definir o controle de acesso aos itens de configuração e sistema de gerência de configuração. 3) Armazenar, restaurar, compartilhar, atualizar, preservar e transferir itens de configuração no ambiente de gerência de configuração. • **Gerar baselines e releases.** Uma baseline é um conjunto de produtos de trabalho formalmente revisados que serve como base para desenvolvimento futuro e que só pode ser modificado a partir de procedimentos de controle de mudança (CMMI, 2010, pp. 436). 1) Gerar a baseline a partir dos itens no ambiente de gerencia de configuração. 2) Documentar os itens de gerencia de configuração que fazem parte da baseline. 3) Disponibilizar baseline.
Produtos de trabalho:	• Plano de gerência de configuração. • Sistema de gerência de configuração.

Prática:	3.3- Acompanhar e controlar mudanças.
Objetivo:	Controlar e acompanhar as solicitações de mudanças nos produtos de trabalho que estão sob gerência de configuração.
Subpráticas:	• **Acompanhar solicitações de mudanças** As mudanças no projeto podem ser decorrentes de requisitos novos ou modificados, como também a partir de falhas e defeitos nos produtos de trabalho. Para isso, é necessário: 1) Registrar solicitação de mudanças no sistema de solicitação de mudança. 2) Analisar o impacto da mudança. 3) Priorizar a mudança. 4) Acompanhar a solicitação de mudança até seu fechamento. • **Controlar itens de configuração** A baseline dos produtos de trabalho precisa ser controlada, o que incluir acompanhar a configuração de cada item de configuração, aprovar uma nova configuração, se necessário, e atualizar a baseline. Para isso, é necessário:

Prática:	**3.3- Acompanhar e controlar mudanças.**
	1) Controlar as mudanças nos itens de configuração durante todo o ciclo de vida do projeto. 2) Obter autorização prévia para realização das mudanças. 3) Incorporar as mudanças no sistema de gerência de configuração. 4) Registrar, apropriadamente, as mudanças e suas razões.
Produtos de trabalho:	• Sistema de solicitação de mudanças.

APÊNDICE H – ÁREA DE SUPORTE: 4- MEDIÇÃO E ANÁLISE

Área de processo:	**4- Medição e análise**
Objetivo da área de processo:	Desenvolver e manter mecanismos de medição para apoiar as atividades de automação de testes no projeto (TMMI, 2012).
Práticas:	4.1 Estabelecer mecanismos de medição e análise. 4.2 Fornecer os resultados da medição da automação de teste.
Referências:	BASILI e WEISS (1984) CMMI (2010) TMMI (2012) VAN SOLINGEN e BERGHOUT (1999)

Prática:	**4.1- Estabelecer mecanismos de medição e análise**
Objetivo:	Definir, objetivamente, os indicadores da automação de teste no projeto.
Subpráticas:	• **Estabelecer objetivos de medição da automação de teste.** O foco desta prática é estabelecer os objetivos de medição da automação que podem ser utilizados para avaliar a qualidade do processo de automação, monitorar melhoria e fornecer dados para o planejamento de produtividade e custo do projeto. Os objetivos de medição devem descrever os propósitos para os quais a medição e a análise são feitas e devem ser documentados e priorizados de acordo com a realidade do projeto de automação de teste. • **Especificar medidas da automação de teste.** Uma medida é uma variável à qual um valor é atribuído como resultado da medição (CMMI, 2013, p. 445). A partir de cada objetivo definido, devem-se especificar, de maneira objetiva e quantificável, medidas do processo de automação de teste. A execução dessa prática pode ser através do uso da técnica GQM (BASILI e WEISS 1984; VAN SOLINGEN e BERGHOUT, 1999) que fornece um guia para apoiar a definição de métricas a partir de objetivos de medição previamente estabelecidos. As métricas devem ser especificadas em termos precisos e inequívocos para que possam ser devidamente revisadas e priorizadas. Alguns exemplos de indicadores que podem ser utilizados, de acordo com categorias: 1) Custo da automação: o Criação / Implementação dos casos de teste. o Manutenção dos casos de teste. o Execução dos casos de teste. o Retorno sobre o investimento – ROI. 2) Processo de automação: o Percentual de testes automatizados. o Progresso da automação. o Percentual de cobertura de testes automatizados. o Cobertura dos testes o Densidade de defeitos. • **Especificar procedimentos de coleta e armazenamento das medidas.** O procedimento de coleta e armazenamento das métricas deve especificar como os dados das métricas são obtidos e armazenados. Para isso, é necessário:

Prática:	4.1- Estabelecer mecanismos de medição e análise
	1) Identificar as fontes de dados existentes decorrentes do processo de automação de teste. 2) Especificar como coletar e armazenar os dados para cada medida. 3) Criar mecanismos de coleta dos dados. 4) Criar mecanismos de armazenamento das medidas 5) Revisar, priorizar e atualizar os procedimentos de coleta e armazenamento sempre que necessário. • **Especificar procedimentos de análise das medidas.** O procedimento de análise descreve, de forma objetiva, como cada indicador deve ser avaliado para assegurar que as análises serão adequadamente conduzidas e comunicadas para as partes interessadas. Para isso, é necessário: 1) Especificar o mecanismo de análise das medidas. 2) Selecionar métodos e ferramentas, como por exemplo, o uso de gráficos, dados estatísticos, e ferramentas de apoio à análise de indicadores. 3) Revisar, priorizar e atualizar os procedimentos de análise sempre que necessário.
Produtos de trabalho:	• Plano de medição da automação de teste.

