UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA

CIÊNCIA DA COMPUTAÇÃO

PEDRO PIRES BARRETO

Uma Proposta de Métricas Para Avaliar a Efetividade da Execução de Testes de Software

TRABALHO DE CONCLUSÃO DE CURSO

Recife

2018

PEDRO PIRES BARRETO

Uma Proposta de Métricas Para Avaliar a Efetividade da Execução de Testes de Software

Monografia apresentado ao curso de Ciência da Computação, como parte dos requisitos necessários à obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Ana Paula Carvalho Cavalcanti Furtado

Recife
2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Central, Recife-PE, Brasil

B273p Barreto, Pedro Pires.
Uma proposta de métricas para avaliar a efetividade da Execução de testes de software / Pedro Pires Barreto. – Recife, 2018.
79 f.: il.

Orientador(a): Ana Paula Carvalho Cavalcanti Furtado.
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal Rural de Pernambuco, Departamento de Estatística e Informática, Recife, BR-PE, 2019.
Inclui referências.

1. Software 2. Software – Medição 4. Teste de esforço I. Furtado, Ana Paula Carvalho Cavalcanti, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Pedro Pires Barreto às 10 horas do dia 23 de janeiro de 2019, a SALA 07 - CEAGRI II, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado " **Avaliação de efetividade da execução de casos de testes** ", orientado por Ana Paula Carvalho Cavalcanti Furtado e aprovado pela seguinte banca examinadora:

Ana Paula Carvalho Cavalcanti Furtado
DC/UFRPE

Lucas Albertins de Lima
DC/UFRPE

Dedico este trabalho à minha mãe Bel e ao meu pai Eli

Agradecimentos

Agradeço aos meus pais, Bel e Eli, que sempre fizeram tudo que estava a seu alcance para que eu pudesse atingir meus objetivos pessoais e profissionais da melhor maneira possível.

Agradeço à minha irmã por ser um grande exemplo de luta e resistência.

Agradeço a minha orientadora Ana Paula Cavalcanti, com quem tive muita afinidade em uma disciplina que a mesma lecionou e que foi peça fundamental desta pesquisa.

Agradeço aos meus amigos da universidade Dennys, Leonardo, Daniel, Rodrigo e Thomás que puderam compartilhar diversos momentos, felizes e difíceis durante essa longa jornada.

Agradeço aos amigos do colégio que estão comigo por mais de 15 anos me ensinando o significado da palavra amizade.

Agradeço aos participantes do grupo focal, que se dispuseram e gastaram um pouco do seu tempo com o intuito de contribuir com a pesquisa e acrescentando assim, uma melhor qualidade ao trabalho desenvolvido.

Resumo

O mercado exige software de qualidade e que respeite os prazos e os custos do projeto. Uma das principais preocupações da indústria de software é a garantia da qualidade do produto gerado, o que promoveu a busca pelo desenvolvimento de software aliada com conceitos definidos de qualidade de software. Muitas organizações têm investido no processo de testes buscando a prevenção e detecção de erros. Teste de software é um importante aliado à garantia da qualidade, o que explica uma parcela dos custos de desenvolvimento estar relacionada às atividades de teste. Durante todo o ciclo de desenvolvimento de software, os testes são executados com objetivo de garantir que os defeitos sejam minimizados ao máximo antes da entrega do produto ao cliente. O objetivo dessa pesquisa é desenvolver uma abordagem para avaliar as execuções de testes de software. Para alcançar esta meta foi utilizada a abordagem Goal-Question-Metric, que busca gerar um conjunto de métricas de acordo com os objetivos definidos pela necessidade da situação. A partir da formulação dos objetivos para avaliar uma execução de testes de software, foi proposta a adoção de um conjunto de métricas para facilitar o acompanhamento e melhoria da execução de testes de software. Para validar o conjunto de métricas propostas, foi conduzido um grupo focal com especialistas na área de teste de software. Assim esta pesquisa oferece contribuições sobre as métricas utilizadas para avaliar a execução de testes de software que atualmente significa uma parte com grande custo para empresas de software.

Palavras-chave: Testes de software, Métricas de testes, Medição de software e Estimativa de esforço de teste.

Abstract

The market demands high-quality software, which is delivered on time at the agreed cost. One of the main software industries' worries is the assurance of the created product quality that has generated the establishment of the software development associated with the main concepts of software quality. Many companies have been investing in the testing process to prevent and identify defects. Software testing is an important ally in quality assurance, which justifies a portion of the development costs being related to testing activities. During the development lifecycle and the software maintenance, tests are executed with the purpose of ensuring that the number of defects is minimized before the final product delivers to the client. The purpose of this research is to develop an approach to evaluate the execution of software test. To achieve this goal, it was used the Goal-Question-Metric approach, which seeks to generate a set of metrics according to the objectives defined by the need of the situation. After the definition of the objectives to evaluate a software test execution, it was proposed the adoption of a set of metrics to facilitate the monitoring and improvement of the software tests execution. To validate the set of proposed metrics, a focus group was conducted with specialists in the area of software testing. Thus, this research offers contributions to the metrics used to evaluate the execution of software tests that currently means a piece with a high cost for software companies.

Keywords: Software testing, Testing metrics, Software measurement and Estimation of testing effort.

Lista de ilustrações

Lista de quadros

Lista de abreviaturas e siglas

ACM	Association of Computing Machinery
CMM	Capability Maturity Model
CR	Change Request
EDC	Electronic Commerce Development
GQM	Goal/Question/Metric
GQS	Garantia da Qualidade de Software
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
ISO	International Organization for Standardization, ou Organização Internacional para Padronização, em português
TMM	Testing Maturity Model

Sumário

1 Introdução

Nas últimas décadas a nossa sociedade sofreu uma transformação com a introdução de sistemas de software. Sistemas de saúde, transporte, segurança, todos eles necessitam de softwares qualificados para garantir seu funcionamento. Com isso a indústria de software e a concorrência cresceram bastante principalmente a partir do século XXI exigindo cada vez mais que os desenvolvedores sejam capazes de produzir softwares de alta qualidade em prazos cada vez menores(AMMANN; OFFUTT, 2008).

Desta forma, cada vez mais existe uma preocupação das empresas de software com a produtividade da equipe, os custos dos projetos e com a satisfação dos clientes. Para garantir que esses aspectos sejam obtidos de forma eficiente, o primeiro passo a ser dado é incorporar qualidade ao produto. Um software de qualidade é um objetivo para qualquer empresa de desenvolvimento de software (SUWANNASART; SRICHAIVATTANA, 1999). A qualidade de software acaba se tornando de fundamental importância para criar uma relação de confiança entre os fornecedores e os clientes(QUINTELLA; QUEIROZ, 2006).

Teste de software é uma etapa de fundamental importância na garantia da qualidade de um produto(MIRANDA; ARANHA; IYODA, 2012). O impacto na indústria devido à insuficiência de testes de software é enorme. Entre eles pode-se destacar o aumento de falhas causados pela falta de qualidade, crescimento dos custos no desenvolvimento, aumento do tempo de comercialização devido a testes ineficientes e maiores custos nas transações de mercado (FAROOQ; DUMKE, 2008).

Teste de software pode ser definido como o processo de validar e verificar se um produto ou aplicação funciona como esperado e pode ser implementado com as características definidas nos requisitos técnicos e de negócio(MYERS; SANDLER; BADGETT, 2011). A verificação busca avaliar se as funções do software foram implementadas corretamente, enquanto a validação avalia se os requisitos informados pelo cliente foram implementados consistentemente no software. Além disso, os testes de software possuem dois objetivos principais (SOMMERVILLE, 2007):

- Demonstrar para o desenvolvedor e para o cliente que o software está de acordo com os requisitos definidos anteriormente. Para que isso ocorra, deve existir testes para cada funcionalidade do sistema.
- Descobrir situações que o comportamento do software está incorreto, indesejável, ou não está de acordo com a sua especificação.

As atividades de testes são responsáveis por mais de 40% do custo total de um projeto (SOMMERVILLE, 2007). Devido à grande quantidade de limitações existentes

num processo de teste como a busca por manter um equilíbrio entre tempo, custo e busca por perfeição, é impossível garantir que após a execução de um ciclo de testes, foram removidas todas as falhas de um sistema (FAROOQ; QUADRI, 2010). Desta forma, construir um modelo eficiente acaba sendo de fundamental importância para garantir qualidade e reduzir custos durante o processo de desenvolvimento de software(FAROOQ; GEORGIEVA; DUMKE, 2008).

Umas das formas de trabalhar no processo de medições é utilizando métricas. Métricas são definidas como padrões de medida, elas estão sendo utilizadas por muito tempo na indústria de TI como método para alcançar a efetividade e eficiência de uma atividade particular de um projeto (LAZIC; MASTORAKIS, 2008). Para garantir que essas medições ocorram de forma correta, é importante definir medida, que é identificar um valor da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo do processo ou produto (IEEE, 1990). Também é importante definir indicadores; um indicador compara a métrica com um resultado esperado. Desta forma, permite avaliar o status de um projeto em andamento, acompanhar riscos em potencial, descobrir áreas problemáticas, ajustar o fluxo de trabalho ou tarefas e avaliar a capacidade da equipe de controlar a qualidade dos produtos produzidos (VICENTE, 2010). Utilizando as medidas, métricas e indicadores de forma correta é possível antecipar problemas e conseguir um melhor controle de custos, redução de riscos, melhoria de qualidade (FLORAC; PARK; CARLETON, 1997). Todas essas vantagens tornam maiores as chances dos objetivos sejam alcançados. Nesse ambiente de rápidas e constantes mudanças onde a competitivade é extremamente alta e é cada vez mais fundamental trabalhar de maneira produtiva, eficiente e com alto nível de qualidade, o processo de medição se tornou uma parte importante e necessária nas organizações de desenvolvimento de software (SCHNAIDER et al., 2004).

Nesta pesquisa será proposto um conjunto de métricas que permitam avaliar a efetividade de uma das etapas do processo de testes de software, que é a execução. Esse conjunto de métricas permitiria a identificação de possíveis melhorias nesta etapa. O conjunto de métricas será definido após um estudo nos trabalhos relacionados. Para a elaboração do conjunto métricas, será utilizado o método GQM e após isso um grupo focal deve ser feito com profissionais na área para certificar que o conjunto proposto consegue atender as expectativas, sendo capaz de avaliar a efetividade da execução de testes no desenvolvimento de software.

1.1 Problema

Testes ineficientes ou a ausência dos mesmos já causaram diversas falhas em projetos de software, ocasionando problemas financeiros devido à incapacidade de alcançar as expectativas dos clientes. Segundo Bandeira (2008) falta de documentação,

incompetência dos engenheiros de software, falta de planejamento, domínio de software são alguns dos principais desafios encontrados na busca pela garantia da qualidade do software.

Uma solução para melhorar o processo de testes seria testar o sistema de forma completa, mas essa opção não se torna viável devido ao grande aumento de custo e de tempo durante o processo de desenvolvimento (FAROOQ; QUADRI, 2010). O processo de teste possui diversas limitações, entre elas pode-se destacar que o teste é apenas capaz de mostrar a presença de erros, mas não pode garantir a ausência deles (DIJKSTRA, 1972).

Desta forma, cada teste selecionado precisa ser executado de forma eficiente para que não existam falhas e grandes perdas nesta etapa. Características individuais do engenheiro de teste são cruciais para que a execução ocorra de forma esperada. Para que isso aconteça, é necessário que o engenheiro conheça e entenda o domínio do sistema a ser testado.

Além de necessidades individuais, é necessário um planejamento da equipe para que a execução aconteça de forma mais eficiente. A característica dos envolvimentos também influência na formação da equipe. Uma equipe com pouca diversidade de características, como idades, personalidades, experiência e habilidades comunicativas provavelmente irá apresentar mais problemas que uma equipe que consiga possuir todas essas características (KANIJ; MERKEL; GRUNDY, 2012).

Para garantir que a execução ocorra de forma eficiente é necessário a existência de um feedback constante. Esse feedback é apoiado pela área de trabalho informativa que pode fornecer instrumentos e dados sobre o andamento do projeto e de sua execução. Esses dados devem ser coletados através de métricas de software que devem medir o progresso, apontar falhas e qualidades durante a execução do projeto (VICENTE, 2010).

A comunidade de desenvolvimento de software continua buscando como garantir que uma métrica é adequada e aceitável para o propósito. Para isso é usado um sistema de regras que garantem o valor de uma métrica, que é conhecido como validação de métricas de software. Pesquisadores de engenharia de software têm debatido o que constitui essa validação por quase 50 anos, mas, ainda não chegaram a um consenso sobre o sistema de regras (MENEELY; SMITH; WILLIAMS, 2013).

Algumas dificuldades surgem durante a utilização das métricas. A principal delas é como analisá-las, pois a métrica sem a correta análise acaba trazendo um dado inútil e possivelmente gerando informações que podem prejudicar o andamento do estudo (JONES, 1994). Desta forma é necessário compreender os objetivos para interpretar os dados para transformá-los em informações úteis para a equipe.

Kaner e Bond (2004) ilustram a importância de definir de forma exata a semântica de uma métrica. Problemas de significado de métricas podem trazer diversos riscos e armadilhas para o processo de medição. Essas inconsistências de definição atrapalham no processo de melhoria, uma vez que a comparação dos resultados é afetada. Desta forma, é perceptível que é difícil de identificar e usar as métricas apropriadas para esse tipo de avaliação.

Assim, diante desses problemas apresentados, essa pesquisa irá apresentar um conjunto de métricas capaz de ajudar na busca por uma maior eficiência da execução de testes de software.

1.2 Justificativa

Medição está no cerne de muitos sistemas que são importantes nas nossas vidas. Medições econômicas determinam o aumento de custos e ganhos. Sem medições, tecnologias possuem grande dificuldade de funcionamento(CHEN; PROBERT; ROBESON, 2004).

Medição possui um papel crítico na eficiência e na efetividade do processo e na execução de testes. A exigência para que os sistemas se tornem cada vez maiores, mais robustos e seguros continua crescendo. Consequentemente, a cobrança por melhor qualidade no gerenciamento dos testes também. As métricas permitem que o gerente de projeto consiga tendências, antecipe problemas, garanta um melhor controle de custos, reduza riscos, melhore a qualidade e garanta que os objetivos foram alcançados (LAZIC; MASTORAKIS, 2008)

O processo efetivo de medição é capaz ajudar os engenheiros de testes permitindo que eles acompanhem a capacidade da equipe. Desta forma, tornando possível o desenvolvimento de planos possíveis para a equipe e para garantir a qualidade dos testes e consequentemente do produto.

O trabalho de Chen, Probert e Robeson (2004) propôs alguns conjuntos de métricas para acompanhar as estratégias de testes e visualizar a evolução. No artigo de Lazic e Mastorakis (2008) foi discutido métricas de software e suas capacidades para mostrar objetivamente as evidências necessárias para melhorar o processo de testes em empresas de desenvolvimento.

Já Unterkalmsteiner et al. (2012) fez uma revisão sistemática para analisar a avaliação e medição do processo de melhoria de softwares, A revisão incluiu mais de 148 publicações. Também foi observado que existe um foco em processos de software e alguns problemas foram levantados. Além disso, listas de indicadores de sucesso mais utilizados e atributos de qualidade foram identificados podendo ajudar na seleção de métricas da nossa pesquisa.

1.3 Objetivo

A pesquisa busca identificar os modelos de avaliação de softwares mais utilizados na literatura, identificando suas qualidades e problemas. Desta forma, O objetivo principal deste trabalho é criar um conjunto de métricas capaz de avaliar de forma inicial a efetividade da execução de casos de teste durante o processo de desenvolvimento de software.

1.4 Contribuições

Esta pesquisa contribui para testes de softwares com os seguintes resultados:

- Criação de métricas capazes de avaliar a qualidade da execução de casos de testes durante o processo de desenvolvimento de software.
- Validação das métricas selecionadas por engenheiros de testes que atuam em empresas de TI no mercado trabalho.

1.5 Estrutura do Trabalho

Este trabalho está dividido em 7 capítulos, além da introdução e uma visão geral do trabalho apresentado no capítulo 1. No capítulo 2, a metodologia proposta no trabalho é descrita, além das premissas e dos critérios estabelecidos para sua definição. Já no capítulo 3 temos uma breve revisão sobre testes de software e medição de software, incluindo seus principais conceitos e etapas. O capítulo 4 aborda os trabalhos relacionados, citando cerca de 10 publicações com temas similares aos abordados nessa pesquisa, para melhor conhecimento do assunto e seus problemas. No capítulo 5, são apresentadas as métricas escolhidas durante a pesquisa. O capítulo 6 apresenta o planejamento e os resultados do grupo focal feito para analisar as métricas definidas durante a pesquisa. Por fim, no capítulo 7, temos as contribuições deste trabalho e as perspectivas de trabalhos futuros.

2 Metodologia

Este capítulo descreve como o estudo foi executado durante todas as fases deste pesquisa. De acordo com Gil (2002), as pesquisas exploratórias têm como principal finalidade desenvolver e elucidar conceitos, tendo em vista a formulação de problemas mais precisos ou hipóteses para estudos posteriores.

2.1 Método da Pesquisa

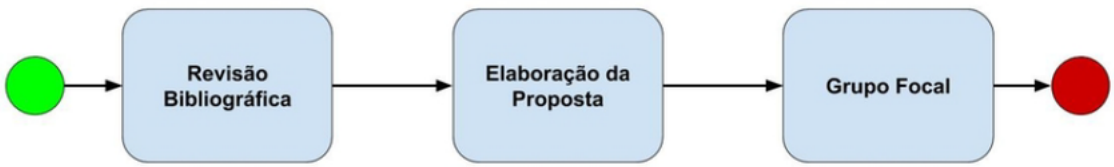
No que diz respeito à abordagem de investigação, esta pesquisa seguiu um comportamento qualitativo. Segundo Terence e Escrivão Filho (2006/10) o propósito de pesquisas qualitativas são aprimorar o conhecimento do tema de acordo com a visão dos praticantes que adquiriram experiências diretas com o problema da pesquisa sem se preocupar com a representação numérica.

Segundo Miles e Huberman (1994), as abordagens qualitativas têm o objetivo de melhorar a compreensão das percepções locais para explicar as formas como as pessoas, em cenários particulares, entendem, contabilizam, agem e gerenciam as suas atitudes cotidianas.

2.2 Fases da Pesquisa

A pesquisa foi dividida em 3 principais etapas conforme mostrado na Figura1 .

Figura 1 – Etapas da pesquisa



Elaborado pelo autor

2.2.1 Revisão bibliográfica

Foi realizada uma revisão da literatura acerca de artigos e teses sobre os testes de software e métricas de software, a fim de criar um referencial teórico dos principais tópicos que abordam este trabalho, assim como buscar por trabalhos relacionados.

Durante a execução desta atividade, foram utilizadas ferramentas de busca que contribuíram para que fossem encontrados trabalhos relevantes ao estudo:

- Google Acadêmico;
- IEEE Xplore Digital Library;
- ACM Digital Library;

O Google Acadêmico fornece uma maneira simples de pesquisar amplamente a literatura acadêmica (GOOGLE, 2018). Assim como o Google tradicional, apresenta os resultados de forma ordenada com base na relevância dos documentos.

O IEEE Xplore Digital Library é um recurso poderoso para a descoberta de conteúdo científico e técnico publicado pelo IEEE (Institute of Electrical and Electronics Engineers) e seus parceiros editoriais. Fornece acesso a mais de quatro milhões de documentos de texto completos de algumas das publicações mais citadas do mundo em engenharia elétrica, informática e eletrônica (IEEE, 2018).

