CONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO
ISSN 1984-9354

ANÁLISE COMPARATIVA ENTRE TÉCNICAS DE ESTIMATIVAS DE TESTE DE SOFTWARE: UM LEVANTAMENTO DA LITERATURA TÉCNICO-CIENTÍFICA

Área temática: Gestão pela Qualidade

Luciana Matieli
lumatieli@hotmail.com
Fernando Araujo
fernandoaraujo@id.uff.br

Resumo: *Software testing is an important activity of the software development life cycle, in order to contribute to the quality of the product /service. In this sense, invest time and resources in the test are important inputs to support the software testing estimates. Despite the empirical emphasis on software testing area, in Brazil, there are few the theoretical investigations on estimate of software testing. So in terms of methodology, this paper makes use of a related bibliometric survey the latest academic publications on the subject, in order to systematize the literature. As a result, the article provides a comparative analysis of software testing estimation techniques found in scientific and technical literature.*

Palavras-chaves:.

1. INTRODUÇÃO

Empresas Brasileiras de Telecomunicações, usualmente, contratam empresas terceiras para realizarem o desenvolvimento e o teste dos seus sistemas. Toda criação e/ou alteração de sistemas computacionais, além do desenvolvimento do software, demanda que sejam feitos testes funcionais e/ou testes de performance para que se tenha certeza de que a alteração foi realizada conforme os requisitos funcionais e técnicos especificados.

O teste é uma atividade fundamental para garantir a qualidade do software. Alguns dos principais desafios em testes são combinar os casos de teste com requisitos corretamente, para fornecer informações precisas, estimativas e acompanhar o andamento de teste em conformidade (LIOU, 2010).

Segundo Nguyen *et alli* (2013), teste de software é uma atividade importante no desenvolvimento de software e projetos de manutenção, garantindo a qualidade, aplicabilidade e utilidade de produtos de software. Na verdade, nenhum software pode ser liberado sem uma quantidade razoável de testes envolvidos. Para atingir a qualidade aceitável, as equipes de projeto de software dedicam parte substancial do esforço de desenvolvimento total para a realização teste. De acordo com relatórios da indústria, teste de software consome cerca de 10-25% do esforço total do projeto, e em alguns projetos este número pode chegar a 50%.

Para Zhou *et alli* (2013), o teste de software, definido como a execução sistemática do software com o objetivo de revelar falhas, é uma importante fase para validar a correção do software. As atividades de teste de software são compostas da definição dos casos de teste e validação do comportamento da execução. Em geral, a execução dos casos de teste é limitada, uma vez que validar todos os caminhos de execução tende a ser inviável em termos de prazo e custo. Assim, a qualidade dos casos de teste afeta diretamente a qualidade do software, e um dos atributos dos casos de teste de qualidade é a capacidade de detecção de falhas ainda por desvendar.

Para Lazic & Mastorakis (2008), o teste de software é uma atividade que pode dar visibilidade ao produto e qualidade do processo. Métricas de teste estão entre os fatos que no projeto os gestores podem utilizar para compreender a sua atual posição e priorizar suas atividades, de modo que eles podem reduzir o risco (ou impacto) de ficar sem tempo antes que o software esteja pronto para lançamento.

Na mesma linha de pensamento, Aranha & Borba (2007) mencionam que o teste de software é uma atividade que vem agregando cada vez mais qualidade ao processo de desenvolvimento e ao produto final, pois, vêm sendo enriquecido por metodologias próprias, ferramentas de automação e equipes independentes e capacitadas. Mas, assim como os projetos de desenvolvimento de software, os projetos de teste possuem os mesmos desafios e problemas em relação a custo e prazo.

Uma atividade comum realizada para garantir a qualidade de software é a execução de testes (ARANHA & BORBA, 2007).