Prática:	4.2. Fornecer os resultados da medição da automação de teste.
Objetivo:	Coletar, analisar e comunicar o resultado da medição.
Sub-práticas:	• **Coletar dados da medição da automação.** Os dados da medição devem ser coletados e analisados quanto à sua integridade. Para isso é necessário avaliar a consistência dos dados sempre que possível. • **Analisar dados da medição da automação.** Os dados coletados devem ser analisados e interpretados conforme planejado na descrição das medidas. Nesse contexto, deve-se 1) Analisar e interpretar os resultados iniciais 2) Realizar medições adicionais, se necessário, para formar um maior embasamento na análise da medida. 3) Interpretar resultados e gerar conclusões. • **Comunicar resultados.** Os interessados do projeto devem ser comunicados da análise decorrente do processo de medição. • **Armazenar dados e resultados.** Os dados precisam ser devidamente armazenados para que se possa fazer o uso de dados e resultados históricos. Para isso, é necessário: 1) Revisar os dados para garantir a sua integridade e precisão. 2) Armazenar dados conforme os procedimentos de armazenamento. 3) Disponibilizar o conteúdo para uso apenas de grupos apropriados.

Prática:	4.2. Fornecer os resultados da medição da automação de teste.
Produtos de trabalho:	• Plano de medição da automação de teste.

APÊNDICE I – ÁREA DE SUPORTE: 5- REQUISITOS

Área de processo:	**5- Requisitos.**
Objetivo da área de processo:	Definir, analisar, estabelecer e manter os requisitos da automação de testes.
Práticas:	5.1 Definir requisitos da automação. 5.2 Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho. 5.3 Gerenciar mudanças de requisitos da automação.
Referências:	IEEE 829 (2008) IEEE 1012 (2012) MPS.BR (2016)

Prática:	**5.1. Definir requisitos da automação.**
Objetivo:	Identificar e analisar os requisitos da automação do teste com intuito de analisar sua testabilidade (IEEE 829, 2008).
Subpráticas:	• **Definir critérios objetivos para avaliação dos requisitos.** Os requisitos da automação devem ser avaliados com base em critérios objetivos previamente estabelecidos, contemplando as condições mínimas que os mesmos devem alcançar para serem considerados como válidos para o projeto de automação. Alguns exemplos de critérios são: possuir identificação única; estar claro e apropriadamente declarado; não ser ambíguo; ser relevante; ser completo; estar consistente com os demais requisitos; ser implementável, testável e rastreável, conforme o nível G do MPS.BR (2016). • **Elicitar requisitos da automação.** A elicitação dos requisitos compreende o levantamento e identificação dos requisitos da automação que não foram explicitamente coletados. Algumas técnicas podem ser utilizadas para apoiar a realização desta atividade, tais como (CMMI, 2010): 1) Protótipos; 2) Modelos; 3) Brainstorm; 4) Casos de uso; 5) Histórias dos usuários; e 6) Análise de regras de negócio. Além disso, os tipos de requisitos devem ser levantados para guiar a definição dos tipos de teste de automação, tais como requisitos funcionais e requisitos não funcionais.
Produtos de trabalho:	• Critérios para análise dos requisitos • Requisitos da automação.

Prática:	**5.2-. Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho**
Objetivo:	Definir a rastreabilidade bidirecional entre os requisitos da automação e seus produtos de trabalho relacionados.

Prática:	**5.2-. Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho**
Subpráticas:	• **Determinar esquema de rastreabilidade entre requisitos e produtos de trabalho da automação.** O esquema conceitual da rastreabilidade determina o mapeamento da forma com que os produtos de trabalho podem estar relacionados. O objetivo dessa prática é estrutura a rastreabilidade do projeto de teste para que a mesma possa ser sistematicamente aplicada. • **Definir rastreabilidade bidirecional dos requisitos.** A rastreabilidade tem o objetivo de apoiar a análise de impacto a partir de mudanças no escopo do projeto. Ela pode ser aplicada de maneira horizontal ou vertical. A rastreabilidade horizontal De acordo com a IEEE 829 (2008), a matriz de rastreabilidade deve contemplar: - Verificar que todos os requisitos estão relacionados a casos de teste. - Garantir que todos os casos de teste estejam relacionados a requisitos. - Garantir que exista um relacionamento entre o plano de teste, projeto de teste, casos de teste e procedimentos de teste.
Produtos de trabalho:	• Esquema conceitual de rastreabilidade. • Matriz de rastreabilidade.

Prática:	**5.3. Gerenciar mudanças de requisitos da automação.**
Objetivo:	Gerenciar as mudanças dos requisites da automação que ocorrem ao longo do projeto.
Subpráticas:	• **Gerenciar mudança de escopo da automação de teste.** As mudanças que acontecem ao longo do projeto podem ser decorrentes de adição de novos requisitos, retirada e/ou mudanças podem ser feitas nos requisitos existes. O objetivo é que a partir de uma solicitação de mudança possa ser realizada uma análise de impacto no projeto para orientar a gestão dos requisitos do projeto de automação de teste. • **Manter rastreabilidade bidirecional dos requisitos da automação de teste.** As mudanças de requisitos no projeto de automação que são realizadas ao longo do projeto devem gerar atualizações na rastreabilidade, com o objetivo de garantir integridade do esquema de rastreabilidade entre os produtos de trabalho do projeto e os artefatos relacionados à automação de testes. A manutenção da rastreabilidade é uma atividade que é aplicável a partir da de qualquer mudança no projeto.
Produtos de trabalho:	• Solicitação de mudança. • Análise de impacto. • Matriz de rastreabilidade.

