PERCEPÇÃO DO ESFORÇO NO PROCESSO DE TESTE DE SOFTWARE

Thiago Silva de Souza

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Prof. Guilherme Horta Travassos

Rio de Janeiro
Outubro de 2018

PERCEPÇÃO DO ESFORÇO NO PROCESSO DE TESTE DE SOFTWARE

Thiago Silva de Souza

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof. Guilherme Horta Travassos, D.Sc.

_____
Profa. Ana Regina Cavalcanti da Rocha, D.Sc.

_____
Prof. Eber Assis Schmitz, Ph.D.

_____
Prof. Adenilso da Silva Simão, D.Sc.

_____
Prof. Marcos Kalinowski, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 2018

Souza, Thiago Silva de

Percepção do Esforço no Processo de Teste de Software. – Rio de Janeiro: UFRJ/COPPE, 2018.

XXV, 264 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 194-201.

1. Teste de software. 2. Estimativa de esforço. 3. Fatores de esforço. I. Travassos, Guilherme Horta. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha filha Alice.

AGRADECIMENTOS

Ao meu orientador, Prof. Guilherme Horta Travassos, por toda atenção dada a este trabalho e por toda dedicação, paciência, compreensão e confiança demonstradas.

Aos professores Ana Regina Cavalcanti da Rocha, Eber Assis Schmitz, Adenilso da Silva Simão e Marcos Kalinowski por, gentilmente, aceitarem o convite para participar da banca examinadora deste trabalho e pelas contribuições e sugestões dadas.

Novamente ao Prof. Guilherme Horta Travassos e à Profa. Ana Regina Cavalcanti da Rocha, por todos os ensinamentos transmitidos durante as disciplinas do curso de Doutorado.

Ao meu pai João Roberto e à minha mãe Selma, pelo amor incondicional e por todo sacrifício feito para que seus filhos se tornassem pessoas corretas, trabalhadoras e estudiosas.

À minha amada esposa Barbara, que sempre esteve ao meu lado nesta caminhada tão difícil, com muito amor e compreensão, suportando as minhas ausências e me ajudando em momentos de extrema dificuldade.

Ao meu sogro Reginaldo e à minha sogra Suely, pelo apoio dado ao longo deste período e, especialmente, pela ajuda com a criação da Alice.

A todos os meus familiares, em especial à minha irmã Thielle e à minha sobrinha Lara.

Aos professores Arnaldo Vieira da Rocha Filho e Wilson Chagas de Araújo, grandes responsáveis pelo meu fascínio pela ciência, por toda ajuda e apoio prestados, imprescindíveis para minha formação profissional e, consequentemente, para a realização desse curso.

Ao Prof. Alessandro de Almeida Castro Cerqueira, ex-coordenador dos cursos de Tecnologia da Informação da UNIGRANRIO, por todo o incentivo e apoio prestados ao longo desses anos.

Ao Prof. Daniel de Oliveira, atual coordenador dos cursos de Tecnologia da Informação da UNIGRANRIO, por aguardar tão pacientemente a conclusão desse curso.

Ao Prof. Marco Antônio de Melo Britto e a todos os demais colegas professores da UNIGRANRIO que, de alguma forma, me ajudaram ou que torceram para que eu concluísse essa etapa.

Aos alunos e ex-alunos dos cursos de Tecnologia da Informação da UNIGRANRIO, por proporcionarem tantos momentos de alegria na minha carreira docente.

Aos amigos que dividiram comigo as alegrias e angústias do Doutorado: Victor Vidigal, Breno França, Paulo Sérgio Medeiros, Talita Vieira, Rebeca Motta, Luciana Nascimento, Wladmir Chapetta, Fábio Farzat, Hélvio Jerônimo, Hilmer Nery, Rafael Maiani e todos os demais colegas do grupo ESE.

Aos amigos que, mesmo distantes, ficaram na torcida pela conclusão desse curso: Kele Teixeira Belloze e Cláudio César Fraga Cavalcante.

Ao Serviço Federal de Processamento de Dados (SERPRO), pela permissão para cursar o Doutorado durante o horário de trabalho.

Aos chefes que tive no SERPRO durante o período de Doutorado, Paulo Ferraz de Souza Júnior e José Eduardo Alves de Carvalho pela compreensão demonstrada durante a realização deste trabalho.

Aos colegas de SERPRO que, de alguma forma, contribuíram para a realização deste trabalho, em especial ao Alexandre Araújo Moreira (*in memoriam*) pelas discussões sobre o modelo conceitual para representação da percepção do esforço de teste.

A todos os participantes do *survey*, pelo precioso tempo dedicado e pelo empenho demonstrado na realização do estudo.

Ao amigo Leonardo Silva, pelo apoio na implementação da aplicação Web que representa o modelo desenvolvido neste trabalho.

Ao pessoal da secretaria do PESC, por toda ajuda prestada durante esses anos, especialmente ao funcionário Gutierrez da Costa, por sua eficiência em tratar dos meus assuntos acadêmicos.

Por fim, agradeço a todos aqueles que, de alguma forma, contribuíram na elaboração deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

PERCEPÇÃO DO ESFORÇO NO PROCESSO DE TESTE DE SOFTWARE

Thiago Silva de Souza

Outubro/2018

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

Contexto: Diversas ações típicas de projetos de software são realizadas com base no esforço de teste e na compreensão que se tem a seu respeito. Esta compreensão passa pela identificação e caracterização dos fatores que afetam o esforço de teste, levando-se em conta diferentes níveis, tipos, técnicas e formas de execução dos testes. *Objetivos*: Construir um modelo da percepção do esforço no processo de teste de software que represente os fatores que influenciam o esforço em diferentes configurações de estratégia de teste e que seja capaz de apoiar a tomada de decisão em projetos de teste de software. *Método*: Revisão sistemática da literatura (RSL), estudo de observação em campo e *survey* com profissionais de teste de software. *Resultados*: Os estudos realizados possibilitaram a identificação e caracterização de 62 fatores de esforço distintos. Com o *survey* foi possível mapear a prevalência desses fatores em seis atividades típicas de um processo de teste de software, considerando seis diferentes configurações de estratégia de teste. Com base nos resultados dos três estudos, foi desenvolvido um modelo da percepção do esforço de teste de software, adaptável a diferentes configurações de estratégia de teste. *Conclusão*: O modelo proposto mostrou-se útil para apoiar atividades gerenciais que devem levar em conta a compreensão sobre o esforço de teste, dentre as quais a estimativa de esforço.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

EFFORT PERCEPTION IN THE SOFTWARE TESTING PROCESS

Thiago Silva de Souza

October/2018

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

Context: Some typical actions in software projects are taken based on the test effort and the understanding one has about it. This understanding involves the identification and characterization of the factors influencing the test effort, taking into account different test levels, types, techniques, and execution forms. *Goals*: Building a model of effort perception in software testing process that represents the factors influencing the effort in different test strategy configurations, supporting decision making in software testing projects. *Method*: Systematic Literature Review (SLR), observation study in industry, and survey with software test practitioners. *Results*: The studies supported the identification and characterization of 62 different effort factors. The survey led to identify the prevalence of these factors in six typical activities of the software testing process considering six different test strategy configurations. Based on the results of the three studies, a software test effort perception model was developed, adaptable to different test strategy configurations. *Conclusion*: The proposed model was considered useful to support managerial activities that must take into account the comprehension about the test effort, among them the effort estimation.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1. COCOMO – *COnstructive COst MOdel.*
2. CRUD – *Create, retrieve, update and delete*.
3. ECEC – *Effort and Cost Estimation Corpus.*
4. EO – Estudo de Observação.
5. EV – *Engineering Village*.
6. GQM – *Goal, Question, Metric*.
7. HH – Homem-hora.
8. IA – Inteligência artificial.
9. KWIC – *Key Word in Context*.
10. MRAE – *Mean Relative Absolute Error*.
11. MRE – *Mean Relative Error*.
12. MMRE – *Mean Magnitude of Relative Error*.
13. NSM/MIS – *New, Search, Modify/Management Information System functionality*.
14. PICO – *Population, intervention, comparison and outcome*.
15. QAMH – Questionário de Avaliação do Modelo HyperTest.
16. QCP - Questionário de Caracterização do Participante.
17. QE – Questão de pesquisa específica.
18. QPET - Questionário de Percepção do Esforço de Teste.
19. RCPT – Relatório de Conclusão do Projeto de Teste.
20. RSL – Revisão Sistemática da Literatura.
21. SDC – *System Development Corporation*.
22. SOAP – *Simple Object Access Protocol*.
23. Softex - Associação para Promoção da Excelência do Software Brasileiro.
24. SUT – *System under test*.

25. SWEBoK - *Software Engineering Body of Knowledge*.

26. TCLE - Termo de Consentimento Livre e Esclarecido.

27. TPA – *Test Point Analysis*.

28. V&V – Verificação e Validação.

29. WoS – *Web of Science*.

1 INTRODUÇÃO[1]

Este capítulo está dividido em seis seções. A seção 1.1 introduz a motivação deste trabalho e o contexto no qual ele está inserido. A seção 1.2 descreve o problema tratado nesta proposta de tese. A seção 1.3 apresenta as questões de pesquisa investigadas. A seção 1.4 mostra os objetivos e o escopo deste trabalho. A seção 1.5 descreve a metodologia utilizada nessa pesquisa. Por fim, a seção 1.6 descreve a estrutura segundo a qual este trabalho está organizado.

1.1 Motivação e Contexto

Uma das medidas mais importantes em projetos de software é o esforço. A medida de esforço é frequentemente utilizada para orientar a alocação de recursos, definição de cronogramas, cálculo de custos e outras atividades gerenciais que requerem algum tipo de tomada de decisão. Apesar de ser útil para a realização de diversas atividades típicas de projetos de software, a medida de esforço é especialmente importante como *estimativa*. Em geral, o esforço é estimado no início dos projetos visando apoiar gerentes no planejamento dos projetos sob sua responsabilidade.

Estimar esforço, portanto, é uma das atividades mais críticas em um projeto de software. A literatura técnica de Engenharia de Software oferece inúmeros modelos para estimativa de esforço (muitas vezes chamado de "custo"), baseados nas mais variadas técnicas de estimativa (JøRGENSEN; SHEPPERD, 2007). Cada modelo contempla uma série de fatores ou direcionadores (*drivers*) que se acredita influenciar no esforço do projeto sob determinadas condições de contexto. Um *fator de esforço* é qualquer característica do ambiente de software que exerce uma influência significativa sobre o esforço requerido para realizar alguma tarefa relacionada ao desenvolvimento de um software.

Modelos de estimativa como o COCOMO (BOEHM, 1981) (BOEHM *et al.*, 2000) fornecem uma estimativa de esforço do projeto como um todo, cobrindo diversas fases do ciclo de desenvolvimento de software. Consequentemente, o COCOMO, assim

[1] Uma versão preliminar deste capítulo foi publicada parcialmente no resumo *Towards Effort Estimation in Software Testing Projects*, apresentado na forma de pôster no *ICSE 2017 PhD and Young Researchers Warm Up Symposium* (SILVA-DE-SOUZA, 2014).

como diversos outros modelos, realiza suas estimativas com base em fatores relativamente genéricos, que não capturam a variabilidade intrínseca aos processos e projetos de desenvolvimento de software – um determinado *cost driver* (fator de esforço) do COCOMO, e.g. "uso de ferramentas de software", pode exercer influência em sentidos e proporções diferentes dependendo da atividade do processo e do contexto do projeto. Modelos de estimativa voltados para atividades específicas do ciclo de desenvolvimento, tais como Requisitos (VAZ, 2013) (MENDES; VAZ; MURADAS, 2016) ou Testes, tendem a ser mais precisos na identificação dos fatores que influenciam o esforço daquela atividade. Uma maior precisão na identificação dos fatores de esforço pode resultar em estimativas com maior grau de acurácia.

Uma das atividades que mais demandam esforço em projetos de desenvolvimento de software é a atividade de Teste de Software. Geralmente, o esforço de teste é obtido a partir do esforço total do projeto. Pressman (PRESSMAN, 2006) chega a afirmar que a atividade de teste, combinada com a depuração subsequente, consome de 30 a 40% do esforço total do projeto. Os modelos de estimativa de esforço tradicionais observam o processo de desenvolvimento como um todo, tomando como insumo alguma especificação do software para fornecer uma noção de todo o esforço a ser despendido com o desenvolvimento do software. Essa característica pode ser reflexo da influência do modelo de ciclo de vida em cascata sobre os primeiros modelos de estimativa. No entanto, ao longo dos anos a atividade de teste foi adaptada a novos modelos de processo, como o modelo V (FORSBERG; MOOZ, 1991) e os modelos iterativos, passando a absorver particularidades dificilmente levadas em conta pelos métodos tradicionais de estimativa de esforço.

Estimar o **esforço de teste** é importante para qualquer organização que desenvolva software pelos motivos já citados anteriormente mas, em especial, para aquelas organizações que contam com equipes independentes de teste e para aquelas que realizam *outsourcing*, as chamadas "fábricas de teste" (SOFTEX, 2011). *Outsourcing* é o termo comumente utilizado na indústria para representar uma forma de organização em que uma empresa transfere, via subcontratação, a realização de parte de seu processo produtivo para outra empresa. Países como Índia, Paquistão e Malásia são reconhecidos internacionalmente por concentrar grande parte das organizações que realizam *outsourcing* de atividades relacionadas ao desenvolvimento de software. Essa afirmação pode ser corroborada pelos gráficos de tendência de buscas no Google gerados pela ferramenta *Google Trends* (GOOGLE, 2018). O *Google Trends* é uma

ferramenta que permite analisar a frequência de buscas por determinados termos em várias regiões do mundo. A Figura 1 e a Figura 2 representam, respectivamente, as tendências de busca no *Google* pelas expressões "*effort estimation*" e "*cost estimation*" entre 2004 e 2018. A tendência de busca é apresentada através de um índice entre 0 e 100, em que 100 representa a região que atingiu o maior volume de buscas por um determinado termo em um período estabelecido. As representações indicam, especialmente na Figura 2, que a maior parte das buscas por tais expressões provêm daqueles países que concentram empresas de *outsourcing*.

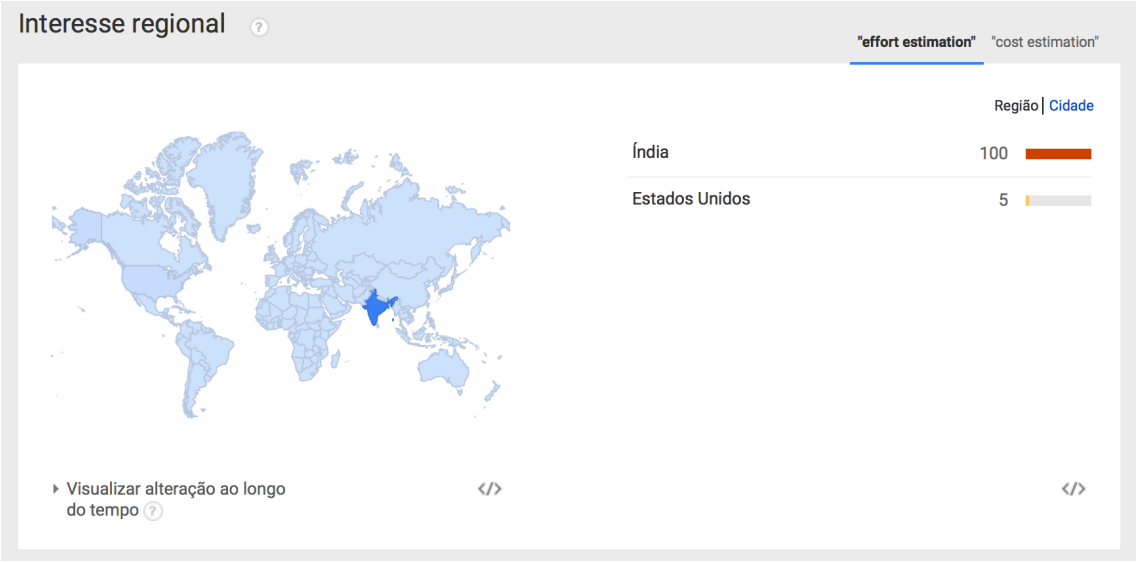


Figura 1: Tendência de buscas pela expressão "*effort estimation*".

Figura 2: Tendência de buscas pela expressão "*cost estimation*".

Apesar de não figurar entre os países mais interessados nas expressões pesquisadas, o que pode ser explicado pela diferença de idioma, o Brasil também conta com diversas empresas que atuam nesse ramo e que provavelmente necessitam estimar o esforço para realizar as atividades para as quais são contratadas. Quanto às empresas que atuam especificamente como fábricas de teste, essa percepção é reforçada pela preocupação da Associação para Promoção da Excelência do Software Brasileiro (Softex) em fornecer um guia com orientações para a implementação do modelo de referência MR-MPS-SW em organizações do tipo "fábrica de teste" (SOFTEX, 2011), onde o interesse pela estimativa de esforço de teste é evidenciado através do resultado esperado GPR4, do processo Gerência de Projetos.

A importância de modelos de estimativa de esforço de teste é potencializada à medida que a demanda por *outsourcing* de testes aumenta. Shah, Sinha & Harrold (SHAH; SINHA; HARROLD, 2011) citam que a atividade de teste é a segunda mais terceirizada na indústria de software (atrás da atividade de codificação) e que a indústria de *outsourcing offshore* de testes de software cresce a uma taxa de 20% ao ano.

Considerando essa necessidade de se estimar especificamente o esforço inerente ao teste de software, nos últimos anos diversos modelos de estimativa foram propostos (ALMEIDA; ABREU; MORAES, 2009) (ARANHA; BORBA, 2007) (ARANHA; BORBA, 2009) (NAGESWARAN, 2001) (SILVA; ABREU; JINO, 2009) (XIAOCHUN *et al.*, 2008) (ZHU; ZHOU; CHEN, 2009). Tais modelos lançam mão das mais diferentes técnicas de estimativa e consideram variados fatores de influência sobre o esforço. No entanto, as limitações e restrições de uso dos modelos disponíveis evidenciam a necessidade de novos modelos de estimativa, que capturem as variáveis de contexto às quais estão expostos os projetos de teste e que fornecem estimativas com maior grau de acurácia.

Embora a medida de esforço seja especialmente importante no contexto de modelos de estimativa, diversas outras atividades típicas de projetos de software podem ser apoiadas pela mensuração do esforço ou, simplesmente, pelo conhecimento a respeito dos fatores que o afetam. No contexto de teste de software, por exemplo, o conhecimento sobre quais fatores afetam o esforço e de que forma afetam pode ser útil para apoiar a tomada de decisão nas seguintes situações:

- *Seleção de profissionais para compor equipes de teste*. Levando-se em conta os objetivos e restrições de um determinado projeto, um gerente de testes pode

selecionar os membros da equipe com base em determinados fatores que, sabidamente, influenciam o esforço de teste.

- *Priorização ou seleção de casos de teste a serem automatizados*. A medida de esforço pode ser utilizada como critério para priorização ou seleção de casos de teste a serem automatizados. Automatizar testes é uma abordagem de projeto que deve passar por uma decisão bem fundamentada, tendo em vista os custos decorrentes da automação. Levando-se em conta o custo-benefício da automação, gerentes de testes podem decidir que somente um subconjunto dos casos de teste de um projeto deve ser automatizado. Para chegar à essa conclusão é fundamental conhecer quais fatores afetam o esforço no contexto de testes automatizados. Eventualmente, um caso de teste caracterizado por fatores que aumentam demasiadamente o esforço de automação pode ser preterido ou despriorizado.
- *Melhoria de processos de teste*. O esforço de um projeto de software é naturalmente influenciado pelo processo utilizado. Um processo de teste demasiadamente prescritivo pode elevar significativamente o esforço do projeto de teste. Cabe, no entanto, ao gerente de teste e à equipe do projeto equilibrar o esforço aplicado com o valor resultante desse esforço. Portanto, processos de teste podem ser redesenhados ou evoluídos tomando como base os fatores de esforço relacionados às suas tarefas e produtos de trabalho. Desta forma, em uma nova versão do processo de teste, aqueles fatores que aumentam o esforço poderiam ser evitados, enquanto que os fatores que diminuem o esforço poderiam ser estimulados.
- *Terceirização*. Como já discutido nesta seção, a demanda por *outsourcing* de testes tem aumentado nos últimos anos. Até mesmo organizações que contam com equipes de teste internas têm demandado esse serviço a terceiros. A decisão por testar internamente ou terceirizar os testes de um projeto pode passar pela análise das características do projeto com relação ao esforço. Assim, uma organização poderia decidir por não testar internamente um sistema que ela própria tenha desenvolvido porque, eventualmente, sua equipe de testes é caracterizada por fatores que aumentariam o esforço, sendo mais vantajoso financeiramente contratar outra empresa.

1.2 Problema

A literatura sobre estimativa de esforço em teste de software, de forma geral, não apresenta evidências que apoiem a compreensão do fenômeno do esforço. Diversos trabalhos dessa área abordam o problema com um viés quantitativo, mas não se ocupam de explicar porque o esforço acontece (VAN VEENENDAAL, 1995) (NAGESWARAN, 2001) (XIAOCHUN et al., 2008) (ZHU; ZHOU; CHEN, 2009). A compreensão deste fenômeno deve passar pela identificação dos fatores que o afetam. No entanto, apesar da existência de diversos trabalhos relacionando certos fatores ao esforço de teste de software, encontrados principalmente em modelos de estimativa, algumas questões ainda afetam a compreensão deste fenômeno e, consequentemente, os resultados das estimativas, podendo impor riscos aos projetos:

- Diversos modelos de estimativa oferecem uma estimativa de esforço de teste levando em consideração características intrínsecas ao desenvolvimento do software como um todo (BOEHM, 1981) (BOEHM *et al.*, 2000) (JONES, 2007), sem evidência de que tais características efetivamente afetam o esforço de teste. No entanto, os fatores que influenciam o esforço de teste de software são claramente diferentes daqueles fatores que afetam o esforço de outras atividades de desenvolvimento de software (LU; YIN, 2013), podendo, inclusive, exercer diferentes níveis de influência dependendo do contexto do projeto. É necessário, portanto, evidenciar quais fatores de projeto afetam o esforço de teste e avaliar como eles devem ser considerados para estimar o esforço de teste;
- A terminologia sobre estimativa de esforço não é utilizada de modo consistente na literatura técnica (GRIMSTAD; JøRGENSEN; MOLøKKEN-ØSTVOLD, 2006), o que pode levar gestores a tomarem decisões equivocadas, além de representar um obstáculo para a melhoria da acurácia das estimativas;
- A unidade de medida utilizada para representar o esforço predominantemente físico e repetitivo presente na indústria manufatureira desde o final do século XIX ("homem-hora" e todas as suas variações), da qual a indústria de software se apropriou, não se mostra adequada para representar o esforço empregado nas atividades de desenvolvimento de software, caracterizado pela criatividade, aplicação de técnicas e pelo raciocínio lógico (BROOKS JR, 1995). É preciso,

portanto, definir uma unidade de medida que capture essas diferenças e melhor represente o esforço intelectual inerente a projetos de software;

- Grande parte dos modelos de estimativa específicos para teste de software, por exemplo os propostos por Aranha e Borba (ARANHA; BORBA, 2007), Silva, Abreu e Jino (SILVA; ABREU; JINO, 2009) e Xiaochun *et al*. (XIAOCHUN *et al.*, 2008), estimam apenas o esforço da **execução** de testes. Entretanto, um processo de teste contempla outras atividades, tais como planejamento, projeto (*design*) & implementação, configuração de ambiente e análise dos resultados (ISO/IEC/IEEE, 2013). Desta forma, é preciso considerar o esforço inerente a essas atividades ao estimar o esforço total em projetos de teste.

As questões apresentadas expõem lacunas deste campo de pesquisa, bem como fragilidades dos modelos existentes e ratificam a necessidade de um modelo que considere as atividades típicas do processo de teste de software e o contexto de cada projeto e organização. Esse novo modelo não necessita ser necessariamente um modelo de estimativa, mas um modelo que apoie a tomada de decisão em projetos e, eventualmente, seja aproveitado por novos modelos de estimativa de esforço de teste.

1.3 Questões de Pesquisa

Considerando as questões-problema discutidas na seção anterior, a principal questão de pesquisa deste trabalho é formulada da seguinte forma:

Quais fatores podem influenciar o esforço nas atividades do processo de teste de software e de que forma?

Essa questão de pesquisa principal foi decomposta nas seguintes questões de pesquisa específicas, nomeadas com o prefixo "QE" (questão específica), que norteiam os diferentes estudos apresentados nos capítulos 3, 4 e 5, e no Apêndice D:

- **QE1**: O que se entende por **esforço** em teste de software e como ele pode ser medido?
- **QE2**: Quais **fatores** influenciam o esforço em teste de software?
    - **QE2.1**: Que **modelos de estimativa de esforço** utilizam esses fatores?
        - **QE2.1.1**: Esses modelos são baseados em que tipos de técnicas?

        - **QE2.1.2**: Esses modelos são adequados para qual contexto de teste?
        - **QE2.1.3**: Como esses modelos têm sido empiricamente avaliados?
- **QE3**: Qual a relação linguística entre os termos "esforço" e "custo" em Engenharia de Software?
- **QE4**: Como a Linguística pode apoiar na resolução de confusões terminológicas em Engenharia de Software?
- **QE5**: Quais fatores de esforço relacionados ao processo de teste de software podem ser identificados em projetos industriais de larga escala?
    - **QE5.1**: Qual a influência exercida por cada fator em cada tarefa do processo de teste de software?
- **QE6**: Quais atividades do processo de teste de software demandam mais esforço?
    - **QE6.1**: A ordem das atividades que demandam mais esforço varia de acordo com a configuração de estratégia de teste?
- **QE7**: Quais fatores de esforço relacionados ao processo de teste de software são percebidos pelos profissionais da área?
    - **QE7.1**: Qual o impacto de cada fator sobre o esforço em cada atividade do processo de teste de software?
    - **QE7.2**: Como cada fator pode ser observado em cada atividade do processo de teste de software?
    - **QE7.3**: Os fatores de esforço variam de acordo com a configuração de estratégia de teste?
- **QE8**: O uso do modelo HyperTest afeta a estimativa baseada em opinião de profissionais de software sobre o esforço de teste?
    - **QE8.1**: Quais atividades do processo de teste têm suas estimativas afetadas com o uso do HyperTest?
    - **QE8.2**: Quais fatores de esforço são mais frequentemente negligenciados em estimativas que não se baseiam no HyperTest?
- **QE9**: Como o modelo HyperTest pode contribuir para a estimativa de esforço em projetos de teste de software?

1.4 Objetivos e Escopo do Trabalho

O objetivo principal deste trabalho é propor um modelo da percepção do esforço no processo de teste de software, em resposta à questão de pesquisa apresentadas. Este objetivo principal pode ser decomposto e melhor detalhado nos seguintes sub-objetivos:

- Investigar o que se entende por esforço e termos correlacionados na literatura técnica e na indústria, para propor definições que minimizem a confusão conceitual existente e, consequentemente, permitam a elaboração de modelos de estimativa mais realistas e precisos;
- Identificar na literatura técnica e na indústria quais fatores exercem influência sobre o esforço de teste de software, em que sentido e por que, levando-se em conta diferentes estratégias de teste de software;
- Definir um modelo que represente a percepção de profissionais da indústria de software a respeito do esforço nas atividades típicas de um processo de teste de software, levando em conta as variáveis de contexto de cada projeto e organização;
- Avaliar o modelo proposto com o apoio de profissionais envolvidos em atividades de teste de software com o intuito de caracterizar sua utilidade.

1.5 Metodologia da Pesquisa

Desde o seu início, esta pesquisa planejava executar dois grandes estudos: 1) revisão sistemática da literatura e 2) *survey* na indústria. Esses estudos continuaram fazendo parte do escopo da pesquisa, entretanto, dois outros estudos se fizeram necessários para apoiar a análise dos resultados da revisão sistemática: a *análise linguística*, apresentada no Apêndice D, e o *estudo de observação*, descrito no capítulo 4. Tais estudos são tratados aqui como estudos complementares. Todas essas etapas estão representadas na Figura 3.

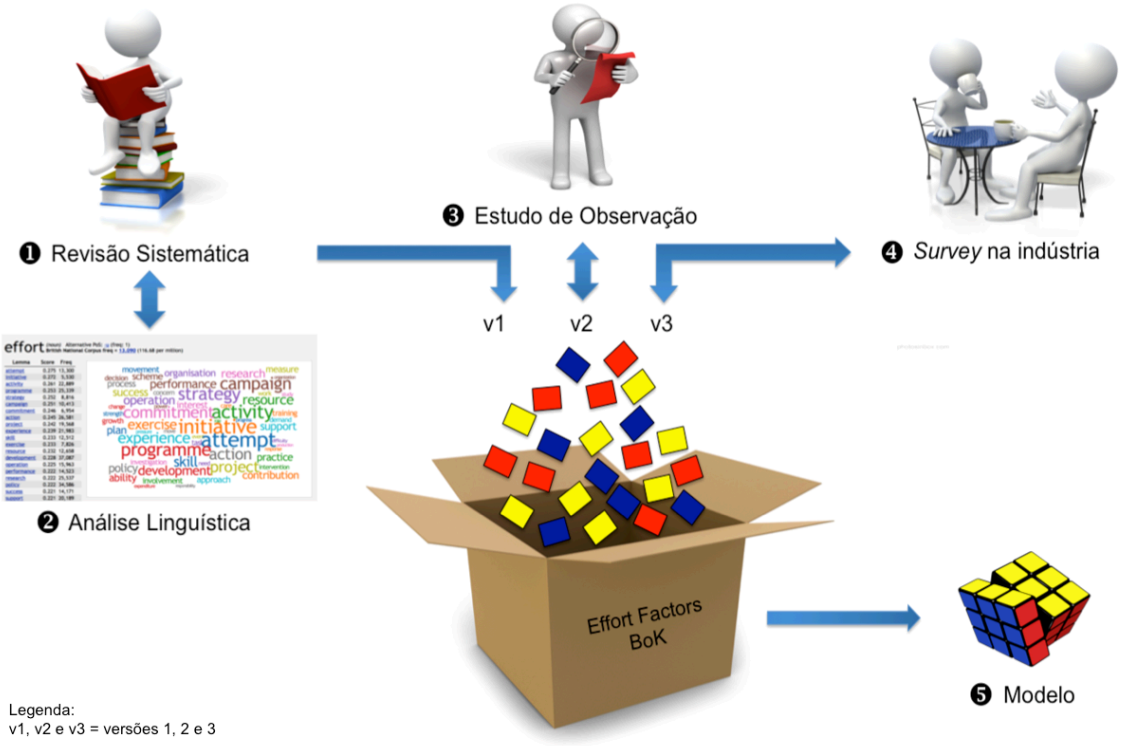


Figura 3: Visão geral das etapas que compõem a pesquisa.

Inicialmente, o objetivo deste trabalho era definir um modelo de estimativa de esforço de teste de software. Tal modelo seria construído a partir de um grande conjunto de fatores de esforço que seriam identificados na literatura técnica e observados e medidos em projetos reais. No entanto, à medida que a pesquisa avançava, percebeu-se que seria inviável definir um modelo de estimativa baseado em dados de projetos contemplando tantos fatores de esforço. Assim, houve um redirecionamento nos objetivos da tese, a qual passou a objetivar a definição de um modelo da percepção do esforço no processo de teste de software.

1.5.1 Revisão sistemática da literatura

A primeira atividade realizada nesta pesquisa foi uma revisão sistemática da literatura (descrita no capítulo 3). Tal revisão procurou identificar os fatores de esforço de teste descritos na literatura técnica de Engenharia de Software (ES), bem como caracterizar os modelos de estimativa que utilizam esses fatores.

A revisão sistemática foi realizada entre janeiro/2014 e fevereiro/2015 e atualizada em duas oportunidades: de dezembro/2015 a fevereiro/2016 e em julho/2018.

A longa duração desse estudo se justifica pela disparidade conceitual e pela ausência de informações de contexto observada nos artigos selecionados.

Apesar de toda confusão conceitual observada na literatura técnica, a revisão sistemática proporcionou a identificação de diversos fatores que podem influenciar o esforço em teste, bem como permitiu perceber as lacunas deixadas pelos modelos de estimativa existentes, o que serviu para direcionar este trabalho.

1.5.2 Análise linguística

Como a revisão sistemática não obteve de imediato respostas às suas questões de pesquisa a partir dos artigos, dada a confusão conceitual presente na área de ES, houve a necessidade de recorrer à uma análise pautada na Linguística, baseada em uma análise etimológica e uma análise de *corpus*: a primeira com objetivo de identificar o significado original atribuído aos termos "esforço" e "custo"; já a segunda, com o objetivo de identificar em que sentido os termos "esforço" e "custo" são utilizados atualmente de forma geral e de forma específica na área de Engenharia de Software.

Esse estudo foi realizado de outubro/2015 a dezembro/2015 e contou com o apoio de um estudante do último período do curso de Letras da UFRJ. A realização desse estudo serviu para melhor compreender o conceito de *esforço* em projetos de software e, consequentemente, para fundamentar a exclusão de determinadas variáveis de contexto previamente classificadas como fatores de esforço na revisão sistemática.

1.5.3 Estudo de observação

Um estudo de observação na indústria se fez necessário para melhor compreender quais fatores podem afetar o esforço em projetos de teste de software e de que maneira esses fatores afetam o esforço (aumentando ou diminuindo). Esse estudo foi realizado ao longo do ano de 2015 na empresa onde o pesquisador atua. Dentre suas contribuições, o estudo possibilitou a confirmação de parte dos fatores identificados na revisão sistemática, bem como a identificação de quatro novos fatores de esforço.

1.5.4 Survey na indústria

O principal estudo realizado para subsidiar a definição do modelo proposto neste trabalho foi um *survey* com profissionais da indústria envolvidos em atividades de teste de software. Esse *survey* foi planejado no final de 2017 e aplicado no início de 2018. A aplicação do *survey* na indústria teve três objetivos: 1) avaliar a pertinência dos fatores identificados na revisão sistemática; 2) identificar outros fatores que influenciam o esforço de teste e 3) identificar qual a influência que cada fator pode exercer sobre o esforço de teste e em quais circunstâncias. Os resultados desse estudo foram, portanto, consolidados com os resultados dos três estudos anteriores em um modelo de percepção do esforço de teste, que se apresenta como a principal contribuição deste trabalho.

1.6 Organização do Trabalho

Esta Tese de Doutorado está organizada em sete capítulos. O capítulo 1 contém a introdução deste trabalho, ressaltando a sua motivação, o problema tratado e os objetivos da pesquisa. O capítulo 2 discorre sobre o referencial teórico deste trabalho. O capítulo 3 descreve uma revisão sistemática da literatura conduzida com objetivo de investigar os modelos de estimativa e fatores de esforço em teste de software. O capítulo 4 descreve um relato de experiência realizado com o objetivo de observar fatores que influenciam o esforço em um projeto de teste. O capítulo 5 descreve um *survey* realizado com profissionais da indústria de software. O capítulo 6 apresenta um modelo da percepção do esforço de teste de software. O capítulo 7 apresenta a conclusão deste trabalho. Em seguida, é apresentada a lista de referências bibliográficas utilizadas. Finalmente, são apresentados na forma de apêndices os diversos instrumentos utilizados na pesquisa.

2 REFERENCIAL TEÓRICO

Neste capítulo é discutido o referencial teórico deste trabalho por meio de quatro seções. A seção 2.1 apresenta conceitos fundamentais sobre teste de software. A seção 2.2 introduz aspectos sobre estimativa de esforço em projetos de software. A seção 2.3 descreve os principais trabalhos relacionados à estimativa de esforço de teste de software. A seção 2.4 apresenta as considerações finais do capítulo.

2.1 Teste de Software

O teste de software é um processo relacionado ao desenvolvimento de software que tem como principal objetivo revelar *falhas* por meio da análise *dinâmica* de programas (ISO/IEC/IEEE, 2013). Trata-se de uma análise dinâmica porque há a necessidade de colocar o software em execução para avaliar se determinados dados de entrada geram determinados dados de saída (resultados esperados). Para delimitar o escopo do teste, além de ressaltar sua natureza dinâmica, é importante esclarecer os tipos de incidentes comuns em um projeto de software (ISO/IEC/IEEE, 2013) (FIRESMITH, 2014):

- *Erro*: também conhecido como engano, representa a ação humana cometida ao interpretar incorretamente uma informação relativa ao software, que pode implicar na inserção de defeitos no produto;
- *Defeito*: também conhecido como *falta*, trata-se da materialização do erro em um artefato do software, seja ele um documento de texto ou o código-fonte;
- *Falha*: é a manifestação do defeito que ocorre durante a execução do programa, quando o comportamento operacional do software é diferente do comportamento esperado, considerando as condições de utilização.

O teste de software, portanto, se concentra em revelar falhas. Como consequência, o teste é capaz de prover informações sobre o nível de qualidade do software sob avaliação (POL; TEUNISSEN; VEENENDAAL, 2002). A identificação direta de defeitos, por sua vez, é feita geralmente por meio de técnicas estáticas, tais como inspeções e análise automática de código, aplicadas sobre os documentos do projeto ou no código-fonte do sistema.

Por se tratar de um processo, o teste de software consiste em uma série de atividades (ou sub-processos) coordenadas a fim de transformar insumos em artefatos de teste (ISO/IEC/IEEE, 2013). As atividades que compõem um processo de teste podem variar em cada organização. No entanto, um processo de testes típico costuma incluir as seguintes atividades (ISO/IEC/IEEE, 2013):

- *Planejamento de testes*: atividade caracterizada pela elaboração do Plano de Teste, documento frequentemente utilizado para descrever o escopo do projeto de testes e para definir a estratégia de testes, os recursos alocados e o cronograma (ISO/IEC/IEEE, 2013);
- *Design e implementação de testes*: atividade na qual são derivados os casos e procedimentos de teste a partir dos requisitos do software. Esta atividade requer que os testadores apliquem uma ou mais técnicas de *design* de testes com o objetivo de alcançar os critérios de conclusão estabelecidos, tipicamente descritos em termos de medidas de cobertura. Esta atividade também contempla a implementação dos procedimentos de teste na forma de *scripts* de teste automatizados;
- *Configuração do ambiente de testes*: atividade em que se estabelece o ambiente operacional no qual os testes serão executados, incluindo o software sob teste e ferramental de apoio para sua operação;
- *Execução de testes*: atividade em que os procedimentos de teste definidos na atividade de *design e implementação de testes* são executados no ambiente de testes estabelecido, de forma manual ou automatizada;
- *Comunicação de incidentes*: atividade em que os resultados observados com a execução de testes são analisados e eventuais incidentes de teste são registrados e comunicados às partes interessadas;
- *Conclusão do projeto de testes*: esta atividade marca o encerramento do projeto de testes e consiste em uma série de tarefas, incluindo o registro e comunicação dos resultados do projeto de testes às partes interessadas, a "limpeza" do ambiente de testes e a identificação de lições aprendidas.

Vale destacar que, dependendo do processo e das características do projeto, essas atividades podem compreender as mais variadas tarefas, demandar a elaboração de diferentes artefatos e podem estar organizadas de forma sequencial ou iterativa.

Independentemente de como estão organizadas, em um projeto de desenvolvimento de software, as atividades de teste devem começar o mais cedo possível. Por muitos anos, o paradigma em vigor na indústria de software colocava a fase de testes imediatamente após a atividade de implementação, uma herança do modelo cascata (*waterfall*), proposto em 1970 (ROYCE, 1970), representado na Figura 4. Aquela abordagem poderia trazer uma série de dificuldades ao projeto, como o aumento do seu custo, tendo em vista que os defeitos eram identificados tardiamente, a tempo de se propagarem por todo o produto desenvolvido, tornando mais dispendiosa a sua correção.

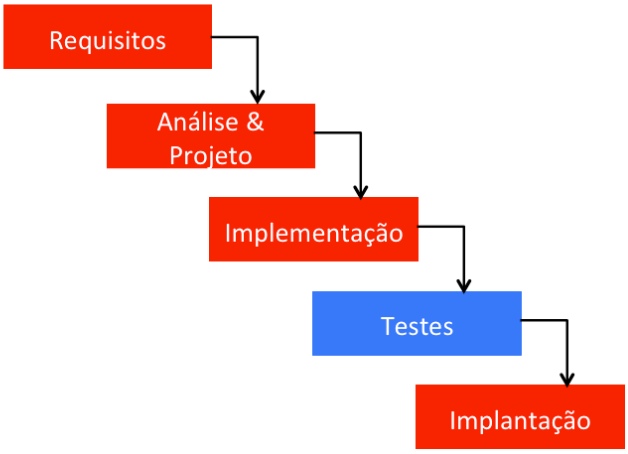


Figura 4: A atividade de Teste no modelo cascata (ROYCE, 1970).

Rook (1986) observou essa limitação do modelo cascata e propôs o chamado Modelo V (ROOK, 1986). Esse modelo de ciclo de vida considera que, paralelamente às atividades de desenvolvimento (construção) do produto, devem ser realizadas atividades de testes. Dessa forma, os defeitos poderiam ser identificados antecipadamente, diminuindo o custo de correção de defeitos e aumentando a confiança no produto desenvolvido.

Como representado na Figura 5, o modelo V apresenta quatro níveis ou fases de teste: i) *teste de unidade*; ii) *teste de integração*; iii) *teste de sistema*; iv) *teste de aceitação*. Cada nível de teste tem objetivos específicos e está associado a uma fase do ciclo de desenvolvimento. De forma geral, à medida que o produto vai sendo desenvolvido, os testes vão sendo planejados nos diferentes níveis. Mais tarde, com o código-fonte implementado, os testes são executados, do nível mais baixo (unidade) ao nível mais alto (aceitação), para confirmar os resultados referentes aos casos de teste planejados em cada nível (FORSBERG; MOOZ, 1991).

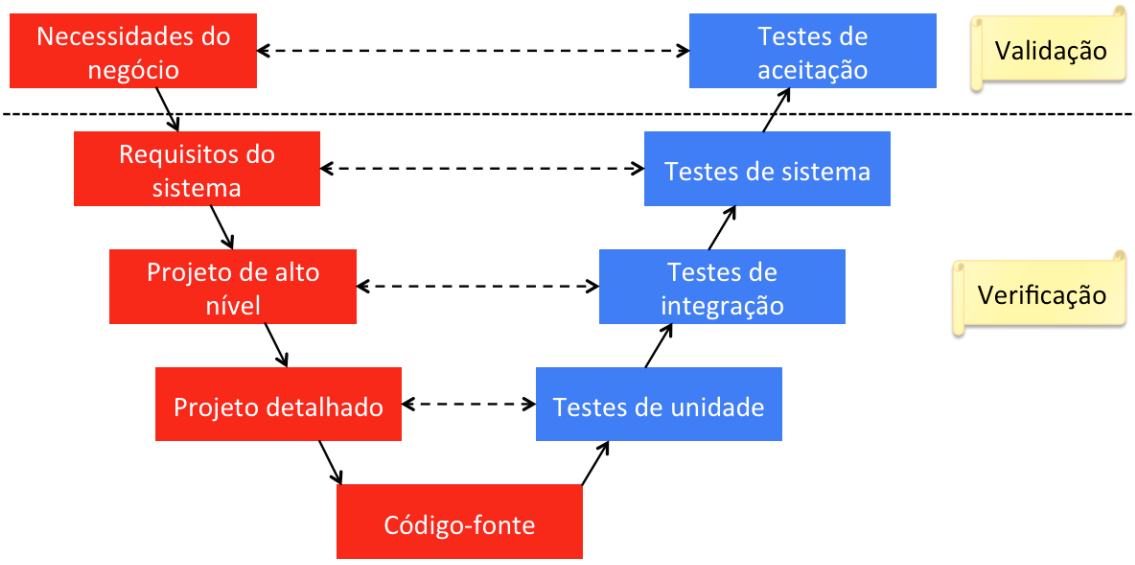


Figura 5: A atividade de Teste no modelo V (ROOK, 1986).

Trata-se de um modelo de ciclo de vida que dá grande ênfase às atividades de Verificação e Validação (V&V). A *verificação* consiste em avaliar se o software foi desenvolvido corretamente, ou seja, se o produto está aderente às suas especificações. Já a *validação* tem como objetivo avaliar se o software correto foi desenvolvido, isto é, se o produto atende às necessidades do cliente (ISO/IEC/IEEE, 2013). A verificação, normalmente, é realizada por membros da equipe de desenvolvimento, em especial, os testadores. Já a validação é de responsabilidade do cliente, sendo geralmente realizada por representantes dos usuários do produto. Os dois tipos de avaliação de produtos de software são complementares e um não substitui o outro. Os incidentes observados por meio da verificação são diferentes daqueles incidentes típicos da validação. Portanto, não se recomenda aplicar somente a verificação ou a validação. Cada uma avalia a qualidade do produto sob uma perspectiva diferente. Como representa a Figura 5, no modelo V, a Verificação é realizada pelos testes de unidade, de integração e de sistema, enquanto que a Validação é realizada por meio dos testes de aceitação (FIRESMITH, 2014).

Como já citado, cada processo de testes pode demandar a elaboração de diferentes artefatos (ou produtos de trabalho). O conjunto de artefatos de teste é conhecido como *testware* (ISO/IEC/IEEE, 2013). Dentre os principais componentes do *testware* estão os casos de teste e os procedimentos de teste, termos comumente confundidos na indústria. Um *caso de teste* é um conjunto de pré-condições, dados de entrada e resultados esperados, selecionados para direcionar o teste de um item

específico do software (ISO/IEC/IEEE, 2013). Os dados de entrada, também conhecidos como *dados de teste*, são criados ou selecionados para satisfazer determinados requisitos de entrada de um ou mais casos de teste. Um caso de teste, porém, não indica a sequência lógica para sua execução. Essa função cabe ao *procedimento de teste*, que descreve os passos necessários para executar um determinado caso de teste e por eventuais pré e pós-condições de execução (ISO/IEC/IEEE, 2013).

Dentre os artefatos típicos de um processo de testes, vale destacar o *Plano de Testes*. Trata-se de uma descrição detalhada dos objetivos, escopo, cronograma e, especialmente, da estratégia de testes do projeto, organizados para coordenar as atividades de teste de algum item de software (ISO/IEC/IEEE, 2013). Uma *estratégia de testes* descreve a abordagem de testes para um projeto específico. Ela deve conter, entre outros itens, a classificação dos testes que serão realizados. O teste de software pode ser classificado em pelo menos três dimensões, que indicam "quando", "o quê", e "como" testar (CRESPO *et al.*, 2004).

A primeira dimensão contempla os *níveis de teste*, que dizem respeito ao estágio da atividade de teste em relação à correspondente atividade de desenvolvimento, de acordo com o modelo V (ROOK, 1986). Além disso, o nível de teste indica o nível de granularidade com item sob teste. *Testes de unidade* são realizados isoladamente sobre as menores unidades independentes do produto. Os *testes de integração* avaliam o comportamento das unidades quando são gradualmente acopladas. Já os *testes de sistema* avaliam o comportamento do software completo, com as unidades já integradas. Finalmente, os *testes de aceitação* têm a finalidade de avaliar se o software está apto a entrar em produção, com base nos critérios de aceitação (ISO/IEC/IEEE, 2013).

A segunda dimensão representa os *tipos de teste*, que se referem às características do software que serão testadas. Assim, o tipo de teste varia de acordo com o tipo de requisito a ser avaliado. Desta forma, requisitos funcionais são avaliados por meio de *testes funcionais*, enquanto que requisitos não-funcionais requerem *testes não-funcionais*. Estes últimos se subdividem em função do tipo de requisito não-funcional avaliado. Por exemplo, para avaliar requisitos de desempenho, como tempo de resposta, consumo de recursos e capacidade do sistema, realizam-se *testes de desempenho*. De modo semelhante, testes de usabilidade são realizados para avaliar requisitos de usabilidade, incluindo facilidade de compreensão e operação do software, eficiência no uso, acessibilidade, entre outros.

Finalmente, a terceira dimensão considera as *técnicas de teste*, cuja classificação é determinada pelo tipo de insumo utilizado para definição dos casos de teste: 1) a *técnica baseada em especificação*, ou caixa preta, em que o teste considera apenas a especificação do elemento a ser testado, ignorando sua estrutura lógica interna e; 2) a *técnica estrutural*, ou caixa branca, onde o teste leva em conta o código-fonte do software a ser testado e tem como objetivo avaliar o comportamento interno do componente de software. A literatura sobre teste de software considera outras técnicas de teste, como o *teste baseado em experiência* e o *teste baseado em defeitos*. Testes baseados em experiência são derivados da habilidade e experiência do testador com aplicativos e tecnologias semelhantes, enquanto que testes baseados em defeitos são realizados de acordo com uma lista de possíveis defeitos e falhas elaborada com base na nos defeitos e falhas identificados em projetos similares ou, até mesmo, na experiência do testador (BLACK, 2011). Porém, essas técnicas podem ser classificadas somente como abordagens ao teste baseado em especificação, diferenciando-se apenas o tipo de insumo usado para derivar os casos de teste.

As três dimensões do teste de software descritas estão representadas na Figura 6:

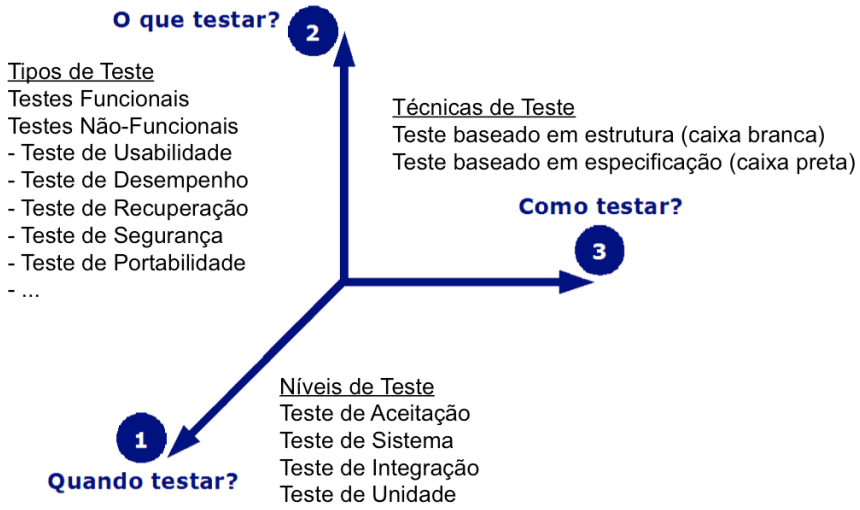


Figura 6: Dimensões do teste de software, adaptado de (CRESPO *et al.*, 2004).

As dimensões representadas na Figura 6 devem compor a estratégia do projeto de testes. Uma estratégia de testes inclui, além das dimensões apresentadas, informações a respeito das práticas de teste a serem utilizadas, critérios para criação ou seleção de dados de teste, critérios de conclusão, ambiente e ferramentas de teste, entre outras (ISO/IEC/IEEE, 2013). A definição da estratégia de teste deve levar em conta as

características de cada projeto, tendo em vista que um único requisito a ser testado pode exigir a aplicação de sub-processos e práticas de teste específicos.

2.2 Estimativa de Esforço em Projetos de Software

Estimativa de esforço é um dos problemas mais antigos e mais importantes para o gerenciamento de projetos de software. Seu objetivo é proporcionar uma aproximação (estimativa) do montante de esforço necessário para executar as atividades do projeto e entregar os produtos ou serviços correspondentes, apoiando a tomada de decisão sobre a alocação de recursos, definição de cronogramas e viabilidade financeira do projeto (TRENDOWICZ; JEFFERY, 2014). É uma área onde tem havido um grande esforço de pesquisa no desenvolvimento e aperfeiçoamento de modelos mais confiáveis e precisos, com o propósito de produzir estimativas mais acuradas.

Esforço é tradicionalmente tratado na indústria como a quantidade de tempo de trabalho totalmente produtivo que uma pessoa precisaria para completar um determinado trabalho (TRENDOWICZ; JEFFERY, 2014). Desta forma, utiliza-se a unidade de medida denominada homem-hora (HH) ou medidas equivalentes do tipo pessoa-tempo, cuja origem não é precisa mas remonta ao final do século XIX, período marcado pelo surgimento da administração científica de Taylor (TAYLOR, 1995), que revolucionou os métodos tradicionais de trabalho através da aplicação do método científico. No entanto, vale observar que as definições de esforço normalmente presentes em dicionários denotam a ideia de trabalho mas não fazem relação com tempo (CAMBRIDGE UNIVERSITY PRESS, 2015) (DICTIONARY.COM, 2015) (MERRIAM-WEBSTER, 2015).

É importante também atentar para dois outros conceitos fundamentais nesta área: *estimativa* e *predição*. Trendowicz e Jeffery (TRENDOWICZ; JEFFERY, 2014) definem *estimativa* como a tentativa de julgar antecipadamente o valor aproximado ou o significado de alguma coisa. Já a *predição* é definida como a declaração antecipada da ocorrência de um fenômeno futuro com base em observação, experiência ou razão científica. Apesar das diferenças apresentadas, esses termos são utilizados de modo intercambiável em Engenharia de Software, tendo em vista que possuem em comum o mesmo objeto de estudo: a incerteza.

Ao longo dos últimos anos, diferentes modelos de estimativa de esforço foram propostos. Exemplos incluem o modelo de Putnam (PUTNAM; MYERS, 2003) e o COCOMO de Boehm (BOEHM, 1981) (BOEHM *et al.*, 2000). Esses e outros modelos, geralmente, têm uma abordagem comum: o esforço é expresso como uma função de uma ou mais variáveis, tais como tamanho do produto, a capacidade produtiva dos desenvolvedores e o nível de reutilização. Além disso, os modelos tradicionais costumam prover estimativas com alto grau de precisão, em valores discretos (exatos) para representar o esforço (por exemplo, *40 homens-dia*).

A *precisão* é a diferença entre a média das medições e o valor de referência de uma estimativa (JCGM, 2012). Ela é determinada pelo tamanho do intervalo de confiança utilizado. Quanto menor for o intervalo de confiança, mais precisa será a estimativa. Já a *acurácia* de uma estimativa é definida pela distância do valor real, independentemente do intervalo de confiança utilizado (JCGM, 2012). Quanto menor for a diferença entre a estimativa e o valor real observado posteriormente, maior terá sido a sua acurácia.

Outra similaridade entre os modelos tradicionais é o fato de focarem na estimativa de esforço de todo o projeto de desenvolvimento do software (BOEHM, 1981) (BOEHM *et al.*, 2000) (PUTNAM; MYERS, 2003), em vez de se concentrar em fases específicas do ciclo de vida. Assim, o esforço de atividades como a de teste de software geralmente está embutido no esforço de desenvolvimento.

2.3 Estimativa de Esforço de Teste de Software

Provavelmente, o primeiro método proposto para estimar especificamente o esforço de teste de software foi o *Test Point Analysis* (TPA), publicado em 1995 (VAN VEENENDAAL, 1995). O TPA é direcionado a projetos que envolvam testes funcionais (caixa preta) em nível de sistema ou de aceitação. Para tal, o método leva em conta a medida do tamanho funcional do sistema (em pontos por função), características do ambiente de teste (incluindo a estratégia de teste a ser utilizada) e o nível de produtividade da equipe envolvida. Inicialmente, o tamanho funcional do sistema é utilizado para determinar o volume de trabalho de teste em uma unidade denominada *pontos de teste*. Em seguida, o método utiliza equações para estimar a quantidade de horas de teste necessárias, combinando a quantidade de pontos de teste com fatores

ponderados relacionados à produtividade da equipe e ao ambiente de teste. Todavia, é importante observar que o TPA possui um alto nível de subjetividade, caracterizado pelas classificações utilizadas para ponderar o grau de influência de cada fator, o que pode afetar negativamente a acurácia das estimativas. Fatores como "importância do usuário" e "interface", usados para calcular a quantidade de pontos de teste, utilizam termos imprecisos como "baixa", "normal" e "alta" para classificar as funcionalidades que serão testadas. Além disso, o método TPA carece de evidências experimentais. A principais publicações do método (VAN VEENENDAAL, 1995) (POL; TEUNISSEN; VEENENDAAL, 2002) apenas o descrevem, sem exemplificar ou demonstrar seu uso em qualquer contexto.

Em 2001, Nageswaran (NAGESWARAN, 2001) propôs uma abordagem para estimar o esforço de teste com base em especificações de casos de uso, possibilitando a realização de estimativas no início de cada projeto. O modelo proposto é direcionado a testes funcionais em nível de sistema, cobrindo as atividades de planejamento, *design* e execução de testes. A estimativa de esforço é obtida através de uma equação que leva em conta fatores ponderados relacionados à quantidade de cenários (fluxos) dos casos de uso e à complexidade dos atores, além de aspectos técnicos relacionados ao ambiente de teste. Nageswaran (NAGESWARAN, 2001) cita que o modelo foi avaliado através de um estudo de caso no âmbito de uma grande companhia norte-americana de software, alcançando um erro relativo médio (*Mean Magnitude of Relative Error* – MMRE) de 5,9%. Todavia, o trabalho não descreve claramente como o modelo foi concebido e qual o contexto do estudo, o que introduz uma série de ameaças à validade de constructo e externa.

Mais recentemente, Almeida, Abreu e Moraes (ALMEIDA; ABREU; MORAES, 2009) propuseram três modelos de estimativa distintos, todos baseados no modelo de Nageswaran (NAGESWARAN, 2001). Cada modelo proposto possui modificações pontuais em relação ao modelo original. O primeiro modelo proposto, denominado *N-Weighted*, modifica a regra de ponderação dos casos de uso em função da quantidade de cenários. A principal diferença é o tratamento desigual dado aos cenários normais e de exceção dos casos de uso. A decisão de tratar cenários normais distintamente dos cenários de exceção foi tomada com base na observação de que um cenário normal requer mais tempo para testar do que um cenário de exceção. Isto é devido ao fato de o cenário de exceção estar contido em um cenário normal, uma vez que os pontos de exceção nas descrições de casos de uso são normalmente associados

com passos de um cenário normal. O segundo modelo, chamado de *New, Search, Modify*/*Management Information System functionality* (NSM/MIS) provê apenas duas opções para classificar os casos de uso: simples e complexo. Casos de uso que realizam apenas operações de cadastro, consulta, atualização e exclusão (CRUD) são classificados como simples e, consequentemente, recebem peso 1. Por outro lado, aos casos de uso não-CRUD é atribuído o peso 2. Vale destacar que essa classificação não leva em conta o tipo e a quantidade de cenários presentes em cada caso de uso, como faz o método *N-Weighted.* Por fim, o terceiro modelo proposto, intitulado de *Simplified Method*, é uma adaptação do modelo NSM/MIS, com a diferença de que os atores são classificados em apenas duas categorias, em vez de três como nos três primeiros modelos. O estudo ainda compara o modelo de Nageswaran (NAGESWARAN, 2001) com os modelos propostos através de um estudo de caso envolvendo cinco sistemas distintos relacionados ao mesmo domínio de aplicação. Os resultados indicam que o método NSM/MIS foi o mais eficiente, apresentando estimativas com margem de erro de até 5% em dois dos cinco projetos analisados. Porém, de forma geral, os quatro modelos analisados apresentaram baixos níveis de acurácia, incluindo estimativas com diferenças de mais de 100% em relação ao esforço real.

Zhu, Zhou e Chen (ZHU; ZHOU; CHEN, 2009) propõem e avaliam diferentes métricas relacionadas às estimativas de tamanho e de esforço em projetos incrementais de software, dentre as quais uma métrica para estimar o esforço em projetos de testes funcionais em nível de sistema. A estimativa de esforço obtida com essa métrica compreende as atividades de análise & *design*, execução e suporte a testes. Para isso, a equação utilizada leva em conta o esforço aplicado nas atividades de desenvolvimento anteriores, além de fatores relacionados ao tipo de software a ser testado (sistema *batch* ou baseado em interface gráfica) e à produtividade e experiência da equipe de teste. Um estudo realizado ao longo de cinco anos, envolvendo 20 *releases* de um projeto real, apontou que 83,3% das estimativas apresentaram diferença de no máximo 25% em relação ao esforço real. Entretanto, apesar da quantidade de *releases* analisadas, o estudo foi realizado sobre um só projeto, o que representa uma ameaça à validade externa.

Outros trabalhos se restringem a estimar apenas o esforço de execução de testes (ARANHA; BORBA, 2007) (ARANHA; BORBA, 2009) (XIAOCHUN *et al.*, 2008) (SILVA; ABREU; JINO, 2009). Aranha e Borba (ARANHA; BORBA, 2007) (ARANHA; BORBA, 2009) apresentam um modelo híbrido que combina julgamento

do especialista com dados históricos de projetos. Esse modelo é indicado para projetos que envolvam testes funcionais e não-funcionais em nível de sistema. As estimativas são realizadas a partir da contagem de passos que compõem os procedimentos de teste. Em seguida, para cada passo de cada procedimento de teste, deve-se identificar características funcionais ou não-funcionais e determinar sua complexidade de execução. Por fim, deve-se aplicar uma fórmula que contempla, além da quantidade de passos e a complexidade de execução, preditores de esforço obtidos através de análise de regressão sobre dados históricos. Esse modelo, no entanto, se mostra útil apenas em fases mais adiantadas dos projetos, quando os casos e procedimentos de teste já estão documentados. Além disso, a necessidade de uma base de dados históricos pode dificultar a sua utilização, especialmente por equipes de teste iniciantes ou que não tenham armazenado dados de projetos anteriores.

Tal qual o trabalho de Aranha e Borba (ARANHA; BORBA, 2007) (ARANHA; BORBA, 2009), o modelo proposto por Xiaochun *et al*. (XIAOCHUN *et al.*, 2008), denominado *Test Suite Execution Vector*, se concentra em estimar o esforço de execução de testes. A grande diferença está na possibilidade de produzir estimativas já nas fases iniciais do projeto, antes mesmo de se especificar os casos e procedimentos de teste. Isto é possível porque o modelo realiza inicialmente a estimativa da quantidade de casos de testes para o projeto, através de uma equação que considera características de cada caso de uso, como a quantidade de passos e de entidades relacionadas. Em seguida, a estimativa da quantidade de casos de teste é associada a fatores relacionados à complexidade de execução e à experiência do testador designado. A utilização desse modelo, semelhantemente ao observado no modelo de Aranha e Borba (ARANHA; BORBA, 2007) (ARANHA; BORBA, 2009), depende da existência de uma base de dados históricos de projetos, sobre a qual devem ser aplicados algoritmos de aprendizagem de máquina (*machine learning*) para realizar regressão linear e, consequentemente, estimar o esforço.

Silva, Abreu e Jino (SILVA; ABREU; JINO, 2009), por sua vez, propuseram um modelo algorítmico de estimativa de esforço de execução manual de testes funcionais em nível de sistema. O modelo em si é composto por diversas equações, que tomam como entrada o número de casos de teste e o número de passos de cada caso de teste (*sic*) – na realidade o número de passos de cada "procedimento de teste". A estimativa de esforço é obtida a partir da medida de *eficiência acumulada*, que é o somatório de todos os passos dos procedimentos de teste executados por uma equipe até um dia *i*,

dividido pela soma do tempo de execução correspondente. O esforço estimado é, portanto, representado como o tempo necessário para execução dos procedimentos de teste. Os resultados do estudo indicam que 13 dos 19 ciclos de execução de teste realizados apresentaram erros de estimativa abaixo de 13%. Entretanto, vale ressaltar que o modelo é limitado a estimar somente o tempo de execução de testes manuais, o que representa apenas um subconjunto do esforço total empregado em um projeto de teste.

Trabalhos que tratem exclusivamente de fatores que afetam o esforço relacionado ao teste de software não são comuns. Geralmente, os trabalhos relacionados apresentam modelos de estimativa de esforço compostos por vários fatores, que costumam estar tão insuficientemente descritos que avaliar sua aplicabilidade pode ser uma tarefa tão dispendiosa quanto infrutífera. Ainda assim, vale destacar os trabalhos de Jones (JONES, 2007) e Black (BLACK, 2011), tendo em vista a influência que aparentemente exercem na indústria de software.

Jones (JONES, 2007) lista 10 fatores que exerceriam influência sobre os custos, o cronograma e a eficiência dos testes. Além de não indicar explicitamente como esses fatores poderiam afetar o esforço, o autor não descreve como tais fatores foram derivados e como podem ser medidos ou observados. Jones (JONES, 2007) ainda sugere que a estimativa de esforço de teste (expressa inicialmente em número de casos de teste) seja obtida a partir de uma relação com o tamanho funcional do software (em pontos por função), e convertida em pessoas-hora através de um fator de conversão obtido de dados históricos dos projetos. Contudo, o modelo de Jones (JONES, 2007) leva consigo a incerteza inerente à contagem de pontos por função, além de não distinguir os casos de teste estimados em função do nível, tipo e técnica de teste.

Black (BLACK, 2011), por sua vez, apresenta 39 fatores que afetariam o esforço de teste, divididos em quatro categorias: processo, recursos, pessoas e atrasos. Na categoria "processo" figuram fatores como "quantidade de atividades de teste que permeiam o projeto" e "modelo de ciclo de vida do processo". Na categoria "recursos" há fatores como "existência de processo e ferramentas de automação de testes" e "disponibilidade de um ambiente de testes dedicado e seguro". A categoria "pessoas" possui fatores como "habilidades, experiência e atitudes apropriadas ao projeto" e "estabilidade da equipe do projeto". Já a categoria "atrasos" é representada por fatores tais como "quantidade de *stakeholders*" e "existência de equipes geograficamente distribuídas". Apesar de boa parte dos fatores apresentados parecerem inquestionáveis,

o autor, da mesma forma que Jones (JONES, 2007), não apresenta evidências experimentais ou qualquer análise racional de como os fatores foram derivados. Não há, ainda, qualquer indicação sobre como os fatores podem ser medidos ou observados, o que aumenta o nível de incerteza sobre a utilidade dos fatores apresentados.

2.4 Considerações Finais

Este capítulo apresentou de forma resumida o referencial teórico relacionado a este trabalho. Inicialmente, foram apresentados conceitos e definições sobre Teste de Software e Estimativa de Esforço, com o objetivo de estabelecer a terminologia utilizada nos próximos capítulos. Em seguida, foram descritos diversos trabalhos sobre Estimativa de Esforço em Teste de Software, visando retratar o estado-da-arte desse campo de estudo.

É possível perceber que os trabalhos relacionados deixam muitas lacunas abertas, o que reforça a necessidade de se explorar o tema em profundidade. Desta forma, para caracterizar mais precisamente essas lacunas e tentar identificar outras limitações, o capítulo 3 apresenta uma Revisão Sistemática da Literatura, que debruçou-se sobre diversos modelos de estimativa de esforço de teste.

3 REVISÃO SISTEMÁTICA DA LITERATURA[2]

Este capítulo apresenta uma Revisão Sistemática da Literatura (RSL) a respeito de estimativa de esforço em teste de software e, para tal, está dividido em cinco seções. A seção 3.1 caracteriza o cenário de investigação. A seção 3.2 descreve o protocolo utilizado nesta revisão. A seção 3.3 apresenta os principais resultados encontrados. A seção 3.4 discute os resultados, ressaltando as lacunas deixadas pelos trabalhos analisados. A seção 3.5 discute sobre as ameaças à validade deste estudo. Finalmente, a seção 3.6 apresenta as considerações finais deste capítulo.

3.1 Cenário de Investigação

Tradicionalmente, o esforço inerente ao teste de software tem sido observado através de modelos que tentam capturar o esforço total de desenvolvimento de software. Provavelmente, essa característica ocorra por influência do modelo de ciclo de vida em cascata, no qual o teste é visto apenas como uma etapa do processo de desenvolvimento de software e não como um processo independente com características próprias. Modelos de estimativa de esforço clássicos como COCOMO 81 (BOEHM, 1981), COCOMO II (BOEHM *et al.*, 2000) e o modelo de Putnam (PUTNAM; MYERS, 2003) ratificam essa visão, tendo em vista que observam o processo de desenvolvimento como um todo, sem distinguir o esforço em função de cada atividade do processo. Para tal, esses modelos tomam como insumo alguma especificação do software para fornecer uma noção de todo o esforço a ser despendido com o desenvolvimento do software.

No entanto, a atividade de teste de software possui particularidades dificilmente levadas em conta pelos modelos tradicionais de estimativa de esforço. Os fatores que influenciam o esforço de teste de software são claramente diferentes daqueles observados em outras atividades de desenvolvimento de software. Portanto, não é apropriado usar diretamente modelos de estimativa de esforço de desenvolvimento de software para estimar o esforço de teste de software. Provavelmente, modelos de

[2] Uma versão preliminar deste capítulo foi publicada no artigo *Estimativa de Esforço em Teste de Software: Modelos, Fatores e Incertezas*, apresentado no XX *Congreso Argentino de Ciencias de la Computación* (CACIC 2014) (SILVA-DE-SOUZA; RIBEIRO; TRAVASSOS, 2014). Em agosto de 2018, uma versão completa e atualizada deste estudo foi submetida ao *Journal of Systems and Software*, com o título *Effort Factors in Software Testing: Findings of a Secondary Study*.

estimativa específicos por atividade do processo irão proporcionar estimativas de esforço com maior nível de acurácia.

Desta forma, torna-se relevante a construção de modelos que apoiem a estimativa de esforço de teste de software, permitindo que essa seja realizada com base em fatores explícitos, e não apenas com base em conhecimento tácito de especialistas. A construção de tais modelos exige, portanto, a identificação das variáveis de contexto que influenciam a quantidade de esforço necessário para as atividades de teste de software.

Portanto, essa RSL objetiva identificar os fatores que influenciam o esforço de teste de software, bem como verificar se esses fatores estão sendo utilizados em modelos de estimativa e quais são as características desses eventuais modelos. De acordo com o paradigma *Goal-Question-Metric* (GQM) (BASILI; CALDIERA; ROMBACH, 1994), esse objetivo pode ser representado da seguinte forma:

Tabela 1: Objetivos da RSL.

Analisar	estimativa de esforço em Teste de Software
Com o propósito de	caracterizar
Em relação à	fatores que influenciam o esforço em Teste de Software
Do ponto de vista	pesquisadores em Engenharia de Software
No contexto de	modelos de estimativa de esforço de Teste de Software descritos na literatura técnica

Considerando o objetivo apresentado, dois alunos de doutorado (incluindo o autor desta tese), supervisionados pelo seu professor-orientador, definiram um protocolo de revisão sistemática da literatura com o objetivo de identificar e selecionar estudos empíricos sobre estimativa de esforço em testes de software. A primeira bateria de execução foi inicialmente realizada em 2015, cobrindo artigos publicados até 2014. No início de 2017, uma segunda bateria atualizou os resultados do estudo, cobrindo o período de 2015 a 2016. Finalmente, em meados de 2018, uma terceira bateria foi executada, cobrindo o período de 2017 a 2018. Assim, este estudo identificou, em 37 estudos selecionados, 47 fatores que influenciam o esforço de teste de software, organizados em cinco categorias diferentes. Apesar da falta de informações de contexto sobre a maioria dos estudos analisados, os fatores de esforço identificados podem orientar a construção de modelos de estimativa de esforço de teste com maiores níveis de acurácia, considerando que alguns modelos de estimativa existentes não tinham conhecimento dos fatores de esforço revelados neste estudo.

3.2 Protocolo do Estudo

Esta seção apresenta o protocolo do estudo. Este protocolo de revisão sistemática segue as diretrizes descritas por Biolchini *et al*. (BIOLCHINI *et al.*, 2005) e, portanto, apresenta a revisão de modo completo, possibilitando a outros pesquisadores auditá-la ou reproduzi-la.

3.2.1 Questões de pesquisa

Os principais objetivos deste estudo são compreender o que é o esforço de teste e identificar fatores que podem influenciar o esforço das atividades de teste de software (planejamento, *design* e implementação, execução, comunicação de incidentes, etc.). Além disso, pretende-se caracterizar os modelos que fazem uso desses fatores. As questões de pesquisa, definidas em função do cenário apresentado na seção 3.1, são as seguintes:

- **QE1**: O que se entende por **esforço** em teste de software e como ele pode ser medido?
- **QE2**: Quais **fatores** influenciam o esforço em teste de software?
    - **QE2.1**: Que **modelos de estimativa de esforço** utilizam esses fatores?
        - **QE2.1.1**: Esses modelos são baseados em que tipos de técnicas?
        - **QE2.1.2**: Esses modelos são adequados para qual contexto de teste?
        - **QE2.1.3**: Como esses modelos têm sido empiricamente avaliados?

3.2.2 Estratégia de busca

Uma *string* de busca, representada na Figura 7, foi definida com base na abordagem PICO (PAI *et al.*, 2004). Nessa abordagem, a string de busca é estruturada de acordo com as seguintes dimensões: **p**opulação de interesse, **i**ntervenção avaliada, **c**omparação da intervenção (se houver) e ***o****utcome* (resultado). Este estudo, em particular, não possui qualquer elemento ou *baseline* que permita uma comparação com

a intervenção. Desta forma, a dimensão "comparação" é representada como um conjunto vazio. Portanto, este estudo pode ser classificado como uma *quasi*-Revisão Sistemática da Literatura (TRAVASSOS *et al.*, 2008).

(("software testing" OR "software test" OR "testing of systems" OR "system test" OR "system testing" OR "software verification" OR "software validation" OR "verification & validation" OR "software evaluation" OR "software quality" OR "test activities" OR "testing activities" OR "test activity" OR "testing activity" OR "test planning" OR "test design" OR "test automation" OR "automated test" OR "test execution" OR "manual test" OR "test case" OR "test specification" OR "testing process" OR "testing project" OR "test project") **AND** ("software test estimation" OR "effort estimation" OR "cost estimation" OR "estimate the cost" OR "effort prediction" OR "cost prediction" OR "effort measurement" OR "cost measurement" OR "execution effort" OR "time estimation") **AND** ("factor" OR "variable" OR "predictor" OR "indicator" OR "driver" OR "measure" OR "model" OR "approach" OR "method" OR "process" OR "technique" OR "metric"))

Figura 7: String de busca utilizada.

As palavras-chave que compõem essa *string de busca* foram identificadas a partir do conhecimento prévio dos pesquisadores e derivadas de sete **artigos de controle** identificados através de uma busca *ad hoc* (vide **Apêndice A**). Tais artigos foram considerados de controle porque apresentam respostas para as questões de pesquisa apresentadas e serviram para calibrar a *string* de busca. A identificação dos artigos relacionados foi realizada através da execução da *string* de busca nas máquinas de busca Scopus, IEEEXplore, Engineering Village e Web of Science. A escolha dessas máquinas de busca foi motivada pela cobertura que elas oferecem. Por exemplo, a Scopus indexa artigos de diferentes fontes, incluindo ACM e IEEE. Além disso, tais máquinas de busca são conhecidas pela estabilidade e interoperabilidade com diferentes sistemas de referência. A busca ainda foi complementada pela técnica de *snowballing* (*forward* e *backward* em um nível) (JALALI; WOHLIN, 2012) através do Google Scholar.

3.2.3 Critérios de seleção de artigos

Após a execução das buscas, os pesquisadores realizaram a leitura do título e do *abstract* dos artigos retornados e aplicaram os critérios de inclusão (I1 e I2 e I3 e I4 e (I5 ou I6) e I7) e critérios de exclusão (E1 ou E2), representados na Tabela 2. Inicialmente, dois pesquisadores classificaram os trabalhos em "Incluído", "Excluído" ou "Dúvida". Em seguida, um terceiro pesquisador analisou os casos de desacordo e reclassificou os artigos.

Tabela 2: Critérios de seleção dos artigos.

Critérios de Inclusão	Critérios de Exclusão
(I1) Estar disponível na Web	(E1) Apresentar apenas exemplos ou raciocínio especulativo sobre a viabilidade do modelo proposto
(I2) Estar escrito em Inglês	(E2) Utilizar bases públicas de projetos, tais como Promise e ISBSG
(I3) Tratar de Teste de Software	
(I4) Abordar estimativa de esforço	
(I5) Apresentar fatores que exercem influência sobre o esforço do Teste de Software	
(I6) Apresentar ou avaliar modelos para estimar o esforço no Teste de Software	
(I7) Apresentar qualquer avaliação experimental ou prova de conceito	

A maior parte dos critérios de inclusão e exclusão apresentados são auto-explicativos. No entanto, os critérios I7 e E2 merecem uma explicação adicional por não serem tão óbvios.

Quanto ao critério I7, este limita os trabalhos selecionados àqueles que apresentem algum tipo de avaliação experimental, incluindo *survey*, simulação, estudo de caso, *quasi*-experimento ou experimento. Esta limitação serve para filtrar e excluir aqueles trabalhos com baixa qualidade metodológica, que podem trazer resultados meramente especulativos. Esse critério, porém, poderia ser demasiado excludente, limitando as observações a um conjunto pequeno de trabalhos. Desta forma, o critério I7 também incluiu provas de conceito, assumindo-se o risco de selecionar trabalhos com alguma carga especulativa.

O critério E2 se justifica pelas limitações no uso de bases de projetos multiorganização, em especial as seguintes (FERNÁNDEZ-DIEGO; GONZÁLEZ-LADRÓN-DE-GUEVARA, 2014):

- a falta de confiabilidade no conteúdo dessas bases, motivada pela ausência de informações de contexto sobre os dados;
- a heterogeneidade típica de bases de dados multiorganizacionais, cujos efeitos são potencializados quando se tentam produzir modelos para organizações que não contribuíram com tais bases.

Além dessas limitações, derivar modelos a partir de bases de dados muito grandes e com dados redundantes ou irrelevantes é mais difícil que derivar um modelo a partir de uma base de dados homogênea e limitada a um conjunto de atributos altamente preditivos.