Os principais termos pesquisados foram: “software testing”, “software metrics”, “testing metrics”, “software measurements”, “test management” e “test measurement”. Trabalhos muito antigos, que foram publicados em um ano inferior a 2005, passaram por uma análise antes de serem utilizados. Essa análise foi feita com o intuito de evitar trabalhos com argumentos antigos e ultrapassados.

2.2.2 Elaboração da proposta

A partir do levantamento realizado na etapa anterior, foi desenvolvido a proposta deste trabalho. Utilizando o GQM, proposto por (BASILI; CALDIERA; ROMBACH, 1994), foi gerado um conjunto de métricas para ajudar a avaliar a efetividade da execução de casos de testes.

2.2.3 Grupo focal

Grupos focais é uma técnica de pesquisa que coleta dados por meio das interações grupais ao se discutir um tópico especial sugerido pelo pesquisador. Como técnica, ocupa uma posição intermediária entre a observação participante e as entrevistas em profundidade. Pode ser caracterizada também como um recurso para compreender o processo de construção das percepções, atitudes e representações sociais de grupos humanos (GONDIM, 2002).

Idealmente, os integrantes não devem pertencer a um mesmo círculo de amizade ou trabalho. Isto visa evitar que a livre expressão de ideias no grupo seja prejudicada pelo temor do impacto (real ou imaginário) que essas opiniões podem causar posteriormente. O recrutamento dos participantes ocorre em função do grupo social

a ser estudado, devendo abranger sua variabilidade (faixa etária, gênero ou classe social)(BORGES; SANTOS, 2005).

O objetivo do grupo focal nesse trabalho foi avaliar a proposta desta pesquisa, com intuito de coletar sugestões de melhoria, para isso foi utilizado as etapas descritas por Kontio (2004):

1) Definição do problema de pesquisa
2) Planejamento
3) Seleção dos participantes
4) Execução do grupo focal
5) Análise dos dados

2.3 Considerações finais

Neste capítulo foi apresentada a metodologia da pesquisa utilizada por este trabalho. Foram apresentados também os passos necessários para realizar a revisão da literatura, a elaboração da proposta de acordo com o GQM e o planejamento e execução do grupo focal que validou a proposta produzida na etapa anterior.

3 Revisão Bibliográfica

Neste capitulo será apresentado a fundamentação teórica dos principais tópicos que abordam este trabalho, com objetivo de descrever os principais conceitos e estados da arte relacionados ao tema da pesquisa, que trata de testes de software e sua relação com a qualidade do software e as metricas de software para avaliar a qualidade dos software.

3.1 Qualidade de Software

Qualidade no dicionário Michaelis (2018) é definido como “Atributo, condição natural, propriedade pela qual algo ou alguém se individualiza; Conjunto de características que fazem parte da personalidade de um indivíduo e que o diferenciam de todos os outros”.

A qualidade não é mais um diferencial para empresas de software no mercado atual. Ela se tornou indispensável para qualquer grupo que busque competir com sucesso no mercado atual satisfazendo seus clientes.

O conceito de qualidade tem mudado durante os anos, recebendo novos valores. PHILLIP CROSBY (1979) definiu qualidade como a “conformidade com os requisitos”. Essa definição exige que os requisitos estejam bem definidos para que não exista nenhum tipo de engano durante o processo de validação de requisitos. Com esta definição, para ter um produto de qualidade, os requisitos devem ser mensuráveis e os requisitos do produto serão atingidos ou não. Desta forma, a qualidade é um estado binário, ou seja, um produto é considerado de qualidade ou não. Os requisitos podem ser completos ou podem ser simples, mas contanto que sejam mensuráveis, pode ser determinado se a qualidade foi ou não alcançada.

Outro conceito de qualidade é o que define como “aptidão para uso”. Esse conceito tem grande relação com a expectativa de o cliente e a forma como ele irá utilizar o produto (JURAN, 1991). Para isso é importante que exista uma descrição da finalidade do produto, normalmente disponível em um documento de requisitos. Os requisitos são de extrema importância e o sistema de qualidade gira em torno dele.

Já a SOARES (2017 apud ABNT ISO 10015, 2001) qualidade é o grau em que um conjunto de características inerentes de um objeto satisfaz um objeto.

Sommerville (2007) também fala que a qualidade de um software não está apenas relacionada com a funcionalidade do software estar implementada de forma correta, além disso, a qualidade depende de atributos não funcionais, mostrados no quadro 1. Não é possível que um sistema possua boas características para todos os atributos, por exemplo, o aumento de robustez acaba trazendo uma perda de

desempenho. É importante que exista um plano de qualidade que defina os atributos mais importantes para o software que está em desenvolvimento.

Quadro 1 – Atributos de qualidade de software

Proteção	Compreensibilidade	Portabilidade
Segurança	Testabilidade	Usabilidade
Confiabilidade	Adaptabilidade	Reusabilidade
Resiliência	Modularidade	Eficiência
Robustez	Complexidade	Aprendizagem

Elaborado pelo autor

3.2 Testes de Software

Essa seção busca identificar os conceitos de testes de software, as etapas do processo de teste durante o desenvolvimento de software, além das técnicas e ferramentas mais mencionadas na literatura.

3.2.1 Conceitos

A atividade de teste de software desempenha um papel central em atividades de garantia de qualidade de software (GQS) (TIAN, 2005). Segundo (MYERS; SANDLER; BADGETT, 2011) uma definição apropriada é “Testar é o processo de executar um programa com a intenção de achar erros“. Essa definição é importante, pois destaca a importância de achar erros. Se a busca tiver como intenção mostrar que o software não possui erros, a equipe será levada subconscientemente para esse objetivo, tendendo a selecionar dados de teste que tenham baixa probabilidade encontrar falhas.

Uma definição formal de teste dada pelo IEEE (1990) é:

> “O processo de operar um sistema ou um componente sobre as condições especificadas, observando ou gravando os resultados, e fazendo uma evolução de alguns aspectos do sistema ou componente”.

Teste de software é o processo de executar o software de uma maneira controlada com o objetivo de encontrar defeitos antes de colocar o sistema em uso e avaliar se ele se comporta conforme especificado (SOMMERVILLE, 2007). O processo de testes então possui 2 objetivos distintos:

- Mostrar para desenvolvedores e clientes que o software está de acordo com os requisitos.
- Descobrir casos que o comportamento do sistema é incorreto, indesejável ou não está de acordo com as especificações.

De acordo com Pressman (2006) a atividade de testes é um elemento importante na garantia da qualidade e representa a última revisão de especificação, projeto e codificação. Desta forma ele sugeriu alguns princípios de testes:

1) Todos os testes devem ser rastreáveis aos requisitos do cliente.
2) Os testes devem ser planejados bem antes do processo de execução.
3) Os testes devem começar nos componentes individuais, com o andamento do processo deve-se começar a testar a integração dos componentes e no fim se testa o sistema na totalidade.
4) É impossível executar todas as combinações de possibilidades durante o processo de testes.

Testes é parte dos processos de verificação e validação de software. Os processos de verificação e validação dizem respeito à checagem de que o software desenvolvido atende às suas especificações e oferece as funcionalidades esperadas pelas pessoas que pagam por ele. O processo de verificação confirma que o software atende às suas especificações técnicas. Uma “especificação” é uma descrição de uma função como um valor de saída mensurável dado um valor de entrada específico em condições prévias específicas. Conforme a MPS.BR SOFTEX (2016) este processo consiste em:

- Identificação dos produtos a serem verificados;
- Desenvolvimento e implementação de uma estratégia de verificação;
- Estabelecimento de um ambiente de verificação e identificação de critérios e procedimentos para verificação dos produtos;
- Execução das atividades de verificação;
- Identificação e registro de defeitos;
- Analise e disponibilidade dos resultados da atividade de verificação.

Já a validação confirma que o software atende aos requisitos da empresa e principalmente as expectativas do cliente (HUTCHESON, 2003). Para o MPS.BR SOFTEX (2016) o processo de Validação consiste em confirmar que um produto atenderá os objetivos de uso, quando colocado no ambiente de produção. Os resultados esperados com estes processos são:

- A identificação dos produtos a serem validados;
- Desenvolvimento de uma estratégia de validação;
- Estabelecimento de um ambiente de validação e identificação de critério e procedimentos de validação;
- Execução da atividade de validação para comprovar que os produtos estão prontos para serem usados;
- Identificação e registro de problemas;
- Disponibilidade e análise dos resultados da atividade de validação;
- Fornecimento das evidências de que o software desenvolvido está pronto para o uso.

Outra forma de definir é (PRESSMAN, 2006):

- Verificação: Estamos produzindo o produto corretamente?
- Validação: Estamos produzindo o produto correto?

3.2.2 Atividades de teste

O processo de teste pode ser descrito como fases genéricas do processo ou como uma série de vários níveis de teste. Depois de bem estabelecido o processo de teste pode trazer diversos benefícios para os envolvidos, entre eles: permite que a atividade de teste se torne compreensível, ensinável e gerenciável, além de tornar as atividades mais consistentes e maduras permitindo a reutilização (TIAN, 2005) (FAROOQ; GEORGIEVA; DUMKE, 2008).

A atividade de testes pode ser dividida em 3 etapas como mostra a figura 2:

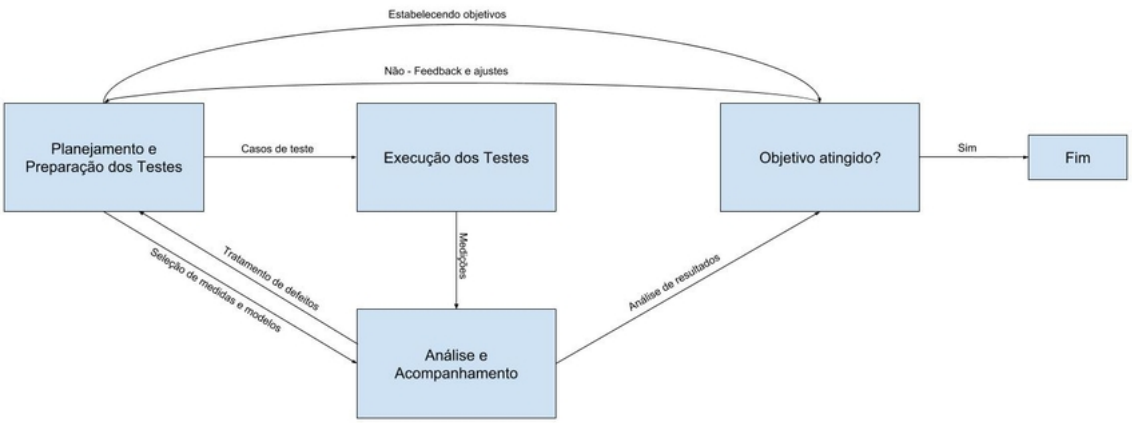
1) Planejamento e preparação dos testes - Estabelece as metas dos testes e os recursos necessários para implementá-los. Além disso, é importante identificar os requisitos dos testes e selecionar uma estratégia de teste global e o procedimento de teste geral. É importante destacar que o planejamento não é

estático, ele deve ser iniciado de forma mais rápida possível, mas deve estar em constante evolução de acordo com o amadurecimento do projeto. Nesta etapa os casos de testes também são projetados, sendo descritos os passos a serem seguidos e os resultados esperados. Para que a preparação dos testes seja conduzida de forma correta, o planejamento deve ser bem descrito. Desta forma, os casos de teste serão escritos de acordo com a necessidade e os estágios corretos. Existindo qualquer alteração nos requisitos, é necessário avaliar qual caso de teste está relacionado ao requisito e fazer alterações para que ele esteja de acordo com a mudança.

2) Execução dos casos de teste - A execução dos testes deve ocorrer de acordo com o planejamento, cada caso de teste deve ser executado de acordo com o planejado. Essa execução pode ocorrer tanto de forma manual como de forma automática.

3) Avaliação dos testes por meio da comparação do resultado dos testes - Nesta etapa existe a verificação e análise de resultados para determinar se uma falha foi observada. Em caso afirmativo, as atividades de seguimento são iniciadas e monitoradas para garantir a remoção das causas subjacentes ou falhas que levaram às falhas observadas em primeiro lugar. Se forem encontrados diversos erros graves, observa-se que o produto possui uma qualidade questionável. Se os erros forem encontrados em menor número e principalmente com menor nível de criticidade, então é dito que o produto possui uma confiabilidade ao menos aceitável.

Figura 2 – Processo de testes



Elaborado pelo autor

3.2.3 Testes manuais x Testes automáticos

A diferença entre os testes manuais e os testes automáticos são discutidos nessa seção. A execução sem utilização de ferramentas de suporte é conhecida como teste manual. Quando utilizado alguma ferramenta de suporte e executando os testes com a ajuda de um suporte de automação é chamado de testes automáticos.

3.2.3.1 Testes Manuais

Teste manual é o processo onde uma equipe executa os testes de forma manual, comparando os resultados esperados com os resultados atuais buscando encontrar erros do software. Desta forma o testador acaba utilizando o software de forma similar que o usuário final utilizaria (MAILEWA; HERATH; HERATH, 2015).

Dentre as vantagens de utilizar testes manuais podemos citar:

- Testes manuais são mais flexíveis que testes automáticos. Os testes automáticos são definidos e programados de acordo com diversas regras. Quando mudanças ocorrem no projeto, o processo precisa ser reprogramado e rodado novamente, enquanto com testes manuais essas mudanças são facilmente incorporadas na rotina de testes.
- O teste manual faz com que o programa seja utilizado da mesma forma que o usuário final usaria. O usuário pode utilizar o sistema de forma única, trazendo erros que são mais fáceis de serem escapados em testes automáticos.
- Ferramentas de automação de testes são caras e exigem um certo tempo para serem lucrativas. Em termos de custos de curto prazo, os testes manuais são mais baixos que os testes automáticos.

3.2.3.2 Testes automáticos

Teste automático é o processo onde ferramentas de automação executam testes que repetem ações predefinidas, comparando os resultados esperados com os encontrados. Se os resultados são os esperados, o sistema está se comportando de forma esperada, se não provavelmente existe um erro que deve ser analisado e possivelmente corrigido (TIAN, 2005).

As principais vantagens dos testes automáticos são:

- Enquanto a configuração inicial dos testes automáticos pode ser um pouco demorada, depois de feita a execução automática costuma ser mais rápida que a manual, permitindo o reuso dos testes.

- Os testes automáticos são mais rentáveis a longo prazo, pois são capazes de executar mais testes que em execução manual e se forem produzidos de forma correta, vão encontrar mais erros.
- Testes automatizados são mais adequados quando o projeto é grande e possui muitos usuários.

3.2.4 Fases de Teste de Software

Durante a produção de um software grande é importante distinguir as fases de testes, da mesma forma que é divido as fases de construção de software. Em cada fase de teste, existe um foco em determinados aspectos consequentemente é necessário escolher diferente estratégias de seleção de testes e medidas de cobertura de teste (LINNENKUGEL; MÜLLERBURG, 1990).

3.2.4.1 Teste de Unidade

Os testes de unidade buscam fazer verificação dos menores elementos possíveis de serem testados. Eles são testados de forma separada buscando garantir que todas as especificações sejam atendidas. A descrição do projeto é usada como guia, caminhos de controle importantes são testados para descobrir erros dentro dos menores elementos do software (PRESSMAN, 2006). O objetivo dessa fase é revelar defeitos de lógica e implementação em cada unidade, garantindo que a unidade testada funciona corretamente de forma isolada. Normalmente esses testes são realizados logo após a implementação e são executados pelos próprios desenvolvedores que acessam o código fonte.

Teste de software é uma área bastante custosa e que demanda muito tempo para ser executada, então é importante escolher casos de teste de unidade efetivos. De acordo com Sommerville (2007), efetividade significa: Os casos de teste devem mostrar que quando usados de forma correta, o componente testado faz o que é projetado para ser feito. Se existem defeitos no componente, ele deve ser revelado pelo caso de teste.

3.2.4.2 Testes de Integração

Após testar os elementos de forma individual, o teste de integração é realizado buscando garantir a construção da estrutura do sistema. É nesta fase que as partes do software são integradas e o funcionamento entre elas ocorre de forma esperada e sem erros.

Existem duas categorias de integração, a não incremental e a incremental. Na não incremental os módulos são combinados antecipadamente e o programa é testado

na totalidade. Na incremental, o programa é desenvolvido e testado em pequenas partes, desta forma, os erros são mais fáceis de serem isolados e corrigidos(DOMINGUES, 2002).

3.2.4.3 Testes sistêmicos

O teste do sistema é uma série de testes diferentes, cujo objetivo principal é o exercício completo do sistema baseado em computador. Embora cada teste tenha uma finalidade diferente, todos trabalham para verificar se os elementos do sistema foram devidamente integrados e desempenham de forma correta suas funções. Além disso os testes sistêmicos analisam os requisitos não funcionais como segurança, desempenho e confiabilidade e normalmente são executados por uma equipe de testes (PRESSMAN, 2006).

3.2.4.4 Testes de Aceitação

Este é o estágio final do processo de teste antes do sistema ser aceito para uso operacional. O sistema é a testado com dados fornecidos pelo cliente do sistema e não com dados de teste simulados. O teste de aceitação pode revelar erros e omissões na definição dos requisitos do sistema, porque o dados reais exercitam o sistema de diferentes maneiras a partir dos dados de teste. Testes de aceitação também podem revelar problemas de requisitos onde as instalações do sistema realmente não atende às necessidades do usuário ou o desempenho do sistema é inaceitável (SOMMERVILLE, 2007).

3.3 Medição de Software

De acordo com Rocha, Souza e Barcellos (2012), medir está presenta em diversos aspectos da vida humana. Medidas são essenciais para o conhecimento, o controle e a tomada de decisão em diversos casos de nossas vidas. Mede-se tamanho e peso das pessoas e objetos para melhor defini-los.

Para Sommerville (2007), uma métrica de software é qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada. Para muitos, isto significa avaliar o profissional e, pelo contrário, as métricas servem para coletar dados para analisar e propor melhorias durante o processo de desenvolvimento. Um dos princípios da métrica de software é especificar a coleta de dados para análise, de modo a avaliar o desempenho e analisar pontos de melhoria no projeto.

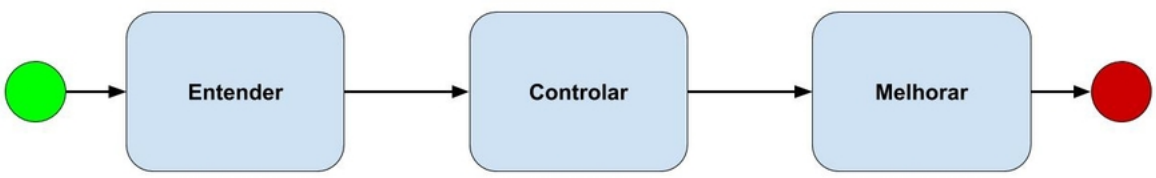
Para realizar medições de software de forma qualificada, é necessário um programa de medição bem-planejado. Os fatores envolvidos são bastante complexos,

existindo características para serem medidas, mas devido ao grande custo, é necessário selecionar conjuntos restritos para viabilizar a análise. Esse programa de medição precisa ser fortemente fundamentado. Isso normalmente não ocorre nas organizações, que acabam definindo programas de medições incapazes de produzir informações uteis as suas necessidades (ROCHA; SOUZA; BARCELLOS, 2012).