APÊNDICE J – ÁREA DE SUPORTE: 6- GESTÃO DE DEFEITOS

Área de processo:	**6- Gestão de defeitos.**
Objetivo da área de processo:	Identificar, analisar e tratar os defeitos decorrentes da automação de teste. (TMMI, 2012)
Práticas:	6.1 Estabelecer sistema de gestão de defeitos. 6.2 Analisar resultado do teste. 6.3 Estabelecer ações para eliminar a causa-raiz dos defeitos.
Referências:	TMMI (2012)

Prática:	**6.1- Estabelecer sistema de gestão de defeitos.**
Objetivo:	Definir e manter um sistema em que os defeitos decorrentes da execução da automação de teste possam ser registrados e acompanhados até seu fechamento.
Subpráticas:	• **Definir estratégia de gestão de defeitos.** A estratégia de gestão de defeitos deve contemplar a máquina de estados das fases em que o defeito deve passar, desde sua abertura, a partir da execução da automação de teste, para a correção, re-teste e até seu fechamento. Além disso, também é necessário estabelecer critérios objetivos para a classificação e priorização dos defeitos no projeto. • **Definir um sistema de gestão de defeitos.** Definir requisitos necessários e estabelecer um sistema que permita o registro e acompanhamento dos defeitos encontrados a partir da execução da automação de teste. • **Manter sistema de gestão de defeitos.** O sistema de gestão de defeitos precisa ser mantido conforme especificações planejadas e/ou a partir de solicitações de mudanças para se adequar a evolução do projeto.
Produtos de trabalho:	• Máquina de estados do defeito. • Sistema de gestão de defeitos.

Prática:	**6.2- Analisar resultado do teste**
Objetivo:	Sistematicamente analisar o resultado dos testes para determinar a causa raiz dos defeitos encontrados.
Subpráticas:	• **Registrar, classificar e priorizar os defeitos.** Os defeitos encontrados decorrentes da execução da automação de teste devem ser registrados, classificados e priorizados conforme estratégia estabelecida. • **Analisar a causa raiz dos defeitos selecionados.** A partir dos defeitos priorizados, analisar e registrar a causa raiz dos mesmos com objetivo de identificar quais são as causas comuns dos defeitos encontrados. Alguns exemplos de técnicas que podem apoiar a análise da causa raiz (TMMI, 2012, pp. 167): - Diagrama de causa e efeito; - Diagrama de espinha de peixe Ishikawa; e - Checklists.

Prática:	6.2- Analisar resultado do teste
Produtos de trabalho:	• Registro do defeito. • Análise da causa raiz dos defeitos.

Prática:	6.3- Estabelecer ações para eliminar a causa-raiz dos defeitos.
Objetivo:	Traçar ações corretivas para eliminar a causa-raiz dos defeitos encontrados decorrente da automação de teste.
Subpráticas:	• **Propor ações corretivas.** O objetivo é determinar ações corretivas para que se possa eliminar a causa-raiz dos defeitos e assim evitar que o mesmo reincida no projeto. Ações corretivas podem estar relacionadas com (TMMI, 2012, p. 168): - Melhoria de processo; - Treinamento; - Adequação da estratégia de automação; - Padrões de codificação; • **Acompanhar execução das ações corretivas.** O objetivo é garantir que as ações corretivas definidas estão sendo realizadas até seu fechamento e garantir que a solução corretiva é efetiva para solucionar a causa raiz dos problemas encontrados.
Produtos de trabalho:	• Ações corretivas.

APÊNDICE L – MAPA DA ÁREA TÉCNICA

1. TESTE UNITÁRIO

Objetivo da área de processo:	Determinar a corretude e completude de uma implementação com base nos requisitos da unidade.	
Referências:	• ANSI/IEEE Std. 1008 (1987). • ISO/IEC/IEEE 29.119-1 (2013).	
PRÁTICA	**SUBPRÁTICA**	**ARTEFATO**
1.1. Projetar teste unitário	• Identificar requisitos e procedimentos adicionais que precisam ser testados. • Identificar os tipos de dados de entrada e saída. • Projetar a arquitetura do conjunto de testes. • Especificar os procedimentos de teste requeridos. • Especificar os casos de teste.	• Projeto do teste unitário. • Especificação dos procedimentos de teste. • Especificação dos casos de teste.
1.2. Implementar teste unitário	• Obter e verificar os dados do teste. • Obter itens de teste • Implementar testes unitários.	• Dados de teste. • Recursos de apoio ao teste. • Configuração dos itens de teste. • Sumário de teste.
1.3. Executar teste unitário.	• Executar testes unitário • Determinar resultados do teste unitário.	• Log da execução de teste. • Sumário de teste. • Especificações de teste revisadas. • Dados de teste revisados.
1.4. Finalizar teste unitário.	• Avaliar finalização normal do teste unitário. • Avaliar finalização anormal do teste unitário. • Incrementar o conjunto de teste.	• Sumário de teste. • Especificações de teste revisadas. • Dados de teste adicionais