3.2.4 Resultados da busca e seleção

A Tabela 3 apresenta os resultados da busca e da seleção de artigos considerando os critérios de seleção da Tabela 2. A aplicação da técnica de *snowballing* resultou na inclusão de cinco artigos que estavam fora das bases cobertas pelas máquinas de busca citadas anteriormente. A lista de **artigos incluídos** está disponível no **Apêndice A**. A extração das informações dos artigos foi apoiada por um **formulário de extração** (disponível no **Apêndice B**), definido com base nas questões de pesquisa apresentadas. Os artigos foram, então, distribuídos aleatoriamente para extração entre o autor desta tese e o estudante de doutorado que colaborou nesta revisão sistemática. Posteriormente, cada formulário de extração foi revisado em conjunto com o professor-orientador a fim de garantir a consistência da forma de extração e de apresentação dos dados.

Tabela 3: Resultados da busca e seleção.

Status dos Artigos	Quantidade
Retornados pelas bases (com duplicatas)	1.557
Retornados pelas bases (sem duplicatas)	735
Excluídos	711
Incluídos	24
Controle presentes nas bases	6
Controle fora das bases	1
Incluídos por *snowballing*	6
Total de artigos incluídos	**37**

Pode-se notar na Tabela 3 que 30 dos 37 artigos incluídos estavam presentes nas bases cobertas pelas máquinas de busca (seis artigos de controle e 24 artigos incluídos após aplicação dos critérios de seleção). Além disso, seis outros artigos foram incluídos via *snowballing* (todos eles por *forward snowballing*). Com o objetivo de avaliar a cobertura das máquinas de busca utilizadas, foram contabilizadas as ocorrências de cada artigo incluído por máquina de busca. Sendo assim, a Figura 8 representa, através de um diagrama de Venn, a distribuição dos artigos incluídos por máquina de busca. A Figura 8 mostra que a *Scopus* apresentou melhor cobertura que as demais máquinas de busca (25 artigos). Ainda pode-se observar que *Web of Science* (WoS) e *Engineering Village* (EV) não retornaram nenhum artigo que não havia sido retornado pela *Scopus* ou *IEEEXplore*, o que sugere que sua utilização nesta pesquisa foi desnecessária.

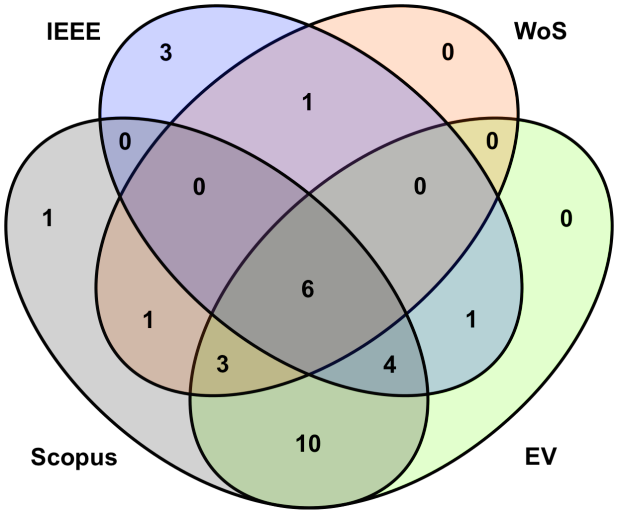


Figura 8: Distribuição dos artigos incluídos por máquina de busca.

3.2.5 Avaliação da qualidade dos artigos

A avaliação da qualidade dos artigos incluídos foi composta por 12 questões nomeadas de QA1 a QA12, as quais são descritas a seguir. Essa avaliação foi realizada com um triplo objetivo. Primeiro, de QA1 a QA4, pretendia-se saber quais questões de pesquisa eram respondidas por cada artigo. Por exemplo, a questão QA1 está relacionada com a questão de pesquisa QE1 e avalia se o artigo apresenta alguma definição de esforço. Depois, a questão QA5 avalia se o trabalho especifica as dimensões do teste de software cobertas pela abordagem/técnica proposta (nível, tipo, técnica, forma de execução e atividade/fase de teste). Por fim, os demais itens avaliam a maneira como as abordagens propostas pelos artigos foram avaliadas. Portanto, os artigos foram pontuados de acordo com a completude de que responderam as questões de pesquisa, a cobertura das abordagens propostas quanto às dimensões do teste e de acordo com o nível de controle do estudo realizado para avaliar a abordagem/técnica proposta. Os resultados da avaliação da qualidade são apresentados na seção 3.3.4.

- QA1: O trabalho apresenta uma definição clara do que é esforço em teste de software? (1 ponto)
- QA2: O documento descreve claramente os fatores que influenciam o esforço de teste de software? (2 pontos)
    - Apresenta fatores de influência com uma descrição de como eles podem ser medidos ou observados e indica o valor ou a direção dessa influência (por exemplo, positivo, negativo). (2 pontos)

    - Apresenta fatores de influência e indica o valor ou a direção dessa influência (por exemplo, positivo, negativo). (1 ponto)
    - Apresenta fatores de influência com uma descrição de como eles podem ser medidos ou observados. (1 ponto)
    - Apenas apresenta fatores de influência. (0,5 ponto)
    - Não apresenta nenhum fator de influência. (0 ponto)
- QA3: O trabalho apresenta um modelo para estimativa de esforço de teste de software? (2 pontos)
    - Existe um modelo de estimativa e há uma descrição de como esse modelo foi definido. (2 pontos)
    - Existe um modelo de estimativa, mas não há uma descrição de como esse modelo foi definido. (1 ponto)
    - Não existe um modelo de estimativa. (0 ponto)
- QA4: O trabalho descreve o domínio de aplicação ou o tipo de sistema para o qual a abordagem/técnica de estimativa foi proposta? (1 ponto)
- QA5: O trabalho descreve o nível, tipo, técnica, forma de execução e atividade/fase de teste coberta pela abordagem/técnica proposta? (2 pontos)
    - Pelo menos quatro deles. (2 pontos)
    - Apenas três deles. (1,5 ponto)
    - Apenas dois deles. (1 ponto)
    - Apenas um deles. (0,5 ponto)
    - Nenhum deles. (0 ponto)
- QA6: O trabalho explicita as condições e restrições para o uso da abordagem/técnica proposta? (1 ponto)
- QA7: O documento fornece alguma indicação da acurácia da abordagem/técnica proposta? (1 ponto)
- QA8: Que tipo de estudo foi realizado para avaliar a abordagem/técnica proposta? (2 pontos)
    - Experimento (2 pontos)
    - *quasi*-Experimento (1,5 ponto)
    - Estudo de caso (1 ponto)
    - Survey (0,5 ponto)
    - Simulação (0 ponto)

    - Prova de conceito (0 ponto)
- QA9: O estudo é descrito em um nível apropriado de detalhes que permite compreender como os resultados foram obtidos e identificar suas limitações? (1 ponto)
- QA10: Houve algum tratamento estatístico nos dados utilizados no estudo para garantir sua consistência? (1 ponto)
- QA11: Que tipo de participante (*subject*) foi usado no estudo? (1 ponto)
    - Profissional (1 ponto)
    - Projeto da indústria (1 ponto)
    - Estudante (0,5 ponto)
    - Projeto fictício (0 ponto)
- QA12: O estudo descreve adequadamente as ameaças à validade? (1 ponto)
    - Ameaças à validade foram identificadas e tratadas. (1 ponto)
    - Ameaças à validade só foram identificadas. (0,5 ponto)
    - Ameaças à validade não foram identificadas. (0 ponto)

3.3 Resultados

Esta seção apresenta os resultados obtidos através da revisão sistemática, organizados pelas questões de pesquisa apresentadas no protocolo de estudo.

3.3.1 QE1: O que se entende por esforço em teste de software e como ele pode ser medido?

A palavra "esforço" é definida pelo dicionário de *Cambridge* como "a atividade física ou mental necessária para alcançar algum objetivo" (CAMBRIDGE UNIVERSITY PRESS, 2015). Entretanto, no contexto deste estudo se fez necessário compreender o entendimento de esforço entre os pesquisadores de modelos de estimativa de esforço em teste de software. Desta forma, procurou-se identificar nos artigos analisados qualquer definição sobre esforço em teste de software, bem como quais unidades de medida são utilizadas para representá-lo.

Para a maioria dos 37 trabalhos analisados, o conceito de esforço é definido de forma tácita. Somente cinco artigos apresentam alguma definição explícita do que é

esforço em teste de software. Mizuno *et al*. [R10] considera o esforço de teste como "*a soma do esforço gasto na atividade de teste e o esforço gasto na atividade de depuração*". Similarmente a Mizuno *et al*. [R10], Kumari e Sharma [R24] afirmam que "*o esforço de teste é a adição do esforço gasto na atividade de teste e o esforço gasto na atividade de depuração*". Marchetto [R14] define o esforço de teste como "*o esforço necessário para testar um componente ou um conjunto de componentes agrupados com base em suas funcionalidades*". Shaheen e Du Bousquet [R15] definem o esforço de teste como custo do teste. Para Cotroneo *et al*. [R30], o esforço de teste é a quantidade de tempo e de recursos gastos em atividades de teste para atingir uma determinada meta de teste. As duas primeiras definições incluem depuração como parte do esforço de teste (consideramos a depuração como uma tarefa da atividade de implementação). A terceira definição é recursiva (usa a palavra "esforço" para definir esforço). A quarta definição confunde esforço e custo. Finalmente, a quinta definição usa o tempo e os recursos como *surrogates* de esforço.

Devido à falta de artigos que conceituem clara e consistentemente o esforço de teste, assumiu-se neste estudo que o **esforço de teste** é a atividade física ou mental necessária para executar as atividades de teste em um determinado projeto de software. Esta definição tomou como base a definição de "esforço" apresentada pelo dicionário de *Cambridge*.

Já em relação à unidade de medida de esforço, é possível notar na Tabela 4 que as unidades de medida do tipo pessoa-tempo (e.g. homem-hora e pessoa-mês) são as mais frequentes para mensurar esforço (há trabalhos que indicam mais de uma medida).

Tabela 4: Tipos de unidades de medida de esforço.

Unidade de Medida de Esforço	Ocorrências
<Pessoa>-<tempo>	25
Tempo	10
Tamanho	2
% Esforço de desenvolvimento	1
Não especificada	5

3.3.1.1 Esforço x Custo

Esta RSL evidenciou não apenas a ausência de uma definição precisa sobre "esforço" e de uma unidade de medida que o represente de forma consistente, mas também uma certa confusão conceitual com o termo "custo" observada na maioria dos artigos

analisados. Em virtude desses problemas, foi realizado um estudo adicional que aplicou princípios do campo da Linguística em busca de uma definição consistente dos termos "esforço" e "custo" na área de Engenharia de Software, o qual está descrito no Apêndice D.

No entanto, os resultados reforçam a impressão que "esforço" e "custo" são utilizados de forma inconsistente na literatura técnica de ES. Em parte dos trabalhos, há relações de dependência claras entre esses termos, seja indicando equivalência, seja indicando que um é parte do outro. Porém, em alguns trabalhos eles se apresentam totalmente distintos.

Apesar desse estudo não ter resolvido o problema da confusão conceitual entre "esforço" e "custo" em ES, ele serviu para evidenciar um problema terminológico que pode representar uma ameaça à validade dos modelos de estimativa de esforço e custo propostos em ES.

3.3.2 QE2: Que fatores influenciam o esforço em teste de software?

A correta definição dos fatores que influenciam o esforço de teste de software em cada tipo de projeto pode apoiar a construção de modelos de estimativa mais realistas. Sendo assim, procurou-se identificar nos artigos incluídos evidências sobre os fatores utilizados pelos modelos para estimar o esforço em teste de software. Sendo assim, procedeu-se a realização da etapa de *open coding*, conforme definida no método *Grounded Theory* (STRAUSS; CORBIN, 1998).

Inicialmente, procurou-se derivar as definições e realizar agrupamentos com base no nome e no contexto onde cada fator foi identificado. Todavia, nem todos os trabalhos apresentavam definições explícitas sobre cada fator ou informações de contexto que pudessem direcionar a conceituação e a classificação. Além disso, parte dos trabalhos apresentava explicitamente os fatores utilizados, enquanto outros realizavam definições implícitas através das equações dos modelos. Desta forma, os autores deste estudo precisaram entrar em consenso sobre o significado de diversos fatores. Ainda assim, outros fatores ficaram sem definição, dada a falta de informação a respeito. Para preencher essa lacuna e proporcionar definições mais precisas foi realizado um estudo de observação, descrito no capítulo 4, a partir do qual os fatores foram finalmente definidos.

O estudo revelou 47 fatores de esforço distintos, os quais foram organizados em cinco diferentes categorias: **1) software sob teste** (ou *system under testing* – SUT); **2) equipe de teste**; **3) projeto de teste**; **4) ambiente de teste**; e **5) *testware*** (artefatos de teste produzidos). Tais categorias emergiram das classificações apresentadas nos artigos e da experiência do autor.

Vale ressaltar que apesar de um mesmo modelo de estimativa de esforço poder apresentar mais de uma característica relacionada ao mesmo fator de influência, cada fator foi contabilizado apenas uma vez. Por exemplo, o trabalho de Ashraf e Janjua [R34] cita duas variáveis relacionadas às tarefas do processo de teste, "tarefas de teste requeridas" e "atividades de reporte de testes", as quais foram contabilizadas apenas uma vez através do fator "tarefas do processo de teste". Isso se deve às diferenças de níveis de abstração e granularidade presentes nos trabalhos analisados.

Pode-se observar na Tabela 5 que a categoria SUT apresenta mais tipos de fatores de influência (21), dentre os quais os fatores mais frequentes são "tamanho da especificação de requisitos" e "nível de desempenho requerido". Em geral, o primeiro tem sido observado em função da quantidade de casos de uso do SUT. Já o nível de desempenho requerido tem sido representado em termos de restrições para o tempo de resposta do sistema.

Tabela 5: Fatores de esforço relacionados ao sistema sob teste.

Mnemônico[3]	Fator de esforço	Descrição	Ocorrências
ACD	Diversidade de atores (*Actors diversity*)	A medida da variedade de diferentes tipos de atores que interagem com o SUT, sejam eles humanos ou interfaces com outros sistemas.	[R1] [R4] [R19] [R23] [R32] *Subtotal*: 5
DOQ	Qualidade da documentação (*Documentation quality*)	A medida em que a documentação do SUT contempla coerente e consistentemente seus requisitos.	[R8] [R26] *Subtotal*: 2
LOC	Nível de criticidade (*Level of criticalness*)	A medida da importância de uma funcionalidade ou pedaço de código no contexto do sistema.	[R19] [R32] *Subtotal*: 2
LOS	Nível de segurança física pessoal requerida (*Level of safety*)	O nível de perigo ou risco oferecido pelo SUT para a equipe ou outras partes interessadas.	[R17] *Subtotal*: 1
RCE	Coexistência requerida (*Required co-existence*)	A medida em que o SUT deve desempenhar suas funções de maneira eficiente, enquanto compartilha o mesmo ambiente de hardware ou software com outro sistema.	[R17] *Subtotal*: 1
RDC	Capacidade de dados requerida (*Required data capacity*)	O volume de dados a serem armazenados em um arquivo ou banco de dados do SUT.	[R3] [R12] [R24] *Subtotal*: 3

[3] Os mnemônicos não correspondem exatamente a acrônimos dos fatores apresentados porque foram originalmente definidos em Inglês.

REC	Complexidade dos requisitos (*Requirements complexity*)	O nível de dificuldade para compreender os requisitos do SUT, influenciado pela quantidade e variedade de regras de negócios.	[R3] [R12] [R14] [R16] [R24] [R34] *Subtotal*: 6
REP	Nível de desempenho requerido (*Required level of performance*)	A medida em que o SUT deve desempenhar suas funções sob limites de tempo de processamento e taxas de *throughput* específicos.	[R1] [R4] [R11] [R19] [R20] [R21] [R22] [R23] [R32] [R35] [R37] *Subtotal*: 11
RER	Nível de confiabilidade requerido (*Required level of reliability*)	A medida em que o SUT deve executar suas funções sob determinadas condições sem apresentar falhas por um período de tempo especificado.	[R8] [R11] [R13] [R14] [R21] [R22] [R37] *Subtotal*: 7
RLI	Nível de interoperabilidade requerido (*Required level of interoperability*)	A medida em que o SUT deve trocar informações com outros sistemas ou componentes para desempenhar suas funções.	[R17] [R21] [R22] *Subtotal*: 3
RLP	Nível de portabilidade requerido (*Required level of portability*)	A medida em que o SUT deve ser transferível de um ambiente de hardware ou software para outro.	[R11] [R17] [R21] [R22] [R37] *Subtotal*: 5
RLR	Nível de recuperabilidade requerido (*Required level of recoverability*)	A medida em que, em caso de uma interrupção ou uma falha, o SUT deve recuperar os dados diretamente afetados e restabelecer o estado desejado do sistema.	[R8] *Subtotal*: 1
RLS	Nível de segurança requerido (*Required level of security*)	A medida em que o SUT deve proteger suas informações e dados.	[R1] [R4] [R19] [R20] [R23] [R32] [R35] *Subtotal*: 7
RLU	Nível de usabilidade requerido (*Required level of usability*)	A medida em que a operação do SUT deve ser compreensível pelos usuários em um contexto de utilização.	[R8] [R21] [R22] [R37] *Subtotal*: 4
ROC	Legibilidade do código (*Readability of the code*)	A medida da facilidade e clareza com que um leitor humano pode compreender a finalidade, o fluxo de controle e a operação do código-fonte.	[R8] *Subtotal*: 1
RSS	Tamanho da especificação de requisitos (*Requirements specification size*)	A medida da quantidade de requisitos contemplados pelo sistema.	[R1] [R4] [R16] [R17] [R18] [R19] [R23] [R25] [R32] [R34] *Subtotal*: 10
REI	Instabilidade dos requisitos (*Requirements instability*)	A medida em que os requisitos mudam depois que eles foram aprovados - geralmente após a fase de especificação de requisitos.	[R13] [R17] [R21] [R25] [R26] [R29] *Subtotal*: 6
SCC	Complexidade do código-fonte (*Source code complexity*)	O nível de dificuldade para compreender o código-fonte, influenciado pela complexidade ciclomática, tempo de execução, profundidade de herança ou da árvore de chamadas.	[R8] [R9] [R15] [R19] [R25] [R32] [R37] *Subtotal*: 7
SCS	Tamanho do código-fonte (*Source code size*)	A quantidade de código-fonte usado para desenvolver o sistema.	[R8] [R9] [R18] [R26] [R37] *Subtotal*: 5
SIC	Complexidade das interfaces do software (*Software interface complexity*)	A medida da quantidade e diversidade de integrações de SUT com outros sistemas, influenciada pela quantidade e variedade de interfaces.	[R1] [R4] [R17] [R19] [R20] [R23] [R26] [R32] [R35] *Subtotal*: 9
SYC	Complexidade do sistema (*System complexity*)	A medida da quantidade e diversidade de componentes que formam o SUT, influenciada pela quantidade e variedade de operações, entidades e dispositivos manipulados pelo sistema.	[R13] [R21] [R22] [R25] [R26] *Subtotal*: 5

Os fatores de influência mais frequentes relacionados com a equipe de teste dizem respeito à experiência da equipe. Foram identificados quatro tipos de experiência: experiência com teste, experiência com o domínio de aplicação, experiência com a tecnologia de programação e experiência com o ambiente operacional, sendo os dois primeiros os mais frequentes, conforme mostra a Tabela 6. Em geral, os modelos medem a experiência pela extensão de tempo que cada equipe atuou seja em testes, no domínio de aplicação, com o ambiente operacional ou com a tecnologia de programação. No entanto, o tempo de experiência se mostra apenas como um *surrogate* (representante) para o nível de conhecimento prático das equipes, que eventualmente pode não possuir correlação com o tempo de experiência.

Tabela 6: Fatores de esforço relacionados à equipe de teste.

Mnemônico	Fator de esforço	Descrição	Ocorrências
EAD	Experiência com o domínio de aplicação (*Experience with the application domain*)	Conhecimento prático sobre o negócio do cliente e as características do sistema, acumulado ao longo do tempo pelos membros da equipe de teste.	[R3] [R8] [R12] [R13] [R17] [R21] [R22] [R26] [R37] *Subtotal*: 9
EOE	Experiência com o ambiente operacional (*Experience with the operating environment*)	Conhecimento prático sobre o ambiente operacional do SUT, acumulado ao longo do tempo pelos membros da equipe de testes.	[R8] [R26] [R37] *Subtotal*: 3
EPT	Experiência com a tecnologia de programação (*Experience with the programming technology*)	Conhecimento prático sobre a tecnologia de programação empregada no projeto, acumulado ao longo do tempo pelos membros da equipe de teste.	[R8] [R17] [R21] [R22] [R23] [R26] [R37] *Subtotal*: 7
EXT	Experiência com teste (*Experience with testing*)	Conhecimento prático em testes de software, acumulado ao longo do tempo pelos membros da equipe de teste.	[R3] [R7] [R11] [R12] [R17] [R18] [R23] [R26] [R28] [R33] [R34] *Subtotal*: 11
TCA	Capacidade da equipe (*Team capacity*)	A habilidade da equipe de teste para realizar um projeto de testes de forma eficaz e com maestria.	[R11] [R17] [R21] [R22] [R26] [R28] [R37] *Subtotal*: 7
TCN	Continuidade da equipe (*Team continuity*)	A capacidade da equipe de teste de manter seus membros trabalhando juntos por um longo período sem alterações (baixo *turnover*).	[R11] [R17] [R26] [R37] *Subtotal*: 4
TCO	Cooperação da equipe (*Team cooperation*)	O grau de cooperação entre engenheiros de teste, desenvolvedores de software, gerentes de projeto, coordenadores e outras partes interessadas para realizar o trabalho de teste de software.	[R17] [R26] *Subtotal*: 2
TEP	Produtividade da equipe (*Team productivity*)	A medida do número de tarefas e produtos de trabalho realizados pela equipe de teste ao longo do tempo.	[R2] [R7] [R24] [R28] *Subtotal*: 4
TTS	Tamanho da equipe de teste (*Test team size*)	O número de testadores disponíveis para executar as atividades de teste necessárias.	[R17] [R18] [R19] *Subtotal*: 3

Quanto à categoria de fatores "projeto de teste" vale ressaltar que, inicialmente, era dividida em duas categorias: projeto de teste e processo de teste. Entretanto, havia fatores de influência que poderiam estar em ambas as categorias, como "Ciclos de teste" (TET), já que podem ser determinados pelo processo ou por necessidades específicas do projeto. Trata-se da categoria com a menor quantidade total de ocorrências (20), o que pode representar uma omissão da maioria dos modelos de estimativa de esforço. Como mostra a Tabela 7, o fator mais frequente desta categoria é "Tarefas do processo de teste" (TPT), uma medida que indica o volume de trabalho a ser feito.

Tabela 7: Fatores de esforço relacionados ao projeto de teste.

Mnemônico	Fator de esforço	Descrição	Ocorrências
PLC	Nível de confidencialidade do projeto (*Project's level of confidentiality*)	A confidencialidade exigida das informações sobre o projeto	[R17] *Subtotal*: 1
PTD	Distribuição física da equipe do projeto (*Project's team distribution*)	A distribuição geográfica e organizacional da equipe do projeto.	[R17] [R26] *Subtotal*: 2
RTC	Nível de cobertura requerido (*Required test coverage*)	A medida em que os casos de teste do projeto devem cobrir os requisitos do SUT.	[R30] *Subtotal*: 1
SCP	Pressão sobre o cronograma (*Schedule pressure*)	A restrição de tempo imposta ao projeto de teste, frequentemente colocando pressão sobre a equipe de teste.	[R8] [R17] [R21] [R26] [R37] *Subtotal*: 5
TPC	Capacidade do processo de teste (*Test process capability*)	A medida de como o processo de testes do projeto apoia os membros da equipe de teste em produzir os resultados necessários.	[R17] [R26] [R33] *Subtotal*: 3
TPT	Tarefas do processo de teste (*Testing process tasks*)	A quantidade de tarefas de teste necessárias para testar um produto de software.	[R13] [R21] [R22] [R26] [R34] *Subtotal*: 5
TET	Ciclos de teste (*Testing trials*)	O número de ciclos de teste necessários para atingir os critérios de qualidade exigidos pelo projeto.	[R16] [R28] [R34] *Subtotal*: 3

A categoria "ambiente de teste" procura representar fatores relacionados às instalações, hardware, software, procedimentos e ferramentas usados para realizar as atividades de teste. Os fatores de influência com maior ocorrência relacionados ao ambiente de teste são a "Complexidade do ambiente de teste" (CTE) e o "Suporte de ferramentas de teste" (STT), conforme a Tabela 8. É possível perceber nos trabalhos analisados que a complexidade do ambiente de teste é frequentemente influenciada pela necessidade de integração deste ambiente com aplicações externas. Quanto ao uso de ferramentas de teste, os artigos, em geral, consideram não apenas ferramentas de apoio a execução dos testes, mas também aquelas que apoiam o planejamento dos testes e a gestão de defeitos.

Tabela 8: Fatores de esforço relacionados ao ambiente de teste.

Mnemônico	Fator de esforço	Descrição	Ocorrências
UIR	Indisponibilidade de recursos de infraestrutura (*Unavailability of infrastructure resources*)	A falta de disponibilidade de recursos de infraestrutura e seus substitutos para as várias fases de testes.	[R17] *Subtotal*: 1
CTE	Complexidade do ambiente de teste (*Complexity of the testing environment*)	O nível de dificuldade para configurar o ambiente de teste, influenciado pelo número de diferentes plataformas, o número de ambientes de teste diferentes e a necessidade de simuladores.	[R1] [R2] [R3] [R12] [R13] [R16] [R17] [R19] [R23] [R24] [R32] [R34] [R35] *Subtotal*: 13
SPA	Procedimentos específicos para acessar o sistema (*Specific procedures to access the system*)	Os procedimentos específicos necessários para acessar o SUT devido à política de segurança da empresa.	[R3] [R12] [R17] [R24] *Subtotal*: 4
STT	Suporte de ferramentas de teste (*Support of testing tools*)	A medida em que ferramentas de teste estão efetivamente apoiando as atividades de teste de software.	[R1] [R4] [R13] [R16] [R17] [R19] [R20] [R21] [R22] [R23] [R26] [R32] [R33] [R35] [R37] *Subtotal*: 15

Já a categoria "*testware*" procura representar os fatores de esforço relacionados aos artefatos do processo de teste de software. Como mostra a Tabela 9, esta categoria possui o fator de esforço mais frequente entre todas as categorias: "Complexidade de execução de teste" (TEC). Trata-se de um fator de esforço que sofre influência de muitas variáveis, o que pode justificar a maior quantidade de ocorrências. Essa preocupação dos modelos com esse fator também pode ser explicada pelo fato de que a execução de testes é a atividade mais frequentemente coberta pelos modelos de estimativa de esforço, conforme irá mostrar a seção 3.3.3.

Uma representação da cobertura de fatores por cada trabalho analisado pode ser encontrada na Tabela 10. É possível observar que o trabalho de Deonandan *et al*. [R17] é o que apresenta a maior cobertura de fatores (26), o que é natural por se tratar de um *survey*, seguido pelo trabalho de Lu e Yin [R26] que cobre 18 dos 47 fatores apresentados. O trabalho de Lu e Yin [R26], no entanto, apresenta um modelo de fato, com os fatores divididos em categorias de modo semelhante à classificação apresentada nesta revisão.

Tabela 9: Fatores de esforço relacionados aos artefatos de teste (*testware*).

Mnemônico	Fator de esforço	Descrição	Ocorrências
ARA	Disponibilidade de artefatos de teste reusáveis (*Availability of reusable test artifacts*)	A medida da disponibilidade de artefatos de teste (e.g. casos de teste, dados de teste, etc.) reusáveis requeridos para realização do projeto de teste.	[R1] [R4] [R13] [R17] [R19] [R20] [R23] [R31] [R32] [R35] *Subtotal*: 10
QTI	Quantidade de incidentes de teste (*Quantity of testing incidents*)	O número de incidentes relatados no relatório de teste.	[R8] [R10] [R16] [R17] *Subtotal*: 4
NTC	Número de casos de teste (*Number of test cases*)	O número de casos de teste necessários para atingir os critérios de qualidade do projeto.	[R3] [R5] [R6] [R7] [R8] [R12] [R16] [R17] [R18] [R24] *Subtotal*: 10
TDQ	Quantidade de dados de teste (*Test data quantity*)	A quantidade de dados criados ou selecionados para satisfazer os requisitos de entrada para executar um ou mais casos de teste.	[R2] [R3] [R12] [R24] [R26] [R27] [R34] *Subtotal*: 7
TEC	Complexidade de execução de teste (*Test execution complexity*)	O nível de dificuldade para executar cada *script* de teste, considerando a dependência entre casos de teste, dados de teste, tipos de itens de tela a serem verificados, o número de ações na interface do usuário e a necessidade de manipulação de arquivos.	[R1] [R2] [R3] [R4] [R5] [R7] [R8] [R12] [R16] [R17] [R18] [R19] [R23] [R24] [R27] [R32] [R34] [R35] *Subtotal*: 18
TSZ	Tamanho dos *scripts* de teste (*Test scripts size*)	O número de passos necessários para executar cada caso de teste.	[R2] [R3] [R6] [R12] [R18] [R24] [R31] [R34] *Subtotal*: 8

Tabela 10: Fatores de esforço identificados por artigo analisado.

Fatores		Artigos selecionados																																						
Categoria	Id	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22	R23	R24	R25	R26	R27	R28	R29	R30	R31	R32	R33	R34	R35	R36	R37
SUT	ACD	X			X															X				X									X					
	SCC								X	X						X				X						X							X					X
	SCS								X	X									X								X											X
	DOQ								X																		X											
	LOC																			X													X					
	REP	X			X							X								X	X	X	X	X									X			X		X
	RER								X			X		X	X							X	X															X
	LOS																	X																				
	REC			X									X		X		X								X										X			
	RCE																	X																				
	RDC			X									X												X													
	RLI																	X				X	X															
	RLP											X						X				X	X															X
	RLR								X																													
	RLS	X			X															X	X			X									X			X		
	RLU								X													X	X															X
	ROC								X																													
	RSS	X			X												X	X	X	X				X		X							X		X			
	REI													X				X				X				X	X			X								
	SYC													X								X	X			X	X											
	SIC	X			X													X		X	X			X			X						X			X		
Equipe de teste	EAD			X					X				X	X				X				X	X				X										X
	EOE								X																		X											X
	EPT								X									X				X	X	X			X											X
	EXT			X				X				X	X					X	X					X			X		X					X	X			
	TCA											X						X				X	X				X		X									X
	TCN											X						X									X											X
	TCO																	X									X											
	TEP		X					X																	X				X									
	TTS																	X	X	X																		
Projeto de teste	PLC																	X																				
	PTD																	X									X											
	TPC																	X									X							X				
	TPT													X								X	X				X									X			
	TET																X											X							X			
	RTC																														X							
	SCP								X									X				X					X											X
Ambiente de teste	UIR																	X																				
	CTE	X	X	X									X	X			X	X		X				X	X							X		X	X		
	SPA			X									X					X							X													
	STT	X			X									X			X	X		X	X	X	X	X			X						X	X		X		X
Testware	ARA	X			X									X				X		X	X			X								X	X			X		
	QTI								X		X						X	X																				
	NTC			X		X	X	X	X				X				X	X	X						X													
	TDQ		X	X									X												X		X	X							X			
	TEC	X	X	X	X	X		X	X				X				X	X	X	X				X	X			X					X		X	X		
	TSZ		X	X			X						X						X						X							X						

3.3.3 QE2.1: Que modelos utilizam esses fatores?

As respostas às questões de pesquisa QE2.1.1, QE2.1.2 e QE2.1.3 são apresentadas nas seções 3.3.3.1, 3.3.3.2 e 3.3.3.3.

3.3.3.1 Tais modelos são baseados em que tipos de técnicas? (QE2.1.1)

Uma das preocupações deste trabalho foi como classificar os modelos de acordo com o tipo de técnica de estimativa. Há diversos trabalhos que propõem formas de classificação de modelos de estimativa de esforço (JøRGENSEN; SHEPPERD, 2007) (MENDES, 2008). Entretanto, tais classificações não se mostraram adequadas para os trabalhos aqui analisados. Sendo assim, decidiu-se por deixar as categorias emergirem dos próprios trabalhos sob análise, evitando uma possível perda de informação relevante. A Tabela 11, portanto, representa a distribuição de frequência dos modelos analisados por tipo de técnica de estimativa.

Tabela 11: Modelos por tipo de técnica de estimativa

Tipo de técnica	Ocorrências
Modelo algorítmico	14
Inteligência artificial (IA)	8
Modelo algorítmico + julgamento de especialistas	4
Julgamento de especialistas	2
Modelo probabilístico	1
Modelo algorítmico + regressão linear	1
Método dos mínimos quadrados + regressão linear	1
Modelo algorítmico + IA	1
Julgamento de especialistas + analogia	1
Raciocínio baseado em casos (CBR)	1
Validação cruzada (K-*fold*)	1
Regressão múltipla *fuzzy*	1
Não se aplica (apenas fatores)	1

É possível observar que a maior parte dos modelos analisados segue uma abordagem algorítmica – dos 37 trabalhos, 14 apresentam modelos estritamente algorítmicos e outros seis apresentam modelos algorítmicos associados a outras técnicas, em especial julgamento do especialista. Cabe destacar que nove modelos são baseados (total ou parcialmente) em técnicas de inteligência artificial (IA), dos quais quatro utilizam os mesmos dados de projetos, variando apenas a técnica de IA utilizada.

3.3.3.2 Esses modelos são adequados para qual contexto de teste? (QE2.1.2)

O contexto de teste foi caracterizado neste estudo por cinco dimensões: nível, tipo, técnica, fase (atividade) e forma de execução de testes, de acordo com a norma ISO/IEC/IEEE 29119-1 (ISO/IEC/IEEE, 2013). Na Figura 9 pode-se notar que a maior parte dos trabalhos não contextualiza claramente cada dimensão de teste para a qual seu

modelo é proposto. Em todas as dimensões de teste consideradas a falta de especificação se mostrou mais frequente.

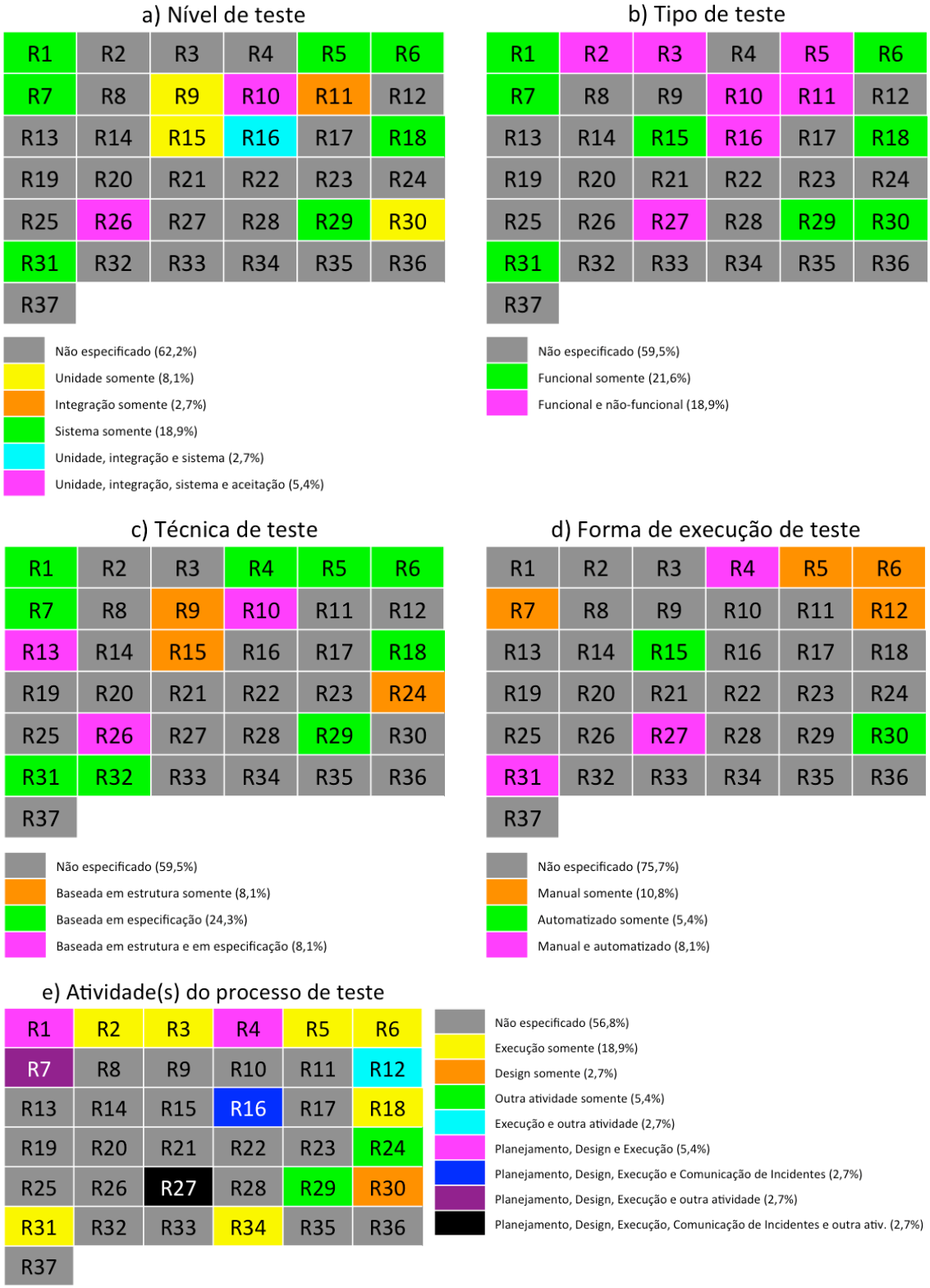


Figura 9: Distribuição de níveis, tipos, técnicas, formas de execução e atividades de teste nos trabalhos analisados.

Apenas cerca de 1/3 dos artigos analisados indica explicitamente o nível de teste coberto pelo modelo, dentre os quais se percebe que o teste em nível de sistema é o mais frequente, como mostra a Figura 9 **(a)**. A Figura 9 **(b)** indica que o tipo de teste funcional é mais frequente do que o teste não-funcional. Em relação à técnica de teste, a Figura 9 **(c)** mostra que a técnica funcional (caixa preta) é mais frequente que a técnica estrutural (caixa branca). Cabe ressaltar também que, dos 37 trabalhos analisados, apenas sete indicam claramente qual a forma de execução dos testes (manual ou automatizada). A Figura 9 **(d)** mostra que destes sete estudos, quatro abrangem apenas a execução manual, um compreende apenas a execução automatizada e dois abrangem ambas as formas de execução. Quanto à fase de teste, a Figura 9 **(e)** mostra que a atividade de execução é a mais frequente entre as atividades de teste indicadas nos modelos que caracterizam essa dimensão.

De acordo com a Figura 9, por exemplo, se um gerente de teste estiver procurando um modelo de estimativa de esforço para um projeto de teste com a seguinte estratégia de teste, ele poderá escolher entre os estudos [R5] e [R6]:

- **Nível de teste**: teste do sistema;
- **Tipo(s) de teste**: funcional;
- **Técnica(s) de teste**: baseada em especificação (caixa preta);
- **Forma de execução de teste**: manual;
- **Fase(s) de teste**: somente execução.

Neste caso, o gerente de teste poderia levar em conta a acurácia dos modelos apresentados nos estudos [R5] e [R6]. As principais medidas da acurácia de modelos de estimativa são a *Mean Magnitude of Relative Error* (MMRE) e a contagem do número de estimativas dentro de *m*% do esforço real, PRED(*m*), onde *m* é geralmente considerado como 0,25. Diversos trabalhos sobre estimativa de esforço de software assumem que os modelos de estimativa de esforço devem ter um MMRE <= 0,2 e um PRED(0,25) >= 0,75 (JøRGENSEN, 2007). No entanto, a acurácia aceitável de um modelo de estimativa depende de muitos fatores inerentes ao contexto em que o modelo está inserido.

Dos 37 artigos analisados nesta RSL, somente 17 apresentam alguma medida da acurácia. Dentre as medidas de acurácia identificadas nos trabalhos analisados, estão o desvio padrão, o *Mean Relative Error* (MRE) e o *Mean Relative Absolute Error*

(MRAE), além do MMRE e PRED(*m*). Alguns desses trabalhos apresentam mais de um resultado de acurácia, quando se referem a mais de um teste realizado ou software, projeto ou modelo avaliado. A Tabela 12 apresenta as medidas de acurácia informadas nos artigos analisados. Porém, cabe ressaltar que dadas as diferentes medidas de acurácia representadas, qualquer comparação é propensa a erros de julgamento.

Tabela 12: Medidas de acurácia dos modelos retratados nos artigos.

Artigo	Desvio	MRE	MMRE	MRAE	PRED(.20)	PRED(.25)
[R1]	-	5,9%	-	-	-	-
[R2]	-	Teste 1: 36,75% Teste 2: 36,12% Teste 3: 17,19%	-	-	Teste 1: 100% Teste 2: 33,33% Teste 3: 50%	-
[R3]	-	-	-	-	-	80%
[R4]	-	Soft. 1: -0,767% Soft. 2: -0,630% Soft. 3: -0,267% Soft. 4: 0,378% Soft. 5: -0,080%	-0,273%	-	-	-
[R5]	-	-	66,02%	-	-	93,75%
[R7]	-	-	-	0,139%	-	83,3%
[R8]	33%	-	-	-	-	-
[R18]	-	-	-	38,4%	-	57,1%
[R19]	-	UCP: 2,8% TPA: 7,2%	-	-	-	-
[R23]	-	1,25%	-	-	-	-
[R25]	-	-	Mod. 1: 0,189% Mod. 2: 0,158%	-	-	75% 79%
[R26]	Proj. 1: 14,03% Proj. 2: -6,40% Proj. 3 9,49%	-	-	-	-	-
[R27]	Caso 1: 16,5% Caso 2: -10%	-	-	-	-	-
[R28]	Proj. 1: -11% Proj. 2: 2% Proj. 3: 3% Proj. 4: 6%	-	-	-	-	-
[R32]	8%	-	-	-	-	-
[R33]	-	-	Proj. A: 0,073% Proj. B: 0,299%	-	-	-
[R36]	-	Estudo 1: 2,4% Estudo 2: 10% Estudo 3: 3,23%	-	-	-	-

O processo de escolha de um modelo de estimativa também pode levar em consideração os fatores de esforço cobertos por cada modelo pré-selecionado. Assim, a Tabela 10 pode ser usada para apoiar esta escolha.

No entanto, como a maioria dos estudos selecionados não indica suas dimensões de teste adequadas, o gerente de teste teria que ler cada estudo para avaliar se cada modelo apresentado é apropriado ao contexto do seu projeto.

3.3.3.3 Que tipos de estudo têm sido realizados para avaliar tais modelos? (QE2.1.3)

Os estudos relatados nos trabalhos foram classificados de acordo com seu tipo. Apesar de parte dos trabalhos analisados indicarem qual o tipo de estudo realizado sob a perspectiva de seus autores, para garantir uma classificação consistente, os estudos foram reclassificados de acordo com os seguintes tipos de estudo e suas correspondentes definições:

- Experimento: estudo com alto nível de controle, geralmente realizado em laboratório, que manipula um fator ou variável de um cenário estudado. Com base na aleatorização, diferentes tratamentos são aplicados aos participantes (*subjects*), mantendo outras variáveis constantes e medindo os efeitos nas variáveis de saída (WOHLIN *et al.*, 2000).
- *quasi*-Experimento: tipo de estudo semelhante ao experimento, em que a atribuição de tratamentos aos participantes não pode se dar de forma aleatória, mas emerge das características dos participantes ou objetos em si (WOHLIN *et al.*, 2000).
- Estudo de caso: estudo conduzido com o propósito de se investigar um fenômeno contemporâneo específico dentro de um espaço de tempo limitado e em um contexto real (WOHLIN *et al.*, 2000). Geralmente, estudos de caso são conduzidos dentro das organizações, observando-se as características de interesse de cada estudo.
- Simulação: estudo de caracterização envolvendo a combinação de vários fatores e níveis, com possíveis interações entre fatores, realizado quando os riscos relativos ao fenômeno real inviabilizam um estudo de campo (DE FRANÇA, 2015). Geralmente, estudos de simulação são realizados com base em modelos que representam o contexto real. No caso deste estudo, os trabalhos classificados como simulação geralmente aplicavam algoritmos de inteligência artificial sobre bases de dados de projetos já concluídos.
- *Survey*: estudo realizado em retrospectiva com o objetivo de coletar informações de pessoas sobre um determinado fenômeno. Geralmente, a coleta de informações é feita por meio de entrevistas e questionários (WOHLIN *et al.*, 2000).

- Prova de conceito: demonstração da aplicabilidade prática de uma determinada proposta de solução, muita vezes realizada por meio de exemplos fictícios geralmente enviesados. Trata-se, portanto, de uma avaliação sem rigor metodológico, mas com custo baixo, o que pode motivar a escolha por esse tipo de estudo.

Portanto, a Tabela 13 mostra que os tipos de estudo mais frequentes são estudo de caso e prova de conceito (13 e 12 ocorrências, respectivamente), seguidos por *survey* (com 11 ocorrências). Vale salientar que há artigos que apresentam mais de um estudo (e.g. um *survey* e dois estudos de caso).

Tabela 13: Frequência de tipos de estudo.

Tipo de Estudo	Ocorrências
Estudo de caso	13
Prova de conceito	12
Survey	11
Simulação	6
quasi-Experimento	2

Para tornar mais precisa a caracterização dos estudos o perfil dos participantes também foi considerado. Observou-se que os estudos utilizam com mais frequência projetos da indústria (25 ocorrências), seguidos de projetos fictícios (dez ocorrências), profissionais (quatro ocorrências) e, por fim, estudantes (uma ocorrência).

3.3.4 Resultados da avaliação de qualidade dos artigos

A Tabela 15 apresenta os resultados da avaliação da qualidade dos artigos incluídos, de acordo com as questões descritas na seção 3.2.5. Esta avaliação visa destacar os artigos mais relacionados ao tema de pesquisa e, consequentemente, dar mais confiança aos resultados deste estudo.

A Tabela 14 mostra que os cinco artigos com maior pontuação foram [R2], [R3], [R4], [R5] e [R30]. Em comum, esses artigos respondem a pelo menos duas questões de pesquisa e apresentam certo rigor metodológico, demonstrando preocupação com a caracterização do contexto, a repetibilidade dos estudos e as ameaças à validade.

Essa avaliação não teve a intenção de excluir da revisão os artigos que acabaram tendo uma pontuação baixa. Nota-se que trabalhos como [R22] e [R29], apesar de

receberem escore total baixo, foram incluídos. Depois que o artigo respondeu às perguntas da pesquisa e atendeu aos critérios de seleção, ele foi automaticamente aceito para revisão. Isso se deve ao fato de que as questões de avaliação da qualidade só poderiam ser respondidas após a revisão e extração de cada trabalho.

A avaliação da qualidade também evidenciou a falta de padronização e má qualidade do relato entre os trabalhos analisados. Dos 37 artigos analisados, 16 apresentaram estudos em um nível de detalhamento que não permitiu entender como os resultados foram obtidos e apenas 14 artigos citaram ou trataram as ameaças à validade.

Tabela 14: Resultados da avaliação de qualidade dos artigos.

Artigo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Total
R1	0	2	1	0	1.5	0	1	1	0	0	1	0	7.5
R2	0	2	2	0	1	1	1	1.5	1	0	1	1	11.5
R3	0	2	2	0	1	1	1	1.5	1	1	1	0.5	12
R4	0	2	2	0	1	1	1	1	1	1	1	1	12
R5	0	2	2	1	2	1	1	0.5	1	1	1	1	13.5
R6	0	0.5	2	1	2	1	1	1	1	0	1	0	10.5
R7	0	1	1	1	2	0	1	0.5	0	0	1	0	7.5
R8	0	1	0	1	0	0	1	0	0	1	0	0	4
R9	0	1	1	1	0.5	0	1	0.5	0	1	1	0	7
R10	1	1	2	1	1.5	0	0	0	1	1	0	0.5	9
R11	0	0.5	2	1	1	0	0	0	0	0	0	0	4.5
R12	0	2	2	1	0.5	1	1	0.5	0	1	1	0.5	10.5
R13	0	0.5	2	1	0	0	0	1	1	0	0	1	6.5
R14	1	1	2	1	0	0	1	0	1	0	1	0	8
R15	0	2	0	1	1.5	0	0	0.5	1	1	1	1	9
R16	0	2	2	1	1.5	1	0	0.5	1	1	1	0	11
R17	0	2	0	1	0	1	0	0.5	1	0	1	1	7.5
R18	0	0.5	0	1	2	0	1	1	1	1	1	1	9.5
R19	0	0	1	0	0	1	1	0	1	0	1	0	5
R20	0	2	2	0	0	1	0	0	1	0	0	0.5	6.5
R21	0	2	2	0	0	1	0	0	0	0	1	0	6
R22	0	2	1	0	0	0	0	0	0	0	0	0	3
R23	0	2	2	0	0	0	1	0	0	0	1	0	6
R24	1	2	1	1	1	1	0	0	0	0	0	0	7
R25	0	0.5	0	0	0	0	1	0.5	1	1	1	0	5
R26	0	2	2	0	1	1	1	0.5	0	0	1	0	8.5
R27	0	2	2	0	1	1	1	1	1	0	1	0	10
R28	0	1	2	1	0	0	1	1	1	0	1	0	8
R29	0	2	0	0	2	0	0	0	0	0	0	0	4
R30	1	2	0	1	2	1	1	1	1	1	1	0	12
R31	0	1	0	1	2	1	0	1	1	1	1	1	10
R32	0	2	2	0	0	0	1	0	0	0	0	0	5
R33	0	0	2	1	0	1	1	1	0	0	1	0.5	7.5
R34	0	1	2	1	1	1	0	1	0	0	1	0	8
R35	0	2	2	0	0	1	0	0	1	0	0	0.5	6.5
R36	0	0	2	0	0	0	1	0	1	0	1	0	5
R37	0	0.5	2	1	0	0	1	0	0	0	0	0	4.5

3.4 Discussão

Analisando os resultados apresentados, é possível verificar que existe falta de consenso no campo sobre o significado de esforço e não há uma visão homogênea sobre o que é teste de software. Em geral, os artigos analisados tratam esses conceitos de forma tácita e os consideram de entendimento comum. Além disso, as poucas definições apresentadas sobre esforço em teste não são detalhadas e são, por vezes, ambíguas. Por exemplo, dos quatro trabalhos que apresentam algum tipo de definição sobre esforço em teste software, três o fazem recursivamente (definem o termo usando o próprio termo que está sendo definido). Outros estudos ([R10] e [R24]) incluem no esforço de teste o esforço gasto com as atividades de depuração - as atividades de depuração correspondem à identificação e correção de defeitos; portanto, são uma consequência dos testes e podem representar retrabalho. Entende-se que, nesse caso, o esforço de depuração deve incrementar o esforço de construção, e não o de teste. Neste estudo, é considerado como esforço de teste o conjunto de ações necessárias para realizar o planejamento, *design* & implementação, execução, comunicação de incidentes e análise dos resultados dos testes, de acordo com a norma ISO/IEC/IEEE 29119-1 (ISO/IEC/IEEE, 2013).

As diferentes unidades de medida de esforço identificadas nos estudos evidenciam essa falta de compreensão do que é o esforço do teste de software. A maioria dos estudos usa medidas do tipo "pessoa-tempo" para representar o esforço de teste. "Pessoa-tempo" ("homem-hora" e todas as suas variações) é uma unidade de medida usada para representar o esforço predominantemente físico e repetitivo empregado na indústria de manufatura desde o final do século XIX, do qual a indústria de software se apropriou mais tarde. Esta unidade de medida não é adequada para representar o esforço empregado nas atividades de desenvolvimento de software, caracterizado pela criatividade, aplicação de técnicas e raciocínio lógico. Enquanto o esforço para realização de atividades físicas e repetitivas é aparentemente linear ao longo do tempo e, portanto, mais previsível, o esforço para realização de atividades de desenvolvimento de software é inconstante, portanto mais difícil de relacionar com o fator tempo. Apesar de a maioria dos trabalhos utilizar medidas do tipo pessoa-tempo, ainda existem trabalhos que consideram equivocadamente apenas tamanho ou tempo como medidas de esforço. O uso de uma unidade de medida inadequada pode levar os gerentes a tomar decisões erradas e pode representar um obstáculo para melhorar a

acurácia das estimativas. Portanto, é necessário investigar uma unidade de medida que capture essas diferenças e represente melhor o esforço intelectual inerente aos projetos de software.

Quanto aos fatores de esforço, observou-se que muitos estudos não os descrevem explicitamente e não informam como medi-los, o que torna seu entendimento suscetível a múltiplas interpretações. Por essa razão, este estudo limitou-se a realizar as atividades previstas na etapa de *open coding* do método *Grounded Theory* (STRAUSS; CORBIN, 1998), considerando que qualquer tentativa de estabelecer relacionamentos entre códigos representaria um nível muito alto de incerteza. Outro ponto crítico observado é a falta de indicação da influência do fator, isto é, se o fator contribui para aumentar ou diminuir o esforço e em que proporção.

A maior parte dos modelos analisados segue uma abordagem algorítmica, apresentando, em geral, equações compostas por fatores que não foram avaliados experimentalmente – a avaliação foi realizada apenas sobre o modelo, com diversos fatores incluídos. Desta forma, não é possível determinar quais fatores realmente exercem influência sobre o esforço de teste, porém, também não é possível dissociá-los de seus modelos. Esta incerteza pode levar gerentes de testes de software a tomar decisões com base em fatores que não exercem influência efetiva sobre o esforço de teste.

Os modelos de estimativa analisados não indicam as dimensões de teste para as quais foram concebidos. Cada dimensão de teste possui fatores de influência específicos ou em diferentes proporções. Por exemplo, a complexidade de execução de casos de teste pode apresentar diferentes medidas de esforço para teste manual e para teste automatizado. A falta de indicação dessas dimensões pode ser explicada pela falta de consenso a respeito do que é a atividade de teste de software.

Em relação aos estudos realizados para avaliar os modelos, uma parcela significativa apresenta baixo nível de controle, como provas de conceito e *surveys*, reduzindo a confiança em seus resultados. Não foi possível observar estudos com maiores níveis de controle e resultados robustos. Esse recurso pode ser explicado pela complexidade da maioria dos modelos de estimativa de esforço, o que pode desencorajar seu uso na indústria e dificultar a realização de estudos experimentais para aumentar a confiança em seu uso.

Portanto, sugere-se que novos estudos sobre estimativa de esforço em testes de software procurem evitar os problemas observados na literatura técnica e revelados

nesta revisão: 1) falta de compreensão do que é o esforço de teste de software; 2) falta de clareza sobre como os fatores de esforço podem ser medidos e que tipo de influência eles exercem no esforço de teste, e 3) falta de informação sobre as dimensões e o contexto do teste. Assim, os gerentes de teste de software terão mais informações para apoiar a escolha de um ou outro modelo de estimativa ou até mesmo para construir um modelo adequado aos seus contextos organizacionais.

3.5 Ameaças à validade

A principal ameaça à validade deste estudo é o viés que o pesquisador pode introduzir ao interpretar o conteúdo de cada trabalho analisado. Para atenuar esta ameaça, a revisão foi realizada em parceria com outro pesquisador (aluno de Doutorado) sob a supervisão de um pesquisador sênior (professor orientador). Assim, todos os formulários de extração foram submetidos a uma revisão por pares. Além disso, sempre que havia algum tipo de impasse entre os dois pesquisadores, o pesquisador sênior era consultado e indicava o tratamento mais adequado.

Outras ameaças à validade incluem:

- As palavras-chave usadas para recuperar os trabalhos relacionados podem ser demasiado restritivas ou inadequadas. Para minimizar esta ameaça foram selecionados sete artigos de controle, os quais serviram para compor e calibrar a *string* de busca;
- A revisão sistemática foi limitada a quatro motores de busca. No entanto, a busca por artigos foi complementada por uma etapa de *snowballing*, a partir da qual mais cinco artigos foram incluídos;
- As publicações selecionadas podem apresentar resultados imprecisos ou incorretos, o que se tentou mitigar através da aplicação de uma avaliação da qualidade dos artigos.

3.6 Considerações Finais

Este capítulo apresentou os resultados de um estudo secundário sobre modelos de estimativa de esforço em testes de software e os fatores de influência que os

compõem. O estudo foi orientado por questões de pesquisa que objetivaram caracterizar o que se entende por esforço em teste de software, quais fatores influenciam o esforço desta atividade e quais modelos utilizam tais fatores.

Foram identificados 47 fatores de esforço em 37 estudos que apresentaram evidência sobre o esforço de teste de software. Esses fatores foram organizados em cinco categorias diferentes, que indicam o perspectiva do ambiente de software em que cada fator está presente. As principais dificuldades encontradas para classificar e organizar os fatores identificados foram a falta de uma terminologia consistente entre os estudos e a falta de clareza conceitual sobre cada fator na maioria dos artigos analisados.

Pode-se concluir que os modelos e fatores identificados não são adequados para qualquer natureza de projeto de software, devido a sua variabilidade. Além disso, uma aparente falta de consenso sobre as atividades de teste de software e o esforço de teste torna a escolha de qualquer modelo um risco.

Na medida em que sabemos, esta é a primeira revisão sistemática da literatura sobre modelos de estimativa e fatores de esforço em testes de software. Apesar das lacunas evidenciadas nos modelos analisados, acredita-se que esses resultados possam contribuir para as pesquisas na área e trazer contribuições iniciais para a prática no setor.

Uma melhor percepção de como cada fator identificado influencia o esforço de teste pode ser obtida através da observação em projetos reais. Eventualmente, cada fator de esforço pode exercer influência em diferentes direções e proporções variáveis de acordo com o contexto do projeto.

4 ESTUDO DE OBSERVAÇÃO[4]

Este capítulo, composto por oito seções, descreve um relato de experiência na indústria no qual fatores de esforço de teste foram observados em dois projetos distintos. A seção 4.1 descreve a motivação e os objetivos da realização deste estudo. A seção 4.2 descreve sucintamente o design do estudo. A seção 4.3 caracteriza os projetos observados. A seção 4.4 descreve quais fatores foram observados em cada tarefa do processo utilizado pelos projetos. A seção 4.5 apresenta como cada fator foi observado nos projetos e qual tipo de influência exerceu. A seção 4.6 discute os resultados encontrados. A seção 4.7 descreve as ameaças à validade identificadas e os respectivos tratamentos adotados. Finalmente, a seção 4.8 apresenta as considerações finais deste estudo.

4.1 Motivação e objetivos

A motivação deste estudo reside na falta de clareza em relação aos fatores de esforço identificados na RSL, como discutido na seção 3.3.2. Com a intenção de melhor compreender os fatores de esforço nela identificados, realizou-se um estudo de observação em projetos reais de teste de software, com o objetivo de observar quais fatores de esforço identificados na RSL se apresentam em projetos na indústria, bem como se outros fatores de esforço poderiam ser identificados nos projetos analisados.

De forma complementar, este estudo também é motivado pela ausência de informações relacionadas à *influência* de cada fator de esforço catalogado na RSL. A influência do fator de esforço indica se o esforço em teste é afetado positiva ou negativamente por um determinado fator de esforço, aumentando ou diminuindo o esforço de uma tarefa específica do processo.

Desta forma, foram definidas as seguintes questões de pesquisa para este estudo:

- **QE5**: Quais fatores de esforço relacionados ao processo de teste de software podem ser identificados em projetos industriais de larga escala?

[4] Uma versão deste capítulo foi publicada no artigo *Observing Effort Factors in the Test Design & Implementation Process of Web Services Projects,* apresentado no II Simpósio Brasileiro de Teste de Software Sistemático e Automatizado (SAST 2017) (SILVA-DE-SOUZA; TRAVASSOS, 2017), recebendo o prêmio de melhor trabalho do evento.

- **QE5.1**: Qual a influência exercida por cada fator em cada tarefa do processo de teste de software?

Quanto à questão QE5.1, cabe enfatizar que o objetivo não é quantificar o esforço, mas observar se o esforço aumenta (influência positiva) ou diminui (influência negativa) de acordo com o fator de esforço e a tarefa observada.

Portanto, o objetivo deste estudo é identificar os fatores de esforço presentes em projetos reais de teste de software de uma determinada companhia do Governo Brasileiro, bem como caracterizar a influência dos fatores de esforço observados. Assim sendo, esse objetivo pode ser representado conforme a Tabela 15, de acordo com o paradigma *Goal-Question-Metric* (GQM) (BASILI; CALDIERA; ROMBACH, 1994):

Tabela 15: Objetivos do estudo de observação de acordo com o paradigma GQM.

Analisar	fatores de esforço
Com o propósito de	caracterizar
Em relação à	influência sobre o esforço de teste de software
Do ponto de vista	profissional de teste de software
No contexto de	projetos de software de larga escala na indústria

4.2 Design do Estudo

Embora este estudo contenha alguns elementos típicos de um estudo de caso como sugerido por Runeson *et al*. (RUNESON *et al.*, 2012), ele foi classificado como um estudo observacional, já que um estudo de caso pressupõe maior nível de isenção do pesquisador. No entanto, é importante ressaltar que trata-se de um estudo *in vivo*, o que pode aumentar o nível de confiança nos resultados observados.

A realização deste estudo envolveu dois papéis distintos: o pesquisador, responsável por observar a incidência de fatores de esforço nos projetos, e o testador, encarregado de testar os sistemas seguindo o processo de testes estabelecido na empresa. No entanto, para realizar esse estudo com um testador seria necessário acompanhar seu trabalho por horas, do início ao fim de um projeto, interrompendo-o constantemente para fazer perguntas. Ademais, o testador recrutado deveria ter capacidade técnica e analítica suficientes para executar, descrever e refletir sobre suas tarefas de teste, o que restringia a escolha de um profissional para cumprir esse papel.

Portanto, por uma questão de disponibilidade de tempo e de viabilidade técnica, a mesma pessoa, o autor deste trabalho, desempenhou os dois papéis.

4.3 Caracterização dos Projetos

Em respeito a acordos contratuais, os projetos descritos neste estudo receberão nomes fictícios: projeto Alfa e projeto Beta. Pela mesma razão, a empresa onde esses projetos foram observados será denominada de XPTO.

A empresa XPTO é uma empresa pública que atua na indústria de tecnologia da informação, prestando serviços para órgãos do Governo Brasileiro. Essa empresa possui unidades instaladas em diversos estados do Brasil. Este estudo ocorreu especificamente na unidade do Rio de Janeiro, na qual há diversas equipes de desenvolvimento de software e uma equipe centralizada que presta serviços de teste de software às equipes de desenvolvimento. Todas essas equipes estão localizadas no mesmo prédio. A equipe de teste é denominada *Célula de Teste* e conta atualmente com 12 profissionais, entre analistas (nove) e técnicos (três), a maioria atuando nesta função há mais de seis anos.

Os projetos analisados foram realizados no ano de 2015: o projeto Alfa no primeiro semestre e o projeto Beta no segundo semestre. Ambos possuem características muito próximas, especialmente pelo fato de se tratarem de projetos de testes de Web Services. Além disso, os sistemas de ambos os projetos seguem uma arquitetura similar, composta por uma camada de interface Web, uma camada de serviços (*Web Services*) e duas outras camadas nas quais as regras de negócio e o banco de dados estão concentrados, baseadas nas tecnologias Natural/ADABAS.

Em ambos os projetos, as atividades de *design*, implementação e execução de testes foram apoiadas pela ferramenta SoapUI (SMARTBEAR, 2015), uma ferramenta específica para testes de serviços. A Figura 10 representa a arquitetura dos sistemas observados, enfatizando a camada na qual os testes foram realizados.

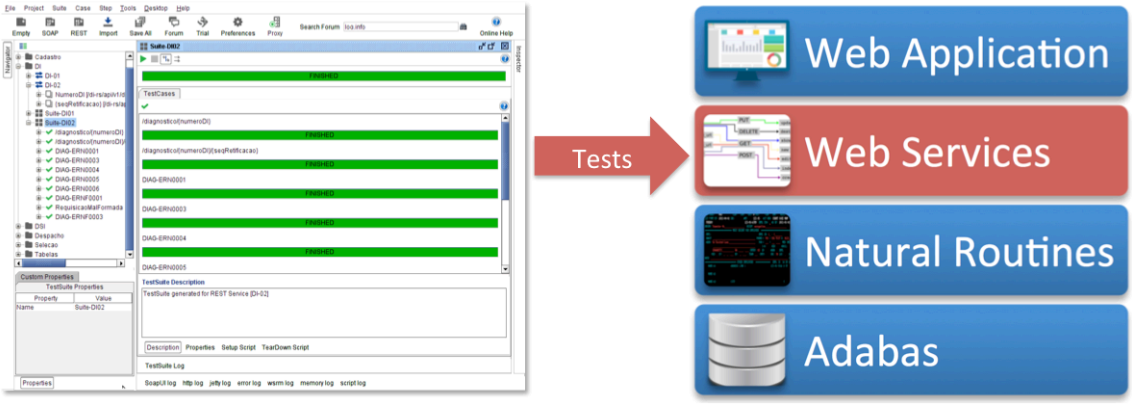


Figura 10: Arquitetura dos sistemas testados nos projetos Alfa e Beta.

O projeto Alfa envolveu o teste de cinco serviços baseados no padrão *Simple Object Access Protocol* (SOAP), em um sistema cujo domínio está relacionado ao registro e legalização de empresas. Neste projeto foram implementados 166 *scripts* de teste, que resultaram na identificação de 33 falhas.

Já o projeto Beta representa um sistema de sistemas relacionados ao domínio de comércio exterior. Para integrar os sete sistemas envolvidos a equipe de desenvolvimento escolheu a tecnologia de Web Services RESTful (FIELDING, 2000). Este sistema compreende mais de 60 serviços, no entanto, este estudo de observação compreendeu apenas 34 deles, já que os demais serviços ainda não estavam desenvolvidos no período de realização do estudo. Para testar esses 34 serviços foram implementados 853 *scripts* de teste, que resultaram na identificação de 210 falhas.

Ambos os projetos utilizaram a mesma estratégia de teste: testes de serviços em nível de unidade, para avaliar requisitos funcionais e de desempenho, aplicando a técnica baseada em especificação (caixa preta). Na etapa de *design* e implementação dos testes foi aplicada a abordagem definida por Silva-de-Souza *et al*. (SILVA-DE-SOUZA *et al.*, 2012), a qual fornece uma sistemática que apoia desde a seleção de casos de teste até a implementação de *scripts* de teste para Web Services.

4.4 Fatores de Esforço

Como já mencionado na seção 1.1, um *fator de esforço* é qualquer característica do ambiente de software que exerce uma influência significativa sobre o esforço requerido para realizar alguma tarefa relacionada ao desenvolvimento de um software.

Neste estudo, parte-se do entendimento que *esforço* é a ação de dispender energia para realizar uma atividade física ou mental necessária para alcançar algum objetivo. Logo, o esforço é classificado em dois tipos: esforço físico e esforço mental. Entende-se por *esforço físico* aquele decorrente da aplicação de movimentos corporais para a realização de uma tarefa. Já o *esforço mental* é aquele que envolve o uso da memória e a aplicação de raciocínio lógico para compreender, interpretar e tomar decisões.

Este estudo procura identificar fatores que influenciam tanto o esforço físico quanto o mental em teste de software, sem a intenção de mensurá-los. Sendo assim, a lista de fatores de esforço identificados na RSL foi utilizada como *baseline*, de modo que procurou-se inicialmente identificar aqueles fatores nos projetos analisados e, somente depois de esgotá-los, procurou-se identificar novos fatores, específicos aos projetos observados.

Os fatores de esforço foram identificados a partir de cada atividade do processo de teste, conforme sugere Pressman (PRESSMAN, 2006). Desta forma, a aplicação de esforço foi analisada em função das tarefas necessárias a realização de cada atividade do processo, considerando, inclusive, o esforço relacionado à criação e manutenção dos artefatos de ambos os projetos.

Conforme ilustra a Figura 11, o processo de teste utilizado para os projetos Alfa e Beta é composto pelas seguintes atividades, de modo aderente à norma ISO/IEC/IEEE 29119-2 (ISO/IEC/IEEE, 2013):

- Planejar Testes
- Configurar Ambiente de Testes
- Projetar e Implementar Testes
- Executar Testes
- Comunicar Incidentes de Teste
- Finalizar Projeto de Testes

O pesquisador-testador já havia executado esse processo de teste em inúmeros projetos ao longo dos últimos dez anos, considerando diferentes níveis, tipos e técnicas de teste. No entanto, os projetos Alfa e Beta foram suas primeiras experiências em projetos reais de testes de Web Services.

É possível observar na Figura 11 que os fatores de esforço identificados estão indicados em pequenas notas conectadas a cada tarefa. Essas notas contém o

mnemônico de cada fator de esforço observado. Vale ressaltar que, dentre os fatores identificados com a observação dos projetos, há quatro fatores que não estavam presentes na RSL, os quais são descritos na Tabela 16:

Tabela 16: Fatores de esforço identificados não-presentes na RSL.

Mnemônico	Fator de esforço	Descrição
NWS	Número de Web Services	A medida da quantidade de serviços a serem testados no projeto.
ETT	Experiência com ferramentas de teste	Conhecimento prático sobre as ferramentas de teste de software utilizadas no projeto acumulado ao longo do tempo pelos membros da equipe de teste.
TSS	Tamanho da especificação técnica	A medida da quantidade de informações técnicas contidas nos documentos que descrevem os serviços sob teste.
ADI	Quantidade e duração de interrupções	A quantidade e duração das interrupções do trabalho em cada projeto, motivadas por tarefas relacionadas a outros projetos, pausas para refeições, descanso e férias, assim como para aguardar a disponibilização de uma informação ou artefato do projeto.

4.5 Rationale

A presença e a influência de cada fator de esforço é justificada neste estudo através de um *rationale* do ponto-de-vista do testador-pesquisador. Trata-se de uma evidência anedótica cujo objetivo é apresentar o raciocínio que levou o testador-pesquisador a associar os fatores de esforço a cada tarefa do processo.

Este *rationale* é descrito nas tabelas a seguir, nas quais cada fator de esforço é identificado por seu mnemônico. Ao lado do mnemônico de cada fator há uma seta que indica o sentido de influência do fator: seta para cima (↑) indica que o fator exerce influência positiva sobre o esforço, ou seja, quanto maior a presença daquele fator maior será o esforço relacionado àquela tarefa; de forma inversa, a seta para baixo (↓) indica que quanto maior a presença daquele fator, menor será o esforço naquela tarefa.

A Tabela 17 apresenta o *rationale* sobre os fatores de esforço transversais, identificados na legenda da Figura 11 por um asterisco. Esses fatores são assim classificados porque podem ocorrer em qualquer tarefa do processo, portanto, não aparecem nas notas conectadas às tarefas.

Tabela 17: *Rationale* sobre os fatores de esforço transversais.

Fator	*Rationale*
REI (↓)	Os requisitos do projeto Alfa se mostraram estáveis durante o projeto. No entanto, no projeto Beta houve mudanças em diversos requisitos após seus testes terem sido executados. Isso implicou em atualizar o Plano de Teste e modificar diversos *scripts* de teste que já haviam sido executados, implicando em retrabalho nas atividades "Planejar Teste", "Projetar e Implementar Testes" e "Executar Testes". Desta forma, percebeu-se que quanto maior a estabilidade dos requisitos, menor será o esforço do projeto de teste.
SCP (↑)	No projeto Beta havia pressão por parte da equipe de desenvolvimento para concluir os testes em um prazo relativamente curto. Isso contribuiu para aumentar o esforço de teste, tendo em vista a preocupação com o cumprimento de prazos.
TCA (↓)	Ambos os projetos foram realizados por um testador que já havia liderado e participado de diversos projetos de teste que obtiveram êxito, inclusive conduzindo-os individualmente. Além disso, o testador possui conhecimento em outras disciplinas de Engenharia de Software, o que influenciou na sua capacidade de realizar os projetos analisados de forma eficaz. Desta forma, a capacidade do testador influenciou negativamente o esforço de ambos os projetos.
TCN (↓)	Como os projetos utilizaram o mesmo testador, não houve necessidade de realocar pessoas e consequentemente treiná-las para realizar as tarefas de cada projeto. Desta forma, a continuidade do testador exerceu influência negativa sobre o esforço de teste.
ADI (↑)	Em ambos os projetos a quantidade e a duração das interrupções causaram efeito positivo sobre o esforço, tendo em vista que, dependendo da interrupção, poderia haver esforço para relembrar em que ponto o trabalho foi interrompido e qual solução estava sendo aplicada.
TEP (↓)	Em ambos os projetos, em especial no projeto Beta, devido à quantidade maior de serviços testados, observou-se que à medida que o testador aumentava sua produtividade para testar serviços, o esforço necessário para cada tarefa do processo diminuía. Assim, a produtividade do testador afetou negativamente o esforço de teste.
TTS (↑)	Ambos os projetos contaram com apenas um testador. Desta forma, não havia a necessidade de o testador se reunir ou passar informações para outros membros da equipe de teste. Se a equipe de teste alocada aos projetos tivesse mais membros, o esforço de comunicação da equipe em todas as tarefas do processo seria influenciado positivamente.

Legenda: REI = Instabilidade dos requisitos; SCP = Pressão sobre o cronograma; TCA = Capacidade da equipe; TCN = Continuidade da equipe; ADI = Quantidade e duração de interrupções; TEP = Produtividade da equipe; TTS = Tamanho da equipe de teste.

A partir da subseção 4.5.1 os fatores de esforço são descritos em função das atividades e tarefas que compõem o processo de teste da organização.

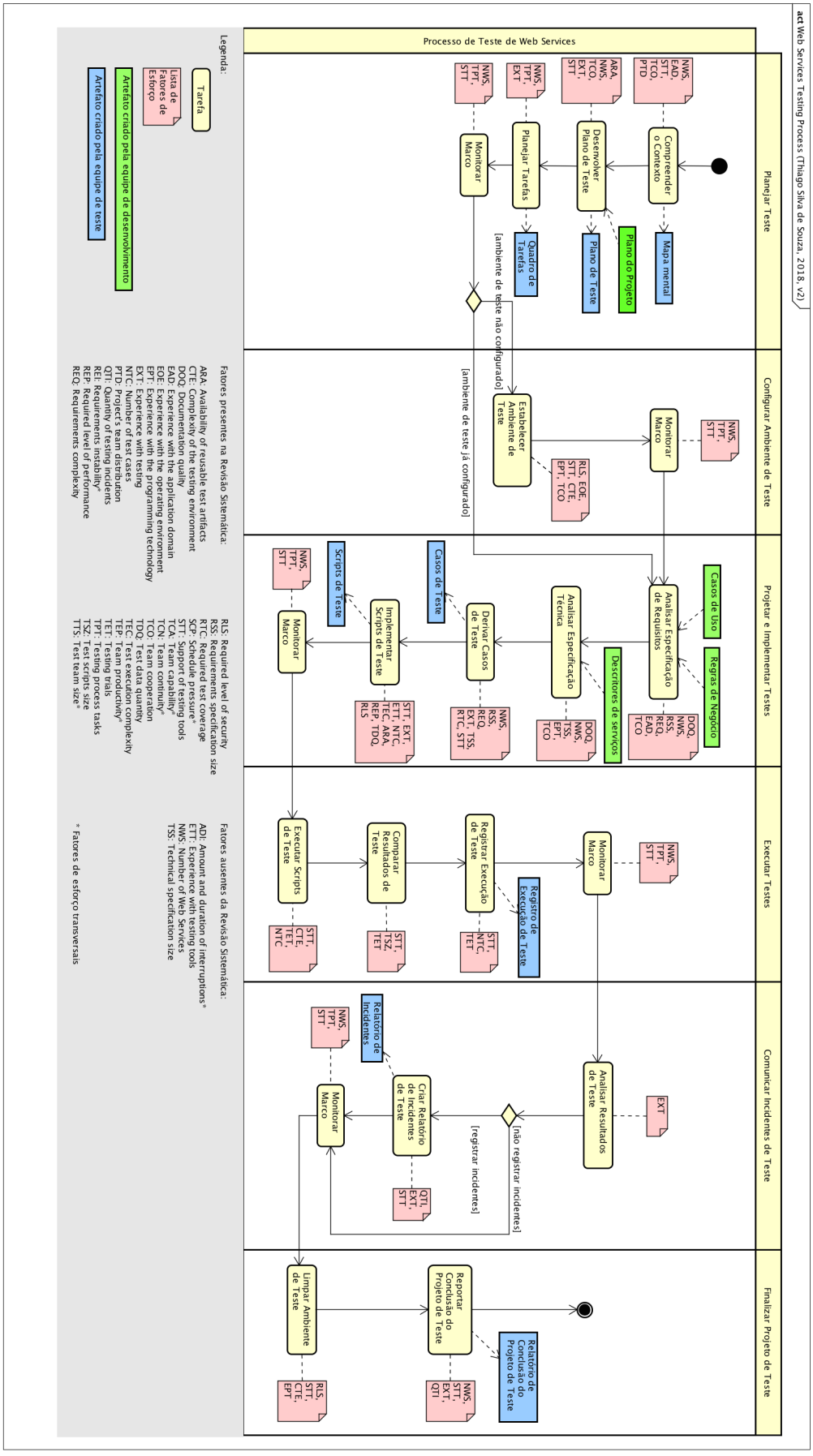


Figura 11: Mapeamento de fatores de esforço no processo de teste analisado.

4.5.1 Atividade “Planejar Teste”

A atividade “Planejar Teste” tem como objetivo principal desenvolver o *Plano de Teste*, artefato que direciona a realização do projeto de teste. Para isso, essa atividade é composta por quatro tarefas que, de forma geral, apoiam a criação do Plano de Teste. A Figura 12 representa as tarefas dessa atividade, bem como os artefatos envolvidos e os fatores de esforço observado em cada tarefa.