Outra dificuldade encontrada acontece na definição e padronização do processo de coleta e análise dos dados. É necessário criar e documentar procedimentos, padronizar os documentos de coleta e análise e principalmente estabelecer critérios claros de medição. Se os critérios não forem bem estabelecidos, diferentes interpretações das regras irão influenciar os resultados obtidos e consequentemente a confiabilidade da medição será afetada (BANDEIRA, 2008).

Em geral, medição é o processo onde números ou símbolos são atribuídos para atributos de entidades, descrevendo em regras claramente definidas. A medição de software é o processo contínuo de definir, coletar e analisar dados no processo de desenvolvimento de software e no produto produzido para entender e controlá-los, trazendo informações uteis para melhorar o processo e o produto (BERGHOUT; SOLINGEN, 1999). A medição é utilizada para trazer informações que podem ser aplicadas para três propósitos diferentes. Primeiramente, os dados dão visibilidade, isto é importante pois aumenta o entendimento do processo e do produto, com o entendimento é possível determinar as diversas variáveis que existem durante o processo. Após o entendimento os dados coletados e analisados são usados para controlar o processo e os produtos. Por último, baseado nas análises, os dados coletados são usados para avaliar o processo, desta forma, melhorias podem ser feitas mudando as variáveis do processo e suas relações. A ordem da medição é ilustrada na figura 3.

Figura 3 – Três propósitos da medição



Elaborado pelo autor

Segundo (IEEE, 1990), a utilização de medições diminui a subjetividade na avaliação da qualidade de software. As métricas são capazes de fornecer informações

quantitativas sobre o produto e o processo.

A coleta de métricas é importante para acompanhar o andamento de um processo, sabendo se ele está caminhando como planejado e os objetivos estão sendo atingidos. O sucesso das métricas consiste em especificar os objetivos e metas a serem alcançados. Além de identificar o principal objetivo da criação de métricas que é fornecer informações necessárias para apoiar as decisões gerências durante o processo de testes.

A escolha das métricas utilizadas em cada projeto deve ser feita com critérios bem fundamentados, devido ao grande número de opções disponíveis e ao alto custo para levantar essas medidas. A escolha adequada acaba sendo um fator fundamental para o sucesso de uma política de medição. Deve ser feita de forma cuidadosa e seguindo critérios claros para que não exista a escolha arbitraria de métricas (BORGES, 2003).

Para o guia da MPS.BR (SOFTEX, 2016) o propósito das medições é obter, observar e relatar as informações referentes aos produtos desenvolvidos e aos processos executados na organização, de forma a apoiar seus objetivos. Os resultados esperados são:

- Determinação dos objetivos de medição a partir dos objetivos da organização e das necessidades de informação de processos técnicos e gerencias;
- O grupo de medidas ajustadas com os objetivos de medição é definido, priorizado, documentado, revisado e atualizado;
- Especificar os procedimentos para a análise da medição existente;
- As informações exigidas são obtidas e estudadas;
- Guardar os dados e os resultados;
- As informações produzidas são utilizadas para apoiar decisões, estando disponíveis em uma base para que todos os interessados possam ter acesso.

3.3.1 Conceitos Gerais

Medidas, métricas e indicadores são conceitos fundamentais na área de métricas de software, comumente eles são confundidos e utilizados de forma errada. Apesar da similaridade, eles são diferentes e devem ser definidos de forma correta.

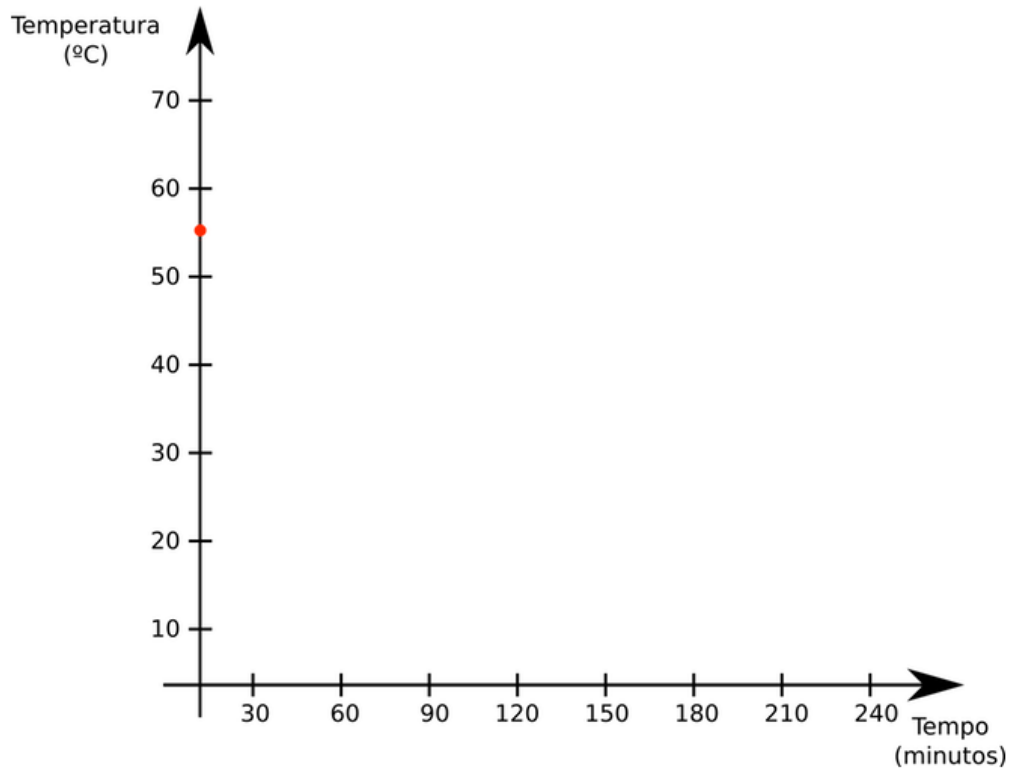
Segundo o IEEE, uma medida é uma avaliação em relação a um padrão (ANSI/. . . , 1983). Outra definição é que uma medida é um mapeamento entre um atributo empírico

e uma escala matemática (KITCHENHAM; PFLEEGER; FENTON, 1995). As medidas são utilizadas para atender os seguintes objetivos (HUMPHREY, 1989):

- Obter mais um conhecimento de um item ou processo;
- Verificar se o produto ou processo atende aos critérios de aceitação;
- Acompanhar e controlar uma atividade;
- Gerar indicadores de tendências ou estimativas.

Medição é o ato de medir, de determinar uma medida. É o conjunto de operações com o objetivo de determinar o valor de uma medida. Exemplificando, a temperatura de uma determinado ser em um momento específico é uma medida. O gráfico 4exemplifica a medida da temperatura.

Figura 4 – Medida de temperatura em um determinado tempo



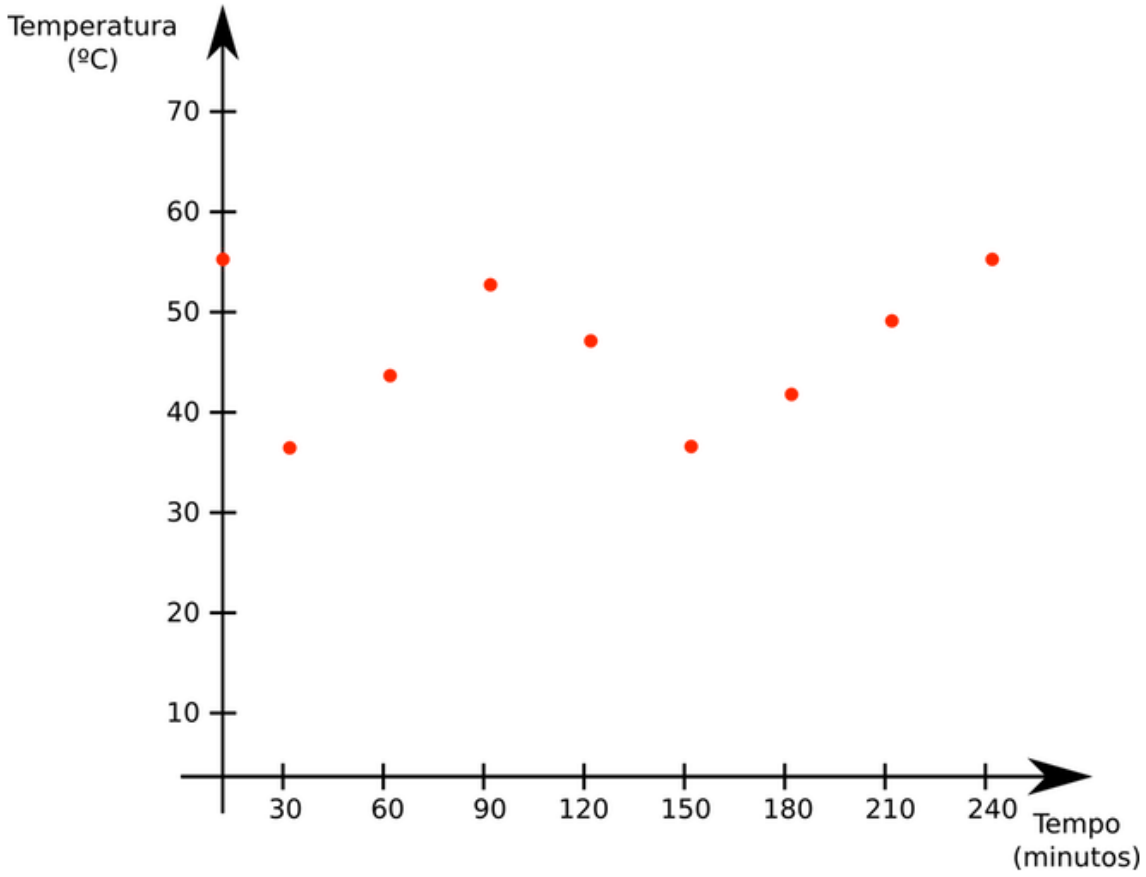
Elaborado pelo autor

Uma métrica é qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada (SOMMERVILLE, 2007). Métricas também podem ser definidas como padrões quantitativos de medidas de vários aspectos do projeto de software, apoiando as estimativas, o controle de qualidade, a produtividade da equipe e o controle do projeto, para avaliar o desempenho e analisar pontos de

melhoria do que está sendo analisado. Geralmente a métrica é composta por duas ou mais medidas (PRESSMAN, 2006).

Um exemplo de métrica seria o acompanhamento das temperaturas de algum ser durante um determinado período de tempo. Um exemplo pode ser visto no gráfico 5, onde foi coletada nove vezes a temperatura em um intervalo de 240 minutos.

Figura 5 – Métrica da temperatura medida de 30 em 30 minutos



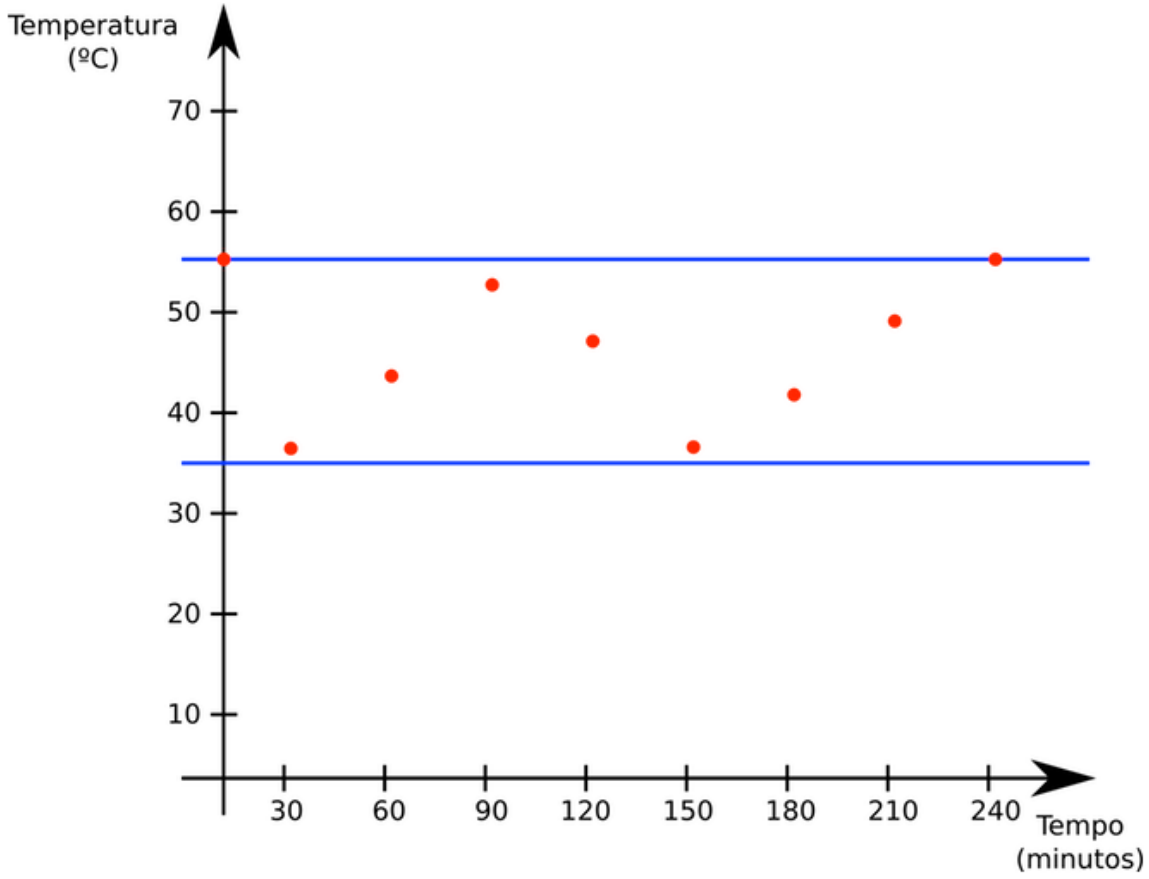
Elaborado pelo autor

Segundo o IEEE um indicador é algo que chama a atenção para uma situação particular, geralmente está relacionado a uma métrica e provê a interpretação daquela métrica em um certo contexto (IEEE, 1990). Métricas que fornecem informações adicionais de estimativas ou avaliações de um determinado atributo (INTERNATIONAL STANDARD ISCO/IEC FDIS 15939, 2002). Busca chamar atenção para uma situação particular. O indicador permite avaliar o status de algo em andamento, indicando potenciais riscos, ajustar o fluxo de trabalho e avaliando a capacidade da equipe de controlar a qualidade do que foi produzido (KAN, 2002). Então o indicador compara a métrica com um resultado esperado, permitindo a tomada de decisões de acordo com a visão da situação.

Um exemplo de indicador pode ser visto no gráfico 6. Nesse caso o indicador

diz que a temperatura pode variar entre 35º a 55ºC. À partir do momento em que é possível comparar as medidas que compõem a métrica de temperatura (55, 36, 44, 53, 47, 36, 40 e 49)ºC com as referências encontradas no estudo (35 a 55ºC), obteve-se um indicador.

Figura 6 – Referências da mínima (35º) e máxima (55º) temperatura comparadas a métrica de temperatura obtida



Elaborado pelo autor

3.4 Abordagem Objetivo-Pergunta-Métrica (Goal Question Metric)

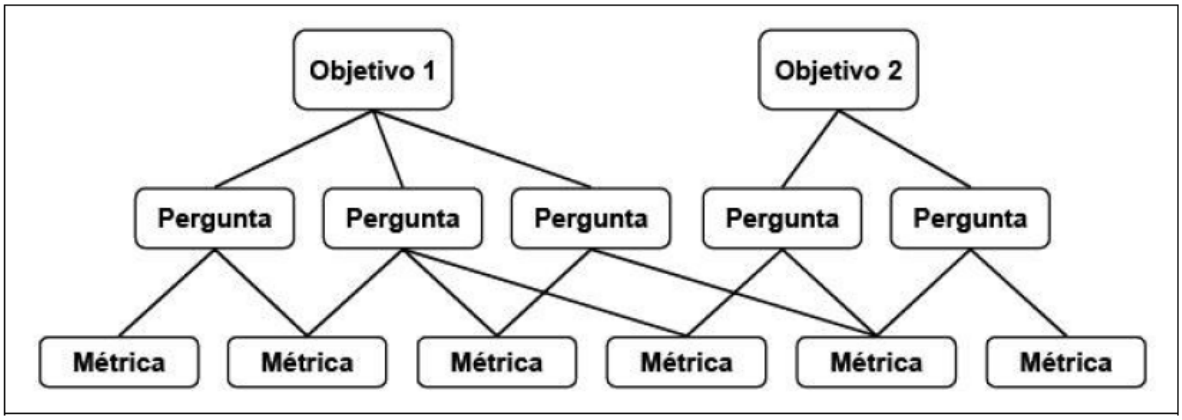
A abordagem GQM é um mecanismo para definir e avaliar um conjunto de objetivos utilizando medições. É um método bastante utilizado pelos grupos de engenharia de software tanto para estudos acadêmicos como para medir e melhorar a qualidade em empresas de desenvolvimento de software. Inicialmente a abordagem foi desenvolvida para avaliar defeitos de um conjunto de projetos no ambiente da NASA/GSFC. Depois disso, o uso da abordagem foi expandido para um contexto maior (BASILI; CALDIERA; ROMBACH, 1994).

A abordagem GQM tem como base a suposição que para uma organização medir de forma significativa e importante, é necessário primeiro definir seus objetivos. Depois rastrear essas metas para os dados destinados a definir esses objetivos opera-

cionalmente e finalmente, fornecer uma estrutura para a interpretação dos dados com respeito aos objetivos declarados.

O GQM possui uma estrutura hierárquica iniciando com um objetivo, especificando o propósito da medição, objeto a ser medido, questão a ser medida junto com um ponto de vista. Esse objetivo é transformado em perguntas que por último são refinadas em métricas. Ela possui uma abordagem top-down e possui três passos como mostra a figura 7 .

Figura 7 – Abordagem GQM



Elaborado pelo autor

- Nível conceitual (Objetivo): São identificados os objetivos para o produto ou processo de acordo com uma variedade de motivos, respeitando os diversos modelos de qualidade e de acordo com vários pontos de vista.
- Nível operacional (Pergunta): Um conjunto de perguntas é criado para moldar como a avaliação de um objetivo específico será realizada. As perguntas buscam caracterizar o objeto de medição em relação ao ponto de vista escolhido.
- Nível quantitativo (Métrica): Definição de métricas que irão fornecer uma resposta quantitativa das questões. As métricas podem ser objetivas, se elas dependerem apenas do objeto que está sendo medido ou subjetivas, se elas dependerem tanto do objeto que está sendo medido quanto do ponto de vista adotado.

O GQM possui o objetivo como ponto central de toda sua estrutura de medição, desta forma ele estabelece um conjunto de informações que a definição de um objetivo deve conter (BASILI; CALDIERA; ROMBACH, 1994):

- Uma proposta: Especifica o objeto alvo da medição e sua razão. Normalmente o objeto é um processo ou produto e exemplos de razões são avaliações, melhoria ou previsão do objeto alvo.
- Uma perspectiva: Levanta uma referência a ser utilizada e a visão de um dos stakeholders que será utilizada, podendo ser de usuários, clientes, desenvolvedores, etc.
- Um ambiente: Descreve o ambiente que está sendo considerado. O ambiente atua como variável para as medições, exemplos de ambiente são características da equipe, processo utilizado, infraestrutura da equipe, etc.