2. TESTE DE INTEGRAÇÃO

Objetivo da área de processo:	Avaliar a integração e relação entre os componentes de software para garantir que o sistema esteja consistente com sua arquitetura.	
Referências:	• ISO/IEC/IEEE 24.765 (2010) • IEEE 1012 (2012) • IEEE 1008 (1987) • ISO/IEC/IEEE 29.119-1 (2013) • ISO/IEC/IEEE 29.119-4 (2015) • NAIK e TRIPATHY (2008)	
PRÁTICA	**SUBPRÁTICA**	**ARTEFATO**
2.1. Estabelecer técnicas de teste de integração.	• Estabelecer granularidade do teste de integração. • Selecionar técnicas de projeto de teste de integração.	• Projeto de teste.
2.2. Projetar teste de integração.	• Derivar as condições do teste. • Derivar os itens de cobertura do teste. • Derivar casos de teste. • Montar os conjuntos de teste. • Estabelecer os procedimentos de teste.	• Especificações do projeto de teste.
2.3. Implementar teste de integração	• Implementar scripts de teste de integração.	• Scripts de teste de integração.
2.4. Executar teste de integração	• Executar scripts de testes de integração. • Determinar resultados do teste.	• Sumário de testes. • Resultados da execução.
2.5. Finalizar teste de integração.	• Avaliar finalização normal do teste de integração. • Avaliar finalização anormal do teste de integração. • Incrementar o conjunto de teste. • Executar pós-condições para os casos de teste.	• Sumário de teste. • Dados de teste.

3. TESTE DE SISTEMAS

Objetivo da área de processo:	Realizar testes em um sistema completo e integrado para avaliar a conformidade do sistema com seus requisitos especificados.
Referências:	• IEEE 610.12 (1990) • ISTQB Standard Glossary (2012) • ISO/IEC/IEEE 29.119-1 (2013) • ISO/IEC/IEEE 29.119-4 (2015) • IEEE 829 (2008) • FEWSTER e GRAHAM (1999) • HAYES (2004)

PRÁTICA	SUBPRÁTICA	ARTEFATO
3.1. Estabelecer tipos e técnicas de teste de sistema.	• Selecionar tipos de teste de sistemas. • Selecionar técnicas para o projeto de teste de sistemas. • Selecionar técnicas de geração de dados. • Selecionar técnicas para geração dos scripts.	• Projeto de teste.
3.2. Projetar teste de sistema.	• Derivar as condições do teste. • Derivar os itens de cobertura do teste. • Derivar casos de teste. • Montar os conjuntos de teste. • Estabelecer os procedimentos de teste.	• Especificações do projeto de teste.
3.3. Implementar teste de sistema.	• Implementar os scripts de teste de sistemas.	• Scripts de teste de sistemas.
3.4. Executar teste de sistema.	• Executar scripts de testes de sistema. • Determinar resultados do teste.	• Sumário de testes. • Resultados da execução.
3.5. Finalizar teste de sistema.	• Avaliar finalização normal do teste de sistemas. • Avaliar finalização anormal do teste de sistemas. • Incrementar o conjunto de teste de sistemas. • Executar pós-condições para os casos de teste.	• Sumário de teste. • Dados de teste.

3. TESTE DE ACEITAÇÃO

Objetivo da área de processo:	Conduzir testes formais para determinar se um sistema satisfaz ou não seus critérios de aceitação.
Referências:	• ISO/IEC/IEEE 29.119-1 (2013) • MPT.BR, (2011) • NAIK e TRIPATHY (2008)

PRÁTICA	SUBPRÁTICA	ARTEFATO
4.1. Estabelecer tipos e critérios de teste de aceitação.	• Definir tipo de teste de aceitação. • Definir critérios de aceitação.	• Projeto de teste.
4.2. Projetar teste de aceitação.	• Derivar as condições do teste. • Derivar os itens de cobertura do teste. • Derivar casos de teste. • Montar os conjuntos de teste. • Estabelecer os procedimentos de teste.	• Especificação do projeto de teste.
4.3. Implementar os testes de aceitação.	• Implementar os scripts de teste de sistemas.	• Scripts de teste de aceitação.
4.4. Executar teste de aceitação.	• Executar scripts de testes de aceitação. • Determinar resultados do teste.	• Sumário de testes. • Resultados da execução.
4.5. Finalizar teste de aceitação.	• Avaliar finalização normal do teste de aceitação. • Avaliar finalização anormal do teste de aceitação. • Incrementar o conjunto de teste de aceitação. • Executar pós-condições para os casos de teste.	• Sumário de testes. • Dados de teste.

APÊNDICE M – MAPA DA ÁREA DE SUPORTE

1. PLANEJAMENTO DO PROJETO DE TESTE

Objetivo da área de processo:	Estabelecer o planejamento da estratégia e atividades de automação de testes no projeto.
Referências	• TMMI (2012) • PMBOK (2013)