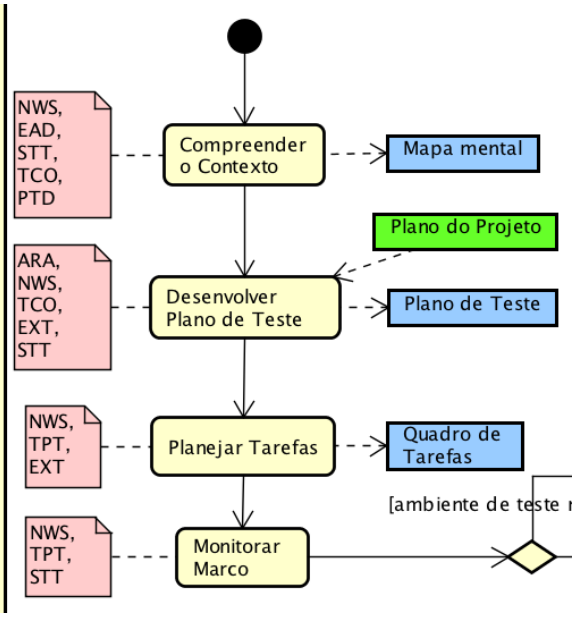


Figura 12: Tarefas, artefatos e fatores de esforço da atividade Planejar Teste.

Na organização observada neste estudo, o Plano de Teste é preenchido por meio de uma ferramenta de gestão de projetos de teste. Esse documento é composto por uma série de seções, onde são definidos o escopo do projeto, a estratégia de teste a ser utilizada, o tratamento dado aos riscos identificados, bem como é onde são anexados o Plano do Projeto, elaborado pela equipe de desenvolvimento, e o Relatório de Conclusão do Projeto de Teste, criado no final do projeto.

A primeira tarefa desta atividade, “Compreender o Contexto”, envolve a realização de encontros formais e informais entre as equipes de desenvolvimento e de testes, a fim de que a equipe de testes compreenda o conjunto de circunstâncias que compõem o contexto do projeto. A Tabela 18 apresenta o *rationale* que envolveu a identificação de cada fator de esforço desta tarefa.

Tabela 18: *Rationale* sobre os fatores observados na tarefa "*Compreender o Contexto*".

Fator	*Rationale*
NWS (↑)	Em ambos os projetos observou-se que o esforço para o testador compreender o contexto era proporcional à quantidade de serviços que compunham o escopo do projeto. Quanto mais serviços a testar, mais esforço de leitura de documentos e discussão com outras partes interessadas para compreender o contexto.
EAD (↓)	O projeto Alfa contemplava um domínio de aplicação nunca antes exercitado pelo testador. Já no projeto Beta observou-se que o conhecimento prévio do domínio de aplicação proporcionou um efeito negativo sobre o esforço de compreensão do contexto, tendo em vista que o testador havia atuado por cerca de dois anos como analista de requisitos e implementador em dois dos sete sistemas envolvidos no projeto Beta.
STT (↓)	Em função da quantidade de sistemas e serviços envolvidos no projeto Beta, foi necessária a criação de um mapa mental para facilitar a compreensão do contexto. Este mapa mental foi criado com apoio de uma ferramenta específica para este fim, composta por recursos que proporcionaram um efeito negativo sobre o esforço.
TCO (↓)	Inicialmente, no projeto Alfa o desenvolvedor alocado como ponto focal para dirimir quaisquer dúvidas não se mostrava solícito, o que contribuiu para um esforço maior por parte do testador e, consequentemente, um atraso considerável no projeto de teste. Isto motivou uma reunião na qual as partes interessadas firmaram um novo compromisso de cooperação. A partir de então, o desenvolvedor passou a cooperar mais com o testador, o que resultou na aplicação de menos esforço de compreensão do contexto. Já no projeto Beta, os gerentes de projeto envolvidos, bem como suas equipes, mostraram-se solícitos e disponíveis na maior parte do tempo para auxiliar o testador a compreender o contexto, o que exerceu efeito negativo sobre o esforço.
PTD (↑)	A falta de cooperação observada inicialmente no projeto Alfa pode ser explicada em parte pela distância física entre as áreas de desenvolvimento e de testes. Apesar de estarem localizadas no mesmo andar de um prédio, sempre que o testador precisava resolver algum impedimento sobre o projeto que necessitasse de uma conversa face a face, ele se deslocava até a sala da equipe de desenvolvimento, aplicando esforço físico para compreender o contexto. No caso do projeto Beta, dois dos três departamentos de desenvolvimento envolvidos são vizinhos ao departamento de teste, o que favorece a cooperação e, consequentemente, exige menos esforço com deslocamentos para compreender o contexto.

Legenda: NWS = Número de Web Services; EAD = Experiência com o domínio de aplicação; STT = Suporte de ferramentas de teste; TCO = Cooperação da equipe; PTD = Distribuição física da equipe do projeto.

Compreendido o contexto, a próxima tarefa é desenvolver o Plano de Teste. Para cada projeto deve-se elaborar um Plano de Teste. Vale ressaltar que os planos de teste seguem um modelo (*template*) já implementado pela ferramenta de gestão do projeto, na qual este documento é editado. A Tabela 19 descreve o *rationale* aplicado para identificar os fatores de esforço relacionados.

Tabela 19: *Rationale* sobre os fatores observados na tarefa "*Desenvolver Plano de Teste*".

Fator	*Rationale*
ARA (↓)	A elaboração dos planos de teste do projeto Beta teve seu esforço reduzido porque utilizou como base o plano de teste do projeto Alfa, tendo em vista que a seção "Estratégia de Teste" possuía um conjunto de informações comuns e que poderiam ser aproveitadas.
NWS (↑)	Uma das seções que compõem o Plano de Teste é denominada "Escopo". Nesta seção são listadas todas as funcionalidades a serem cobertas pelo projeto de teste. Em ambos os projetos analisados, o esforço para escrever esta seção aumentava proporcionalmente à quantidade de serviços a serem testados.
EXT (↓)	Em ambos os projetos, o esforço de elaboração do Plano de Teste foi diminuído em função da larga experiência do testador em projetos de teste de software, tendo em vista que todo projeto requer tal artefato. O conhecimento acumulado pelo testador possibilitou um efeito negativo sobre o esforço para realizar esta tarefa.
STT (↓)	Os planos de teste de ambos os projetos foram elaborados através de uma ferramenta apropriada para tal finalidade, o que provocou efeito negativo sobre o esforço.
TCO (↓)	O Plano de Teste deve referenciar os requisitos que serão testados na seção "Link para Coleção de Requisitos". Para que o testador possa associar os requisitos ao Plano de Teste é necessário definir uma coleção de requisitos na ferramenta de gestão de requisitos, o que geralmente é feito pela equipe de desenvolvimento. Caso não o faça, o próprio testador deve fazê-lo, o que não foi necessário em ambos os projetos. Portanto, a cooperação com a equipe de desenvolvimento exerceu influência negativa sobre o esforço para desenvolver o Plano de Teste.

Legenda: ARA = Disponibilidade de artefatos de teste reusáveis; NWS = Número de Web Services; EXT = Experiência com teste; STT = Suporte de ferramentas de teste; TCO = Cooperação da equipe.

O planejamento de tarefas de ambos os projetos foi realizado com o auxílio de um quadro de tarefas, conforme mostra a Figura 13. Esta tarefa compreende a divisão do escopo de teste em lotes de acordo com a granularidade do componente sob teste, além da definição das atividades de teste a serem realizadas, de acordo com o contexto do projeto. Portanto, além do esforço mental relacionado a essas decisões, a montagem do quadro implicou em dispêndio de esforço físico, como descrevem os fatores relacionados na Tabela 20. Cabe ressaltar que o custo financeiro decorrente da aquisição do material utilizado está fora desta análise.

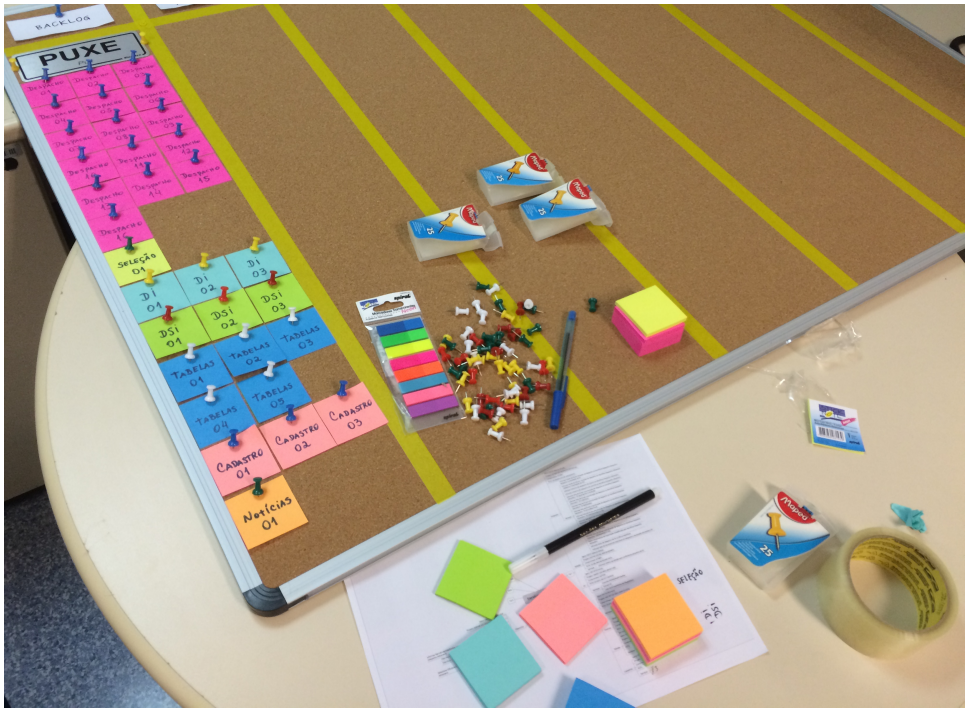


Figura 13: Montagem do quadro de tarefas para o projeto Beta.

Tabela 20: *Rationale* sobre os fatores observados na tarefa "*Planejar Tarefas*".

Fator	*Rationale*
NWS (↑)	Em ambos os projetos, a quantidade de Web Services determinou quantos *post-its* compunham o quadro de tarefas. Cada serviço foi representado por um *post-it*, exigindo esforço predominantemente físico. Desta forma, a quantidade de serviços a serem testados afetou positivamente o esforço.
TPT (↑)	A quantidade de atividades do processo de teste influenciou positivamente o esforço para planejar tarefas, já que cada atividade do processo foi representada por uma raia no quadro de tarefas, demandando esforço majoritariamente físico.
EXT (↓)	Além do quadro de tarefas, houve a necessidade de se estimar o tempo para realização de cada projeto. Essa estimativa é uma informação obrigatório em cada Plano de Teste. Portanto, considerando o momento do projeto, tal estimativa foi realizada com base na experiência do testador, considerando seu conhecimento acumulado.

Legenda: NWS = Número de Web Services; TPT = Tarefas do processo de teste; EXT = Experiência com teste.

A tarefa "Monitorar Marco" está presente ao final de diversas atividades desse processo. Seu objetivo é verificar se todas as tarefas de cada atividade foram efetivamente cumpridas, bem como atualizar o *status* do projeto. A verificação do cumprimento de cada tarefa foi apoiada pela ferramenta de gestão de projetos da organização, enquanto que a atualização do *status* dos projetos foi realizada por meio da mudança de raia dos *post-its* que compõem o quadro de tarefas, conforme ilustra a Figura 14.

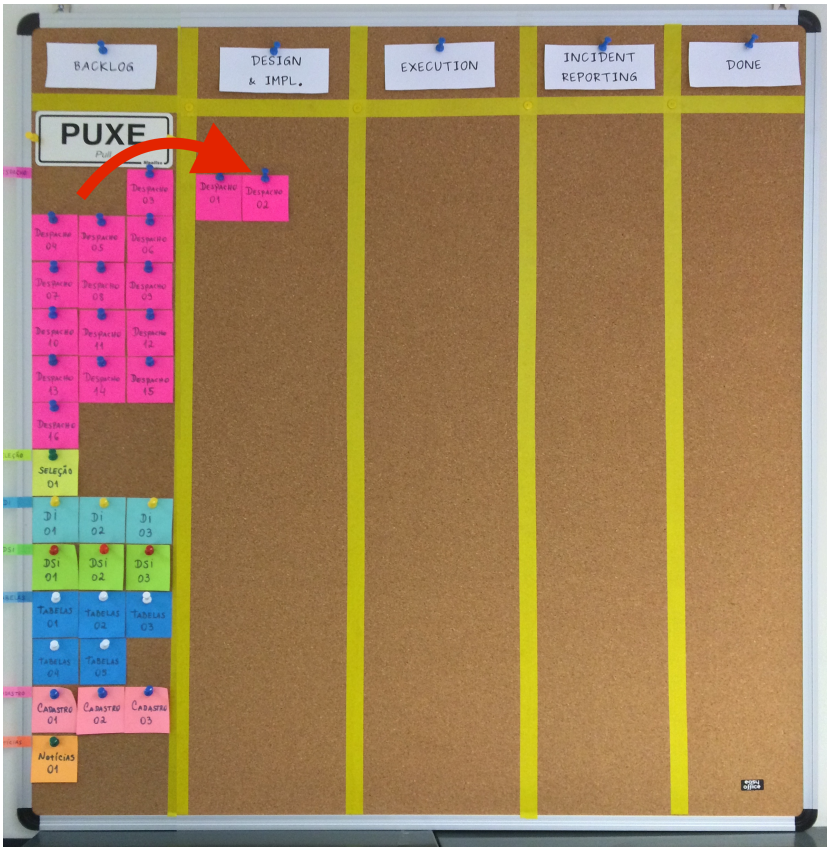


Figura 14: Atualização do quadro de tarefas.

A Tabela 21 descreve o *rationale* usado nesta tarefa. Esta explicação não será repetida para as demais atividades, tendo em vista que a tarefa oferece fatores de esforço comuns independentemente da atividade.

Tabela 21: *Rationale* sobre os fatores observados na tarefa "*Monitorar Marco*".

Fator	*Rationale*
NWS (↑)	Tanto no projeto Alfa quanto no projeto Beta, a atualização do *status* (tarefa atual) de cada serviço no quadro de tarefas demanda um esforço predominantemente físico, por mínimo que seja, de passar um *post-it* de uma raia para outra. Além disso, sempre que os testes de cada serviço terminavam, era necessário comunicar tal fato às equipes envolvidas, através da ferramenta de gestão do projeto. Portanto, a quantidade de serviços implica em aumento do esforço para monitorar os marcos do projeto.
TPT (↑)	O esforço para monitorar cada marco foi positivamente influenciado pela quantidade de tarefas que compõem cada atividade do processo. Por exemplo, a atividade "Configurar Ambiente de Teste", por ser composta por apenas uma outra tarefa, requer menos esforço para monitorar o marco do que a atividade "Projetar e Implementar Testes", que é composta por outras quatro tarefas.
STT (↓)	O esforço para monitorar cada marco foi negativamente influenciado pelo suporte de ferramentas de teste, já que a ferramenta utilizada para gestão do projeto fornece uma série de facilidades para acompanhar a execução dos projetos.

Legenda: NWS = Número de Web Services; TPT = Tarefas do processo de teste; STT = Suporte de ferramentas de teste.

4.5.2 Atividade "Configurar Ambiente de Teste"

A atividade "Configurar Ambiente de Teste" é realizada com o objetivo de se estabelecer um ambiente isolado para a realização de testes, incluindo a instalação e configuração de servidores, a carga de bancos de dados e a configuração de ferramentas de suporte. A Figura 15 representa as tarefas e fatores de esforço identificados nesta atividades. Tais fatores de esforço foram observados apenas no projeto Alfa, tendo em vista que a realização desta atividade não foi necessária no projeto Beta, pois já existia um ambiente de teste estabelecido antes de iniciar o projeto de testes. A Tabela 22 detalha como esses fatores foram observados.

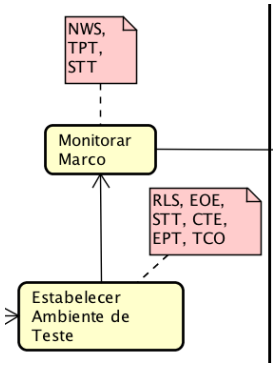


Figura 15: Tarefas e fatores de esforço da atividade Configurar Ambiente de Teste.

Tabela 22: *Rationale* sobre os fatores observados na tarefa "*Estabelecer Ambiente de Teste*".

Fator	*Rationale*
RLS (↑)	Para acessar os serviços do projeto Alfa era necessário realizar autenticação. Desta forma, houve um esforço no sentido de configurar o ambiente de teste de modo que o processo de autenticação fosse feito de modo automático, a partir da leitura de um arquivo de senhas. Portanto, este fator implicou em aumento do esforço para estabelecer o ambiente de teste.
EOE (↓)	O ambiente operacional do sistema sob teste era baseado em um conjunto de ferramentas nunca antes utilizadas pelo testador. Sendo assim, houve a necessidade de recorrer a manuais para realizar a configuração dessas ferramentas, o que implicou em aumento do esforço. Portanto, se o testador tivesse mais experiência com o ambiente operacional, o esforço inerente a esta tarefa seria menor.
STT (↓)	O suporte de ferramentas de teste resultou em diminuição do esforço nesta tarefa, tendo em vista que determinadas tarefas repetitivas de configuração tinham seu esforço atenuado com as ferramentas disponíveis.
CTE (↑)	O esforço relacionado à complexidade de configuração do ambiente de teste foi positivamente influenciado, em especial, quantidade de plataformas distintas: parte dos serviços estava publicada no ambiente TIBCO e outra parte em um ambiente WebSphere, o que exigiu esforço dobrado para configuração do ambiente de teste.
EPT (↓)	Parte desta tarefa envolveu a configuração da ferramenta Eclipse, usada para programar os serviços que compõem o SUT e com a qual o testador possui bastante familiaridade. Assim, o conhecimento prévio dessa ferramenta implicou em diminuição do esforço.
TCO (↓)	Além de consultar manuais, o testador recorreu a um membro da equipe de desenvolvimento para auxiliá-lo na configuração do ambiente de teste. Desta forma, a cooperação da equipe afetou negativamente o esforço.

Legenda: RLS = Nível de segurança requerido; EOE = Experiência com o ambiente operacional; STT = Suporte de ferramentas de teste; CTE = Complexidade de configuração do ambiente de teste; EPT = Experiência com a tecnologia de programação; TCO = Cooperação da equipe.

4.5.3 *Atividade "Projetar e Implementar Testes"*

Dentre as atividades que compõem o processo de teste sob análise, provavelmente, a atividade "Projetar e Implementar Testes" é aquela que requer mais esforço, por se tratar de uma atividade intensiva em raciocínio e aplicação de técnicas. O principal objetivo desta atividade é derivar casos e procedimentos de teste a partir das especificações dos serviços fornecidas pela equipe de desenvolvimento. Nos dois projetos analisados, a documentação de requisitos era formada por especificações de casos de uso e de regras de negócio, enquanto que a especificação técnica foi representada por arquivos descritores no formato *Web Services Description Language* (WSDL), no caso do projeto Alfa, e por documentos no padrão *Swagger* no projeto Beta. Esta atividade contou, ainda, com o apoio da ferramenta SoapUI (SMARTBEAR, 2015), utilizada para implementar casos e procedimentos de teste para Web Services. A Figura 16 representa as tarefas, artefatos e fatores de esforço presentes nesta atividade.

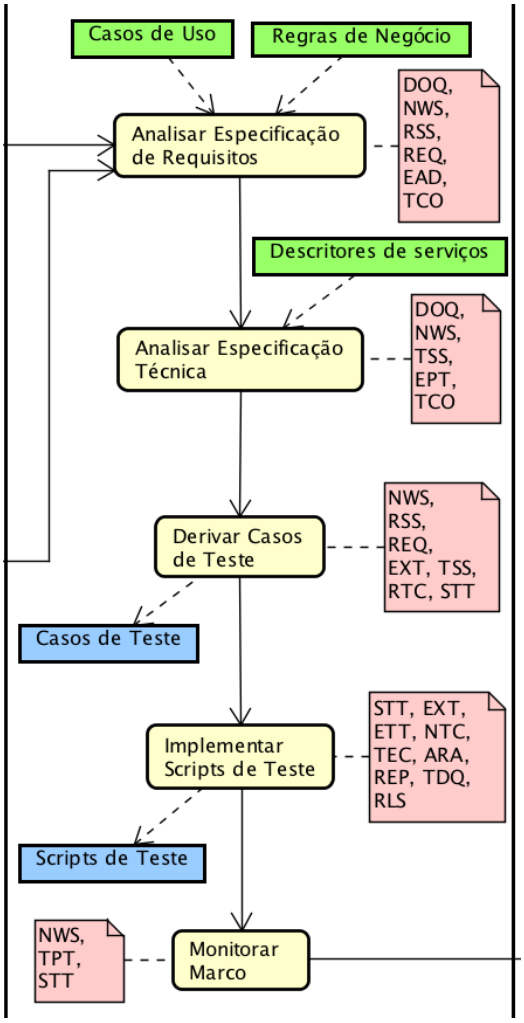


Figura 16: Tarefas, artefatos e fatores de esforço da atividade Projetar e Implementar Testes.

Esta atividade se inicia com a realização da análise da especificação de requisitos, que consiste na leitura, interpretação e compreensão das especificações de casos de uso e de regras de negócio referentes a cada serviço sob teste. A Tabela 23 descreve o *rationale* de cada fator de esforço observado nessa tarefa.

Tabela 23: *Rationale* sobre os fatores observados na tarefa "*Analisar Especificação de Requisitos*".

Fator	*Rationale*
DOQ (↓)	De forma geral, ambos os projetos possuíam especificações de requisitos coerentes e consistentes, o que favoreceu sua compreensão. Logo, o esforço para analisar as especificações de requisitos foi negativamente influenciado pela qualidade da documentação.
NWS (↑)	Dado que cada serviço sob teste em ambos os projetos era representado por um caso de uso específico, o esforço desta tarefa aumentava proporcionalmente ao número de serviços. Cada serviço representava mais um caso de uso para analisar.
RSS (↑)	Além da quantidade de casos de uso para analisar, a quantidade de informações presentes neles e nas especificações de regras de negócio influenciou positivamente o esforço em ambos os projetos – quanto maior o documento de especificação de requisitos, maior o esforço para analisá-lo.
REC (↑)	Os requisitos envolvidos em cada projeto possuíam níveis distintos de complexidade. Um requisito correspondente à uma operação de cadastro, em geral, apresentava maior complexidade em relação a um requisito de uma operação de consulta. Portanto, a complexidade dos requisitos exerceu influência positiva sobre o esforço para analisar a especificação de requisitos.
EAD (↓)	Como o domínio de aplicação do projeto Beta já era conhecido pelo testador, o esforço para compreender os requisitos sofreu efeito negativo. Já o projeto Alfa envolvia requisitos que eram desconhecidos pelo testador, o que afetou positivamente o esforço. Assim, percebeu-se que a experiência com o domínio de aplicação influencia negativamente o esforço para analisar a especificação de requisitos.
TCO (↓)	Em ambos os projetos os analistas de requisitos se mostravam solícitos sempre que havia a necessidade de dirimir dúvidas a respeito da documentação de requisitos, o que indicou que a cooperação da equipe exerceu influência negativa sobre o esforço.

Legenda: DOQ = Qualidade da documentação; NWS = Número de Web Services; RSS = Tamanho da especificação de requisitos; REC = Complexidade dos requisitos; EAD = Experiência com o domínio de aplicação; TCO = Cooperação da equipe.

Feita a análise da especificação de requisitos, a próxima tarefa é "Analisar Especificação Técnica", que consiste da leitura, interpretação e compreensão dos documentos que descrevem os serviços, os quais representam requisitos técnicos como o tipo de método HTTP utilizado por cada serviço, quais os códigos de resposta esperados em caso de sucesso e de erro, bem como o esquema da entrada de dados.

Como dito anteriormente, os serviços do projeto Alfa foram descritos através de documentos WSDL, enquanto que os serviços do projeto Beta foram descritos através de páginas *Swagger*. O primeiro formato, por se tratar de um documento XML, exigia mais esforço de compreensão do que o segundo, caracterizado por tabelas em páginas Web contendo esquemas de mensagens no formato *JavaScript Object Notation* (JSON). Esta característica foi denominada "tipo de descritor de serviço" e, assim como o "tipo de teste", foi classificado como um *moderador de esforço* e não como um *fator de*

esforço. A Figura 17 mostra um exemplo de especificação técnica baseada em *Swagger*. Em seguida, a Tabela 24 descreve o *rationale* de cada fator de esforço observado.

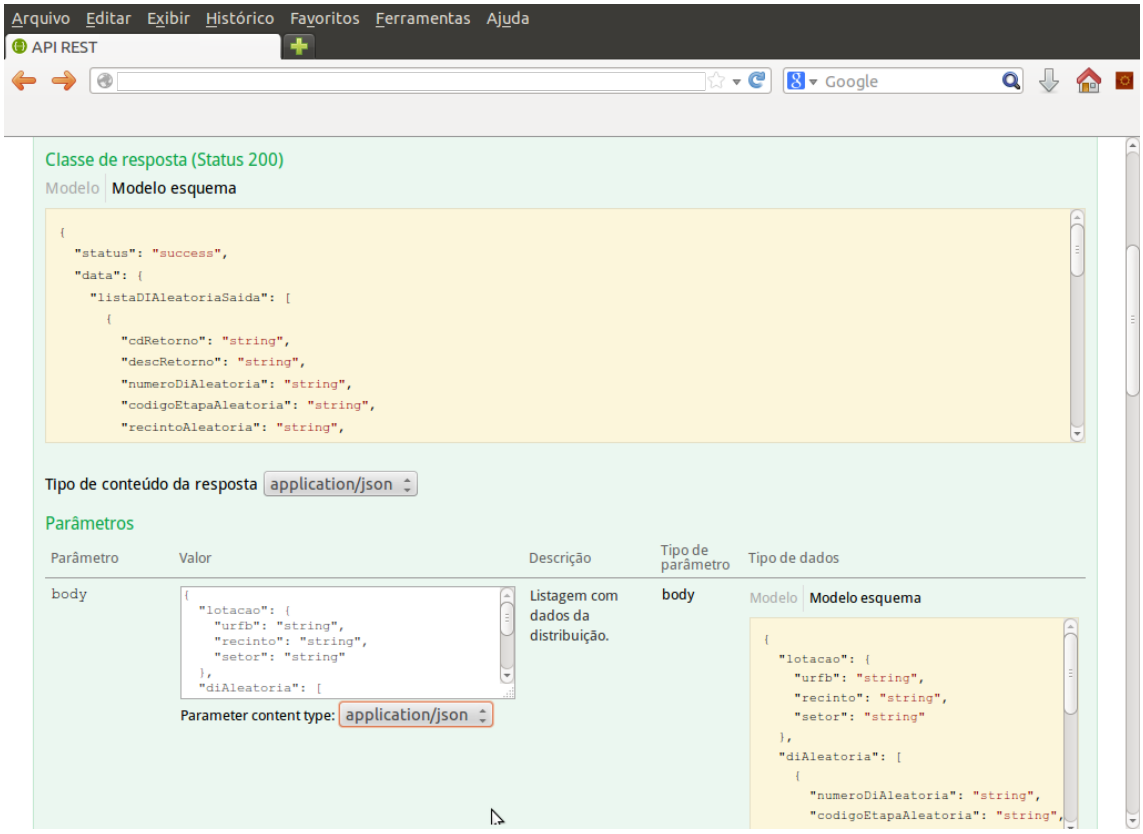


Figura 17: Exemplo de especificação técnica em *Swagger*.

Tabela 24: *Rationale* sobre os fatores observados na tarefa "*Analisar Especificação Técnica*".

Fator	*Rationale*
DOQ (↓)	Diferentemente da especificação de requisitos, as especificações técnicas de ambos os projetos possuíam inúmeros problemas, provavelmente porque foram geradas automaticamente a partir de um código-fonte com defeitos de implementação. No projeto Beta, em especial, frequentemente era necessário pedir ajuda a algum desenvolvedor para compreender o que a especificação tentava representar. Assim sendo, observou-se que a qualidade da documentação técnica afeta negativamente o esforço.
NWS (↑)	Cada serviço sob teste possui um documento técnico que o descreve. Assim, quanto mais serviços a serem testados, mais especificações técnicas devem ser analisadas. Portanto, a quantidade de serviços exerce influência positiva sobre o esforço.
TSS (↑)	Eventualmente, se juntadas as especificações técnicas de dois ou três serviços ainda seriam menores que a especificação técnica de outro serviço que, por representar mais informações exigia mais esforço de análise. Desta forma, observou-se em ambos os projetos que o tamanho da especificação técnica afetou positivamente o esforço desta tarefa.
EPT (↓)	O esforço para analisar a especificação técnica foi atenuado pelo fato de o testador possuir experiência de mais de 10 anos com desenvolvimento de Web Services. Deste modo, a experiência do testador contribuiu negativamente com o esforço de analisar a especificação técnica dos serviços.
TCO (↓)	Como já citado para o fator DOQ, no projeto Beta foi necessário pedir ajuda aos desenvolvedores para compreender as especificações técnicas. Por isso, observou-se que o grau de cooperação da equipe afeta negativamente o esforço.

Legenda: DOQ = Qualidade da documentação; NWS = Número de Web Services; TSS = Tamanho da especificação técnica; EPT = Experiência com a tecnologia de programação; TCO = Cooperação da equipe.

Depois que as especificações de requisitos e técnica foram analisadas, partiu-se para a tarefa "Derivar Casos de Teste". Esta tarefa tem por objetivo selecionar os casos de teste para cada serviço a ser testado, com a definição dos dados de entrada e seus respectivos resultados esperados. A Tabela 25 apresenta o *rationale* referente a esta tarefa.

Tabela 25: *Rationale* sobre os fatores observados na tarefa "*Derivar Casos de Teste*".

Fator	*Rationale*
NWS (↑)	Cada serviço a ser testado exige no mínimo a definição de um caso de teste, independentemente de seus requisitos ou regras de negócio. Desta forma, a quantidade de serviços exerceu influência positiva sobre o esforço para derivar casos de teste.
RSS (↑)	Para cada serviço havia uma especificação de requisitos, materializada na forma de casos de uso e regras de negócio. Observou-se que quanto maior a especificação de requisitos (em quantidade de informações), mais esforço era necessário para derivar casos de teste, sugerindo que o tamanho dessas especificações influenciava positivamente o esforço desta tarefa.
REC (↑)	Ainda em relação aos requisitos, apenas a quantidade de informações presentes nas especificações de requisitos não é suficiente para determinar o esforço de derivar casos de teste. A complexidade que envolve esses requisitos também afeta positivamente o esforço, considerando que quanto mais complexo for o requisito, maior será a dificuldade para o testador compreendê-lo e derivar casos de teste.
EXT (↓)	A experiência com testes exerceu influência negativa sobre o esforço, dado que o testador aplicou técnicas de teste de seu conhecimento prévio, já utilizadas em outros projetos.
TSS (↑)	O tamanho da especificação técnica também exerceu influência positiva sobre o esforço. Quanto maior for a especificação técnica maior será o esforço para derivar casos de teste, dado que o tamanho dessa especificação pode indicar a quantidade de condições de teste a serem cobertas pelos testes.
RTC (↑)	O nível de cobertura requerido em ambos os projetos foi de 100% das regras de negócio e das restrições tecnológicas relacionadas às respostas dos serviços, tais como códigos de resposta HTTP, valores contidos no header "*Content-Type*" e o formato da resposta. Portanto, a cobertura de testes exerceu uma influência positiva sobre o esforço, dado que se implicou em aumento do volume de trabalho.
STT (↓)	Esta tarefa foi apoiada pela ferramenta *SoapUI* e por outras ferramentas complementares, o que permitiu que os casos de teste fossem derivados e registrados com certa facilidade. Sendo assim, o suporte de ferramentas de teste exerceu influência negativa sobre o esforço.

Legenda: NWS = Número de Web Services; RSS = Tamanho da especificação de requisitos; REC = Complexidade dos requisitos; EXT = Experiência com teste; TSS = Tamanho da especificação técnica; RTC = Cobertura de testes requerida; STT = Suporte de ferramentas de teste.

Com os casos de teste definidos, realizou-se a implementação de procedimentos de teste automatizados (*scripts* de teste) utilizando a ferramenta *SoapUI*. Na sua grande maioria, tais *scripts* de teste eram formados por uma única requisição HTTP, composta por uma série de asserções implementadas com a linguagem *Groovy*, para avaliar se determinadas condições de teste eram satisfeitas pelos serviços. A Figura 18 apresenta um exemplo de *script* de teste instrumentado por asserções. Em seguida, a Tabela 26 descreve o *rationale* sobre os fatores observados nesta tarefa.

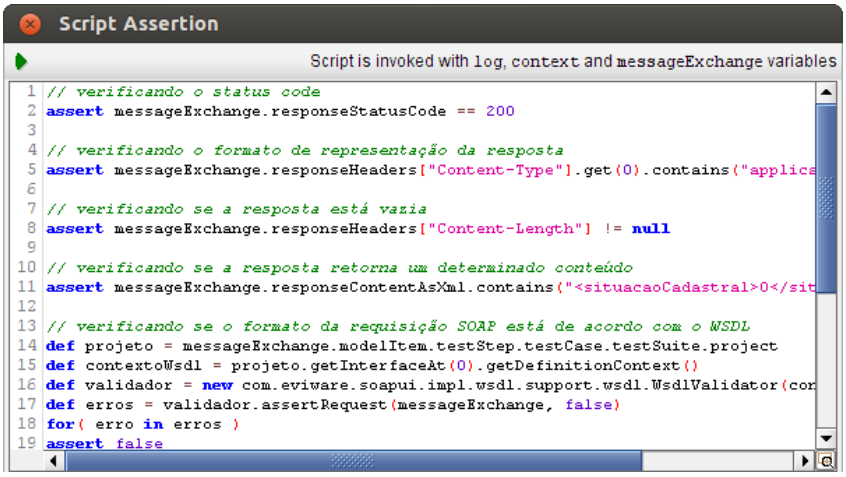


Figura 18: Exemplo de *script* de teste do *SoapUI* com asserções em *Groovy*.

Tabela 26: *Rationale* sobre os fatores observados na tarefa "*Implementar Scripts de Teste*".

Fator	*Rationale*
STT (↓)	O suporte ferramental exerce papel preponderante na tarefa de implementação de *scripts* de teste. O *SoapUI* possui recursos que facilitam o trabalho de implementação dos *scripts*, tais como recursos "*drag and drop*". Além disso, o uso dessa ferramenta permitiu a reutilização de *scripts* de teste, possibilitando que fossem implementados com menos esforço e em menor tempo. Desta forma, o suporte de ferramentas de teste influenciou negativamente o esforço.
EXT (↓)	A experiência com testes também exerceu efeito negativo sobre o esforço, tendo em vista que, à medida que o testador testava mais serviços, sua habilidade para implementar *scripts* também aumentava, diminuindo o esforço necessário a esta tarefa.
ETT (↓)	A experiência com ferramentas de teste exerceu efeito negativo sobre o esforço, uma vez que o testador conhecia profundamente a ferramenta *SoapUI*, tendo a utilizado em vários projetos.
NTC (↑)	Cada caso de teste exige a criação de pelo menos um *script* de teste. Deste modo, o esforço para implementar *scripts* de teste cresce proporcionalmente ao número de casos de teste.
TEC (↑)	Em ambos os projetos observados havia serviços que exigiam a seleção de uma nova massa de teste a cada vez que eram executados. Desta forma, os *scripts* de teste precisaram ser adaptados para buscar por dados de teste em uma base de dados, o que afetou positivamente o esforço para implementá-los.
ARA (↓)	Os *scripts* de teste construídos em ambos os projetos possuíam partes comuns, o que permitiu a criação de um modelo de *scripts* genérico. Logo, todo *script* de teste foi produzido com base nesse modelo, o que afetou negativamente o esforço.
REP (↑)	Ambos os projetos possuíam restrições de tempo de execução para cada serviço. Assim sendo, foi necessário definir asserções de tempo de resposta nos *scripts* de teste, aumentando o esforço para implementá-los.
TDQ (↑)	O esforço para implementação dos scripts de teste é proporcional à quantidade de dados de entrada e verificações que devem ser feitas sobre a mensagem de resposta. Desta forma, observou-se, especialmente nos serviços que representavam operações de cadastro, que quanto mais dados de teste a serem submetidos, maior será o esforço para implementar os *scripts* de teste.
RLS (↑)	Os serviços de ambos os projetos requeriam autenticação de usuários. Para os *scripts* do projeto Alfa, foi criado um arquivo de senhas para ser lido pelas requisições que compunham os *scripts* a cada execução de testes. No projeto Beta, todas as chamadas aos serviços continham um *header* com *login* e senha definidos estaticamente. Desta forma, o nível de segurança requerido aumentou o esforço para implementar os *scripts* de teste.

Legenda: STT = Suporte de ferramentas de teste; EXT = Experiência com teste; ETT = Experiência com ferramentas de teste; NTC = Número de casos de teste; TEC = Complexidade de execução de teste; ARA = Disponibilidade de artefatos de teste reusáveis; REP = Nível de desempenho requerido; TDQ = Quantidade de dados de teste; RLS = Nível requerido de segurança.

4.5.4 Atividade "Executar Testes"

Na atividade "Executar Testes", os testes são executados e os seus resultados são comparados e registrados para posterior avaliação. Trata-se da atividade que possibilita a identificação de falhas no software sob teste. A Figura 19 representa as tarefas, artefatos e fatores de esforço relacionados à esta atividade.

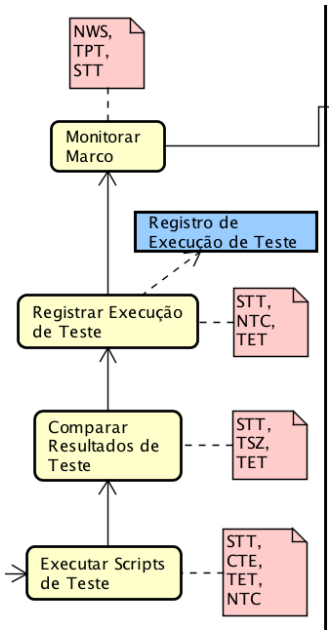


Figura 19: Tarefas, artefatos e fatores de esforço da atividade Executar Testes.

Levando-se em conta que os testes foram automatizados na forma de *scripts*, a atividade "Executar Testes" como um todo foi simplificada em ambos os projetos, exigindo apenas o acionamento da execução dos testes na ferramenta *SoapUI*. Desta forma a tarefa "Executar *Scripts* de Teste" exigiu mais esforço físico do que mental, determinado especialmente pela quantidade de execuções necessárias a cada *script* de teste. A Tabela 27, portanto, apresenta o *rationale* sobre os fatores desta tarefa.

Tabela 27: *Rationale* sobre os fatores observados na tarefa "*Executar Scripts de Teste*".

Fator	*Rationale*
STT (↓)	O suporte ferramental é imprescindível para executar automaticamente os *scripts* de teste. Quanto mais automatizada por ferramentas for a atividade de execução de testes, menos esforço será necessário para executar os *scripts* de teste.
CTE (↑)	Determinados serviços do projeto Alfa exigiam a configuração manual do ambiente de teste a cada execução de teste, aumentando o esforço para executar os *scripts* de teste.
TET (↑)	A quantidade de *trials* determina a quantidade de execuções dos *scripts* de teste. Mesmo contando com *scripts* de teste automatizados, sua execução exige acionamento manual, o que afeta positivamente o esforço de execução de testes.
NTC (↑)	A quantidade de casos de teste determinou, em ambos os projetos, a quantidade de *scripts* de teste. Como cada *script* de teste deve ser executado pelo menos uma vez, exigindo a intervenção manual do testador, o número de casos de teste afetou positivamente o esforço de execução de *scripts* de teste.

Legenda: STT = Suporte de ferramentas de teste; CTE = Complexidade do ambiente de teste; TET = Ciclos de teste; NTC = Número de casos de teste.

O fato de os testes terem sido automatizados também contribuiu para que a tarefa "Comparar Resultados de Teste" tivesse seu esforço amenizado. Os *scripts* de teste tiveram não apenas as chamadas dos serviços automatizadas, mas também verificação dos resultados esperados frente às mensagens de resposta. Os *scripts* de teste executados com sucesso são apresentados pelo *SoapUI* em verde; já aqueles *scripts* que apresentam alguma falha são apresentados em vermelho. Essa característica foi implementada através de asserções nos próprios *scripts* de teste. A Figura 20 representa o resultado da execução de determinados *scripts* de teste. O *rationale* a respeito dos fatores desta tarefa é apresentado na Tabela 28.

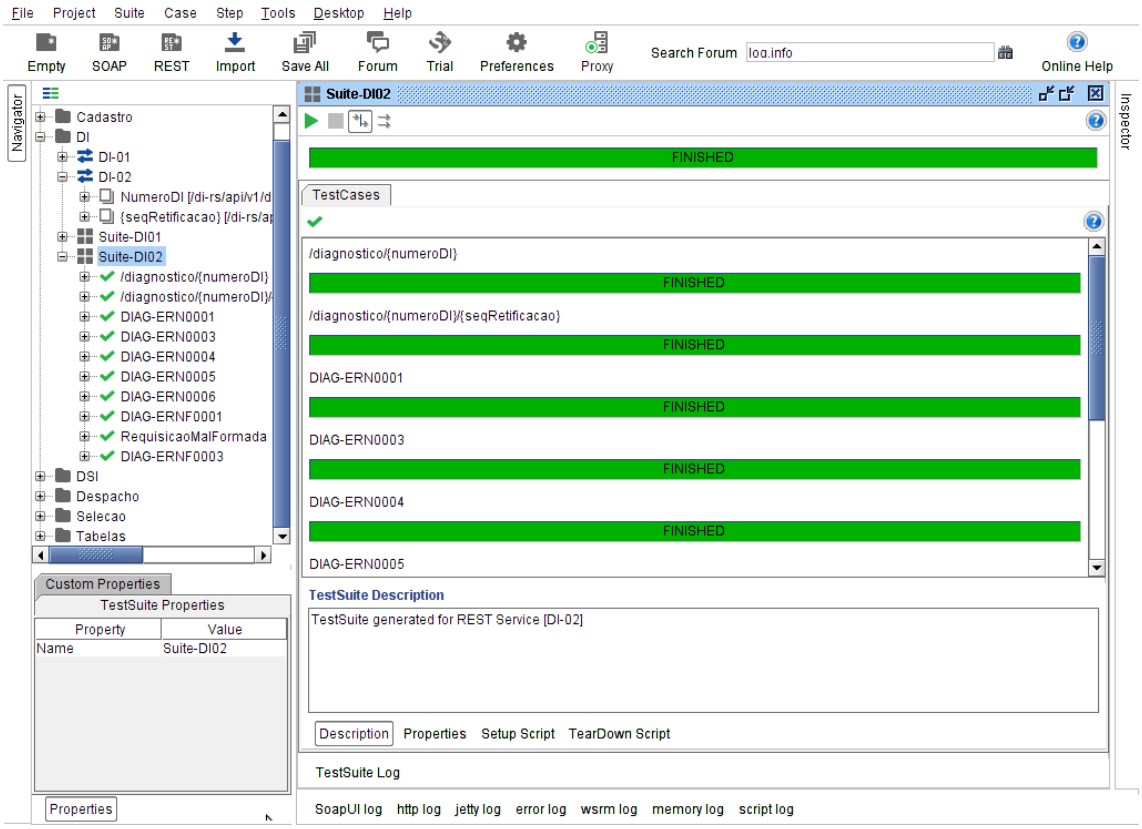


Figura 20: Perspectiva da ferramenta *SoapUI* após a execução de *scripts* de teste.

Tabela 28: *Rationale* sobre os fatores observados na tarefa "*Comparar Resultados de Teste*".

Fator	*Rationale*
STT (↓)	A criação das asserções que comparam automaticamente os resultados observados com os resultados esperados só é possível com a utilização de ferramentas. Assim sendo, o suporte ferramental diminuiu o esforço inerente à esta tarefa.
TSZ (↑)	Apesar de os *scripts* de teste estarem instrumentados para comparar os resultados automaticamente, o testador costumava eventualmente realizar inspeções visuais sobre as mensagens de resposta, tomando como base as asserções definidas nos *scripts* de teste. Desta forma, quanto mais asserções presentes no *script* de teste, maior o esforço para comparar os resultados de teste.
TET (↑)	Esta tarefa é realizada toda vez que um script de teste é executado. Portanto, quanto mais *trials* de teste forem realizados, maior será o esforço inerente à esta tarefa.

Legenda: STT = Suporte de ferramentas de teste; TSZ = Tamanho dos *scripts* de teste; TET = Ciclos de teste.

O processo de teste da organização preconiza que toda e qualquer execução de teste deve ser registrada na ferramenta de gestão de projetos, indicando se o teste passou ou falhou, bem como data e hora de execução. Sendo assim, os fatores de esforço observados na tarefa "Registrar Execução de Teste" são justificados na Tabela 29.

Tabela 29: *Rationale* sobre os fatores observados na tarefa "*Registrar Execução de Teste*".

Fator	*Rationale*
STT (↓)	Para ambos os projetos, a ferramenta usada para executar os testes estava integrada à ferramenta de gestão do projeto, na qual cada execução deve ser registrada. Desta forma, ao executar cada *script* de teste era produzido automaticamente um registro de execução na ferramenta de gestão. Desta forma, o alto nível de suporte ferramental influenciou negativamente o esforço desta tarefa.
NTC (↑)	Como cada caso de teste corresponde a no mínimo um procedimento de teste, e cada procedimento de teste é executado pelo menos uma vez, resultando em um registro de execução, o número de casos de teste afeta positivamente o esforço para registrar a execução do teste.
TET (↑)	A quantidade de registros de execução de testes é diretamente proporcional à quantidade de ciclos de teste – a cada execução um novo registro deve ser criado, aumentando o esforço.

Legenda: STT = Suporte de ferramentas de teste; NTC = Número de casos de teste; TET = Ciclos de teste.

4.5.5 Atividade "Comunicar Incidentes de Teste"

A finalidade da atividade "Comunicar Incidentes de Teste" é relatar às partes interessadas os incidentes observados durante a execução de testes. Cada incidente deve ser devidamente registrado em um módulo de *bugtracking* que faz parte da ferramenta de gestão de projetos utilizada na empresa XPTO. A Figura 21 representa as tarefas, artefatos e fatores de esforço presentes nesta atividade.

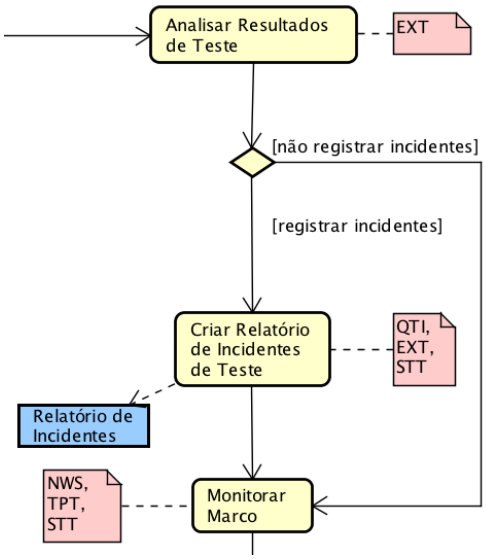


Figura 21: Tarefas, artefatos e fatores de esforço da atividade Comunicar Incidentes de Teste.

Como indica a Figura 21, a primeira tarefa da atividade de comunicação de incidentes de teste é "Analisar Resultados de Teste". Esta tarefa consiste em avaliar se os incidentes percebidos durante a execução dos testes, como a ocorrência de falhas, devem ou não ser registrados como novos relatos de incidentes. A justificativa para a identificação do único fator de esforço observado nesta tarefa é apresentada na Tabela 30.

Tabela 30: *Rationale* sobre os fatores observados na tarefa "*Analisar Resultados de Teste*".

Fator	*Rationale*
EXT (↓)	A experiência com teste é fundamental para analisar o resultados de teste, de modo que quanto mais experiente for o testador, mais facilidade e, consequentemente, menor esforço ele terá para analisar os resultados dos registros de execução.

Legenda: EXT = Experiência com teste.

Uma vez definidos os incidentes a serem reportados, o testador deve registrá-los no módulo de *bugtracking*, atribuindo-os ao líder de desenvolvimento. Cada incidente deve ser registrado na forma de um relato de incidente, que é uma descrição textual do incidente acompanhada por uma evidência em imagem ou vídeo. O conjunto de relatos de incidentes formam o Relatório de Incidentes de Teste. A Tabela 31 descreve o *rationale* utilizado para identificar os fatores de esforço desta tarefa.

Tabela 31: *Rationale* sobre os fatores observados na tarefa "*Criar Relatório de Incidentes de Teste*".

Fator	*Rationale*
QTI (↑)	O Relatório de Incidentes de Teste é composto por relatos de incidentes. Cada relato de incidente deve ser descrito textualmente e evidenciado por meio de imagens ou vídeos de modo que possa ser reproduzido pelos desenvolvedores. Portanto, quanto mais incidentes relatados, maior o esforço aplicado.
EXT (↓)	Um testador experiente, já habituado a relatar incidentes, tende a aplicar menos esforço para relatá-los do que um testador inexperiente, já que não precisa raciocinar tanto para descrever um incidente e, provavelmente, já possui um estilo de escrita definido para relatar incidentes. Assim, a experiência com teste atua negativamente sobre o esforço desta tarefa.
STT (↓)	Cada incidente relatado necessita de evidências em imagem ou vídeo. Em alguns casos torna-se necessário editar imagens ou vídeos para indicar adequadamente o incidente ocorrido. Desta forma, ferramentas de edição de imagens e vídeos diminuem o esforço do testador nesta tarefa, já que possuem recursos que facilitam a edição das evidências de teste.

Legenda: QTI = Quantidade de incidentes de teste; EXT = Experiência com teste; STT = Suporte de ferramentas de teste.

4.5.6 Atividade "Finalizar Projeto de Teste"

A atividade "Finalizar Projeto de Teste" contempla a limpeza do ambiente de teste e o registro formal de encerramento do projeto, através de um Relatório de Conclusão do Projeto de Teste, conforme mostra a Figura 22.

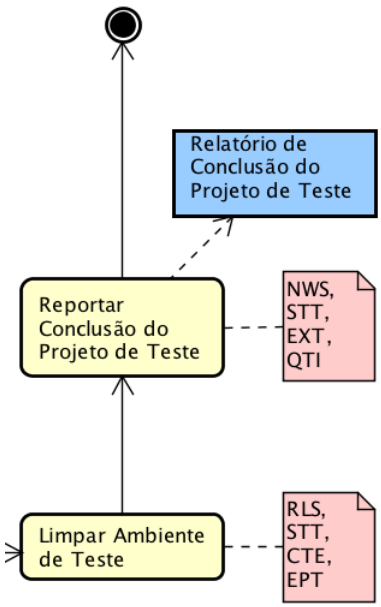


Figura 22: Tarefas, artefatos e fatores de esforço da atividade Finalizar Projeto de Teste.

A limpeza do ambiente de teste consiste em liberar os recursos de hardware e software que estavam configurados para a realização dos testes, incluindo a remoção de ferramentas eventualmente instaladas e a remoção de dados de teste. A Tabela 32 descreve o *rationale* desta tarefa. Como apenas o projeto Alfa exigiu do testador a configuração de um ambiente de teste, todo o *rationale* é baseado nas observações daquele projeto.

Tabela 32: *Rationale* sobre os fatores observados na tarefa "*Limpar Ambiente de Teste*".

Fator	*Rationale*
RLS (↑)	Os serviços do projeto Alfa exigiam autenticação. Para tal, foi criado um arquivo de senhas que era consumido a cada execução de testes. Por questões de confidencialidade, esse arquivo não poderia permanecer no ambiente de teste. Ao limpar o ambiente de teste foi necessário excluir esse arquivo, o que aumentou o esforço nesta tarefa.
STT (↓)	A limpeza do ambiente de teste foi apoiada pelas próprias ferramentas nele instaladas, facilitando o trabalho do testador, o que afetou negativamente o esforço.
CTE (↑)	A complexidade do ambiente de teste exerceu influência positiva sobre o esforço desta tarefa, uma vez que uma das plataformas onde os serviços estavam sendo executados (ambiente TIBCO) exigiu a remoção e reconfiguração de diversos arquivos para proceder a limpeza do ambiente de teste.
EPT (↓)	Para proceder a limpeza do ambiente de teste foi necessário manipular ferramentas utilizadas pela equipe de desenvolvimento para construir e disponibilizar os serviços. Essas ferramentas já haviam sido utilizadas no momento do estabelecimento do ambiente de teste. Quando foi limpar o ambiente de teste, o testador já possuía conhecimento prévio dessas ferramentas, o que influenciou negativamente o esforço.

Legenda: RLS = Nível de segurança requerido; STT = Suporte de ferramentas de teste; CTE = Complexidade de configuração do ambiente de teste; EPT = Experiência com a tecnologia de programação.

A última tarefa do processo contempla a elaboração do "Relatório de Conclusão do Projeto de Teste" (RCPT), um artefato no qual se registram as lições aprendidas e os

demais resultados do projeto de teste. A Tabela 33 descreve a justificativa a respeito de cada fator de esforço observado.

Tabela 33: *Rationale* sobre os fatores observados na tarefa "*Reportar Conclusão do Projeto de Teste*".

Fator	*Rationale*
NWS (↑)	No RCPT há uma seção na qual devem ser listadas todas as funcionalidades testadas no projeto. Como cada serviço testado em ambos os projetos implementa uma funcionalidade específica, o número de Web Services exerceu influência positiva sobre o esforço, uma vez que quanto mais serviços testados, maior será a lista de funcionalidades nesta seção do documento.
STT (↓)	O artefato produzido nesta tarefa é editado em uma ferramenta de edição de textos e grande parte dos dados que devem estar contidos nesse documento é extraída automaticamente da ferramenta de gestão de projetos da empresa. Desta forma, o suporte de ferramentas de teste afetou negativamente o esforço desta tarefa.
EXT (↓)	O RCPT possui diversas seções nas quais os resultados do projeto são resumidos. O esforço para escrever seções como "Lições Aprendidas" é negativamente influenciado pela experiência do testador, que se já está acostumado a escrever esse tipo de informação tende a se esforçar cada vez menos a cada projeto concluído.
QTI (↑)	Uma das seções do Relatório de Conclusão do Projeto de Teste diz respeito aos incidentes de teste relatados. Se não houver qualquer incidente, a seção deve ser preenchida apenas com a expressão "Não houve relatos de incidentes de teste". Caso contrário, todos os incidentes devem ser contabilizados e classificados como "falha" ou "sugestão", exigindo o preenchimento de uma seção com lições aprendidas, o que exerce efeito positivo sobre o esforço.

Legenda: NWS = Número de Web Services; STT = Suporte de ferramentas de teste; EXT = Experiência com teste; QTI = Quantidade de incidentes de teste.

4.6 Resultados e Discussão[5]

Este estudo serviu para compreender melhor os fatores que influenciam o esforço em projetos de teste de software, em especial como a influência desses fatores variou em projetos reais na indústria. Para identificar os fatores e a influência realizou-se um *rationale* do ponto de vista do testador em cada tarefa que compõe o processo de teste da organização. Esse *rationale* pode contribuir para a construção de um eventual modelo de estimativa que tenha como base as opiniões de profissionais de teste de software.

Além de responder às questões de pesquisa propostas, este estudo serviu para:

- *Consolidar o entendimento sobre o conceito de esforço*, reforçando a ideia de que esse conceito deve levar em conta que desenvolver software é uma tarefa

[5] Uma versão desta seção foi parcialmente publicada no *position paper Reflexões sobre o Esforço em Testes de Desempenho de Produtos de Software,* apresentado no II Workshop em Qualidade de Produtos de Software (WQPS 2017) (SILVA-DE-SOUZA, 2017).

que envolve técnica, criatividade e raciocínio lógico e não pode ser confundida com tarefas repetitivas típicas da manufatura.

- *Com base nesse entendimento, identificar com mais precisão os fatores de esforço.*

Além de possibilitar a observação de 26 dos 47 fatores de esforço identificados na RSL, este estudo propiciou a identificação de outros quatro fatores nos projetos analisados. A ausência desses quatro fatores na RSL provavelmente se deve às especificidades do processo e dos projetos de teste analisados. A observação desses fatores foi fundamental para consolidar o conceito de *fator de esforço* apresentado na seção 1.1: qualquer característica do ambiente de software que exerce uma influência significativa sobre o esforço requerido para realizar alguma tarefa relacionada ao desenvolvimento de um software. No entanto, é importante salientar que outros fatores poderiam ter sido observados caso o testador e/ou o processo tivessem sido outros.

No início do levantamento dos fatores de esforço foram identificados os fatores "desempenho do hardware" e "dependência de rede", levando-se em conta que o esforço do profissional de teste é afetado quando o sistema sob teste apresenta lentidão ou é interrompido em função do mal desempenho do hardware utilizado para hospedá-lo ou de problemas relacionados à infraestrutura de conectividade. No entanto, há de se estabelecer uma importante diferença: o tempo resposta é uma medida que representa o quão performático é o produto de software. Esta medida não é apropriada para capturar o quanto de esforço é necessário para testar determinado produto. Enquanto o desempenho é uma característica intrínseca ao produto de software, o esforço é uma propriedade inerente ao ser humano que executa determinada tarefa para testar aquele produto. Assim, considerando o conceito de esforço apresentado na seção 4.1, esses fatores foram excluídos do estudo já que afetam apenas o tempo de alocação do profissional à tarefa. Enquanto o testador aguarda pelo reestabelecimento da aplicação ele não necessariamente realiza algum esforço significativo relacionado àquela tarefa. Este tempo de espera foi tratado como **desperdício** e não como esforço, já que representa ociosidade do testador. De forma análoga, os gastos com material de escritório e outros suprimentos foram classificados como **custo** e, portanto, estão fora do escopo deste trabalho. Naturalmente, desperdícios e custos de projetos devem ser monitorados e tratados por gerentes de teste para que sejam reduzidos ou minimizados.

Outro conceito inicialmente elencado como fator de esforço foi "tipo de teste", com a ideia de que o esforço depende do tipo de teste realizado. Entretanto, ao longo da análise o tipo de teste foi reclassificado como ***moderador de esforço***, isto é, para qualquer tipo de teste os fatores identificados podem exercer influência sobre o esforço, mas dependendo do tipo de teste a intensidade dessa influência pode variar. A mesma analogia pode ser feita em relação ao "tipo de sistema", característica muitas vezes citada como fator de esforço na literatura. Como o objetivo deste estudo não era quantificar a intensidade da influência, mas apenas identificar o seu sentido, não foi feita qualquer análise sobre moderadores de esforço.

- *Perceber se havia variação da influência dos fatores em função da tarefa ou do projeto* - a maior parte dos fatores pôde ser observada nos dois projetos analisados, porém não foi observado qualquer fator que apresentasse sentidos opostos de influência em ambos os projetos.

- *Observar que a forma pela qual determinados fatores de esforço são medidos não captura a sua essência*.

Tradicionalmente, a literatura técnica de Engenharia de Software sugere que a experiência de um profissional seja medida pela quantidade de tempo ou de projetos que ele tenha atuado com alguma tecnologia. No entanto, quando se procura um profissional experiente não necessariamente se quer alguém com muitos anos de experiência, e sim alguém que tenha conhecimento profundo no assunto. O tempo é apenas um *surrogate* (representante) para o nível de conhecimento, o que é mais difícil de medir. Portanto, quanto à experiência, pouco importa a extensão de tempo; o que importa é a intensidade da prática.

4.7 Ameaças à validade

Este estudo envolveu uma série de ameaças à validade, que foram assumidas pelo pesquisador para viabilizar sua realização. Dentre as principais ameaças à validade identificadas, destacam-se as seguintes:

- O pesquisador e o testador são a mesma pessoa. Essa característica por si só pode inserir um ou mais fatores de confusão, pelo possível viés que pode

introduzir. No entanto, essa característica possibilitou uma observação mais próxima ao processo e aos projetos, proporcionando um relato de quem realmente vivenciou os projetos e não apenas atuou como observador.

- O *rationale* utilizado para justificar a presença e influência de cada fator é uma evidência anedótica, mas que, por se basear em projetos reais na indústria, pode auxiliar a compreender como o esforço de teste é de fato influenciado pelas variáveis de contexto.
- O *rationale* foi desenvolvido em um contexto específico: foram observados apenas dois projetos, ambos com a mesma estratégia de teste, seguindo o mesmo processo e na perspectiva de um único profissional. Torna-se necessário, portanto, analisar projetos baseados em outros processos e estratégias de teste, acompanhando o trabalho de outros profissionais de teste, para reforçar a pertinência dos fatores observados e, eventualmente, identificar novos fatores de esforço.

4.8 Considerações Finais

Este capítulo apresentou um estudo observacional na indústria, que foi necessário para entender melhor quais fatores podem afetar o esforço em projetos de teste de software e como esses fatores afetam o esforço (aumentando ou diminuindo). Este estudo também possibilitou distinguir os conceitos de fator de esforço e moderador de esforço, melhorando a compreensão das variáveis de contexto que afetam o esforço em um projeto de teste de software. As observações e análises foram limitadas a projetos de teste de Web Services, portanto, a generalização dos resultados depende da execução de vários outros estudos.

Outro aspecto importante deste estudo foi observar se um mesmo fator poderia influenciar o esforço em direções opostas dependendo da tarefa realizada no processo. Observou-se que a direção da influência de cada fator permaneceu consistente, independentemente da tarefa. No entanto, foi apontado que os fatores que afetam o esforço em um projeto podem não influenciar o esforço em outros projetos.

Obviamente, outros fatores de esforço de teste podem ser observados em outros projetos e outras atividades do processo de teste de software. Além disso, outros estudos

devem ser conduzidos com o objetivo de identificar outros fatores e observar como eles influenciam o esforço do teste de software nos projetos.

5 SURVEY COM PROFISSIONAIS

Este capítulo, composto por sete seções, descreve um *survey* que coletou a percepção de profissionais da indústria de software a respeito do esforço em teste de software. A seção 5.1 descreve a motivação e os objetivos deste estudo. A seção 5.2 caracteriza o design experimental utilizado. A seção 5.3 descreve como o estudo foi executado. A seção 5.4 apresenta os resultados observados no estudo, em função de seis diferentes configurações de estratégias de teste. A seção 5.5 discute os resultados encontrados. A seção 5.6 descreve as ameaças à validade identificadas e os respectivos tratamentos adotados. Por fim, a seção 5.7 apresenta as considerações finais deste estudo.

5.1 Motivação e Objetivos

Ainda que o estudo de observação apresentado anteriormente tenha lançado luz sobre como diferentes fatores de esforço podem ser observados em projetos reais de teste de software, uma série de dúvidas ainda persiste, tendo em vista as limitações daquele estudo. Nele, foi apresentada a percepção de um profissional de teste de software sobre o esforço em apenas dois projetos, os quais assumiam a mesma estratégia de testes. Considerando a diversidade de estratégias de teste que podem ser aplicadas levando-se em conta o contexto de cada projeto, qualquer tentativa de generalização das observações daquele estudo é algo propenso a erros. Portanto, torna-se necessário identificar a percepção de outros profissionais de teste sobre o esforço em projetos que tenham se baseado em estratégias de teste variadas.

Neste estudo, considera-se como "estratégia de testes" a abordagem de testes planejada para um projeto específico. Ela deve definir, entre outros itens, o nível de teste, o tipo de teste, a técnica de teste e a forma de execução de testes referentes ao projeto, conceitos que foram discutidos no capítulo 2. Como cada uma dessas quatro dimensões pode assumir valores diferentes, há uma grande variedade de combinações de estratégia de teste. Cada possível combinação dessas quatro dimensões é denominada *configuração de estratégia de teste*. A Tabela 34 representa os possíveis valores, assumidos neste estudo, para cada uma das quatro dimensões que compõem uma configuração de estratégia de teste.

Tabela 34: Dimensões de uma configuração de estratégia de teste.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Unidade	Funcional	Baseada em estrutura	Manual
Integração	Não-Funcional	Baseada em especificação	Automatizada
Sistema			
Aceitação			

Teoricamente, há 32 combinações possíveis entre os valores apresentados na Tabela 34. No entanto, nem todas as combinações fazem sentido na prática. Por exemplo, é difícil imaginar um teste de aceitação, que é de responsabilidade do cliente, sendo realizado com base na estrutura do código-fonte. Também vale ressaltar que a quantidade de possíveis combinações tende ao infinito, uma vez que o tipo de teste não-funcional pode se decompor em inúmeros tipos de teste, de acordo com o tipo de requisito não-funcional (desempenho, usabilidade, segurança, etc.).

Outro aspecto importante que foi tratado no estudo de observação e que merece aprofundamento, diz respeito à percepção dos fatores de esforço de acordo com a atividade do processo de teste. O estudo de observação evidenciou que alguns fatores podem influenciar o esforço de forma diferente em cada atividade, especialmente se a análise for realizada no nível de granularidade de tarefa. Por exemplo, o fator "Nível de segurança requerido" (RLS) afetou positivamente o esforço de teste nas tarefas "Estabelecer ambiente de teste", "Implementar *scripts* de teste" e "Limpar ambiente de teste", que compõem diferentes atividades do processo de teste. Porém, os motivos apresentados são diferentes, ainda que mantenham algum tipo de relação.

Assim, é possível conjecturar que dependendo da configuração de estratégia de teste, diferentes fatores podem afetar o esforço de teste das diversas atividades do processo de teste de diferentes maneiras. Eventualmente, um mesmo fator pode influenciar o esforço em diferentes sentidos (positiva ou negativamente) em diferentes configurações de estratégia de teste e em diferentes organizações. Além disso, uma mesma atividade do processo de teste pode exigir mais esforço em uma configuração de estratégia de teste específica do que em outra.

Diante dessas conjecturas, este estudo tem como objetivo capturar a percepção de profissionais da indústria de software a respeito do esforço nas atividades do processo de teste de software. Esse objetivo pode ser representado de acordo com o paradigma GQM (BASILI; CALDIERA; ROMBACH, 1994), conforme a Tabela 35:

Tabela 35: Objetivos do *survey*.

Analisar	fatores de esforço
Com o propósito de	caracterizar
Em relação à	influência sobre o esforço de teste de software
Do ponto de vista	profissional de teste de software
No contexto de	atividades do processo de teste de software

Para alcançar o objetivo proposto para este estudo, foram definidas as seguintes questões de pesquisa, cada qual com sua respectiva métrica, a fim de mensurar as respostas:

- **QE6**: Quais atividades do processo de teste de software demandam mais esforço?
    - **QE6.1**: A ordem das atividades que demandam mais esforço varia de acordo com a configuração de estratégia de teste?
- **QE7**: Quais fatores de esforço relacionados ao processo de teste de software são percebidos pelos profissionais da área?
    - **QE7.1**: Qual o impacto de cada fator sobre o esforço em cada atividade do processo de teste de software?
    - **QE7.2**: Como cada fator pode ser observado em cada atividade do processo de teste de software?
    - **QE7.3**: Os fatores de esforço variam de acordo com a configuração de estratégia de teste?

Em relação à questão QE6, não se pretende mensurar a distribuição do esforço entre as atividades do processo de teste, mas apenas identificar quais atividades demadam mais esforço. Quanto à questão QE7.1, vale ressaltar que o objetivo não é quantificar o esforço, mas somente observar se ele aumenta (influência positiva) ou diminui (influência negativa) de acordo com o fator de esforço e a atividade do processo.

5.2 Design do Estudo

Este estudo foi estruturado na forma de um *survey* com o objetivo de coletar a percepção de profissionais da indústria de software a respeito do esforço em atividades do processo de teste de software, considerando diversas configurações de estratégia de

teste. Entende-se aqui como "processo de teste de software" o processo típico de teste de software sugerido pela norma ISO/IEC/IEEE 29119-2 (ISO/IEC/IEEE, 2013), composto pelas seguintes atividades, já descritas no estudo de observação:

- Planejar Testes
- Configurar Ambiente de Testes
- Projetar e Implementar Testes
- Executar Testes
- Comunicar Incidentes de Teste
- Finalizar Projeto de Testes

5.2.1 Configurações de estratégia de teste cobertas no estudo

Este estudo cobre seis possíveis combinações de nível, tipo, técnica e forma de execução de teste, representadas na Tabela 36. Essas configurações foram escolhidas pelo pesquisador considerando a sua relevância para a prática de teste de software, bem como a disponibilidade de profissionais dentro da sua rede de contatos profissionais que poderiam responder a respeito de cada uma delas.

Tabela 36: Configurações de estratégia de teste cobertas no estudo.

Configuração	Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
0	Unidade	Funcional	Baseada em estrutura	Automatizada
1	Unidade	Funcional	Baseada em especificação	Automatizada
2	Unidade	Desempenho	Baseada em especificação	Automatizada
3	Integração	Funcional	Baseada em especificação	Automatizada
4	Sistema	Funcional	Baseada em especificação	Manual
5	Sistema	Funcional	Baseada em especificação	Automatizada

A configuração "0" é aquela em que programadores desenvolvem *scripts* de teste para avaliar o comportamento funcional das unidades de implementação (tipicamente, classes e métodos). Nesta configuração, o programador projeta e implementa os casos de teste tendo como base o código-fonte do SUT. Em geral, os casos de teste são implementados na mesma linguagem de programação pela qual foi desenvolvido o SUT. A execução dos testes é apoiada por algum *framework* de testes de unidade, conhecidos genericamente como XUnit.

A configuração "1" possui um significado específico neste estudo. Trata-se de uma configuração onde testadores desenvolvem *scripts* de teste para avaliar o

comportamento funcional de *web services* (em geral, *web services* RESTful). Muitos autores e profissionais da indústria associam o teste de unidade ao teste baseado em estrutura ("caixa branca"). No entanto, no caso de *web services*, assim como de componentes de software, percebe-se que nem todo teste de unidade é conduzido de modo "caixa branca". A dimensão "técnica de teste" leva em conta o insumo utilizado para a definição dos casos de teste: a estrutura interna (código-fonte) ou alguma especificação. Mas além disso, a técnica de teste também diz respeito à superfície sobre a qual os testes são executados: código-fonte, interface de serviços ou interface gráfica. Assim, testes executados sobre o código-fonte são denominados "caixa branca", enquanto que os testes realizados sobre a interface de serviços ou interface gráfica são classificados como "teste caixa preta". Desta forma, assumindo uma perspectiva "caixa preta", *web services* e componentes são as menores unidades de software reconhecidas por um testador, mesmo que internamente esses itens de software sejam compostos por unidades ainda menores. Portanto, neste estudo, o conceito de "unidade" referente ao teste de unidade depende do observador: tratando-se de um teste "caixa preta" de um serviço ou componente não é possível para um testador vislumbrar o que há na estrutura interna daquele item de software. Nesta configuração, o testador projeta e implementa os casos de teste tendo como base uma especificação. Neste caso, geralmente o testador conta com uma especificação dos requisitos em linguagem natural e com uma especificação técnica de cada serviço sob teste, descrita geralmente nos formatos Swagger ou WSDL. Os *scripts* de teste são tipicamente implementados e executados com apoio de alguma ferramenta de testes que permite a criação de requisições HTTP. A Figura 23 representa um caso de teste de um serviço REST fictício de registro de entrada de veículos em um estacionamento, no qual alguns parâmetros de entrada (<<INPUT>>) são submetidos ao serviço por meio de uma requisição HTTP, para os quais é esperada uma resposta específica (<<OUTPUT>>).

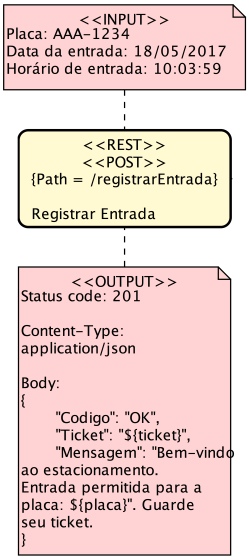


Figura 23: Exemplo de caso de teste referente à configuração 1.

A configuração de estratégia de teste "2" posssui características muito próximas à configuração "1". Os testes também são realizados sobre serviços, com base em especificação e com execução automatizada. A diferença básica está no tipo de teste. Nesta configuração são realizados testes de desempenho de serviços. Neste caso, os testadores tipicamente projetam seus casos de teste com base em requisitos de desempenho, como tempo de resposta e capacidade de acessos concorrentes, e implementam e executam os *scripts* de teste correspondentes usando alguma ferramenta específica para esse tipo de teste. Dentre as configurações de estratégia de teste previstas no estudo, esta é a única cujo tipo de teste é não-funcional.

A configuração "3" também possui elementos semelhantes à configuração "1", pelo fato de ambas se basearem em alguma especificação para avaliar o comportamento de serviços. No entanto, diferentemente da configuração "1", a configuração "3" não avalia cada serviço como uma unidade independente, mas avalia a integração de serviços quando estes compõem uma orquestração. Trata-se, portanto, de um teste de integração, cujas unidades são os serviços. Presume-se que, neste caso, os serviços já tenham sido submetidos a testes de unidade, conforme a configuração "1". Esta configuração também depende de ferramentas para apoiar a implementação e execução dos *scripts* de teste. A Figura 24 representa um caso de teste que envolve três serviços REST orquestrados para automatizar um processo de um estacionamento fictício.

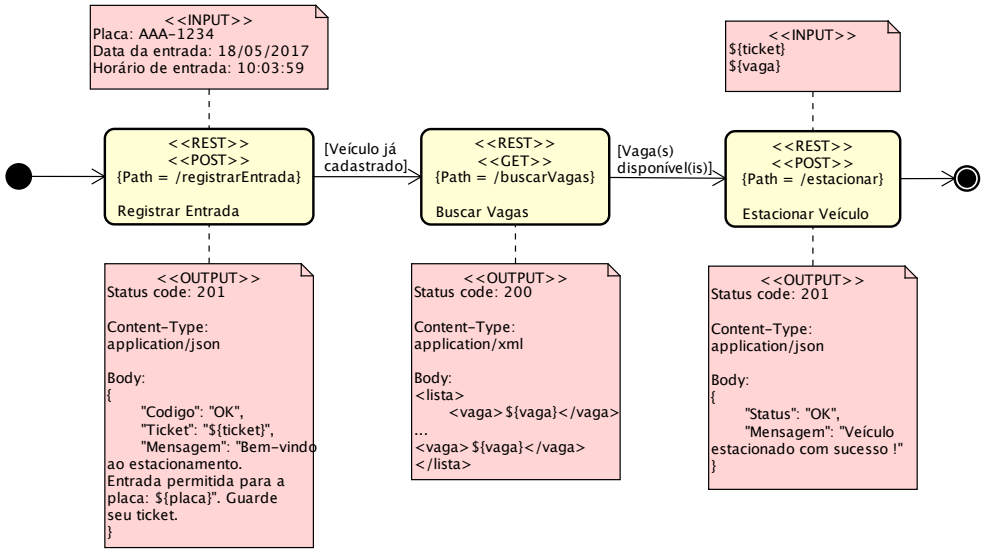


Figura 24: Exemplo de caso de teste referente à configuração 3.

A configuração "4" representa uma estratégia de teste muito comum na indústria de software: o teste funcional, em nível de sistema, executado manualmente sob uma perspectiva "caixa preta". Neste ponto, espera-se que o sistema já tenha sido submetido a testes de integração. Trata-se da configuração de estratégia de teste na qual os testadores derivam casos e procedimentos de teste a partir de requisitos funcionais e os executam sobre a interface gráfica do sistema. Geralmente, o profissional de testes que atua nessa configuração tem um perfil próximo ao de um analista de requisitos. Nesta configuração, apesar de não ser comum utilizar ferramentas para a execução dos testes, tendo em vista que são manuais, os testadores tipicamente utilizam ferramentas para apoiar a documentação dos testes de forma geral, incluindo planejamento, design e comunicação de incidentes.

Por fim, a configuração "5" reúne características semelhantes às da configuração "4", diferenciando-se apenas pela forma de execução de testes que, neste caso, é automatizada. Apesar da semelhança com a configuração "4", a configuração "5" exige um perfil profissional bem diferente. Um caso de teste desta configuração implica na navegação e no acionamento de elementos visuais em uma interface gráfica, seja ela *web*, *desktop* ou *mobile*. Tipicamente, os profissionais de teste desta configuração têm perfil de programador, tendo em vista que precisam implementar *scripts* de teste que automatizem a navegação pela interface gráfica do sistema sob teste. Esses profissionais geralmente contam com ferramentas e *frameworks* que apoiam a automação dos *scripts* de teste.

5.2.2 Instrumentação

Neste estudo foram utilizados os seguintes instrumentos:

a) Termo de Consentimento Livre e Esclarecido (TCLE);
b) Questionário de Caracterização do Participante (QCP);
c) Questionário de Percepção do Esforço de Teste (QPET);
d) Continuação do Questionário de Percepção do Esforço de Teste.

O TCLE é um documento que contém informações a respeito do procedimento realizado no *survey*, ressaltando a liberdade de desistência do participante e o acordo de confidencialidade dos resultados. O TCLE é representado no Apêndice E.

O questionário QCP foi utilizado para o participante caracterizar algumas informações pessoais e descrever alguns itens interessantes para o estudo como, por exemplo, grau de experiência em desenvolvimento e em teste de software. O QCP está disponível no Apêndice F.

O questionário QPET é o principal instrumento do coleta de dados deste *survey*. Trata-se de um questionário composto, inicialmente, por conceitos a respeito de teste de software, bem como sobre esforço em projetos de software. Em seguida, o documento descreve um conjunto de 51 fatores que eventualmente afetariam o esforço de teste de software: 47 oriundos da Revisão Sistemática de Literatura apresentada no capítulo 3 e outros quatro provenientes do Estudo de Observação apresentado no capítulo 5. O documento, que está disponível no Apêndice G possui um quadro onde o participante do estudo deveria, inicialmente, marcar a configuração de estratégia de teste sobre a qual responderia. Em seguida, o participante deveria marcar as atividades do processo de teste que foram praticadas por ele no último projeto de teste de software do qual participou sob a respectiva configuração de estratégia de teste. Por fim, para cada combinação fator-atividade, ele deveria indicar com um sinal de "+" se aquele fator exerceu influência positiva (aumentou o esforço) ou com um sinal de "-" se o fator exerceu influência negativa (diminuiu o esforço) sobre aquela atividade específica. Caso o fator não tenha sido observado, bastaria deixar o campo em branco. O questionário também possui espaços reservados para que os participantes pudessem sugerir outros fatores além daqueles 51 sugeridos pelo instrumento. A Figura 25 mostra um recorte de um QPET preenchido.

campo em branco. Fique à vontade para indicar outros fatores que estejam ausentes desta lista.