Diversos benefícios são observados pelo GQM (BORGES, 2003):

- Garante que as medições estão alinhadas com as metas da empresa. Consequentemente o risco de coletar dados sem utilidade acaba sendo reduzido.
- O objetivo da coleta de cada medida fica claramente definido. Tornando mais fácil mostrar os envolvidos a importância de coletar os dados de acordo com os objetivos.
- As relações entre metas e medidas estão bastante claras, então, ao existir uma mudança de metas, é mais fácil identificas qual medida está relacionada a meta e adapta-la de acordo com a nova meta.

3.5 Considerações finais

Neste capítulo foram apresentados os principais tópicos para auxiliar o entendimento do leitor acerca do que será abordado neste trabalho. Podem ser ressaltados os principais conceitos sobre testes de software como testes automáticos e testes manuais e suas características e benefícios. Outra questão abordada foi às etapas durante o processo de testes: planejamento, execução e analise.

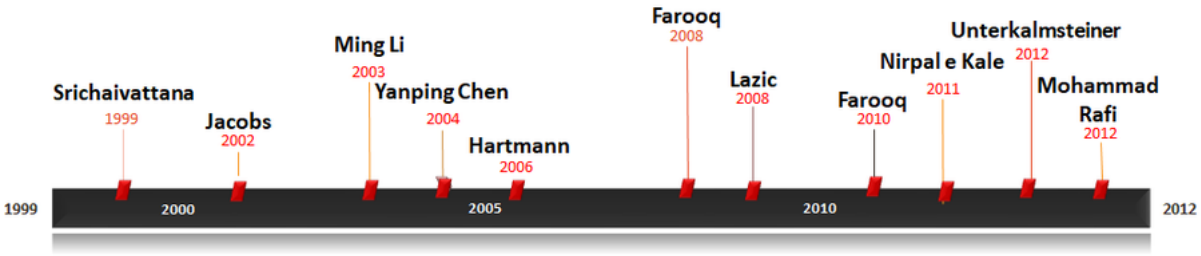
Em seguida, conceitos relacionados a métricas de software foram discutidos. Neste tópico foi apresentado suas principais definições como medição, métrica e indicador. Além do proposito principal de utilizar medições e métricas durante o processo de desenvolvimento de software.

Por último a abordagem GQM foi apresentada. Uma breve descrição da abordagem foi feita onde todas as três etapas foram descritas: objetivo, pergunta e métrica. Além disso alguns beneficios foram listados, mostrando alguns dos motivos de utilizar o GQM.

4 Trabalhos relacionados

Neste capítulo, serão apresentados os principais trabalhos relacionados, os quais foram utilizados para a compreensão do estado da arte desta área de pesquisa, para construção, fundamentação teórica e para identificação da problemática que procura ser solucionada neste projeto de pesquisa. A figura 8 mostra uma linha do tempo de data de publicação destes trabalhos relacionados.

Figura 8 – Linha do tempo dos trabalhos relacionados



Elaborado pelo autor

A equipe de desenvolvimento de comércio eletrônico da IBM (ECD) definiu um conjunto de métricas para medir o processo de teste. Porém o conjunto de métricas que buscava medir as melhorias nos testes de software foram incorporadas apenas em um documento de estratégia de testes, mas não foram incorporados ao processo de testes. Desta forma Chen, Probert e Robeson (2004) resolveu avaliar conjunto de métricas proposto pela ECD. As métricas definidas pela ECD foram divididas em 3 grupos. As 3 categorias e as métricas específicas em cada categoria são:

- Métricas de Qualidade: Qualidade do código, qualidade do produto, melhoria do teste e eficiência do teste;
- Métricas de “Time-to-Market”: Tempo do teste durante o tempo de desenvolvimento;
- Métricas de custo para mercado: O custo do teste é normalizado para o tamanho do produto, o custo do teste como proporção do custo de desenvolvimento e custo por unidade de defeito ponderada.

As métricas analisadas foram:

1) Qualidade do código entregue por desenvolvimento;

2) Qualidade do código testado entregue por teste;

3) Efetividade do teste;

4) Efetividade do teste para testes de *drive-out*;

5) Tempo necessário normalizado ao tamanho do produto;

6) Tempo de teste como proporção de tempo de desenvolvimento;

7) Custo de teste normalizado usando tamanho do produto;

8) Custo como proporção de desenvolvimento Custo;

9) Custo por defeito ponderado.

O artigo avaliou as métricas e conseguiu perceber que elas foram capazes de informar melhorias entre etapas do projeto estudado. Porém foi notado que elas são muito alto nível e não são suficientes. Desta forma ele fez algumas sugestões de melhorias que facilitam o entendimento e seriam válidas em todas as organizações de testes.

O Testing Maturity Model (TMM) Burnstein, Suwanassart e Carlson (1996) foi desenvolvido para ajudar os desenvolvedores de software a avaliar e melhorar os seus processos de teste. O TMM recomenda um conjunto de práticas de testes para cada nível de maturidade, no total ele possui 5 níveis. Suwannasart e Srichaivattana (1999) propôs um conjunto de medições para garantir que as organizações sejam capazes de executar o TMM. Foi utilizado o paradigma Goal/Question/Metric (GQM). Para cada nível de maturidade, exceto o primeiro nível, foi gerado objetivos relacionados e a partir dos objetivos foi definido um conjunto de métricas conforme mostra o Quadro 2 :

Quadro 2 – Métricas e objetivos propostos

Nível de maturidade	**Objetivos**	**Métricas**
2 - Fase de definição	2.1. Desenvolver objetivos de testes e depuração	2.1. Definir custo de teste
		2.1. Custo de depuração
	2.2. Iniciar o processo de planejamento de testes	2.2. Custo para desenvolver os planos de teste
		2.2. Calendário de testes
	2.3. Institucionalizar técnicas e métodos básicos de teste	2.3. Número de testes para cada técnica de teste
3 - Integração	3.1. Estabelecendo a organização do teste de software	3.1. Número de testadores
		3.1. Custo da organização de teste
	3.2. Estabelecendo um programa técnico de treinamento	3.2. Número de cursos relacionados a testes oferecidos por ano
	3.3. Integração de Testes no Ciclo de Vida do Software	3.3. Número de defeitos encontrados por fase
	3.4. Controlando e monitorando o processo de testes	3.4. Número de módulos
		3.4. Testes passados
		3.4. Testes falhados
4 - Gerenciamento e medição	4.1. Estabelecer um programa de revisão para toda a organização	4.1. Número de defeitos removidos pela revisão
	4.2 Estabelecer um programa de medição de testes	4.2. Número de defeitos encontrados por tipo de defeito
		4.2. Número de defeitos encontrados por gravidade
		4.2. Produtividade do testador
	4.3. Avaliação de qualidade de software	4.3. Correção
		4.3. Eficiência
		4.3. Flexibilidade
		4.3. Portabilidade
		4.3. Confiabilidade
		4.3. Usabilidade
5 - Prevenções de erros e controle de qualidade	5.1. Aplicação do processo de Prevenção de defeitos	5.1. Esforço total na busca da raiz de defeitos
	5.2. Controle de qualidade	5.2. Porcentagem de cobertura
	5.3. Otimização do processo de teste	5.3. Esforço total na definição de quando parar de testar
		5.3. Esforço total na avaliação de novas ferramentas e tecnologias de teste

Elaborado pelo autor

No artigo ele define o que deve ser coletado para refletir o processo de teste de uma organização e não como é feita a coleta.

Farooq, Georgieva e Dumke (2008) busca criar uma estrutura conceitual para especificar e avaliar explicitamente os aspectos de qualidade do processo de teste. O conceito utilizado por eles foi derivado da Teoria da avaliação (FAROOQ; DUMKE, 2008 apud SCRIVEN, 1991). A teoria define seis elementos principais de qualquer processo de avaliação, são eles: alvo, critérios de avaliação, padrão de referência, técnicas de avaliação, técnicas de síntese e processo de avaliação. Foram identificados 5 atributos de qualidade que podem ser mensuráveis que são capazes de capturar os objetivos mais comuns de qualquer processo de avaliação, são eles: estabilidade, conformidade, capacidade, eficiência e efetividade. Para cada atributo foi definido alguns sub-atributos que totalizaram 13. Os dados coletados através da medição das entidades do processo de teste devem ser transformados em informações de avaliação úteis sobre aspectos de qualidade do processo de teste, para isso foi criado uma matriz de qualidade. Após isso é necessário avaliar os atributos de qualidade utilizando as métricas de teste escolhidas e as fórmulas de medição dos atributos e dos sub-atributos. As métricas não são definidas no artigo. O final da avaliação deve fornecer descrições significativas de qualidade e identificar possíveis áreas de melhoria.

Jacobs e Trienekens (2002) apresenta uma iniciativa que ocorreu na Holanda com a junção de várias indústrias, agências de consultoria e institutos acadêmicos que decidiram criar, validar e disseminar um modelo de melhoria de teste. Esse modelo seria capaz de juntar as qualidades e eliminar as fraquezas dos modelos mais famosos. Foi utilizado como base o modelo TMM. Desta forma foi feito um estudo detalhado desse modelo de forma comparativa com os outros modelos, definindo um conjunto de qualidades e falhas encontradas no TMM. Foi criado o modelo MT-TMM, o modelo possui quase as mesmas áreas de processo do TMM. Nele foi adicionado a terminologia do CMM, uma descrição mais consistente e compreensível das áreas de processo, um glossário de termos mais compreensíveis, áreas de ambiente de testes e alinhamento organizacional, etc. Além disso foi criado um conjunto de “recursos comuns”, que são atributos predefinidos que agrupam práticas genéricas em categorias. Os recursos comuns são componentes do modelo que não são classificados de forma alguma, mas são usados apenas para estruturar as práticas genéricas. São distinguidos cinco recursos comuns, estruturando um total de onze práticas genéricas, são eles: compromisso para executar, habilidade para executar, atividades executadas, implementação direta e verificando a implementação. Devido a essas características, o modelo MT-TMM é capaz de:

- Descrever um conjunto de etapas e ações para melhorar um processo de teste;

- Fornecer níveis de maturidade de teste bem definidos;
- Fornecer um instrumento para avaliar a maturidade do teste;
- Definir medidas para determinar a eficácia das ações de melhoria;
- Recomenda uma base de métricas para selecionar melhorias de processos, acompanhar e controlar a implementação de ações de melhoria e ajustar e focar as melhorias do processo.

Farooq e Quadri (2010) define a efetividade dos testes como a porcentagem de erros encontrados pelos testadores divido pela quantidade total de erros como mostra a fórmula abaixo:

$$EfetividadeTeste = \frac{ErrosReportadosTestadores}{TotalErros}$$

Porém a maior contribuição do artigo é tentar definir as técnicas de testes de software em escalas de medição. A escala proposta contém 5 diferentes níveis: escala nominal, escala ordinal, escala de intervalo, escala de razão e escala absoluta. Segundo o artigo não existe uma escala geral de técnicas de teste de software. Todo mundo classifica as técnicas de teste em alguma ordem de eficácia de acordo com os resultados empíricos que obtêm em algum software. Devido a isso as técnicas de testes devem ser definidas na escala nominal (menos informativa). Para tornar as técnicas melhores e ao menos na escala ordinal é necessário realizar experiências em maior escala e utilizar parâmetros comuns permitindo comparações e apenas pequenas variações nos objetivos.

Li e Smidts (2003) tiveram como objetivo identificar medidas de engenharia de software que podem ser consideradas como bons indicadores de confiabilidade de software. Foi decidido utilizar a opinião de 10 especialistas na área para fazer o ranqueamento. No total foram 30 medidas de engenharia de software ranqueadas de acordo com sua importância. Essas medidas foram avaliadas de acordo com 4 etapas do desenvolvimento de software: Requisitos, planejamento, implementação e testes. Na etapa de testes as medidas melhores ranqueadas foram: “Taxa de falha”, “Densidade de defeito de código” e “Tempo médio de falha”.

Unterkalmsteiner et al. (2012) fez uma revisão sistemática sobre avaliação e medidores no processo de melhoria de softwares. Foi destacado nessa revisão que a maioria dos artigos possui problemas em descrever o contexto de pesquisa, dificultando a análise de reuso. A validade da avaliação dos artigos também foi questionada devido ao fato que a maioria não discute alguns fatores de confusão que podem afetar a pesquisa. A validade dos medidores também é posta em questão, em muitos casos uma melhor definição, seleção e validação ajudaria a validar. Além dessas possíveis melhorias que podem ser analisadas para futuras pesquisas, algumas tabelas

foram criadas com diversas informações, por exemplo, a estratégia de avaliação “pre-post comparison” é a mais utilizada. Qualidade do processo, precisão da estimativa e produtividade são os indicadores de sucesso mais utilizados e confiabilidade e manutenção são os atributos de qualidade mais citados.

Lazic e Mastorakis (2008) inicia o artigo destacando a importância de testes e sua complexidade, destacando que o processo de teste tem seu próprio ciclo de vida e ele deve estar presente durante todo o processo de desenvolvimento de software. A pesquisa foca mais na área de métricas, destacando a importância de criar métricas com significado e que sejam objetivas para que não existam dúvidas quanto ao seu valor. Ele sugere algumas métricas básicas que devem ser utilizadas na maioria dos projetos e algumas métricas calculadas que possuem relação com as métricas básicas, porém podem trazer alguma informação mais importante. Desta forma ele gerou um conjunto de métricas capazes de avaliar a efetividade e a eficiência de um processo de testes.

Rafi et al. (2012) fez uma revisão sistemática de artigos produzidos no período entre 1999 e 2011 buscando encontrar os benefícios e as limitações de testes automatizados. No total 25 artigos foram escolhidos para leitura completa. Nesses artigos foram listados 9 benefícios e 7 limitações. Após a revisão sistemática foi feito um questionário para checar que esses benefícios e limitações são encontradas na indústria de software. Essas limitações acabam apoiando a utilização de testes manuais que são o foco principal da pesquisa. Desta forma todo esse conjunto acaba sendo importante para justificar uma melhor qualidade dos testes manuais.

Hartmann e Dymond (2006) teve como objetivo mostrar como avaliar de forma ágil atividades utilizando métricas. Para isso foi criado uma lista de verificação que deve ser preenchido durante a criação das métricas para que elas possuam informações suficientes para trazer informações de retorno para os envolvidos. Para os autores o checklist possui:

1) Nome: deve ser bem escolhido para evitar ambiguidade, confusão e excesso de simplificação.
2) Questão: deve responder uma pergunta especifica e clara para um papel particular. Se existem múltiplas perguntas, deve ser criado múltiplas medidas.
3) Base de medição: indicação clara do que está sendo medido, incluindo unidades.
4) Suposições: devem ser identificadas para garantir uma compreensão clara dos dados representados.
5) Nível e Uso: indicam os usos pretendidos em vários níveis da organização. Indique limites de uso, se houver.

6) Tendência esperada: quem criou as métricas deve ter alguma ideia dos resultados esperados. Quando a métrica calculada, documente tendências comuns.

7) Quando usar: o que motivou a criação ou o uso dessa métrica? Como tem sido usado anteriormente?

8) Quando parar de usá-la: quando será que vai perder à sua utilidade ou tornar-se enganosa? Projetar isso desde o começo.

9) Como manuseá-la: pense nas maneiras naturais pelas quais as pessoas distorcem o comportamento ou a informação para produzir resultados mais “favoráveis”.

10) Avisos: recomende métricas de balanceamento, limites sobre o uso e os perigos do uso indevido.

Essa forma de preenchimento para que as métricas se tornem mais úteis pode ser utilizada e talvez alterada para criação das métricas durante a pesquisa.

Nirpal e Kale (2011) no seu artigo indicou a importância do uso de métricas. Além disso, ele definiu métricas como uma técnica baseada em medidas que é aplicada a processos, produtos e serviços para fornecer informações de engenharia e gerenciamento e trabalhar as informações fornecidas para melhorar os processos, produtos e serviços, se necessário. As métricas de testes de software foram sugeridas 21 métricas divididas em 3 tipos como pode ser visto na tabela 9 .

Figura 9 – Métricas de teste de software

Manual	• Produtividade dos casos de teste • Resumo de execução de teste • Aceitação de Defeito • Rejeição de Defeito • Defeito de correção • Produtividade de execução de teste • Teste de eficiência • Índice de gravidade de defeitos
Performance	• Produtividade de scripts de desempenho • Resumo de execução de desempenho • Dados de Execução de Desempenho - Lado do Cliente • Dados de Execução de Desempenho - Lado do Servidor • Eficiência de teste de desempenho • Índice de gravidade de desempenho
Automação	• Produtividade de scripts de automação • Produtividade de Execução de Teste de Automação • Cobertura de Automação • Compressão de custos
Métricas em comum	• Variação do esforço • Desvio de cronograma • Mudança de escopo

Nirpal e Kale (2011)

4.1 Considerações finais

Este capítulo apresentou os trabalhos relacionados a esta pesquisa, buscando descrever abordagens de avaliação de testes de software além de buscar artigos que apresentam métricas como ferramentas para avaliar projetos de software e testes de software. Alguns dos trabalhos não apresentam métricas diretamente relacionadas a testes de software, mas a metodologia utilizada por eles serve como guia.

5 Métricas para avaliar a execução de casos de testes

Este capítulo tem como objetivo apresentar a proposta deste trabalho, formada por um conjunto de métricas que vão ajudar na avaliação da execução das execuções de teste de software.

5.1 Visão geral da proposta

Está proposta será apresentada em forma de um conjunto de métricas, Desta forma, os campos que compõem podem ser vistos no quadro 3 (SOARES, 2017):

Quadro 3 – Modelo de métricas

Título	Nome que representa de forma resumida a métrica. Deve ser bem escolhido para evitar ambiguidade e excesso de simplificação
Código Indicador	Numeração que serve como identificador de cada métrica
Descrição	Descreve o que é esta métrica, mostrando o contexto da métrica
Objetivo de medição	Sinaliza o que está sendo verificado pela métrica
Medidas	As medidas que envolvem as métricas, a partir dessas medidas as métricas são calculadas
Formula	Formula com as medidas para calculo da métrica
Exemplo	Um exemplo para melhor entendimento da métrica
Unidade de Medida	Unidade de medida da métrica (ex: Inteiro e porcentagem)
Procedimento de coleta	Como os dados utilizados para calculo da métrica serão coletados
Periodicidade de coleta	Frequência da coleta dos dados (ex: diário, semanal e mensal)
Responsável pela coleta	Pessoa responsável pela coleta da métrica (ex: Gerente de Projeto e Gerente de testes)
Ferramentas	Ferramenta onde os dados serão coletados
Procedimento de análise	Traz informações de como analisar e avaliar o resultado obtido pela métrica
Procedimento para armazenamento	Forma para armazenar as métricas
Periodicidade do armazenamento	Frequência de armazenamento dos dados (ex: diário, semanal e mensal)
Interessados	Parte da equipe que deve receber as informações geradas

Elaborado do autor

5.2 Definição do GQM para definição das métricas

Para a definição das métricas, será utilizado o paradigma Goal-Question-Metric (GQM) apresentado na seção 3.4. O modelo GQM tem uma estrutura top-down e começa com a definição explicita dos objetivos de medição. Estes objetivos são refinados

em várias questões. Cada questão é então refinada em métricas que devem fornecer informações para responder a essas perguntas.