PRÁTICA	SUBPRÁTICA	ARTEFATO
1.1. Analisar riscos do produto.	• Definir categorias dos riscos. • Identificar riscos. • Analisar riscos.	• Estrutura analítica do risco – EAR. • Tabela de riscos do produto. • Matriz de exposição do risco.
1.2. Estabelecer estratégia de automação do teste.	• Identificar escopo da automação do teste. • Definir estratégia de automação de teste. • Definir critérios de entrada. • Definir critérios de saída. • Definir critérios de suspensão do teste. • Definir critérios de reinício do teste.	• Estratégia de automação de teste. • Critérios de automação de teste.
1.3. Estabelecer estimativas.	• Estabelecer Estrutura Analítica do Projeto – EAP. • Definir o ciclo de vida do teste. • Determinar estimativas de tamanho e custo.	• Estrutura analítica do projeto - EAP. • Ciclo de vida do projeto de automação; • Estimativas de tamanho do projeto de automação de teste. • Estimativa de custo do projeto de automação de teste.
1.4. Desenvolver o plano do projeto de automação de teste.	• Planejar cronograma. • Planejar recursos humanos. • Planejar envolvimento das partes interessadas do projeto. • Identificar os riscos. • Estabelecer o plano.	• Cronograma do projeto de automação de teste. • Matriz de recursos humanos. • Matriz de envolvimento das partes interessadas. • Planilha de riscos do projeto.

2. ACOMPANHAMENTO DO PROJETO DE TESTE

Objetivo da área de processo:	Avaliar se o progresso do projeto está ocorrendo de acordo com o planejado e tomar ações corretivas caso existam desvios significativos em relação ao progresso planejado, atividades ou outros aspectos do plano de teste.
Referências:	• TMMI, (2012) • PMBOK (2013) • ISO/IEC/IEEE 29.119-2 (2013)

PRÁTICA	SUBPRÁTICA	ARTEFATO
2.1. Monitorar o projeto com base no plano.	• Monitorar o progresso do projeto.	• Planilha de acompanhamento do projeto de teste.
2.2. Gerenciar ações corretivas.	• Identificar desvios do projeto. • Registrar ações corretiva. • Acompanhar ação corretiva até seu fechamento.	• Planilha de acompanhamento do projeto de teste.

3. GERÊNCIA DE CONFIGURAÇÃO

Objetivo da área de processo:	Estabelecer e manter a integridade do ambiente de automação seus produtos de trabalho no contexto do projeto de automação de teste.
Referências:	• CMMI (2010) • TMMI (2012) • ISO/IEC/IEEE 29.119-2 (2013)

PRÁTICA	SUBPRÁTICA	ARTEFATO
3.1. Estabelecer ambiente de automação de teste.	• Estabelecer requisitos do ambiente • Implementar ambiente de automação teste. • Manter o ambiente de automação de teste.	• Plano de gerência de configuração.
3.2. Estabelecer as baselines do projeto de automação de teste.	• Identificar itens de configuração. • Estabelecer o ambiente de gerência de configuração. • Gerar baselines e releases.	• Plano de gerência de configuração. • Sistema de gerência de configuração.
3.3. Acompanhar e controlar mudanças.	• Acompanhar solicitações de mudanças. • Controlar itens de configuração.	• Sistema de solicitação de mudanças.

4. MEDIÇÃO E ANÁLISE

Objetivo da área de processo:	Desenvolver e manter mecanismos de medição para apoiar as atividades de automação de testes no projeto.	
Referências:	• BASILI e WEISS (1984) • CMMI (2010) • TMMI (2012) • VAN SOLINGEN e BERGHOUT (1999)	
PRÁTICA	**SUB-PRÁTICA**	**ARTEFATO**
4.1. Estabelecer mecanismos de medição e análise.	• Estabelecer objetivos de medição da automação de teste. • Especificar medidas da automação de teste. • Especificar procedimentos de coleta e armazenamento das medidas. • Especificar procedimentos de análise das medidas.	• Plano de medição da automação de teste.
4.2. Fornecer os resultados da medição da automação de teste.	• Coletar dados da medição da automação. • Analisar dados da medição da automação. • Comunicar resultados. • Armazenar dados e resultados.	• Plano de medição da automação de teste.

5. REQUISITOS

Objetivo da área de processo:	Definir, analisar, estabelecer e manter os requisitos da automação de testes.	
Referências:	• IEEE 829 (2008) • IEEE 1012 (2012) • MPS.BR (2016)	
PRÁTICA	**SUBPRÁTICA**	**ARTEFATO**
5.1. Definir os requisitos da automação.	• Definir critérios objetivos para avaliação dos requisitos. • Elicitar requisitos da automação.	• Critérios de avaliação dos requisitos. • Requisitos da automação.
5.2. Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho.	• Determinar esquema de rastreabilidade entre requisitos e produtos de trabalho da automação. • Definir rastreabilidade bidirecional dos requisitos.	• Esquema conceitual de rastreabilidade • Matriz de rastreabilidade.
5.3. Gerenciar mudanças de requisitos da automação.	• Gerenciar mudança de escopo da automação de teste. • Manter rastreabilidade bidirecional dos requisitos da automação de testes.	• Solicitação de mudanças. • Análise de impacto. • Matriz de rastreabilidade.