Fator de esforço	Atividade do processo					
	☑ Planejar testes	☑ Configurar ambiente de testes	☑ Projetar e implementar testes	☑ Executar testes	☑ Comunicar incidentes de teste	☐ Finalizar projeto de testes
Diversidade de atores	+		+	+		
Quantidade e duração de interrupções	+	+	+	+	+	
Disponibilidade de recursos de infraestrutura		-		-		
Disponibilidade de artefatos de teste reusáveis	-		-	-	-	
Complexidade do código-fonte						
Tamanho do código-fonte						
Complexidade do ambiente de teste		+		+	+	
Qualidade da documentação	-		-		-	
Experiência com teste	-		-	-	-	
Experiência com ferramentas de teste		-	-	-	-	
Experiência com o domínio de aplicação	-		-		-	
Experiência com o ambiente operacional	-		-		-	
Experiência com a tecnologia de programação			-	-		
Nível de criticidade	+	+	+	+		
Nível de segurança física pessoal						
Número de casos de teste			+	+		
Número de Web Services					+	
Nível de confidencialidade do projeto						
Distribuição da equipe do projeto					+	
Quantidade de incidentes de teste			+	+	+	
Legibilidade de código			+		+	
Coexistência requerida		+		+		
Capacidade de dados requerida						
Nível de interoperabilidade requerido						
Nível de desempenho requerido						
Nível de portabilidade requerido		+	+	+		

Figura 25: Exemplo de QPET preenchido.

Como os participantes tinham a possibilidade de indicar outros fatores de esforço, após a primeira etapa de análise dos dados coletados, observou-se a necessidade de se criar um novo questionário, denominado "Continuação do Questionário de Percepção do Esforço de Teste". Ele foi utilizado para questionar os participantes a respeito de um novo conjunto de fatores de esforço, que foram identificados na primeira rodada do *survey*.

Uma versão preliminar do QPET foi aplicada em um estudo-piloto, detalhado mais adiante na seção 5.3.1. No entanto, aquela versão mostrou-se inadequada para coletar os dados de interesse do estudo, inserindo ameaças à validade de construto que poderiam inibir o participante a realizar uma marcação ou, até mesmo, confundir o pesquisador quando da análise das marcações. Após o estudo-piloto, melhorias foram feitas no questionário, cuja versão final é a que se encontra no Apêndice G.

O QPET materializa um modelo conceitual que foi desenhado com o objetivo de representar as respostas às questões de pesquisa deste estudo. Esse modelo conceitual, representado na Figura 26, considera que cada conjunto de respostas dadas por um participante a respeito de uma configuração de estratégia de teste é um *ponto de vista*. Cada ponto de vista é composto por julgamentos a respeito da influência de cada um dos fatores de esforço apresentados. Um *julgamento* representa um juízo ou apreciação a respeito da influência de um fator sobre o esforço de uma atividade específica do processo de teste. Cada julgamento possui um tipo de impacto e um *rationale*. O *rationale* é basicamente a fundamentação ou justificativa lógica que sustenta o juízo realizado. Já o tipo de impacto pode ser classificado como "Nenhum", "Positivo", "Negativo", "Positivo e negativo" ou "Não avaliado". O tipo de impacto "Nenhum" representa a situação em que o participante julga que determinado fator não exerce influência sobre o esforço daquela atividade específica. O impacto "Positivo" representa que o fator aumenta o esforço, ao passo que o impacto "Negativo" indica que o fator diminui o esforço de teste na atividade. Já o impacto "Positivo e negativo" representa aquelas situações em que o participante julga que, dependendo de alguma circunstância, o fator pode aumentar ou diminuir o esforço. Finalmente, o tipo de impacto "Não avaliado" é usado nas situações em que o participante não praticou e, portanto, não respondeu a respeito de um determinada atividade do processo. Portanto, neste caso o participante declara que não se sente capaz de avaliar se os fatores de esforço exercem influência sobre determinadas atividades.

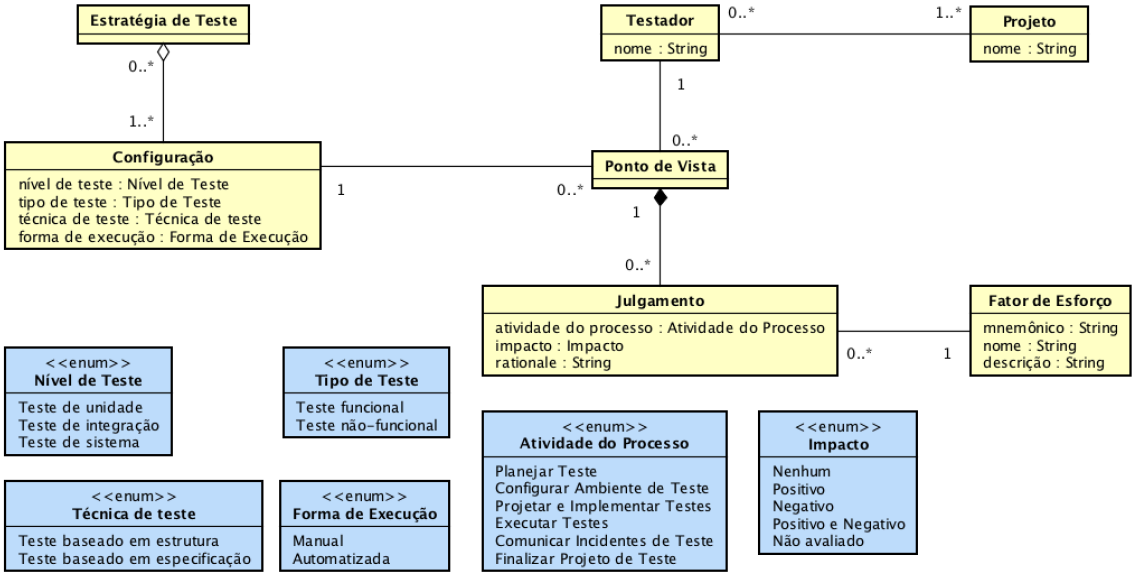


Figura 26: Modelo conceitual usado no instrumento de coleta de dados.

5.2.3 Variáveis

As variáveis independentes deste estudo são as seguintes:

- o conjunto inicial de fatores de esforço, contendo 47 fatores oriundos da Revisão Sistemática e quatro fatores identificados no Estudo de Observação;
- as atividades do processo de teste de software, conforme definido na norma ISO/IEC/IEEE 29119-2 (ISO/IEC/IEEE, 2013);
- as configurações de estratégia de teste escolhidas pelo pesquisador.

As variáveis dependentes, por sua vez, são as seguintes:

- as atividades do processo de teste de software ordenadas pelo volume de esforço demandado em cada configuração de estratégia de teste;
- o conjunto final de fatores de esforço.

5.2.4 Seleção do contexto

Após receber os instrumentos do coleta de dados, os participantes poderiam respondê-los onde quisessem e sem restrição de tempo. Também não houve qualquer tipo de monitoramente.

Assim que cada participante sinalizava que havia finalizado o preenchimento dos instrumentos, era realizada uma entrevista com o objetivo de capturar o *rationale* dos julgamentos indicados no QPET. Essas entrevistas eram realizadas em salas nas organizações dos participantes sem a presença de terceiros.

5.2.5 Seleção dos participantes

Os participantes do estudo foram escolhidos por conveniência, por se tratarem de pessoas inseridas em empresas do relacionamento do pesquisador. A partir de uma consulta prévia a um universo de cerca de quarenta pessoas, incluindo profissionais de diversas organizações, vinte e oito profissionais com experiência em teste de software se candidataram, voluntariamente, a participar do estudo. Dos vinte e oito voluntários, dezessete atuam na mesma empresa do pesquisador e os outros onze atuam em onze

empresas distintas. Somente três participantes trabalham em empresas sediadas fora do município do Rio de Janeiro: um em Brasília, outro em Belém e o terceiro em Montreal (Canadá). Dentre os dezessete participantes que atuam na mesma empresa do pesquisador, dezesseis atuam na mesma sede regional, tendo participado de pelo menos um projeto em comum nos últimos anos.

O convite para participação no estudo foi feito levando-se em conta as seis configurações de estratégia de teste apresentadas. Inicialmente, pretendia-se recrutar participantes que tivessem experiência e pudessem responder sobre todas as seis configurações. Tendo os mesmos participantes respondendo a respeito de todas as configurações de estratégia de teste possibilitaria a realização de comparações entre as configurações. No entanto, observou-se ainda no planejamento do estudo que dificilmente seriam encontrados profissionais com conhecimento prático e com disponibilidade para responder sobre tantos fatores de esforço em tantas configurações de estratégia de teste.

Desta forma, o estudo conta com apenas um participante que respondeu a respeito de três configurações distintas. Os demais, responderam sobre apenas uma configuração cada. Portanto, o estudo conta com vinte e oito profissionais que forneceram trinta pontos de vista sobre o esforço de teste: seis configurações de estratégia de teste, cada uma com cinco pontos de vista.

5.2.6 Caracterização do survey

Este *survey* pode ser caracterizado da seguinte forma (MELLO, 2016):

- **Quanto à instrumentação:** entrevistas estruturadas (o pesquisador aplicou um questionário seguido de entrevista da mesma forma para cada participante).
- **Quanto ao apoio aos participantes:** supervisionado (com as entrevistas sendo realizadas após o preenchimento dos questionários, o pesquisador procurou garantir que todas as questões fossem entendidas e respondidas).
- **Quanto ao objetivo:** descritivo (tendo em vista que o objetivo da pesquisa é identificar opiniões de uma população e capturar a sua percepção sobre a prática de teste de software).
- **Audiência-alvo:** profissionais da indústria de software.

- **Caracterização do participante:** profissionais da indústria de software que atendem aos seguintes critérios de inclusão:
    - pelo menos dois anos de experiência em projetos de software;
    - experiência em pelo menos um projeto de teste de software referente a uma das seis configurações de estratégia de teste cobertas no estudo;
    - no máximo dois anos desde a última experiência em projetos de teste de software.
- **Design da amostragem:** amostragem acidental (por conveniência), tendo em vista que a participação foi voluntária.
- **Processo de recrutamento:** inicialmente, o pesquisador levantou quais profissionais dentro da sua rede de contatos atenderiam aos critérios de inclusão; em seguida, o pesquisador entrou em contato com cada um desses profissionais para convidar a participar do estudo.
- **Método de convite:** o convite foi feito pessoalmente, por telefone ou por e-mail, de acordo com a disponibilidade e a distância física do pesquisador em relação ao candidato a participante do estudo.
- **Método de recompensa:** não foi oferecida qualquer recompensa. A participação ocorreu de forma voluntária.
- **Tempo estimado:** entre 20 e 30 minutos para preenchimento do questionário e cerca de 1h para a entrevista (esta estimativa foi calibrada a partir de um estudo-piloto).
- **Período de realização das entrevistas:** janeiro/2018 (entrevista principal); maio e junho/2018 (entrevista complementar).

5.3 Execução

A execução do estudo se deu em duas etapas: um estudo-piloto com estudantes de pós-graduação e o estudo com profissionais, os quais estão descritos a seguir.

5.3.1 Estudo-piloto

O estudo-piloto foi realizado no último trimestre de 2017, com 12 alunos da disciplina de Verificação, Validação e Teste do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. No âmbito da referida disciplina, os alunos receberam treinamento sobre testes funcionais de aplicações *mobile*, bem como sobre testes funcionais e de desempenho de *web services* RESTful, usando as ferramentas TestLink, SoapUI e Apache JMeter. Ao final da disciplina, os alunos foram organizados em duplas ou trios para realizar um trabalho prático que consistia em realizar testes, incluindo a documentação e implementação de *scripts*, sobre uma aplicação real que havia sido desenvolvida por alunos de graduação em Engenharia de Computação e Informação da UFRJ. Trata-se de uma aplicação *mobile* baseada em *web services* RESTful para controle de entrada e saída de pessoas em laboratórios. As configurações de estratégia de teste cobertas no estudo-piloto foram as seguintes, representadas na Tabela 37:

Tabela 37: Configurações de estratégia de teste cobertas no estudo-piloto.

Configuração	Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
1	Unidade	Funcional	Baseada em especificação	Automatizada
2	Unidade	Desempenho	Baseada em especificação	Automatizada
5	Sistema	Funcional	Baseada em especificação	Manual

Após a entrega dos trabalhos, os alunos foram convidados a preencher um questionário indicando quais fatores de esforço foram observados em cada atividade do processo de teste ao longo daquele trabalho em cada uma das três configurações de estratégia de teste exercitadas. Era necessário, portanto, associar os fatores previamente listados no questionário com as atividades do processo de teste. Para cada marcação realizada, era necessário indicar, com os sinais de "+" e "-", se o fator aumentava ou diminuía o esforço da respectiva atividade. A Figura 27 representa um exemplo de questionário preenchido no estudo-piloto.

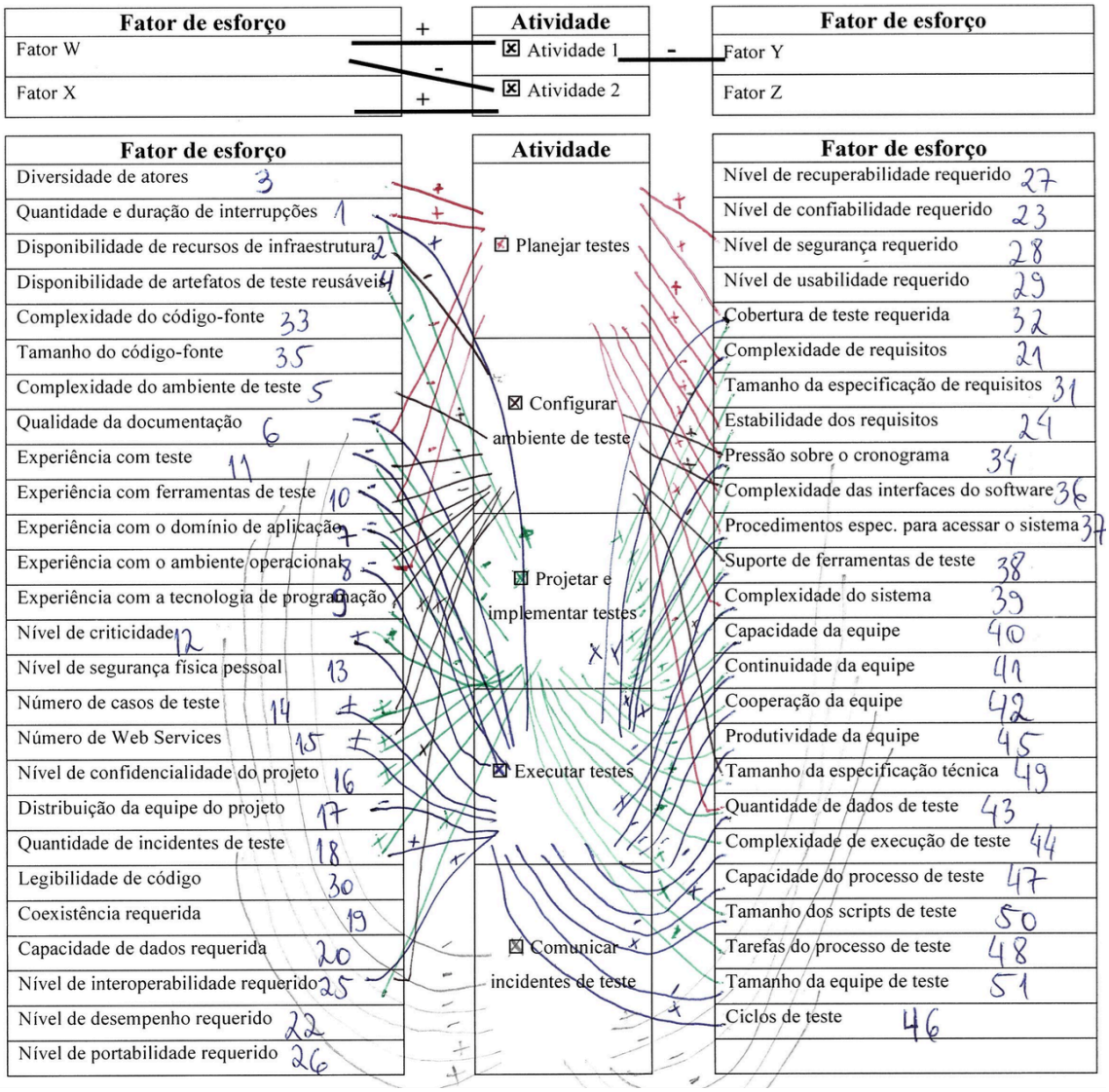


Figura 27: Versão preliminar do questionário de percepção de esforço de teste.

Inicialmente, havia a intenção de aproveitar os resultados do estudo-piloto para compor um modelo de percepção do esforço de teste. No entanto, os resultados se mostraram pouco confiáveis para tal. Alguns participantes deixaram de assinalar fatores de esforço que aparentemente eram óbvios. Outros, provavelmente inverteram a representação do tipo de influência do fator: usaram o sinal de "+" para representar que o fator exercia influência positiva, considerando que isso significa diminuir o esforço, quando na verdade é o contrário. Esses resultados podem ser atribuídos a dois motivos: a inexperiência dos participantes com teste de software e a inadequação do instrumento de coleta de dados.

Como pode ser observado na Figura 27, a quantidade de associações entre fatores e atividades pode causar confusão tanto para quem responde quanto para quem

analisa as respostas do questionário. Eventualmente, um participante poderia realizar uma associação de forma equivocada, dado o emaranhado de associações realizadas. Além disso, o participante poderia deixar de realizar uma associação para não ter o trabalho de tentar representá-la de forma clara. O pesquisador, por sua vez, poderia ter dificuldade de identificar cada associação realizada, por não haver clareza de onde inicia e onde termina cada marcação. Portanto, a primeira contribuição do estudo-piloto foi evidenciar que a versão preliminar do questionário de percepção do esforço de teste apresentava ameaças à validade de construto que poderiam afetar negativamente a confiança dos resultados. Outra contribuição foi a evidência de que capturar a percepção de estudantes ou profissionais pouco experientes sobre um assunto que exige maturidade e experiência profissional é uma iniciativa propensa a erros.

5.3.2 Estudo com profissionais

Uma vez que os profissionais contatados pelo pesquisador confirmavam o intenção de participar do estudo, eles recebiam os instrumentos do estudo impressos ou de forma digital para preenchimento. Neste momento, eram orientados a entrar em contato com o pesquisador caso tivessem qualquer dúvida quanto ao preenchimento dos instrumentos. Alguns participantes levaram semanas para iniciar o preenchimento, devido a compromissos profissionais. Outros, responderam no mesmo dia do recebimento.

Assim que cada participante informava o término do preenchimento dos instrumentos, uma entrevista era agendada para que o pesquisador pudesse coletar o *rationale* de cada julgamento indicado no questionário com impacto "Positivo", "Negativo" ou "Positivo e negativo". Como pode ser observado na Figura 25, muitos julgamentos foram remarcados durante as entrevistas, seja porque o participante havia se lembrado de alguma situação referente a determinados fatores no momento da entrevista, seja porque o participante havia tido uma compreensão equivocada sobre a definição de determinados fatores.

As entrevistas ocorreram em dois momentos. No primeiro, em janeiro de 2018, foi aplicado o questionário QPET e realizadas as respectivas entrevistas. O segundo, realizado entre maio e junho de 2018, consistiu na aplicação da continuação do QPET, além da realização de novas entrevistas, com foco nos fatores sugeridos pelos

participantes na primeira rodada de entrevistas. Somando-se os tempos de entrevistas dos dois momentos, a entrevista mais rápida referente a um ponto de vista durou cerca de 50min. Já a entrevista mais longa levou 2h12min. Ao todo, foram realizadas cerca de 60h de entrevistas com os participantes.

Finalizadas as entrevistas, os dados informados pelos participantes foram tabulados para proceder a sua análise. Dada a quantidade e variedade de dados, além da necessidade de se realizar inúmeros recortes, optou-se por armazená-los em um banco de dados e visualizá-los usando *dashboards*. Assim, foi criado um banco de dados de acordo com o modelo conceitual representado na Figura 26 usando a ferramenta MySQL. Após a inclusão e verificação de todos os dados no banco de dados, foram criados diversos *dashboards* com a ferramenta de visualização de dados Grafana, representada na Figura 28, possibilitando que os dados fossem analisados sob diferentes perspectivas.

Figura 28: Perspectiva de um *dashboard* criado com a ferramenta Grafana.

5.4 Resultados

Os resultados do estudo estão divididos nas quatro subseções a seguir. Inicialmente, na seção 5.4.1, é apresentada a caracterização dos participantes, incluindo o nível de formação acadêmica, experiência profissional e nível de conhecimento sobre uma série de práticas de engenharia de software de interesse do estudo. Essas informações foram coletadas por meio do questionário QCP. Os resultados dessa

caracterização são apresentados tanto de forma geral como segmentada por configuração de estratégia de teste.

A seção 5.4.2 refere-se à questão de pesquisa QE6, incluindo a questão QE6.1. Os seus resultados mostram, por meio de um *ranking*, a percepção de esforço demandado por atividade de teste na perspectiva dos participantes do estudo.

A seção 5.4.3 refere-se à questão de pesquisa QE7 e todas as suas subquestões. Trata-se de um conjunto de dados volumoso, que mostra a percepção dos participantes a respeito da influência de um conjunto de fatores sobre o esforço nas atividades do processo de teste, de forma geral e segmentada por configuração de estratégia de teste.

Por último, a seção 5.4.4 contempla uma análise de concordância dos participantes, realizada com o objetivo de medir o nível de consenso existente entre as respostas fornecidas ao estudo.

5.4.1 Caracterização dos participantes

Quanto ao nível de formação acadêmica, os 28 participantes do estudo têm, no mínimo, a graduação completa. Conforme a Figura 29, seis deles concluíram o Mestrado (M.Sc.) e outros sete realizaram cursos de Especialização (Pós-Graduação *Lato Sensu*). Dentre os mestres, há dois participantes em fase de conclusão do Doutorado e entre os especialistas, há outros dois em fase avançada do Mestrado. Portanto, essa característica sugere que tratam-se de profissionais que receberam treinamento acadêmico adequado ao nível de complexidade exigido pelo estudo.

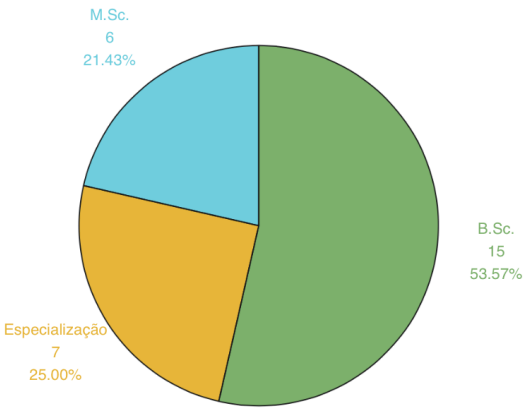


Figura 29: Grau máximo de formação dos participantes do estudo.

Quanto ao tempo de experiência, como mostra a Tabela 38, os participantes do estudo têm média de 13,4 e mediana de 12 anos. Um dos participantes declarou ter dois anos de experiência em atividades ligadas à engenharia de software, que é o tempo mínimo definido como critério para participação no estudo. Outros dois participantes declararam atuar na área há 40 anos. Esses números não refletem diretamente o nível de conhecimento dos participantes, mas reforçam seu grau de vivência na área de software.

Tabela 38: Experiência dos participantes do estudo em ES (em anos).

Média	Mediana	Mínima	Máxima
13,4	12	2	40

O nível de conhecimento dos participantes em engenharia de software e, especialmente, em teste de software foi coletado por meio de 16 práticas, listadas na segunda página do questionário QCP (Apêndice F). Algumas das práticas são mais direcionadas à uma configuração de estratégia de teste específica, como "Execução de teste de desempenho" que é fundamental à configuração 2. Outras, no entanto, possuem um caráter mais geral, fazendo sentido em qualquer configuração, tais como "Ferramentas de controle de versão". Para cada prática, o participante deveria assinalar uma das respostas a seguir:

- Pratiquei em vários projetos na indústria;
- Pratiquei em um projeto na indústria;
- Pratiquei em projetos de curso;
- Estudei em cursos ou livros;
- Nenhuma experiência.

Os resultados sobre o conhecimento dessas práticas pelos participantes estão representados no Apêndice I (Figura 56 e Figura 57). Os resultados segmentados por configuração de estratégia de teste são apresentados nas subseções seguintes.

Em relação à experiência com "Especificação de requisitos de software", mais da metade dos participantes (15) declarou ter praticado em mais de um projeto na indústria. Apesar de não ser uma tarefa típica de testes, conhecimentos sobre como especificar requisitos podem auxiliar na compreensão dos documentos usados como base para criação de diversos artefatos de testes.

Quanto à experiência com "Revisão/inspeção de requisitos", novamente, mais da metade dos participantes (15) declarou ter praticado em mais de um projeto na indústria.

Essa prática torna-se importante de ser medida considerando que é comum que testadores identifiquem defeitos na documentação de requisitos, durante a atividade "Projetar e implementar testes", quando estão derivando casos e procedimentos de teste.

Sobre "Ferramentas de controle de versão", 24 participantes (86%) declararam ter utilizado em vários projetos. Isso pode sugerir que a maioria dos participantes têm experiência em projetos de larga escala e que são desenvolvidos de forma colaborativa (em equipes), uma vez que ferramentas dessa natureza são essenciais para projetos com essas características. Ferramentas de controle de versão são fundamentais para gerenciar as versões dos diversos artefatos do projeto, incluindo os artefatos de testes.

Relativamente à experiência com "Desenvolvimento Web", 23 participantes (82%) declararam ter praticado em vários projetos. Considerando que a maior parte dos julgamentos realizados teve como base experiências dos participantes em projetos de testes de sistemas Web, essa característica reforça o quão habilidosos eles são nessa plataforma. Uma vez que um profissional conhece as especificidades de uma tecnologia de software, da perspectiva do desenvolvedor, ele pode projetar e implementar testes mais específicos e de forma mais eficiente. Por exemplo, um participante referente à configuração 5, que presume a automação de testes com base na interface gráfica, pode se valer da sua experiência como desenvolvedor de sistemas web para implementar *scripts* de teste de GUI com esforço relativamente menor.

Quanto ao "Planejamento de testes", 19 participantes (68%) já praticaram em mais de um projeto na indústria. Geralmente, o líder de projeto é responsável por planejar o projeto de testes. Assim, o número apresentado pode indicar que a maioria dos participantes exerce ou já exerceu liderança em projetos de teste. Por outro lado, aqueles que responderam não ter experiência real nesta prática podem estar nessa condição pelo simples fato de que a configuração de estratégia de teste que pratica não exige a realização de uma etapa de planejamento.

"Design e implementação de testes" foi relatada por 21 participantes (75%) como tendo sido praticada em vários projetos na indústria. Apesar de ser uma das atividades mais importantes do processo de teste, quatro participantes declararam que nunca a praticaram em projetos reais. Mesmo assim eles foram incluídos no estudo, tendo em vista que atendiam aos critérios de inclusão e porque suas experiências em outras atividades do processo de teste poderiam ser de grande utilidade para o estudo.

"Execução de testes funcionais" foi a prática com maior parcela de participantes indicando que realizaram em mais de um projeto na indústria: 26 (93%). Vale ressaltar

que os outros dois participantes também tiveram experiência prática na indústria, porém em apenas um projeto. Ao responder, o participante poderia considerar qualquer experiência com testes que tenha avaliado requisitos funcionais, não importando o nível, a técnica ou a forma de execução dos testes.

Já "Execução de testes de desempenho" recebeu respostas bem difusas. É a única prática entre as 16 analisadas cuja resposta mais frequente é "Pratiquei em um projeto na indústria", com 9 respostas (32%). No entanto, como se trata de uma prática referente a um tipo de teste mais específico, esses números não chegam a representar uma ameaça à validade do estudo, desde que os participantes referentes à configuração 2 tenham a experiência adequada.

"Uso de *frameworks* de teste de unidade" foi relatada pela metade dos participantes (14) como tendo sido realizada em vários projetos reais. Apesar dessa prática estar fortemente relacionada à configuração 0, outras configurações fazem uso de *frameworks* de teste de unidade, especialmente a confuguração 5, uma vez que alguns dos principais *frameworks* de teste de GUI trabalham em conjunto com *frameworks* de teste de unidade.

Enquanto isso, o "Uso de *frameworks* de teste de GUI" foi indicado como praticado em vários projetos por 13 participantes (46%). Vale destacar que outros sete participantes declaram ter usado *frameworks* de teste de GUI em um projeto. Assim, 20 participantes (71%) tiveram experiência em pelo menos um projeto real implementando *scripts* de teste baseados na interface gráfica do sistema sob teste. Essa prática está fortemente relacionada à configuração de estratégia de teste 5, pois esta considera o teste "caixa preta" com execução automatizada.

"Uso de ferramentas de integração contínua" foi praticado por dez participantes (36%) em mais de um projeto na indústria, com outros oito participantes tendo experiência em apenas um projeto real. Essa prática se torna relevante ao estudo à medida que cinco das seis configurações de estratégia de teste abordadas envolve a execução automática dos testes. Nesse contexto, ferramentas de integração contínua cumprem um papel fundamental, possibilitando o acionamento automático dos *scripts* de teste.

Em relação ao "Uso de ferramentas de teste de *web services*", metade dos participantes (14) declarou ter experiência em vários projetos na indústria, além de outros sete participantes que indicaram ter participado de apenas um projeto. Trata-se de uma prática especialmente relevante às configurações de estratégia de teste 1 e 3. Faz-se

necessário levantar a experiência dos participantes nessa prática por se referir a testes de *web services*, uma categoria de testes que impõe desafios que exigirão conhecimentos específicos do profissional de teste, especialmente sobre o protocolo HTTP.

"Uso de ferramentas de teste de desempenho" é uma das três práticas em que a resposta mais frequente foi "Nenhuma experiência". Apenas cinco participantes (18%) declararam ter utilizado ferramentas desse tipo em mais de um projeto na indústria. Percebe-se, portanto, que o perfil médio dos participantes tem grande experiência com testes funcionais e pouca experiência com testes de desempenho.

Sobre "Comunicação de incidentes de teste", 20 participantes (71%) indicaram ter tido experiência em vários projetos industriais. Trata-se de uma prática referente à uma atividade do processo de teste ("Comunicar incidentes de teste") potencialmente comum a todas as configurações de estratégia de teste cobertas no estudo. Ao responder, o participante poderia considerar experiências em projetos em que a comunicação de incidentes fosse feita de modo formal, gerenciada por uma ferramenta de *bugtracking*, ou de modo informal, onde a comunicação era feita de forma oral sem qualquer registro.

O questionário apurou o nível de experiência dos participantes em três práticas envolvenvo a tecnologia de *web services RESTful*, considerando o fato dessa tecnologia ser o foco das configurações 1 e 3. Sobre "Desenvolvimento de *web services RESTful*", dez participantes (36%) relataram ter praticado em mais de um projeto; sobre "Design e implementação de testes de *web services RESTful*", onze participantes (39%) responderam dessa forma; já em relação à "Execução de testes de *web services RESTful*" foram 12 participantes (43%). Esse último evidencia uma certa incoerência em relação às respostas dadas à prática "Uso de ferramentas de teste de *web services*", onde 14 participantes declararam ter experiência em vários projetos na indústria, tendo em vista que, tipicamente, a execução de testes de *web services RESTful* se dá por meio de uma ferramenta de teste de *web services*.

Por fim, as respostas sobre as práticas relacionadas a aplicações *mobile* evidenciaram a pouca experiência dos participantes com esse tipo de tecnologia. Sobre "Desenvolvimento de aplicações *mobile*", a maior parcela de respostas, composta por oito participantes (29%), indicou ter estudado apenas em cursos ou livros. Outros seis participantes (21%) declararam não ter qualquer experiência com essa prática. Já "Design e implementação de testes de aplicações *mobile*" e "Execução de testes de aplicações *mobile*" foram outras práticas cuja resposta mais frequente foi "Nenhuma experiência", com 17 (61%) e 16 (57%) participantes, respectivamente.

5.4.1.1 Participantes da configuração 0

Neste estudo, a configuração de estratégia de teste "0" é aquela formada pelo teste em nível de unidade, do tipo funcional, utilizando a técnica de teste baseado em estrutura, com a execução automatizada, como mostra a Tabela 39.

Tabela 39: Dimensões da configuração de estratégia de teste 0.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de unidade	Teste funcional	Teste baseado em estrutura	Automatizada

Os cinco participantes do estudo referentes a essa configuração de estratégia de teste atuam em três organizações distintas: três deles na mesma empresa. As três organizações envolvidas são vinculadas ao Governo Federal. O tempo médio de experiência dos participantes na indústria de software é de nove anos. Quanto ao nível de formação além da graduação, dos cinco profissionais entrevistados, um possui Mestrado e dois cursaram Especialização (*Lato Sensu*).

Dentre as configurações de teste tratadas no estudo, a configuração 0 é aquela mais próxima do código-fonte. Por esse motivo, os participantes do estudo são profissionais com perfil de programador. Sendo assim, dentre as habilidades esperadas dos participantes, "Uso de ferramentas de testes de unidade" seria a mais importante, seguida daquelas habilidades relacionadas a desenvolvimento de software, como "Desenvolvimento Web" e "Desenvolvimento de *web services* RESTful".

A respeito dessas práticas, é possível observar na Figura 58 que todos os cinco participantes têm experiência em vários projetos na indústria tanto com "Uso de ferramentas de testes de unidade", bem como em "Desenvolvimento Web". Além disso, como mostra a Figura 59, os cinco participantes também têm experiência com "Desenvolvimento de *web services* RESTful" em projetos reais, sendo que a maior parte dos participantes (60%) já participou de mais de um projeto.

Além disso, vale destacar as habilidades com "Ferramentas de controle de versão", "Execução de testes funcionais" e "Uso de ferramentas de integração contínua", que fazem parte do contexto de trabalho do programador que pratica a configuração 0, para as quais os cinco participantes relataram ter experiência em diversos projetos industriais. Portanto, as características apresentadas reforçam que os profissionais entrevistados têm conhecimento suficiente para responder às questões apresentadas no estudo.

5.4.1.2 Participantes da configuração 1

Neste estudo, a configuração de estratégia de teste "1" é aquela formada pelo teste em nível de unidade, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como mostra a Tabela 40.

Tabela 40: Dimensões da configuração de estratégia de teste 1.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de unidade	Teste funcional	Teste baseado em especificação	Automatizada

Os participantes do estudo referentes a essa configuração de estratégia de teste atuam em apenas duas organizações diferentes: quatro deles na mesma empresa (Governo Federal). O tempo médio de experiência com software é de 10,4 anos. Após a graduação, dois participantes cursaram Pós-Graduação *Lato Sensu*. Entre as configurações tratadas no estudo, esta é, juntamente com as configurações 3 e 4, uma daquelas com menor diversidade em relação às organizações de origem dos participantes. Isso se deve ao fato de que trata-se de uma configuração de estratégia de teste mais específica, para a qual assumiu-se que as unidades testadas são *web services RESTful*. Apesar de quatro dos cinco participantes trabalharem na mesma empresa, cada um pertence a uma equipe diferente e atuam em projetos distintos. O outro participante atua em um empresa privada especializada em segurança da informação.

Dentre as habilidades esperadas dos participantes, "Uso de ferramentas de testes de *web services*", "Design e implementação de testes de *web services RESTful*" e "Execução de testes de *web services RESTful*" seriam as mais relevantes. Seria desejável, porém não fundamental, que os participantes também tivessem experiência com "Desenvolvimento de *web services RESTful*", por conta das especificidades da tecnologia REST.

É possível observar na Figura 61 que para as três habilidades mais relevantes, 60% dos participantes tiveram experiência em vários projetos e os outros 40% tiveram experiência em um projeto na indústria. Apesar de 40% dos participantes não ter realizado essas práticas em mais de um projeto, tratam-se de profissionais com larga experiência em outras práticas úteis a essa configuração, relacionadas ao planejamento, design e implementação, e execução de testes funcionais, como mostra a Figura 60, nas quais todos os cinco participantes têm experiência em vários projetos reais.

5.4.1.3 Participantes da configuração 2

A configuração de estratégia de teste "2" é formada pelo teste em nível de unidade, do tipo de desempenho, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como mostra a Tabela 41.

Tabela 41: Dimensões da configuração de estratégia de teste 2.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de unidade	Teste de desempenho	Teste baseado em especificação	Automatizada

Os profissionais entrevistados atuam em três organizações distintas: uma empresa do Governo Federal, uma empresa de telefonia e uma empresa de desenvolvimento de sistemas de pagamento eletrônico. Na referida empresa do Governo atuam três participantes. A experiência média dos participantes na indústria de software é de 15,8 anos. Quanto ao nível de formação além da graduação, dos cinco profissionais entrevistados, um possui Mestrado e outro cursou Especialização.

Dentre as configurações de estratégia de teste deste estudo, trata-se da única cujo tipo de teste é não-funcional. Por esse motivo, espera-se que o perfil do profissional que pratica essa configuração apresente diferenças significativas em relação aos profissionais que praticam as demais configurações. Provavelmente, as habilidades mais importantes de serem observadas nos profissionais que praticam a configuração 2 são aquelas específicas de testes de desempenho: "Execução de testes de desempenho" e "Uso de ferramentas de teste de desempenho". A respeito dessas duas características, pode-se observar na Figura 62 e na Figura 63 que todos os participantes haviam tido experiência na indústria em projetos de teste de desempenho, sendo que quatro dos cinco participantes tinham experiência em mais de um projeto.

Também vale enfatizar que os cinco participantes possuem larga experiência com "Design e implementação de testes de *web services RESTful*" e "Execução de testes de *web services RESTful*", habilidades relevantes neste caso considerando que focou-se na experiência dos participantes em testes de desempenho de *web services RESTful*, descartando experiências com outras tecnologias de software. Outro ponto a destacar: essa configuração é aquela que apresenta mais práticas com 100% de respostas "Pratiquei em vários projetos na indústria" (sete). Portanto, essas habilidades, somadas às demais representadas na Figura 62 e na Figura 63, sugerem que os participantes desta configuração têm domínio sobre a prática, fortalecendo, assim, as evidências do estudo.

5.4.1.4 Participantes da configuração 3

Neste estudo, a configuração de estratégia de teste "3" é aquela formada pelo teste em nível de integração, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como mostra a Tabela 42.

Tabela 42: Dimensões da configuração de estratégia de teste 3.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de integração	Teste funcional	Teste baseado em especificação	Automatizada

Semelhantemente à configuração 1, os participantes do estudo referentes a essa configuração de estratégia de teste atuam em apenas duas organizações distintas, sendo que quatro participantes atuam na mesma empresa (Governo Federal). O outro participante atua em uma *startup* de tecnologia. A experiência média dos participantes na indústria de software é de 14,4 anos. Quanto ao nível de formação, dois deles possuem mestrado, sendo um doutorando em fase final do curso.

Dentre as configurações de estratégia de teste tratadas no estudo, esta foi aquela que apresentou mais dificuldade para recrutar participantes, dada a especificidade da configuração (testes de integração) e da tecnologia de software contemplada (*web services RESTful*). Por esse motivo, nem todos os profissionais recrutados apresentam o mesmo nível de senioridade em teste de software sob esta configuração do que os participantes das demais configurações.

Assim como a configuração 1, as habilidades mais relevantes para esta configuração são "Uso de ferramentas de testes de *web services*", "Design e implementação de testes de *web services RESTful*" e "Execução de testes de *web services RESTful*". Como mostram a Figura 64 e a Figura 65, em cada uma dessas três práticas, apenas três participantes (60%) haviam tido experiência em mais de um projeto real. Isso evidencia que trata-se de um grupo um pouco menos experiente na configuração de estratégia de teste que outros grupos.

Apesar da aparente falta de experiência nesta configuração, os participantes possuem experiência em outras habilidades complementares, tais como "Desenvolvimento Web", em que todos os cinco declararam ter praticado em vários projetos e em "Uso de ferramentas de integração contínua", onde quatro participantes já praticaram na indústria, sendo três deles em vários projetos.

5.4.1.5 Participantes da configuração 4

Neste estudo, a configuração de estratégia de teste "4" é aquela formada pelo teste em nível de sistema, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução manual, como mostra a Tabela 43.

Tabela 43: Dimensões da configuração de estratégia de teste 4.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de sistema	Teste funcional	Teste baseado em especificação	Manual

Os cinco participantes do estudo referentes a essa configuração de estratégia de teste atuam em apenas duas organizações distintas: quatro deles na organização vinculada ao Governo Federal citada nas seções anteriores. O outro participante atuava em empresa privada no ramo de sistemas de resultados de eventos esportivos. Os profissionais entrevistados têm em média 15 anos de experiência na indústria de software. Quanto ao nível de formação além da graduação, dois participantes cursaram Especialização.

A configuração 4 é a única tratada no estudo cuja forma de execução é manual. Como possível consequência, as habilidades típicas dos profissionais que atuam nessa configuração são diferentes das habilidades relevantes às demais configurações. Neste caso, as habilidades mais relevantes seriam "Planejamento de testes", "Design e implementação de testes", "Execução de testes" e "Comunicação de incidentes de teste". Conforme a Figura 66 e a Figura 67, para todas essas práticas, os participantes responderam que possuíam experiência em vários projetos na indústria, o que corrobora a ideia de que se tratam de profissionais que dominam a prática dessa configuração de estratégia de teste.

Essa configuração é aquela com mais práticas cuja resposta mais frequente é "Nenhuma experiência": nove no total. Isso pode sugerir que os participantes, apesar de dominarem a configuração em questão, possuem conhecimentos limitados nas demais. Outra explicação pode ser o fato dessa configuração ser a única entre as seis tratadas no estudo cuja execução de testes é manual. Provavelmente, essa característica é preponderante para determinar o perfil do profissional.

5.4.1.6 Participantes da configuração 5

Neste estudo, a configuração de estratégia de teste "5" é aquela formada pelo teste em nível de sistema, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como mostra a Tabela 44.

Tabela 44: Dimensões da configuração de estratégia de teste 5.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de sistema	Teste funcional	Teste baseado em especificação	Automatizada

Esta é a única configuração de estratégia de teste do estudo cujos cinco participantes provêm de cinco empresas distintas. Há desde *startup* de tecnologia até empresa de Governo, passando por empresa de comunicação e jornalismo. A experiência média dos participantes na indústria de software é de 15,6 anos, a maior entre as seis configurações do estudo. Em relação ao nível de formação além da graduação, dois participantes possuem Mestrado.

Trata-se de mais uma configuração cujos participantes possuem perfil mais próximo ao de um programador, por conta da necessidade de se implementar *scripts* de teste. Neste caso, dentre as habilidades analisadas, "Uso de *frameworks* de testes de GUI" e "Desenvolvimento Web" seriam as mais relevantes. A primeira porque a implementação dos *scripts* de teste de interface gráfica geralmente é realizada com apoio ferramental e a segunda porque nos casos em que a interface gráfica do sistema sob teste é Web, conhecer as tecnologias de desenvolvimento pode facilitar o trabalho de automação de testes. Como mostram a Figura 68 e a Figura 69, todos os cinco participantes tiveram experiência na indústria com "Uso de *frameworks* de testes de GUI", sendo três deles em mais de um projeto, e quatro participantes (80%) indicaram ter experiência com "Desenvolvimento Web" em vários projetos.

Outras habilidades desse grupo merecem destaque pela relevância para a configuração 5, tais como "Ferramentas de controle de versão", já que os *scripts* de teste geralmente precisam ser versionados, e "Uso de *frameworks* de testes de unidade", considerando que alguns dos principais *frameworks* de testes de GUI atuam sobre uma camada de *frameworks* de testes de unidade. Quanto à primeira prática, todos os cinco participantes possuem experiência em vários projetos na indústria. Já em relação à segunda prática, três participantes experimentaram em mais de um projeto.

5.4.1.7 Ferramentas de teste utilizadas

Com o objetivo de tornar ainda mais precisa a caracterização dos participantes do estudo, o questionário QPET possuía um espaço para que o profissional entrevistado relacionasse as ferramentas que costuma utilizar na configuração de estratégia de teste sobre a qual estava respondendo. A Figura 30 representa a frequência de ferramentas usadas pelos participantes do estudo por configuração.

A configuração 0 apresentou grande variedade de ferramentas. No entanto, das ferramentas citadas pelos participantes, há apenas três *frameworks* de testes de unidade para testes estruturais: JUnit (Java), PyTest (Python) e RSpec (Ruby). As demais ferramentas citadas são *frameworks* para criação de *mocks* em Java (Mockito, EasyMock e PowerMock), além da ferramenta de integração contínua Jenkins, que é utilizada para automatizar o acionamento da execução de testes automatizados.

A configuração 1 foi aquela que apresentou menor variedade de ferramentas. Das três ferramentas, duas são usadas para implementação e execução de testes (SoapUI e Postman) e a outra é usada para edição e gestão de artefatos de teste (ALM), podendo compreender todas as atividades do processo de teste. Essa concentração em poucas ferramentas é um reflexo da baixa variedade de organizações representadas pelos participantes dessa configuração.

A configuração 2 também apresentou grande variedade de ferramentas. No entanto, a maioria das ferramentas citadas não são concorrentes e, sim, complementares, já que não desempenham o mesmo papel. Por exemplo, somente o Apache JMeter e o HP LoadRunner são ferramentas especializadas em testes de desempenho. Ferramentas como Padim e Hudson servem para automatizar outros aspectos envolvidos nessa configuração.

A configuração 3, semelhantemente à configuração 1, apresentou a ferramenta SoapUI como a mais frequente. Essa semelhança pode ser explicada por dois motivos: a) ambas as configurações estão focadas na tecnologia de *web services RESTful*; b) em ambas as configurações há somente duas empresas representadas. Outra ferramenta de destaque nessa configuração é o Jenkins, uma ferramenta de integração contínua que pode ser utilizada, por exemplo, para automatizar a execução de testes. Como essa configuração trata de testes de integração, uma ferramenta de integração contínua pode desempenhar um papel fundamental no sentido de facilitar o trabalho dos profissionais de teste em relação à execução dos testes.

Na configuração 4, observa-se uma predominância de ferramentas de gestão de testes: ferramentas que possibilitam tanto a documentação de planos, casos e procedimentos de teste, quanto a gestão de incidentes de teste. Três ferramentas citadas são usadas para edição e gestão de artefatos de teste (ALM-IBM, ALM-HP e TestLink), enquanto que o Mantis é uma ferramenta de *bugtracking* e o Wink é uma ferramenta usada para captura de evidências de teste na forma de vídeos.

Quanto à configuração 5, pode-se perceber que apresentou a maior variedade de ferramentas de teste, o que pode ser explicado pela maior diversidade de empresas de origem dos entrevistados. À exceção do TestLink, todas as demais ferramentas são usadas pelos participantes para implementação e execução de *scripts* de teste automatizados. Ferramentas como Selenium WebDriver, Cucumber e Phantom JS são específicas para testes de aplicações Web, enquanto que a ferramenta própria citada por um participante foi utilizada para automação de testes em um sistema *desktop*.

Figura 30: Ferramentas usadas pelos participantes do estudo por configuração.

5.4.2 Ranking de esforço demandado por atividade

As questões de pesquisa específicas deste estudo QE6 e QE6.1 referem-se ao esforço demandado por atividade do processo de teste. Para responder a essas questões, no início de cada entrevista, o participante era questionado pelo pesquisador sobre quais atividades do processo de teste, exercidas por ele, demandavam mais esforço. Assim, o participante era orientado a colocá-las em ordem, iniciando pela atividade que exigiu mais esforço em projetos da respectiva configuração de estratégia de teste.

A consolidação dos resultados, no entanto, revelou-se como um desafio. Inicialmente, tentou-se utilizar a Contagem de Borda, um método proposto pelo matemático francês Jean-Charles Borda em 1770, utilizado em processos eleitorais (CARDOSO, 2009). No entanto, o *ranking* resultante apresentava algumas distorções, decorrentes do fato que nem todas as atividades do processo de teste foram praticadas pelos participantes do estudo. Assim, as respostas de um participante que tivesse respondido a respeito de seis atividades teriam pesos diferentes das respostas de quem respondeu sobre três, quatro ou cinco atividades. O método de classificação que se mostrou mais justo, do ponto-de-vista do pesquisador, foi o *Gold First Method* (CHOI; CHOI, 2017), que é o método de classificação do quadro de medalhas em Jogos Olímpicos. Neste método, o primeiro colocado é aquele que possui mais indicações ao primeiro lugar ("medalha de ouro"), independentemente do total de indicações (ou medalhas). O número de indicações para as posições seguintes (por exemplo, medalhas de prata e bronze) é utilizado como critério de desempate. Portanto, com a aplicação desse critério, as atividades do processo de teste foram ranqueadas em função do esforço tipicamente demandado. A Tabela 45 apresenta o *ranking* resultante.

Em todas as seis configurações analisadas, a atividade de "Projetar e implementar testes" foi considerada a que demanda mais esforço. Os participantes justificavam essa percepção levando em conta o esforço cognitivo inerente à derivação de casos e procedimentos de teste a partir de requisitos, bem como, à implementação de *scripts* de teste automatizados, tarefas que requerem do profissional de testes raciocínio lógico e aplicação de técnicas que podem implicar em intensa atividade mental.

Na configuração 0, os participantes relataram praticar apenas três atividades, na seguinte ordem de esforço: "Projetar e implementar testes", "Configurar ambiente de teste" e "Executar testes". Quando questionados por que as demais atividades não foram citadas, os participantes respondiam dizendo que elas não são necessárias nessa

configuração. Provavelmente, essa visão é motivada pelo fato de que nessa configuração de teste o testador é o próprio programador que desenvolveu o código-fonte. Por exemplo, ao executar um teste e detectar uma falha, pode não fazer sentido para o programador/testador abrir um relato de incidente de teste para si mesmo.

As configurações 1 e 3 apresentaram a mesma ordem de atividades até a terceira posição, o que pode ser justificado pela semelhança entre as configurações (somente o nível de teste é diferente). A atividade "Executar testes" foi considerada a segunda que mais demanda esforço, o que chama a atenção pelo fato dos testes serem executados automaticamente.

A configuração 2 apresentou a diferença mais significativa em relação às demais. A atividade "Finalizar projeto de teste" foi classificada como a segunda que mais demanda esforço. Em outras configurações, essa atividade figura na última posição, quando praticada pelos profissionais entrevistados. No caso da configuração 2, a justificativa se deve ao fato de que a elaboração do relatório final do projeto de testes exige um trabalho intenso de análise dos resultados de testes de desempenho.

Já as configurações 4 e 5, apesar de só se distinguirem pela forma de execução dos testes, apresentaram ordens de atividades bem diferentes. Enquanto que na configuração 4 (execução manual) a atividade "Executar testes" foi classificada na segunda posição, tal atividade foi classificada somente em quarto lugar na configuração 5, já que nessa configuração a execução de testes é automatizada.

Tabela 45: Ranking de esforço (configuração de estratégia de teste x atividade do processo de teste).

Configuração	1º.	2º.	3º.	4º.	5º.	6º.
0	Projetar e implementar testes	Configurar ambiente de teste	Executar testes	-	-	-
1	Projetar e implementar testes	Executar testes	Configurar ambiente de teste	Comunicar incidentes de teste	Planejar teste	Finalizar projeto de teste
2	Projetar e implementar testes	Finalizar projeto de teste	Configurar ambiente de teste	Executar testes	Planejar teste	Comunicar incidentes de teste
3	Projetar e implementar testes	Executar testes	Configurar ambiente de teste	Planejar teste	Comunicar incidentes de teste	-
4	Projetar e implementar testes	Executar testes	Configurar ambiente de teste	Comunicar incidentes de teste	Planejar teste	Finalizar projeto de teste
5	Projetar e implementar testes	Configurar ambiente de teste	Planejar teste	Executar testes	Comunicar incidentes de teste	-

Portanto, diante do que está representado na Tabela 45, a questão de pesquisa QE6 pode ser respondida da seguinte forma: geralmente, a atividade do processo de teste que demanda mais esforço é "Projetar e implementar testes", independentemente da configuração de estratégia de teste. Quanto à questão QE6.1, os resultados sugerem que a ordem das atividades do processo de teste por esforço demandado varia de acordo com a configuração de estratégia de teste.

5.4.3 Fatores de esforço no processo de teste de software

A questão de pesquisa QE7, incluindo suas subquestões, refere-se aos fatores de esforço relacionados ao processo de teste de software. Para responder a essas questões, o participante precisava indicar no questionário QPET o impacto de cada fator previamente listado em cada atividade do processo de teste, de acordo com a configuração de estratégia de teste que estava respondendo. Cada interseção entre fator de esforço e atividade do processo recebia um julgamento do participante. Fatores com julgamentos assinalados com "+", "-" ou "+/-" são considerados "com impacto" para o esforço de teste. Fatores com julgamentos deixados em branco a respeito de atividades praticadas por cada participante são considerados "sem impacto" para o esforço de teste. Já os fatores com julgamentos deixados em branco a respeito de atividades não-praticadas por cada participante são classificados como "sem pertinência".

Considerando que o estudo contou com 30 pontos de vista sobre o esforço de teste (seis configurações de estratégia, cada qual com cinco respondentes), foram coletados 11.160 julgamentos (372 por ponto de vista). Esse total de julgamentos foi dividido da seguinte forma:

- 3.326 julgamentos com indicação de impacto ("+", "-" ou "+/-");
- 4.672 julgamentos sem indicação de impacto;
- 3.162 julgamentos sem pertinência.

Assim, considerando todos os julgamentos coletados com indicação de impacto, a Figura 31 representa sua distribuição por configuração de estratégia de teste e a Figura 32 representa a distribuição por atividade do processo de teste. Na Figura 31 percebe-se que a configuração 4 é aquela com mais julgamentos com algum tipo de impacto (718)

e a configuração 0 é a com menos julgamentos desse tipo (290). Essas diferenças são justificadas principalmente pela quantidade de atividades do processo praticadas pelos participantes de cada configuração: os participantes da configuração 0 responderam somente por três atividades, enquanto que todos os participantes da configuração 4 responderam pelas seis atividades do processo. Já na Figura 32 observa-se que a atividade com mais julgamentos com impacto foi "Projetar e implementar testes" (1093), reflexo da quantidade de fatores que influenciam o esforço dessa atividade. A com atividade com menos julgamentos com impacto foi "Finalizar projeto de testes" (80), o que pode ser explicado pelo fato de que poucos participantes responderam a respeito dessa atividade.

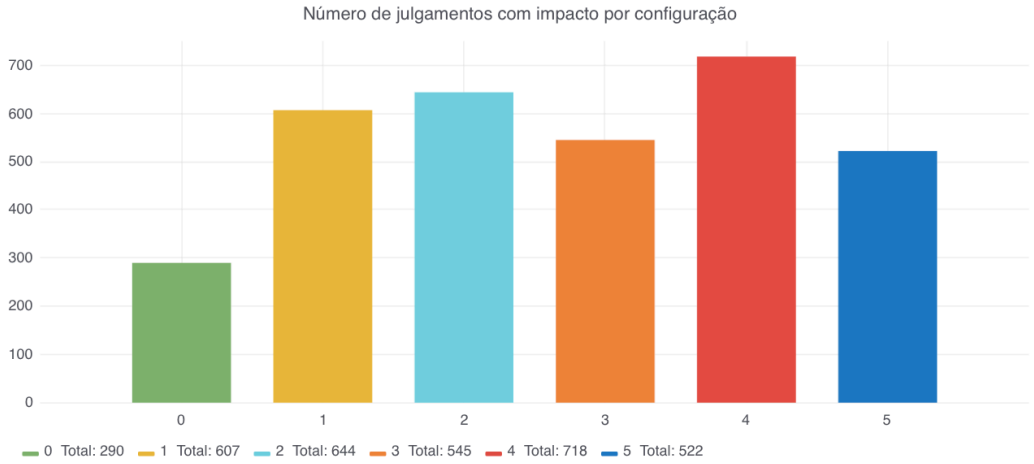


Figura 31: Distribuição dos julgamentos com impacto por configuração.

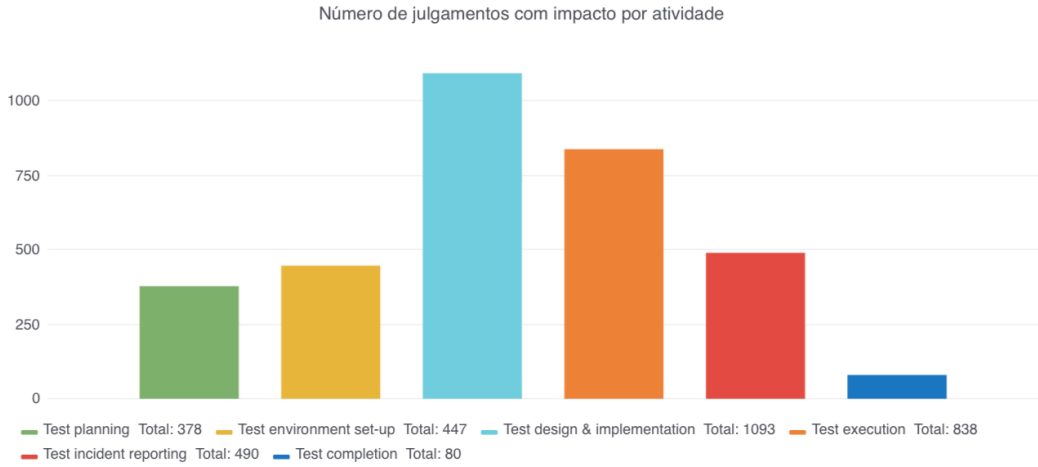


Figura 32: Distribuição dos julgamentos com impacto por atividade do processo.

A quantidade de fatores percebidos por cada participante do estudo em cada atividade do processo de teste pode ser influenciada por vários aspectos, como o

processo de teste específico da organização, os projetos vivenciados, a formação profissional, o tipo de tecnologia utilizada, o tempo de experiência, entre outros. Quanto ao tempo de experiência especificamente, pode-se supor que, eventualmente, profissionais com mais tempo de experiência podem ter um ponto de vista do esforço de teste composto por uma maior variedade de fatores de esforço, o que seria justificado pelo fato de terem hipoteticamente experimentado uma maior quantidade e variedade de situações em projetos de teste de software. A variedade de fatores percebidos por cada participante pode ser calculada em função do número de julgamentos informados que indicam algum tipo de impacto.

Para avaliar essa suposição, foi realizado um teste de correlação entre os valores de tempo de experiência dos participantes e a quantidade de diferentes (variedade) fatores de esforço percebidos. Tendo em vista que os valores dessas variáveis não seguem distribuições normais, foi aplicado o coeficiente de correlação de postos de Spearman. O teste foi realizado para as seis configurações de estratégia de testes tratadas no estudo, porém considerando apenas a atividade "Projetar e Implementar Testes", por conta dos seguintes motivos:

- juntamente com a atividade "Executar Testes", foi exercitada por todos os participantes das seis configurações;
- é a atividade considerada pelos participantes como a que demanda mais esforço entre as seis configurações;
- é a atividade que possui a maior quantidade total de julgamentos com impacto informado.

A Figura 33 representa, por configuração de estratégia de testes, os coeficientes de correlação de Spearman entre o tempo de experiência dos participantes e a quantidade de diferentes fatores de esforço percebidos referentes à atividade "Projetar e Implementar Testes". Como pode ser observado na Figura 33, não é possível afirmar que há correlação entre as duas variáveis avaliadas, mesmo para a configuração 3 que apresentou um coeficiente de correlação consideravelmente maior que nas demais configurações (-0.78). Levando-se em conta que o tamanho da amosta em cada grupo é relativamente pequeno (N=5), para representar um grau estatisticamente significativo de correlação entre as duas variáveis, a um nível de confiança de 95%, seria necessário que os coeficientes calculados fossem muito próximos a 1.0 ou -1.0 (TRIOLA, 2008).

Portanto, não se pode afirmar que o tempo de experiência dos participantes está correlacionado com a quantidade de diferentes fatores percebidos.

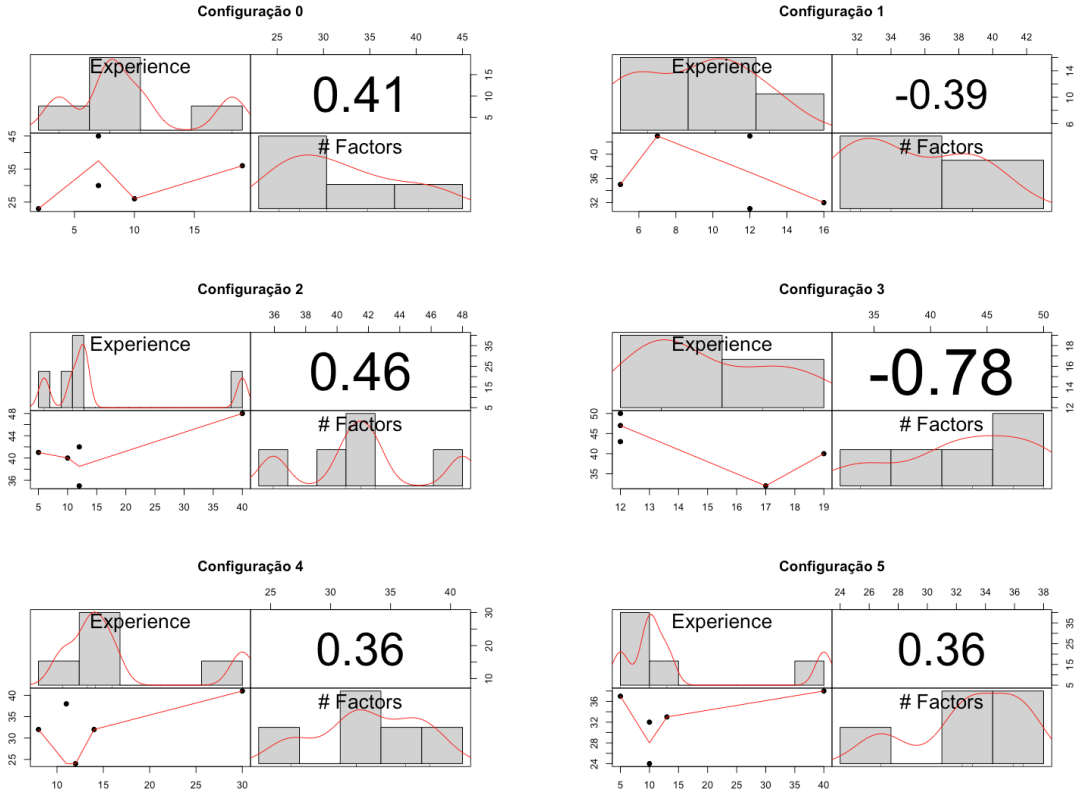


Figura 33: Correlação entre a experiência dos participantes e a quantidade de fatores percebidos.

A Tabela 46 apresenta os fatores mais frequentemente julgados com algum tipo de impacto, seja "Positivo", "Negativo" ou "Positivo e negativo", agregando-se as respostas de todas as configurações de estratégia de teste do estudo. Como se pode observar, fatores relacionados à experiência da equipe, como "Experiência com teste" e "Experiência com ferramentas de teste" são os frequentes. Outro aspecto interessante é que os quatro fatores com maior frequência de julgamentos com impacto são inerentes à equipe de teste, e todos os quatro apresentaram impacto negativo sobre o esforço.

Dentre os fatores listados na Tabela 46, 16 foram julgados com impactos diferentes, indicados com o sinal "+/-". Nesses casos, pelo menos um participante indicou um impacto diferente dos demais participantes em alguma atividade. Por exemplo, dado um fator de esforço, se um participante tivesse julgado o seu impacto como "Positivo" ou "Positivo e negativo" e todos os demais participantes tivessem julgado com impacto "Negativo", o sinal representado nessa tabela seria "+/-".

Tabela 46: Julgamentos com impacto informado.

Mnemônico	Fator de esforço	Impacto	Frequência
EXT	Experiência com teste	-	113
ETT	Experiência com ferramentas de teste	-	112
TCA	Capacidade da equipe	-	106
TCO	Cooperação da equipe	-	102
STT	Suporte de ferramentas de teste	-	100
ADI	Quantidade e duração de interrupções	+	98
TCN	Continuidade da equipe	-	94
EAD	Experiência com o domínio de aplicação	-	88
REI	Instabilidade dos requisitos	+	86
SYC	Complexidade do sistema	+	84
TEP	Produtividade da equipe	+/-	84
SCP	Pressão sobre o cronograma	+/-	82
RTC	Cobertura de teste requerida	+	80
DOQ	Qualidade da documentação	-	78
NTC	Número de casos de teste	+	77
CTE	Complexidade do ambiente de teste	+	74
PTC	Comprometimento da equipe do projeto	+/-	72
REC	Complexidade dos requisitos	+	71
UIR	Indisponibilidade de recursos de infraestrutura	+	67
RSS	Tamanho da especificação de requisitos	+	65
TTS	Tamanho da equipe de teste	+/-	65
TPC	Capacidade do processo de teste	-	63
TEC	Complexidade de execução de teste	+	62
PTD	Distribuição física da equipe do projeto	+/-	62
UTE	Indisponibilidade do ambiente de teste	+	61
TPT	Tarefas do processo de teste	+/-	59
STL	Nível de testabilidade do software	+/-	59
ARA	Disponibilidade de artefatos de teste reusáveis	+/-	59
SIC	Complexidade das interfaces do software	+	58
ACD	Diversidade de atores	+	57
QTI	Quantidade de incidentes de teste	+	56
NWS	Número de Web Services	+	55
EOE	Experiência com o ambiente operacional	-	51
NOR	Número de versões	+	51
TET	Ciclos de teste	+/-	50
TSC	Complexidade dos *scripts* de teste	+	50
LOC	Nível de criticidade	+	49
NTS	Número de suítes de teste	+/-	49
SPA	Procedimentos específicos para acessar o sistema	+	49
TDQ	Quantidade de dados de teste	+/-	47
TSZ	Tamanho dos *scripts* de teste	+	47
NAS	Número de asserções por passo	+	45
EPT	Experiência com a tecnologia de programação	-	41
TSS	Tamanho da especificação técnica	+/-	38
RLS	Nível de segurança requerido	+	38
RLI	Nível de interoperabilidade requerido	+	34
UII	Instabilidade da interface gráfica com o usuário	+	34
NPU	Número de parâmetros por unidade	+	30
RER	Nível de confiabilidade requerido	+/-	24
PLC	Nível de confidencialidade do projeto	+	20
REP	Nível de desempenho requerido	+	17
RCE	Coexistência requerida	+	16
SCC	Complexidade do código-fonte	+	16
RLP	Nível de portabilidade requerido	+	15
ROC	Legibilidade do código	+/-	14
SCS	Tamanho do código-fonte	+	11
RLU	Nível de usabilidade requerido	+	10
PLU	Nível de usabilidade apresentado	+/-	10
RDC	Capacidade de dados requerida	+	9
LFT	Falta de fluência no idioma do projeto	+	6
RLR	Nível de recuperabilidade requerido	+/-	6

A Tabela 47 apresenta os fatores mais frequentemente julgados como sem impacto, isto é, que não afetam o esforço de teste nas configurações de estratégia de teste cobertas pelo estudo. Pode-se observar que os fatores "Nível de segurança física pessoal requerida" e "Falta de fluência no idioma no projeto" foram os dois mais frequentes.

Tabela 47: Julgamentos sem impacto informado.

Mnemônico	Fator de esforço	Frequência
LOS	Nível de segurança física pessoal requerida	129
LFT	Falta de fluência no idioma do projeto	123
RLR	Nível de recuperabilidade requerido	123
RDC	Capacidade de dados requerida	120
PLU	Nível de usabilidade apresentado	119
RLU	Nível de usabilidade requerido	119
SCS	Tamanho do código-fonte	118
ROC	Legibilidade do código	115
RLP	Nível de portabilidade requerido	114
SCC	Complexidade do código-fonte	113
RCE	Coexistência requerida	113
REP	Nível de desempenho requerido	112
PLC	Nível de confidencialidade do projeto	109
RER	Nível de confiabilidade requerido	105
NPU	Número de parâmetros por unidade	99
UII	Instabilidade da interface gráfica com o usuário	95
RLI	Nível de interoperabilidade requerido	95
TSS	Tamanho da especificação técnica	91
RLS	Nível de segurança requerido	91
EPT	Experiência com a tecnologia de programação	88
NAS	Número de asserções por passo	84
TDQ	Quantidade de dados de teste	82
TSZ	Tamanho dos *scripts* de teste	82
LOC	Nível de criticidade	80
NTS	Número de suítes de teste	80
SPA	Procedimentos específicos para acessar o sistema	80
TSC	Complexidade dos *scripts* de teste	79
TET	Ciclos de teste	79
EOE	Experiência com o ambiente operacional	78
NOR	Número de versões	78
NWS	Número de Web Services	74
QTI	Quantidade de incidentes de teste	73
ACD	Diversidade de atores	72
SIC	Complexidade das interfaces do software	71
TPT	Tarefas do processo de teste	70
ARA	Disponibilidade de artefatos de teste reusáveis	70
STL	Nível de testabilidade do software	70
UTE	Indisponibilidade do ambiente de teste	68
PTD	Distribuição física da equipe do projeto	67
TEC	Complexidade de execução de teste	67
TPC	Capacidade do processo de teste	66
TTS	Tamanho da equipe de teste	64
RSS	Tamanho da especificação de requisitos	64
UIR	Indisponibilidade de recursos de infraestrutura	62
REC	Complexidade dos requisitos	58
PTC	Comprometimento da equipe do projeto	57
CTE	Complexidade do ambiente de teste	55
NTC	Número de casos de teste	52
DOQ	Qualidade da documentação	51
RTC	Cobertura de teste requerida	49
SCP	Pressão sobre o cronograma	47
SYC	Complexidade do sistema	45
TEP	Produtividade da equipe	45
REI	Instabilidade dos requisitos	43
EAD	Experiência com o domínio de aplicação	41
TCN	Continuidade da equipe	35
ADI	Quantidade e duração de interrupções	31
STT	Suporte de ferramentas de teste	29
TCO	Cooperação da equipe	27
TCA	Capacidade da equipe	23
ETT	Experiência com ferramentas de teste	17
EXT	Experiência com teste	16

5.4.3.1 Configuração 0

Como mostra a Tabela 48, a configuração de estratégia de teste "0" é formada pelo teste em nível de unidade, do tipo funcional, utilizando a técnica de teste baseado em estrutura, com a execução automatizada.

Tabela 48: Dimensões da configuração de estratégia de teste 0.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de unidade	Teste funcional	Teste baseado em estrutura	Automatizada

A Figura 34 representa a frequência de julgamentos por fator de esforço na configuração 0, agregando as respostas sobre todas as atividades. Os fatores com mais julgamentos foram "Experiência com ferramentas de teste" (ETT) e "Suporte de ferramentas de teste" (STT), ambos com 13 ocorrências. Esses resultados podem sugerir que trata-se de uma configuração de estratégia de teste mais dependente de ferramentas, uma vez que a forma de execução é automatizada.

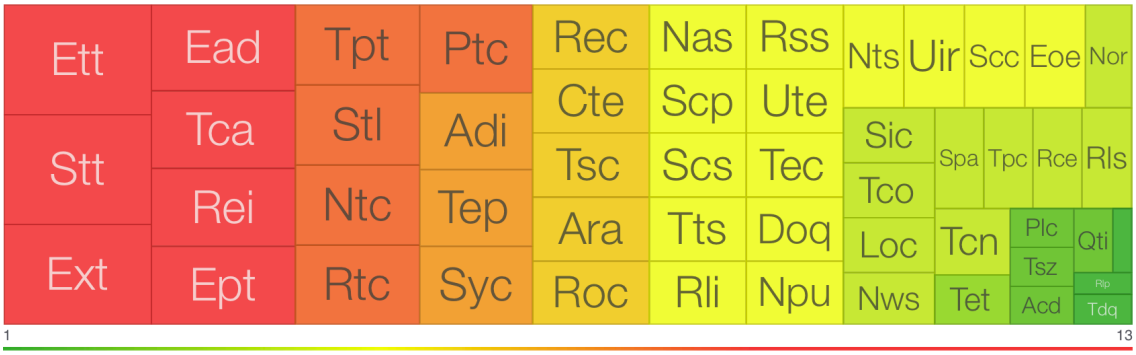


Figura 34: Frequência de julgamentos por fator de esforço na configuração 0.

Para essa configuração, somente três atividades do processo de teste foram contempladas nas respostas dos participantes: "Configurar ambiente de teste", "Projetar e implementar testes" e "Executar testes". A frequência de julgamentos por fator de esforço nessas atividades está representada, respectivamente, na Tabela 49, Tabela 50 e Tabela 51.

Tabela 49: Julgamentos da atividade "Configurar Ambiente de Teste" na configuração 0.

Mnemônico	Fator de esforço	Impacto	Frequência
EXT	Experiência com teste	-	3
ETT	Experiência com ferramentas de teste	-	3
PTC	Comprometimento da equipe do projeto	-	3
STT	Suporte de ferramentas de teste	-	3
CTE	Complexidade do ambiente de teste	+	3
TCA	Capacidade da equipe	-	2
TEP	Produtividade da equipe	-	2
EPT	Experiência com a tecnologia de programação	-	2
EAD	Experiência com o domínio de aplicação	-	2
EOE	Experiência com o ambiente operacional	-	2
UIR	Indisponibilidade de recursos de infraestrutura	+	2
ADI	Quantidade e duração de interrupções	+	2
TCO	Cooperação da equipe	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
STL	Nível de testabilidade do software	-	1
DOQ	Qualidade da documentação	-	1
NTS	Número de suítes de teste	+	1
REP	Nível de desempenho requerido	+	1
REI	Instabilidade dos requisitos	+	1
TSC	Complexidade dos scripts de teste	+	1
RLP	Nível de portabilidade requerido	+	1
SYC	Complexidade do sistema	+	1
NOR	Número de versões	+	1
RLS	Nível de segurança requerido	+	1
TPT	Tarefas do processo de teste	+	1
SCP	Pressão sobre o cronograma	+	1
SPA	Proced. específicos para acessar o sistema	+	1
RCE	Coexistência requerida	+	1
UTE	Indisponibilidade do ambiente de teste	+	1
RLI	Nível de interoperabilidade requerido	+	1

Tabela 50: Julgamentos da atividade "Projetar e Implementar Testes" na configuração 0.

Mnemônico	Fator de esforço	Impacto	Frequência
EAD	Experiência com o domínio de aplicação	-	5
EPT	Experiência com a tecnologia de programação	-	5
EXT	Experiência com teste	-	5
STT	Suporte de ferramentas de teste	-	5
ETT	Experiência com ferramentas de teste	-	5
RTC	Nível de cobertura requerido	+	5
NPU	Número de parâmetros por unidade	+	5
REC	Complexidade dos requisitos	+	5
SCC	Complexidade do código-fonte	+	5
TSC	Complexidade dos scripts de teste	+	5
REI	Instabilidade dos requisitos	+	5
NTC	Número de casos de teste	+	5
NAS	Número de asserções por passo	+	5
SYC	Complexidade do sistema	+	5
ROC	Legibilidade do código	+/-	5
PTC	Comprometimento da equipe do projeto	-	4
TCA	Capacidade da equipe	-	4
ARA	Disponibilidade de artefatos de teste reusáveis	-	4
ADI	Quantidade e duração de interrupções	+	4
RSS	Tamanho da especificação de requisitos	+	4
SCS	Tamanho do código-fonte	+	4
NTS	Número de suítes de teste	+	4
SIC	Complexidade das interfaces do software	+	4
STL	Nível de testabilidade do software	+/-	4
TPC	Capacidade do processo de teste	-	3
TCN	Continuidade da equipe	-	3
TCO	Cooperação da equipe	-	3
DOQ	Qualidade da documentação	-	3
TEP	Produtividade da equipe	-	3
TPT	Tarefas do processo de teste	+	3
NWS	Número de Web Services	+	3
RLI	Nível de interoperabilidade requerido	+	3
TTS	Tamanho da equipe de teste	+/-	3
SCP	Pressão sobre o cronograma	+/-	3
SPA	Proced. específicos para acessar o sistema	+	2
LOC	Nível de criticidade	+	2
RCE	Coexistência requerida	+	2
ACD	Diversidade de atores	+	2
RLS	Nível de segurança requerido	+	2
EOE	Experiência com o ambiente operacional	-	1
TET	Ciclos de teste	+	1
NOR	Número de versões	+	1
QTI	Quantidade de incidentes de teste	+	1
CTE	Complexidade do ambiente de teste	+	1
PLC	Nível de confidencialidade do projeto	+	1
TEC	Complexidade de execução de teste	+	1
TDQ	Quantidade de dados de teste	+	1
TSZ	Tamanho dos scripts de teste	+	1

Tabela 51: Julgamentos da atividade "Executar Testes" na configuração 0.