A primeira etapa do GQM consiste em definir os objetivos das métricas e é chamado de nível conceitual. Um objetivo é definido para um objeto, por uma variedade de razões, com respeito a vários modelos de qualidade, sob vários pontos de vista, ou em relação a um ambiente particular. Objetos de medição podem ser produtos, processos ou recursos (BASILI; CALDIERA; ROMBACH, 1994).

Inicialmente foram definidos 4 objetivos:

1) Mensurar o tempo gasto durante as execuções dos planos de teste.
2) Mensurar a quantidade de falhas encontradas durante as execuções dos planos de teste.
3) Mensurar a qualidade e a criticidade das falhas encontradas durante as execuções dos planos de teste.
4) Mensurar a contribuição da automação de testes nos planos de teste.

Na segunda etapa, também chamada de nível operacional, um conjunto de perguntas são usadas para caracterizar a forma como a avaliação / realização de um objetivo específico será realizada com base em algum modelo de caracterização. As perguntas tentam caracterizar o objeto de medição (produto, processo, recurso) em relação a um problema de qualidade selecionado e determinar sua qualidade a partir do ponto de vista selecionado.

As perguntas criadas para o Objetivo 1 - Mensurar o tempo gasto durante as execuções dos planos de teste, foram:

- Q1.1 Quantos testes são executados durante um determinado período de tempo?
- Q1.2 Como está o andamento da execução de um determinado plano de teste?

A pergunta criada para o Objetivo 2 - Mensurar a quantidade de falhas encontradas durante as execuções dos planos de teste:

- Q2.1 Quantas falhas foram encontradas durante a execução de um plano de teste?

As perguntas criadas para o Objetivo 3 - Mensurar a qualidade e a criticidade das falhas encontradas durante as execuções dos planos de teste, foram:

- Q3.1 Qual a criticidade das falhas encontradas?
- Q3.2 As falhas encontradas foram corrigidas?

As perguntas criadas para o Objetivo 4 - Mensurar a contribuição da automação de testes nos planos de teste, foram:

- Q4.1 Qual percentual do ciclo está automatizado?
- Q4.2 Os testes automatizados estão sendo executados de forma correta?
- Q4.3 Qual o ganho de tempo com a execução automatizada em relação à execução manual de testes?

A terceira etapa do GQM é a definição de métricas, onde um conjunto de dados está associado a todas as perguntas para respondê-lo de forma quantitativa a partir das perguntas criadas na etapa 2:

- Número de falhas
- Velocidade
- Tempo estimado x Tempo gasto
- Correção dos defeitos
- Severidad das falhas
- Total de Testes automatizados
- Eficiência de automação
- Ganho de tempo de automação

O quadro 4 mostra a estrutura do GQM para definir as métricas nessa pesquisa. As métricas serão apresentadas de forma detalhada nas próximas seções.

Quadro 4 – GQM para métricas de execução de testes

Objetivos	Perguntas	Métricas
Mensurar a quantidade de falhas encontradas durante as execuções dos planos de teste.	Quantas falhas foram encontradas durante a execução de um plano de teste?	Número de falhas
Mensurar o tempo gasto durante as execuções dos planos de teste	Quantos testes são executados durante um determinado período de tempo?	Velocidade
	Como está o andamento da execução de um determinado plano de teste?	Tempo estimado x Tempo gasto
Mensurar a qualidade e a criticidade das falhas encontradas durante as execuções dos planos de teste	As falhas encontradas foram corrigidas?	Correção dos defeitos
	Qual a criticidade das falhas encontradas?	Severidade das falhas
Mensurar a contribuição da automação de testes nos planos de teste	Qual percentual do ciclo está automatizado?	Número de Testes automatizados
	Os testes automatizados estão sendo executados de forma correta?	Eficiência de automação
	Qual o ganho de tempo com a execução automatizada em relação à execução manual de testes?	Ganho de tempo de automação

Elaborado pelo autor

5.3 Número de falhas

É o número total de falhas reportados durante a execução de um plano específico de testes. Um defeito reportado pode ou não levar a uma alteração no software ou na documentação.

É um dos primeiros indicadores que deve ser calculado após a execução de um plano de testes. Fornece informação básica sobre a estabilidade do teste, mas é importante que esse indicador seja combinado com outras informações para trazer algum significado para análise. O quadro 5 detalha a métrica número de falhas.

Outras métricas que devem ser acompanhadas em conjunto com o número de falhas são:

- Quantidade de testes executados;
- Quantidade de testes que passaram;
- Quantidade de testes que falharam;
- Quantidade de testes bloqueados.

Quadro 5 – Exemplo detalhado do número de falhas

NÚMERO DE FALHAS	
Código Indicador	I001
Descrição	Quantidade de falhas encontradas durante a execução
Objetivo de medição	Verificar a quantidade de falhas encontradas durante a execução de um determinado ciclo de testes.
Medidas	- Número de falhas encontradas = NumeroFalhas - Total de testes que ja foram executados = TestesTotal - Total de testes que falharam = TestesFalha - Total de testes que foram bloquearam = TestesBlock - Total de testes que passaram = TestesPass - Porcentagem de testes que não passaram na execução do ciclo = p_falhas
Fórmula	p_falhas = (TestesFalha+TestesBlock)/TestesTotal * 100
Exemplo	- NumeroFalhas = 14 - TestesTotal = 80 - TestesFalha = 14 - TestesBlock = 6 - TestesPass = 60 - p_falhas = (14+6)/80*100 = 25%
Unidade de Medida	Porcentagem
COLETA	
Procedimento de coleta	Análise de CRs para falhar ou bloquear casos de testes na ferramenta de gestão de projeto
Periodicidade de coleta	Diaria
Responsável pela coleta	Lider da equipe
Ferramentas	Ferramenta de Gestão de Projeto
ANÁLISE	
Procedimento de análise	Deve ser indetificado a quantidade de defeitos encontrados durante o ciclo e qual a porcentagem do ciclo falhou/bloqueou devido a esses defeitos.
ARMAZENAMENTO	
Procedimentos para armazenamento:	Ferramentas de integração
Periodicidade do armazenamento:	Fim de sprint
Responsável pelo armazenamento:	Lider da equipe
Interessados:	Clientes e gerente de projeto
Forma de Comunicação aos interessados:	Reports de atividades

Elaborado pelo autor

Desta forma, é possível relacionar o total de testes executados com as falhas encontradas. A partir dessa relação pode ser feita uma análise se o ambiente e a aplicação estão de acordo com o esperado, utilizando a quantidade de testes bloqueados e de falhas na execução.

5.4 Velocidade

É a quantidade de trabalho realizado pela equipe em um determinado período de tempo, pode se utilizar qualquer unidade de tempo como horas, dias e semanas. Ela indica a quantidade de testes executados em um determinado tempo, tornando possível comparar a velocidade atual com a esperada, possibilitando tomadas de decisões

de acordo com a indicação do gráfico representado no gráfico 11.

Essa métrica confronta o tempo com esforço, com o objetivo de visualizar o quão adiantada ou atrasada a equipe está em relação a sua estimativa inicial. O gráfico 11 é formado pelos eixos X e Y, onde X representa a quantidade de dias para a realização da sprint e Y representa a quantidade de casos de testes.

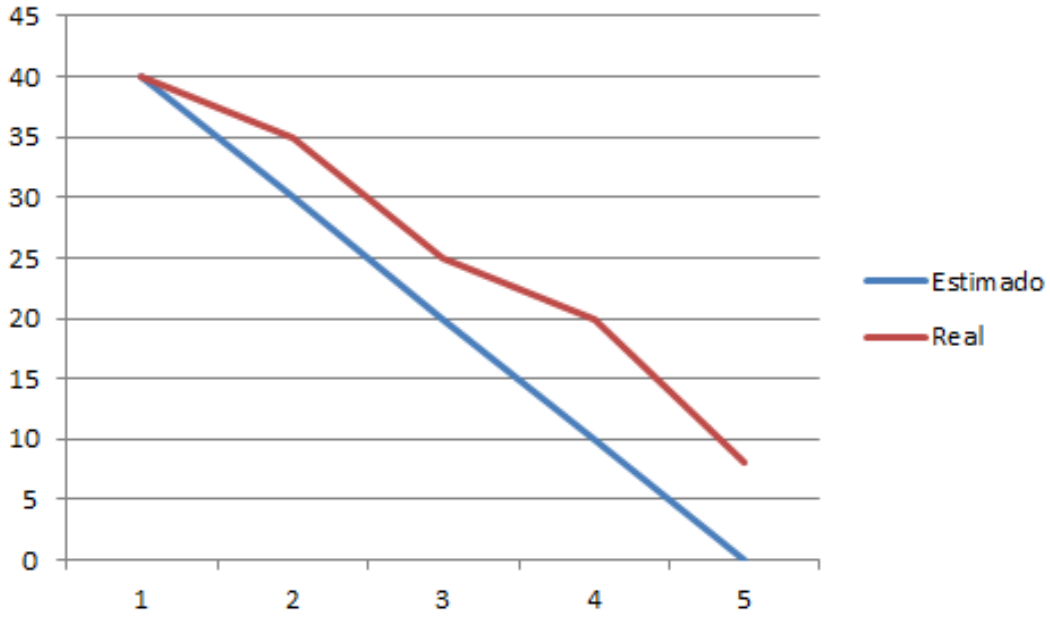
É possível verificar duas linhas traçadas no gráfico, onde a linha azul representa o progresso ideal da Sprint, ou seja, ela deve ser tomada de referência para a velocidade que o time deve atingir para acompanhar o progresso. Já a linha vermelha, representa o progresso real da equipe, ou seja, seu progresso é dado a partir do total de testes executados até o momento. É recomendável que o gráfico seja atualizado diariamente e esteja visível para toda a equipe.

Figura 10 – Métrica velocidade detalhada

VELOCIDADE	
Código Indicador	I002
Descrição	Métrica que indica quantos testes foram executados e um determinado período de tempo
Objetivo de medição	Medir a velocidade de produtividade da equipe
Medidas	Número de casos de testes Tempo gasto
Fórmula	Velocidade = Casos de testes / Tempo
Exemplo	Dia 1: 5 casos de teste por dia Dia 2: 10 casos de teste por dia Dia 3: 5 casos de teste por dia Dia 4: 12 casos de teste por dia Media total: 8 casos de teste por dia
Unidade de Medida	Casos de testes / Tempo
COLETA	
Procedimento de coleta	Obter na ferramenta de gestão de projeto o número de casos de teste rodados em um determinado período de tempo
Periodicidade de coleta	Diária
Responsável pela coleta	Lider da equipe
Ferramentas	Gestão de Projeto
ANÁLISE	
Procedimento de análise	Comparar o tempo gasto nas execuções anteriores com a execução atual
ARMAZENAMENTO	
Procedimentos para armazenamento:	Ferramentas de integração
Periodicidade do armazenamento:	Diário
Responsável pelo armazenamento:	Lider de equipe
Interessados:	Toda equipe
Forma de Comunicação aos interessados:	Sprint Review

Elaborado pelo autor

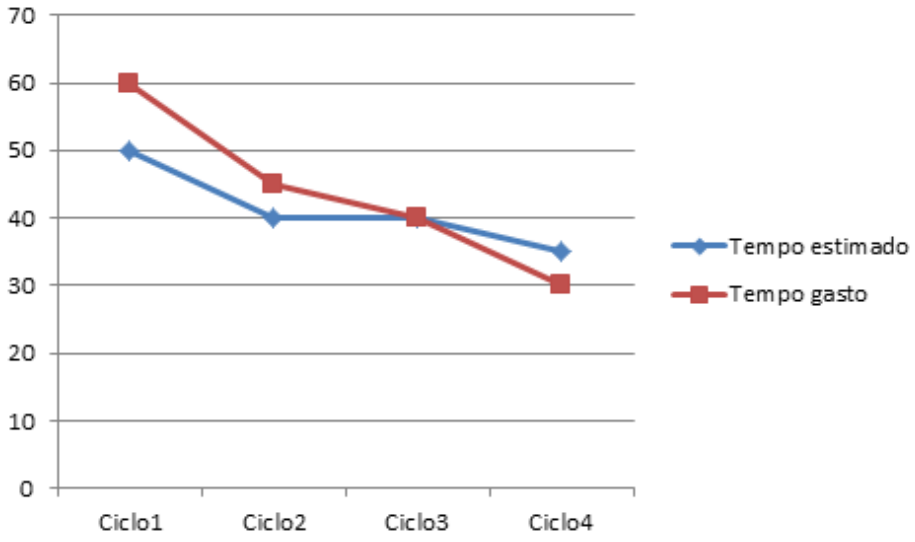
Figura 11 – Gráfico de velocidade



5.5 Tempo estimado x Tempo gasto

É uma métrica que compara o tempo calculado durante o planejamento para executar um ciclo completo de testes com o tempo real que foi gasto durante a fase de testes. É uma métrica que mostra se o tempo gasto durante o ciclo está de acordo com o esperado ou se existiu algum atraso durante a execução. É importante destacar que as antigas execuções servem como base para definir o tempo estimado, fornecendo informações úteis para as próximas execuções. A figura 12 exemplifica o Tempo estimado x Tempo gasto.

Figura 12 – Tempo estimado x Tempo gasto



Elaborado pelo Autor

Para que essa métrica funcione de forma correta, é necessário estimar de forma precisa o tempo necessário para executar um determinado ciclo. Se a estimativa for feita de forma inadequada, provavelmente a diferença vai ser bem alta, tornando a métrica de comparação não útil. O quadro 6 detalha a métrica Tempo estimado x Tempo gasto. Nessa métrica é feita a comparação entre as execuções para indicar se o ciclo já finalizado foi executado de acordo com o tempo esperado ou não.

Quadro 6 – Exemplo Tempo Estimado x Tempo Gasto

TEMPO ESTIMADO X TEMPO GASTO	
Código Indicador	I003
Descrição	Verificar se o time está executando os testes de acordo com o esperado
Objetivo de medição	Avaliar a velocidade de execução comparada com a velocidade esperada
Medidas	- Tempo estimado = TempoEstimado - Tempo total = TempoTotal
Fórmula	(TempoTotal - TempoEstimado)/TempoEstimado * 100
Exemplo	Tempo estimado = 75h Tempo Total = 95h Acompanhamento = (95-75)/75 * 100 Acompanhamento = 26
Unidade de Medida	Horas/Story points
COLETA	
Procedimento de coleta	Coletar o tempo logado na atividade ao final do dia na ferramenta de gestão de projeto
Periodicidade de coleta	Ao fim do ciclo
Responsável pela coleta	Gerente de Projeto
Ferramentas	Gestão de Projeto (Jira)
ANÁLISE	
Procedimento de análise	Os valores X e Y devem ser definidos pelo gerente de projeto de acordo com as necessidades do projeto Acompanhamento >= X ==========> Inadequado 0 < Acompanhamento =< X ========> Pouco acima do esperado -Y < Acompanhamento =< 0 ========> Pouco abaixo do esperado Acompanhamento < Y ==========> É necessario recalcular o tempo esperado
ARMAZENAMENTO	
Procedimentos para armazenamento:	Ferramentas de integração
Periodicidade do armazenamento:	Fim de sprint
Responsável pelo armazenamento:	Lider de equipe
Interessados:	Toda equipa
Forma de Comunicação aos interessados:	Sprint Review

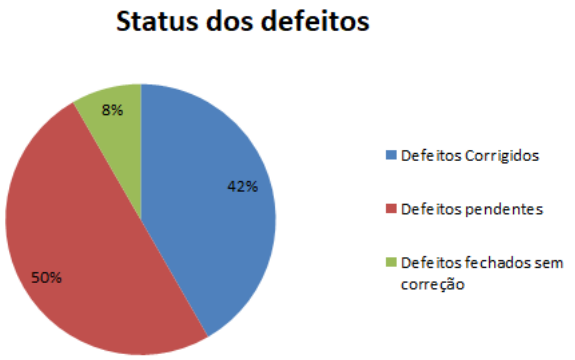
Elaborado pelo autor

5.6 Correção dos defeitos

O Índice de correção é uma métrica que mostra a situação atual dos defeitos encontrados em determinado plano de teste. Existem diversos status para defeitos encontrados, por exemplo:

- Corrigido - O defeito foi corrigido, já existiu a validação e a correção foi aprovada pela equipe.
- Pendente - A equipe de teste abriu o defeito, mas a equipe de desenvolvedores ainda não retornou uma solução para o defeito.
- Fechado, sem correção - O defeito foi aberto pela equipe de teste, mas o desenvolvedor acabou decidindo por motivos que devem ser justificados, que o defeito reportado não deve ser corrigido.

Figura 13 – Exemplo de status dos defeitos abertos num ciclo



Elaborado pelo autor

É importante destacar que a porcentagem das falhas corrigidas e dos defeitos fechados sem correção deve ser feita em relação ao número total de falhas já resolvidos, sem incluir os defeitos ainda abertos, sem resolução definida. Isto faz com que não se utilize defeitos ainda não analisados. A figura 14 detalha a métrica correção dos defeitos.

Figura 14 – Status dos defeitos fechados

Elaborado pelo autor

Nem todo defeito registrado representam uma falha real, algumas pequenas melhorias e até mesmo mudanças fora do escopo podem ser registradas. Com essa métrica é possível acompanhar esse número para descobrir possíveis problemas na abertura de defeitos e também é possível acompanhar a evolução dos defeitos. O quadro 15 detalha a métrica de correção dos defeitos.

Figura 15 – Exemplo Status dos defeitos

CORREÇÃO DOS DEFEITOS	
Código Indicador	I004
Descrição	Faz a comparação dos testes que foram corrigidos com o total de testes
Objetivo de medição	Mostrar se a maioria das falhas foram corrigidas ou está existindo problemas na execução que estão gerando falhas que não serão corrigidas
Medidas	- TotalBug = Número de defeitos encontrados - FixedBug = Número de defeitos corrigidos - BadBug = Número de defeitos fechados sem correção
Fórmula	Fixed_p = (FixedBug/TotalBug)* 100 Bad_p = (BadBug/TotalBug)* 100
Exemplo	- TotalBug = 200 - FixedBug = 84 - BadBug = 16 Fixed_p = (100/100)* 84 = 84% Bad_p = (100/100)* 16 = 16%
Unidade de Medida	Inteiro
COLETA	
Procedimento de coleta	Coletar as CRs utilizadas durante a execução e classifica-las por status na ferramenta de gestão de projeto
Periodicidade de coleta	Mensal
Responsável pela coleta	Gerente de Projeto
Ferramentas	Gestão de Projeto (Jira)
ANÁLISE	
Procedimento de análise	Os valores X e Y devem ser definidos pelo gerente de projeto de acordo com as necessidades do projeto. FixedBug < X% ==========> Inadequado X% <= FixedBug <= Y% ====> Tolerável FixedBug > Y% =========> Desejável
ARMAZENAMENTO	
Procedimentos para armazenamento:	Ferramentas de integração
Periodicidade do armazenamento:	Mensal
Responsável pelo armazenamento:	Gerente de Projeto
Interessados:	Gerente de Projeto
Forma de Comunicação aos interessados:	Sprint Review

Elaborado pelo autor

5.7 Severidade da falha:

Essa métrica é utilizada para definir a gravidade da falhas encontradas durante um ciclo. A severidade da falha é uma métrica importante, pois indica um potencial impacto no negócio para o usuário final. O impacto no negócio é a relação entre o

efeito para o usuário final e a frequência em que ocorre.

É uma métrica que indica a qualidade do produto em teste, quanto maior o número de defeitos com alta severidade, maior a indicação de um produto com baixa qualidade. Com essa informação é possível tomar decisões sobre a liberação de um produto ou versão. A figura 16 exemplifica possíveis níveis de severidade.