6. GESTÃO DE DEFEITOS

Objetivo da área de processo:	Identificar, analisar e tratar os defeitos decorrentes da automação de teste.
Referências	• TMMI (2012)

PRÁTICA	SUB-PRÁTICA	ARTEFATO
6.1. Estabelecer o sistema de gestão de defeitos.	• Definir estratégia de gestão de defeitos. • Definir sistema de gestão de defeitos. • Manter sistema de gestão de defeitos.	• Máquina de estados do defeito. • Sistema de gestão de defeitos.
6.2. Analisar o resultado do teste.	• Registrar, classificar e priorizar os defeitos. • Analisar causa raiz dos defeitos selecionados.	• Registro do defeito. • Análise da causa raiz dos defeitos.
6.3. Estabelecer ações para eliminar a causa-raiz dos defeitos.	• Propor ações corretivas. • Acompanhar execução das ações corretivas.	• Ações corretivas

APÊNDICE N – ACORDO DE CONFIDENCIALIDADE

A EMPRESA está participando de um estudo de caso de implantação do Framework para Automação de Teste – FAST, como parte de um trabalho de pesquisa de doutorado. No sentido de prover suporte à comunicação aberta e livre ficam acordadas as seguintes condições:

1. Toda informação reunida ou derivada através dos instrumentos de pesquisa (como questionários), discussões, ou entrevistas serão tratados como confidenciais e não serão reportados para ninguém fora da equipe de avaliação com atribuição a projetos ou indivíduos.
2. Todos os resultados serão documentados e apresentados sem atribuição a indivíduos ou projetos específicos.

Nós, assinados abaixo, entendemos e estamos de acordo com estas condições.

APÊNDICE O – CRONOGRAMA DE EXECUÇÃO DO CASO 1

DATA	ATIVIDADE	OBJETIVOS	TEMPO	PARTICIPANTES
21/12/2016	Apresentação da proposta.	Apresentação dos objetivos e visão geral do estudo de caso através de uma reunião.	1h	P1C1
28/12/2016	Realização do diagnóstico	Análise do contexto de automação de um projeto da empresa para avaliar, com base nas áreas de processo e suas práticas, se o projeto	1h	P1C1
04/01/2017	Reunião de Kickoff	Apresentação do resultado do diagnóstico e cronograma de trabalho	1,5h	Toda equipe
11/01/2017	Execução dos Treinamentos	Área de Suporte --> Áreas de Processo: AS.1. Planejamento do projeto de teste. AS.2. Acompanhamento do projeto de teste. AS.3. Gerência de configuração.	1h	P1C1, P2C1, P3C1
18/01/2017	Execução dos Treinamentos	Área de Suporte --> Áreas de Processo: AS.4. Medição e análise. AS.5. Requisitos. AS.6. Gestão de incidentes.	2h	P1C1, P2C1, P3C1
25/01/2017	Execução dos Treinamentos	Área Técnica --> Áreas de Processo: AT.1. Teste unitário AT.2. Teste de integração AT.3. Teste de sistemas AT.4. Teste de aceitação.	1,5h	P2C1, P3C1
01/02/2017	Acompanhamento	Acompanhamento da implantação.	1,5h	P1C1, P2C1, P3C1
08/02/2017	Acompanhamento	Acompanhamento da implantação.	1h	P2C1, P3C1
15/02/2017	Acompanhamento	Acompanhamento da implantação.	1h	P2C1, P3C1
22/02/2017	Acompanhamento	Contagem dos pontos de função	1h	P3C1
08/03/2017	Acompanhamento	Acompanhamento da implantação.	1h	P2C1, P3C1
15/03/2017	Acompanhamento	Acompanhamento da implantação.	1h	P3C1
21/03/2017	Preparação para coleta de dados	Apresentação das métricas a serem coletadas para o estudo de caso.	1h	P2C1
29/03/2017	Realização da entrevista		3h	P2C1, P3C1

DATA	ATIVIDADE	OBJETIVOS	TEMPO	PARTICIPANTES
18/04/2017	Realização da entrevista		1h	P1C1
25/04/2017	Fechamento do estudo de caso	Coletar as métricas do estudo de caso.	-	

APÊNDICE P – CRONOGRAMA DE EXECUÇÃO DO CASO 2

DATA	ATIVIDADE	OBJETIVO	TEMPO	PARTICIPANTES
05/01/2017	Apresentação da proposta	Apresentação dos objetivos e visão geral do estudo de caso através de uma reunião.	1h	P1C2
12/01/2017	Realização do diagnóstico	Análise do contexto de automação de um projeto da empresa para avaliar, com base nas áreas de processo e suas práticas, se o projeto	1h	P1C2, P2C2, P3C2
19/01/2017	Reunião de Kickoff Execução dos treinamentos	Apresentação do resultado do diagnóstico e plano de trabalho Área de Suporte --> Áreas de Processo: AS.1. Planejamento do projeto de teste. AS.2. Acompanhamento do projeto de teste.	1h	P1C2, P2C2, P3C2
10/02/2017	Acompanhamento	Revisão da área de suporte.	1,5h	P2C2, P3C2
03/03/2017	Acompanhamento	Análise da introdução da automação.	1h	P2C2, P3C2
23/03/2017	Acompanhamento	Reunião de alinhamento para a retomada do projeto.	1h	P1C2, P2C2, P3C2
17/04/2017	Acompanhamento	Apresentar atividades principais que possam ser desenvolvidas no projeto.	1h	P1C2, P3C2
11/05/2017	Realização das entrevistas		1h	P3C2
18/05/2017	Realização das entrevistas		1h	P1C2
19/05/2017	Fechamento do projeto.	Recebimento dos indicadores e dados da equipe.	-	NA