Mnemônico	Fator de esforço	Impacto	Frequência
ETT	Experiência com ferramentas de teste	-	5
STT	Suporte de ferramentas de teste	-	5
EXT	Experiência com teste	-	4
TEC	Complexidade de execução de teste	+	4
TPT	Tarefas do processo de teste	+	4
UTE	Indisponibilidade do ambiente de teste	+	4
EAD	Experiência com o domínio de aplicação	-	3
TCA	Capacidade da equipe	-	3
REI	Instabilidade dos requisitos	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
NTC	Número de casos de teste	+	3
RTC	Nível de cobertura requerido	+	3
STL	Nível de testabilidade do software	+/-	3
TEP	Produtividade da equipe	-	2
EOE	Experiência com o ambiente operacional	-	2
EPT	Experiência com a tecnologia de programação	-	2
CTE	Complexidade do ambiente de teste	+	2
TET	Ciclos de teste	+	2
LOC	Nível de criticidade	+	2
NOR	Número de versões	+	2
TTS	Tamanho da equipe de teste	+/-	2
DOQ	Qualidade da documentação	-	1
ROC	Legibilidade do código	-	1
PTC	Comprometimento da equipe do projeto	-	1
TPC	Capacidade do processo de teste	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
TCN	Continuidade da equipe	-	1
REC	Complexidade dos requisitos	+	1
TSZ	Tamanho dos scripts de teste	+	1
RCE	Coexistência requerida	+	1
RSS	Tamanho da especificação de requisitos	+	1
RLI	Nível de interoperabilidade requerido	+	1
QTI	Quantidade de incidentes de teste	+	1
RLS	Nível de segurança requerido	+	1
SCP	Pressão sobre o cronograma	+	1
SYC	Complexidade do sistema	+	1
NWS	Número de Web Services	+	1
SCS	Tamanho do código-fonte	+	1
ADI	Quantidade e duração de interrupções	+	1
PLC	Nível de confidencialidade do projeto	+	1
SPA	Proced. específicos para acessar o sistema	+	1

5.4.3.2 Configuração 1

Como mostra a Tabela 52, a configuração de estratégia de teste "1" é formada pelo teste em nível de unidade, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução automatizada.

Tabela 52: Dimensões da configuração de estratégia de teste 1.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de unidade	Teste funcional	Teste baseado em especificação	Automatizada

A Figura 35 representa a frequência de julgamentos por fator de esforço na configuração 1, agregando as respostas sobre todas as atividades. O fator com maior frequência de julgamentos foi "Capacidade do processo de teste" (TCA), com 23 ocorrências, seguido de "Experiência com ferramentas de teste" (ETT) e "Experiência com teste" (EXT) com 22 e 21 ocorrências, respectivamente.

Figura 35: Frequência de julgamentos por fator de esforço na configuração 1.

Todas as seis atividades do processo de teste foram contempladas nas respostas dos participantes da configuração 1. As tabelas 53 a 58 representa a frequência de julgamentos por fator de esforço nas respectivas atividades.

Tabela 53: Julgamentos da atividade "Planejar Teste" na configuração 1.

Mnemônico	Fator de esforço	Impacto	Frequência
TPC	Capacidade do processo de teste	-	4
TCN	Continuidade da equipe	-	4
ARA	Disponibilidade de artefatos de teste reusáveis	-	4
DOQ	Qualidade da documentação	-	4
EXT	Experiência com teste	-	4
EAD	Experiência com o domínio de aplicação	-	4
TCA	Capacidade da equipe	-	4
SCP	Pressão sobre o cronograma	+	4
TCO	Cooperação da equipe	-	3
TTS	Tamanho da equipe de teste	-	3
TEP	Produtividade da equipe	-	3
ETT	Experiência com ferramentas de teste	-	3
ADI	Quantidade e duração de interrupções	+	3
REC	Complexidade dos requisitos	+	3
RSS	Tamanho da especificação de requisitos	+	3
REI	Instabilidade dos requisitos	+	3
PTD	Distribuição física da equipe do projeto	+	3
SYC	Complexidade do sistema	+	3
RTC	Nível de cobertura requerido	+	3
PTC	Comprometimento da equipe do projeto	+/-	3
STT	Suporte de ferramentas de teste	-	2
TPT	Tarefas do processo de teste	+	2
CTE	Complexidade do ambiente de teste	+	2
NTC	Número de casos de teste	+	2
NWS	Número de Web Services	+	2
SIC	Complexidade das interfaces do software	+	2
EOE	Experiência com o ambiente operacional	-	1
EPT	Experiência com a tecnologia de programação	-	1
TSS	Tamanho da especificação técnica	+	1
LOC	Nível de criticidade	+	1
TDQ	Quantidade de dados de teste	+	1
ACD	Diversidade de atores	+	1
TEC	Complexidade de execução de teste	+	1

Tabela 54: Julgamentos da atividade "Configurar Ambiente de Teste" na configuração 1.

Mnemônico	Fator de esforço	Impacto	Frequência
EXT	Experiência com teste	-	3
ETT	Experiência com ferramentas de teste	-	3
TCA	Capacidade da equipe	-	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
SPA	Proced. específicos para acessar o sistema	+	3
TDQ	Quantidade de dados de teste	+	3
SYC	Complexidade do sistema	+	3
ADI	Quantidade e duração de interrupções	+	3
NOR	Número de versões	+	3
TTS	Tamanho da equipe de teste	-	2
TEP	Produtividade da equipe	-	2
ARA	Disponibilidade de artefatos de teste reusáveis	-	2
STT	Suporte de ferramentas de teste	-	2
TPC	Capacidade do processo de teste	-	2
STL	Nível de testabilidade do software	-	2
PTC	Comprometimento da equipe do projeto	-	2
EAD	Experiência com o domínio de aplicação	-	2
TCN	Continuidade da equipe	-	2
TCO	Cooperação da equipe	-	2
SIC	Complexidade das interfaces do software	+	2
NWS	Número de Web Services	+	2
CTE	Complexidade do ambiente de teste	+	2
REC	Complexidade dos requisitos	+	2
RLI	Nível de interoperabilidade requerido	+	2
REI	Instabilidade dos requisitos	+	2
TSZ	Tamanho dos scripts de teste	+	2
SCP	Pressão sobre o cronograma	+	2
NTC	Número de casos de teste	+	2
EOE	Experiência com o ambiente operacional	-	1
DOQ	Qualidade da documentação	-	1
EPT	Experiência com a tecnologia de programação	-	1
TSS	Tamanho da especificação técnica	+	1
ACD	Diversidade de atores	+	1
PTD	Distribuição física da equipe do projeto	+	1
NTS	Número de suítes de teste	+	1
RTC	Nível de cobertura requerido	+	1
TEC	Complexidade de execução de teste	+	1
TSC	Complexidade dos scripts de teste	+	1
RLS	Nível de segurança requerido	+	1
TPT	Tarefas do processo de teste	+	1

Tabela 55: Julgamentos da atividade "Projetar e Implementar Testes" na configuração 1.

Mnemônico	Fator de esforço	Impacto	Frequência
STT	Suporte de ferramentas de teste	-	5
ARA	Disponibilidade de artefatos de teste reusáveis	-	5
TCA	Capacidade da equipe	-	5
DOQ	Qualidade da documentação	-	5
EXT	Experiência com teste	-	5
ETT	Experiência com ferramentas de teste	-	5
ADI	Quantidade e duração de interrupções	+	5
TSC	Complexidade dos scripts de teste	+	5
RTC	Nível de cobertura requerido	+	5
NAS	Número de asserções por passo	+	5
REC	Complexidade dos requisitos	+	5
TDQ	Quantidade de dados de teste	+	5
SYC	Complexidade do sistema	+	5
NTC	Número de casos de teste	+	5
REI	Instabilidade dos requisitos	+	5
NWS	Número de Web Services	+	5
TSZ	Tamanho dos scripts de teste	+	5
SIC	Complexidade das interfaces do software	+	5
TTS	Tamanho da equipe de teste	+/-	5
PTC	Comprometimento da equipe do projeto	+/-	5
TEP	Produtividade da equipe	-	4
EAD	Experiência com o domínio de aplicação	-	4
EPT	Experiência com a tecnologia de programação	-	4
TCN	Continuidade da equipe	-	4
TCO	Cooperação da equipe	-	4
TSS	Tamanho da especificação técnica	+	4
CTE	Complexidade do ambiente de teste	+	4
RSS	Tamanho da especificação de requisitos	+	4
TEC	Complexidade de execução de teste	+	4
SCP	Pressão sobre o cronograma	+	4
NPU	Número de parâmetros por unidade	+	4
TPC	Capacidade do processo de teste	-	3
SPA	Proced. específicos para acessar o sistema	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
NTS	Número de suítes de teste	+	3
LOC	Nível de criticidade	+	3
RLS	Nível de segurança requerido	+	3
ACD	Diversidade de atores	+	3
PTD	Distribuição física da equipe do projeto	+	3
NOR	Número de versões	+	3
STL	Nível de testabilidade do software	-	2
TPT	Tarefas do processo de teste	+	2
QTI	Quantidade de incidentes de teste	+	2
UTE	Indisponibilidade do ambiente de teste	+	2
RLI	Nível de interoperabilidade requerido	+	2
UII	Instabilidade da interface gráfica com o usuário	+	2
EOE	Experiência com o ambiente operacional	-	1

Tabela 56: Julgamentos da atividade "Executar Testes" na configuração 1.

Mnemônico	Fator de esforço	Impacto	Frequência
STT	Suporte de ferramentas de teste	-	5
TCA	Capacidade da equipe	-	5
STL	Nível de testabilidade do software	-	5
TCO	Cooperação da equipe	-	5
ETT	Experiência com ferramentas de teste	-	5
RTC	Nível de cobertura requerido	+	5
NTC	Número de casos de teste	+	5
NWS	Número de Web Services	+	5
SCP	Pressão sobre o cronograma	+	5
TTS	Tamanho da equipe de teste	+/-	5
EXT	Experiência com teste	-	4
TEP	Produtividade da equipe	-	4
EAD	Experiência com o domínio de aplicação	-	4
TET	Ciclos de teste	+	4
NOR	Número de versões	+	4
UIR	Indisponibilidade de recursos de infraestrutura	+	4
UTE	Indisponibilidade do ambiente de teste	+	4
TEC	Complexidade de execução de teste	+	4
REI	Instabilidade dos requisitos	+	4
QTI	Quantidade de incidentes de teste	+	4
TPC	Capacidade do processo de teste	-	3
TCN	Continuidade da equipe	-	3
ADI	Quantidade e duração de interrupções	+	3
CTE	Complexidade do ambiente de teste	+	3
RLI	Nível de interoperabilidade requerido	+	3
SPA	Proced. específicos para acessar o sistema	+	3
PTC	Comprometimento da equipe do projeto	-	2
TDQ	Quantidade de dados de teste	+	2
TSC	Complexidade dos scripts de teste	+	2
REC	Complexidade dos requisitos	+	2
SYC	Complexidade do sistema	+	2
RSS	Tamanho da especificação de requisitos	+	2
LOC	Nível de criticidade	+	2
TSZ	Tamanho dos scripts de teste	+	2
TPT	Tarefas do processo de teste	+	2
PTD	Distribuição física da equipe do projeto	+	2
TSS	Tamanho da especificação técnica	+	2
SIC	Complexidade das interfaces do software	+	2
ACD	Diversidade de atores	+	2
UII	Instabilidade da interface gráfica com o usuário	+	2
NPU	Número de parâmetros por unidade	+	2
EOE	Experiência com o ambiente operacional	-	1
EPT	Experiência com a tecnologia de programação	-	1
DOQ	Qualidade da documentação	-	1
NAS	Número de asserções por passo	+	1
RCE	Coexistência requerida	+	1
SCC	Complexidade do código-fonte	+	1
NTS	Número de suítes de teste	+	1

Tabela 57: Julgamentos da atividade "Comunicar Incidentes de Teste" na configuração 1.

Mnemônico	Fator de esforço	Impacto	Frequência
ETT	Experiência com ferramentas de teste	-	5
TCA	Capacidade da equipe	-	5
TCO	Cooperação da equipe	-	5
SCP	Pressão sobre o cronograma	+	5
STT	Suporte de ferramentas de teste	-	4
EXT	Experiência com teste	-	4
TPC	Capacidade do processo de teste	-	4
EAD	Experiência com o domínio de aplicação	-	4
TCN	Continuidade da equipe	-	4
PTC	Comprometimento da equipe do projeto	-	4
QTI	Quantidade de incidentes de teste	+	4
TTS	Tamanho da equipe de teste	+/-	4
DOQ	Qualidade da documentação	-	3
ARA	Disponibilidade de artefatos de teste reusáveis	-	3
STL	Nível de testabilidade do software	-	3
TEP	Produtividade da equipe	-	3
NAS	Número de asserções por passo	+	3
PTD	Distribuição física da equipe do projeto	+	3
CTE	Complexidade do ambiente de teste	+	2
NOR	Número de versões	+	2
RTC	Nível de cobertura requerido	+	2
TDQ	Quantidade de dados de teste	+	2
SYC	Complexidade do sistema	+	2
TEC	Complexidade de execução de teste	+	2
UTE	Indisponibilidade do ambiente de teste	+	2
NPU	Número de parâmetros por unidade	+	2
NTC	Número de casos de teste	+	2
ACD	Diversidade de atores	+	2
TPT	Tarefas do processo de teste	+	2
EOE	Experiência com o ambiente operacional	-	1
NWS	Número de Web Services	+	1
SIC	Complexidade das interfaces do software	+	1
UII	Instabilidade da interface gráfica com o usuário	+	1
TSS	Tamanho da especificação técnica	+	1
TET	Ciclos de teste	+	1
LOC	Nível de criticidade	+	1
REC	Complexidade dos requisitos	+	1
RLI	Nível de interoperabilidade requerido	+	1
ADI	Quantidade e duração de interrupções	+	1
RSS	Tamanho da especificação de requisitos	+	1
RLS	Nível de segurança requerido	+	1
UIR	Indisponibilidade de recursos de infraestrutura	+	1
REI	Instabilidade dos requisitos	+	1
TSZ	Tamanho dos scripts de teste	+	1

Tabela 58: Julgamentos da atividade "Finalizar Projeto de Teste" na configuração 1.

Mnemônico	Fator de esforço	Impacto	Frequência
ETT	Experiência com ferramentas de teste	-	1
STT	Suporte de ferramentas de teste	-	1
TCA	Capacidade da equipe	-	1
TPC	Capacidade do processo de teste	-	1
EXT	Experiência com teste	-	1
RLS	Nível de segurança requerido	+	1
TET	Ciclos de teste	+	1

5.4.3.3 Configuração 2

A configuração de estratégia de teste "2" é formada pelo teste em nível de unidade, do tipo desempenho, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como representa a Tabela 59.

Tabela 59: Dimensões da configuração de estratégia de teste 2.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de unidade	Teste de desempenho	Teste baseado em especificação	Automatizada

A Figura 36 representa a frequência de julgamentos por fator de esforço na configuração 2, agregando todas as atividades. Os fatores com maior frequência de julgamentos foram "Cooperação da equipe" (TCO) e "Produtividade da equipe" (TEP), com 24 e 23 ocorrências, respectivamente. É importante enfatizar que esses dois fatores, bem como "Continuidade da equipe" (TCN), "Capacidade da equipe" (TCA), "Experiêcia com ferramentas de teste" (ETT) e "Experiência com teste" (EXT), que ficaram entre os mais frequentes, são da mesma categoria de fatores: equipe de teste.

Figura 36: Frequência de julgamentos por fator de esforço na configuração 2.

Todas as seis atividades do processo foram citadas por pelo menos um participante desse grupo. As tabelas 60 a 65 representam a frequência de julgamentos por fator de esforço em cada atividade.

Tabela 60: Julgamentos da atividade "Planejar Teste" na configuração 2.

Mnemônico	Fator de esforço	Impacto	Frequência
DOQ	Qualidade da documentação	-	4
TCO	Cooperação da equipe	-	4
TEP	Produtividade da equipe	-	4
REC	Complexidade dos requisitos	+	4
RSS	Tamanho da especificação de requisitos	+	4
TSS	Tamanho da especificação técnica	+	4
NWS	Número de Web Services	+	4
EXT	Experiência com teste	-	3
TCA	Capacidade da equipe	-	3
TCN	Continuidade da equipe	-	3
EAD	Experiência com o domínio de aplicação	-	3
SYC	Complexidade do sistema	+	3
REI	Instabilidade dos requisitos	+	3
ACD	Diversidade de atores	+	3
ADI	Quantidade e duração de interrupções	+	3
RER	Nível de confiabilidade requerido	+	3
SIC	Complexidade das interfaces do software	+	3
RLS	Nível de segurança requerido	+	3
NTC	Número de casos de teste	+	3
RTC	Nível de cobertura requerido	+	3
EPT	Experiência com a tecnologia de programação	-	2
ETT	Experiência com ferramentas de teste	-	2
PTC	Comprometimento da equipe do projeto	-	2
STT	Suporte de ferramentas de teste	-	2
TPT	Tarefas do processo de teste	+	2
RCE	Coexistência requerida	+	2
TET	Ciclos de teste	+	2
RLI	Nível de interoperabilidade requerido	+	2
SCP	Pressão sobre o cronograma	+	2
PTD	Distribuição física da equipe do projeto	+	2
SPA	Proced. específicos para acessar o sistema	+	2
CTE	Complexidade do ambiente de teste	+	2
TPC	Capacidade do processo de teste	-	1
TDQ	Quantidade de dados de teste	+	1
TSZ	Tamanho dos scripts de teste	+	1
REP	Nível de desempenho requerido	+	1
NPU	Número de parâmetros por unidade	+	1
LOC	Nível de criticidade	+	1
RLR	Nível de recuperabilidade requerido	+	1

Tabela 61: Julgamentos da atividade "Configurar Ambiente de Teste" na configuração 2.

Mnemônico	Fator de esforço	Impacto	Frequência
TCO	Cooperação da equipe	-	4
ETT	Experiência com ferramentas de teste	-	4
TCA	Capacidade da equipe	-	4
REP	Nível de desempenho requerido	+	4
TEP	Produtividade da equipe	+/-	4
EXT	Experiência com teste	-	3
PTC	Comprometimento da equipe do projeto	-	3
TCN	Continuidade da equipe	-	3
CTE	Complexidade do ambiente de teste	+	3
SCP	Pressão sobre o cronograma	+	3
TDQ	Quantidade de dados de teste	+	3
ADI	Quantidade e duração de interrupções	+	3
UTE	Indisponibilidade do ambiente de teste	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
EOE	Experiência com o ambiente operacional	-	2
STT	Suporte de ferramentas de teste	-	2
EAD	Experiência com o domínio de aplicação	-	2
ARA	Disponibilidade de artefatos de teste reusáveis	-	2
DOQ	Qualidade da documentação	-	2
SPA	Proced. específicos para acessar o sistema	+	2
PTD	Distribuição física da equipe do projeto	+	2
TPT	Tarefas do processo de teste	+	2
SYC	Complexidade do sistema	+	2
NOR	Número de versões	+	2
RLS	Nível de segurança requerido	+	2
TTS	Tamanho da equipe de teste	-	1
EPT	Experiência com a tecnologia de programação	-	1
TPC	Capacidade do processo de teste	-	1
RLR	Nível de recuperabilidade requerido	+	1
TSC	Complexidade dos scripts de teste	+	1
RER	Nível de confiabilidade requerido	+	1
SIC	Complexidade das interfaces do software	+	1
UII	Instabilidade da interface gráfica com o usuário	+	1
ACD	Diversidade de atores	+	1
RCE	Coexistência requerida	+	1
RDC	Capacidade de dados requerida	+	1
TET	Ciclos de teste	+	1
RLI	Nível de interoperabilidade requerido	+	1
RLP	Nível de portabilidade requerido	+	1

Tabela 62: Julgamentos da atividade "Projetar e Implementar Testes" na configuração 2.

Mnemônico	Fator de esforço	Impacto	Frequência
TCO	Cooperação da equipe	-	5
DOQ	Qualidade da documentação	-	5
EXT	Experiência com teste	-	5
ETT	Experiência com ferramentas de teste	-	5
TCA	Capacidade da equipe	-	5
TCN	Continuidade da equipe	-	5
NTC	Número de casos de teste	+	5
RTC	Nível de cobertura requerido	+	5
TSS	Tamanho da especificação técnica	+	5
NWS	Número de Web Services	+	5
TSC	Complexidade dos scripts de teste	+	5
REC	Complexidade dos requisitos	+	5
RSS	Tamanho da especificação de requisitos	+	5
TSZ	Tamanho dos scripts de teste	+	5
REI	Instabilidade dos requisitos	+	5
NAS	Número de asserções por passo	+	5
TEP	Produtividade da equipe	+/-	5
NTS	Número de suítes de teste	+/-	5
PTC	Comprometimento da equipe do projeto	-	4
ARA	Disponibilidade de artefatos de teste reusáveis	-	4
STT	Suporte de ferramentas de teste	-	4
EAD	Experiência com o domínio de aplicação	-	4
TTS	Tamanho da equipe de teste	-	4
EPT	Experiência com a tecnologia de programação	-	4
UIR	Indisponibilidade de recursos de infraestrutura	+	4
REP	Nível de desempenho requerido	+	4
SIC	Complexidade das interfaces do software	+	4
SPA	Proced. específicos para acessar o sistema	+	4
NPU	Número de parâmetros por unidade	+	4
TDQ	Quantidade de dados de teste	+	4
RLS	Nível de segurança requerido	+	4
PTD	Distribuição física da equipe do projeto	+	4
SYC	Complexidade do sistema	+	4
TEC	Complexidade de execução de teste	+	4
ADI	Quantidade e duração de interrupções	+	4
STL	Nível de testabilidade do software	-	3
EOE	Experiência com o ambiente operacional	-	3
ACD	Diversidade de atores	+	3
CTE	Complexidade do ambiente de teste	+	3
RER	Nível de confiabilidade requerido	+	3
UTE	Indisponibilidade do ambiente de teste	+	3
TPT	Tarefas do processo de teste	+	3
SCP	Pressão sobre o cronograma	+	3
TPC	Capacidade do processo de teste	-	2
ROC	Legibilidade do código	-	2
RLI	Nível de interoperabilidade requerido	+	2
SCC	Complexidade do código-fonte	+	2
UII	Instabilidade da interface gráfica com o usuário	+	2
NOR	Número de versões	+	2
RDC	Capacidade de dados requerida	+	2
RLR	Nível de recuperabilidade requerido	+/-	2
PLU	Nível de usabilidade apresentado	-	1
TET	Ciclos de teste	+	1
LOC	Nível de criticidade	+	1
RLP	Nível de portabilidade requerido	+	1
SCS	Tamanho do código-fonte	+	1
PLC	Nível de confidencialidade do projeto	+	1
QTI	Quantidade de incidentes de teste	+	1
RCE	Coexistência requerida	+	1

Tabela 63: Julgamentos da atividade "Executar Testes" na configuração 2.

Mnemônico	Fator de esforço	Impacto	Frequência
TCO	Cooperação da equipe	-	5
ETT	Experiência com ferramentas de teste	-	5
STT	Suporte de ferramentas de teste	-	5
UIR	Indisponibilidade de recursos de infraestrutura	+	5
TEC	Complexidade de execução de teste	+	5
TET	Ciclos de teste	+	5
QTI	Quantidade de incidentes de teste	+	5
UTE	Indisponibilidade do ambiente de teste	+	5
ADI	Quantidade e duração de interrupções	+	5
NTC	Número de casos de teste	+	5
TCA	Capacidade da equipe	-	4
STL	Nível de testabilidade do software	-	4
TCN	Continuidade da equipe	-	4
EXT	Experiência com teste	-	4
EOE	Experiência com o ambiente operacional	-	4
NWS	Número de Web Services	+	4
PTD	Distribuição física da equipe do projeto	+	4
REI	Instabilidade dos requisitos	+	4
TEP	Produtividade da equipe	+/-	4
ARA	Disponibilidade de artefatos de teste reusáveis	-	3
TTS	Tamanho da equipe de teste	-	3
RER	Nível de confiabilidade requerido	+	3
TPT	Tarefas do processo de teste	+	3
NOR	Número de versões	+	3
RTC	Nível de cobertura requerido	+	3
CTE	Complexidade do ambiente de teste	+	3
SCP	Pressão sobre o cronograma	+	3
ACD	Diversidade de atores	+	3
SPA	Proced. específicos para acessar o sistema	+	3
NTS	Número de suítes de teste	+/-	3
PTC	Comprometimento da equipe do projeto	-	2
TPC	Capacidade do processo de teste	-	2
EAD	Experiência com o domínio de aplicação	-	2
RCE	Coexistência requerida	+	2
REP	Nível de desempenho requerido	+	2
DOQ	Qualidade da documentação	-	1
RLR	Nível de recuperabilidade requerido	-	1
EPT	Experiência com a tecnologia de programação	-	1
RLP	Nível de portabilidade requerido	+	1
SYC	Complexidade do sistema	+	1
TSZ	Tamanho dos scripts de teste	+	1
TSS	Tamanho da especificação técnica	+	1
SIC	Complexidade das interfaces do software	+	1
UII	Instabilidade da interface gráfica com o usuário	+	1
RDC	Capacidade de dados requerida	+	1
RLI	Nível de interoperabilidade requerido	+	1
TSC	Complexidade dos scripts de teste	+	1
TDQ	Quantidade de dados de teste	+	1

Tabela 64: Julgamentos da atividade "Comunicar Incidentes de Teste" na configuração 2.

Mnemônico	Fator de esforço	Impacto	Frequência
TCN	Continuidade da equipe	-	5
TCO	Cooperação da equipe	-	5
QTI	Quantidade de incidentes de teste	+	5
TEP	Produtividade da equipe	+/-	5
STL	Nível de testabilidade do software	-	4
EXT	Experiência com teste	-	4
TCA	Capacidade da equipe	-	4
ETT	Experiência com ferramentas de teste	-	4
PTD	Distribuição física da equipe do projeto	+	4
STT	Suporte de ferramentas de teste	-	3
EAD	Experiência com o domínio de aplicação	-	3
RER	Nível de confiabilidade requerido	+	3
UTE	Indisponibilidade do ambiente de teste	+	3
SCP	Pressão sobre o cronograma	+	3
TET	Ciclos de teste	+	3
ADI	Quantidade e duração de interrupções	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	2
SYC	Complexidade do sistema	+	2
REP	Nível de desempenho requerido	+	2
REC	Complexidade dos requisitos	+	2
REI	Instabilidade dos requisitos	+	2
SIC	Complexidade das interfaces do software	+	2
EOE	Experiência com o ambiente operacional	-	1
EPT	Experiência com a tecnologia de programação	-	1
TPC	Capacidade do processo de teste	-	1
ROC	Legibilidade do código	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
PTC	Comprometimento da equipe do projeto	-	1
TTS	Tamanho da equipe de teste	-	1
DOQ	Qualidade da documentação	-	1
RTC	Nível de cobertura requerido	+	1
RSS	Tamanho da especificação de requisitos	+	1
RDC	Capacidade de dados requerida	+	1
NTC	Número de casos de teste	+	1
NAS	Número de asserções por passo	+	1
TEC	Complexidade de execução de teste	+	1
NWS	Número de Web Services	+	1
LOC	Nível de criticidade	+	1
TSC	Complexidade dos scripts de teste	+	1
ACD	Diversidade de atores	+	1
RLI	Nível de interoperabilidade requerido	+	1
NOR	Número de versões	+	1
TSZ	Tamanho dos scripts de teste	+	1
RLP	Nível de portabilidade requerido	+	1
SPA	Proced. específicos para acessar o sistema	+	1
SCC	Complexidade do código-fonte	+	1
TPT	Tarefas do processo de teste	+	1
RLR	Nível de recuperabilidade requerido	+	1
SCS	Tamanho do código-fonte	+	1
RLS	Nível de segurança requerido	+	1
RCE	Coexistência requerida	+	1
CTE	Complexidade do ambiente de teste	+	1

Tabela 65: Julgamentos da atividade "Finalizar Projeto de Teste" na configuração 2.

Mnemônico	Fator de esforço	Impacto	Frequência
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
TCO	Cooperação da equipe	-	1
EXT	Experiência com teste	-	1
TEP	Produtividade da equipe	-	1
ETT	Experiência com ferramentas de teste	-	1
PTC	Comprometimento da equipe do projeto	-	1
STT	Suporte de ferramentas de teste	-	1
TCA	Capacidade da equipe	-	1
TCN	Continuidade da equipe	-	1
TET	Ciclos de teste	+	1
NWS	Número de Web Services	+	1
ADI	Quantidade e duração de interrupções	+	1

5.4.3.4 Configuração 3

A configuração de estratégia de teste "3" é composta pelo teste em nível de integração, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como representa a Tabela 66.

Tabela 66: Dimensões da configuração de estratégia de teste 3.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de integração	Teste funcional	Teste baseado em especificação	Automatizada

A Figura 37 representa a frequência de julgamentos por fator de esforço na configuração 3, agregando todas as atividades. O fator de esforço mais frequente nos julgamentos realizados foi "Quantidade e duração das interrupções" (ADI) com 18 ocorrências, seguido de "Pressão sobre o cronograma" (SCP), com 16 julgamentos. Considerando o fato de que a maioria dos participantes dessa configuração trabalha na mesma empresa e atuou em um mesmo projeto de larga escala, pode-se inferir que aquele contexto específico interferiu sobremaneira nas respostas.

Figura 37: Frequência de julgamentos por fator de esforço na configuração 3.

Com exceção da atividade “Finalizar projeto de teste”, as demais atividades do processo de teste foram citadas por pelo menos um participante. As tabelas 67 a 71 representam a frequência de julgamentos em cada fator de esforço por atividade.

Tabela 67: Julgamentos da atividade “Planejar Teste” na configuração 3.

Mnemônico	Fator de esforço	Impacto	Frequência
DOQ	Qualidade da documentação	-	1
EXT	Experiência com teste	-	1
EAD	Experiência com o domínio de aplicação	-	1
TCA	Capacidade da equipe	-	1
TCN	Continuidade da equipe	-	1
TCO	Cooperação da equipe	-	1
ADI	Quantidade e duração de interrupções	+	1
SCP	Pressão sobre o cronograma	+	1
REP	Nível de desempenho requerido	+	1
TTS	Tamanho da equipe de teste	+	1
SIC	Complexidade das interfaces do software	+	1
RLS	Nível de segurança requerido	+	1
SYC	Complexidade do sistema	+	1
RTC	Nível de cobertura requerido	+	1
REC	Complexidade dos requisitos	+	1
NWS	Número de Web Services	+	1
RSS	Tamanho da especificação de requisitos	+	1
RDC	Capacidade de dados requerida	+	1
ACD	Diversidade de atores	+	1
REI	Instabilidade dos requisitos	+	1
RLI	Nível de interoperabilidade requerido	+	1
TSS	Tamanho da especificação técnica	+	1

Tabela 68: Julgamentos da atividade "Configurar Ambiente de Teste" na configuração 3.

Mnemônico	Fator de esforço	Impacto	Frequência
EXT	Experiência com teste	-	3
TCN	Continuidade da equipe	-	3
TCO	Cooperação da equipe	-	3
ADI	Quantidade e duração de interrupções	+	3
RLI	Nível de interoperabilidade requerido	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
SYC	Complexidade do sistema	+	3
CTE	Complexidade do ambiente de teste	+	3
SCP	Pressão sobre o cronograma	+	3
EPT	Experiência com a tecnologia de programação	-	2
STT	Suporte de ferramentas de teste	-	2
ARA	Disponibilidade de artefatos de teste reusáveis	-	2
TPC	Capacidade do processo de teste	-	2
TCA	Capacidade da equipe	-	2
EOE	Experiência com o ambiente operacional	-	2
ETT	Experiência com ferramentas de teste	-	2
RLS	Nível de segurança requerido	+	2
TSC	Complexidade dos scripts de teste	+	2
SPA	Proced. específicos para acessar o sistema	+	2
RTC	Nível de cobertura requerido	+	2
LOC	Nível de criticidade	+	2
NOR	Número de versões	+	2
REC	Complexidade dos requisitos	+	2
TDQ	Quantidade de dados de teste	+	2
NTC	Número de casos de teste	+	2
RSS	Tamanho da especificação de requisitos	+	2
NWS	Número de Web Services	+	2
REI	Instabilidade dos requisitos	+	2
PLC	Nível de confidencialidade do projeto	+	2
ACD	Diversidade de atores	+	2
NTS	Número de suítes de teste	+	2
SIC	Complexidade das interfaces do software	+	2
TSS	Tamanho da especificação técnica	+/-	2
TEP	Produtividade da equipe	-	1
STL	Nível de testabilidade do software	-	1
TTS	Tamanho da equipe de teste	-	1
PTC	Comprometimento da equipe do projeto	-	1
EAD	Experiência com o domínio de aplicação	-	1
TEC	Complexidade de execução de teste	+	1
TSZ	Tamanho dos scripts de teste	+	1
UTE	Indisponibilidade do ambiente de teste	+	1
TPT	Tarefas do processo de teste	+	1
TET	Ciclos de teste	+	1
PTD	Distribuição física da equipe do projeto	+	1
QTI	Quantidade de incidentes de teste	+	1

Tabela 69: Julgamentos da atividade "Projetar e Implementar Testes" na configuração 3.

Mnemônico	Fator de esforço	Impacto	Frequência
EAD	Experiência com o domínio de aplicação	-	5
TCN	Continuidade da equipe	-	5
EPT	Experiência com a tecnologia de programação	-	5
TCO	Cooperação da equipe	-	5
DOQ	Qualidade da documentação	-	5
EXT	Experiência com teste	-	5
ETT	Experiência com ferramentas de teste	-	5
ADI	Quantidade e duração de interrupções	+	5
REI	Instabilidade dos requisitos	+	5
TSZ	Tamanho dos scripts de teste	+	5
RLI	Nível de interoperabilidade requerido	+	5
NTS	Número de suítes de teste	+	5
SIC	Complexidade das interfaces do software	+	5
TSC	Complexidade dos scripts de teste	+	5
RLS	Nível de segurança requerido	+	5
NTC	Número de casos de teste	+	5
RTC	Nível de cobertura requerido	+	5
NWS	Número de Web Services	+	5
REC	Complexidade dos requisitos	+	5
SYC	Complexidade do sistema	+	5
RSS	Tamanho da especificação de requisitos	+	5
PTD	Distribuição física da equipe do projeto	+/-	5
SCP	Pressão sobre o cronograma	+/-	5
TTS	Tamanho da equipe de teste	+/-	5
TSS	Tamanho da especificação técnica	+/-	5
TCA	Capacidade da equipe	-	4
ARA	Disponibilidade de artefatos de teste reusáveis	-	4
STT	Suporte de ferramentas de teste	-	4
UIR	Indisponibilidade de recursos de infraestrutura	+	4
LOC	Nível de criticidade	+	4
CTE	Complexidade do ambiente de teste	+	4
SPA	Proced. específicos para acessar o sistema	+	4
ACD	Diversidade de atores	+	4
TDQ	Quantidade de dados de teste	+	4
NAS	Número de asserções por passo	+	4
TPC	Capacidade do processo de teste	-	3
PTC	Comprometimento da equipe do projeto	-	3
TEP	Produtividade da equipe	-	3
STL	Nível de testabilidade do software	-	3
TEC	Complexidade de execução de teste	+	3
NPU	Número de parâmetros por unidade	+	3
QTI	Quantidade de incidentes de teste	+	3
UTE	Indisponibilidade do ambiente de teste	+	3
NOR	Número de versões	+	3
TPT	Tarefas do processo de teste	+	3
EOE	Experiência com o ambiente operacional	-	2
RDC	Capacidade de dados requerida	+	2
TET	Ciclos de teste	+	2
UII	Instabilidade da interface gráfica com o usuário	+	2
PLC	Nível de confidencialidade do projeto	+	2
SCC	Complexidade do código-fonte	+	2
ROC	Legibilidade do código	-	1
RCE	Coexistência requerida	+	1
RER	Nível de confiabilidade requerido	+	1
REP	Nível de desempenho requerido	+	1
SCS	Tamanho do código-fonte	+	1

Tabela 70: Julgamentos da atividade "Executar Testes" na configuração 3.

Mnemônico	Fator de esforço	Impacto	Frequência
ETT	Experiência com ferramentas de teste	-	5
STT	Suporte de ferramentas de teste	-	5
UTE	Indisponibilidade do ambiente de teste	+	5
ADI	Quantidade e duração de interrupções	+	5
TEC	Complexidade de execução de teste	+	5
REI	Instabilidade dos requisitos	+	4
QTI	Quantidade de incidentes de teste	+	4
UIR	Indisponibilidade de recursos de infraestrutura	+	4
NTC	Número de casos de teste	+	4
CTE	Complexidade do ambiente de teste	+	4
NWS	Número de Web Services	+	4
PTD	Distribuição física da equipe do projeto	+/-	4
NTS	Número de suítes de teste	+/-	4
SCP	Pressão sobre o cronograma	+/-	4
TTS	Tamanho da equipe de teste	+/-	4
DOQ	Qualidade da documentação	-	3
EXT	Experiência com teste	-	3
TSZ	Tamanho dos scripts de teste	+	3
TPT	Tarefas do processo de teste	+	3
SPA	Proced. específicos para acessar o sistema	+	3
TET	Ciclos de teste	+	3
RTC	Nível de cobertura requerido	+	3
TSS	Tamanho da especificação técnica	+	3
LOC	Nível de criticidade	+	3
REC	Complexidade dos requisitos	+	3
STL	Nível de testabilidade do software	+/-	3
PTC	Comprometimento da equipe do projeto	-	2
TCN	Continuidade da equipe	-	2
TCO	Cooperação da equipe	-	2
EOE	Experiência com o ambiente operacional	-	2
EAD	Experiência com o domínio de aplicação	-	2
SIC	Complexidade das interfaces do software	+	2
RER	Nível de confiabilidade requerido	+	2
RLI	Nível de interoperabilidade requerido	+	2
TDQ	Quantidade de dados de teste	+	2
SYC	Complexidade do sistema	+	2
UII	Instabilidade da interface gráfica com o usuário	+	2
RSS	Tamanho da especificação de requisitos	+	2
TPC	Capacidade do processo de teste	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
TEP	Produtividade da equipe	-	1
EPT	Experiência com a tecnologia de programação	-	1
TCA	Capacidade da equipe	-	1
NOR	Número de versões	+	1
NAS	Número de asserções por passo	+	1
RLS	Nível de segurança requerido	+	1
RCE	Coexistência requerida	+	1

Tabela 71: Julgamentos da atividade "Comunicar Incidentes de Teste" na configuração 3.

Mnemônico	Fator de esforço	Impacto	Frequência
TCO	Cooperação da equipe	-	4
ADI	Quantidade e duração de interrupções	+	4
QTI	Quantidade de incidentes de teste	+	4
PTD	Distribuição física da equipe do projeto	+/-	4
TCN	Continuidade da equipe	-	3
EXT	Experiência com teste	-	3
EAD	Experiência com o domínio de aplicação	-	3
TPC	Capacidade do processo de teste	-	3
TCA	Capacidade da equipe	-	3
REI	Instabilidade dos requisitos	+	3
UTE	Indisponibilidade do ambiente de teste	+	3
NTC	Número de casos de teste	+	3
SCP	Pressão sobre o cronograma	+/-	3
TTS	Tamanho da equipe de teste	+/-	3
DOQ	Qualidade da documentação	-	2
ETT	Experiência com ferramentas de teste	-	2
TEP	Produtividade da equipe	-	2
PTC	Comprometimento da equipe do projeto	-	2
STT	Suporte de ferramentas de teste	-	2
RER	Nível de confiabilidade requerido	+	2
NWS	Número de Web Services	+	2
ACD	Diversidade de atores	+	2
NOR	Número de versões	+	2
NAS	Número de asserções por passo	+	2
TEC	Complexidade de execução de teste	+	2
TSZ	Tamanho dos scripts de teste	+	2
SYC	Complexidade do sistema	+	2
NPU	Número de parâmetros por unidade	+	2
REC	Complexidade dos requisitos	+	2
TSC	Complexidade dos scripts de teste	+	2
LOC	Nível de criticidade	+	2
STL	Nível de testabilidade do software	+/-	2
NTS	Número de suítes de teste	+	1
RSS	Tamanho da especificação de requisitos	+	1
SPA	Proced. específicos para acessar o sistema	+	1
PLC	Nível de confidencialidade do projeto	+	1
TSS	Tamanho da especificação técnica	+	1
TDQ	Quantidade de dados de teste	+	1
RLI	Nível de interoperabilidade requerido	+	1
RLS	Nível de segurança requerido	+	1
TPT	Tarefas do processo de teste	+	1
RTC	Nível de cobertura requerido	+	1
TET	Ciclos de teste	+	1

5.4.3.5 Configuração 4

Como mostra a Tabela 72, a configuração de estratégia de teste "4" é formada pelo teste em nível de sistema, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução manual.

Tabela 72: Dimensões da configuração de estratégia de teste 4.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de sistema	Teste funcional	Teste baseado em especificação	Manual

A Figura 38 representa a frequência de julgamentos por fator de esforço na configuração 4, agregando todas as atividades. Os participantes dessa configuração deram ênfase a fatores relacionados à equipe. Todos os cinco fatores com maiores frequências de julgamentos fazem parte da categoria "equipe de teste". O fator mais frequente foi "Experiência com teste" (EXT), com 27 ocorrências, seguido de "Capacidade da equipe" (TCA) e "Experiência com ferramentas de teste" (ETT), com 25 e 24 ocorrências, respectivamente. Fatores como esses acabam sendo indicados mais frequentemente porque tendem a influenciar todas as atividades do processo.

Figura 38: Frequência de julgamentos por fator de esforço na configuração 4.

Quanto à prevalência do fator ETT especificamente, vale ressaltar que, apesar dessa configuração presumir a execução manual dos testes, as demais atividades do processo podem contar com o apoio ferramental, como por exemplo ferramentas para gestão da documentação de testes ou para gestão de defeitos.

Nesta configuração, todos os cinco participantes responderam a respeito das seis atividades do processo de teste. Sendo assim, as tabelas 73 a 78 representam a frequência de julgamentos por fator de esforço em cada atividade.

Tabela 73: Julgamentos da atividade "Planejar Teste" na configuração 4.

Mnemônico	Fator de esforço	Impacto	Frequência
EAD	Experiência com o domínio de aplicação	-	5
TCA	Capacidade da equipe	-	5
DOQ	Qualidade da documentação	-	5
EXT	Experiência com teste	-	5
SYC	Complexidade do sistema	+	5
TCN	Continuidade da equipe	-	4
TCO	Cooperação da equipe	-	4
ETT	Experiência com ferramentas de teste	-	4
ADI	Quantidade e duração de interrupções	+	4
REC	Complexidade dos requisitos	+	4
RSS	Tamanho da especificação de requisitos	+	4
ACD	Diversidade de atores	+	4
RTC	Nível de cobertura requerido	+	4
EOE	Experiência com o ambiente operacional	-	3
ARA	Disponibilidade de artefatos de teste reusáveis	-	3
TEP	Produtividade da equipe	-	3
REI	Instabilidade dos requisitos	+	3
LOC	Nível de criticidade	+	3
CTE	Complexidade do ambiente de teste	+	3
SCP	Pressão sobre o cronograma	+/-	3
TPC	Capacidade do processo de teste	-	2
PTC	Comprometimento da equipe do projeto	-	2
UIR	Indisponibilidade de recursos de infraestrutura	+	2
NTC	Número de casos de teste	+	2
PTD	Distribuição física da equipe do projeto	+	2
SIC	Complexidade das interfaces do software	+	2
SPA	Proced. específicos para acessar o sistema	+	2
RER	Nível de confiabilidade requerido	-	1
STT	Suporte de ferramentas de teste	-	1
TET	Ciclos de teste	+	1
TDQ	Quantidade de dados de teste	+	1
LFT	Falta de fluência no idioma do projeto	+	1
TTS	Tamanho da equipe de teste	+	1
SCC	Complexidade do código-fonte	+	1
UII	Instabilidade da interface gráfica com o usuário	+	1
SCS	Tamanho do código-fonte	+	1
NWS	Número de Web Services	+	1
PLC	Nível de confidencialidade do projeto	+	1
QTI	Quantidade de incidentes de teste	+	1
RLS	Nível de segurança requerido	+	1
TSS	Tamanho da especificação técnica	+	1
TEC	Complexidade de execução de teste	+	1
RLU	Nível de usabilidade requerido	+	1

Tabela 74: Julgamentos da atividade "Configurar Ambiente de Teste" na configuração 4.

Mnemônico	Fator de esforço	Impacto	Frequência
TCA	Capacidade da equipe	-	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
CTE	Complexidade do ambiente de teste	+	3
SYC	Complexidade do sistema	+	3
TCN	Continuidade da equipe	-	2
DOQ	Qualidade da documentação	-	2
TCO	Cooperação da equipe	-	2
EXT	Experiência com teste	-	2
TEP	Produtividade da equipe	-	2
ETT	Experiência com ferramentas de teste	-	2
EAD	Experiência com o domínio de aplicação	-	2
TPC	Capacidade do processo de teste	-	2
REI	Instabilidade dos requisitos	+	2
SIC	Complexidade das interfaces do software	+	2
REC	Complexidade dos requisitos	+	2
ADI	Quantidade e duração de interrupções	+	2
SCP	Pressão sobre o cronograma	+/-	2
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
EOE	Experiência com o ambiente operacional	-	1
PTC	Comprometimento da equipe do projeto	-	1
STT	Suporte de ferramentas de teste	-	1
ACD	Diversidade de atores	+	1
LOC	Nível de criticidade	+	1
LFT	Falta de fluência no idioma do projeto	+	1
PTD	Distribuição física da equipe do projeto	+	1
UII	Instabilidade da interface gráfica com o usuário	+	1
SPA	Proced. específicos para acessar o sistema	+	1
RTC	Nível de cobertura requerido	+	1
TEC	Complexidade de execução de teste	+	1
UTE	Indisponibilidade do ambiente de teste	+	1
TSS	Tamanho da especificação técnica	+	1
RSS	Tamanho da especificação de requisitos	+	1
TTS	Tamanho da equipe de teste	+	1

Tabela 75: Julgamentos da atividade "Projetar e Implementar Testes" na configuração 4.

Mnemônico	Fator de esforço	Impacto	Frequência
EAD	Experiência com o domínio de aplicação	-	5
STT	Suporte de ferramentas de teste	-	5
ARA	Disponibilidade de artefatos de teste reusáveis	-	5
TCA	Capacidade da equipe	-	5
TCN	Continuidade da equipe	-	5
DOQ	Qualidade da documentação	-	5
TCO	Cooperação da equipe	-	5
EXT	Experiência com teste	-	5
ETT	Experiência com ferramentas de teste	-	5
ADI	Quantidade e duração de interrupções	+	5
RTC	Nível de cobertura requerido	+	5
REC	Complexidade dos requisitos	+	5
SYC	Complexidade do sistema	+	5
RSS	Tamanho da especificação de requisitos	+	5
REI	Instabilidade dos requisitos	+	5
NTS	Número de suítes de teste	+	5
TSC	Complexidade dos scripts de teste	+	5
EOE	Experiência com o ambiente operacional	-	4
NAS	Número de asserções por passo	+	4
NTC	Número de casos de teste	+	4
SIC	Complexidade das interfaces do software	+	4
ACD	Diversidade de atores	+	4
PTC	Comprometimento da equipe do projeto	+/-	4
TPC	Capacidade do processo de teste	-	3
TEP	Produtividade da equipe	-	3
TDQ	Quantidade de dados de teste	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
TEC	Complexidade de execução de teste	+	3
TET	Ciclos de teste	+	3
UII	Instabilidade da interface gráfica com o usuário	+	3
LOC	Nível de criticidade	+	3
CTE	Complexidade do ambiente de teste	+	3
PTD	Distribuição física da equipe do projeto	+	3
SCP	Pressão sobre o cronograma	+/-	3
RLS	Nível de segurança requerido	+	2
RLU	Nível de usabilidade requerido	+	2
TPT	Tarefas do processo de teste	+	2
TSZ	Tamanho dos scripts de teste	+	2
QTI	Quantidade de incidentes de teste	+	2
SPA	Proced. específicos para acessar o sistema	+	2
TSS	Tamanho da especificação técnica	+/-	2
TTS	Tamanho da equipe de teste	+/-	2
RER	Nível de confiabilidade requerido	+/-	2
RLI	Nível de interoperabilidade requerido	+	1
NPU	Número de parâmetros por unidade	+	1
LFT	Falta de fluência no idioma do projeto	+	1
UTE	Indisponibilidade do ambiente de teste	+	1
PLC	Nível de confidencialidade do projeto	+	1
NWS	Número de Web Services	+	1
RLP	Nível de portabilidade requerido	+	1

Tabela 76: Julgamentos da atividade "Executar Testes" na configuração 4.

Mnemônico	Fator de esforço	Impacto	Frequência
TCA	Capacidade da equipe	-	5
TCN	Continuidade da equipe	-	5
TCO	Cooperação da equipe	-	5
EXT	Experiência com teste	-	5
STL	Nível de testabilidade do software	-	5
TEP	Produtividade da equipe	-	5
ETT	Experiência com ferramentas de teste	-	5
EAD	Experiência com o domínio de aplicação	-	5
STT	Suporte de ferramentas de teste	-	5
ADI	Quantidade e duração de interrupções	+	5
TEC	Complexidade de execução de teste	+	5
SYC	Complexidade do sistema	+	5
UIR	Indisponibilidade de recursos de infraestrutura	+	5
CTE	Complexidade do ambiente de teste	+	5
NAS	Número de asserções por passo	+	5
REI	Instabilidade dos requisitos	+	5
TSZ	Tamanho dos scripts de teste	+	5
NTC	Número de casos de teste	+	5
TET	Ciclos de teste	+	5
NOR	Número de versões	+	5
RTC	Nível de cobertura requerido	+	5
UTE	Indisponibilidade do ambiente de teste	+	5
PTC	Comprometimento da equipe do projeto	+/-	5
EOE	Experiência com o ambiente operacional	-	4
TPC	Capacidade do processo de teste	-	4
DOQ	Qualidade da documentação	-	4
TTS	Tamanho da equipe de teste	-	4
RSS	Tamanho da especificação de requisitos	+	4
TSC	Complexidade dos scripts de teste	+	4
PTD	Distribuição física da equipe do projeto	+	4
QTI	Quantidade de incidentes de teste	+	4
SIC	Complexidade das interfaces do software	+	4
UII	Instabilidade da interface gráfica com o usuário	+	4
ACD	Diversidade de atores	+	4
SCP	Pressão sobre o cronograma	+/-	4
PLU	Nível de usabilidade apresentado	+/-	4
REC	Complexidade dos requisitos	+	3
TPT	Tarefas do processo de teste	+	3
LOC	Nível de criticidade	+	3
SPA	Proced. específicos para acessar o sistema	+	3
NTS	Número de suítes de teste	+/-	3
TDQ	Quantidade de dados de teste	+/-	3
RLS	Nível de segurança requerido	+	2
PLC	Nível de confidencialidade do projeto	+	2
RLP	Nível de portabilidade requerido	+	2
TSS	Tamanho da especificação técnica	+/-	2
RER	Nível de confiabilidade requerido	+/-	2
EPT	Experiência com a tecnologia de programação	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
SCC	Complexidade do código-fonte	+	1
LFT	Falta de fluência no idioma do projeto	+	1
RLU	Nível de usabilidade requerido	+	1
NPU	Número de parâmetros por unidade	+	1
NWS	Número de Web Services	+	1
RDC	Capacidade de dados requerida	+	1

Tabela 77: Julgamentos da atividade "Comunicar Incidentes de Teste" na configuração 4.

Mnemônico	Fator de esforço	Impacto	Frequência
TEP	Produtividade da equipe	-	5
EXT	Experiência com teste	-	5
ETT	Experiência com ferramentas de teste	-	5
TCA	Capacidade da equipe	-	5
TCN	Continuidade da equipe	-	5
UIR	Indisponibilidade de recursos de infraestrutura	+	5
QTI	Quantidade de incidentes de teste	+	5
PTC	Comprometimento da equipe do projeto	+/-	5
TET	Ciclos de teste	+/-	5
DOQ	Qualidade da documentação	-	4
STT	Suporte de ferramentas de teste	-	4
TCO	Cooperação da equipe	-	4
UTE	Indisponibilidade do ambiente de teste	+	4
REI	Instabilidade dos requisitos	+	4
PTD	Distribuição física da equipe do projeto	+	4
TPC	Capacidade do processo de teste	-	3
EAD	Experiência com o domínio de aplicação	-	3
CTE	Complexidade do ambiente de teste	+	3
TEC	Complexidade de execução de teste	+	3
SYC	Complexidade do sistema	+	3
NOR	Número de versões	+	3
ADI	Quantidade e duração de interrupções	+	3
UII	Instabilidade da interface gráfica com o usuário	+	3
EOE	Experiência com o ambiente operacional	-	2
SIC	Complexidade das interfaces do software	+	2
PLC	Nível de confidencialidade do projeto	+	2
SPA	Proced. específicos para acessar o sistema	+	2
RTC	Nível de cobertura requerido	+	2
ACD	Diversidade de atores	+	2
TSZ	Tamanho dos scripts de teste	+	2
LOC	Nível de criticidade	+	2
NTC	Número de casos de teste	+	2
SCP	Pressão sobre o cronograma	+/-	2
TSS	Tamanho da especificação técnica	-	1
NTS	Número de suítes de teste	-	1
STL	Nível de testabilidade do software	-	1
TTS	Tamanho da equipe de teste	+	1
LFT	Falta de fluência no idioma do projeto	+	1
NAS	Número de asserções por passo	+	1
REC	Complexidade dos requisitos	+	1
SCC	Complexidade do código-fonte	+	1
TSC	Complexidade dos scripts de teste	+	1
RSS	Tamanho da especificação de requisitos	+	1
RLS	Nível de segurança requerido	+	1
RLU	Nível de usabilidade requerido	+	1

Tabela 78: Julgamentos da atividade "Finalizar Projeto de Teste" na configuração 4.

Mnemônico	Fator de esforço	Impacto	Frequência
EXT	Experiência com teste	-	5
STT	Suporte de ferramentas de teste	-	3
TEP	Produtividade da equipe	-	3
ETT	Experiência com ferramentas de teste	-	3
TET	Ciclos de teste	+	3
PTC	Comprometimento da equipe do projeto	-	2
EAD	Experiência com o domínio de aplicação	-	2
TCA	Capacidade da equipe	-	2
DOQ	Qualidade da documentação	-	2
TCN	Continuidade da equipe	-	2
TCO	Cooperação da equipe	-	2
EOE	Experiência com o ambiente operacional	-	2
REI	Instabilidade dos requisitos	+	2
SCP	Pressão sobre o cronograma	+	2
ADI	Quantidade e duração de interrupções	+	2
UIR	Indisponibilidade de recursos de infraestrutura	+	2
QTI	Quantidade de incidentes de teste	+	2
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
NTS	Número de suítes de teste	-	1
TPT	Tarefas do processo de teste	-	1
STL	Nível de testabilidade do software	-	1
RLS	Nível de segurança requerido	+	1
REP	Nível de desempenho requerido	+	1
LFT	Falta de fluência no idioma do projeto	+	1
UTE	Indisponibilidade do ambiente de teste	+	1
RLU	Nível de usabilidade requerido	+	1
RTC	Nível de cobertura requerido	+	1
ACD	Diversidade de atores	+	1
SIC	Complexidade das interfaces do software	+	1
CTE	Complexidade do ambiente de teste	+	1
NOR	Número de versões	+	1
SYC	Complexidade do sistema	+	1
LOC	Nível de criticidade	+	1
NPU	Número de parâmetros por unidade	+	1
PLC	Nível de confidencialidade do projeto	+	1
NTC	Número de casos de teste	+	1
PTD	Distribuição física da equipe do projeto	+	1

5.4.3.6 Configuração 5

A configuração de estratégia de teste "5" é composta pelo teste em nível de sistema, do tipo funcional, utilizando a técnica de teste baseado em especificação, com a execução automatizada, como mostra a Tabela 79.

Tabela 79: Dimensões da configuração de estratégia de teste 5.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Teste de sistema	Teste funcional	Teste baseado em especificação	Automatizada

A Figura 39 representa a frequência de julgamentos por fator de esforço na configuração 5, agregando todas as atividades. O fator de esforço com mais julgamentos nessa configuração foi "Suporte de ferramentas de teste" (STT), com 19 ocorrências, seguido por "Experiência com teste" (EXT), "Experiência com ferramentas de teste" (ETT), "Quantidade e duração de interrupções" (ADI) e "Cooperação da equipe" (TCO), todos com 18 ocorrências. O fato dos fatores STT e ETT aparecerem entre os mais frequentes pode indicar que os participantes sentem falta de apoio ferramental ou de treinamento no uso de ferramentas, tendo em vista que trata-se de uma configuração extremamente dependente de ferramentas de automação de testes.

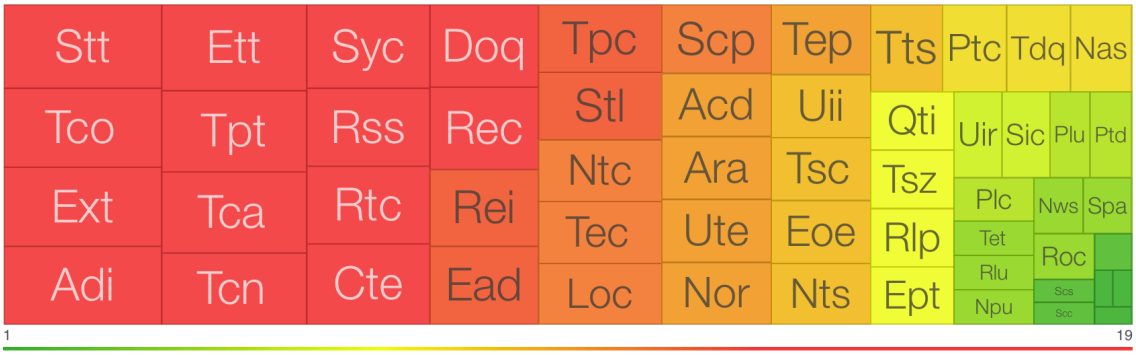


Figura 39: Frequência de julgamentos por fator de esforço na configuração 5.

Assim como na configuração 3, à exceção da atividade "Finalizar projeto de teste", todas as demais atividades do processo de teste foram citadas por pelo menos um dos cinco participantes dessa configuração. As tabelas 80 a 84 representam as frequências de julgamentos por fator de esforço em cada atividade.

Tabela 80: Julgamentos da atividade "Planejar Teste" na configuração 5.

Mnemônico	Fator de esforço	Impacto	Frequência
DOQ	Qualidade da documentação	-	4
EXT	Experiência com teste	-	3
EAD	Experiência com o domínio de aplicação	-	3
TPC	Capacidade do processo de teste	-	3
TCA	Capacidade da equipe	-	3
TCO	Cooperação da equipe	-	3
REI	Instabilidade dos requisitos	+	3
RTC	Nível de cobertura requerido	+	3
ADI	Quantidade e duração de interrupções	+	3
RSS	Tamanho da especificação de requisitos	+	3
TEP	Produtividade da equipe	-	2
STT	Suporte de ferramentas de teste	-	2
TCN	Continuidade da equipe	-	2
SCP	Pressão sobre o cronograma	+	2
LOC	Nível de criticidade	+	2
TPT	Tarefas do processo de teste	+	2
SYC	Complexidade do sistema	+	2
ACD	Diversidade de atores	+	2
REC	Complexidade dos requisitos	+	2
TTS	Tamanho da equipe de teste	+/-	2
EOE	Experiência com o ambiente operacional	-	1
ETT	Experiência com ferramentas de teste	-	1
NTS	Número de suítes de teste	-	1
PTC	Comprometimento da equipe do projeto	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
TDQ	Quantidade de dados de teste	+	1
SIC	Complexidade das interfaces do software	+	1
NAS	Número de asserções por passo	+	1
PLC	Nível de confidencialidade do projeto	+	1
SPA	Proced. específicos para acessar o sistema	+	1
NOR	Número de versões	+	1
CTE	Complexidade do ambiente de teste	+	1

Tabela 81: Julgamentos da atividade "Configurar Ambiente de Teste" na configuração 5.

Mnemônico	Fator de esforço	Impacto	Frequência
TCO	Cooperação da equipe	-	5
CTE	Complexidade do ambiente de teste	+	5
TCA	Capacidade da equipe	-	4
EXT	Experiência com teste	-	4
TCN	Continuidade da equipe	-	4
ETT	Experiência com ferramentas de teste	-	4
STT	Suporte de ferramentas de teste	-	4
ADI	Quantidade e duração de interrupções	+	4
TPT	Tarefas do processo de teste	+	3
TSC	Complexidade dos scripts de teste	+	3
UTE	Indisponibilidade do ambiente de teste	+	3
RLP	Nível de portabilidade requerido	+	3
TDQ	Quantidade de dados de teste	+	3
TEC	Complexidade de execução de teste	+	3
SYC	Complexidade do sistema	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
STL	Nível de testabilidade do software	+/-	3
EPT	Experiência com a tecnologia de programação	-	2
DOQ	Qualidade da documentação	-	2
EOE	Experiência com o ambiente operacional	-	2
TPC	Capacidade do processo de teste	-	2
NWS	Número de Web Services	+	2
REC	Complexidade dos requisitos	+	2
RSS	Tamanho da especificação de requisitos	+	2
NOR	Número de versões	+	2
NTC	Número de casos de teste	+	2
SIC	Complexidade das interfaces do software	+	2
PTC	Comprometimento da equipe do projeto	-	1
TTS	Tamanho da equipe de teste	-	1
TEP	Produtividade da equipe	-	1
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
EAD	Experiência com o domínio de aplicação	-	1
RCE	Coexistência requerida	+	1
PLC	Nível de confidencialidade do projeto	+	1
SPA	Proced. específicos para acessar o sistema	+	1
RTC	Nível de cobertura requerido	+	1
PTD	Distribuição física da equipe do projeto	+	1
RLI	Nível de interoperabilidade requerido	+	1
ACD	Diversidade de atores	+	1
REI	Instabilidade dos requisitos	+	1
NPU	Número de parâmetros por unidade	+	1
NTS	Número de suítes de teste	+	1
SCP	Pressão sobre o cronograma	+	1
LOC	Nível de criticidade	+	1
NAS	Número de asserções por passo	+	1

Tabela 82: Julgamentos da atividade "Projetar e Implementar Testes" na configuração 5.

Mnemônico	Fator de esforço	Impacto	Frequência
DOQ	Qualidade da documentação	-	5
EXT	Experiência com teste	-	5
ETT	Experiência com ferramentas de teste	-	5
EAD	Experiência com o domínio de aplicação	-	5
STT	Suporte de ferramentas de teste	-	5
NTC	Número de casos de teste	+	5
RTC	Nível de cobertura requerido	+	5
REC	Complexidade dos requisitos	+	5
TPT	Tarefas do processo de teste	+	5
TSC	Complexidade dos scripts de teste	+	5
UII	Instabilidade da interface gráfica com o usuário	+	5
ADI	Quantidade e duração de interrupções	+	5
STL	Nível de testabilidade do software	+/-	5
ARA	Disponibilidade de artefatos de teste reusáveis	-	4
TCO	Cooperação da equipe	-	4
TCA	Capacidade da equipe	-	4
TCN	Continuidade da equipe	-	4
EPT	Experiência com a tecnologia de programação	-	4
NTS	Número de suítes de teste	+	4
SYC	Complexidade do sistema	+	4
LOC	Nível de criticidade	+	4
NAS	Número de asserções por passo	+	4
RSS	Tamanho da especificação de requisitos	+	4
REI	Instabilidade dos requisitos	+	4
SCP	Pressão sobre o cronograma	+/-	4
PTC	Comprometimento da equipe do projeto	-	3
TEP	Produtividade da equipe	-	3
TPC	Capacidade do processo de teste	-	3
TEC	Complexidade de execução de teste	+	3
TSZ	Tamanho dos scripts de teste	+	3
ACD	Diversidade de atores	+	3
ROC	Legibilidade do código	+/-	3
PLU	Nível de usabilidade apresentado	+/-	3
EOE	Experiência com o ambiente operacional	-	2
PTD	Distribuição física da equipe do projeto	+	2
SCC	Complexidade do código-fonte	+	2
CTE	Complexidade do ambiente de teste	+	2
RLU	Nível de usabilidade requerido	+	2
UTE	Indisponibilidade do ambiente de teste	+	2
NOR	Número de versões	+	2
NPU	Número de parâmetros por unidade	+	2
TTS	Tamanho da equipe de teste	+/-	2
SIC	Complexidade das interfaces do software	+	1
RLP	Nível de portabilidade requerido	+	1
RER	Nível de confiabilidade requerido	+	1
SCS	Tamanho do código-fonte	+	1
NWS	Número de Web Services	+	1
TDQ	Quantidade de dados de teste	+	1
PLC	Nível de confidencialidade do projeto	+	1
TET	Ciclos de teste	+	1
QTI	Quantidade de incidentes de teste	+	1

Tabela 83: Julgamentos da atividade "Executar Testes" na configuração 5.

Mnemônico	Fator de esforço	Impacto	Frequência
STT	Suporte de ferramentas de teste	-	5
ETT	Experiência com ferramentas de teste	-	5
CTE	Complexidade do ambiente de teste	+	5
UTE	Indisponibilidade do ambiente de teste	+	5
TEC	Complexidade de execução de teste	+	5
EXT	Experiência com teste	-	4
RSS	Tamanho da especificação de requisitos	+	4
SYC	Complexidade do sistema	+	4
NTC	Número de casos de teste	+	4
RTC	Nível de cobertura requerido	+	4
ADI	Quantidade e duração de interrupções	+	4
TPT	Tarefas do processo de teste	+	4
TCA	Capacidade da equipe	-	3
TCN	Continuidade da equipe	-	3
EOE	Experiência com o ambiente operacional	-	3
TCO	Cooperação da equipe	-	3
NOR	Número de versões	+	3
TET	Ciclos de teste	+	3
UIR	Indisponibilidade de recursos de infraestrutura	+	3
UII	Instabilidade da interface gráfica com o usuário	+	3
ACD	Diversidade de atores	+	3
RLP	Nível de portabilidade requerido	+	3
LOC	Nível de criticidade	+	3
TSZ	Tamanho dos scripts de teste	+	3
REC	Complexidade dos requisitos	+	3
ARA	Disponibilidade de artefatos de teste reusáveis	+/-	3
NTS	Número de suítes de teste	+/-	3
STL	Nível de testabilidade do software	+/-	3
SCP	Pressão sobre o cronograma	+/-	3
TEP	Produtividade da equipe	-	2
TPC	Capacidade do processo de teste	-	2
PTC	Comprometimento da equipe do projeto	-	2
TTS	Tamanho da equipe de teste	-	2
QTI	Quantidade de incidentes de teste	+	2
SPA	Proced. específicos para acessar o sistema	+	2
NAS	Número de asserções por passo	+	2
REI	Instabilidade dos requisitos	+	2
TDQ	Quantidade de dados de teste	+/-	2
EPT	Experiência com a tecnologia de programação	-	1
PLU	Nível de usabilidade apresentado	-	1
EAD	Experiência com o domínio de aplicação	-	1
SIC	Complexidade das interfaces do software	+	1
NPU	Número de parâmetros por unidade	+	1
RCE	Coexistência requerida	+	1
TSC	Complexidade dos scripts de teste	+	1
RLU	Nível de usabilidade requerido	+	1
RLS	Nível de segurança requerido	+	1
SCS	Tamanho do código-fonte	+	1
PLC	Nível de confidencialidade do projeto	+	1
NWS	Número de Web Services	+	1

Tabela 84: Julgamentos da atividade "Comunicar Incidentes de Teste" na configuração 5.

Mnemônico	Fator de esforço	Impacto	Frequência
QTI	Quantidade de incidentes de teste	+	4
STT	Suporte de ferramentas de teste	-	3
TCA	Capacidade da equipe	-	3
ETT	Experiência com ferramentas de teste	-	3
TCO	Cooperação da equipe	-	3
TPT	Tarefas do processo de teste	+	3
TPC	Capacidade do processo de teste	-	2
DOQ	Qualidade da documentação	-	2
EXT	Experiência com teste	-	2
TCN	Continuidade da equipe	-	2
EAD	Experiência com o domínio de aplicação	-	2
TEP	Produtividade da equipe	-	2
NOR	Número de versões	+	2
ADI	Quantidade e duração de interrupções	+	2
SYC	Complexidade do sistema	+	2
REI	Instabilidade dos requisitos	+	2
PTD	Distribuição física da equipe do projeto	+	2
TTS	Tamanho da equipe de teste	+/-	2
ARA	Disponibilidade de artefatos de teste reusáveis	-	1
EOE	Experiência com o ambiente operacional	-	1
PTC	Comprometimento da equipe do projeto	-	1
SIC	Complexidade das interfaces do software	+	1
PLC	Nível de confidencialidade do projeto	+	1
TDQ	Quantidade de dados de teste	+	1
STL	Nível de testabilidade do software	+	1
ACD	Diversidade de atores	+	1
RTC	Nível de cobertura requerido	+	1
PLU	Nível de usabilidade apresentado	+	1
CTE	Complexidade do ambiente de teste	+	1
REC	Complexidade dos requisitos	+	1
TSZ	Tamanho dos scripts de teste	+	1
UII	Instabilidade da interface gráfica com o usuário	+	1
RSS	Tamanho da especificação de requisitos	+	1
LOC	Nível de criticidade	+	1
ROC	Legibilidade do código	+	1
SCP	Pressão sobre o cronograma	+	1
RLU	Nível de usabilidade requerido	+	1

5.4.4 Concordância entre participantes

Os julgamentos evidenciados neste estudo podem apoiar o trabalho de gerentes de testes de diferentes formas. Independentemente da aplicação que for realizada, gerentes de testes podem sentir a necessidade de apoiar suas decisões em um limiar de pontuação com um critério de corte. No caso deste estudo, a pertinência dos fatores de esforço em cada combinação de configuração e atividade de teste pode ser evidenciada pelo nível de concordância entre avaliadores (*inter-rater agreement*), que são os participantes do estudo. A concordância entre avaliadores pode ser definida como o grau em que dois ou mais avaliadores, utilizando a mesma escala de avaliação,

fornecem igual classificação para uma mesma situação observável. Trata-se, portanto, de uma medida da consistência entre o valor absoluto das classificações dos avaliadores (MATOS, 2014).

Assim, com o objetivo de perceber o nível de concordância entre as respostas dos participantes, foram aplicadas três diferentes medidas de concordância, também chamadas de medidas de confiabilidade entre avaliadores: concordância média (*average agreement*), Kappa de Fleiss e Alfa de Krippendorff. O cálculo realizado por cada um desses métodos leva em conta apenas o impacto declarado em cada julgamento feito no estudo. Essa análise não contempla a concordância semântica do *rationale* indicado em cada julgamento. A Tabela 85 apresenta os resultados por configuração e atividade.

Tabela 85: Medidas de concordância entre as respostas dos participantes.

Configuração	Medida de concordância	Planejar teste	Configurar ambiente de teste	Projetar e implementar testes	Executar testes	Comunicar incidentes de teste	Finalizar projeto de teste
0	Avg. agreement	NA	-	72,097%	69,516%	NA	NA
	Fleiss' Kappa	NA	-	0,556	0,288	NA	NA
	Krippendorff's α	NA	0,347	0,558	0,29	NA	NA
1	Avg. agreement	-	-	75,806%	68,065%	68,226%	-
	Fleiss' Kappa	-	-	0,623	0,466	0,379	-
	Krippendorff's α	0,556	0,417	0,625	0,468	0,381	-
2	Avg. agreement	-	-	66,935%	69,516%	64,194%	-
	Fleiss' Kappa	-	-	0,49	0,49	0,282	-
	Krippendorff's α	0,472	0,382	0,491	0,491	0,285	-
3	Avg. agreement	-	-	71,613%	63,226%	-	NA
	Fleiss' Kappa	-	-	0,555	0,346	-	NA
	Krippendorff's α	-	0,344	0,556	0,349	0,283	NA
4	Avg. agreement	70,645%	-	70,968%	72,419%	69,516%	71,613%
	Fleiss' Kappa	0,429	-	0,543	0,577	0,457	0,154
	Krippendorff's α	0,431	0.305	0,544	0,579	0,458	0,156
5	Avg. agreement	-	65,806%	67,742%	61,129%	-	NA
	Fleiss' Kappa	-	0,293	0,491	0,323	-	NA
	Krippendorff's α	0,325	0,295	0,493	0,325	0,204	NA

Legenda: "-": A quantidade de dados ausentes (não-respostas) não permitiu a realização do cálculo.
"NA": Não se aplica, visto que nenhum participante respondeu a respeito.

A medida de concordância média não possui um padrão para interpretação dos resultados, ficando essa a cargo de quem tem interesse no resultado. Já as medidas Kappa de Fleiss e Alfa de Krippendorff possuem escalas para auxiliar a interpretação dos resultados, as quais são apresentadas, respectivamente, na Tabela 86 e na Tabela 87.

Tabela 86: Níveis de concordância do Kappa de Fleiss.

Kappa	Nível de concordância
< 0	Concordância menor que o acaso
0.01 - 0.20	Concordância leve
0.21 - 0.40	Concordância justa
0.41 - 0.60	Concordância moderada
0.61 - 0.80	Concordância substancial
0.81 - 0.99	Concordância quase perfeita

Tabela 87: Níveis de concordância do Alfa de Krippendorff.

Alfa	Nível de concordância
> 0.67	Confiabilidade entre avaliadores muito baixa
0.67 - 0.8	Confiabilidade entre avaliadores baixa
> 0.8	Confiabilidade entre avaliadores forte

Observando-se os resultados da Tabela 85 e comparando-os com as escalas apresentadas na Tabela 86 e na Tabela 87, percebe-se que, de acordo com o Kappa de Fleiss somente as respostas referentes à atividade "Projetar e implementar testes" da configuração 1 obtiveram um nível de concordância substancial, o nível mais alto atingindo neste estudo. Utilizando o Alfa de Krippendorff, todos os resultados são classificados como "Confiabilidade entre avaliadores muito baixa". Esses resultados sugerem que há diferentes pontos de vista sobre quais fatores afetam o esforço de teste e como afetam, o que reforça a importância do modelo proposto, assumindo um papel de base de conhecimento.

Vale lembrar que quatro dos cinco participantes entrevistados na configuração 1 atuam na mesma organização, aplicando o mesmo processo e utilizando as mesmas ferramentas, o que poderia explicar o maior nível de concordância apresentado por essa configuração. Essa hipótese é reforçada pelos resultados da configuração 5, composta por participantes de cinco organizações diferentes, que apresentam o segundo menor nível de concordância neste estudo.

Levando-se em conta que os resultados apresentados na Tabela 85 refletem a concordância entre respostas de uma amostra relativamente pequena (cinco participantes em cada configuração), foi aplicada a técnica de reamostragem *bootstrap*, com o objetivo de estabelecer intervalos de confiança para os resultados do Alfa de Krippendorff. A reamostragem foi restrita ao Alfa de Krippendorff por ser o único dentre os três métodos de concordância aplicados que calculou a concordância para todos os cenários com respostas de pelo menos um participante.

A aplicação do método *bootstrap* considerou R = 1000, onde R é o número de reamostragens realizadas a partir dos dados fornecidos. Os intervalos de confiança resultantes são apresentados na Figura 40 para cada uma das configurações tratadas no estudo. Os intervalos de confiança que representam os maiores níveis de concordância de respostas são aqueles com maiores valores e com menor variação. Por exemplo, o intervalo de confiança do Alfa de Krippendorff referente à atividade "Projetar e

implementar testes" da configuração 1, alcança o valor máximo de 0,729, o que o reclassificaria como "Confiabilidade entre avaliadores baixa".

Em geral, a atividade "Projetar e implementar testes" apresentou os maiores valores para o Alfa de Krippendorff em relação às demais atividades. Somente nas configurações 2 e 4, a atividade "Executar testes" apresentou resultados ligeiramente superiores. As demais atividades do processo de teste apresentaram valores de Alfa de Krippendorff menores e com intervalos de confiança mais amplos.

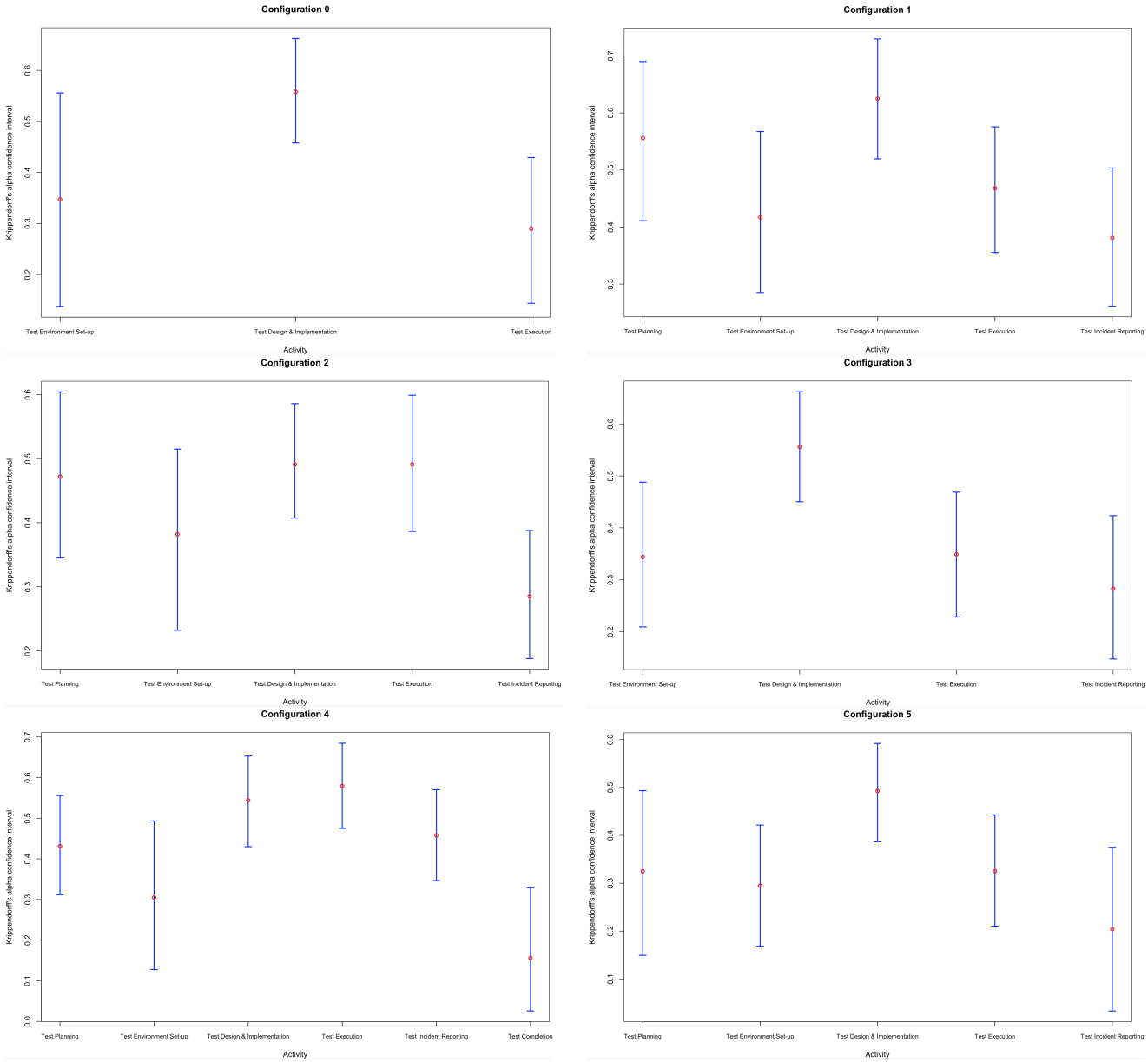


Figura 40: Intervalos de confiança do Alfa de Krippendorff após *bootstrap*.