Figura 16 – Níveis de severidade dos defeitos

Nível	Descrição
1 – Crítico	Resolver imediatamente. O programa cessa a operação.
2 – Sério	Prioridade alta. Erro severo, porém a aplicação continua.
3 – Médio	Fila normal. Resultado inesperado ou operação inconsistente.
4 – Baixo	Prioridade baixa – design ou sugestão.

Hutcheston

Se um engenheiro de testes achar um defeito e outro funcionário achar dez defeitos num mesmo período de tempo, não necessariamente o segundo engenheiro de testes pode ser considerado mais eficiente que o primeiro. O defeito encontrado pelo primeiro funcionário pode ter uma severidade muito alta, afetando diretamente o desempenho do produto. Já as outras dez falhas encontradas, podem ser apenas superficiais, sem grandes agravantes para o funcionamento do produto.

O índice representa a média das severidades dos defeitos, fornecendo uma medida direta da qualidade do produto. Cada defeito possui um nível de severidade e não existe um padrão para defini-lo. A magnitude da severidade está sempre aberta para debate.

Cada ocorrência é multiplicada por seu nível de severidade, e todos os resultados são adicionados, formando um número total que será dividido pelo número de defeitos. O resultado é considerado o índice de severidade dos defeitos. O quadro 17 detalha a métrica de severidade de defeitos.

Figura 17 – Exemplo severidade das falhas

ÍNDICE DE SEVERIDADE DA FALHA	
Código Indicador	I005
Descrição	Agrupar a quantidade de bugs de acordo com sua severidade (Baixa, Media, Alta e Bloqueante) por ciclos de teste.
Objetivo de medição	Avaliar os bugs encontrados durante um ciclo de teste.
Medidas	- Bugs por ciclo executado = TotalBug - Severidade => Baixa (Ba = 2), Media (M = 3), Alta (A = 4) e Bloqueante (Bl = 5) - Pontos totais por BUG = PontosTotalBug - Porcentagem de bugs por severidade - Indice de severidade das falhas = Indice_severidade
Fórmula	TotalBug = Bugs.Ba + Bugs.M + Bugs.A + Bugs.Bl PontoBug.Ba = (Bugs.Ba x (Ba)) PontoBug.M = (Bugs.M x (M)) PontoBug.A = (Bugs.A x (A)) PontoBug.Bl = (Bugs.Bl x (Bl)) PontosTotalBug = PontoBug.Ba + PontoBug.M + PontoBug.A + PontoBug.B Indice_severidade = PontosTotalBug/TotalBug
Exemplo	BugsBa = 13, BugsM = 17, BugsA = 5, BugsBl = 10 totalBugs = 45 PontoBug.Ba = 13 * 2 PontoBug.M = 17 * 3 PontoBug.A = 5 * 4 PontoBug.Bl = 10 * 5 PontosTotalBug = 26 + 51 + 20 + 50 = 147 Indice_severidade = 147/45 = 3,2
Unidade de Medida	Inteiro
COLETA	
Procedimento de coleta	Coletar as CRs utilizadas durante a execução e classifica-las por status na ferramenta de gestão de projeto
Periodicidade de coleta	Ao fim do ciclo
Responsável pela coleta	Gerente de Projeto
Ferramentas	Gestão de Projeto (Jira)
ANÁLISE	
Procedimento de análise	A análise do Indice deve variar de acordo com o projeto, um exemplo de análise é: Índice < 3 -> Maioria dos erros com baixa severidade 3 <= Índice <= 4 -> Severidade média Índice > 4 -> Severidade alta nas falhas encontradas
ARMAZENAMENTO	
Procedimentos para armazenamento:	Ferramentas de integração
Periodicidade do armazenamento:	A cada Sprint
Responsável pelo armazenamento:	Gerente de Projeto
Interessados:	Toda a equipe
Forma de Comunicação aos interessados:	Reunião de retrospectiva

Elaborado pelo autor

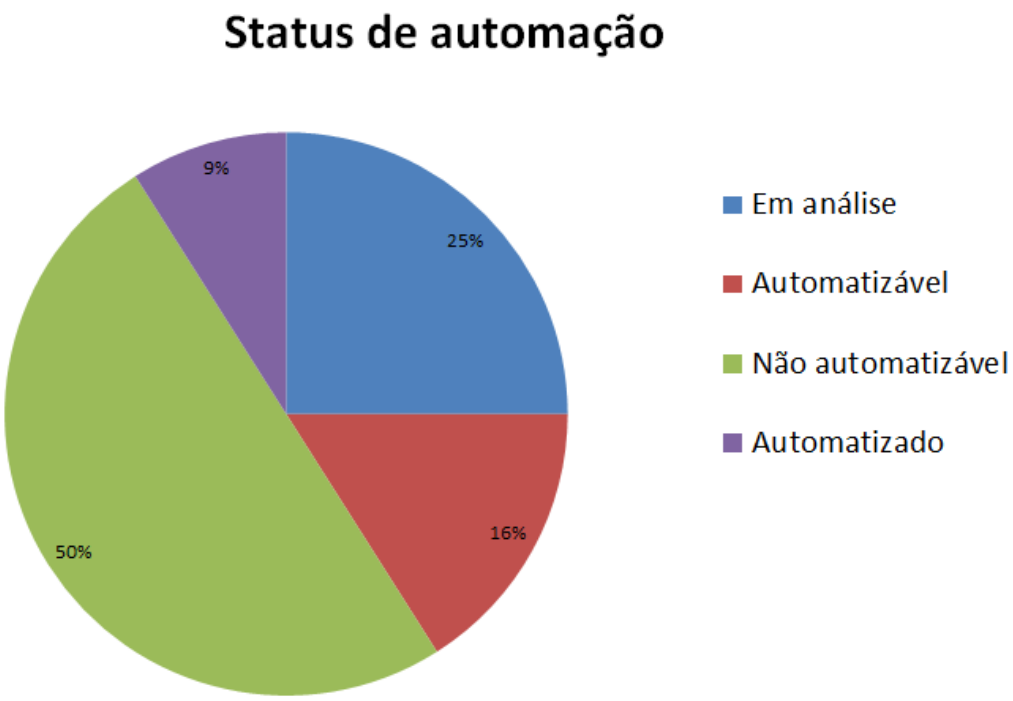
É importante que essa severidade não seja definida pelos próprios testadores, ela deve ser definida por uma pessoa isenta, normalmente a equipe de desenvolvimento é capaz de fazer essa análise outra possibilidade é o lider tecnico. Isso ocorre para que não exista risco que todos os defeitos sejam considerados prioritários.

5.8 Número de testes automatizados

É uma métrica que indica a quantidade de testes já automatizados e a quantidade de testes que ainda podem ser automatizados em um determinado plano. É necessário que uma pessoa capacitada em automação de testes faça a análise completa do ciclo de testes para definir quais testes podem ser automatizados com o ambiente e as tecnologias já disponíveis na empresa, como mostra o exemplo na figura18 . Os status possíveis para os casos de teste são:

- Em análise
- Automatizados
- Automatizáveis
- Não automatizáveis

Figura 18 – Exemplo de status de automação de um ciclo



Elaborado pelo autor

Quadro 7 – Métrica número de testes automatizados

NÚMERO DE TESTES AUTOMATIZADOS	
Código Indicador	I006
Descrição	Verificar se o ciclo possui o nível esperado de testes ja automatizados
Objetivo de medição	Avaliar se a quantidade de testes automatizados está de acordo com o planejado pela equipe.
Medidas	- total_testes = Quantidade de testes - total_automatizados = Quantidade de testes automatizados - total_automatizaveis = Quantidade de testes automatizaveis - p_automatizados = Porcentagem de testes automatizados no ciclo - p_automatizaveis = Porcentagem de testes automatizaveis no ciclo
Fórmula	p_automatizados = (total_automatizados/total_testes) *100 p_automatizados2 = (total_automatizados/total_automatizaveis) *100
Exemplo	TCs: 200 Numero de testes automatizaveis: 32 N° automatizados: 100 Aplicação da Formula: p_automatizados = (100/200) * 100 p_automatizados2 = (132/200) * 100 p_automatizados = 50% p_automatizaveis = 66%
Unidade de Medida	Porcentagem
COLETA	
Procedimento de coleta	Contabilidade dos testes com as labels de automação na ferramenta de gestão de projeto
Periodicidade de coleta	Mensal
Responsável pela coleta	Gerente de Projeto
Ferramentas	Gestão de Projeto (Jira)
ANÁLISE	
Procedimento de análise	Os valores para análise devem ser definidos pelo gerente de projeto, um exemplo é: 0 a 60 => Abrangência adequada 61 a 80 => Abrangência parcialmente adequada acima de 81 => Abrangência adequada
ARMAZENAMENTO	
Procedimentos para armazenamento:	Atualizar crescimento de testes automatizados em uma planilha de automação
Periodicidade do armazenamento:	Mensal
Responsável pelo armazenamento:	Gerente de Projeto
Interessados:	Toda a equipe
Forma de Comunicação aos interessados:	Reunião de retrospectiva

Elaborado pelo autor

5.9 Eficiência de automação

Indica o aproveitamento dos testes automáticos no ciclo, desta forma é possível identificar quais melhorias ainda podem ser feitas em testes já automatizados para garantir a estabilidade da execução. A métrica compara a quantidade de testes automatizados que foram utilizados no ciclo com a quantidade total de testes. Outra comparação valida para ser feita é a entre a quantidade de testes automatizados que foram utilizados no ciclo com a quantidade de testes automaticos que existem no ciclo, desta forma é possivel obter uma relação mais detalhada entre os testes automatizados.

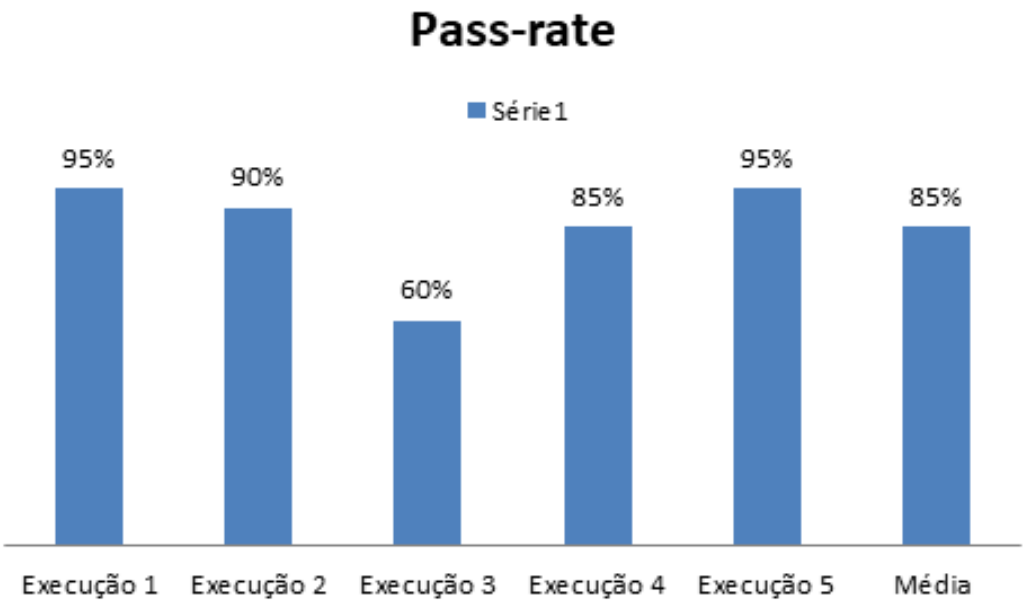
O quadro 8 detalha a métrica de eficiência de automação. A figura 20 exemplifica a eficiência de automação em cinco execuções e indica a média da eficiência.

Quadro 8 – Exemplo aproveitamento automação

EFICIÊNCIA DE AUTOMAÇÃO	
Código Indicador	I007
Descrição	Métrica que mostra se os testes automatizados estão trazendo um retorno para a execução
Objetivo de medição	Verificar se os testes automatizados estão sendo aproveitados como esperado
Medidas	- Quantidade de testes = total_testes - Quantidade de testes automatizados = total_auto - Testes executados automaticamente = total_exec
Fórmula	Eficiencia = (total_exec / total_testes) * 100
Exemplo	total_testes = 237 total_exec = 100 eficiencia = (100/237)*100 eficiencia = 42%
Unidade de Medida	Porcentagem
COLETA	
Procedimento de coleta	Criação de um campo para indicar se o teste foi executado de forma automatica ou manual na ferramenta de gestão de projeto
Periodicidade de coleta	Por ciclo
Responsável pela coleta	Gerente de Projeto
Ferramentas	Gestão de Projeto (Jira)
ANÁLISE	
Procedimento de análise	Os valores X e Y devem ser definidos pelo gerente de projeto de acordo com as necessidades do projeto. 0 a X => Abrangencia inadequado X a Y => Abrangencia parcialmene adequados acima de Y => Abrangencia adequado
ARMAZENAMENTO	
Procedimentos para armazenamento:	Atualizar aproveitamento de testes automatizados em uma planilha de automação
Periodicidade do armazenamento:	A cada Sprint
Responsável pelo armazenamento:	Gerente de Projeto
Interessados:	Toda a equipe de automação
Forma de Comunicação aos interessados:	Reunião de retrospectiva

Elaborado pelo autor

Figura 19 – Exemplo de Pass-rate em 5 execuções



5.10 Tempo automação x tempo manual

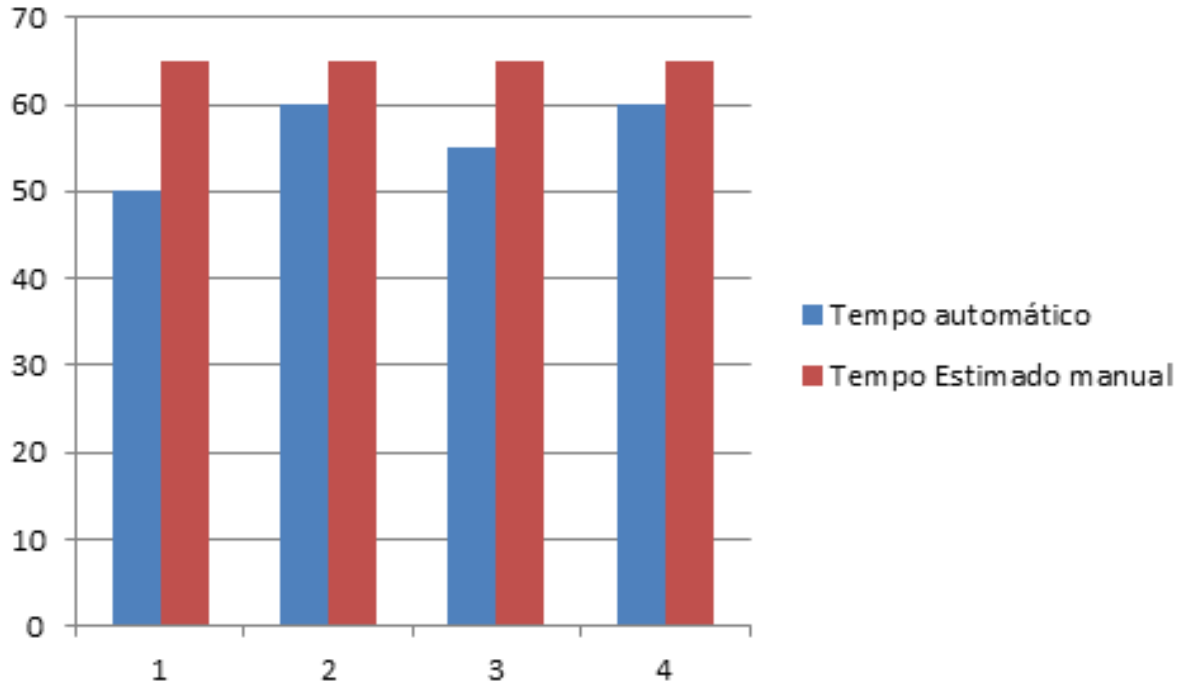
É uma métrica comparativa onde é necessário medir o tempo gasto para a execução automática ser realizada e comparar esse valor com o valor estimado que seria gasto com a execução manual dos mesmos testes. É uma métrica importante para mostrar que os testes automáticos estão trazendo ganho de tempo em relação à execução manual. Desde que a execução automática estiver gastando mais tempo que a execução manual, deve ser analisado os motivos disso e o custo para melhorar a execução automática. A figura 20 exemplifica um gráfico comparativo entre o tempo gasto numa execução automática e o tempo estimado para a execução manual. O quadro 9 detalha a métrica Tempo automação x tempo manual.

Quadro 9 – Métrica Tempo automação x Tempo estimado

GANHO DE TEMPO AUTOMAÇÃO	
Código Indicador	I008
Descrição	O indicador mostra se os testes automatizados estão conseguindo executar de forma correta, sendo util para todo o projeto
Objetivo de medição	Verificar se os testes rodados de forma automática estão sendo executados de forma mais rápida que se eles fossem executados de forma manual.
Medidas	- Quantidade de testes - Quantidade de testes automatizados - Testes executados automaticamente - Tempo_a = Tempo total gasto automaticamente - TempoEstimado = Tempo estimado para executar de forma manual
Fórmula	GanhoTempo = ((Tempo_a - TempoEstimado)/TempoEstimado) * 100
Exemplo	TCs: 237 Numero de testes automatizaveis: 100 N° automatizados: 63 Tempo total gasto automaticamente = 26 Tempo total gasto manualmente = 30 GanhoTempo = (26 -30)/30 * 100 GanhoTempo = 13%
Unidade de Medida	Porcentagem
COLETA	
Procedimento de coleta	Utilizar as story points/horas logadas pela execução automatica para servir de comparação com o tempo estimado
Periodicidade de coleta	Por ciclo
Responsável pela coleta	Gerente de Projeto
Ferramentas	Gestão de Projeto (Jira)
ANÁLISE	
Procedimento de análise	I6< 90 = Esperado 90 < I6 < 100 = Ganho pequeno I6 > 100 = Tempo maior q execução manual
ARMAZENAMENTO	
Procedimentos para armazenamento:	Comparação dos tempos é apresentada em uma planilha de automação
Periodicidade do armazenamento:	A cada Sprint
Responsável pelo armazenamento:	Gerente de Projeto
Interessados:	Toda a equipe de automação
Forma de Comunicação aos interessados:	Reunião de retrospectiva

Elaborado pelo autor

Figura 20 – Comparação execução automática com tempo estimado



Elaborado pelo autor

5.11 Considerações finais

Este capítulo apresentou algumas métricas, que tem como objetivo auxiliar a avaliação da execução de testes de software de um projeto dentro de uma organização de software.

São métricas que buscam avaliar as execuções de testes em termos de tempo, velocidade de execução, quantidade e criticidade de falhas encontradas além de avaliar a contribuição dos testes automatizados na execução. Como são recomendações, as organizações podem escolher quais delas se adéquam a necessidade do projeto e quais podem ser usadas para que uma estimativa mais precisa seja alcançada.

O custo para utilizar essas métricas está diretamente relacionado com a identificação, coleta e análise em informações úteis das métricas. Porém estes custos podem ser compensados com a tentativa de encontrar informações mais próximas do real. Assim com valores mais precisos será possível tomar as melhores decisões a respeito do que deve ser feito para obter uma execução mais eficiente na equipe de testes.