APÊNDICE Q – DIAGNÓSTICO DO CASO 1

ÁREA TÉCNICA			
ÁREA DE PROCESSO	**PRÁTICA**	**STATUS**	**OBSERVAÇÕES**
1. TESTE UNITÁRIO	1.1. Projetar teste unitário.	Parcialmente	
	1.2. Implementar teste unitário.	Parcialmente	
	1.3. Executar teste unitário.	Parcialmente	
	1.4. Finalizar teste unitário.	Parcialmente	
2. TESTE DE INTEGRAÇÃO	2.1. Estabelecer técnicas de teste de integração.	Não Contemp.	Teste de integração é feito, mas não de maneira sistemática. O escopo do teste de integração é exploratório.
	2.2. Projetar teste de integração.	Não Contemp.	
	2.3. Implementar teste de integração.	Parcialmente	
	2.4. Executar procedimento de teste de integração.	Parcialmente	
	2.5. Finalizar teste de integração.	Não Contemp.	
3. TESTE DE SISTEMAS	3.1. Estabelecer tipos e técnicas de teste de sistema.	Não Contemp.	
	3.2. Projetar teste de sistema.	Não Contemp.	
	3.3. Implementar teste de sistema.	Não Contemp.	
	3.4. Executar teste de sistema.	Não Contemp.	
	3.5. Finalizar teste de sistema.	Não Contemp.	
4. TESTE DE ACEITAÇÃO	4.1. Avaliar escopo da aceitação.	Não Contemp.	
	4.2. Estabelecer abordagens e técnicas para o teste de aceitação.	Não Contemp.	
	4.3. Projetar teste de aceitação.	Não Contemp.	
	4.4. Implementar teste de aceitação.	Não Contemp.	
	4.5. Executar teste de aceitação.	Não Contemp.	
	3.6. Finalizar teste de aceitação.	Não Contemp.	

ÁREA DE SUPORTE			
ÁREA DE PROCESSO	**PRÁTICA**	**STATUS**	**OBSERVAÇÕES**
1. PLANEJAMENTO DO PROJETO DE TESTE	1.1. Analisar riscos do produto.	Não Contemp.	
	1.2. Estabelecer estratégia de automação do teste.	Não Contemp.	
	1.3. Estabelecer estimativas.	Parcialmente	É realizada estimativa de tamanho do item a ser desenvolvido, contemplando as atividades de automação do teste unitário. Estimativa não contempla automação nos níveis de integração, sistemas nem aceitação.
	1.4. Desenvolver o plano do projeto de automação de teste.	Não Contemp.	
2. ACOMPANHAMENTO DO PROJETO DE TESTE	2.1. Monitorar o projeto de teste com base no plano.	Parcialmente	Existe o acompanhamento sistemático do projeto de desenvolvimento, através de sprints mensais e *daily meetings*.
	2.2. Gerenciar ações corretivas.	Não Contemp.	
3. GERÊNCIA DE CONFIGURAÇÃO	3.1. Estabelecer ambiente de automação de teste no projeto.	Não Contemp.	A gerência de configuração é estabelecida e é sistematicamente mantida no projeto, apesar da prática não estar documentada. Não contempla a perspectiva do ambiente de automação do projeto.
	3.2. Estabelecer as baselines do projeto de automação de teste.	Não Contemp.	
	3.3. Acompanhar e controlar as mudanças.	Não Contemp.	
4. MEDIÇÃO E ANÁLISE	4.1. Estabelecer mecanismos de medição e análise.	Não Contemp.	
	4.2. Fornecer os resultados da medição da automação de teste	Não Contemp.	
5. REQUISITOS	5.1. Definir os requisitos da automação.	Contempla	Os requisitos do projeto estão registrados como tickets no readmine.
	5.2. Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho.	Não Contemp.	

ÁREA DE SUPORTE			
ÁREA DE PROCESSO	**PRÁTICA**	**STATUS**	**OBSERVAÇÕES**
	5.3. Gerenciar mudanças de requisitos da automação.	Não Contemp.	
6. GESTÃO DE DEFEITOS	6.1. Estabelecer o sistema de gestão de defeitos.	Parcialmente	Existe a execução da gestão de defeitos de maneira ad-hoc.
	6.2. Analisar o resultado do teste.	Não Contemp.	
	6.3. Estabelecer ações para eliminar a causa-raiz dos defeitos.	Não Contemp.	

APÊNDICE R – DIAGNÓSTICO DO CASO 2

ÁREA TÉCNICA			
ÁREA DE PROCESSO	**PRÁTICA**	**STATUS**	**OBSERVAÇÕES**
1. TESTE UNITÁRIO	1.1. Projetar teste unitário.	Parcialmente	
	1.2. Implementar teste unitário.	Parcialmente	
	1.3. Executar teste unitário.	Parcialmente	
	1.4. Finalizar teste unitário.	Parcialmente	
2. TESTE DE INTEGRAÇÃO	2.1. Estabelecer técnicas de teste de integração.	Não Contemp.	
	2.2. Projetar teste de integração.	Não Contemp.	
	2.3. Implementar teste de integração.	Não Contemp.	
	2.4. Executar procedimento de teste de integração.	Não Contemp.	
	2.5. Finalizar teste de integração.	Não Contemp.	
3. TESTE DE SISTEMAS	3.1. Estabelecer tipos e técnicas de teste de sistema.	Não Contemp.	
	3.2. Projetar teste de sistema.	Não Contemp.	
	3.3. Implementar teste de sistema.	Não Contemp.	
	3.4. Executar teste de sistema.	Não Contemp.	
	3.5. Finalizar teste de sistema.	Não Contemp.	
4. TESTE DE ACEITAÇÃO	4.1. Avaliar escopo da aceitação.	Não Contemp.	
	4.2. Estabelecer abordagens e técnicas para o teste de aceitação.	Não Contemp.	
	4.3. Projetar teste de aceitação.	Não Contemp.	
	4.4. Implementar teste de aceitação.	Não Contemp.	
	4.5. Executar teste de aceitação.	Não Contemp.	
	3.6. Finalizar teste de aceitação.	Não Contemp.	