5.5 Discussão

Inicialmente, este estudo apresentou aspectos relacionados ao perfil dos participantes, com o objetivo de prover informações de contexto sobre a amostra, tendo em vista que trata-se de uma amostra relativamente pequena e definida por conveniência. Apesar disso, os profissionais recrutados demonstram conhecimento prático suficiente para responder a respeito das diversas configurações e atividades de teste tratadas no estudo.

Os profissionais que participaram do estudo foram denominados "participantes". Evitou-se denominá-los de "especialistas" tendo em vista que, em Português, trata-se de um termo que pode presumir perícia (*expert*) ou, até mesmo, trabalho concentrado em uma só atividade (*specialist*). Alguns dos participantes do estudo poderiam ser classificados como especialistas por ambos os motivos. Porém, boa parte deles pratica a atividade de teste de software na mesma intensidade e profundidade que outras atividades típicas de desenvolvimento de software.

Este estudo foi orientado a duas questões de pesquisa específicas e suas subquestões. A questão QE6 diz respeito a quais atividades do processo de teste demandam mais esforço e se há diferenças em função da configuração de estratégia de teste. Já a questão QE7 concentra-se em identificar e compreender os fatores que influenciam o esforço nas atividades do processo de teste de software.

Quanto à questão QE6, o estudo mostrou por meio de um *ranking* que a atividade "Projetar e implementar testes" é a que tipicamente demanda mais esforço entre as atividades do processo de teste em todas as configurações de estratégia de teste analisadas. Essa constatação é justificada pelos participantes do estudo pelo fato dessa atividade ser a mais intensiva em conhecimento e aplicação de raciocínio lógico entre as atividades do processo de teste. Esse resultado contrasta com o fato de que grande parte dos modelos de estimativa de esforço de teste se concentra em estimar o esforço de execução, havendo poucos modelos que incluem a atividade "Projetar e implementar testes" nas suas estimativas. No entanto, dada a variedade de fatores que afetam o esforço dessa atividade, definir um modelo de estimativa de esforço que a contemple pode ser uma tarefa infrutífera.

Em relação aos fatores de esforço, este estudo permitiu confrontar os 51 fatores até então conhecidos (47 identificados na RSL e outros quatro no Estudo de Observação) com profissionais de diferentes organizações que atuam realizando testes

em sistemas de contextos variados. Ao final, além de evidenciar a prevalência dos 51 fatores previamente identificados, foi possível identificar outros onze, totalizando 62 fatores de esforço, divididos em cinco categorias, como mostra a Figura 41. A maior parte desses fatores é inerente ao sistema sob teste, tais como "Complexidade dos requisitos" e "Nível de desempenho requerido". No entanto, observou-se que os fatores mais frequentes nas respostas dos participantes são aqueles relacionados à equipe de teste, tais como "Experiência com teste" e "Experiência com ferramentas de teste". Essa diferença se deve ao fato de que os fatores relacionados ao sistema sob teste tendem a afetar uma ou duas atividades do processo, enquanto que os fatores relacionados à equipe de teste tendem a ser transversais, afetando todas as atividades do processo. Além disso, pode-se inferir que essa maior frequência de fatores ligados a pessoas é, em parte, resultado do modelo de processo utilizado nas organizações dos profissionais entrevistados. Apesar dessa informação não ter sido coletada nos questionários QCP e QPET, durante as entrevistas foi possível perceber que a maioria dos participantes atua ou atuou em projetos iterativos e incrementais, aplicando uma ou mais práticas de agilidade, as quais dependem mais da habilidade das pessoas do que dos processos.

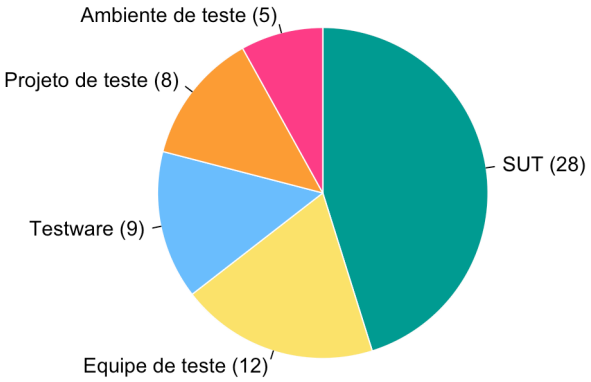


Figura 41: Distribuição dos fatores de esforço consolidados por categoria.

Apenas um fator identificado na RSL e/ou no estudo de observação não foi confirmado no *survey*: o "nível de segurança física pessoal requerida" (LOS). Este fator diz respeito ao conceito de *safety* e foi identificado em um artigo incluído na RSL que aborda testes em sistemas de veículos autônomos não-tripulados (DEONANDAN *et al.*, 2010). Cabe ressaltar que a grande maioria dos participantes do estudo justificou seus julgamentos com base em experiências com testes de sistemas Web ou *desktop*. Assim, apesar de não ter sido reconhecido pelos profissionais entrevistados, este fator provavelmente é pertinente ao contexto de sistemas ciber-físicos.

5.6 Ameaças à Validade

Este estudo envolveu uma série de ameaças à validade, que foram assumidas pelo pesquisador para viabilizar sua realização. Dentre as principais ameaças à validade identificadas, destacam-se as seguintes:

- *O tamanho da amostra é pequeno*. Para cada configuração de estratégia de teste, foram recrutados cinco profissionais para participar do estudo. Esse número é pouco significativo estatisticamente, porém, trata-se de um estudo que exige horas de participação dos indivíduos, o que inviabilizaria estudos com amostras muitos grandes.
- *Amostragem por conveniência*. A seleção da amostra ocorreu de forma acidental, com participantes selecionados a critério do pesquisador, o que impossibilita a generalização dos resultados. Diante disso, tentou-se, na medida do possível, recrutar profissionais de diferentes empresas, de forma a cobrir uma maior variedade de contextos. Por exemplo, sobre a configuração de estratégia de teste 5 foram entrevistados profissionais de cinco empresas diferentes. No entanto, nas configurações 1, 3 e 4 não foi possível definir um arranjo tão diversificado, havendo participantes de apenas duas empresas distintas.
- *Hypothesis guessing*. Possivelmente, os participantes poderiam tentar "adivinhar" quais fatores afetam o esforço de teste ao preencher o questionário QPET, o que representaria uma ameaça à validade de construção. Para minimizá-la, após o preenchimento do QPET, os participantes foram entrevistados para que justificassem cada uma das marcações feitas naquele questionário. Desta forma, marcações feitas por tentativa de adivinhação foram excluídas do QPET pelos próprios participantes quando não conseguiam fornecer as justificativas correspondentes.
- *Os questionários podem inserir fatores de confusão*. Para minimizar essa ameaça, o questionário foi calibrado a partir de um estudo-piloto, que evidenciou melhorias nos instrumentos. Além disso, durante as entrevistas, os participantes tinham a oportunidade de compreender melhor o significado de cada fator de esforço e realizar alterações nas respostas.
- *O rationale dos participantes não foi validado*. Eventualmente, um participante pode indicar um impacto positivo ou negativo de um fator sobre uma atividade utilizando um argumento incoerente ou falacioso. Para minimizar essa ameaça,

ao perceber argumentos incoerentes durante as entrevistas, o pesquisador pedia para o participante explicar com mais detalhes a sua justificativa. No entanto, tomou-se o cuidado para não inibir os participantes de proferirem suas respostas de acordo com suas convicções.

- *Os dados podem ter sido tabulados erroneamente*. As respostas dos questionários foram tabuladas em um banco de dados. Para minimizar a possibilidade de respostas serem tabuladas de forma incorreta, todos os *scripts* de inclusão de respostas no banco de dados foram armazenados e revisados pelo pesquisador.

5.7 Considerações Finais

Este estudo apresentou um *survey* realizado com profissionais envolvidos em atividades de teste de software com o objetivo principal de compreender quais fatores exercem influência sobre o esforço de teste de software e como se dá essa influência. Os resultados evidenciam que o esforço de teste de software é influenciado por diversos fatores, cujos impactos podem ser diferentes de acordo com a configuração de estratégia de teste e a atividade do processo de teste.

O estudo sugere que a atividade "Projetar e Implementar Testes" é a que demanda mais esforço dentre aquelas contempladas no estudo. Essa também é a atividade que apresentou os maiores níveis de concordância entre os participantes do estudo em termos de fatores que afetam o seu esforço, reflexo da maior carga cognitiva que ela exige dos profissionais de teste. Esses resultados podem redirecionar o foco da pesquisa em estimativa de esforço de teste software, historicamente concentrada no esforço de execução de testes.

Embora os resultados apresentados tenham utilidade prática, este estudo apresentou uma série de ameaças à validade, descritas na seção 5.6. No entanto, o estudo tenta compensar essas ameaças com o fato de ter buscado informações com profissionais experientes em teste de software, que se disponibilizaram a participar de um estudo relativamente complexo e que poderia exigir horas de dedicação, dada a quantidade de fatores que deveriam ser avaliados e justificados sob a perspectiva de várias atividades de teste de software.

Trata-se, portanto, de um estudo de difícil execução, que inova ao tentar isolar os fatores de esforço e ao apresentá-los em um nível de detalhe que possibilita uma melhor compreensão sobre o esforço no processo de teste de software. Porém, replicações deste estudo são necessárias para reforçar o conhecimento já evidenciado e para agregar outros aspectos e contextos de projetos de teste de software.

6 HYPERTEST

Este capítulo descreve o modelo de percepção do esforço de teste de software proposto nesta tese. A seção 6.1 apresenta uma introdução ao modelo proposto. A seção 6.2 descreve o modelo proposto. A seção 6.3 mostra uma aplicação Web que instancia o modelo proposto. A seção 6.4 descreve como foi realizada a avaliação do modelo. Por fim, a seção 6.5 apresenta as considerações finais do capítulo.

6.1 Introdução

Com base nos resultados apresentados no *survey*, foi definido um **modelo da percepção do esforço no processo de teste de software**, isto é, um modelo que reflete como um conjunto de profissionais da indústria de software percebe a influência de fatores evidenciados sobre o esforço de teste. Trata-se de um modelo multidimensional, tendo em vista que ele combina as seis configurações de estratégia de teste apresentadas com as seis atividades do processo típico de teste de software, totalizando 36 combinações. Esse modelo recebeu o nome de HyperTest, fazendo referência ao conceito de hipercubo, já que cada configuração de estratégia de teste considera quatro dimensões (nível, tipo, técnica e forma de execução de teste).

O modelo proposto é baseado em um conjunto de evidências coletadas nos diferentes estudos que compõem esta pesquisa. Em um primeiro momento, HyperTest foi concebido com os fatores de esforço que já haviam sido identificados na RSL e/ou no estudo de observação e que foram confirmados no *survey*, além daqueles fatores que foram identificados apenas no *survey*, totalizando 61 fatores. O único fator identificado na RSL e/ou no estudo de observação que não foi confirmado no *survey* (nível de segurança física pessoal requerida - LOS), por hora, não compõe o modelo, ficando armazenado em separado para uma eventual inclusão.

O modelo foi projetado de modo a possibilitar sua evolução, seja pela inclusão ou exclusão de algum fator, seja pela mudança a respeito do impacto que um determinado fator exerce sobre o esforço em uma configuração de estratégia de teste específica. Sendo assim, o modelo HyperTest pode evoluir com a realização de novos *trials* do *survey*, em que outros profissionais de software sejam entrevistados. A Figura

42 representa esse cenário de evolução do modelo à medida em que novas evidências são coletadas.

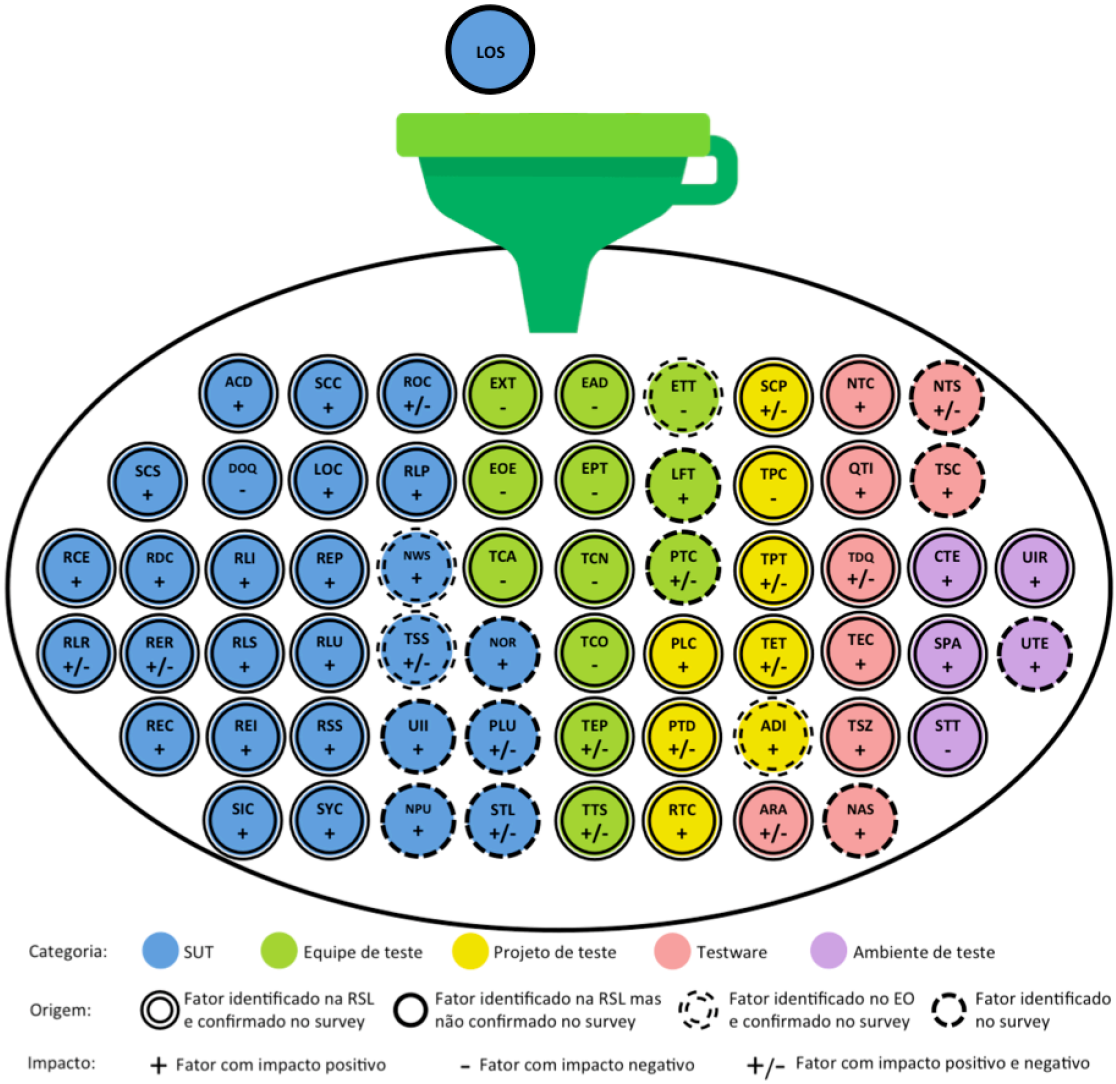


Figura 42: Perspectiva dos fatores que compõem o modelo proposto.

É importante ressaltar que a evolução do HyperTest não se dá somente pela inclusão, alteração ou exclusão de fatores de esforço. O modelo também pode evoluir passando a representar outras configurações de estratégia de testes, além das seis inicialmente contempladas. Desta forma, para proceder qualquer tipo de evolução no modelo proposto é necessário realizar a replicação do *survey*, de modo que as evoluções propostas devam ser fundamentadas nas evidências coletadas com os profissionais de software entrevistados.

6.2 Modelo Conceitual

O modelo HyperTest é uma instância do modelo conceitual da percepção do esforço de teste de software representado na Figura 43. Nesse modelo conceitual as entidades do domínio estão representadas na forma de classes, enquanto que as características dessas entidades estão representadas na forma de atributos. Grande parte dos atributos possui domínio de valores limitado, como por exemplo o atributo "nome" da entidade "Atividade do Processo de Teste". Para casos como esse, o modelo apresenta um conjunto de enumerações, marcadas com o estereótipo <<enum>>, que indicam os possíveis valores que cada atributo pode assumir. No caso do exemplo, os possíveis valores para o atributo "nome" são aqueles definidos na enumeração "Tipo de Atividade".

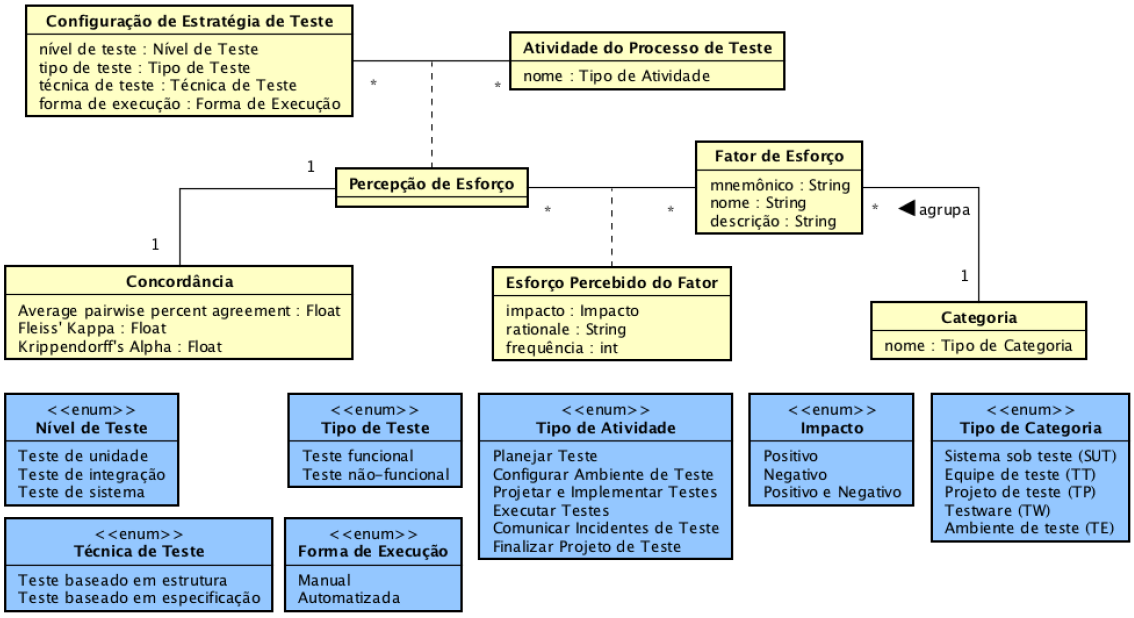


Figura 43: Modelo conceitual do HyperTest.

O modelo conceitual de HyperTest é formado por sete entidades: "Percepção de Esforço", "Configuração de Estratégia de Teste", "Atividade do Processo de Teste", "Fator de Esforço", "Esforço Percebido do Fator", "Categoria" e "Concordância", as quais são descritas em seguida.

Uma *Percepção de Esforço* é definida como uma combinação de uma *Configuração de Estratégia de Teste* e uma *Atividade do Processo de Teste*. Cada percepção representa um possível arranjo entre configurações de estratégia de teste e atividades do processo de teste, consolidando os pontos de vista dos participantes.

Configuração de Estratégia de Teste é a entidade que representa possíveis combinações de nível, tipo, técnica e forma de execução de teste. Inicialmente, HyperTest contempla seis combinações entre os possíveis valores dessas quatro características, já apresentadas no *survey* e novamente representadas na Tabela 88. O modelo conceitual apresentado suporta a inclusão de novas configurações ao modelo HyperTest. Para isso, é necessário realizar a replicação do *survey* e instanciar as entidades do modelo conceitual de acordo com as evidências coletadas que se queiram representar.

Tabela 88: Configurações de estratégia de teste cobertas no estudo.

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução
Unidade	Funcional	Baseada em estrutura	Automatizada
Unidade	Funcional	Baseada em especificação	Automatizada
Unidade	Desempenho	Baseada em especificação	Automatizada
Integração	Funcional	Baseada em especificação	Automatizada
Sistema	Funcional	Baseada em especificação	Manual
Sistema	Funcional	Baseada em especificação	Automatizada

A entidade *Atividade do Processo de Teste* representa basicamente o nome de cada atividade do processo de teste. A enumeração *Tipo de Atividade* restringe essas atividades àquelas seis que compõem um processo típico de teste de software de acordo com a norma ISO/IEC/IEEE 29119-2 (ISO/IEC/IEEE, 2013): "Planejar Teste", "Configurar Ambiente de Teste", "Projetar e Implementar Testes", "Executar Testes", "Comunicar Incidentes de Teste" e "Finalizar Projeto de Teste". O modelo conceitual sugere que toda atividade do processo de teste pode ser exercitada sob qualquer configuração de estratégia de teste. Porém, como o modelo foi concebido inicialmente considerando as seis atividades do processo de teste apresentadas, recomenda-se não adicionar novas atividades àquela enumeração, já que para manter todo o modelo homogêneo e consistente seria necessário replicar o *survey* com os profissionais que participaram do primeiro *trial*, para que tivessem oportunidade de falar a respeito de uma eventual nova atividade do processo de teste.

A entidade *Fator de Esforço* representa as características dos diversos fatores que impactam o esforço das atividades do processo de teste. Essas características se resumem ao mnemônico do fator, seu nome e descrição, além da sua categoria. Trata-se, portanto, de uma entidade que representa apenas as informações de cada fator que são comuns a qualquer perspectiva do modelo.

Em cada perspectiva do modelo os fatores de esforço se apresentam de formas diferentes, seja pela incidência ou não, seja pelo tipo de impacto ou até mesmo pelo *rationale* que os justificam. Como se tratam de características que ocorrem a cada relacionamento entre instâncias das entidades *Percepção de Esforço* e *Fator de Esforço*, elas estão representadas como atributos da entidade *Esforço Percebido do Fator*.

Categoria representa tão somente o nome de cada categoria de fatores de esforço. Essas categorias foram identificadas na RSL e, de acordo, com a enumeração *Tipo de Categoria*, se restringem a: "Sistema sob teste" (ou SUT), "Equipe de teste", "Projeto de teste", "*Testware*" e "Ambiente de teste". A organização dos fatores por categoria pode servir para apoiar a observação da ocorrência de fatores que tenham uma origem comum, facilitando eventuais tratamentos a serem aplicados em projetos. Por exemplo, dois ou mais fatores com impacto positivo relacionados ao ambiente de teste podem ser minimizados utilizando-se uma única decisão de projeto.

Por fim, *Concordância* é a entidade que representa as medidas de concordância entre participantes entrevistados a respeito de cada percepção de esforço. Cada instância de *Concordância* apresenta os valores do percentual médio de concordância, Kappa de Fleiss e Alfa de Krippendorff. Essas medidas foram trazidas do *survey* para esse modelo conceitual. Uma vez instanciado o modelo HyperTest, essas medidas podem apoiar as interpretações dos usuários sobre os modelos. Por exemplo, pode-se assumir que níveis de concordância maiores refletem maior confiança na evidência representada pelo modelo. No entanto, cabe ressaltar que percepções de esforço com altos níveis de concordância podem apenas refletir a diferença de contexto dos participantes do *survey*.

Esse modelo conceitual pode ser exemplificado por meio de instâncias de suas entidades. A Figura 44 representa um recorte das instâncias correspondentes à percepção de esforço composta pela configuração "0" e pela atividade "Projetar e Implementar Testes". Esse diagrama de objetos serve ainda para verificar a consistência dos relacionamentos representados no modelo conceitual. Nele, é possível perceber ocorrências de fatores de esforço com impacto exclusivamente positivo, exclusivamente negativo e simultaneamente positivo e negativo, refletindo as evidências coletadas no *survey*.

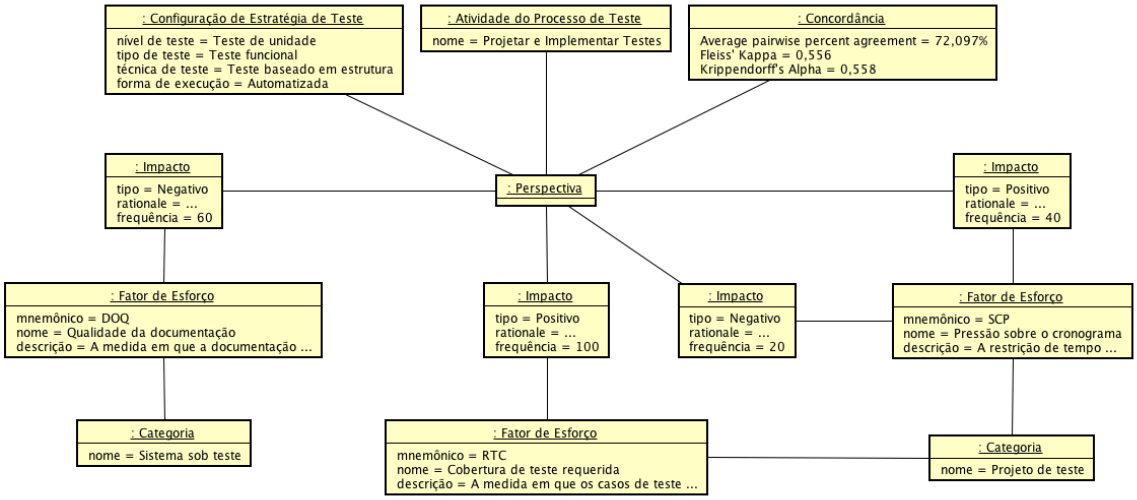


Figura 44: Diagrama de objetos representando uma instância do modelo conceitual.

6.3 Aplicação Web

Dado o caráter dinâmico do modelo de percepção proposto, HyperTest foi implementado como uma aplicação Web, disponível em http://hypertest.tk. Nessa aplicação é possível escolher entre as diversas configurações de estratégia de teste e as atividades do processo de teste cobertas no *survey*. Como descrito na seção 6.2, cada combinação específica de configuração e atividade é denominada *percepção de esforço*. Uma percepção de esforço apresenta, na forma de uma tabela, os fatores de esforço apontados pelos participantes do *survey*, organizados por categoria e pela frequência de julgamentos.

As categorias de fatores são aqueles identificadas na Revisão Sistemática da Literatura: sistema sob teste (*system under testing*, SUT), equipe de teste (*testing team*, TT), projeto de teste (*testing project*, TP), *testware* (TW) e ambiente de teste (*testing environment*, TE). As cinco categorias de fatores são subdivididas em duas, uma para indicar impacto positivo e outra para indicar impacto negativo sobre o esforço. Neste caso, o impacto positivo é representado pelo símbolo "↑" (seta para cima) e o negativo pelo símbolo "↓" (seta para baixo).

Já a frequência de julgamentos é representada em termos percentuais em relação ao total de pontos de vista coletados para cada percepção que compõe o modelo. Assim, o modelo pode evoluir recebendo informações de mais pontos de vista sem alterar a sua estrutura básica. Por exemplo, atualmente cada percepção do modelo conta com cinco pontos de vista de profissionais da indústria. Caso sejam coletadas novos pontos de

vista, as colunas de frequência e das categorias de fatores se manterão inalteradas, bastando, eventualmente, atualizar a posição dos fatores de esforço. A Figura 45 representa uma percepção de esforço que compõe o modelo. Nela, é possível perceber que os fatores mais frequentemente relatados pelos profissionais entrevistados tiveram 60% de frequência, ou seja, foram citados com os mesmos impactos por três dos cinco entrevistados. Todas as 36 percepções do HyperTest estão disponíveis no Apêndice J.

Ao acessar o HyperTest, o usuário deve escolher uma configuração de estratégia de teste (0 a 5) e, em seguida, uma das atividades do processo de teste, passando a visualizar uma percepção de esforço. As percepções formadas por pontos de vista de pelo menos um participante são apresentadas semelhantemente ao exemplo da Figura 45. Já as percepções que não receberam respostas dos participantes apresentam uma mensagem ao usuário, informando que não há evidências de fatores de esforço para aquela combinação de configuração e atividade, como mostra a Figura 46. Essa ausência de evidências é explicada pelo fato de que determinadas atividades de teste não foram, até então, praticadas pelos participantes do *survey* em certas configurações de estratégia de teste.

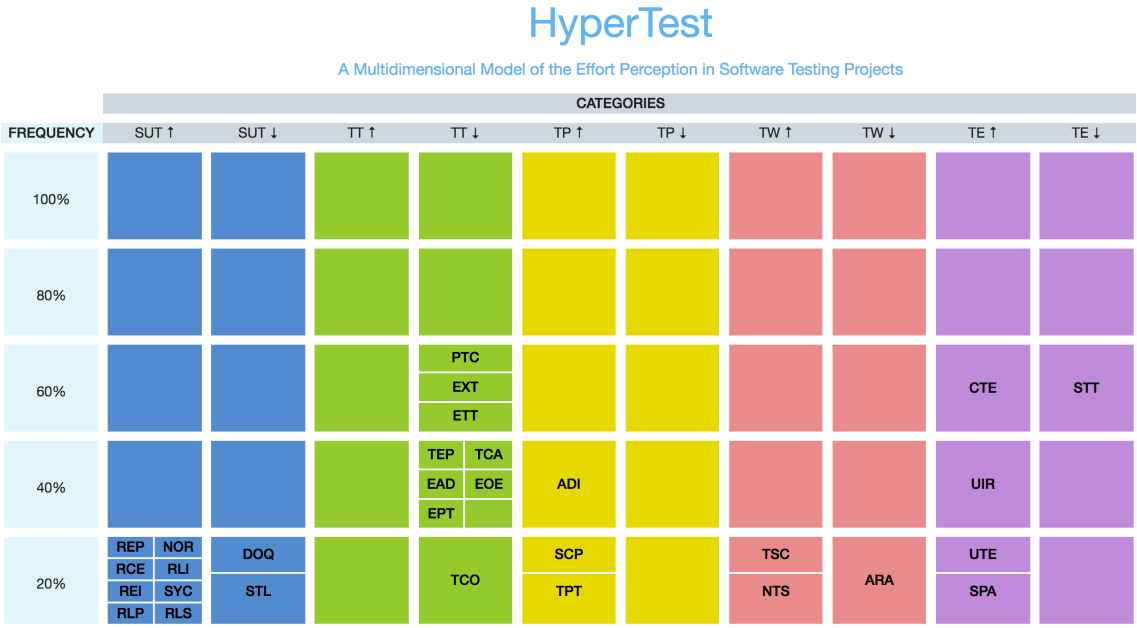


Figura 45: Exemplo de percepção de esforço do modelo HyperTest.

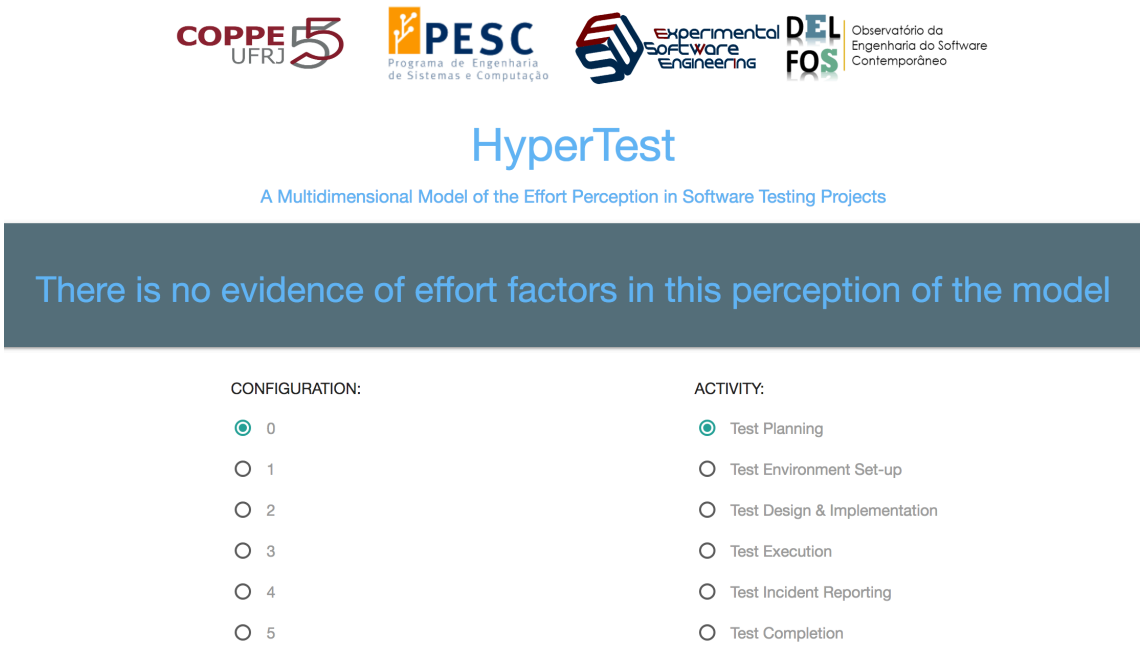


Figura 46: Exemplo de percepção de esforço sem respostas dos participantes do *survey*.

Outra informação presente no modelo são os resultados da análise de concordância. Cada percepção de esforço cujos cálculos de concordância foram realizados apresenta na seção "*Inter-rater reliability*" os respectivos resultados, como representado na Figura 47. O objetivo de apresentar esses valores é dar mais subsídios ao processo de tomada de decisão a respeito do esforço em atividades de teste.

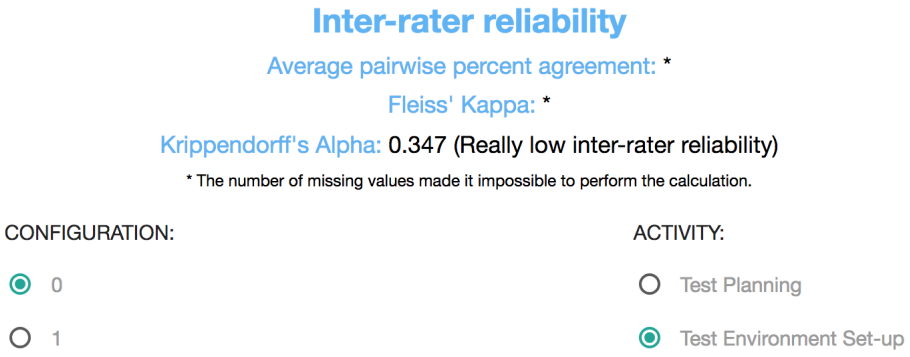


Figura 47: Exemplo de nível de concordância dos participantes do *survey* para uma perspectiva.

Para diversos fatores, os participantes do *survey* indicaram impactos divergentes em seus julgamentos. Há casos em que um participante indicou impacto "Negativo" para um determinado fator e os demais indicaram impacto "Positivo". Há ainda casos em que um participante indicou impacto "Positivo e negativo", quando os demais marcaram apenas "Positivo" ou apenas "Negativo". Ambas as situações são representadas no HyperTest. Os fatores cujos impactos foram relatados sem consenso

pelos participantes são apresentados em vermelho, como representado na Figura 48, na qual os fatores “ROC”, “STL”, “TTS” e “SCP” estão nessa condição. Para efeito do cálculo da frequência, nos casos onde o impacto do fator foi declarado como “Positivo e negativo” conta-se uma ocorrência na coluna correspondente ao impacto “Positivo” e outra ocorrência na coluna de impacto “Negativo”.

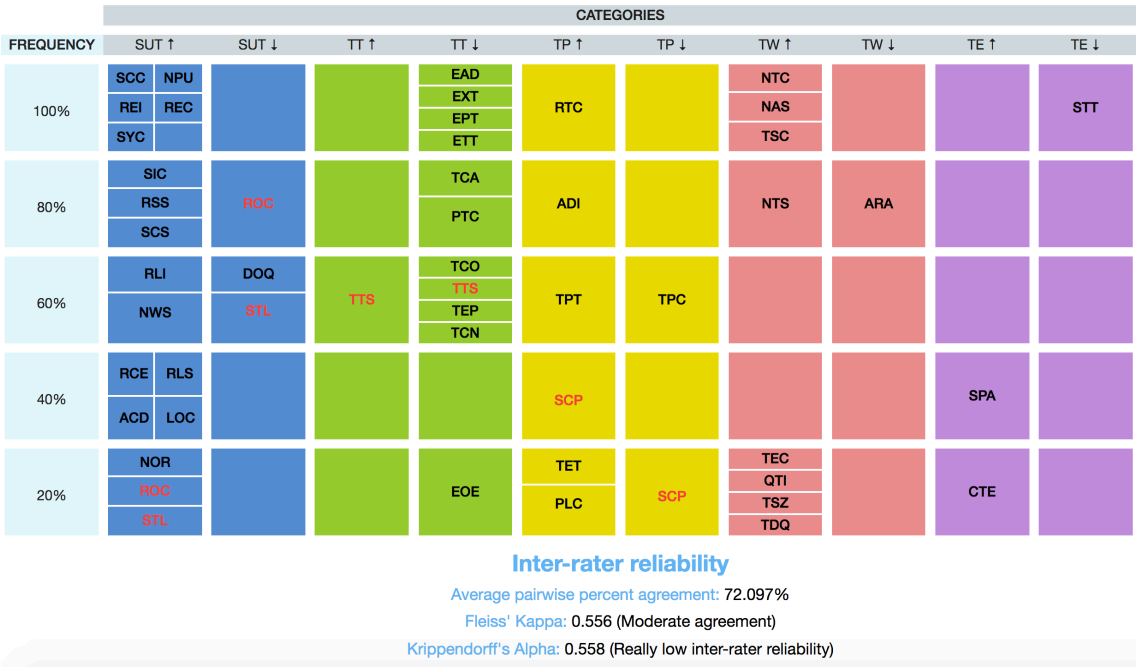


Figura 48: Exemplo de perspectiva composta por fatores com impactos diferentes.

Nas diversas perspectivas do modelo, os fatores de esforço representados possuem uma ficha descritiva contendo, além do nome completo do fator, seu mnemônico, categoria, descrição, impacto e *rationale*. Vale ressaltar que o mnemônico, a categoria e a descrição são informações fixas, que não variam conforme a percepção de esforço. Já o impacto e o *rationale* são específicos por percepção, tendo em vista que um fator pode influenciar o esforço de diferentes maneiras em configurações e atividades de teste diferentes. A Figura 49 apresenta um exemplo de ficha descritiva do fator “Quantidade e duração das interrupções” (ADI).

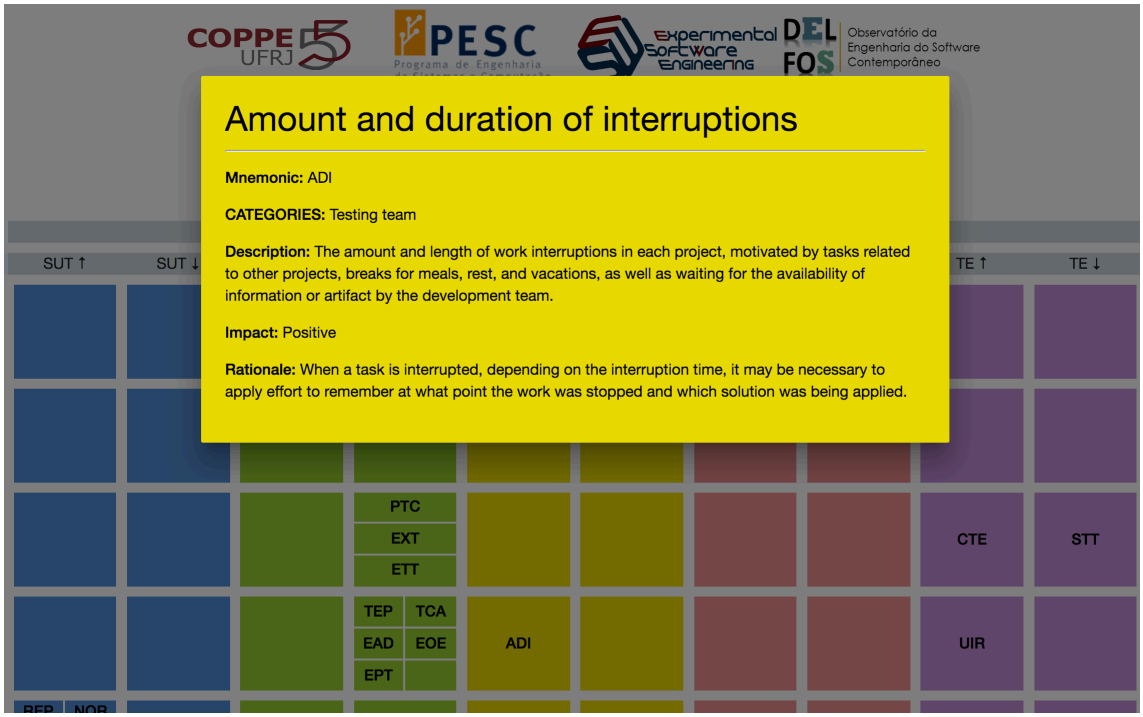


Figura 49: Exemplo de descrição de um fator em uma percepção de esforço.

6.4 Avaliação do Modelo

6.4.1 Introdução

Embora o modelo HyperTest ofereça diversas possibilidades de aplicação, sua avaliação foi conduzida no contexto de estimativa de esforço de teste. Essa avaliação foi realizada por meio de um questionário de *follow-up*, em que se buscou caracterizar quais os efeitos percebidos com o uso do modelo sobre estimativas de esforço de teste baseadas na opinião de profissionais de software. Estimativas baseadas em opinião são aquelas em que o profissional fornece uma estimativa com base na sua experiência e intuição. Esse objetivo está representado na Tabela 89 de acordo com o modelo GQM (BASILI; CALDIERA; ROMBACH, 1994):

Tabela 89: Objetivo da avaliação do modelo HyperTest.

Analisar	modelo HyperTest
Com o propósito de	caracterizar
Em relação à	efeitos do uso do HyperTest sobre estimativas de esforço baseadas em opinião
Do ponto de vista	profissionais de teste de software
No contexto de	cenários de teste de projetos reais

Com base nesse objetivo, foram estabelecidas as seguintes questões de pesquisa:

- **QE8**: O uso do modelo HyperTest afeta a estimativa baseada em opinião de profissionais de software sobre o esforço de teste?
    - **QE8.1**: Quais atividades do processo de teste têm suas estimativas afetadas com o uso do HyperTest?
    - **QE8.2**: Quais fatores de esforço são mais frequentemente negligenciados em estimativas que não se baseiam no HyperTest?
- **QE9**: Como o modelo HyperTest pode contribuir para a estimativa de esforço em projetos de teste de software?

Quanto à medição, a questão QE8 e suas subquestões receberam um tratamento predominantemente quantitativo, baseado nas diferenças relativas entre estimativas realizadas sem e com o apoio do HyperTest, bem como na contagem de atividades e fatores que respondem às questões QE8.1 e QE8.2, respectivamente. Já para a questão QE9, optou-se por uma abordagem qualitativa, baseada nas opiniões fornecidas pelos participantes.

6.4.2 Operação e Instrumentação

Para esta avaliação foram convidados a participar os mesmos profissionais que participaram do *survey*. Porém, como já havia se passado cerca de cinco meses da última rodada de entrevistas, foi definido o seguinte critério de seleção: somente aqueles participantes que estavam envolvidos, no momento desta avaliação, em projetos de teste sob as mesmas configurações de estratégia de testes respondidas no *survey*. Assim, procurou-se evitar a incidência de fatores de confusão decorrentes de experiências mais recentes em outras configurações ou mesmo da falta de prática com teste de software.

Dos 28 profissionais entrevistados no *survey*, somente oito participaram da avaliação do modelo. Os motivos que justificam a não-participação dos demais entrevistados são os seguintes:

- não estavam envolvidos em projetos de teste atualmente: 10
- em licença médica ou viagem: 4
- não estavam disponíveis por conta de compromissos profissionais: 3

- não responderam ao pedido: 2
- não realizou a avaliação no prazo estabelecido: 1

Em relação aos oito participantes recrutados para avaliar o modelo, vale ressaltar que eles cobrem todas as seis configurações de estratégia de testes representadas no HyperTest: a configuração "0" contou com três participantes, enquanto que as demais contaram com um representante cada.

De forma geral, a avaliação do HyperTest se deu pela comparação entre estimativas de esforço de teste feitas primeiramente sem e, em seguida, com o apoio do modelo. Essa avaliação foi guiada por oito questões que compõem o Questionário de Avaliação do Modelo HyperTest (QAMH), representado a seguir. Este questionário foi aplicado separadamente aos participantes desta avaliação em dezembro de 2018.

QAMH	**Questionário de Avaliação do Modelo HyperTest**

1. Descreva um cenário que, em breve, você irá aplicar a mesma configuração de estratégia de teste sobre a qual você respondeu no *survey*.

2. Para cada atividade do processo de teste de software respondida por você no *survey*, estime o esforço que será necessário para testar esse cenário (use a unidade de medida que considerar mais adequada).

3. Acesse o modelo HyperTest em http://hypertest.tk. Escolha a configuração de estratégia de teste correspondente e as atividades do processo. Analise os fatores de esforço representados em cada quadro.

4. Refaça a estimativa de esforço feita no passo 2. Qual foi a estimativa dessa vez?

5. Caso a segunda estimativa tenha sido diferente da primeira, quais motivos justificariam essa diferença?

6. Quais fatores de esforço você considerou na segunda estimativa e não considerou na primeira?

7. De que forma o modelo HyperTest contribuiu ou prejudicou a sua estimativa? Por quê?

8. Você teria alguma crítica ou sugestão para melhoria o modelo HyperTest?

A primeira questão do QAMH pede ao participante para descrever um ou mais cenários de teste que ele estava prestes a aplicar a mesma configuração de estratégia de

teste sobre a qual respondeu no *survey*. Optou-se por solicitar aos participantes que descrevessem cenários de teste reais das suas próprias organizações em vez de submetê-los a cenários reais ou fictícios que porventura fossem muito distantes dos seus contextos.

A questão número 2 solicita aos participantes que estimem, com base nas suas experiências e intuições, o esforço de teste do(s) cenário(s) de teste descrito(s), separadamente por atividade do processo. Neste passo, o participante tinha a liberdade de representar o esforço estimado com a unidade de medida de sua preferência.

Em seguida, na questão 3, os participantes foram apresentados à aplicação Web que representa o HyperTest. Neste passo, o pesquisador explicou a estrutura do modelo e os participantes foram convidados a analisar cada percepção de esforço correspondente às suas respostas no *survey*. Não foi definido um limite de tempo para realização deste passo. Assim, não é possível precisar quanto tempo cada participante analisou o modelo.

Após conhecer o HyperTest, na quarta questão, os participantes realizaram uma nova estimativa de esforço para o mesmo cenário de teste estimado anteriormente. Assim, foi possível comparar as duas baterias de estimativa, o que possibilitou responder as questões QE8 e QE8.1.

A questão 5 tinha como objetivo esclarecer os motivos que justificariam uma eventual diferença entre a primeira e a segunda bateria de estimativas. As respostas serviram para compor parcialmente o entendimento sobre a questão QE9.

A sexta questão está diretamente relacionada com a questão de pesquisa QE8.2, cujo objetivo é identificar os fatores que foram considerados na segunda bateria de estimativas, mas não foram considerados na primeira.

A questão 7 endereça a questão de pesquisa QE9, solicitando que o participante responda se e por que o modelo HyperTest contribuiu ou prejudicou a sua estimativa. Neste caso, o foco está em avaliar se o HyperTest ajudou ou não no ato de estimar, não necessariamente fazendo juízo sobre uma estimativa específica. Para isto, seria necessário avaliar a acurácia da estimativa, o que está fora do escopo desta avaliação, já que demandaria aguardar o término dos projetos de cada participante e confrontar o esforço estimado com o realizado.

Por último, a questão 8 questiona se o participante teria alguma crítica ou sugestão ao modelo HyperTest. Apesar dessa questão não estar diretamente relacionada a nenhuma questão de pesquisa, as respostas podem fornecer importantes direcionamentos para a melhoria do modelo.

6.4.3 Resultados

Foram, portanto, realizadas duas baterias de estimativa com cada participante: a primeira sem e a segunda com o apoio do HyperTest. Como mostra a Tabela 90, em cada bateria foram realizadas 30 estimativas de esforço, uma para cada atividade praticada pelo profissional entrevistado. Na segunda bateria, 12 estimativas (40%) apresentaram variação em relação à primeira bateria, dentre as quais cinco referentes à atividade "Projetar e Implementar Testes", quatro referentes à "Planejar Testes" e três referentes à "Executar Testes".

Tabela 90: Cenários a serem testados e estimativas de esforço correspondentes.

Participante / Configuração	Cenário a ser testado	1ª. bateria de estimativas	2ª. bateria de estimativas
1 / 0	"Um componente responsável por fazer acesso HTTP de forma que as integrações de um macrosistema sejam transparentes quando ao protocolo e tecnologia usados para comunicação".	Conf. Amb. Teste: 30 min Proj. Impl. Testes: 240 min Executar Testes: 10 min	Conf. Amb. Teste: 30 min **Proj. Impl. Testes: 300 min** **Executar Testes: 120 min**
2 / 0	"Consulta parametrizada de um documento no domínio de comércio exterior".	Proj. Impl. Testes: 28 h Executar Testes: 2 min	**Proj. Impl. Testes: 38 h** Executar Testes: 2 min
3 / 0	"Métodos Java para geração de relatórios de contribuintes com benefícios concedidos e com impedimentos".	Proj. Impl. Testes: 8 h Executar Testes: 10 min	Proj. Impl. Testes: 8 h Executar Testes: 10 min
4 / 1	"Serviço que recupera a listagem de documentos distribuídos para análise em uma unidade funcional (ex.: Departamento)".	Planejar Testes: 240 min Proj. Impl. Testes: 480 min Executar Testes: 15 min Com. Inc. de Teste: 5 min	**Planejar Testes: 180 min** **Proj. Impl. Testes: 600 min** Executar Testes: 15 min Com. Inc. de Teste: 5 min
5 / 2	"Serviços referentes à funcionalidade 'Manter cadastro de custodiados' em um sistema Web de gestão de penitenciárias".	Planejar Testes: 8 h Proj. Impl. Testes: 16 h Executar Testes: 8 h Com. Inc. de Teste: 4 h	**Planejar Testes: 10 h** Proj. Impl. Testes: 16 h **Executar Testes: 6 h** Com. Inc. de Teste: 4 h
6 / 3	"Cenário de teste de integração de serviços REST de um portal governamental de comércio exterior. Esse cenário de teste é composto por 10 serviços diferentes".	Conf. Amb. Teste: 30 min Proj. Impl. Testes: 4 h Executar Testes: 5 min Com. Inc. de Teste: 25 min	Conf. Amb. Teste: 30 min **Proj. Impl. Testes: 6 h** **Executar Testes: 10 min** Com. Inc. de Teste: 25 min
7 / 4	"Funcionalidade que permite gerar massa de dados através de uma aplicação que realiza a cópia do banco de produção e replica para o ambiente de teste."	Planejar Testes: 16 h Conf. Amb. Teste: 4 h Proj. Impl. Testes: 16 h Executar Testes: 40 h Com. Inc. de Teste: 10 h Finalizar Proj. de Teste: 4 h	**Planejar Testes: 24 h** Conf. Amb. Teste: 4 h Proj. Impl. Testes: 16 h Executar Testes: 40 h Com. Inc. de Teste: 10 h Finalizar Proj. de Teste: 4 h
8 / 5	"Suíte de testes para um sistema voltado a cidades inteligentes usando o Selenium WebDriver".	Planejar Testes: 20 HH Conf. Amb. Teste: 10 HH Proj. Impl. Testes: 120 HH Executar Testes: 2 h Com. Inc. de Teste: 0,5 HH	**Planejar Testes: 25 HH** Conf. Amb. Teste: 10 HH **Proj. Impl. Testes: 180 HH** Executar Testes: 2 h Com. Inc. de Teste: 0,5 HH

Legenda: em negrito estão as estimativas que apresentaram diferença em relação à primeira bateria.

Os resultados apresentados na Tabela 90 contribuem para responder a questão QE8.2, mas não são suficientes para responder a questão QE8. Para respondê-la, é necessário verificar se as diferenças entre as estimativas são estatisticamente significativas. Neste sentido, foram calculadas as diferenças relativas entre as estimativas e realizado o teste de Wilcoxon.

As estimativas foram realizadas considerando configurações, atividades e cenários de teste diferentes, e foram expressas em unidades de medida distintas: homem-hora, hora e minuto. Não seria adequado, portanto, compará-las em termos de valores absolutos. Desta forma, para calcular a magnitude da influência do modelo HyperTest sobre as estimativas dos participantes foi realizado o cálculo das diferenças relativas, considerando a variação percentual somente daquelas estimativas que apresentaram diferenças entre as duas baterias. A variação percentual pode ser calculada com a equação $\frac{(V2-V1)}{V1}\ x\ 100$, onde *V1* corresponde ao valor da primeira estimativa e *V2* representa o valor da segunda estimativa.

Como mostra a Tabela 91, a média de diferença relativa foi de 128%, resultado influenciado pela diferença de 1.100% na estimativa referente à atividade "Executar Testes" do participante 1. Apesar de ser um *outlier*, esta diferença entre estimativas foi confirmada pelo participante 1, que a justificou dizendo que, após entrar em contato com o HyperTest, se lembrou que a atividade "Executar Testes" contempla também a verificação dos resultados e não apenas o acionamento dos *scripts* de teste automatizados. A mediana das diferenças relativas, por sua vez, foi de 30,36%, o que se mostra um valor representativo para as amostras, já que dez das 12 estimativas apresentaram diferenças entre ± 25 e 50%.

Tabela 91: Diferenças relativas entre as baterias de estimativas.

Participante / Configuração	Diferença relativa
1 / 0	Proj. Impl. Testes: 25% Executar Testes: 1.100%
2 / 0	Proj. Impl. Testes: 35,71%
4 / 1	Planejar Testes: -25% Proj. Impl. Testes: 25%
5 / 2	Planejar Testes: 25% Executar Testes: -25%
6 / 3	Proj. Impl. Testes: 50% Executar Testes: 100%
7 / 4	Planejar Testes: 50%
8 / 5	Planejar Testes: 25% Proj. Impl. Testes: 50%
Média	**128%**
Mediana	**30,36%**

Já o teste de Wilcoxon foi realizado para verificar se há diferenças significativas entre as duas baterias, considerando todas as 30 estimativas realizadas em cada uma. Especificamente, foi aplicado o teste de Wilcoxon pareado bicaudal (*two-tailed*), que é utilizado para comparar se as medidas de posição de duas amostras são iguais no caso em que as amostras são dependentes (TRIOLA, 2008). A escolha desse teste foi motivada não apenas pela possibilidade de avaliar se as diferenças entre estimativas são significativas, mas também por indicar se após o tratamento (modelo HyperTest) os valores das estimativas aumentaram ou diminuíram.

Sendo assim, considerando que o tamanho da amostra $N = 30$, para $\alpha = 5\%$, os valores críticos são $t_1 = 137$ e $t_2 = 327$ (TRIOLA, 2008). O teste apresentou o *p-value* de 0,02012, que é menor que o nível de significância $\alpha = 0,05$. Assim, rejeita-se a hipótese nula e assume-se a hipótese alternativa de que conhecer o modelo HyperTest influencia na estimativa de esforço de testes. Também é possível concluir que conhecer o HyperTest fez aumentar as estimativas de esforço de teste, tendo em vista que $T = 9 <$ que t_1, onde T é a soma dos postos positivos. Portanto, em resposta à questão QE8, há evidências de que uso do modelo HyperTest afeta a estimativa baseada em opinião de profissionais de software sobre o esforço de teste.

Uma vez que se sabe que há diferenças estatisticamente significativas entre as estimativas, faz-se necessário compreender os motivos que influenciaram essas diferenças, o que foi capturado pela questão 5 do QAMH. Assim, a Tabela 92 apresenta as justificativas dadas pelos participantes para a variação nas estimativas da segunda bateria em relação à primeira. De forma geral, as respostas sugerem que o HyperTest fez os participantes se lembrarem de fatores que não foram considerados no momento da primeira bateria de estimativas.

Em relação à questão de pesquisa QE8.1, a Tabela 93 representa os fatores que, de acordo com os participantes, foram ignorados na primeira bateria de estimativas e considerados na segunda bateria, após analisarem o modelo HyperTest. É possível observar que o fator de esforço mais frequentemente negligenciado na primeira bateria de estimativas foi "Qualidade da Documentação" (DOQ), com quatro ocorrências, três delas na atividade "Planejar Testes".

Tabela 92: Justificativas para a variação nas estimativas.

Participante / Configuração	Justificativa
1 / 0	"Alguns fatores como colaboração e cooperação, no cenário analisado, provavelmente demandariam mais esforço para serem implementados. A complexidade do cenário também foi pouco considerada na estimativa."
2 / 0	"Não considerei alguns fatores que estavam explícitos no modelo HyperTest."
3 / 0	Não se aplica.
4 / 1	"No planejamento, acredito que o fator ADI prejudicou, devido às intensas reuniões que têm ocorrido com necessidade de minha presença, porém, o fator ARA contribuiu bastante neste caso por haver um teste semelhante já pronto que irá permitir que partes do teste não precisem ser mais planejados/construídos. Na implementação, o fator NPU geralmente faz o trabalho ser mais complexo pois há um número considerável de parâmetros que preciso ajustar para os serviços e a fonte de valores válidos para estes parâmetros ainda não está bem documentada (fator DOQ - SUT não muito bem documentado / fator ARA ruim no sentido de dados de teste), então é necessário procurar equipes para verificar que valores poderei usar. A execução e a comunicação de incidentes de testes costuma ser muito tranquila e rápida, com raras exceções, por sistemas de apoio (STT) ou o SUT estarem fora do ar, ou seus módulos, prejudicando por exemplo a execução completa de um teste ou a coleta de evidências de um defeito."
5 / 2	"Quanto à atividade Planejar Testes, a estimativa aumentou porque me lembrei de dois fatores que podem aumentar o esforço (REC e DOQ). Quanto à atividade Executar Testes, a estimativa diminuiu porque me lembrei que esse teste será feito em ambiente de produção, que já está preparado, logo fatores como UIR, UTE e CTE não irão afetar o esforço de forma significativa."
6 / 3	"A dificuldade para implementar os testes pode ser maior do que pareceu inicialmente. O problema requer uma solução sofisticada e isso vai aumentar o esforço."
7 / 4	"1 dia a mais de trabalho (8h) em Planejar Testes. Olhando o modelo me lembrei de fatores que não considerei na primeira estimativa."
8 / 5	"Acredito que principalmente pelo fato ter me alertado de questões relacionadas à baixa experiência da equipe, lembranças de que os requisitos são sempre incompletos e mutantes, o sistema está sofrendo alterações constantes e novos desafios técnicos podem ser encontrados."

Tabela 93: Fatores de esforço ignorados na primeira bateria de estimativas.

Participante / Configuração	Atividades do Processo					
	Planejar Testes	Configurar Ambiente de Teste	Projetar e Implementar Testes	Executar Testes	Comunicar Incidentes de Teste	Finalizar Projeto de Testes
1 / 0	***	TCO	CTE	TCO	***	***
2 / 0	***	***	DOQ, REC, NPU	---	***	***
3 / 0	***	***	---	---	***	***
4 / 1	ARA, REI	***	REI	---	---	***
5 / 2	DOQ, REC	***	---	CTE, UIR, UTE	---	***
6 / 3	***	---	SIC, SYC, TSC	QTI	---	***
7 / 4	DOQ, SIC, SYC	---	---	---	---	---
8 / 5	DOQ, EAD	---	ADI, DOQ, ETT, EXT, REQ, TSC, UII	---	TCA, TCO	***

Legenda:
*** Atividade não pertinente ao participante.
--- Atividade sem fatores de esforço ignorados na primeira bateria de estimativas.

Quanto à questão de pesquisa QE9, a Tabela 94 apresenta as contribuições do modelo HyperTest relatadas pelos participantes. Esta questão de pesquisa foi representada pela questão 7 do QAMH, a qual pergunta ao participante se o modelo contribuiu ou prejudicou suas estimativas. Pode-se perceber, no entanto, que todas as respostas indicam que o modelo contribuiu positivamente com as estimativas. Merece destaque a resposta do participante 3, sugerindo que o modelo é útil principalmente para profissionais de teste pouco experientes e para gerentes de testes que precisam tomar decisões fundamentadas em evidências.

Tabela 94: Contribuições do modelo HyperTest às estimativas realizadas.

Participante / Configuração	Contribuição
1 / 0	Contribuiu, porque ajudou a lembrar de alguns fatores que não estavam presentes no momento da estimativa.
2 / 0	Ter uma lista com os fatores que podem influenciar no projeto/execução de testes me fez considerar aspectos que inicialmente eu havia negligenciado.
3 / 0	No meu caso não afetou a estimativa, apesar de considerar o modelo útil principalmente para: a) Profissionais de teste pouco experientes estimarem o esforço de teste, já que não conhecem todos os fatores que podem afetar o esforço de teste; b) Gerentes de testes podem usá-lo para fundamentar decisões sobre investir ou não em determinadas práticas de testes, tendo em vista que o modelo se apoia em evidências.
4 / 1	Especialmente me lembrando de fatores importantes/significativos que num planejamento sem ferramenta/tabelas para auxiliar o planejamento acabo esquecendo ou não considerando a grandeza/relevância dos fatores de esforços no dia a dia, em geral.
5 / 2	Ele me ajudou a lembrar de fatores que realmente afetam o esforço de teste, mas que eu não havia considerado na primeira estimativa por simples esquecimento. Além disso, ele nos faz refletir sobre a incidência de cada fator nos nossos projetos.
6 / 3	Contribuiu porque pude confirmar as minhas suposições sobre como o esforço de teste é afetado, já que o HyperTest compila a opinião de outros profissionais de teste. Além disso, ele me fez refletir sobre como o esforço de teste pode ser minimizado (eu posso tentar priorizar aqueles fatores que diminuem o esforço e evitar os que aumentam).
7 / 4	Contribuiu ajudando a relembrar os fatores que estão relacionados com a atividade do processo de teste.
8 / 5	"Os fatores ajudam a reconsiderar a estimativa, pois quando a estimativa é feita sem o apoio de um 'gabarito', é possível que se esqueça de alguns fatores que influenciam o esforço. Entretanto, ter todos os fatores apresentados de uma só vez acaba fazendo com que a pessoa que irá estimar o esforço considere apenas alguns deles (teoricamente os mais importantes). Ou seja, mesmo sendo apresentados/listados, alguns fatores podem ser ignorados pela própria falta de tempo do responsável pela estimativa."

Finalmente, a oitava questão do QAMH foi direcionada a identificar oportunidades de melhoria no HyperTest. Como mostra a Tabela 95, somente um participante não apresentou críticas ou sugestões ao modelo proposto. Os demais apresentaram críticas e sugestões bastante diversificadas, desde limitações técnicas, como o suporte a determinados navegadores, até sugestões sobre a estrutura e composição do modelo (e.g., "apresentar menos fatores"). Vale destacar as críticas à falta de informações descrevendo o modelo e sobre como utilizá-lo. Essas críticas merecem atenção especial pois podem influenciar negativamente em estimativas de

esforço de teste ou em outras tomadas de decisão que sejam baseadas nos fatores que compõem o modelo.

Tabela 95: Oportunidades de melhoria no modelo HyperTest.

Participante / Configuração	Crítica ou sugestão
1 / 0	"No navegador Firefox 52.9 todas as opções foram apresentadas na tela e não somente a minha configuração."
2 / 0	"A falta de informações descrevendo o modelo (siglas, categorias, etc.) e sobre como usá-lo."
3 / 0	"1. O modelo poderia ter uma lista resumida dos fatores, independentemente das configurações e atividades. 2. Talvez fosse mais fácil de compreender se o modelo fosse representado como o radar de tecnologias da Thoughtworks (https://www.thoughtworks.com/pt/radar). Em vez de 4 áreas, seriam 5 (as categorias do modelo). Dentro de cada área, cada fator seria classificado em 'adote', 'evite', 'avalie' ou termos similares. Por exemplo: a) o fator ADI (*Amount and duration of interruptions*) seria classificado como 'evite'; b) o fator STT (*Support of testing tools*) seria classificado como 'adote'."
4 / 1	"Não."
5 / 2	"Ele poderia fornecer mais informações de contexto sobre cada fator, como em que tipo de projeto foi observado, qual a tecnologia usada, etc."
6 / 3	"Melhorar a usabilidade."
7 / 4	"Não funciona no IE. Executei no Chrome. Falta de um Glossário das siglas no modelo. Não sei como poderia utilizar os resultados da seção *Inter-rater reliability*"
8 / 5	"1. Sugestão seria apresentar os principais fatores (apresentar menos fatores, apenas os mais relevantes) e de acordo com a necessidade de aumentar a confiança na estimativa, apresentaria novos fatores. 2. Ou alguma outra forma de filtrar os fatores. Por exemplo, fornecendo características do software. Assim, o modelo apresentaria apenas fatores relacionadas àquele software. 3. Outra alternativa é fazer em esquema de perguntas que vão aparecendo para o responsável pela estimativa: 'Sua equipe tem experiência baixa, média, alta?', 'Como você classifica o nível de volatilidade de sua equipe?', etc".

6.4.4 Discussão

Apesar da amostra ser relativamente pequena (oito participantes), os resultados apresentados nesta avaliação sugerem que o uso do modelo HyperTest afeta a estimativa baseada em opinião de profissionais de software sobre o esforço de teste, trazendo mais fatores de esforço à tona e, consequentemente, aumentando as estimativas.

Vale ressaltar que esta avaliação não teve como objetivo avaliar a acurácia das estimativas apoiadas pelo HyperTest. Seu objetivo foi somente avaliar o efeitos do uso do modelo sobre estimativas baseadas em opinião. Para avaliar a acurácia das estimativas seria necessário aguardar a conclusão dos projetos considerados nas estimativas para confrontar o esforço estimado e o realizado.

Em relação à questão de pesquisa QE8.1, percebeu-se que a atividade "Projetar e Implementar Testes" foi a mais frequentemente reestimada, o que pode ser explicado pelo fato de ser a atividade com maior variedade de fatores de esforço em todas as

configurações de estratégia de testes cobertas no *survey*. Assim, o processo de estimativa dessa atividade se torna mais complexo, sendo concebível que certos fatores de esforço sejam esquecidos em uma primeira estimativa.

Quanto à questão de pesquisa QE8.2, observou-se que o fator de esforço ignorado com mais frequência nas estimativas feitas sem o apoio do HyperTest foi "Qualidade da Documentação" (DOQ). Esse resultado pode refletir a predisposição dos profissionais de testes em assumir inicialmente que a documentação do SUT estará compreensível e em um nível de detalhamento adequado para ser utilizada nas atividades de teste. Porém, quando se realiza a segunda estimativa tendo em mente os fatores representados no HyperTest, provavelmente os profissionais entrevistados passaram a considerar que a documentação poderia possuir problemas que iriam afetar o esforço.

Já as respostas à questão de pesquisa QE9 sugerem que o modelo HyperTest é especialmente útil como um *checklist* que serve para apoiar não apenas a estimativa de esforço baseada em opinião, mas também processos de tomada de decisão que dependam do conhecimento sobre os fatores que afetam o esforço de teste.

Esta avaliação explorou o modelo proposto em uma perspectiva de estimativa de esforço de teste. Porém, é possível visualizar e descrever situações em que o modelo seja utilizado para apoiar tomadas de decisão em projetos de software. Um exemplo típico de tomada de decisão neste contexto diz respeito à alocação da equipe do projeto.

Considerando um exemplo hipotético em que um gerente de testes tenha que decidir qual analista de testes deve selecionar para atuar em um projeto visando minimizar o esforço de testes, ele poderia fundamentar sua decisão em fatores de esforço representados no modelo HyperTest. Dentre os diversos fatores que compõem o modelo, um subconjunto deles diz respeito à experiência dos profissionais que compõem a equipe de teste: "Experiência com teste" (EXT), "Experiência com ferramentas de teste" (ETT), "Experiência com o domínio de aplicação" (EAD), "Experiência com o ambiente operacional" (EOE) e "Experiência com a tecnologia de programação" (EPT). Esses fatores figuram entre os mais frequentes no *survey* e estão representados na maioria das percepções de esforço que compõem o modelo, sempre com a indicação de que exercem influência negativa sobre o esforço de teste. O gerente de testes poderia, portanto, utilizar esses fatores de esforço como critérios para apoiar a sua tomada de decisão.

Como se trata de uma tomada de decisão que deve levar em conta diversos fatores, cujos pesos podem ser diferentes em função das características do projeto e da configuração de estratégia de testes aplicada, o gerente de testes poderia utilizar métodos de tomada de decisão multicritério, como o *Analytic Hierarchy Process* (AHP). De acordo com esse método, o gerente de testes deveria estabelecer o objetivo da tomada de decisão ("selecionar um analista de testes"), os critérios de seleção (neste caso, os fatores de esforço) com seus respectivos pesos e as alternativas a serem avaliadas. Os pesos dos critérios de seleção são calculados de acordo com a prioridade entre os critérios na perspectiva do responsável pela decisão (gerente de testes), sendo necessárias comparações par a par. As comparações entre cada par de critérios devem ser representadas em uma em uma matriz de comparação usando uma escala de intensidade relativa na forma de frações de 1/9 a 9 (MU; PEREYRA-ROJAS, 2016), como exemplificado na Tabela 96. Neste exemplo, assume-se que o gerente de testes definiu que o fator "Experiência com teste" (EXT) tem maior prioridade (peso) que os demais fatores, resultando em uma prioridade de aproximadamente 50% (0,499). Os 50% restantes ficaram diluídos entre os outros quatro fatores como mostra a Tabela 96.

Tabela 96: Exemplo de matriz de comparação par a par.

Critérios	EXT	ETT	EAD	EOE	EPT	Prioridade
EXT	1	4	3	7	7	0,499
ETT	1/4	1	1/3	3	3	0,129
EAD	1/3	3	1	5	5	0,264
EOE	1/7	1/3	1/5	1	1	0,054
EPT	1/7	1/3	1/5	1	1	0,054

Considere que o gerente de testes avaliou o currículo de três analistas de testes (João, Maria e José), cada qual com diferentes níveis de experiência nos cinco critérios avaliados. João se mostrou o mais experiente em EAD, Maria a mais experiente em EXT e EOE, e José o mais experiente em ETT e EPT. Dependendo da intensidade relativa informada pelo gerente de testes em cada comparação par a par entre os candidatos a analista de testes, o resultado poderia indicar Maria como a candidata mais adequada ao cargo, como representa a Figura 50, com prioridade de 0,473, reflexo do maior peso dado ao critério EXT. Assim, o gerente de testes poderia ter a sua decisão fundamentada nas evidências sobre os fatores de esforço representados no modelo HyperTest.

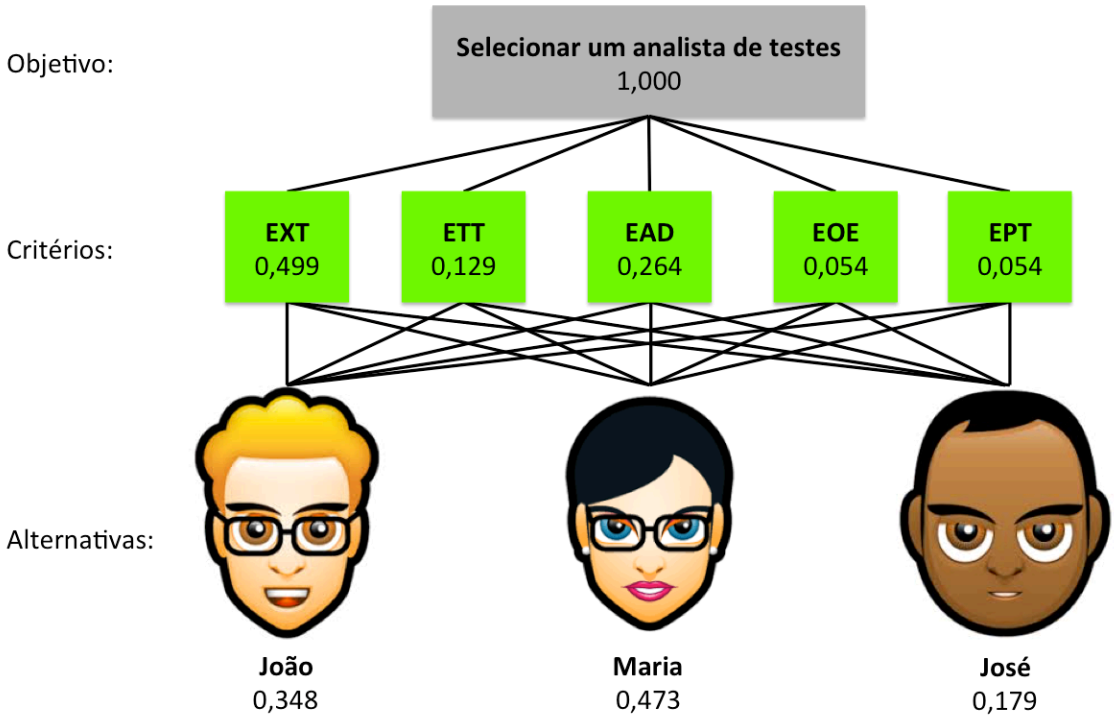


Figura 50: Exemplo de tomada de decisão com base em fatores de esforço.

Outro tipo de processo decisório que pode ser apoiado pelo HyperTest é a identificação de gargalos em processos de teste de software. Uma vez que se tem conhecimento dos fatores que aumentam ou diminuem o esforço, pode-se modificar o processo evitando a incidência de fatores que afetam o esforço positivamente e potencializando aqueles fatores que exercem impacto negativo sobre o esforço. Por exemplo, considerando o fator "Quantidade e duração das interrupções" (ADI), representado na Figura 49, uma organização poderia implementar iniciativas para evitar a interrupção do trabalho, considerando a evidência trazida pelo HyperTest que sugere que quando uma tarefa é interrompida, dependendo do tempo de interrupção, pode ser necessário aplicar mais esforço para lembrar em que ponto o trabalho foi interrompido e qual solução estava sendo aplicada. De forma semelhante, o fator "Suporte de ferramentas de teste" (STT) poderia ser potencializado, de modo que o esforço de teste dos projetos seja diminuído à medida em que ferramentas passam a apoiar o trabalho manual e repetitivo dos testadores.

6.5 Considerações Finais

Este capítulo apresentou o HyperTest, um modelo que consolida a percepção de profissionais de software a respeito do fenômeno do esforço no processo de teste de software. O modelo é composto atualmente por 61 fatores que permeiam 36 combinações de configuração de estratégia de teste com atividade de processo de teste.

O modelo foi avaliado numa perspectiva de estimativa de esforço. Os resultados dessa avaliação evidenciaram que o modelo tem especial utilidade como um *checklist* dos fatores que afetam o esforço de teste, servindo de *baseline* para estimativas baseadas na opinião dos profissionais de teste. Desta forma, o modelo possibilita uma melhor compreensão tanto do processo de teste de software, quanto do fenômeno do esforço que incide sobre esse processo.

O modelo também possui algumas limitações e fragilidades. Provavelmente, sua principal limitação é o fato de não contemplar a magnitude da influência de cada fator. Isso se deve à dificuldade de se observar e medir grande parte dos fatores que compõem o modelo. Essa característica, no entanto, não faz parte do escopo coberto pelo modelo. Quanto às fragilidades, as quais foram evidenciadas na avaliação do modelo, destaca-se a ausência de informações sobre o modelo e como utilizá-lo. Por exemplo, a forma pela qual os percentuais de frequência são apresentados pode levar usuários do modelo a interpretações equivocadas. Tratar essa questão é fundamental para viabilizar o uso adequado do modelo na indústria.

A avaliação do modelo ainda possibilitou a identificação de algumas oportunidades de aplicação, tais como o apoio ao processo de estimativa de esforço de teste e à tomada de decisão multicritério com base no esforço de teste. Seja qual for a aplicação dada ao HyperTest, é importante considerar que ele reflete a experiência de um conjunto de profissionais em determinados contextos de projetos de software, e que outros contextos poderão sofrer a influência de fatores ainda não catalogados no modelo.

7 CONCLUSÃO

Este capítulo apresenta a conclusão desta tese. A seção 7.1 tece as considerações finais do trabalho. A seção 7.2 apresenta as contribuições produzidas com esta tese. A seção 7.3 descreve as limitações deste trabalho. Por fim, a seção 7.4 aponta direcionamentos de trabalhos futuros.

7.1 Considerações Finais

Esta Tese de Doutorado abordou um dos problemas mais críticos em projetos de software: o esforço. Especificamente, tratou do esforço inerente ao processo de teste de software, um dos mais importantes e mais negligenciados pelas organizações que desenvolvem software. Trata-se de um processo relativamente custoso, porém essencial para aferir a qualidade do produto de software desenvolvido e, consequentemente, para determinar o nível de confiança da equipe de desenvolvimento no produto entregue ao cliente.