6 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.

6.1 Planejamento

O grupo focal busca ajudar a responder a questão de pesquisa central deste trabalho, que é como avaliar a efetividade da execução de testes. Desta forma, o objetivo do grupo focal é avaliar o processo do GQM e as métricas geradas pelo processo, sob os aspectos de completude, clareza e adequação.

Ao planejar o grupo focal ficou definido que o mesmo estaria dividido em três fases, como pode ser visto na Figura 21. A primeira fase contemplou a apresentação da proposta de forma resumida, definindo os objetivos do grupo focal. Uma entrevista semiestruturada onde as etapas do GQM e as métricas propostas durante a pesquisa foram validadas. Por último foi feita uma análise da entrevista, onde o pesquisador filtrou as informações coletadas para identificar as sugestões que melhor se enquadram na pesquisa.

Figura 21 – Etapas do Grupo Focal

Elaborado pelo autor

O roteiro foi planejado com as atividades descritas no Quadro 10.

Quadro 10 – Roteiro para realização do grupo focal

Atividade	Duração	Passos
1	10 min	Apresentação dos objetivos para realização do grupo focal; Apresentação da visão geral da proposta
2	10 min	O moderador apresentou a descrição do primeiro **nível do GQM - Objetivos**, e em seguida, iniciou a seguinte discussão sobre os objetivos: • As definições dos objetivos estão claras? • A quantidade de objetivos é suficiente? • É necessário adicionar ou remover algum objetivo?
3	10 min	O moderador apresentou a descrição do segundo **nível do GQM - Perguntas**, e em seguida, iniciou a seguinte discussão sobre os objetivos: • A formulação das perguntas está clara? • A quantidade de perguntas é suficiente? • É necessário adicionar ou remover algum pergunta?
4	10 min	O moderador apresentou a descrição da métrica: **Número de falhas**, e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
5	10 min	O moderador apresentou a descrição da métrica: **Velocidade**, e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
6	10 min	O moderador apresentou a descrição da métrica: **Tempo estimado x Tempo gasto**, e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
7	10 min	O moderador apresentou a descrição da métrica: **Índice de correção dos defeitos** e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
8	10 min	O moderador apresentou a descrição da métrica: **Índice de severidade**, e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
9	10 min	O moderador apresentou a descrição da métrica: **Número de testes automatizados**, e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
10	10 min	O moderador apresentou a descrição da métrica: **Aproveitamento de automação no ciclo**, e em seguida, iniciou a seguinte discussão: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
11	10 min	O moderador apresentou a descrição da primeira métrica: **Tempo automação x tempo manual**, e em seguida, iniciou a seguinte discussão sobre a métrica: • A descrição e o objetivo da métrica estão claros? • As formas de coleta, armazenamento e comunicação estão claras? • O procedimento de analise está de acordo com o esperado?
12	10 min	O moderador apresentou a descrição do terceiro **nível do GQM - Métricas**, e em seguida, iniciou a seguinte discussão sobre as métricas: • A quantidade de métricas é suficiente? • É necessário adicionar ou remover alguma métrica?
Tempo Total	120 min	

Elaborado pelo autor

Neste grupo focal foram selecionadas 5 pessoas que fazem ou fizeram parte de equipe de testes, exercendo assim diversas funções na área de testes, desde execução de testes, planejamento dos testes, automação de testes, análise, criação de casos de testes e análise dos resultados. O Quadro 11 descreve o perfil dos participantes, cujos nomes foram preservados por questões de confidencialidade.

Quadro 11 – Especialistas selecionados para o grupo focal

Especialista 1 (E1)	
Nível de experiência	3 anos de experiência em prática em teste de software
Dados demográficos	Idade: 25 anos Gênero: Masculino Formação: Graduado em Ciência da Computação Função: Engenheiro de Teste de Software
Interesses individuais	Automação em testes de software
Habilidades técnicas	Java, Python, Jira, GitHub
Especialista 2 (E2)	
Nível de experiência	3 anos de experiência em prática em teste de software
Dados demográficos	Idade: 23 anos Gênero: Masculino Formação: Graduado em Engenharia da Computação Função: Técnico em Teste de Software
Interesses individuais	Área de automação em testes de software
Habilidades técnicas	Java, C, Scrum, Testes
Especialista 3 (E3)	
Nível de experiência	3 anos de experiência em prática em teste de software
Dados demográficos	Idade: 27 anos Gênero: Masculino Formação: Graduado em Engenharia da Computação Função: Engenheiro de Teste de Software
Interesses individuais	Análise e requisitos de criação de use case
Habilidades técnicas	Java, Scrum
Especialista 4 (E4)	
Nível de experiência	Profissional com dois anos de experiência no mercado de teste de software
Dados demográficos	Idade: 22 anos Formação: Graduando em Sistemas de Informação Gênero: Masculino Função: Trainee de testes de software
Interesses individuais	Automação de testes e testes exploratórios
Habilidades técnicas	Java, Python, Android
Especialista 5 (E5)	
Nível de experiência	2 anos de experiência em prática em teste de software
Dados demográficos	Idade: 26 anos Gênero: Feminino Formação: Graduado em Ciência da Computação Função: Engenheiro de Teste de Software
Interesses individuais	Automação de testes, análise de qualidade
Habilidades técnicas	Java, Android, Python

Elaborado pelo autor

6.2 Execução

O primeiro contato para realização do grupo focal se deu no início do mês de dezembro através de uns convites via e-mail e hangouts enviado para um total de 10 pessoas. Este e-mail continha um resumo da proposta a ser avaliada, um resumo geral de como seria o grupo focal, o número mínimo de pessoas necessárias e seus perfis. O retorno ao convite foi positivo, o mesmo foi aceito e 5 pessoas se disponibilizaram para participar do grupo focal.

O grupo focal ocorreu no dia 26 de dezembro de 2018, com duração aproximada de 2 horas, sendo registrado em áudio, mediante a permissão dos participantes, além das anotações realizadas pelo moderador, sempre que julgado necessário.

6.3 Análise dos dados

Para a análise dos dados, foi realizado uma análise e comparação das discussões em torno dos tópicos apresentados no grupo focal. Para isso, os dados qualitativos foram analisados a partir da transcrição dos áudios.

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 Quadro 11.

6.3.1 Nível um do GQM - Objetivos

Em relação ao primeiro nível do GQM, que é a definição dos objetivos, após apresentação de seus conceitos, todos os especialistas indicaram que sua descrição estava clara. Três participantes indicaram que a quantidade de objetivos estava adequada, sem necessidade de adição ou remoção de objetivos. Um dos especialistas sugeriu um objetivo que buscaria medir a quantidade de testes executados durante um determinado período de tempo. O mesmo especialista também sugeriu que ao invés de um objetivo para medir a contribuição da automação dos testes, fossem utilizados três objetivos que mediriam o tempo gasto, quantidade e criticidade das falhas encontradas em ciclos rodados de forma automática. Algumas respostas que comunicaram essa análise estão presentes a seguir:

> [. . .] Os quatro objetivos estão bem claros e de acordo com o que é necessario na avaliação de execução de testes. E1
>
> [. . .] Mensurar a quantidade de testes rodados também seria importante. E2
>
> [. . .] O último objetivo poderia ser dividido em três para avaliar a automação da mesma maneira que a execução manual, mensurar o tempo gasto, a quantidade e a criticidade das falhas rodadas de forma automatica. E2

> [. . .] A mesma análise deveria ser feita para testes automaticos, tempo, criticidade e quantidade. E3

6.3.2 Nível dois do GQM - Perguntas

No segundo nível do GQM, onde as perguntas são formuladas de acordo com os objetivos definidos na etapa anterior, as questões produzidas durante a pesquisa foram consideradas claras e importantes. Nenhuma remoção foi aconselhada, mas algumas questões foram propostas. Para o primeiro objetivo, *mensurar o tempo gasto durante as execuções dos planos de teste,* foram sugeridos:

- Quantas pessoas pessoas trabalharam na execução?
- Quantas pessoas são necessárias para executar o plano de teste no prazo estimado?

Um dos comentários feito sobre as perguntas do primeiro objetivo, foi:

> [. . .] Poderia ser perguntado a quantidade de testes executados e o tempo gasto de forma separada para ficar um valor unitario por resposta. E3

Para o segundo objetivo, *mensurar a quantidade de falhas encontradas durante as execuções dos planos de teste,* foram sugeridos:

- Em qual nível de maturidade o software se encontra?
- Quantas vezes o mesmo ciclo já foi executado?

Além disso, foi ressaltado a importância da pergunta definida pela pesquisa:

> [. . .] Quantas falhas foram encontradas é a pergunta principal relacionada com esse objetivo. E3

Já em relação ao terceiro objetivo, *mensurar a qualidade e a criticidade das falhas encontradas durante as execuções dos planos de teste,* foram sugeridos as seguintes perguntas:

- Quantos testes foram bloqueados e falhados pela mesma CR?
- Qual o esforço gasto pelo desenvolvedor para corrigir a CR?

Em relação a essa ultima sugestão, foi dito que:

> [. . .] Em quanto tempo a falha foi corrigida? mesmo nao tendo uma prioridade tão alta, ela foi importante ou simples o suficiente para ser corrigida de forma rapida. E4

As perguntas relacionadas ao último objetivo, *mensurar a contribuição da automação de testes nos planos de teste,* foram consideradas validas pelos especialistas. As sugestões foram:

- Qual o tempo gasto para manutenção dos testes?
- Qual a quantidade de falhas encontradas?
- Qual a criticidade das falhas encontradas?

Ainda em relação às perguntas relacionadas ao ultimo objetivo, alguns trechos devem ser destacados:

> [. . .] Basicamente mensurar a contribuiçao de testes automaticos. Qual o tempo gasto? Qual a quantidade de falhas encontradas? Qual a criticidade das falhas encontradas? E2
>
> [. . .] Deveria tambem levar em consideração o esforço que foi gasto para automatizar os testes.E4
>
> [. . .] Normalmente quando falha, a gente para pra investigar o motivo da falha, existe um gasto de tempo nesse momento para analise. E4

Em relação as 3 primeiras métricas propostas na pesquisa, número de falhas, velocidade e tempo estimado x tempo gasto, o grupo focal concordou com todas as definições como descrição, coleta, análise e armazenamento. Além disso, em relação à métrica *tempo estimado x tempo gasto,* foi ressaltado a importância dessa métrica para todos os participantes do projeto:

> [. . .] É importante que toda equipe saiba dessa métrica, desde os lideres que estarão monitorando até os testadores que vão saber como está seu rendimento E1

Durante o debate sobre a quarta métrica, correção dos defeitos, houve uma sugestão considerada valida por todos os participantes. Foi sugerido que ao invés de comparar as falhas corrigidas com as falhas consideradas ruins, fosse comparado as falhas consideradas boas com as falhas consideradas ruins, existem alguns casos onde algumas falhas não foram corrigidas mas eram erros que deveriam ser reportados pelos testadores. Alto risco e a alto custo são alguns dos motivos que podem fazer com que falhas consideradas boas não sejam corrigidas. Além disso, um dos especialistas comentou que a coleta e o armazenamento dessa métrica poderia ser feita apenas ao fim do ciclo de desenvolvimento do software, diminuindo o custo dessa métrica.

> [. . .] Poderia classificar como bom e ruim, bom e ruim é um erro que é erro mas que por algum motivo não foi fixado, por exemplo, alta complexidade de correção e risco alto. E3

Houve um problema inicial para o entendimento da métrica *severidade das falhas*, pois o cálculo está relacionado com os pesos de cada nível e a quantidade de falhas para cada nível de falha. Após explicação do moderador, os especialistas concordaram com a descrição. Um dos especialistas sugeriu que o nível de severidade de cada falha fosse indicado de acordo com a quantidade de testes falhados e bloqueados no ciclo, mas os outros especialistas não concordaram, pois algumas falhas podem ser consideradas de alta severidade.

Em relação à métrica número de testes automatizados, os especialistas concordaram com o objetivo, analise e descrição, um dos especialistas destacou a importância de medir a quantidade de testes automatizados em relação ao total de testes e ao total de testes automatizáveis. Já em relação à coleta e armazenamento, foi comentado que a equipe de automação acaba sendo responsável em muitos casos pela coleta e armazenamento dessa métrica.

Em relação à métrica ganho tempo automação, houve alguns questionamentos por parte dos especialistas. Um deles questionou se o objetivo definido é realmente valido, pois ,para ele, a comparação em termos de tempo entre uma execução automática onde não existe uma pessoa executando o teste com uma execução manual não é valida. Outro especialista afirmando que a execução automática, na empresa onde ele atua, não leva em conta o tempo de duração da execução automática. A seguir, alguns comentários relativos à área de processo de requisitos são apresentados:

> [. . .] “Por ser automatizado o tempo vira 0, pois a execução automatica não exige uma pessoa nessa atividade” E4

> [. . .] É uma métrica para casos bem especificos onde a execução automatica tem que ser entregue de forma rapida e a execução manual talvez consiga entregar de forma mais rapida que automática. E1

6.4 Limitações e ameaças à validade

Dentre as limitações encontradas no grupo focal, destaca-se que não foi possível que todos os participantes tivessem experiencia em análise e coleta de métricas. Para tentar diminuir essa limitação, buscou especialistas com experiência em empresas que estão sempre utilizando e apresentando suas métricas para monitoramento na fase de testes de software.

Outra limitação é a que alguns participantes podem se intimidar e diminuir suas contribuições devido à menor experiencia na área e até mesmo por serem

mais retraídos socialmente. Como forma de reduzir essa ameaça, foi selecionado um grupo com tempo de experiência similar na área de testes de software.

Os integrantes do grupo focal possuem 2 ou 3 anos de experiência, isso acaba se tornando mais uma ameaça a validade ao grupo focal, pois os integrantes ainda estão no início da carreira profissional, algumas atividades ainda não fazem parte do histórico profissional dos participantes.

Uma das ameaças à validade da aplicação do grupo focal acontece com problemas de compreensão, devido ao limite de tempo para realização do grupo focal. Desta forma, problemas de interpretação podem ocorrer com alguns dos participantes. Para minimizar esta ameaça, o moderador iniciou o grupo focal apresentando o tópico a ser discutido buscando uma compreensão do assunto por todos os envolvidos.

Outra restrição observada foi que o autor da proposta foi o mediador do grupo focal, desta forma, algum dos participantes poderiam restringir seus comentários durante a participação da reunião. Buscando minimizar esta ameaça, buscaram-se participantes que não tivessem laços de amizade que pudessem interferir nas respostas e comentários durante o grupo focal.

6.5 Considerações finais

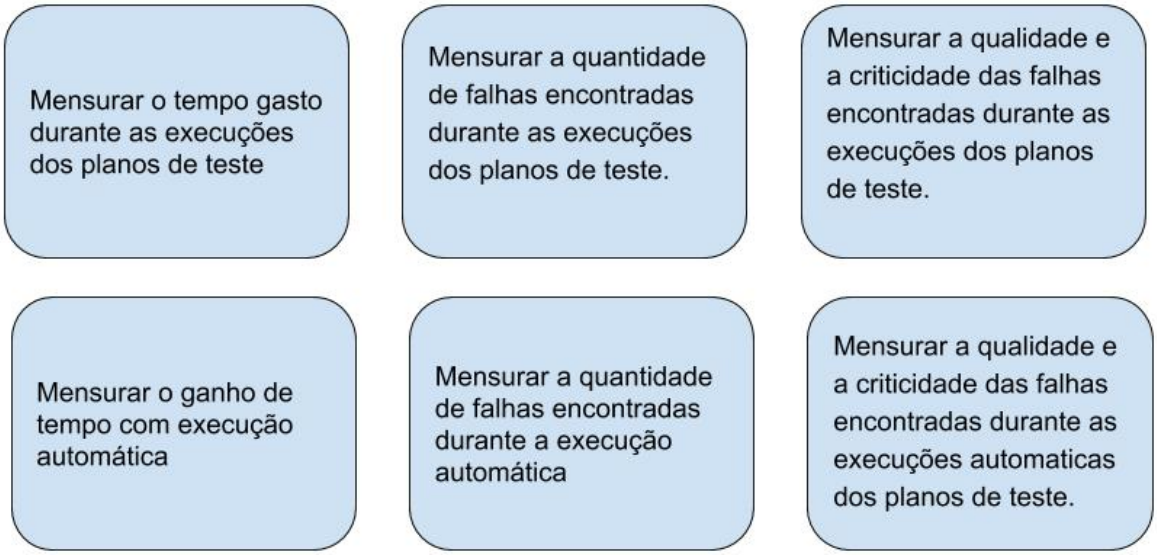
O grupo focal foi o método utilizado para validar a proposta da pesquisa. A visão de especialistas na área ajudou com sugestões de melhorias desde o processo de escolha das métricas até a definição de cada uma delas. O método foi realizado com um grupo de participantes homogêneos, em termos de idade e nível de experiência profissional, contando com a presença de cinco participantes, apesar de o convite ter sido feito para outros quatro.

Os especialistas analisaram inicialmente o GQM que foi utilizado para escolher às métricas, foram analisados às três etapas desse processo, definição dos objetivos, das perguntas e das métricas. Na etapa das métricas, ainda houve um brainstorm para chegar em possíveis métricas que se encaixariam na proposta. Após essa etapa, foi apresentado o conjunto de oito métricas produzidas na pesquisa com o proposito de avaliar em quatro aspectos: definição, coleta, analise e armazenamento. Ao final do grupo focal foi possível consolidar algumas sugestões de melhoria para as métricas, tais como:

- O objetivo 4 poderia ser dividido em mais objetivos para melhor entendimento das contribuições da automação de testes. Da mesma forma que existiram objetivos para mensurar tempo, quantidade de falhas e qualidade de falhas para execução manual, poderia existir os mesmos objetivos para a execução

automática. A imagem 22 mostra as sugestões dos objetivos dos participantes do grupo focal;

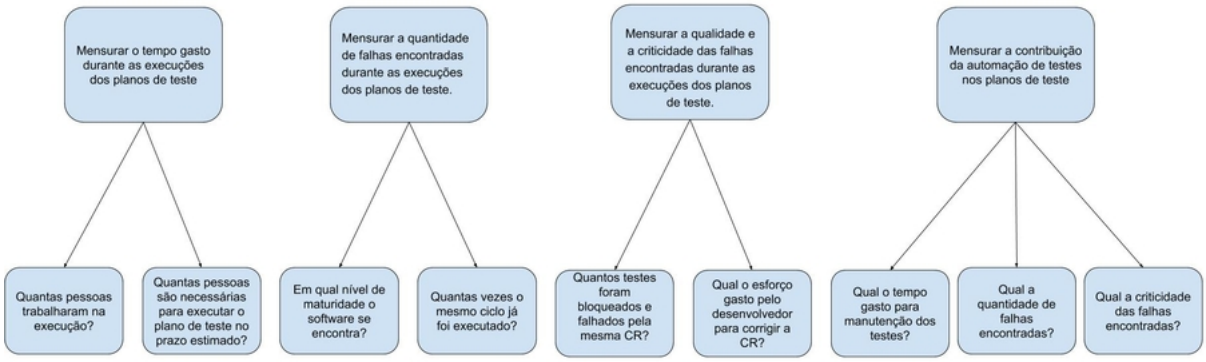
Figura 22 – Sugestões de objetivos



Elaborado pelo autor

- Adição de novas perguntas no GQM, a imagem 23 indica como as perguntas que foram sugeridas pelos participantes;

Figura 23 – Sugestões de perguntas

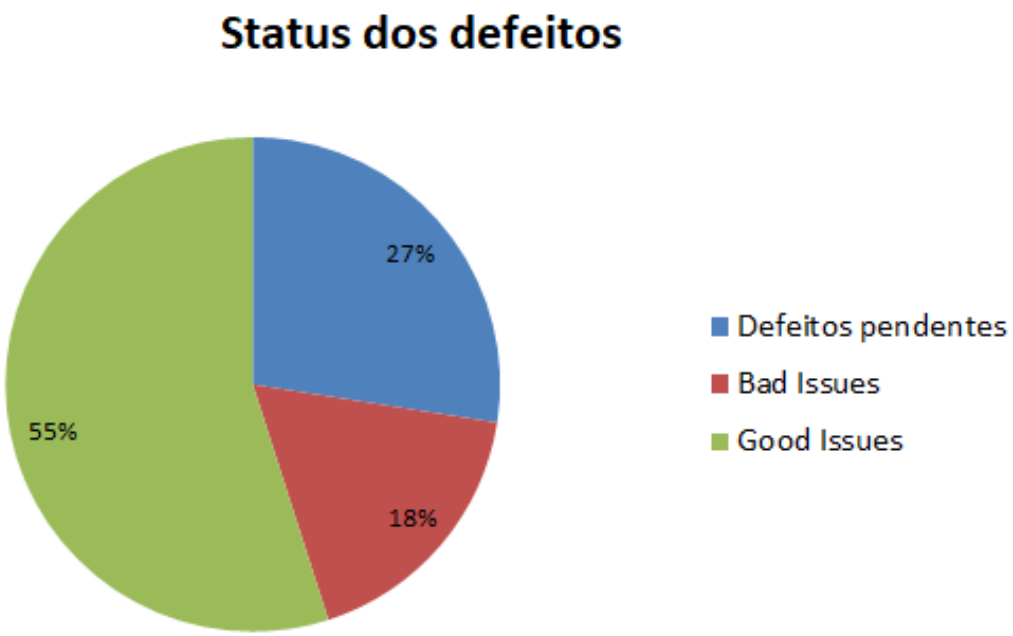


Elaborado pelo autor

- A métrica correção de defeitos poderia checar a quantidade de falhas definidas como boas e falhas definidas como ruins ao invés de verificar as falhas corrigidas, desta forma, também cobriria falhas que não foram corrigidas por motivos como risco e custo alto de correção. A imagem 24 mostra como seria a

métrica correção dos defeitos utilizando as falhas consideradas boas e ruins ao invés de corrigidos x sem correção.