ÁREA DE SUPORTE			
ÁREA DE PROCESSO	**PRÁTICA**	**STATUS**	**OBSERVAÇÕES**
1. PLANEJAMENTO DO PROJETO DE TESTE	1.1 Analisar riscos do produto.	Não contempla	
	1.2 Estabelecer estratégia de automação do teste.	Não contempla	
	1.3. Estabelecer estimativas.	Não contempla	
	1.4. Desenvolver o plano do projeto de automação de teste.	Não contempla	
2. ACOMPANHAMENTO DO PROJETO DE TESTE	2.1. Monitorar o projeto de teste.	Não contempla	
	2.2. Gerenciar ações corretivas até seu fechamento.	Não contempla	Não documentado.
3. GERÊNCIA DE CONFIGURAÇÃO	3.1. Estabelecer ambiente de teste.	Não contempla	Já existe uma estratégia de gerência de configuração para o projeto, mas não contempla a visão da configuração dos testes automatizados.
	3.2. Estabelecer as baselines do projeto.	Não contempla	
	3.3. Acompanhar e controlar as mudanças do projeto.	Não contempla	
4. MEDIÇÃO E ANÁLISE	4.1. Estabelecer mecanismos de medição e análise.	Não contempla	
	4.2. Fornecer os resultados da medição da automação de teste.	Não contempla	
5. REQUISITOS	5.1. Definir os requisitos da automação.	Não contempla	Requisitos do projeto são documentados como histórias na ferramenta trello. Rastreabilidade Ad-hoc.
	5.2. Definir rastreabilidade entre os requisitos da automação e seus produtos de trabalho.	Não contempla	
	5.3. Gerenciar mudanças de requisitos da automação.	Não contempla	
REA DE PROCESSO	**PRÁTICA**	**STATUS**	**OBSERVAÇÕES**
6. GESTÃO DE DEFEITOS	6.1. Estabelecer o sistema de gestão de defeitos.	Não contempla	
	6.2. Analisar o resultado do teste.	Não contempla	
	6.3. Estabelecer ações para eliminar a causa-raiz dos defeitos.	Não contempla	

APÊNDICE S – GLOSSÁRIO

Este apêndice apresenta o glossário utilizado para descrever brevemente alguns termos técnicos utilizados ao longo da definição do FAST.

Caso de teste: conjunto de pré-condições de casos de teste, entradas (incluindo ações, quando aplicável) e resultados esperados, desenvolvidos para direcionar a execução de um item de teste para atender aos objetivos de teste, incluindo a correta implementação, identificação de erros, qualidade de verificação e outras informações valiosas (ISO/IEEE 29.119-1, 2013, p. 7).

Condição de teste: é um aspecto testável de um componente ou sistema, como uma função, transação, recurso, atributo de qualidade ou elemento estrutural identificado como base para o teste, e podem ser usadas para derivar itens de cobertura do teste (ISO/IEC/IEEE 29.119-1, 2013, p. 7).

Conjunto de teste: conjunto de um ou mais casos de teste com a mesma limitação de execução (ISO/IEC/IEEE 29.119-1, 2013, p. 11).

Nível de integridade: a determinação do nível de integridade quantifica a complexidade, criticidade, risco, nível de segurança, performance desejada, confiabilidade ou outras características com base na importância do sistema para o usuário final. Níveis de integridade são utilizados para definição das atividades, rigor e nível de intensidade das atividades de verificação e validação (IEEE 1012, 2012, p 15).

Item de cobertura do teste: atributo ou combinação de atributos que são derivados de uma ou mais condições de teste através do uso das técnicas de projeto de teste que permite a medição detalhada da execução do teste (ISO/IEC/IEEE 29.119-1, 2013, pp. 7).

Procedimento de teste: uma sequência de casos de teste em ordem de execução quaisquer ações associadas que possam ser necessárias para estabelecer as pré-condições iniciais e quaisquer atividades de encerramento após a execução (ISO/IEEE 29.119-1, 2013, pp. 10).

Risco do produto: risco diretamente relacionado ao objeto de teste (TMMI, 2012, p. 210).

Risco do projeto: risco relacionado a gestão e acompanhamento do projeto de teste (TMMI, 2012, p. 210).

Scripts de teste: programa de computador com um conjunto de instruções para que uma ferramenta de teste possa atuar sobre as instruções (FEWSTER e GRAHAM, 1999, pp 83).

Tipos de teste: compreende um grupo de atividades de teste destinadas a testar um sistema focado em um objetivo de teste específico, como por exemplo, teste funcional, teste de usabilidade, regressão, etc. (ISTQB, 2012, p. 47).

Técnicas de teste: procedimento usado para derivar os casos de teste (ISTQB, 2012, pp. 43).

Testwares: são os procedimentos de teste que são executados, o software de apoio, os conjuntos de dados que são usados para executar os testes e a documentação de suporte (ERICSON et al., 1996, pp.10).