Este trabalho procurou contribuir para a melhor compreensão do fenômeno do esforço nas atividades de teste de software. Considerando que a indústria de software, de forma geral, trata o esforço como sinônimo de tempo ou de custo, buscou-se identificar na literatura técnica uma definição consistente de esforço em software. Porém, observou-se que há uma grande confusão terminológica na área, o que pode representar uma séria ameaça à validade de inúmeros trabalhos. Diante disso, estabeleceu-se que a definição de esforço que nortearia este trabalho é aquela encontrada no dicionário de Cambridge: "atividade física ou mental necessária para alcançar algum objetivo" (CAMBRIDGE UNIVERSITY PRESS, 2015). Esta definição, apesar de genérica, é compatível com o esforço em software, caracterizado predominantemente por atividades cognitivas.

O teste de software, por sua vez, é um processo complexo, composto por diversas dimensões que quando combinadas implicam em estratégias de teste diferentes. Desde o início deste trabalho, conjecturou-se que para diferentes configurações de estratégia de teste, cada atividade do processo de teste teria seu esforço influenciado por diferentes fatores e de diferentes maneiras. Essa conjectura foi confirmada nos estudos realizados, em especial no *survey* com profissionais.

A metodologia de pesquisa empregada neste trabalho possibilitou que os resultados de um estudo servissem de insumo para outro. Inicialmente, a Revisão Sistemática da Literatura identificou um conjunto de fatores evidenciados na literatura técnica que teriam influência sobre o esforço de teste. Considerando que a RSL não respondeu completamente às questões de pesquisa propostas, foram realizados uma Análise Linguística, focada na definição do termo "esforço" e na sua relação com o termo "custo", e um Estudo de Observação conduzido com o objetivo de obter melhor compreensão sobre os fatores de esforço identificados na RSL. Assim, quando da realização do *survey*, já havia uma *baseline* de fatores de esforço identificados a literatura técnica e/ou no Estudo de Observação. Finalmente, os resultados do *survey* serviram para elaborar o modelo HyperTest, o modelo de percepção do esforço de teste de software proposto neste trabalho.

O modelo proposto possibilita compreender como diferentes fatores afetam o esforço em seis atividades do processo de teste, de acordo com seis configurações de estratégia de testes distintas. Os resultados apresentados tanto no *survey* quanto na avaliação do modelo sugerem que ele pode contribuir não apenas no processo de estimativa de esforço de teste, mas também em outras processos que envolvam a tomada de decisão com base no esforço do processo de teste.

7.2 Contribuições

As principais contribuições deste trabalho dizem respeito à organização do conhecimento sobre esforço em teste de software, tanto aquele conhecimento relatado na literatura técnica, quanto o conhecimento de profissionais que atuam em projetos de software realizando testes.

Nesse sentido, a primeira contribuição foi a identificação e caracterização dos fatores que influenciam o esforço das atividades de teste de software sob a luz de diversas configurações de estratégia de teste. Diferentemente da maioria dos trabalhos na área, este trabalho procurou contribuir com a compreensão do fenômeno do esforço, em vez de tentar simplesmente estimá-lo. Eventualmente, essa compreensão pode proporcionar a construção de modelos de estimativa com maiores níveis de acurácia.

Essa caracterização dos fatores de esforço relacionados ao teste de software se deu, especialmente, por meio da Revisão Sistemática da Literatura e do *Survey*

apresentados nos capítulos 3 e 6, respectivamente. Até o momento, não foram identificados outros estudos primários ou secundários que tenham detalhado tão a fundo os fatores de esforço de teste.

Por sua vez, o estudo envolvendo Linguística de *Corpus*, apresentado no Apêndice D, representa uma importante contribuição para as pesquisas voltadas à terminologia em Engenharia de Software, já que apresenta um protocolo que pode ser aproveitado por outros pesquisadores. Além disso, esse estudo evidencia a confusão conceitual existente na literatura envolvendo os termos "esforço" e "custo".

A principal contribuição deste estudo é o modelo HyperTest. O modelo consolida o conhecimento coletado por meio de um *survey* com diversos profissionais, apresentando de forma contextualizada diversos pontos de vista sobre o esforço em atividades de teste de software. Um gerente de projetos de testes, quando do planejamento de um projeto, pode consultar o HyperTest para subsidiar suas decisões e estimativas relacionadas ao esforço do projeto. Além disso, o modelo pode ser útil para identificação de gargalos em processos de teste de software, à medida que expõe os fatores que influenciam o esforço.

Além dessas contribuições, o trabalho também proporcionou alguns resultados negativos. Após o exame de qualificação (março/2016), tentou-se construir um modelo de desempenho baseado em regressão linear. No entanto, essa tentativa esbarrou em algumas dificuldades que inviabilizaram a construção daquele modelo. O trabalho, até então, havia catalogado 51 fatores de esforço, a maioria qualitativos. Não havia um *dataset* disponível com dados de projetos contemplando tantos fatores. Também não se sabia ao certo quais daqueles fatores deveriam ser considerados e como poderiam ser medidos. Assim, foi realizado um teste com um *dataset* de projetos de teste de *web services* que contemplava seis variáveis quantitativas, presentes no conjunto de 51 fatores e indicadas pelos responsáveis do projeto como fortemente relacionadas ao esforço de teste. Naquele projeto, o esforço era medido em horas. No entanto, ao realizar o teste de correlação de Pearson, apenas um fator apresentou algum nível de correlação com o esforço, ainda assim considerado baixo (0,602). Assim, desistiu-se de elaborar um modelo de desempenho e partiu-se para a definição de um modelo que capturasse a percepção sobre os fatores que afetam o esforço de teste de software sob a perspectiva de profissionais da indústria.

7.3 Limitações

Apesar das contribuições apresentadas, este trabalho possui algumas limitações, em especial no *survey* e no modelo HyperTest. No que se refere ao *survey*, foram identificadas as seguintes limitações:

- *Amostragem* - A amostragem do *survey* foi realizada por conveniência, tendo em vista que os participantes fazem parte do convívio do pesquisador. Uma extensão deste trabalho poderia envolver amostras mais representativas e diversificadas.
- *Configurações* – O estudo é limitado a seis configurações de estratégia de teste. Apesar de ser uma quantidade significativa de configurações, dado o volume de trabalho e de dados produzidos por cada uma, há muitas outras configurações inexploradas e que podem ser úteis a determinados contextos de projeto.
- *Análise qualitativa* – Até o momento, o estudo apresenta uma análise de viés quantitativo. No entanto, é necessário analisar os resultados sob uma perspectiva mais qualitativa, discutindo os diversos pontos de vista apresentados pelos participantes a respeito do esforço de teste.

Em relação ao modelo HyperTest, foram percebidas as seguintes limitações:

- *Rationale* – Apesar de coletadas por meio de entrevistas, as justificativas a respeito de cada julgamento não foram consolidadas por completo. É necessário consolidar o *rationale* dos julgamentos para concluir o HyperTest.
- *Avaliação do modelo* – O modelo foi avaliado por um conjunto reduzido de profissionais e somente na perspectiva de apoiar a estimativa de esforço. É necessário, portanto, que o modelo seja avaliado por outros profissionais, que não tenham participado do *survey*, e sob outras perspectivas de aplicação.
- *Conteúdo dinâmico* – Atualmente, a aplicação que representa o modelo está implementada como páginas Web de conteúdo estático. Considerando sua possível evolução, incluindo novas percepções sobre o esforço de teste, é fundamental que o modelo seja implementado como uma aplicação Web dinâmica, processando o conteúdo de um banco de dados, de forma a manter seu conteúdo atualizado automaticamente.

7.4 Trabalhos Futuros

As limitações apresentadas representam possibilidades de extensão e melhoria deste trabalho. Porém, o material que já está elaborado também oferece perspectivas de trabalhos futuros, listadas a seguir:

- *Publicações* – Até o término deste trabalho, alguns estudos, apesar de finalizados, não foram publicados ou foram publicados apenas parcialmente. Desta forma, para cada um dos estudos a seguir, pretende-se submeter um artigo para um periódico:
    - *Revisão Sistemática da Literatura* – Em agosto de 2018, um artigo completo sobre a RSL foi submetido ao *Journal of Systems and Software*. O revisor responsável respondeu sugerindo que o artigo seja resubmetido desde que seja realizada uma série de alterações.
    - *Estudo de Observação* – O estudo de observação foi parcialmente publicado no Simpósio Brasileiro de Teste Sistemático e Automatizado (SAST 2017) (SILVA-DE-SOUZA; TRAVASSOS, 2017), onde recebeu o prêmio de melhor trabalho. Como parte da premiação, há um convite para publicação de uma versão estendida daquele artigo, que contemplou somente a atividade "Projetar e Implementar Testes". Uma versão estendida pode apresentar o estudo por completo.
    - *Survey e HyperTest* – Apesar de representar a parte mais densa deste trabalho, o conteúdo correspondente ao *survey* e ao modelo HyperTest ainda não foi publicado. Pretende-se, portanto, submeter um artigo que consolide o *survey* e o modelo proposto.
- *Disponibilização do pacote experimental* – Seguindo a tendência recente da academia de disponibilizar pacotes experimentais de pesquisas, pretende-se disponibilizar o *dataset* que reúne os dados coletados no *survey*. Com isso, outros pesquisadores poderão explorar os dados e, eventualmente, contribuir com a pesquisa nesta área.
- *Ciência de Dados aplicada à Engenharia de Software* – A principal perspectiva de pesquisa decorrente deste trabalho está na possível aplicação de conceitos e tecnologias relacionadas à Ciência de Dados, incluindo *Big Data Analytics* e Aprendizado de Máquina, para resolver problemas típicos da Engenharia de

Software, incluindo o esforço de teste. Este trabalho limitou-se a aplicar técnicas tradicionais da Estatística sobre os dados coletados. As técnicas do campo da Ciência de Dados poderiam abrir uma variedade de oportunidades de pesquisa em Engenharia de Software. Portanto, pretende-se futuramente explorar as diversas vertentes da Ciência de Dados com o intuito de trazer contribuições à Engenharia de Software.

REFERÊNCIAS BIBLIOGRÁFICAS

1. ADALI, O. E. et al. **Software Test Effort Estimation:** State of the Art in Turkish Software Industry. 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). [S.l.]: IEEE. 2017. p. 412-420.

2. ALMEIDA, E.; ABREU, B.; MORAES, R. **An alternative approach to test effort estimation based on use cases**. Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation (ICST 2009). Denver, CO: IEEE. 2009. p. 279-288.

3. APSHVALKA, D.; WENDORFF, P. Reflections on the Body of Knowledge in Software Engineering. In: NILSSON, A. G., et al. **Advances in Information Systems Development**. [S.l.]: Springer, v. 1, 2006. p. 995-1006.

4. ARANHA, E.; BORBA, P. **An estimation model for test execution effort**. Proceedings of the First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). Madrid: ACM/IEEE Computer Society. 2007. p. 107-116.

5. ARANHA, E.; BORBA, P. Estimating manual test execution effort and capacity based on execution points. **International Journal of Computers and Applications**, v. 31, n. 3, p. 167-172, 2009.

6. ATLAS.TI SCIENTIFIC SOFTWARE DEVELOPMENT GMBH. ATLAS.ti. **ATLAS.ti**, 2016. Disponivel em: <http://atlasti.com>. Acesso em: fev. 2016.

7. BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. Goal Question Metrics paradigm. In: MARCINIAK, J. J. **Encyclopedia of Software Engineering**. [S.l.]: Wiley, 1994. p. 528–532.

8. BINDER, R. V. **Testing object-oriented systems:** models, patterns, and tools. Upper Saddle River: Pearson, 2000.

9. BIOLCHINI, J. et al. **Systematic Review in Software Engineering**. COPPE/UFRJ. Rio de Janeiro, p. 1-31. 2005.

10. BLACK, R. **Advanced software testing:** guide to the ISTQB advanced certification as an advanced test analyst. Santa Barbara: Rocky Nook, 2011.

11. BOEHM, B. **Software engineering economics**. Upper Saddle River: Prentice Hall, 1981.

12. BOEHM, B. W. et al. **Software cost estimation with COCOMO II**. Upper Saddle River: Prentice Hall, 2000.

13. BOTELHO, J. M. **Causas e consequências da dialetação da língua latina. Um pouco de história externa da língua portuguesa**. Cadernos do XIV Congresso Nacional de Linguística e Filologia. Rio de Janeiro: CiFEFiL. 2010. p. 231-243.

14. BOURQUE, P.; FAIRLEY, R. E. **SWEBOK: Guide to the Software Engineering Body of Knowledge**. Los Alamitos, CA. 2014.

15. BROOKS JR, F. P. **The Mythical Man-Month:** Essays on Software Engineering. 2. ed. ed. [S.l.]: Pearson Education India, 1995.

16. CAMBRIDGE UNIVERSITY PRESS. Cambridge English Dictionary & Thesaurus. **Cambridge Dictionaries Online**, 2015. Disponivel em: <http://dictionary.cambridge.org>. Acesso em: 2015.

17. CARDOSO, C. G. S. **A matemática das eleições**. Universidade de Lisboa. Lisboa, p. 186. 2009.

18. CELCE-MURCIA, M.; LARSEN-FREEMAN, D. **The grammar book:** an ESL/EFL teacher's course. Stamford: Heinle Cengage Learning, 1999.

19. CHOI, J.; CHOI, I. Happiness is medal-color blind: Happy people value silver and bronze medals more than unhappy people. **Journal of Experimental Social Psychology**, v. 68, p. 78-82, 2017.

20. COGAN, S. **Custos e formação de preços:** análise e prática. São Paulo: Atlas, 2013.

21. CRESPO, A. N. et al. **Uma metodologia para teste de Software no Contexto da Melhoria de Processo**. Anais do III Simpósio Brasileiro de Qualidade de Software (SBQS 2004). [S.l.]: SBC. 2004. p. 271-285.

22. CRYSTAL, D. **Dicionário de linguística e fonética**. Rio de Janeiro: Jorge Zahar, 2008.

23. CUNHA, A. G. D. **Dicionário etimológico da língua portuguesa**. 4. ed. ed. Rio de Janeiro: Lexikon, 2013.

24. DE FRANÇA, B. B. N. **Guidelines for Experimentation with Dynamic Simulation Models in the context of Software Engineering**. Universidade Federal do Rio de Janeiro. Rio de Janeiro, p. 165. 2015.

25. DEONANDAN, I. et al. **Cost and risk considerations for test and evaluation of unmanned and autonomous systems of systems**. Proceedings - 5th International Conference on System of Systems Engineering (SoSE). [S.l.]: [s.n.]. 2010. p. 1-6.

26. DIAS-NETO, A. C. et al. Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay. **Software Quality Journal**, v. 25, n. 4, p. 1145-1183, 2017.

27. DICTIONARY.COM. Dictionary.com, 2015. Disponivel em: <dictionary.reference.com>. Acesso em: nov. 2015.

28. ELBERZHAGER, F. et al. Reducing test effort: A systematic mapping study on existing approaches. **Information and Software Technology**, v. 54, n. 10, p. 1092-1106, 2012.

29. FARR, L.; NANUS, B. **Factors that Affect the Cost of Computer Programming - volume I**. Santa Monica. 1964.

30. FARR, L.; ZAGORSKI, H. J. **Factors that Affect the Cost of Computer Programming - volume II**. Santa Monica. 1964.

31. FENTON, N.; BIEMAN, J. **Software metrics:** a rigorous and practical approach. 3rd. ed. Boca Raton, FL: CRC Press, 2015.

32. FERNÁNDEZ-DIEGO, M.; GONZÁLEZ-LADRÓN-DE-GUEVARA, F. Potential and limitations of the ISBSG dataset in enhancing software engineering research: a mapping review. **Journal of Information and Software Technology**, v. 56, n. 6, p. 527-544, 2014.

33. FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. University of California. Irvine. 2000.

34. FIRESMITH, D. G. **Common system and software testing pitfalls:** how to prevent and mitigate them: descriptions, symptoms, consequences, causes, and recommendations. [S.l.]: Addison-Wesley, 2014.

35. FORSBERG, K.; MOOZ, H. **The Relationship of System Engineering to the Project Cycle**. Proceedings of the National Council for Systems Engineering First Annual Conference. Chattanooga, TN: [s.n.]. 1991. p. 57-61.

36. FRAGAL, V. H. et al. **Reducing the Concretization Effort in FSM-Based Testing of Software Product Lines**. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). Tokyo: IEEE. 2017. p. 329-336.

37. FUZER, C.; CABRAL, S. R. S. **Introdução à gramática sistêmico-funcional em Língua Portuguesa**. Campinas: Mercado de Letras, 2014.

38. GOOGLE. Google Books Ngram Viewer. **Google Books Ngram Viewer**, 2016. Disponivel em: <https://books.google.com/ngrams>. Acesso em: jan. 2016.

39. GOOGLE. Google Trends. **Google Trends**, 2018. Disponivel em: <https://www.google.com.br/trends/>. Acesso em: jan. 2018.

40. GRIMSTAD, S.; JøRGENSEN, M.; MOLøKKEN-ØSTVOLD, K. Software effort estimation terminology: The tower of Babel. **Information and Software Technology**, v. 48, n. 4, p. 302-310, 2006.

41. HEINZE, K.; CLAUSSEN, N.; LABOLLE, V. **Management of Computer Programming for Command and Control Systems: a Survey**. Santa Monica. 1963.

42. ISO/IEC/IEEE. **ISO/IEC/IEEE 29119-1 - Software and systems engineering - Software testing - Part 1: Concepts and definitions**. ISO/IEC/IEEE. Geneva. 2013.

43. ISO/IEC/IEEE. **ISO/IEC/IEEE 29119-2 - Software and systems engineering - Software testing - Part 2: Test processes**. ISO/IEC/IEEE. Geneva. 2013.

44. ISO/IEC/IEEE. **ISO/IEC/IEEE 29119-3 - Software and systems engineering - Software testing - Part 3: Test documentation**. ISO/IEC/IEEE. Geneva. 2013.

45. JøRGENSEN, M. **A critique of how we measure and interpret the accuracy of software development effort estimation**. International Workshop on Software Productivity Analysis and Cost Estimation. Nagoya: Information Processing Society of Japan. 2007.

46. JøRGENSEN, M.; SHEPPERD, M. A Systematic Review of Software Development Cost Estimation Studies. **IEEE Transactions on Software Engineering**, Los Alamitos, CA, v. 33, n. 1, p. 33-53, 2007.

47. JALALI, S.; WOHLIN, C. **Systematic literature studies:** database searches vs. backward snowballing. Proceeedings of the ESEM 2012. New York: ACM. 2012. p. 29-38.

48. JCGM. **International vocabulary of metrology – Basic and general concepts and associated terms (VIM)**. Joint Committee for Guides in Metrology. [S.l.]. 2012.

49. JONES, C. **Estimating software costs:** bringing realism to estimating. 2nd. ed. New York: McGraw-Hill, 2007.

50. KUMARI, A.; SHARMA, A. **Test effort estimation in regression testing**. Proceedings of the 2013 International Conference in MOOC, Inovation and Technology in Education. [S.l.]: [s.n.]. 2013. p. 343-348.

51. LEXICAL COMPUTING. Sketch Engine. **Sketch Engine**, 2015. Disponivel em: <https://www.sketchengine.co.uk>. Acesso em: nov. 2015.

52. LU, Y.-F.; YIN, Y.-F. **A new constructive cost model for software testing project management**. Proceedings of the 19th International Conference on Industrial Engineering and Engineering Management. [S.l.]: Springer. 2013. p. 545-556.

53. MATOS, D. A. S. Confiabilidade e concordância entre juízes: aplicações na área educacional. **Estudos em Avaliação Educacional**, São Paulo, v. 25, n. 59, p. 298-324, set./dez. 2014.

54. MELLO, R. M. **Conceptual framework for supporting the identification of representative samples for surveys in software engineering**. Universidade Federal do Rio de Janeiro. Rio de Janeiro, p. 138. 2016.

55. MENDES, E. **Cost estimation techniques for web projects**. Hershey: IGI Publishing, 2008.

56. MENDES, E. **Practitioner's knowledge representation:** a pathway to improve software effort estimation. Heidelberg: Springer, 2014.

57. MENDES, E.; VAZ, V. T.; MURADAS, F. **An expert-based requirements effort estimation model using bayesian networks**. International Conference on Software Quality. Cham: Springer. 2016. p. 79-93.

58. MERRIAM-WEBSTER. Merriam-Webster: Dictionary and Thesaurus, 2015. Disponivel em: <www.merriam-webster.com/>. Acesso em: nov. 2015.

59. MÄNTYLÄ, M. V.; ITKONEN, J. More testers–The effect of crowd size and time restriction in software testing. **Information and Software Technology**, v. 55, n. 6, p. 986-1003, 2013.

60. MIZUNO, O. et al. **On estimating testing effort needed to assure field quality in software development**. Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSRE). Annapolis: [s.n.]. 2002. p. 139-146.

61. MU, E.; PEREYRA-ROJAS, M. **Practical Decision Making:** An Introduction to the Analytic Hierarchy Process (AHP) Using Super Decisions. [S.l.]: Springer, 2016.

62. NAGESWARAN, S. **Test effort estimation using use case points**. Proceedings of 14th International Software/Internet Quality Week (QW2001). San Francisco: Software Research Institute. 2001. p. 1-6.

63. PAI, M. et al. Systematic Reviews and meta-analyses: An illustrated, step-by-step guide. **The National Medical Journal of India**, v. 17, n. 2, p. 86-95, 2004.

64. PFLEEGER, S. L. **Engenharia de software:** teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.

65. POL, M.; TEUNISSEN, R.; VEENENDAAL, E. V. **Software testing:** a guide to the TMap approach. Harlow: Pearson, 2002.

66. PRESSMAN, R. S. **Engenharia de software**. 6. ed. São Paulo: McGraw-Hill, 2006.

67. PUTNAM, L. H.; MYERS, W. **Five core metrics:** the intelligence behind successful software management. New York: Dorset House Publishing, 2003.

68. RAINER, A. et al. **Teaching Empirical Methods to Undergraduate Students Working Group Results**. Empirical Software Engineering Issues. Critical Assessment and Future Directions. Heidelberg: Springer. 2007. p. 158-162.

69. RIBEIRO, O. M. **Contabilidade de custos**. 4. ed. São Paulo: Saraiva, 2015.

70. ROOK, P. Controlling software projects. **Software Engineering Journal**, Stevenage, v. 1, n. 1, p. 7-16, January 1986.

71. ROYCE, W. W. **Managing the Development of Large Software Systems**. Proceedings of the IEEE WESCON. [S.l.]: IEEE. 1970. p. 1-9.

72. RUNESON, P. et al. **Case Study Research in Software Engineering:** Guidelines and Examples. 1. ed. [S.l.]: Wiley Publishing, 2012.

73. SARDINHA, T. B. **Linguística de Corpus**. Barueri: Manole, 2004.

74. SHAH, H.; SINHA, S.; HARROLD, M. J. **Outsourced, Offshored Software-Testing Practice:** Vendor-Side Experiences. Proceedings of the 6th IEEE International Conference on Global Software Engineering. Helsinki: IEEE. 2011. p. 131-140.

75. SILVA, D. G.; ABREU, B. T.; JINO, M. **A simple approach for estimation of execution effort of functional test cases**. Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation (ICST 2009). Denver, CO: IEEE. 2009. p. 289-298.

76. SILVA-DE-SOUZA, T. **Towards Effort Estimation in Software Testing Projects**. Proceedings of the ICSE 2017 PhD and Young Researchers Warm Up Symposium. Maceió: SBC. 2014. p. 74-75.

77. SILVA-DE-SOUZA, T. **Reflexões sobre o Esforço em Testes de Desempenho de Produtos de Software**. II Workshop em Qualidade de Produtos de Software. Rio de Janeiro: SBC. 2017.

78. SILVA-DE-SOUZA, T. et al. **Testes Funcionais de Web Services RESTful a partir de Modelos UML**. Anais do SBQS 2012 - XI Simpósio Brasileiro de Qualidade de Software. Fortaleza: SBC. 2012.

79. SILVA-DE-SOUZA, T.; PIRES, P. F. Tecnologia de Serviços Web: Conceitos e Aplicações. In: _____ **Minicursos do 20º SBBD e 19º SBES**. Uberlândia: Sociedade Brasileira de Computação, 2005. Cap. 11, p. 303-335.

80. SILVA-DE-SOUZA, T.; RIBEIRO, V. V.; TRAVASSOS, G. H. **Estimativa de Esforço em Teste de Software:** Modelos, Fatores e Incertezas. Libro de Actas - XX Congreso Argentino de Ciencias de la Computación (CACIC). Buenos Aires: [s.n.]. 2014.

81. SILVA-DE-SOUZA, T.; SILVA, T. M.; TRAVASSOS, G. H. **Análise Linguístico-Científica dos Termos "Esforço" e "Custo" e sua Relação com o Software:** Um Estudo Qualitativo Apoiado pela Linguística de Corpus. Proceedings of the 13th Empirical Software Engineering Latin American Workshop (ESELAW 2016). Quito: Universidad de las Fuerzas Armadas "ESPE". 2016.

82. SILVA-DE-SOUZA, T.; TRAVASSOS, G. H. **Observing Effort Factors in the Test Design & Implementation Process of Web Services Projects**. Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing (SAST). New York, NY: ACM. 2017.

83. SMARTBEAR. SoapUI, 2015. Disponivel em: <http://www.soapui.org>. Acesso em: 2015.

84. SOFTEX. **Guia de Implementação - Parte 10: Implementação do MR-MPS em organizações do tipo Fábrica de Teste**. SOFTEX. [S.l.]. 2011.

85. STRAUSS, A.; CORBIN, J. **Basics of qualitative research:** techniques and procedures for developing Grounded Theory. 2. ed. London: SAGE Publications, 1998.

86. SUBRAMANIAN, G. H.; PENDHARKAR, P. C.; PAI, D. R. An Examination of Determinants of Software Testing and Project Management Effort. **Journal of Computer Information Systems**, v. 57, n. 2, p. 123-129, 2017.

87. TAGNIN, S. E. O. **O jeito que a gente diz:** combinações consagradas em Inglês e Português. Barueri: Disal, 2013.

88. TAYLOR, F. W. **Princípios de administração científica**. 8. ed. São Paulo: Atlas, 1995.

89. THE ONLINE ETYMOLOGY DICTIONARY. The Online Etymology Dictionary. **The Online Etymology Dictionary**, 2001. Disponivel em: <http://www.etymonline.com>. Acesso em: nov. 2015.

90. TRAVASSOS, G. H. et al. **An environment to support large scale experimentation in software engineering**. Proceedings of the 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008). [S.l.]: IEEE. 2008. p. 193-202.

91. TRENDOWICZ, A.; JEFFERY, R. **Software Project Effort Estimation:** Foundations and Best Practice Guidelines for Success. [S.l.]: Springer, 2014.

92. TRIOLA, M. F. **Introdução à estatística**. Rio de Janeiro: LTC, 2008.

93. UTTING, M.; LEGEARD, B. **Practical Model-Based Testing:** A Tools Approach. San Francisco: Morgan Kaufmann Publishers Inc., 2007.

94. VAN VEENENDAAL, E. Testpuntanalyse: Methode om het testen te begroten. **Computable**, Utrecht, 28, Maio 1995. 25-27.

95. VAZ, V. T. **Estimativa de Esforço em Projetos de Especificação de Requisitos de Software**. Universidade Federal do Rio de Janeiro (UFRJ). Rio de Janeiro, p. 104. 2013.

96. WOHLIN, C. **Writing for Synthesis of Evidence in Empirical Software Engineering**. Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. New York, NY: ACM. 2014.

97. WOHLIN, C. et al. **Experimentation in software engineering:** an introduction. Norwell: Kluwer Academic Publishers, 2000.

98. XIAOCHUN, Z. et al. **Estimate Test Execution Effort at an Early Stage:** An Empirical Study. Proceedings of the International Conference on Cyberworlds 2008. Hangzhou: IEEE Computer Society. 2008. p. 195-200.

99. ZELKOWITZ, M. V. **Data sharing enabling technologies working group results**. Empirical Software Engineering Issues. Critical Assessment and Future Directions. Heidelberg: Springer. 2007. p. 108-110.

100. ZHU, X.; ZHOU, B.; CHEN, L. **Software testing sizing in incremental development:** a case study. Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009). Lake Buena Vista, FL: IEEE. 2009. p. 490- 493.

APÊNDICE A – Lista de Artigos Incluídos na Revisão Sistemática

Artigos de Controle

RS1. Nageswaran, S. (2001). Test effort estimation using use case points. *Proceedings of 14th International Software/Internet Quality Week (QW2001)* (p. 1-6). San Francisco: Software Research Institute.

RS2. Aranha, E. & Borba, P. (2007). An estimation model for test execution effort. *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, (p. 107-116). Madrid, Spain.

RS3. Xiaochun, Z., Bo, Z., Fan, W., Yi, Q. & Lu, C. (2008). Estimate Test Execution Effort at an Early Stage: An Empirical Study. *Proceedings of the International Conference on Cyberworlds 2008*, (p. 195-200).

RS4. Almeida, E., Abreu, B. & Moraes, R. (2009). An alternative approach to test effort estimation based on use cases. *Proceedings - 2nd International Conference on Software Testing, Verification, and Validation, ICST 2009*, (p. 279-288). Denver, CO.

RS5. Aranha, E. & Borba, P. (2009). Estimating manual test execution effort and capacity based on execution points. *International Journal of Computers and Applications, 31* (3), 167-172.

RS6. Silva, D. G., Abreu, B. T. & Jino, M. (2009). A simple approach for estimation of execution effort of functional test cases. *Proceedings - 2nd International Conference on Software Testing, Verification, and Validation, ICST 2009*, (p. 289-298). Denver, CO.

RS7. Zhu, X., Zhou, B. & Chen, L. Software Testing Sizing in Incremental Development: A Case Study. *Proceedings - 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, (p. 491-494). New York.

Artigos Retornados pelas Máquinas de Busca

RS8. Cheatham, T. J., Yoo, J. P. & Wahl, N. J. (1995). Software testing: a machine learning experiment. *Proceedings - ACM Computer Science Conference*, (p. 135-141). Nashvile.

RS9. Dawson, C. (1998). Artificial neural network approach to software testing effort estimation. *Transactions on Information and Communications Technologies, 20*, 145-155.

RS10. Mizuno, O., Shigematsu, E., Takagi, Y. & Kikuno, T. (2002). On estimating testing effort needed to assure field quality in software development. *Proceedings - 13th International Symposium on Software Reliability Engineering, 2002,* (p. 139-146). Annapolis, MD.

RS11. Naunchan, P. & Sutivong, D. (2007). Adjustable Cost Estimation Model for COTS-Based Development. *Proceedings - 18th Australian Software Engineering Conference, ASWEC 2007* (p. 341-348). Melbourne: IEEE Computer Society.

RS12. Zhu, X., Zhou, B., Hou, L., Chen, J., & Chen, L. (2008). An experience-based approach for test execution effort estimation. *Proceedings - 9th International Conference for Young Computer Scientists* (p. 1193-1198). Hunan: IEEE Computer Society.

RS13. Lazic, L. & Mastorakis, N. (2009). The COTECOMO: COnstractive Test Effort COst MOdel. *Proceedings of the European Computing Conference. 2*, p. 89-110. Lecture Notes in Electrical Engineering (Springer).

RS14. Marchetto, A. (2009). OQMw: An OO quality model for web applications. *Tamkang Journal of Science and Engineering , 12* (4), 459-470.

RS15. Shaheen, M. R. & Du Bousquet, L. (2009). Is depth of inheritance tree a good cost prediction for branch coverage testing? *Proceedings - 1st International Conference on Advances in System Testing and Validation Lifecycle, VALID 2009*, (p. 42-47). Porto.

RS16. Budnik, C., Subramanyan, R. & Tanikella, R. (2010). Scalable V&V Effort Estimation for Ultra-Large-Scale Systems. *Proceedings - 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI)*, (p. 60-68).

RS17. Deonandan, I., Valerdi, R., Lane, J., & Macias, F. (2010). Cost and risk considerations for test and evaluation of unmanned and autonomous systems of systems. *Proceedings - 5th International Conference on System of Systems Engineering (SoSE)*, (p. 1-6).

RS18. Silva, D. G., Jino, M. & de Abreu, B. T. (2010). Machine learning methods and asymmetric cost function to estimate execution effort of software testing. *Proceedings - 3rd International Conference on Software Testing, Verification and Validation (ICST 2010)*, (p. 275-284). Paris.

RS19. Aloka, S., Singh, P., Rakshit, G. & Srivastava, P. (2011). Test effort estimation-particle swarm optimization based approach. *Communications in Computer and Information Science, 168*, 463-474.

RS20. Sharma, A. & Kushwaha, D. S. (2011). A metric suite for early estimation of software testing effort using requirement engineering document and its validation. *Proceedings - 2nd International Conference on Computer and Communication Technology, ICCCT-2011*, (p. 373-378). Allahabad, India.

RS21. Srivastava, P., Kumar, S., Singh, A. & Raghurama, G. (2011). Software testing effort: An assessment through fuzzy criteria approach. *Journal of Uncertain Systems, 5* (3), 183-201.

RS22. Bhattacharya, P., Srivastava, P. & Prasad, B. (2012). Software test effort estimation using particle swarm optimization. *Advances in Intelligent and Soft Computing, 132*, 827-835.

RS23. Srivastava, P. R., Varshney, A., Nama, P. & Yang, X.-S. (2012). Software test effort estimation: a model based on cuckoo search. *International Journal of Bio-Inspired Computation , 4* (5), 278-285.

RS24. Kumari, A. & Sharma, A. (2013). Test effort estimation in regression testing. *Proceedings - 2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, (p. 343-348). Jaipur.

RS25. Fedotova, O., Teixeira, L. & Alvelos, H. (2013). Software Effort Estimation with Multiple Linear Regression: Review and Practical Application. *Journal of Information Science and Engineering, 29* (5), 925-945.

RS26. Lu, Y.-f. & Yin, Y.-f. (2013). A New Constructive Cost Model for Software Testing Project Management. *Proceedings - The 19th International Conference on Industrial Engineering and Engineering Management* (p. 545-556). Springer.

RS27. Nguyen, V., Pham, V. & Lam, V. (2013). qEstimation: A Process for Estimating Size and Effort of Software Testing. *Proceedings of the 2013 International Conference on Software and System Process* (p. 20-28). New York, NY: ACM.

RS28. Parvez, A. W. (2013). Efficiency factor and risk factor based user case point test effort estimation model compatible with agile software development. *Proceedings - 2013*

International Conference on Information Technology and Electrical Engineering (ICITEE), (p. 113-118).

Artigos Retornados por *Snowballing*

RS29. Abhishek, C., Kumar, V. P., Vitta, H. & Srivastava, P. R. (2010). Test Effort Estimation Using Neural Network. *Journal of Software Engineering and Applications*, 3 (4), 331-340.

RS30. Torkar, R., Awan, N. M., Alvi, A. K. & Afzal, W. (2010). Predicting Software Test Effort in Iterative Development Using a Dynamic Bayesian Network. *Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering.* San Jose.

RS31. Ashraf, M. D. & Janjua, N. U. (2011). Test Execution Effort Estimation (TEEE) Model in Extreme Programming. *International Journal of Reviews in Computing , 8* (4), 35-45.

RS32. Sharma, A. & Kushwaha, D. S. (2013). An empirical approach for early estimation of software testing effort using SRS documentation. *CSI Transactions on ICT , 1* (1), 51-66.

RS33. Srivastava, P. R., Bidwai, A., Khan, A., Rathore, K., Sharma, R. & Yang, X. S. (2014). An empirical study of test effort estimation based on bat algorithm. *International Journal of Bio-Inspired Computation, 6* (1), 57-70.

APÊNDICE B – Formulário de Extração

Extraction Form

Version History				
Date	**Version**	**Description**	**Extractor**	**Checker**
<dd/mm/aaaa>	<x.x>	<details>	<name>	<name>

General Information	
Title	
Authors	
Year of publication	
Publication source	
Abstract	
Search engine	[ ] Scopus [ ] IEEE Xplore [ ] Web of Science [ ] Engineering Village [ ] Other. Which one? _____
Technique Detailing	
Name of the technique	
Concept of effort	
Effort measurement unit	
Type of technique	
Application context	
Test level	[ ] Unit (component) [ ] Integration [ ] System [ ] Acceptance [ ] Not specified
Test type	[ ] Functional [ ] Non-functional [ ] Other. Which one? _____ [ ] Not specified
Test technique	[ ] Structural testing (white-box) [ ] Functional testing (black-box) [ ] Not specified
Test phase	[ ] Planning [ ] Design [ ] Execution [ ] Reporting [ ] Other. Which one? _____ [ ] Not specified

Test execution form	[ ] Manual [ ] Automated [ ] Not specified
Usage restrictions	
Influence Factors	
Influence factors list	
How to measure or observe these factors?	
Exerted influence	
Estimation Model	
Input parameters	
Model description	
How was the model built?	
Accuracy metric(s)	[ ] Deviation _____% [ ] MRE _____% [ ] MMRE _____% [ ] PRED(__)_____% [ ] Other. Which one? _____ _____% [ ] Not specified
Study Evaluation	
Type of study	[ ] Experiment [ ] *quasi*-Experiment [ ] Case study [ ] Survey [ ] Simulation [ ] Proof of concept
Study subjects	
Study results	

Quality Assessment							
Q1		**Q2**		**Q3**		**Q4**	
Q5		**Q6**		**Q7**		**Q8**	
Q9		**Q10**		**Q11**		**Q12**	
Total							

APÊNDICE C – Legenda do Formulário de Extração

1. **General Information**

   1.1. Title: title of the paper.

   1.2. Authors: authors of the paper.

   1.3. Year of publication: year of publication of the paper.

   1.4. Publication source: the journal or proceedings in which the paper was published.

   1.5. Abstract: summary of the paper.

   1.6. Search engine: search engine by which the paper was found: “Scopus”, “IEEE Xplore”, “Web of Science” and “Engineering Village” (check one or more). If the paper was included because it is considered relevant but it is not in cited digital libraries should be marked as "Other".

2. **Technique Detailing**

   2.1. Name of the technique: name of the effort estimation technique presented in the paper.

   2.2. Concept of effort: how the authors define the concept of effort in software testing.

   2.3. Effort measurement unit: measurement unit for effort representation (e.g. man-hour).

   2.4. Type of technique: type of technique presented in the paper, according to the classification of its authors.

   2.5. Application context: description of the application domain or kind of system for which the estimation approach was proposed or evaluated.

   2.6. Test level: test level for which the estimation approach was proposed, according to V model. Check one or more of the following options: “Unit”, “Integration”, “System”, “Acceptance” and “Not specified”.

   2.7. Test type: test level for which the estimation approach was proposed, according to the type of requirement to be verified or validated. Check one or more of the following options: “Functional” and “Non-functional”. There are also the options “Other” and “Not specified”.

   2.8. Test technique: test technique for which the estimation approach was proposed. Check one or more of the following options: “Structural Testing”

(white-box) e “Functional Testing” (black-box). There is also the option “Not specified”.

2.9. Test phase: test phase for which the estimation approach was proposed. Check one or more of the following options: “Planning”, “Design”, “Execution” and “Reporting”. There is also the option “Not specified”.

2.10. Test execution form:

- Manual – the approach considers the manual test execution.
- Automated – the approach considers the automated test execution.
- Not specified – if the test execution form is not specified.

2.11. Usage restrictions: pre-conditions and constraints to the proposed technique usage.

3. Influence Factors

3.1. Influence factors list: influence factors on the Software Testing activities reported in the paper.

3.2. How to measure or observe these factors? Description of how these factors can be measured or observed.

3.3. Exerted influence: description about the exerted influence on the software testing effort by each factor.

4. Estimation Model

4.1. Input parameters: input parameters required by the estimation model.

4.2. Model description: description about how to use the presented model.

4.3. How was the model built? Description about how the model was defined (e.g. from a historical database).

4.4. Accuracy metric(s): list of metrics used to indicate the model accuracy (e.g. PRED(0.25), MMRE).

5. Study Evaluation

5.1. Type of study: type of study that was performed to evaluate the proposed estimation approach.

- Experiment
- *quasi*-Experiment
- Case study

- Survey
- Simulation
- Proof of concept

5.2. Study subjects: kind of subject of the study (i.e. industry projects, professionals, students, etc.).

5.3. Study results: summary of the study results.

6. Quality Assessment

6.1. **Q1:** Does the paper presents a clear definition of what is effort in software testing? (1.0 point)

6.2. **Q2:** Does the paper clearly describe the factors that influence the software testing effort? (2 points)

- It presents influence factors with a description of how they can be measured or observed and indicates the value or direction of this influence (e.g. positive, negative) – 2 points
- It presents influence factors and indicates the value or direction of this influence (e.g. positive, negative) – 1.0 point
- It presents influence factors with a description of how they can be measured or observed – 1.0 point
- It just presents influence factors – 0.5 point
- It does not present any influence factors – 0 point

6.3. **Q3:** Does the paper presents a model for effort estimation in software testing? (2 points)

- There is an estimation model and there is a description of how this model was defined – 2 points
- There is an estimation model, but there is not a description of how this model was defined – 1.0 point
- There is not an estimation model – 0 point

6.4. **Q4:** Does the paper describe the application domain or kind of system for which the estimation approach was proposed? (1 point)

6.5. **Q5:** Does the paper describe the test level, type, technique and phase covered by the proposed approach?

- All of them – 2 points

- Only three of them – 1.5 point
- Only two of them – 1 point
- Only one of them – 0.5 point
- None of them – 0 point

6.6. **Q6:** Does the paper makes explicit the conditions and restrictions for using the estimation technique? (1 point)

6.7. **Q7:** Does the paper provide some indication of the accuracy of the proposed technique? (1 point)

6.8. **Q8:** What kind of study was performed to evaluate the proposed technique?

- Experiment – 2 points
- *quasi*-Experiment – 1.5 point
- Case study – 1 point
- Survey – 0.5 point
- Simulation – 0 point
- Proof of concept – 0 point

6.9. **Q9:** Does the study is described in an appropriate level of detail to allow an understanding of how the results were obtained and identify its limitations? (1 point)

6.10. **Q10:** Was there some statistical treatment on the data used in the study to ensure their consistency? (1 point)

6.11. **Q11:** What kind of study subject was used in the study? (1 point)

- Professional – 1 point
- Industry project – 1 point
- Student – 0.5 point
- Dummy project – 0 point

6.12. **Q12:** Does the paper describe appropriately the threats to validity of the study? (1 point)

- Threats to validity were identified and treated – 1 point
- Threats to validity were only identified – 0.5 point

Threats to validity were not identified – 0 point

APÊNDICE D – Análise Linguística[6]

Este apêndice discute os termos "esforço", "custo" e "homem-hora" a partir de uma perspectiva linguística. A seção D.1 introduz o contexto que motivou a realização do trabalho descrito neste apêndice. A seção D.2 apresenta o referencial teórico sobre estimativa de esforço e custo em projetos de software sob uma perspectiva histórica. A seção D.3 procura justificar a abordagem interdisciplinar utilizada e apresenta as questões de pesquisa. A seção D.4 analisa os termos "esforço", "custo" e "homem-hora" sob a luz da Etimologia. A seção D.5 investiga o sentido dado a esses termos por meio da Linguística de *Corpus*. A seção D.6 discute o resultados encontrados, ressaltando como os termos "esforço" e "custo" são utilizados em Engenharia de Software. Por fim, a seção D.7 destaca as considerações finais deste estudo.

D.1 Introdução

Nos últimos anos, diversos trabalhos têm apontado a necessidade de se consolidar a terminologia utilizada na área de Engenharia de Software (ES) (RAINER *et al.*, 2007) (ZELKOWITZ, 2007) (WOHLIN, 2014). Neste contexto, surge, por exemplo, o *Guide to the Software Engineering Body of Knowledge* (SWEBOK) (BOURQUE; FAIRLEY, 2014), conduzido pela IEEE *Computer Society*, com a intenção de categorizar e definir os conceitos relacionados à Engenharia de Software. No entanto, ainda não se pode observar na literatura técnica larga aceitação do SWEBOK, ou de instrumentos equivalentes. O motivo para esta constatação pode estar relacionado ao fato do SWEBOK representar "escolas de pensamento" específicas e por se basear em opiniões de especialistas, deixando de considerar as evidências experimentais (APSHVALKA; WENDORFF, 2006).

A comunidade de Engenharia de Software Experimental, em especial, considera a definição de uma terminologia comum em Engenharia de Software como fundamental para uma melhor compreensão dos objetos de estudo (ZELKOWITZ, 2007) e melhor comunicação dos achados para a indústria e para as futuras gerações de pesquisadores

[6] Uma versão deste capítulo foi publicada no artigo *Análise Linguístico-Científica dos Termos "Esforço" e "Custo" e sua Relação com o Software: Um Estudo Qualitativo Apoiado pela Linguística de Corpus,* apresentado no 13th *Empirical Software Engineering Latin American Workshop* (ESELAW 2016) (SILVA-DE-SOUZA; SILVA; TRAVASSOS, 2016).

(RAINER *et al.*, 2007). O uso de termos imprecisos pode potencializar inconsistências, promover ambiguidades e levar a observações inapropriadas dos objetos de estudo, aumentando, assim, as ameaças à validade de constructo em estudos primários e, especialmente, em estudos secundários.

A motivação deste estudo está relacionada à inconsistência conceitual observada no estudo secundário a respeito de fatores que influenciam o esforço em teste de software apresentado no capítulo 3, no qual foram definidas, entre outras, as seguintes questões de pesquisa específicas:

- *QE1: O que se entende por esforço em teste de software e como ele pode ser medido?*
- *QE2: Quais fatores influenciam o esforço em teste de software?*

Para respondê-las, uma *string* de busca, representada da Figura 7, foi definida com base na abordagem PICO (PAI *et al.*, 2004). Como já discutido no capítulo 3, as palavras-chave que compõem essa *string* foram identificadas a partir do conhecimento prévio dos pesquisadores e derivadas de sete artigos de controle selecionados através de uma busca *ad hoc*.

É possível observar no trecho da Figura 7 "... *effort estimation" OR "cost estimation" OR "estimate the cost" OR "effort prediction" OR "cost prediction" OR "effort measurement" OR "cost measurement* ..." que os termos "*effort*" (esforço) e "*cost*" (custo) tiveram que ser tratados como sinônimos, combinados par a par com os termos "*estimation*", "*prediction*" e "*measurement*". Isso se deveu a observação dos autores de que os termos esforço e custo são utilizados na literatura técnica de Engenharia de Software de modo intercambiável (MENDES, 2008) (MENDES, 2014).

Como mostrou o capítulo 3, após as etapas de busca, remoção de duplicatas e aplicação dos critérios de inclusão e exclusão, foram selecionados 37 trabalhos de um total de 735 artigos retornados. Logo após, iniciou-se a etapa de extração das informações de interesse do estudo. Ao tentar responder a questão QE1, observou-se que a literatura da área trata o conceito de esforço de forma tácita: apenas quatro dos 37 trabalhos selecionados apresentaram alguma definição sobre o que é esforço em teste de software. Mesmo assim, as definições apresentadas são incompletas e controversas, aumentando a incerteza sobre o objeto de estudo.

Em seguida, para selecionar e classificar os fatores que afetam o esforço em teste de software (questão QE2), era necessário ter uma definição sobre o que é

"esforço". Tendo em vista que os artigos selecionados não forneceram resposta à questão QE1 e que a *string* de busca utiliza termos em Inglês, foi utilizada a primeira definição de esforço do dicionário de *Cambridge*: "atividade física ou mental necessária para alcançar algum objetivo" (CAMBRIDGE UNIVERSITY PRESS, 2015). Com base nesta definição, procedeu-se a etapa de codificação utilizando o método *Grounded Theory* (STRAUSS; CORBIN, 1998). Nesta etapa, observou-se que, além da grande diversidade conceitual presente nos trabalhos selecionados, havia fatores que afetavam o "custo" do projeto, segundo a definição do mesmo dicionário (CAMBRIDGE UNIVERSITY PRESS, 2015), sendo atribuídos ao "esforço" necessário para realizá-lo. Percebeu-se ali um problema: "esforço" e "custo" não possuem necessariamente uma relação de sinonímia e, dependendo do contexto, podem assumir significados diversos, podendo levar a interpretações equivocadas. Este fenômeno de ausência de clareza conceitual pode, ainda, aumentar o nível de ruído inerente a *strings* de busca utilizadas em estudos secundários, bem como, dificultar a interpretação dos textos selecionados em função do nível de subjetividade e eventual ambiguidade dos termos.

O trabalho descrito neste capítulo, portanto, tem como objetivo principal investigar o significado dos termos "esforço" e "custo" em Engenharia de Software, enfatizando suas possíveis diferenças e eventuais relações. Uma compreensão mais precisa desses termos e dos conceitos associados pode contribuir para a redução de fatores de confusão na definição de modelos de estimativa de esforço e, consequentemente, torná-los mais representativos e fidedignos com eventual influência positiva em sua acurácia. Para alcançar uma maior clareza conceitual destes dois referidos termos e identificar possíveis inconsistências, a Linguística, em suas diversas subáreas, mostra-se como uma ciência que pode trazer importantes contribuições para solução do problema.

Em complemento, cabe ressaltar que o protocolo de investigação utilizado neste trabalho, tendo em vista seu caráter interdisciplinar e contemporâneo para a Engenharia de Software, pode ser de utilidade para outros estudos qualitativos que, de forma semelhante, enfrentem dificuldades relacionadas à falta de uma terminologia comum.

D.2 Referencial Teórico-Histórico

A literatura técnica a respeito de estimativa de esforço e custo em software é vasta, compreendendo um período de cerca de mais de 50 anos. Por esse motivo e considerando a necessidade de se compreender em qual contexto a terminologia da área surgiu e evoluiu, esta seção apresenta o referencial teórico de um ponto de vista histórico. Tal abordagem serve, ainda, para identificar possíveis motivos para a atual confusão conceitual entre os termos "esforço" e "custo".

Segundo Jones (JONES, 2007), os pioneiros da área de estimativa de custo de software foram Barry Boehm e Larry Putnam, tendo iniciado seus trabalhos no final da década de 1960. Entretanto, os primeiros trabalhos nessa área datam do início da década de 1960, de autoria de membros da *System Development Corporation* (SDC), uma divisão da *RAND Corporation*, que desenvolvia projetos para órgãos vinculados ao Departamento de Defesa dos Estados Unidos (HEINZE; CLAUSSEN; LABOLLE, 1963) (FARR; NANUS, 1964) (FARR; ZAGORSKI, 1964). Esses trabalhos faziam parte de um projeto patrocinado pela *Air Force Electronic Systems Division* cujo objetivo era identificar mecanismos para reduzir e estimar os custos de produção dos programas de computador daquela divisão militar (HEINZE; CLAUSSEN; LABOLLE, 1963).

O trabalho pioneiro de Heinze, Claussen e LaBolle (HEINZE; CLAUSSEN; LABOLLE, 1963) fez um levantamento dos fatores que afetam o custo de gerenciamento de projetos de desenvolvimento de software, com base em dados de sete projetos. Tais fatores foram divididos em três grupos: conteúdo técnico, ambiente e recursos. Este trabalho usou, pela primeira vez, o termo "*man-month*" (homem-mês) para o contexto de software, denotando a ideia de quantidade de trabalho humano realizado por unidade de tempo, de forma semelhante ao que era usado àquela época na manufatura. No ano seguinte, os trabalhos de Farr e Nanus (FARR; NANUS, 1964) e Farr e Zagorski (FARR; ZAGORSKI, 1964) apresentaram cerca de 50 fatores que teriam alguma influência sobre o custo do software. Tais fatores foram divididos em sete categorias, que incluíam fatores relacionados ao trabalho a ser feito ("requisitos e projeto operacionais" e "projeto e produção do programa"), aos recursos disponíveis ("equipamentos de processamento de dados" e "equipe de programação") e ao ambiente de trabalho ("processos de gerenciamento", "ambiente de desenvolvimento" e "instalações, serviços e suprimentos") (FARR; NANUS, 1964) (FARR; ZAGORSKI,

1964). É possível, portanto, perceber nas categorias apresentadas a intenção dos trabalhos da SDC: desenvolver um modelo de custo, onde a mão-de-obra empregada é apenas um dos componentes. Essa percepção é corroborada, em especial, pelas seguintes categorias apresentadas:

- "Equipamentos de processamento de dados": considera o custo relacionado ao hardware necessário para desenvolver e testar o software;
- "Ambiente de desenvolvimento": considera, entre outros fatores, os custos decorrentes das viagens realizadas pela equipe de desenvolvedores;
- "Instalações, serviços e suprimentos": considera os custos do espaço físico, serviços de reprodução e material de escritório.

Vale frisar que os trabalhos da SDC associam o termo "*man-month*" ao termo custo e não a esforço. Esforço (*effort*) é utilizado nesses trabalhos com a acepção de "projeto". Os autores justificam a utilização de "*man-month*" como unidade de medida de custo em detrimento a uma unidade monetária (dólar) pelo fato desta última sofrer variações em função de salários diferenciados, câmbio e despesas gerais (FARR; ZAGORSKI, 1964).

Nas décadas seguintes, os trabalhos de Boehm e Putnam ganharam destaque, tendo sido referenciados por inúmeros outros trabalhos. O trabalho de Putnam e Myers (PUTNAM; MYERS, 2003) apresenta equações relacionando tamanho, esforço, tempo, produtividade e qualidade. Putnam utiliza a unidade de medida "*man-month*" com o mesmo sentido empregado nos trabalhos da SDC: tempo dedicado por determinada quantidade de mão-de-obra exclusivamente ao projeto, sem levar em conta o tempo com interrupções no trabalho. Entretanto, Putnam associa o termo "*man-month*" ao termo "*effort*" e não a "*cost*", como feito nos trabalhos da SDC. Já o modelo COCOMO, proposto por Boehm (BOEHM, 1981) (BOEHM *et al.*, 2000) já indicava seu foco sobre o custo ao utilizar o termo "*cost*" no acrônimo que o nomeia - *Constructive Cost Model*. Além disso, o modelo é baseado em uma série de fatores, denominados "*cost drivers*" (fatores de custo), que são levados em conta em um conjunto de equações que incluem o esforço (medido em homens-mês), produtividade, cronograma e disponibilidade de pessoal. O custo, por sua vez, também é tratado em homens-mês tendo em vista as diferenças entre as organizações e considerando que homem-mês é uma unidade mais

estável que o dólar. O custo em homens-mês deve ser, então, multiplicado por um valor monetário definido por cada organização para cada homem-mês empregado no projeto.

Os fatores de custo do COCOMO contemplam atributos relacionados ao produto a ser desenvolvido, hardware utilizado, recursos humanos e características do projeto. Diferentemente do que é percebido nos trabalhos da SDC, o modelo COCOMO, apesar de se designar um modelo de custo, não considera os custos indiretos do desenvolvimento de software, como aqueles relacionados às instalações e ao material de escritório. Os fatores de custo contemplados pelo modelo COCOMO estão concentrados nos gastos relacionados diretamente ao desenvolvimento do software, em especial aqueles gastos que irão afetar a quantidade de homens-mês do projeto. Pode-se conjecturar de que esta característica do COCOMO tenha influenciado os modelos de estimativa subsequentes a focar apenas em fatores diretamente relacionados ao custo de desenvolvimento, especialmente o custo com mão-de-obra, em detrimento dos custos indiretos do projeto.

Essa percepção de que "esforço" e "custo" são tratados em Engenharia de Software como termos equivalentes pode ser observada nos trabalhos de Mendes (MENDES, 2008) (MENDES, 2014), quando a autora, ao definir o conceito de estimativa de esforço, cita "*... estimativa de esforço, também conhecida como estimativa de custo (ou costimation)...*" e quando a autora, ao descrever o conceito de "*fatores de custo", diz que "... são os fatores que se acredita estar associados com o esforço...*". Essa visão é ratificada pelo trabalho de Grimstad, Jørgensen e Moløkken-Østvold (GRIMSTAD; JøRGENSEN; MOLøKKEN-ØSTVOLD, 2006), que mostra que a terminologia a respeito de estimativa de esforço é raramente utilizada de forma consistente em um grande conjunto de livros-texto e artigos da área de Engenharia de Software.

D.3 Relevância e questões de pesquisa

Considerando sua natureza interdisciplinar, este trabalho buscou apoiar-se na Linguística, a ciência que se ocupa dos estudos da linguagem. Fuzer e Cabral (FUZER; CABRAL, 2014) definem a linguagem como "um recurso para fazer e trocar significados" ou "a instanciação de um potencial amplo de significados que pode (...) construir experiências". A ponte com a literatura de ES torna-se evidente uma vez que o

fenômeno que se busca elucidar neste trabalho é o uso inconsistente dos vocábulos "custo" e "esforço". Estes dois termos são palavras e, como tais, parte constituinte da linguagem. É válido, então, dizer que, como expressões de linguagem, estes termos são usados para fornecer informações, de acordo com a definição de Fuzer e Cabral (FUZER; CABRAL, 2014). É importante a busca por perceber se tais termos evocam sempre as mesmas noções, ou seja, fornecem sempre as mesmas informações no contexto em que são usados (textos técnicos de Engenharia de Software), podendo, portanto serem usados de forma intercambiável ou se essa literatura ora usa-os em relação de sinonímia e ora como fornecedores de informações semelhantes, mas não idênticas a fim de promover maior nível de exatidão a modelos de estimativa.

Em suma, algumas questões são pertinentes, uma vez que a relação entre "esforço" e "custo" nem sempre é clara. São estes sempre sinônimos, podendo ser trocados sem prejuízo de significado, ou a relação construída entre eles é de hiponímia? Entende-se por hiponímia a relação entre itens lexicais específicos e gerais, de forma que o primeiro esteja "incluído" no segundo (CRYSTAL, 2008). Isto é, no plano conceitual, "esforço" estando incluído em "custo", aquele sendo parte deste mas não sendo exatamente a mesma noção. Uma vez que os autores na literatura em questão não fornecem sempre definições prévias do que entendem por "esforço" e o que entendem por "custo", busca-se identificar na construção textual indícios da relação entre estes lançando mão de pressupostos linguísticos. Neste cenário, pode-se estabelecer as seguintes questões de pesquisa para este estudo:

- *QE3: Qual a relação linguística entre os termos "esforço" e "custo" em Engenharia de Software?*
- *QE4: Como a Linguística pode apoiar na resolução de confusões terminológicas em Engenharia de Software?*

A relevância de tal relação interdisciplinar entre a Engenharia de Software e a Linguística está em identificar inconsistências conceituais (ora os termos evocam noções semelhantes, ora não) que podem comprometer os modelos de estimativa de esforço que apresentam tais inconsistências. Para isto, recortes de análise de diferentes áreas da Linguística provam sua valia. Uma vez desenvolvida a argumentação no tocante ao auxílio de pressupostos linguísticos, a metodologia de análise é relatada. O primeiro recorte que é apresentado neste trabalho é de natureza etimológica.

D.4 Análise Etimológica

A etimologia é definida como o estudo da origem e história da forma e da significação das palavras (CRYSTAL, 2008). Etimologicamente, as palavras "esforço" e "custo" provêm da mesma origem: o Latim Vulgar (THE ONLINE ETYMOLOGY DICTIONARY, 2001) (CUNHA, 2013). O Latim Vulgar era a língua falada pelo povo em geral (a *plebe ignara*) no Império Romano, de maneira espontânea, coloquial e sem preocupação com normas gramaticais (BOTELHO, 2010). Idiomas românicos como o Português, o Espanhol e o Francês foram originados a partir do Latim Vulgar e, por tal proximidade linguística, a análise etimológica das palavras "esforço" e "custo" foi conduzida com base em um dicionário etimológico do Português (CUNHA, 2013), complementado por um dicionário etimológico do Inglês (THE ONLINE ETYMOLOGY DICTIONARY, 2001).

A palavra "esforço" é derivada de **exfortiare*[7] (THE ONLINE ETYMOLOGY DICTIONARY, 2001) (CUNHA, 2013). O prefixo *ex*, presente em palavras como "êxodo" e "exorcismo", significa "para fora". O radical *forti*, é uma derivação da palavra *fortis*, que remete ao significado de "força". Já o sufixo *are* era usado para designar verbos que indicavam ação. Desta forma, o termo **exfortiare* foi provavelmente usado com o significado de "ação de dispender força" para realizar algum trabalho.

"Custo", por sua vez, vem da palavra **costare* (CUNHA, 2013) (THE ONLINE ETYMOLOGY DICTIONARY, 2001). O prefixo *cos* é derivado da palavra latina *com*, que significa "em". O radical *stare*, derivado do vocábulo **sta*, significa "situar" ou "posicionar" e provém do tronco linguístico protoindo-europeu (PIE) (THE ONLINE ETYMOLOGY DICTIONARY, 2001), o ancestral comum entre os idiomas falados na Europa. Portanto, a acepção original de **costare* era "estar situado em". Entretanto, no século XIII, sua acepção foi estendida, passando a significar "preço" ou "valor" a ser pago (THE ONLINE ETYMOLOGY DICTIONARY, 2001).

Já o termo "homem-hora", e todas as suas derivações (homem-dia, homem-mês, homem-ano, pessoa-hora, pessoa-mês, dentre outras), não possui registros etimológicos precisos. O dicionário Merriam-Webster (MERRIAM-WEBSTER, 2015), por exemplo, indica que o primeiro registro desse termo data de 1912, enquanto que o Dictionary.com

[7] O asterisco usado antes da palavra serve para indicar que esta não se encontra documentada, sendo, portanto, hipotética (CUNHA, 2013).

(DICTIONARY.COM, 2015) informa que isto ocorreu entre 1915 e 1920. No entanto, é possível encontrar trechos de pronunciamentos proferidos no Parlamento Britânico no final do século XIX citando o termo homem-hora. Nesses registros é possível observar que o termo é empregado com a noção de quantidade de trabalho realizado por uma pessoa durante uma hora, como era comum no contexto da Revolução Industrial.

D.5 Análise de *Corpus*

Para compreender quais significados são atribuídos aos termos e, consequentemente, esboçar alguma teoria, torna-se necessário avaliar grandes volumes de informação, dada a quantidade de variáveis de contexto envolvidas. Para tal, é imprescindível utilizar ferramentas computacionais que possam facilitar o trabalho de organização e interpretação das evidências linguísticas. Neste cenário surge a Linguística de *Corpus*, uma área da Linguística que se apoia em ferramentas computacionais para coleta e análise de *corpus* (TAGNIN, 2013). Trata-se de um método quantitativo e probabilístico para apoiar estudos naturalmente qualitativos. Um *corpus*, por sua vez, é uma coletânea de registros textuais ou falados, necessariamente em formato eletrônico, compilados e organizados criteriosamente para serem objeto de pesquisa linguística (TAGNIN, 2013).

Para realizar análises sobre *corpora* (o plural de *corpus*) é necessário utilizar ferramentas específicas para tal. Ferramentas para análise de *corpus* geralmente trazem uma série de recursos para apoiar análises linguísticas. A ferramenta utilizada neste trabalho, o *Sketch Engine* (LEXICAL COMPUTING, 2015), possui uma série de recursos, sendo os seguintes os mais importantes no contexto deste trabalho:

- Concordanciador: capaz de exibir concordâncias do termo pesquisado, isto é, os fragmentos de texto nos quais o termo investigado está inserido (TAGNIN, 2013). Cada concordância é apresentada de acordo com o padrão *Key Word in Context* (KWIC), onde o termo pesquisado é apresentado de forma destacada dentro do fragmento ou contexto (LEXICAL COMPUTING, 2015).
- *Thesaurus*: utilizado para identificar termos similares ao termo pesquisado, exibindo-os em um *ranking* de frequência (LEXICAL COMPUTING, 2015). Este recurso pode ser útil para identificar relações de sinonímia entre palavras.

- *Word Sketch*: permite comparar termos em função de relações lexicográficas que possuem com outros termos, o que pode ser útil para estudos voltados à sintaxe.

Tais recursos foram utilizados nas análises de *corpora* feitas neste trabalho. As seções D.5.1 e D.5.2 apresentam, respectivamente, uma análise dos termos "*effort*" e "*cost*" em um *corpus* de propósito geral e em um *corpus* preparado para retratar a linguagem utilizada na área de estimativa de esforço e custo de software.

D.5.1 Análise da frequência e relação entre termos no British National Corpus

Como já defendido na seção 0, as noções que as palavras evocam são construídas na sua relação com o todo, o contexto em que estão inseridas. É arriscado realizar análises numéricas que se encerram apenas em si mesmas. Isso não significa dizer que a frequência com que determinadas palavras são usadas de maneira relacionada não é relevante. Estes dados numéricos não devem ser conclusivos em si só, mas apontam caminhos de análise.

Para lidar com a construção de uma relação entre palavras em termos estatísticos, recorreu-se ao *British National Corpus* (*Corpus* Nacional Britânico), um banco de ocorrências de palavras na Língua Inglesa, composto por mais de 100 milhões de palavras, acessado via *Sketch Engine*. Ao se inserir o termo "*effort*" no Thesaurus deste *corpus*, as duas palavras que mais se aproximam em frequência e relação são "*attempt*" (tentativa) e "*initiative*" (iniciativa), enquanto que as palavras que apresentam maior proximidade com "*cost*" são "*price*" (preço) e "*value*" (valor). Esta estatística, como dito anteriormente, não basta. No entanto, estes dados apontam para a realidade de que nem sempre "esforço" e "custo" são necessariamente intercambiáveis sem prejuízo de significado, uma vez que o termo "*cost*" não aparece na lista de termos similares ao termo "*effort*" e vice-versa. Ainda, de maneira geral, os termos semelhantes a cada uma das duas palavras também não são os mesmos, como evidenciado na Figura 51 e na Figura 52.

Figura 51: Termos similares ao termo "*effort*" no *British National Corpus*.

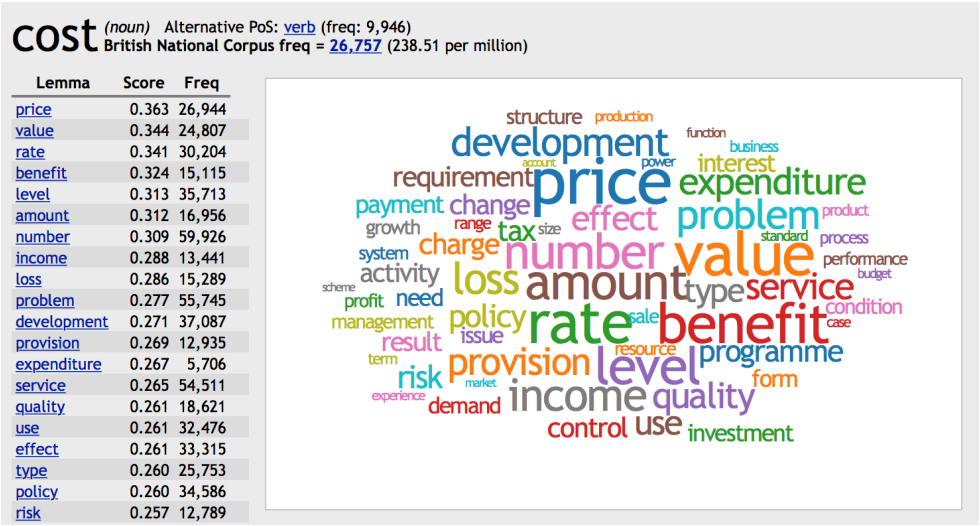


Figura 52: Termos similares ao termo "*Cost*" no *British National Corpus*.

D.5.2 Análise sobre o Corpus de Estimativa de Esforço e Custo de Software

A análise da frequência e relação entre palavras feita na seção D.5.1 foi de natureza numérica. Esta, apesar de instigante, é inconclusiva em si mesma. Na tentativa de trazer o contexto para a análise de *corpus* e superar a limitação de uma análise desprovida de uma esfera contextual, um *corpus* específico foi criado com auxílio da ferramenta *Sketch Engine* (LEXICAL COMPUTING, 2015). Esse *corpus* foi nomeado como *Effort and Cost Estimation Corpus* (ECEC) e todas as ocorrências nele compiladas são oriundas de trabalhos da área de Engenharia de Software.

O ECEC foi carregado com os três relatórios da SDC citados na seção D.2 (HEINZE; CLAUSSEN; LABOLLE, 1963) (FARR; NANUS, 1964) (FARR; ZAGORSKI, 1964), bem como, com artigos indexados pela máquina de busca Scopus. A escolha da Scopus foi motivada pela ampla cobertura oferecida em trabalhos relacionados à Computação. Desta forma, foi submetida a *string* de busca "*TITLE-ABS-KEY("software effort estimation" OR ("software cost estimation" OR ("software effort prediction" OR "("software cost estimation")"* na base da Scopus, retornando 769 ocorrências. O processo de inclusão dos artigos deu-se pela leitura do título e resumo das ocorrências, resultando em 731 ocorrências pré-selecionadas. Em seguida, considerando um nível de confiança de 95% e um intervalo de confiança de 8%, a amostra utilizada para carregar o ECEC contemplou 126 artigos escolhidos aleatoriamente dentre os 731 pré-selecionados.

Em relação a seu tamanho, o ECEC possui pouco mais de um milhão de palavras, distribuídas em 45.300 sentenças. Quanto à sua tipologia, o ECEC pode ser classificado de acordo com os seguintes critérios (SARDINHA, 2004):

- Modo: *escrito*, uma vez que o *corpus* é formado por livros e artigos técnicos.
- Tempo: *sincrônico*, porque representa um período específico de tempo.
- Seleção: *de amostragem* (*sample corpus*), já que o *corpus* é constituído por uma amostra finita da linguagem.
- Conteúdo: *especializado*, considerando que os textos são de uma área de conhecimento específica.
- Autoria: *de aprendiz*, tendo em vista que os autores deste *corpus* não são falantes nativos do idioma dos textos que o compõem.
- Finalidade: *de estudo*, já que pretende-se descrever o conteúdo deste *corpus*.

Neste *corpus* específico, nota-se que – conforme sugerido pela Figura 53 e pela Figura 54 – "*cost*" é a palavra que mais se relaciona como similar a "*effort*" e vice-versa. O *corpus* nos sugere que existe forte aproximação no uso destas duas palavras no campo da Engenharia de Software.

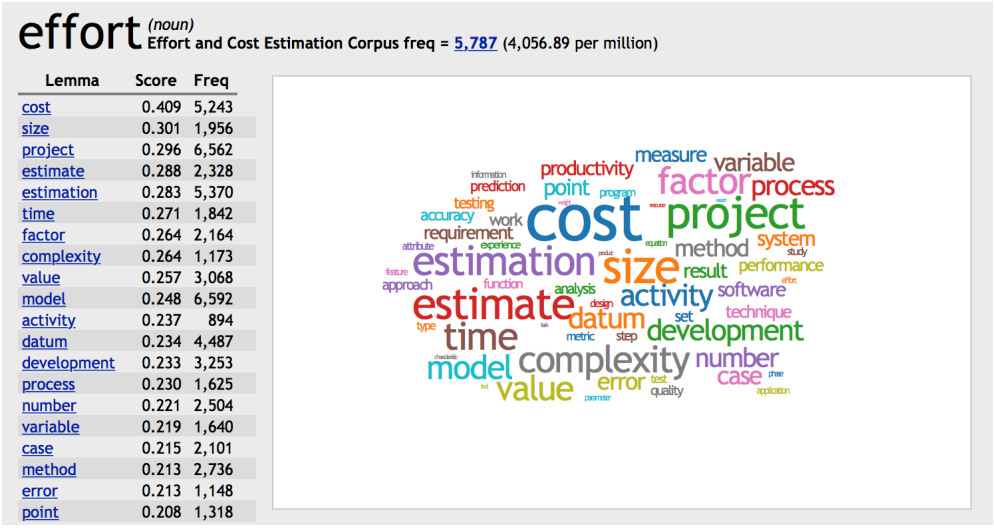


Figura 53: Termos similares ao termo "*Effort*" no *Effort and Cost Estimation Corpus*.

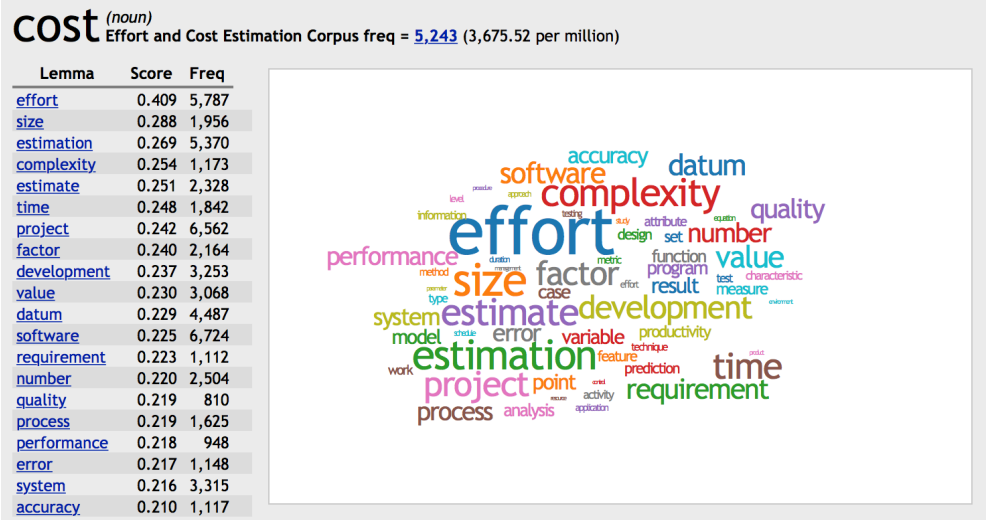


Figura 54: Termos similares ao termo "*Cost*" no *Effort and Cost Estimation Corpus*.

Aqui cabe trazer a pergunta: essa forte relação entre as duas palavras na definida área do conhecimento é construída apenas por aproximação, sem que as duas signifiquem o mesmo? Em suma, qual é a natureza da relação entre estes dois termos na engenharia de software: **sinonímia** (eles abarcam as mesmas noções) ou **hiponímia** (no plano conceitual, um está "contido" no outro, assim como "gato" está contido em "animal")? No intuito de coletar evidências que apontem para respostas a essas perguntas, foram utilizadas quatro concordâncias (fragmentos) do ECEC para análise.

- Concordância 1: "*... proposed a model for **effort and cost estimation** in agile software development by using regression analysis...*"

Na concordância 1, os termos que são foco desta análise não estão mais isolados. Antes, eles estão situados em um contexto de construção de significado e isto possibilita uma análise mais bem estruturada. Recorreu-se, portanto, a um recorte sintático para a análise da expressão textual apresentada.

A sintaxe pode ser definida como o "termo tradicional para o estudo das regras que regem a maneira como as palavras se combinam para formar as sentenças de uma língua" (CRYSTAL, 2008). Existe um conceito na Sintaxe, o conceito de coordenação, que pode sugerir a forma como os dois termos são tratados aqui. Por coordenação entende-se "o processo de combinação de dois constituintes de mesma natureza para a produção de um terceiro elemento, maior e do mesmo tipo" (CELCE-MURCIA; LARSEN-FREEMAN, 1999). É justamente um processo de coordenação que une "*effort*" e "*cost*". Estes dois elementos estão exercendo a função de modificar, fornecer informação sobre o termo "*estimation*" (estimativa). Ou seja, o texto não fala de qualquer estimativa, mas somente de uma estimativa de custo e de esforço. Isto equivale a afirmar que "*effort*" e "*cost*" têm seu eixo de rotação e sua função de existência centrados no termo "*estimation*". Os termos "*effort*" e "*cost*" não são dependentes um do outro para existir. Ambos são subordinados ao termo "*estimation*" e independentes entre si. Os significados dos dois termos coordenados são distintos um do outro, ou seja, a forma como os elementos aqui referidos são justapostos sintaticamente sugere que estes têm significados diferentes. A conjunção que os coordena ("*AND*") reforça tal argumentação, uma vez que adiciona um conceito a outro. Pelo menos nesta concordância e em mais dez outras presentes no ECEC, **custo** e **esforço**, embora relacionados, são tratados como conceitos distintos.

- Concordância 2: "*Software **effort estimation** (SEE) is the prediction about the amount of **effort** required to make a software system and its duration [1]. SEE first appeared in 1950s (...) having the objective of developing useful models that constructively explain the development life-cycle and accurately predict the **cost** of developing a software product... Since then, there were developed a lot of models for the **effort and cost estimation***."

Na concordância 2, os dois termos são usados separadamente. Tal passagem é iniciada com uma definição de estimativa de esforço, sem que o conceito de esforço seja claramente explicitado. Posteriormente, tem-se a ocorrência de custo. A questão central

aqui é: no fragmento citado, nenhum dos dois termos é definido. Não existe a expressão do que se entende por custo e esforço. Pode-se interpretar custo, por exemplo, como "soma de recursos financeiros que é necessária para se desenvolver um produto de software", ainda sem muita certeza. Já esforço pode ser interpretado como "desprendimento de energia". Esta energia, por sua vez, poderia ser definida como tempo, recursos tecnológicos ou ambos. Nada está claro. Finalmente, "custo" e "esforço" são novamente coordenados, como na concordância 1, sugerindo que estes evocam noções distintas, embora relacionadas.

- Concordância 3: "*The financial success of a project can depend on the ability of the software manager to estimate the **cost** of software development accurately, prior to the start of the project. **Software cost and effort estimation** will never be an exact science. Too many variables - human, technical, environmental, political - can affect the ultimate **cost** of software and the **effort** needed to develop it.*"

No fragmento 3, alguns fatores que afetam o custo e o esforço do desenvolvimento de um produto de software são elencados. Uma questão se torna pertinente, uma vez que estes termos aparecem coordenados, sugerindo que abarcam conceitos relacionados, porém, distintos: os fatores elencados afetam igualmente "custo" e "esforço"? Ou alguns fatores afetam apenas o custo e outros somente o esforço?