Figura 24 – Status do defeitos sugerida pelos participantes



Elaborado pelo autor

- Foi recomendado para a métrica ganho de tempo automação deveria utilizar outros valores para fazer essa comparação além do tempo gasto para executar o ciclo automatico.

7 Conclusão

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

7.1 Considerações finais e contribuições

Este trabalho propôs um conjunto de métricas capaz de avaliar a efetividade da execução de casos de testes. Inicialmente foi apresentado o referencial teórico desta pesquisa, com foco principal em testes de software e medição de software.

Foi constatado que produzir software com qualidade tem sido um grande desafio das organizações. Além disso que existe a necessidade de barateamento dos custos de produção e aumento da qualidade do produto gerado aumentando a competitividade da empresa junto ao mercado de trabalho. Desta forma, a atividade de testes tem se tornado cada vez mais importante no processo de desenvolvimento de software, pois ela aumenta a confiabilidade do software e garante a qualidade do produto. A atividade de testes tem se tornado cada vez mais custosa devido a sua importancia e mesmo assim não garante um produto livre de erros.

A solução proposta para garantir a eficiência dos testes de software foi a partir das métricas de testes que avaliariam a execução de testes de software. Como a área de estudo nesse caso é ampla, o trabalho focou apenas na execução de testes de software devido a sua importância.

Dentre as contribuições do projeto, podemos destacar a caracterização do processo, estratégias e práticas ligadas à atividades de testes de software e a utilização de métricas como forma de avaliação desse processo. O processo de criação e escolha de métricas pode ser utilizado por outros pesquisadores para escolha de métricas de acordo com o ambiente proposto. Outra contribuição é a descrição de um conjunto de métricas para avaliação de efetividade da execução de testes de software. Esse conjunto de métricas tem como objetivo fornecer informações para que a equipe acompanhe, avalie e consiga ter suporte para buscar melhorias na sua própria execução de testes de software.

Para criação das métricas foi utilizado a abordagem GQM onde é definido algumas métricas a partir dos objetivos em questão. Após o GQM o conjunto de métricas proposto foi:

- Número de falhas
- Velocidade

- Tempo estimado x Tempo gasto
- Correção dos defeitos
- Severidade das falhas
- Total de Testes automatizados
- Eficiência de automação
- Ganho de tempo de automação

Após a elaboração das métricas foi feito um grupo focal com especialistas na área de testes de software com o intuito de verificar as métricas produzidas. De uma forma geral, o conjunto proposto apresentou bons resultados diante da avaliação dos especialistas e pode ser utilizado como base para empresas que necessitem. As métricas correção de defeitos e ganho de tempo de automação receberam importantes sugestões de melhoria que causarariam efeitos positivos na utilização dessas métricas.

Algumas adaptações deverão ser necessárias na utilização das métricas propostas, já que a utilização da métrica pode variar de acordo com o processo que está sendo utilizado.

A validade de um estudo indica a confiabilidade dos resultados e em que medida os resultados são verdadeiros ou não pelo ponto de vista subjetivo dos pesquisadores. Wohlin et al. (2012) sugeriu um esquema de classificação dividido em 4 aspectos de validade que podem ser resumidos como:

- Validade de construção: Esse aspecto da validade reflete em que proporção as medidas operacionais estudadas realmente representam o que o pesquisador tem em mente e o que é investigado de acordo com a questão de pesquisa, se, por exemplo, as perguntas da entrevista não são interpretadas da mesma forma pelo pesquisador e pelas pessoas entrevistadas, há uma ameaça para construir validade.
- Validade interna: esse aspecto está é levado em conta quando as relações casuais são examinadas. Quando o pesquisador está investigando se um fator afeta um outro fator investigado, existe o risco de que o fator investigado também seja afetado por um terceiro fator. Se o pesquisador não tem conhecimento do terceiro fator e/ou não sabe até que ponto afeta o fator investigado, existe uma ameaça à validade interna
- Validade externa: Este aspecto da validade diz respeito a até que ponto é possível generalizar os achados, e até que ponto os achados são de interesse

para outras pessoas fora do caso investigado. Durante a análise da validade externa, o pesquisador tenta analisar em que medida os achados são relevantes para outros casos.

- Confiabilidade: esse aspecto diz respeito a até que ponto os dados e a análise dependem dos pesquisadores específicos. Hipoteticamente, se outro pesquisador conduzisse posteriormente o mesmo estudo, o resultado deveria ser o mesmo. Ameaças a esse aspecto de validade são, por exemplo, se questionários ou entrevistas não são claros. Para análise quantitativa, a contrapartida da confiabilidade é a validade da conclusão.

Observando esses 4 aspectos foi percebido que a validade externa foi o aspecto que sofreu uma maior ameaça na pesquisa. Devido ao fato do grupo focal ter sido validado por um grupo restrito de participantes que possuem uma experiência máxima de 3 anos na área de testes de software, além da pouca experiência os participantes já trabalham em uma mesma empresa, isso acaba restringindo o conhecimento e trazendo uma possível ameaça a validade externa. Existem muitas métricas de teste de software, consequentemente, dependendo do ambiente proposto, outras métricas poderiam ser utilizadas para avaliar a execução de casos de teste. Logo, o conjunto de métricas proposto é uma sugestão inicial para avaliação de efetividade de execução de testes de software mas com uma analise mais profunda de cada ambiente, esse conjunto pode sofrer alterações.

7.2 Trabalhos futuros

Foram identificadas as seguintes possibilidades de trabalhos futuros:

- Conduzir um estudo de caso para que possa verificar essa avaliação de execução de testes de software, definir valores de referência, além de compor novas métricas de acompanhamento de execução de testes de software a partir das métricas propostas.
- Validação mais profunda das métricas utilizando outras formas de validação experimental, permitindo certificar que as métricas são validas a partir de critérios já definidos.
- Aplicação em projetos reais, durante o desenvolvimento do software.
- Criação que um framework, onde o usuário indicaria as principais características do ambiente de teste da empresa e o framework selecionaria um conjunto de métricas mais adequado para o ambiente escolhido.

Referências

ABNT ISO 10015. *ABNT ISO 10015:2001 - Gestão da qualidade - Diretrizes para Treinamento*. 2001. Norma Regulamentadora.

AMMANN, P.; OFFUTT, J. *Introduction to software testing*. [S.l.]: Cambridge University Press, 2008.

ANSI/ IEEE Std. In: . [S.l.: s.n.], 1983. cap. IEEE Standard Glossary of Software Engineering Terminology, p. 1 – 40.

BANDEIRA, L. R. P. *Metodologia baseada em métricas de teste para indicação de testes a serem melhorados*. 2008. Dissertação (Pós-Graduação em Ciência da Computação) — UFPE.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The Goal Question Metric Approach. In: *Encyclopedia of Software Engineering*. [S.l.]: Wiley, 1994.

BERGHOUT, E.; SOLINGEN, R. van. The Goal/Question/Metric Method: a practical guide for quality improvement of software development. The McGraw-Hill Company, 1999. Disponível em: <https://www.researchgate.net/publication/243765439_The_GoalQuestionMetric_Method_A_Practical_Guide_for_Quality_Improvement_of_Software_Development>.

BORGES, C. D.; SANTOS, M. A. dos. Aplicações da técnica do grupo focal: fundamentos metodológicos, potencialidades e limites. *Revista da SPAGESP*, scielopepsic, v. 6, p. 74 – 80, 06 2005. ISSN 1677-2970.

BORGES, E. P. *Um modelo de medição para processos de desenvolvimento de software*. 2003. 154 p. Dissertação (Departamento de Ciência da Computação) — UFMG.

BURNSTEIN, I.; SUWANASSART, T.; CARLSON, R. Developing a Testing Maturity Model for software test process evaluation and improvement. In: *Proceedings International Test Conference 1996. Test and Design Validity IS* -. [S.l.: s.n.], 1996. p. 581 – 589.

CHEN, Y.; PROBERT, R. L.; ROBESON, K. Effective Test Metrics for Test Strategy Evolution. In: *Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research*. [S.l.]: IBM Press, 2004. (CASCON '04), p. 111 – 123.

DIJKSTRA, E. W. The Humble Programmer. *Commun. ACM*, ACM, New York, NY, USA, v. 15, n. 10, p. 859 – 866, oct 1972. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/355604.361591>.

DOMINGUES, A. L. S. *Avaliação de critérios e ferramentas de teste para programas OO*. 2002. Dissertação (Instituto de Ciências Matemáticas e de computação - ICMC) — USP.

FAROOQ, A.; DUMKE, R. R. Evaluation Approaches in Software Testing. In: . [S.l.: s.n.], 2008.

FAROOQ, A.; GEORGIEVA, K.; DUMKE, R. R. A meta-measurement approach for software test processes. In: *2008 IEEE International Multitopic Conference IS - SN - VO - VL -*. [S.l.: s.n.], 2008. p. 333 – 338.

FAROOQ, S. U.; QUADRI, S. M. *Effectiveness of Software Testing Techniques on a Measurement Scale*. 2010. Disponível em: <http://www.computerscijournal.org/?p=2214>.

FLORAC, W. A.; PARK, R. E.; CARLETON, A. Practical Software Measurement: Measuring for Process Management and Improvement. *Software Engineering Institute*, 1997.

GIL, A. C. *Como elaborar projetos de pesquisa*. 4. ed. São Paulo: Atlas, 2002.

GONDIM, S. M. G. Grupos focais como técnica de investigação qualitativa: desafios metodológicos. *Paidéia (Ribeirão Preto)*, scielo, v. 12, p. 149 – 161, 00 2002. ISSN 0103-863X. Disponível em: <http://www.scielo.br/scieloOrg/php/articleXML.php?lang=pt&pid=S0103-863X2002000300004>.

GOOGLE. *Google Acadêmico*. 2018. Disponível em: <https://scholar.google.com.br/intl/pt-BR/scholar/about.html>.

HARTMANN, D.; DYMOND, R. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: *AGILE 2006 (AGILE'06) IS - SN - VO - VL -*. [S.l.: s.n.], 2006. p. 6 – pp.–134.

HUMPHREY, W. S. *Managing the software process*. 1. ed. [S.l.]: Addisson Wesley, 1989.

HUTCHESON, M. L. *Software Testing Fundamentals*: Methods and Metrics. [S.l.]: John Wiley & Sons, 2003.

IEEE. *IEEE Standard Computer Dictionary*: A Compilation of IEEE Standard Computer Glossaries. [S.l.]: IEEE, 1990.

IEEE. *IEEE Xplore Digital Library*. 2018. Disponível em: <http://ieeexplore.ieee.org/Xplorehelp/#/overview-of-ieee-xplore/about-ieee-xplore>.

INTERNATIONAL STANDARD ISCO/IEC FDIS 15939, 2002. *Software Engineering - Software Measurement Process*.

JACOBS, J.; TRIENEKENS, J. J. M. Towards a metrics based verification and validation maturity model. In: *10th International Workshop on Software Technology and Engineering Practice IS - SN - VO - VL -*. [S.l.: s.n.], 2002. p. 123 – 128.

JONES, C. Software metrics: good, bad and missing. *Computer*, v. 27, n. 9, p. 98 – 100, 1994.

JURAN, J. M. *Controle da Qualidade*: conceitos, políticas e filosofia da qualidade. São Paulo: Makron, McGraw- Hill, 1991. v. 1.

KAN, S. H. *Metrics and models in software quality engineering*. [S.l.]: Addison Wesley, 2002.

KANER, C.; BOND, W. P. Software Engineering Metrics: What Do They Measure and How Do We Know? *10th International Software Metrics Symposium (Metrics 2004)*, Chicago, 2004.

KANIJ, T.; MERKEL, R.; GRUNDY, J. Performance assessment metrics for software testers. In: *2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE)*. [S.l.: s.n.], 2012. p. 63 – 65.

KITCHENHAM, B.; PFLEEGER, S.; FENTON, N. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 1995. Disponível em: <http://ieeexplore.ieee.org/document/489070/>.

KONTIO, J. Using the focus group method in software engineering: obtaining practitioner and user experiences. *International Symposium on Empirical Software Engineering*, Redondo Beach, 2004.

LAZIC, L.; MASTORAKIS, N. Cost Effective Software Test Metrics. *W. Trans. on Comp.*, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, v. 7, n. 6, p. 599 – 619, jun 2008. ISSN 1109-2750. Disponível em: <http://dl.acm.org/citation.cfm?id=1458369.1458372>.

LI, M.; SMIDTS, C. A ranking of software engineering measures based on expert opinion. *IEEE Transactions on Software Engineering*, v. 29, n. 9, p. 811 – 824, 2003.

LINNENKUGEL, U.; MÜLLERBURG, M. Test data selection criteria for (software) integration testing. *First International Conference on Systems Integration*, p. 709 – 719, 1990.

MAILEWA, A.; HERATH, J.; HERATH, S. A Survey of Effective and Efficient Software Testing. *The Midwest Instruction and Computing Symposium*, 2015. Disponível em: <http://www.micsymposium.org/mics2015/ProceedingsMICS_2015/Mailewa_2D1_41.pdf>.

MENEELY, A.; SMITH, B.; WILLIAMS, L. Validating Software Metrics: A Spectrum of Philosophies. *ACM Trans. Softw. Eng. Methodol.*, ACM, New York, NY, USA, v. 21, n. 4, p. 24:1 – 24:28, feb 2013. ISSN 1049-331X. Disponível em: <http://doi.acm.org/10.1145/2377656.2377661>.

MICHAELIS. *Michaelis Dicionário Brasileiro da Língua Portuguesa*. 2018. Disponível em: <http://michaelis.uol.com.br/moderno/portugues>.

MILES, M. B.; HUBERMAN, A. M. *Qualitative Data Analysis*: An Expanded Sourcebook. Londres: Sage: London, 1994.

MIRANDA, B.; ARANHA, E.; IYODA, J. Recommender systems for manual testing. In: *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement IS -*. [S.l.: s.n.], 2012. p. 201 – 210.

MYERS, G. J.; SANDLER, C.; BADGETT, T. *The Art of Software Testing*. 3rd. ed. [S.l.]: Wiley Publishing, 2011. ISBN 1118031962, 9781118031964.

NIRPAL, P.; KALE, K. A Brief Overview Of Software Testing Metrics. *International Journal on Computer Science and Engineering*, v. 3, 01 2011.

PHILLIP CROSBY. *Quality is Free*. [S.l.]: Mc-Graw-Hill, 1979.

PRESSMAN, R. *Software engineering*: a practioner's approach. 6. ed. [S.l.]: McGraw-Hill, 2006.

QUINTELLA, H.; QUEIROZ, R. Análise dos impactos na qualidade de software em instituições financeiras segundo a norma ISO/IEC 9126 na adoção das práticas de testes do modelo CMMI. 2006. Disponível em: <http://www.producao.uff.br/conteudo/rpep/volume62006/RelPesq_V6_2006_03.pdf>. Acesso em: 19/05/2017.

RAFI, D. M. et al. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In: *2012 7th International Workshop on Automation of Software Test (AST) IS - SN - VO - VL -*. [S.l.: s.n.], 2012. p. 36 – 42.

ROCHA, A. R. C. da; SOUZA, G. dos S.; BARCELLOS, M. P. *Medição de Software e Controle Estatístico de Processos*. Brasilia: PBQP Software, 2012.

SCHNAIDER, L. et al. Uma Abordagem para Medição e Análise em Projetos de Desenvolvimento de Software. *Simpósio Brasileiro de Qualidade de Software*, Brasília, 2004.

SCRIVEN, M. *Evaluation Thesaurus*. 4. ed. [S.l.]: Sage Publications Inc., 1991.

SOARES, T. N. S. *Uma Abordagem para Estimativa de Custo em Teste Automático*. 2017. Dissertação (Engenharia de Software) — C.E.S.A.R.

SOFTEX. *MPS.BR - Melhoria de Processo de Software Brasileiro - Guia Geral.* [S.l.], 2016.

SOMMERVILLE, I. *Software Engineering*. 9. ed. [S.l.]: Pearson, 2007. (International computer science series Software engineering).

SUWANNASART, T.; SRICHAIVATTANA, P. A set of measurements to improve software testing process. *Proceedings of the 3rd National Computer Science and Engineering Conference*, 1999.

TERENCE, A. C. F.; ESCRIVÃO FILHO, E. *Abordagem quantitativa, qualitativa e a utilização da pesquisa-ação nos estudos organizacionais*. Fortaleza: [s.n.], 2006/10. Disponível em: <http://www.abepro.org.br/biblioteca/enegep2006_tr540368_8017.pdf>.

TIAN, J. *Software Quality Engineering*. 1. ed. [S.l.]: Wiley-IEEE Press, 2005.

UNTERKALMSTEINER, M. et al. Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review. *IEEE Transactions on Software Engineering*, v. 38, n. 2, p. 398 – 424, 2012.

VICENTE, A. A. *Definição e gerenciamento de métricas de teste no contexto de métodos ágeis*. 2010. Dissertação (Mestrado) — Universidade de São Paulo. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-23062010-083439/>.

WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435.