- Concordância 4: "*The main criterion for including a journal paper in our review is that the paper describes research on software development **effort or cost estimation**. Papers related to estimation of software size, assessment of software complexity, or identification of factors correlated with software development effort, are only included if the main purpose of the studies is to improve software development **effort or cost estimation***."

Na quarta concordância, ambos os termos aparecem coordenados modificando e fornecendo informações a respeito do termo "*estimation*". Todavia, o elemento que os coordena – ou seja, que os põe lado a lado sem um exercer função sobre o outro mas ambos exercendo função de especificar "*estimation*" – não é mais "*AND*" e sim "*OR*".

O valor semântico deste último elemento deixa margem para a interpretação de custo e esforço como sendo sinônimos. Ou seja, a literatura poderia se valer de ambos em relação de sinonímia. Os dois termos aqui não são coordenados como evocando conceitos distintos, como em concordâncias anteriores, mas sim como evocadores dos mesmos conceitos. Este tipo de relação é observado em outras seis concordâncias.

Estes fragmentos são parte de uma amostra maior que evidencia a falta de consenso a respeito do tipo de relação entre estes dois vocábulos na Engenharia de Software. A linguagem coordena estes elementos dando forma linguística e observável àquilo que está no plano dos conceitos, no plano semântico. É, então, através da análise da forma que se evidencia a falta de clareza conceitual.

D.5.3 Frequência de n-gramas do termo "man-hour"

Adicionalmente ao estudo sobre os termos "esforço" e "custo", foi realizada outra análise de natureza linguística, a respeito da frequência de uso do termo "*man-hour*", a unidade de medida padrão para representar esforço em projetos de software. Para isso, foi utilizada a ferramenta *Google Ngram Viewer* (GOOGLE, 2016), uma máquina de busca que gera gráficos de frequência de termos contidos no *corpus* do *Google Books* através da contagem de *n*-gramas. Um *n*-grama é uma sequência contígua de *n* itens de uma dada sequência de texto ou da fala (GOOGLE, 2016). Os itens podem ser letras, fonemas, sílabas, palavras simples ou compostas. Um *n*-grama de tamanho 1 é denominado *unigrama*, de tamanho 2 é um *bigrama* e de tamanho 3 é um *trigrama* (GOOGLE, 2016).

O gráfico apresentado na Figura 55 representa a frequência do termo "*man-hour*" (e equivalentes como "*man-month*", "*person-hour*", etc.) no *corpus* do *Google Books* através do *Ngram Viewer*. O gráfico mostra que o uso deste termo aumenta e atinge seu ápice entre 1920 e 1960, período marcado pela instalação de grandes parques industriais na Europa e nos Estados Unidos e pela produção em massa de bens de consumo. Por outro lado, a frequência diminui à medida que caminha em direção ao século 21, época em que se verifica um aprofundamento do processo de transformação da economia, potencializado sobretudo pela evolução das tecnologias de comunicação e informação. Este comportamento pode sugerir que o termo "*man-hour*" é inapropriado

para representar o tipo de atividade laboral exercida atualmente, de viés intelectual, em vez de atividades majoritariamente físicas, típicas da indústria manufatureira.

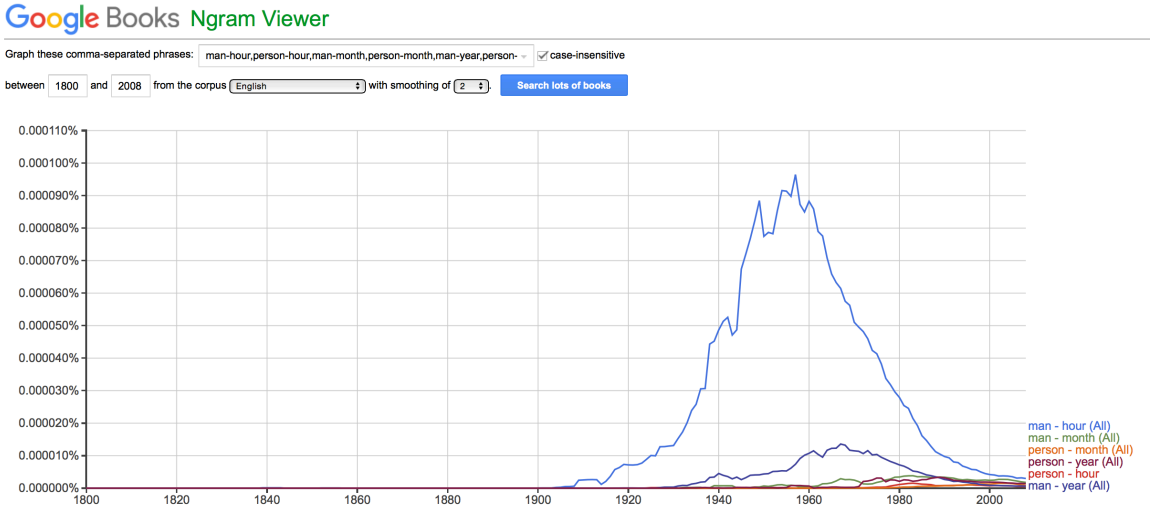


Figura 55: Frequência do termo "*man-hour*" e equivalentes no *corpus* do Google Books.

O percentual indicado no gráfico representa a frequência de cada termo pesquisado proporcionalmente no conjunto de termos de mesmo *n*-grama. Por exemplo, quando um termo unigrama é pesquisado, a frequência apresentada é dada em relação a todos os termos unigrama existentes no *corpus*. Desta forma, quando a pesquisa envolve um termo unigrama e outro bigrama, os resultados não são comparáveis, já que os percentuais apresentados são baseados em conjuntos distintos. É importante observar que todos os termos pesquisados apresentados na Figura 55 são bigramas, o que torna coerente a comparação entre suas frequências.

D.6 Resultados e Discussão

Como mencionado na seção D.1, o estudo secundário que motivou este trabalho, descrito no capítulo 3, fez uso de definições de dicionário para definir o conceito de esforço em software e, consequentemente, identificar os fatores que o afetam (SILVA-DE-SOUZA; RIBEIRO; TRAVASSOS, 2014). No entanto, é importante ressaltar que a presença de diversas entradas no dicionário, por si só, evidencia que as palavras podem evocar múltiplos significados. É arriscado selecionar qualquer uma das definições sem contemplar o contexto em que estas palavras ocorrem, pois existe uma dependência de sentido evocado entre a palavra e o contexto em que é utilizada.

Quanto à questão de pesquisa QE3, do ponto de vista linguístico, pode-se observar que não há consistência no uso dos termos "esforço" e "custo" no contexto da Engenharia de Software. Ora esses termos apresentam relação de sinonímia, ora de hiponímia e, por vezes, nenhuma dependência. Considerando uma perspectiva quantitativa, para que tais termos possam ser considerados equivalentes eles deveriam variar juntos, na mesma direção e na mesma intensidade, o que não é percebido na literatura técnica da área.

Entende-se neste trabalho por "esforço" a ação de dispender energia para realizar uma atividade física ou mental necessária para alcançar algum objetivo. Tal conceito, no entanto, é por vezes tratado como "custo" em Engenharia de Software. A área de Economia, por sua vez, apresenta o conceito de custo de forma consistente, como sendo o gasto financeiro relativo à aplicação de recursos (humanos, equipamentos, insumos) no processo de criação de um produto ou serviço (COGAN, 2013) (RIBEIRO, 2015). Os custos ainda podem ser classificados em fixos ou variáveis, conforme a variação da quantidade de trabalho, o que é incomum observar nos trabalhos de Engenharia de Software.

Pode-se observar também que, respondendo à questão QE4, a linguística de *corpus* possibilita ao pesquisador apoiar-se em traços linguísticos reais e em grande quantidade para tentar realizar generalizações conceituais ou esboçar teorias em áreas carentes de uma terminologia consistente, como a Engenharia de Software.

As principais ameaças à validade deste estudo estão relacionadas ao uso de *corpora* de diferentes contextos para realizar análises semânticas e ao fato de analisar fragmentos específicos de um *corpus* para estabelecer percepções gerais. Quanto à primeira ameaça, trata-se do risco assumido quando se quer avaliar se a linguagem utilizada em uma área específica está de acordo com as acepções gerais. E quanto à segunda ameaça, é importante destacar que os fragmentos analisados apresentam fenômenos linguísticos recorrentes em diversas concordâncias do mesmo *corpus*.

D.7 Considerações Finais

Este capítulo descreveu um trabalho que aplicou princípios da Linguística em busca de uma definição consistente dos termos "esforço" e "custo" na área de Engenharia de Software. No entanto, os resultados indicam que os referidos termos são

utilizados de forma inconsistente na literatura técnica da área, apresentando relações de dependência em alguns trabalhos e sendo totalmente distintos em outros, o que pode representar uma ameaça à validade dos modelos de estimativa de esforço e custo propostos em ES.

O conceito de esforço em ES deve levar em conta que desenvolver software é uma tarefa que envolve técnica, criatividade e raciocínio lógico e não pode ser confundida com tarefas previsíveis e repetitivas típicas da manufatura, usualmente medidas em homem-hora. Neste sentido, quando modelos são intitulados "de estimativa de custo" passam a impressão de que todos os custos do projeto estão sendo considerados. No entanto, ao representar custo como "homem-hora", tais modelos de estimativa se tornam incoerentes, pois passam a representar valor financeiro como eventual esforço humano, medido em função do tempo de alocação de pessoas em atividades criativas.

Neste cenário, entende-se que o conceito de custo tradicionalmente utilizado em Economia mostra-se adequado para ser utilizado na Engenharia de Software. Por sua vez, esforço deve ser observado através da energia despendida para a realização de atividades de criação nos projetos de software. A confusão conceitual entre fatores de esforço e custo representa uma ameaça à validade dos modelos de estimativa disponíveis em Engenharia de Software e pode contribuir para reduzir sua acurácia e aumentar os riscos na tomada de decisão nos projetos de software.

APÊNDICE E - Termo de Consentimento Livre e Esclarecido

TCLE	**Termo de Consentimento Livre e Esclarecido**

Eu declaro ter mais de 18 anos de idade e concordo em participar de um estudo não-invasivo e impessoal conduzido pelo pesquisador Thiago Silva de Souza, sob a orientação do Prof. Dr. Guilherme Horta Travassos. O estudo faz parte de pesquisas relacionadas à tese de doutorado do referido pesquisador, realizada no âmbito do Programa de Engenharia de Sistemas e Computação (PESC) da Universidade Federal do Rio de Janeiro (COPPE/UFRJ).

Objetivo

O objetivo do estudo é melhorar a compreensão sobre os fatores que influenciam o esforço nas diferentes atividades relacionadas a testes de software, especialmente no contexto de Web Services RESTful.

Procedimento

Os participantes deverão responder, individualmente, a um questionário relacionado à sua última experiência com testes de Web Services RESTful. Em seguida, os participantes serão entrevistados para expor as motivações que resultaram em suas respostas.

Benefícios e Liberdade de Desistência

Eu entendo que o estudo não tem a finalidade de avaliação pessoal do participante. Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de tecnologias para a Engenharia de Software.

Confidencialidade

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado nas publicações decorrentes deste estudo. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do estudo.

Participante da Pesquisa

Nome (em letra de forma): _____

_____, _____ de _____ de 201___.

_____

Assinatura

APÊNDICE F - Questionário de Caracterização do Participante

QCP	**Questionário de Caracterização do Participante**

Nome Completo
Origem
☐ Aluno de Pós-Graduação ☐ Profissional da indústria
Formação Universitária
Nível: ☐ Doutor ☐ Mestre ☐ Especialista ☐ Graduado
Conhecimento de Idiomas
Por favor, estime sua habilidade em utilizar material de trabalho em Inglês: ☐ Eu falo, leio e escrevo fluentemente. ☐ Considero o Inglês como sendo um idioma no qual (por favor, complete) Minhas habilidades de leitura e compreensão de textos: ☐ poderiam ser melhores ☐ são moderadas ☐ são altas ☐ são muito altas Minha capacidade de trabalhar/seguir instruções escritas em Inglês: ☐ poderiam ser melhores ☐ são moderadas ☐ são altas ☐ são muito altas
Experiência em Desenvolvimento de Software
Qual é sua experiência anterior com desenvolvimento de software na prática ? (marque aqueles itens que melhor se aplicam) ☐ nunca desenvolvi software. ☐ tenho desenvolvido software para uso próprio. ☐ tenho desenvolvido software como parte de uma equipe, relacionado a um curso. ☐ tenho desenvolvido software como parte de uma equipe, na indústria. Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em desenvolvimento (E.g. “Eu trabalho há 10 anos como analista de testes na indústria...”)

Por favor, indique sua experiência nas questões a seguir, assinalando uma das alternativas da escala de 1 a 5, de acordo com a seguinte orientação:

1 – nenhuma
2 – estudei em cursos ou livros
3 – pratiquei em projetos de curso
4 – utilizei em um projeto na indústria
5 – utilizei em vários projetos na indústria

No contexto de qualquer tecnologia, exceto as de Web Services RESTful e de aplicativos *mobile*:

a) Especificação de requisitos de software	1 2 3 4 5
b) Revisão/inspeção de requisitos	1 2 3 4 5
c) Ferramentas de controle de versão	1 2 3 4 5
d) Desenvolvimento de sistemas Web	1 2 3 4 5
e) Planejamento de testes	1 2 3 4 5
f) Design & implementação de testes a partir de especificação	1 2 3 4 5
g) Execução de testes funcionais (em qualquer nível de teste)	1 2 3 4 5
h) Execução de testes de desempenho (em qualquer nível de teste)	1 2 3 4 5
i) Utilização de *frameworks* de testes de unidade (e.g. JUnit)	1 2 3 4 5
j) Utilização de *frameworks* de testes de GUI (e.g. Selenium)	1 2 3 4 5
k) Utilização de ferramentas de integração contínua (e.g. Jenkins)	1 2 3 4 5
l) Utilização de ferramentas de testes de Web Services (e.g. SoapUI)	1 2 3 4 5
m) Utilização de ferramentas de testes de desempenho (e.g. JMeter)	1 2 3 4 5
n) Comunicação de incidentes de teste	1 2 3 4 5

No contexto de Web Services RESTful:

o) Desenvolvimento de Web Services RESTful	1 2 3 4 5
p) Design & implementação de testes de Web Services RESTful	1 2 3 4 5
q) Execução de testes de Web Services RESTful	1 2 3 4 5

No contexto de aplicativos *mobile*:

r) Desenvolvimento de aplicativos *mobile*	1 2 3 4 5
s) Design & implementação de testes de aplicativos *mobile*	1 2 3 4 5
t) Execução de testes de aplicativos *mobile*	1 2 3 4 5

APÊNDICE G - Questionário de Percepção do Esforço de Teste

QPET	**Questionário de Percepção do Esforço de Teste**

Este instrumento tem a finalidade de apoiar a coleta de dados a respeito da sua percepção de esforço nos **projetos de teste de software** que você tenha participado. Desses projetos, responda às questões a seguir levando em conta somente aqueles que foram realizados sob a mesma configuração de estratégia de teste (mesmos nível, tipo, técnica e forma de execução de testes). Desconsidere suas impressões obtidas em projetos que possuíam outras configurações de estratégia de teste.

Inicialmente, você deve ler e compreender os conceitos apresentados em seguida para, então, preencher o questionário. Você deve preencher um questionário para cada configuração de estratégia de teste realizada pela sua equipe e que você tenha participado.

Tomemos como base as atividades de um processo típico de teste de software, baseado na norma IEEE 29119-2. Esse processo padrão é composto pelas seguintes atividades:

- Planejar Teste
- Configurar Ambiente de Testes
- Projetar e Implementar Testes
- Executar Testes
- Comunicar Incidentes de Teste
- Finalizar Projeto de Teste

A atividade "Planejar Teste" tem como objetivo principal desenvolver o *Plano de Teste*, artefato que direciona a realização do projeto de teste. O Plano de Teste deve conter, entre outros elementos, a estratégia de testes do projeto.

A atividade "Configurar Ambiente de Teste" é realizada com o objetivo de se estabelecer um ambiente isolado para a realização de testes, incluindo a instalação e configuração de servidores, a carga de bancos de dados e a configuração de ferramentas de suporte.

A atividade "Projetar e Implementar Testes" é realizada com o objetivo de se derivar e especificar casos e procedimentos de teste a partir das especificações de requisitos fornecidas pela equipe de desenvolvimento. Os procedimentos de teste especificados podem, eventualmente, ser utilizados como base para implementação de *scripts* de testes automatizados.

Na atividade "Executar Testes", os testes são executados e os seus resultados são comparados e registrados para posterior avaliação. Trata-se da atividade que possibilita a identificação de falhas no software sob teste.

A finalidade da atividade "Comunicar Incidentes de Teste" é relatar às partes interessadas os incidentes observados durante a execução de testes. Cada incidente deve ser devidamente registrado por meio de um mecanismo de *bugtracking*.

A atividade "Finalizar Projeto de Teste" contempla a limpeza do ambiente de teste e o registro formal de encerramento do projeto, através de um Relatório de Conclusão do Projeto de Teste.

Outros dois conceitos importantes para responder ao questionário são "esforço" e "fator de esforço". **Esforço** é a atividade física ou mental necessária para alcançar algum objetivo. Já um **fator de esforço** representa qualquer característica do ambiente de software que exerça um impacto significativo sobre o esforço necessário para executar alguma tarefa relacionada ao

desenvolvimento de software. Uma lista de fatores de esforço relacionados a teste de software é apresentada a seguir.

Tabela 97: Fatores de esforço relacionados a teste de software.

Fator de esforço	Descrição
Diversidade de atores	A medida da variedade de tipos de atores distintos que interagem com o SUT (*system under testing*), sejam eles humanos ou interfaces com outros sistemas.
Quantidade e duração das interrupções	A quantidade e duração das interrupções do trabalho em cada projeto, motivadas por tarefas relacionadas a outros projetos, pausas para refeições, descanso e férias, assim como para aguardar a disponibilização de uma informação ou artefato por parte da equipe de desenvolvimento.
Disponibilidade de recursos de infraestrutura	A medida da disponibilidade de recursos de infraestrutura para realização de testes e substitutos desses recursos para as várias fases de testes.
Disponibilidade de artefatos de teste reusáveis	A medida da disponibilidade artefatos de teste (e.g. casos de teste, dados de teste, etc.) reusáveis requeridos para realização do projeto de teste.
Complexidade do código-fonte	O nível de dificuldade para compreender o código-fonte, influenciado pela complexidade ciclomática, tempo de execução, profundidade de herança ou da árvore de chamadas.
Tamanho do código-fonte	A medida da quantidade de código-fonte usado para desenvolver o sistema.
Complexidade do ambiente de teste	O nível de dificuldade para configurar o ambiente de teste, influenciado pelo número de diferentes plataformas, quantidade de ambientes de teste distintos e necessidade de simuladores.
Qualidade da documentação	O grau em que a documentação do SUT contempla coerente e consistentemente os seus requisitos.
Experiência com teste	Conhecimento prático em teste de software acumulado ao longo do tempo pelos membros da equipe de teste.
Experiência com ferramentas de teste	Conhecimento prático sobre as ferramentas de teste de software utilizadas no projeto acumulado ao longo do tempo pelos membros da equipe de teste.
Experiência com o domínio de aplicação	Conhecimento prático sobre o negócio do cliente e as características do sistema acumulado ao longo do tempo pelos membros da equipe de teste.
Experiência com o ambiente operacional	Conhecimento prático sobre o ambiente operacional do software sob teste acumulado ao longo do tempo pelos membros da equipe de teste.
Experiência com a tecnologia de programação	Conhecimento prático na tecnologia de programação empregada no projeto acumulado ao longo do tempo pelos membros da equipe de teste.
Nível de criticidade	A medida da importância de uma funcionalidade ou pedaço de código específico no contexto do sistema.
Nível de segurança física pessoal	O nível de perigo ou risco oferecido pelo SUT para a equipe ou para outras partes interessadas.
Número de casos de teste	A quantidade de casos de teste necessários para atingir os critérios de qualidade do projeto.
Número de Web Services	A medida da quantidade de serviços a serem testados no projeto.
Nível de confidencialidade do projeto	A confidencialidade exigida das informações sobre o projeto.
Distribuição física da equipe do projeto	A medida em que a equipe do projeto é distribuída geográfica e organizacionalmente.
Quantidade de incidentes de teste	O número de incidentes identificados durante a execução dos testes.
Legibilidade do código	A medida da facilidade e clareza com que um leitor humano pode compreender a finalidade, o fluxo de controle e operação de um código fonte.
Coexistência requerida	O grau em que o SUT deve desempenhar as suas funções de forma eficiente enquanto compartilha o mesmo ambiente de hardware ou software com outro sistema.
Capacidade de dados requerida	O volume de dados a serem armazenados em um arquivo ou banco de dados do SUT.
Nível de interoperabilidade requerido	A medida em que o SUT deve trocar informações com outros sistemas ou componentes para desempenhar suas funções.
Nível de desempenho requerido	A medida em que o SUT deve desempenhar suas funções sob limites de tempo de processamento e taxas de *throughput* específicos.
Nível de portabilidade requerido	A medida em que o SUT deve ser transferível de um ambiente de hardware ou software para outro.
Nível de recuperabilidade requerido	A medida em que, em caso de uma interrupção ou uma falha, o SUT deve recuperar os dados diretamente afetados e restabelecer o estado desejado do

	sistema.
Nível de confiabilidade requerido	A medida em que o SUT deve executar suas funções sob determinadas condições sem apresentar falhas por um período de tempo especificado.
Nível de segurança requerido	A medida em que o SUT deve proteger suas informações e dados.
Nível de usabilidade requerido	A medida em que a operação do SUT deve ser compreensível pelos usuários em um contexto de utilização.
Cobertura de testes requerida	A medida em que os casos e procedimentos de teste exercitam os requisitos técnicos e de negócio do sistema sob teste.
Complexidade dos requisitos	O nível de dificuldade para compreender os requisitos do SUT, influenciado pela quantidade e variedade de regras de negócio.
Tamanho da especificação de requisitos	A medida da quantidade de requisitos contemplados pelo sistema.
Estabilidade dos requisitos	A medida em que os requisitos mudam depois de terem sido aprovados - normalmente após a fase de especificação de requisitos.
Pressão sobre o cronograma	A restrição de tempo imposta ao projeto de teste, frequentemente colocando pressão sobre a equipe de teste.
Complexidade das interfaces do software	A medida da quantidade e diversidade de integrações do SUT com outros sistemas, influenciado pela quantidade e variedade de interfaces.
Procedimentos específicos para acessar o sistema	A medida dos procedimentos específicos necessários para acessar o SUT devido à política de segurança da empresa.
Suporte de ferramentas de teste	A medida em que ferramentas de teste estão efetivamente apoiando as atividades de teste de software.
Complexidade do sistema	A medida da quantidade e diversidade de componentes que formam o SUT, influenciado pela quantidade e variedade de operações, entidades e dispositivos manipulados pelo sistema.
Capacidade da equipe	A habilidade da equipe de teste para realizar um projeto de testes de forma eficaz e com habilidade.
Continuidade da equipe	A capacidade da equipe de teste de manter seus membros trabalhando juntos por um longo período de tempo sem alterações (baixo *turnover*).
Cooperação da equipe	O grau de cooperação entre os engenheiros de teste, desenvolvedores de software, gerentes de projetos, coordenadores e outras partes interessadas para realizar o trabalho de teste de software.
Produtividade da equipe	A medida da quantidade de tarefas e produtos de trabalho realizados pela equipe de teste ao longo do tempo.
Tamanho da especificação técnica	A medida da quantidade de informações técnicas contidas nos documentos que descrevem os serviços sob teste.
Quantidade de dados de teste	A quantidade de dados criados ou selecionados para satisfazer os requisitos de entrada para a execução de um ou mais casos de teste.
Complexidade de execução de teste	O nível de dificuldade para executar cada *script* de teste, considerando a dependência entre casos de teste, dados de teste, tipos de itens de tela a serem verificados, número de ações sobre a interface do usuário e a necessidade de manipulação de arquivos.
Capacidade do processo de teste	A medida de como o processo de testes do projeto apoia os membros da equipe de teste em produzir os resultados necessários.
Tamanho dos *scripts* de teste	A medida do número de passos necessários para executar cada caso de teste.
Tarefas do processo de teste	A quantidade de tarefas de teste necessárias para testar o software.
Tamanho da equipe de teste	O número de testadores disponíveis para executar as atividades de teste necessárias.
Ciclos de teste	O número de ciclos de teste necessários para atingir os critérios de qualidade exigidos pelo projeto.

Configuração de estratégia de teste n. ____

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução	Ferramenta(s) de teste
☐ Unidade ☐ Integração ☐ Sistema	☐ Funcional ☐ Não-funcional (_____)	☐ Caixa branca ☐ Caixa preta	☐ Manual ☐ Automatizada	

Nas colunas das atividades do processo de teste, marque um X em cada atividade que você efetivamente participou no seu último projeto de teste de software. Em seguida, de acordo com a estratégia de testes utilizada e em função de cada atividade do processo de teste, reflita sobre cada fator de esforço listado e indique com um "+" se ele exerceu influência positiva (aumentou o esforço) ou com um "-" se ele exerceu influência negativa (diminuiu o esforço) sobre aquela atividade específica. Caso o fator não tenha sido observado, basta deixar o campo em branco. Fique à vontade para indicar outros fatores que estejam ausentes desta lista.

Fator de esforço	**Atividade do processo**					
	☐ Planejar testes	☐ Configurar ambiente de testes	☐ Projetar e implementar testes	☐ Executar testes	☐ Comunicar incidentes de teste	☐ Finalizar projeto de testes
Diversidade de atores						
Quantidade e duração de interrupções						
Disponibilidade de recursos de infraestrutura						
Disponibilidade de artefatos de teste reusáveis						
Complexidade do código-fonte						
Tamanho do código-fonte						
Complexidade do ambiente de teste						
Qualidade da documentação						
Experiência com teste						
Experiência com ferramentas de teste						
Experiência com o domínio de aplicação						
Experiência com o ambiente operacional						
Experiência com a tecnologia de programação						
Nível de criticidade						
Nível de segurança física pessoal						
Número de casos de teste						
Número de Web Services						
Nível de confidencialidade do projeto						
Distribuição da equipe do projeto						
Quantidade de incidentes de teste						
Legibilidade de código						
Coexistência requerida						
Capacidade de dados requerida						
Nível de interoperabilidade requerido						
Nível de desempenho requerido						
Nível de portabilidade requerido						

Nível de recuperabilidade requerido						
Nível de confiabilidade requerido						
Nível de segurança requerido						
Nível de usabilidade requerido						
Cobertura de teste requerida						
Complexidade de requisitos						
Tamanho da especificação de requisitos						
Estabilidade dos requisitos						
Pressão sobre o cronograma						
Complexidade das interfaces do software						
Procedimentos espec. para acessar o sistema						
Suporte de ferramentas de teste						
Complexidade do sistema						
Capacidade da equipe						
Continuidade da equipe						
Cooperação da equipe						
Produtividade da equipe						
Tamanho da especificação técnica						
Quantidade de dados de teste						
Complexidade de execução de teste						
Capacidade do processo de teste						
Tamanho dos scripts de teste						
Tarefas do processo de teste						
Tamanho da equipe de teste						
Ciclos de teste						

APÊNDICE H – Continuação do QPET

Configuração de estratégia de teste n. ____

Nível de teste	Tipo de teste	Técnica de teste	Forma de execução	Ferramenta(s) de teste
☐ Unidade ☐ Integração ☐ Sistema	☐ Funcional ☐ Não-funcional (_____)	☐ Caixa branca ☐ Caixa preta	☐ Manual ☐ Automatizada	

Nas colunas das atividades do processo de teste, marque um X em cada atividade que você efetivamente participou no seu último projeto de teste de software. Em seguida, de acordo com a estratégia de testes utilizada e em função de cada atividade do processo de teste, reflita sobre cada fator de esforço listado e indique com um "+" se ele exerceu influência positiva (aumentou o esforço) ou com um "-" se ele exerceu influência negativa (diminuiu o esforço) sobre aquela atividade específica. Caso o fator não tenha sido observado, basta deixar o campo em branco.

Fator de esforço	Atividade do processo					
	☐ Planejar testes	☐ Configurar ambiente de testes	☐ Projetar e implementar testes	☐ Executar testes	☐ Comunicar incidentes de teste	☐ Finalizar projeto de testes
Número de suítes de teste O número de suítes de teste requeridas para organizar os casos de teste projetados.						
Número de asserções por passo de teste O número de asserções necessárias para cada passo do script de teste.						
Número de parâmetros por unidade O número de diferentes parâmetros requeridos por cada unidade do sistema.						
Complexidade dos scripts de teste O nível de dificuldade para desenvolver o script de teste, influenciado pela variedade de componentes e pelas dependências requeridas para implementar o script de teste.						
Nível de testabilidade do software O grau em que o sistema está suficientemente pronto para ser testado.						
Comprometimento da equipe do projeto A medida em que os membros da equipe do projeto desejam realizar as tarefas do projeto visando algum ganho.						
Estabilidade da interface gráfica do usuário A medida em que a GUI é modificada após os scripts de teste terem sido implementados.						
Número de versões O número de diferentes versões do SUT.						
Nível de usabilidade apresentado O nível de usabilidade apresentado pelo SUT durante os testes.						
Falta de fluência no idioma do projeto A medida da falta de fluência da equipe de testes no idioma utilizado no projeto.						
Indisponibilidade do ambiente de testes A medida em que o ambiente de testes "cai" ou tem a sua operação interrompida por problemas técnicos.						

APÊNDICE I – Caracterização dos Participantes do Survey

Figura 56: Caracterização dos participantes do *survey* (parte 1).

Uso de ferramentas de integração contínua

	total	percentage
Praticou em vários projetos na indústria	10	36%
Praticou em um projeto na indústria	8	29%
Estudou em cursos ou livros	5	18%
Nenhuma experiência	4	14%
Praticou em projetos de curso	1	4%

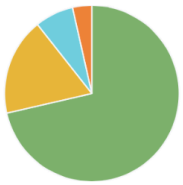
Uso de ferramentas de teste de web services

	total	percentage
Praticou em vários projetos na indústria	14	50%
Praticou em um projeto na indústria	7	25%
Praticou em projetos de curso	4	14%
Estudou em cursos ou livros	3	11%

Uso de ferramentas de teste de desempenho

	total	percentage
Nenhuma experiência	7	25%
Estudou em cursos ou livros	6	21%
Praticou em um projeto na indústria	5	18%
Praticou em vários projetos na indústria	5	18%
Praticou em projetos de curso	5	18%

Comunicação de incidentes de teste



	total	percentage
Praticou em vários projetos na indústria	20	71%
Nenhuma experiência	5	18%
Praticou em projetos de curso	2	7%
Praticou em um projeto na indústria	1	4%

Desenvolvimento de web services RESTful

	total	percentage
Praticou em vários projetos na indústria	10	36%
Nenhuma experiência	7	25%
Praticou em um projeto na indústria	5	18%
Estudou em cursos ou livros	3	11%
Praticou em projetos de curso	3	11%

Design e implementação de testes de web services RESTful

	total	percentage
Praticou em vários projetos na indústria	11	39%
Praticou em um projeto na indústria	7	25%
Nenhuma experiência	5	18%
Estudou em cursos ou livros	3	11%
Praticou em projetos de curso	2	7%

Execução de testes de web services RESTful

	total	percentage
Praticou em vários projetos na indústria	12	43%
Praticou em um projeto na indústria	8	29%
Nenhuma experiência	4	14%
Estudou em cursos ou livros	2	7%
Praticou em projetos de curso	2	7%

Desenvolvimento de aplicações mobile

	total	percentage
Estudou em cursos ou livros	8	29%
Praticou em um projeto na indústria	6	21%
Nenhuma experiência	6	21%
Praticou em projetos de curso	5	18%
Praticou em vários projetos na indústria	3	11%

Design e implementação de testes de aplicações mobile

	total	percentage
Nenhuma experiência	17	61%
Praticou em um projeto na indústria	4	14%
Estudou em cursos ou livros	4	14%
Praticou em projetos de curso	2	7%
Praticou em vários projetos na indústria	1	4%

Execução de testes de aplicações mobile

	total	percentage
Nenhuma experiência	16	57%
Praticou em um projeto na indústria	5	18%
Estudou em cursos ou livros	3	11%
Praticou em projetos de curso	2	7%
Praticou em vários projetos na indústria	2	7%

Figura 57: Caracterização dos participantes do *survey* (parte 2).

Figura 58: Caracterização dos participantes do *survey* – configuração 0 (parte 1).

Figura 59: Caracterização dos participantes do *survey* – configuração 0 (parte 2).

Figura 60: Caracterização dos participantes do *survey* – configuração 1 (parte 1).

Figura 61: Caracterização dos participantes do *survey* – configuração 1 (parte 2).

Figura 62: Caracterização dos participantes do *survey* – configuração 2 (parte 1).

Figura 63: Caracterização dos participantes do *survey* – configuração 2 (parte 2).

Figura 64: Caracterização dos participantes do *survey* – configuração 3 (parte 1).

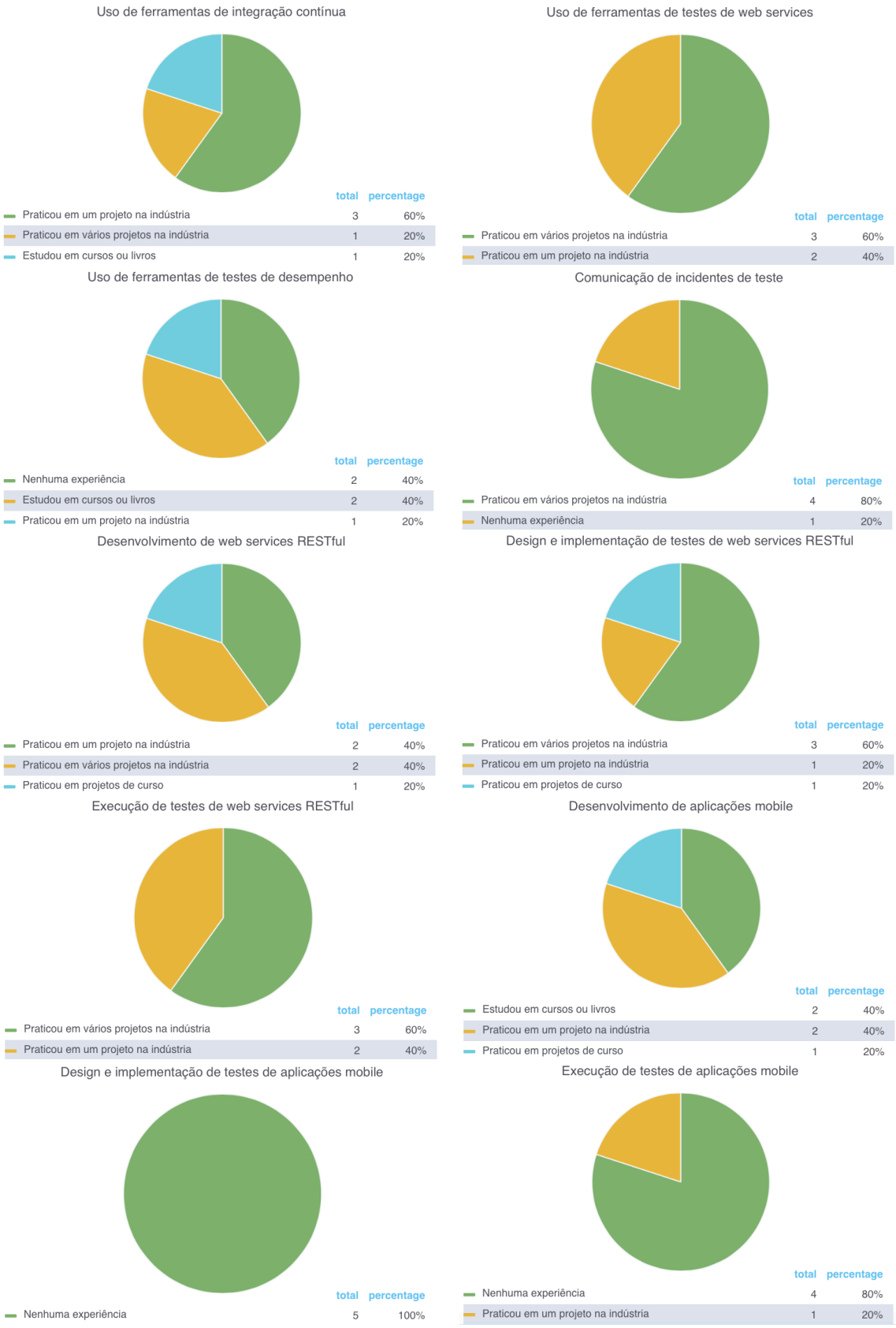


Figura 65: Caracterização dos participantes do *survey* – configuração 3 (parte 2).

Figura 66: Caracterização dos participantes do *survey* – configuração 4 (parte 1).

Figura 67: Caracterização dos participantes do *survey* – configuração 4 (parte 2).

Figura 68: Caracterização dos participantes do *survey* – configuração 5 (parte 1).

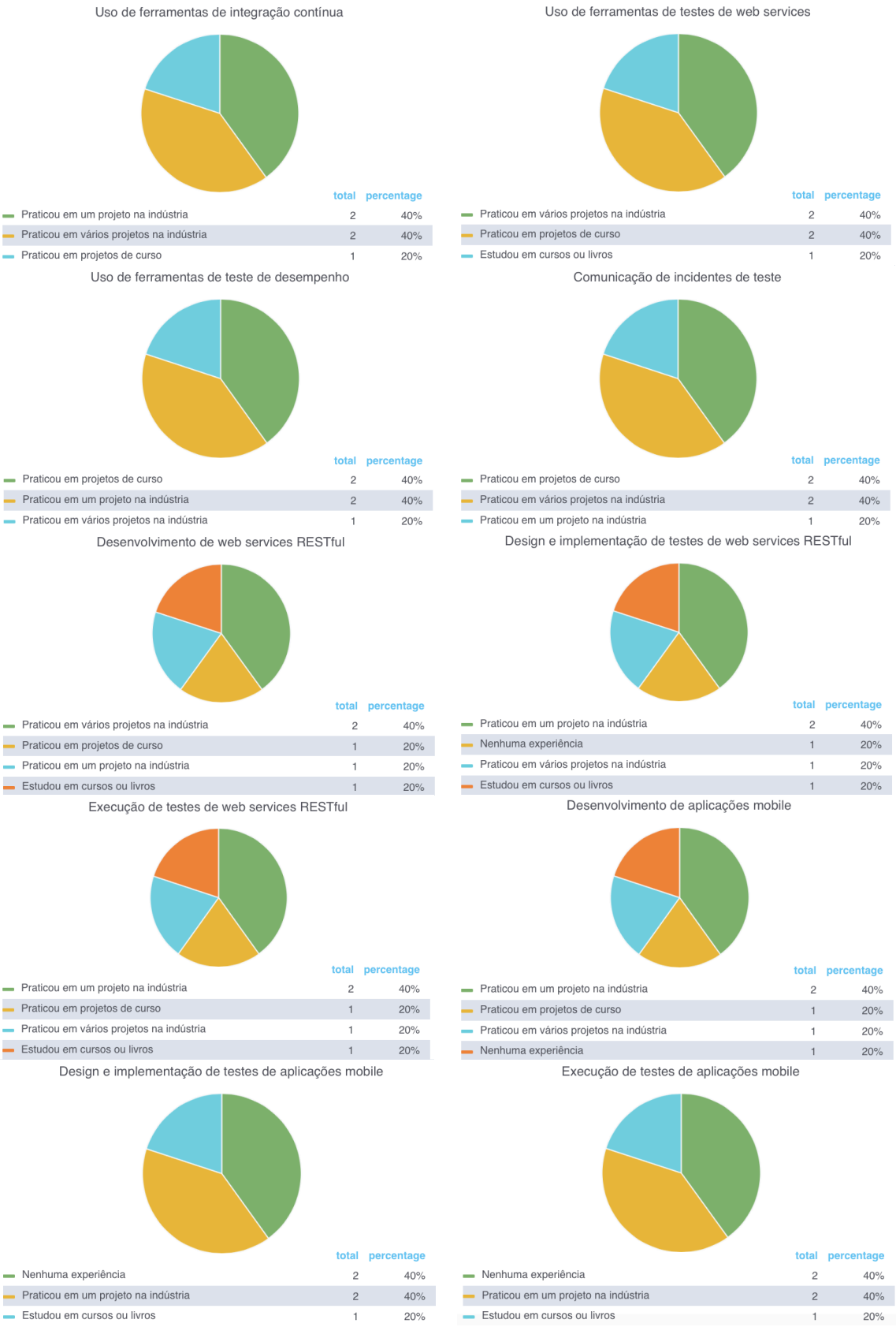


Figura 69: Caracterização dos participantes do *survey* – configuração 5 (parte 2).

APÊNDICE J – Percepções do Modelo HyperTest

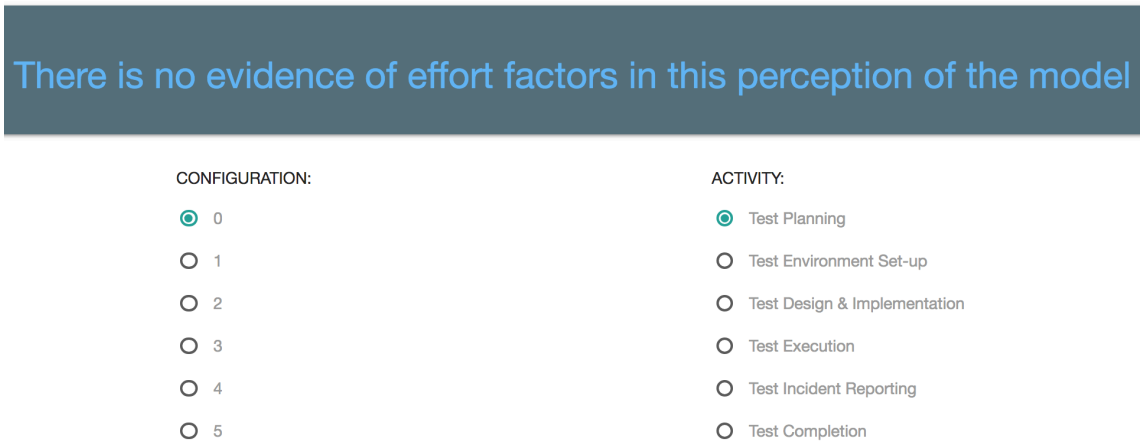


Figura 70: Percepção configuração 0 x "Planejar teste".

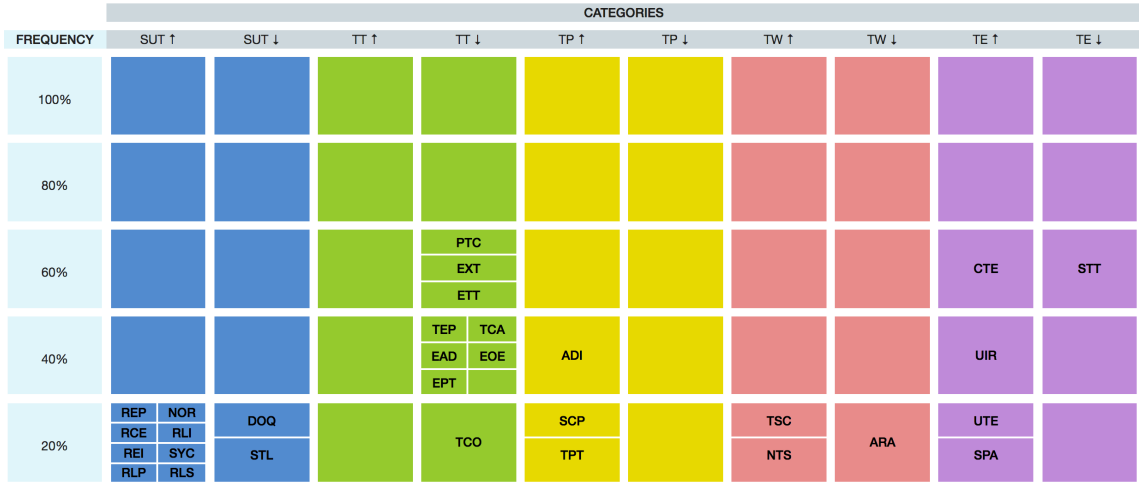


Figura 71: Percepção configuração 0 x "Configurar ambiente de teste".

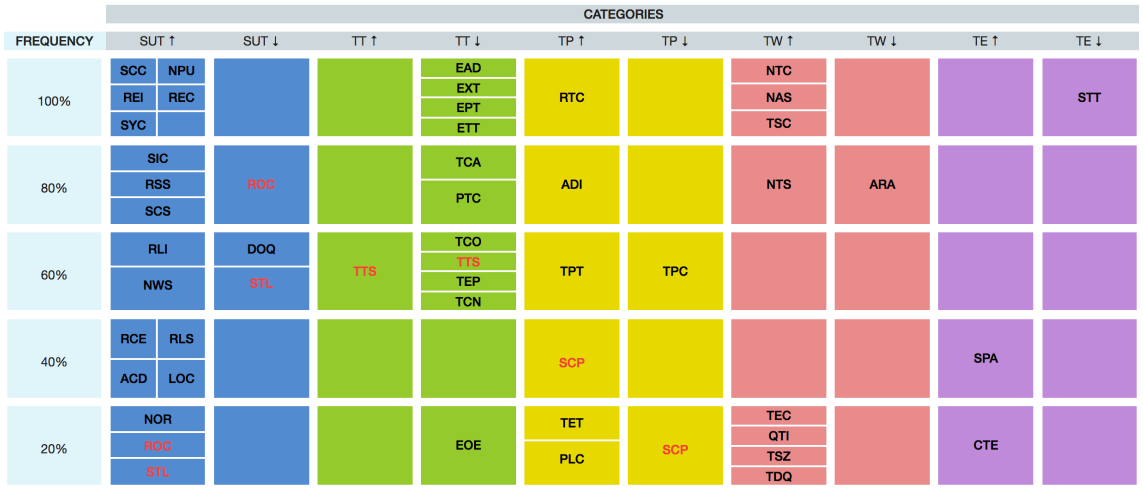


Figura 72: Percepção configuração 0 x "Projetar e implementar testes".

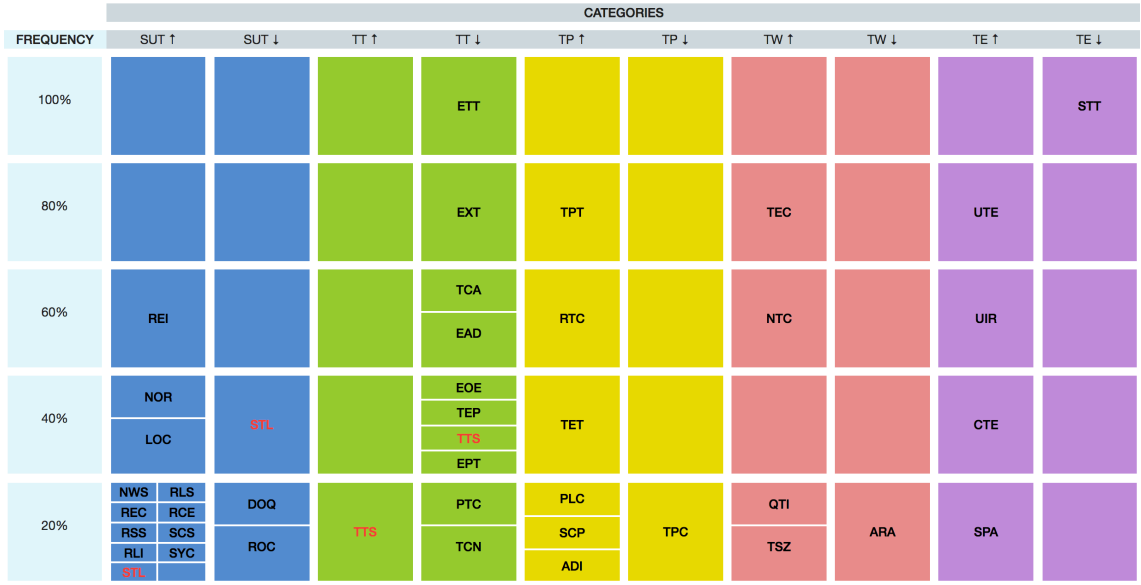


Figura 73: Percepção configuração 0 x "Executar testes".

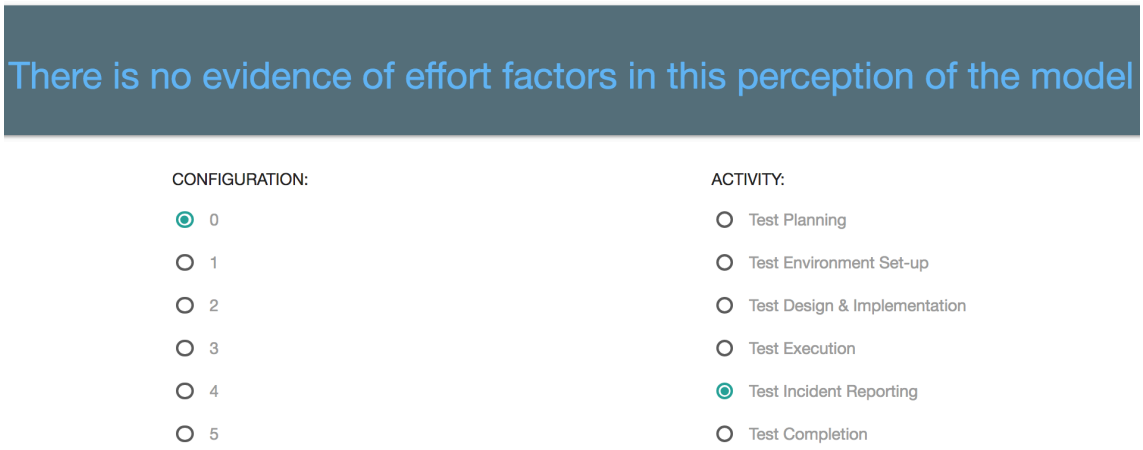


Figura 74: Percepção configuração 0 x "Comunicar incidentes de teste".

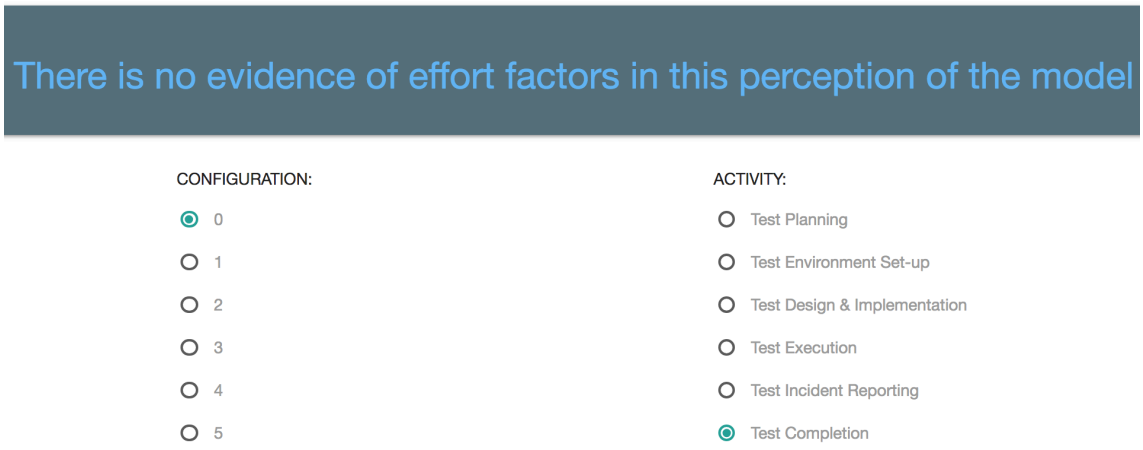


Figura 75: Percepção configuração 0 x "Finalizar projeto de teste".

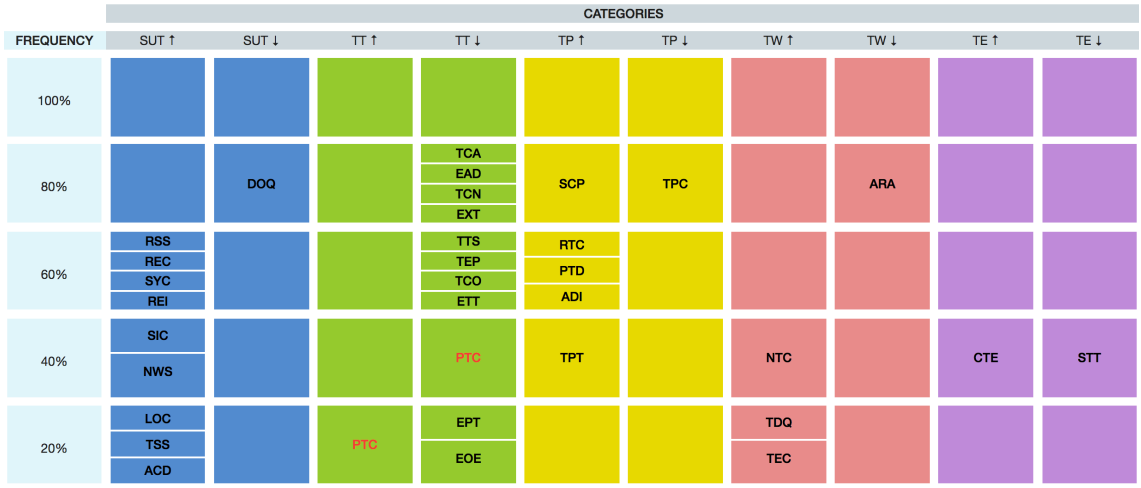


Figura 76: Percepção configuração 1 x "Planejar teste".

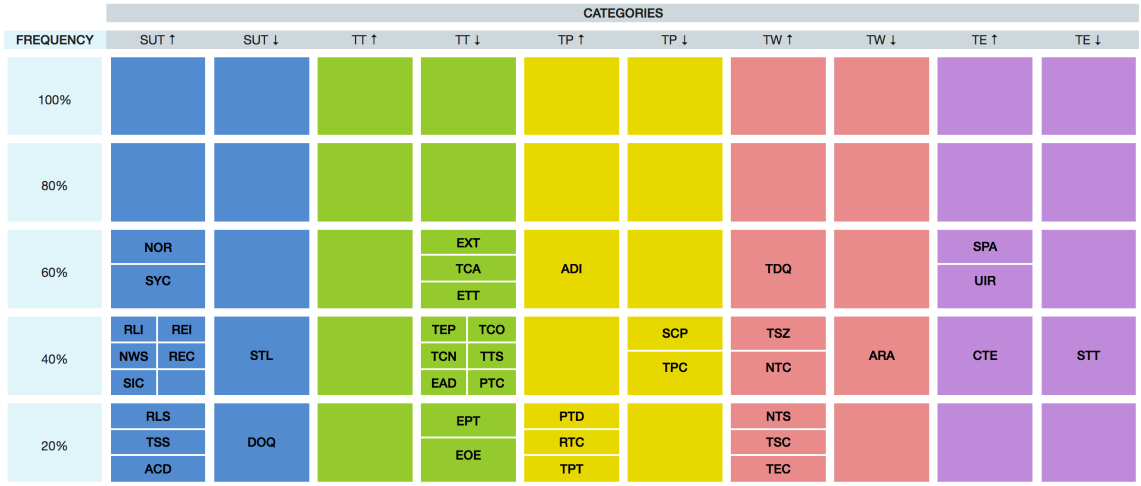


Figura 77: Percepção configuração 1 x "Configurar ambiente de teste".

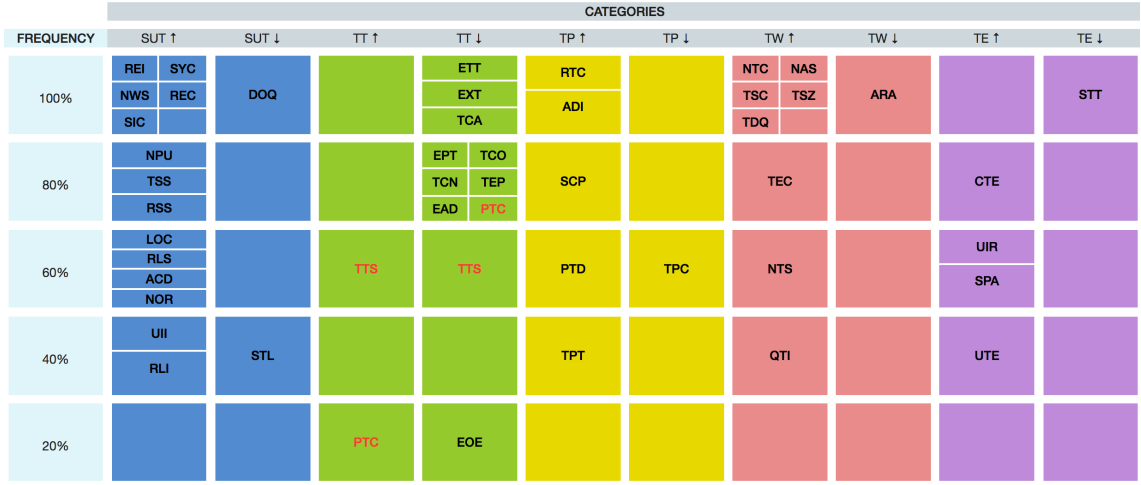


Figura 78: Percepção configuração 1 x "Projetar e implementar testes".

Figura 79: Percepção configuração 1 x "Executar testes".

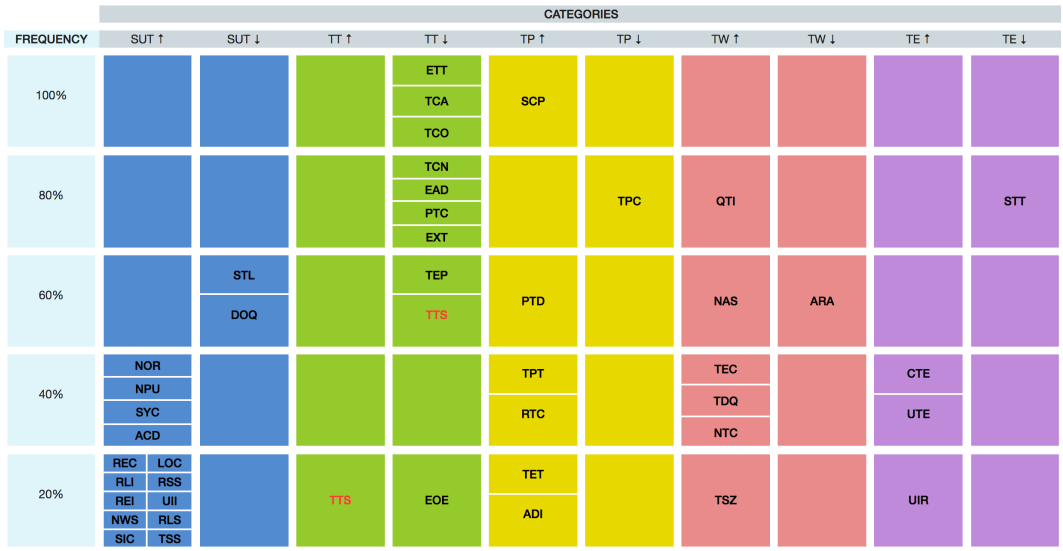


Figura 80: Percepção configuração 1 x "Comunicar incidentes de teste".

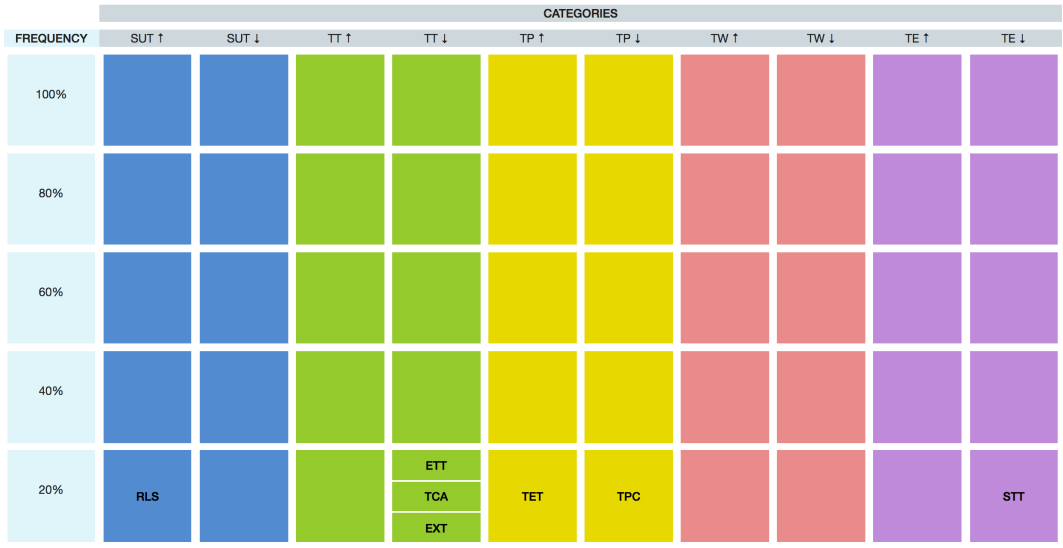


Figura 81: Percepção configuração 1 x "Finalizar projeto de teste".

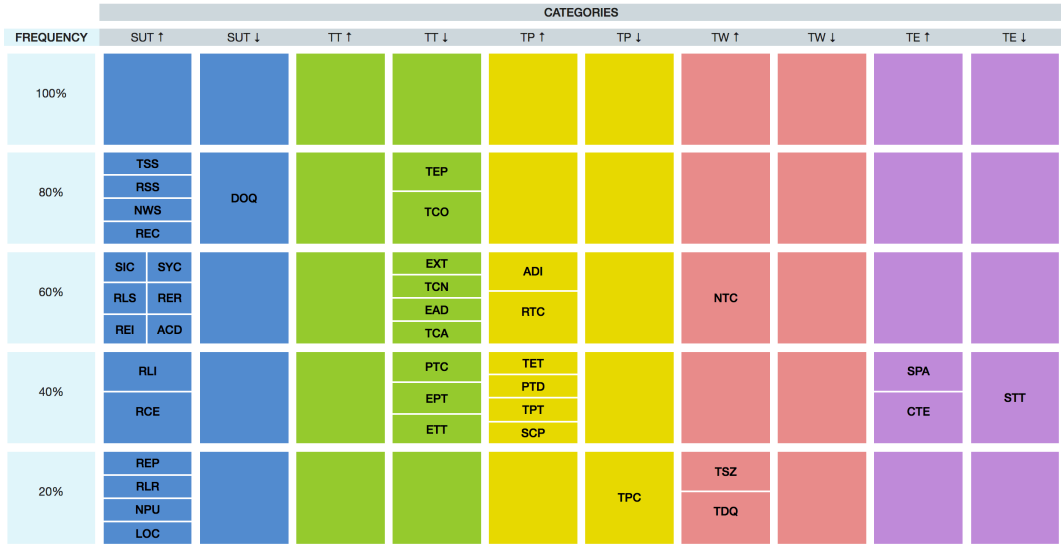


Figura 82: Percepção configuração 2 x "Planejar teste".

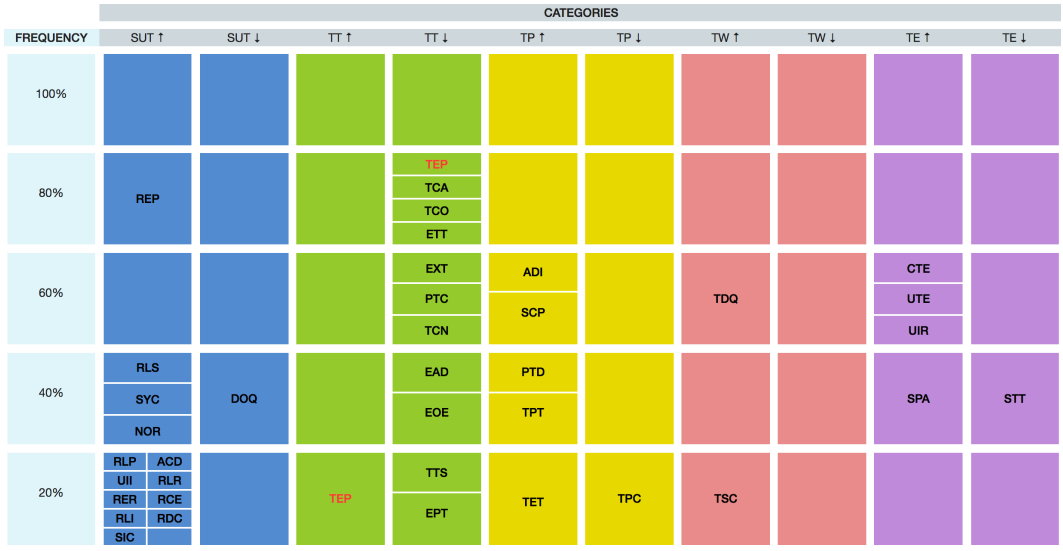


Figura 83: Percepção configuração 2 x "Configurar ambiente de teste".

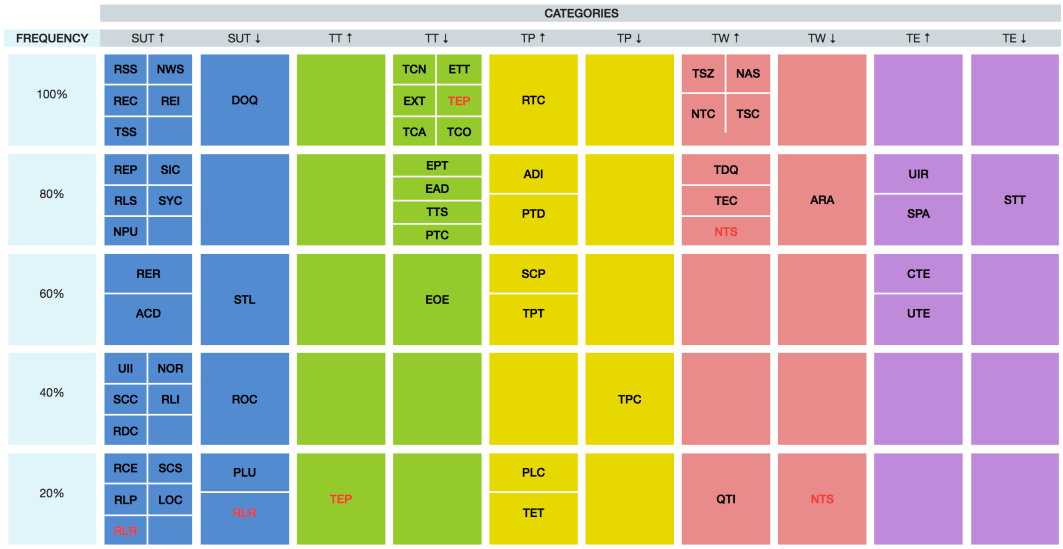


Figura 84: Percepção configuração 2 x "Projetar e implementar testes".

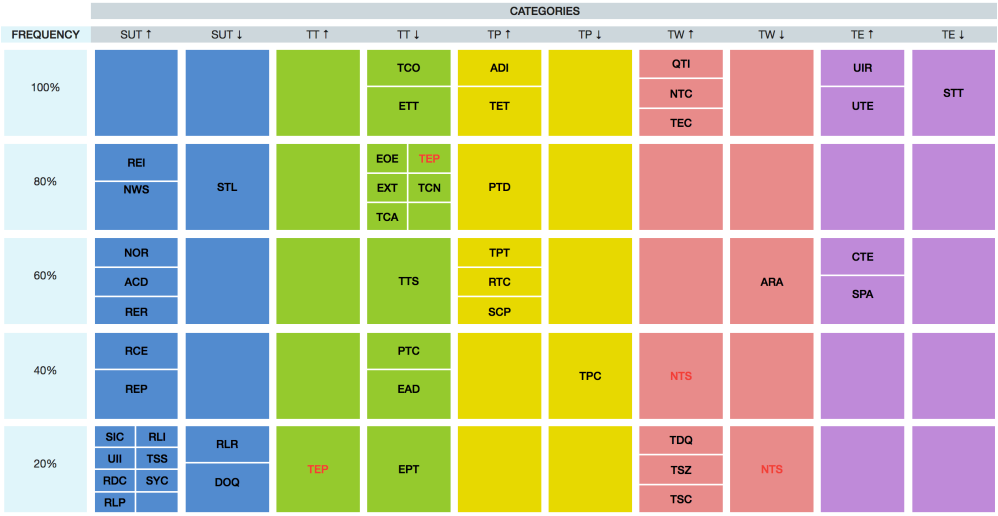


Figura 85: Percepção configuração 2 x “Executar testes”.

Figura 86: Percepção configuração 2 x “Comunicar incidentes de teste”.

Figura 87: Percepção configuração 2 x “Finalizar projeto de teste”.

Figura 88: Percepção configuração 3 x "Planejar teste".

Figura 89: Percepção configuração 3 x "Configurar ambiente de teste".

Figura 90: Percepção configuração 3 x "Projetar e implementar testes".

Figura 91: Percepção configuração 3 x "Executar testes".

Figura 92: Percepção configuração 3 x "Comunicar incidentes de teste".

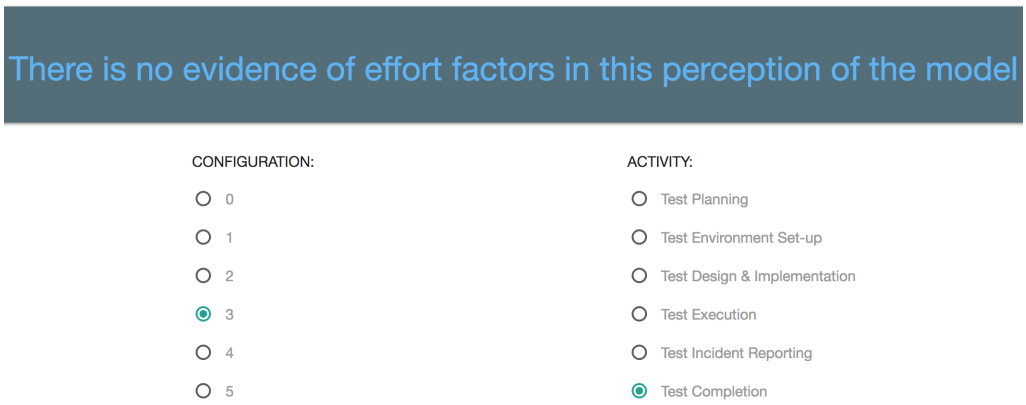


Figura 93: Percepção configuração 3 x "Finalizar projeto de teste".

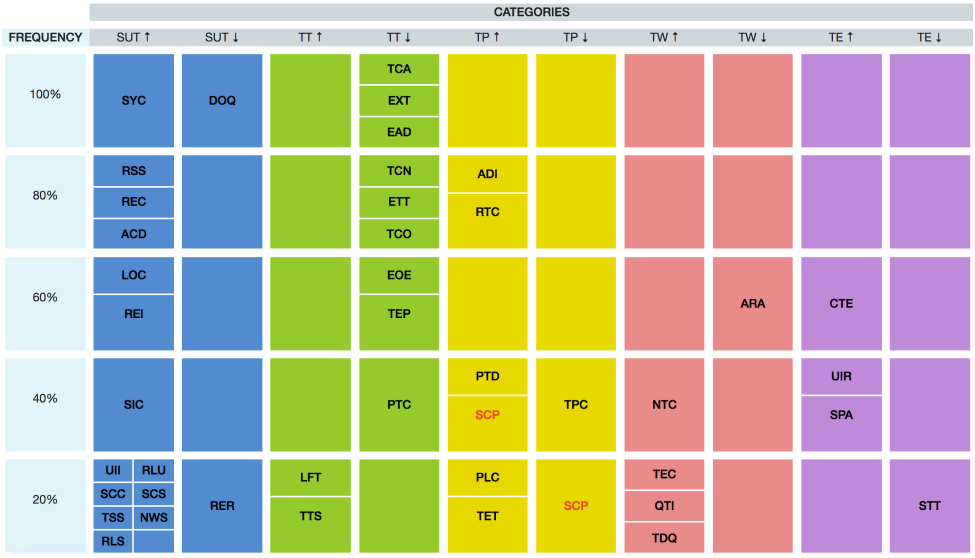


Figura 94: Percepção configuração 4 x "Planejar teste".

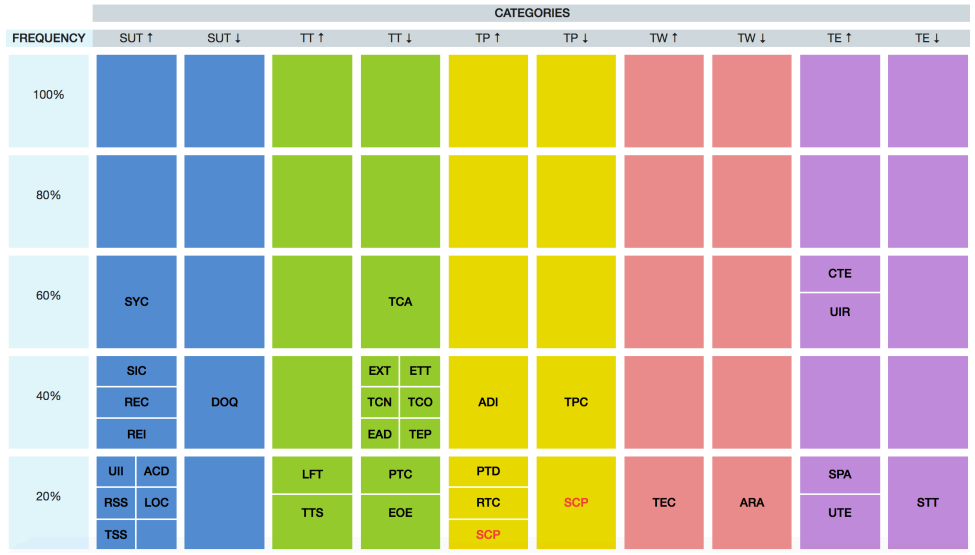


Figura 95: Percepção configuração 4 x "Configurar ambiente de teste".

Figura 96: Percepção configuração 4 x "Projetar e implementar testes".

Figura 97: Percepção configuração 4 x "Executar testes".

Figura 98: Percepção configuração 4 x "Comunicar incidentes de teste".

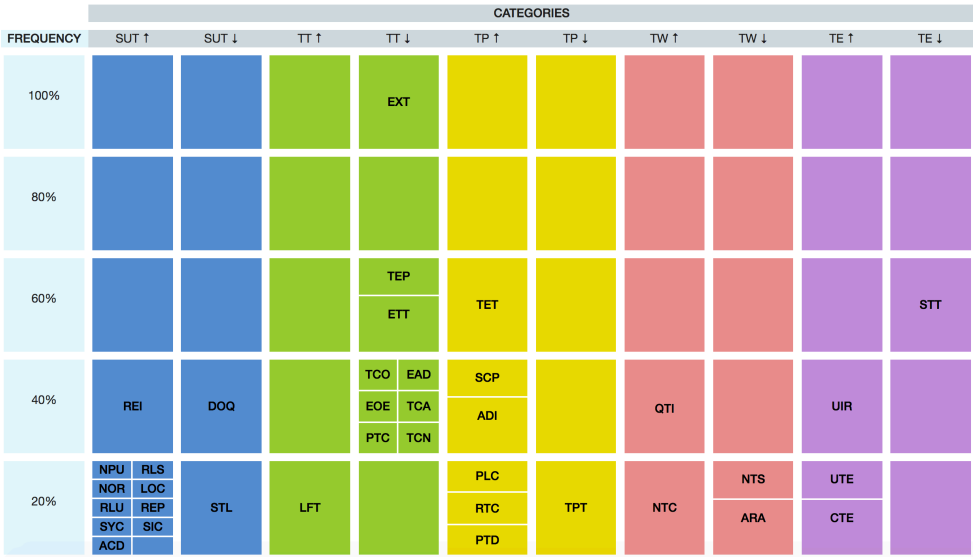


Figura 99: Percepção configuração 4 x "Finalizar projeto de teste".

Figura 100: Percepção configuração 5 x “Planejar teste”.

Figura 101: Percepção configuração 5 x “Configurar ambiente de teste”.

Figura 102: Percepção configuração 5 x “Projetar e implementar testes”.

Figura 103: Percepção configuração 5 x “Executar testes”.

Figura 104: Percepção configuração 5 x “Comunicar incidentes de teste”.

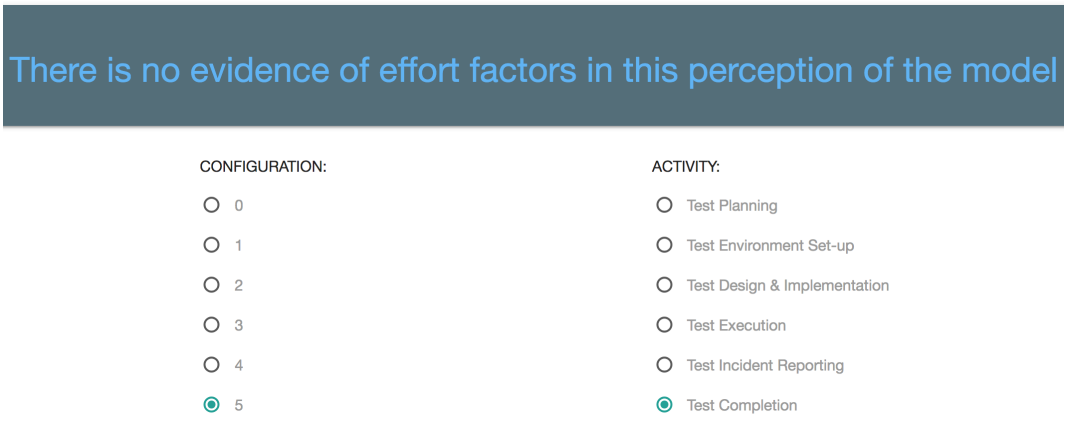


Figura 105: Perspectiva configuração 5 x “Finalizar projeto de teste”.