A estimativa de teste de um sistema é subjetiva, pois existem variáveis que interferem no seu resultado final, como: o nível de conhecimento do profissional que está realizando a estimativa, a característica de cada sistema a ser testado, se o sistema permite executar testes em paralelo, sendo possível orçar mais profissionais em menos tempo, se a equipe que está alocada para planejar ou executar o teste é uma equipe de profissionais com pouca experiência, se existe documentação de apoio ou não, entre outras variáveis. Uma das maiores dificuldades no âmbito das atividades do teste de software é evidenciar com clareza as atividades que serão executadas com os seus custos e prazos.

Segundo Patel *et alli* (2001) gerar estimativa eficaz de um projeto de software é uma das atividades mais importantes e desafiadoras. Apenas com uma estimativa precisa e confiável é possível concluir um projeto no prazo. Estimativas desempenham um papel vital em todas as fases do ciclo de vida de desenvolvimento de software.

Conforme apontam Tronto *et alli* (2008), uma questão crítica no gerenciamento de projetos de software é como realizar uma estimativa precisa do tamanho, esforço, recursos, custo e tempo gasto no desenvolvimento do processo. Subestimativas geradas por pressões de tempo podem comprometer o desenvolvimento funcional e o teste de software. Da mesma forma que uma superestimativa pode não atender o projeto e gerar orçamentos não competitivos.

Na mesma linha de pensamento, Liou (2010) menciona que o objetivo da estimativa de software é gerar estimativas realistas, tanto da equipe do projeto, para o cliente (patrocinador do projeto). Uma estimativa imprecisa pode causar problemas que nenhum custo pode recuperar. Além disso, a produção de uma estimativa com a intenção de agradar o cliente, mas que não esteja de acordo com a equipe do projeto pode levar o projeto ao insucesso.

Reforça-se, nesse contexto a importância das atividades de teste de software além de uma correta estimativa. Contudo, no meio acadêmico internacional é pequeno o número de publicações especificamente voltadas a estimativa de teste de software. Ao fazer uma busca na base Scopus e Web of Science com palavras combinadas “Software Testing” and “Estimation” e “Software Test” and “Estimation” foram encontrados 188 publicações, porém a partir dos títulos e dos resumos puderam ser selecionadas apenas 7 publicações que tivessem alguma aderência ao tema em questão.

Destaca-se ainda uma carência específica de documentos técnico-científicos em português, o que representa uma fragilidade às organizações e profissionais que atuam em atividades de teste de software – ainda fortemente baseadas em empirismo.

Diante do referido desafio teórico, o presente estudo tem como objetivo oferecer uma sistematização da literatura técnico-científica acerca de metodologias e atividades de teste de software, incorporando também uma investigação na fronteira do estado das práticas mercadológicas relacionadas às atividades de teste de software.

Como contribuições esperadas, o presente trabalho visa estreitar uma lacuna observada na literatura, no que concerne à sistematização de estudos relacionados à estimativa de teste de software, visando consolidar as principais evidências e proposições apresentadas.

Em termos de estrutura, esse trabalho está organizado em 4 seções, a saber: a metodologia empregada no estudo está descrita na seção 2, a seção 3 está dedicada a descrever os resultados e discussões. E a seção 4 apresenta as conclusões e sugestões de estudos futuros.

2. METODOLOGIA

O levantamento bibliográfico foi feito a partir de periódicos indexados nas bases de dados multidisciplinares Scopus e Web of Science, acessadas no período de 11 de Novembro de 2013 a 20 de Maio de 2014 por meio do Portal de Periódicos da CAPES.

As primeiras pesquisas foram realizadas nas bases de dados SCOPUS e Web of Science utilizando-se as seguintes palavras-chave isoladamente: “Software Testing”; “Software Test”; “Estimation”. Esta primeira busca tinha como meta inicial identificar e investigar o quantitativo geral de publicações nas respectivas bases de dados, sem nenhuma restrição. Como resultado desta primeira pesquisa foi possível evidenciar muitos temas relacionadas as três palavras-chave pesquisadas isoladamente. Dentro da área de teste de software foram encontrados temas sobre previsão de falhas em grandes

projetos de software, automatização de teste software, teste de regressão, estimativa do número de defeitos encontrados aos longos dos testes, geração de dados de testes (entradas), estimativas para a geração de casos de teste, dentre outros.

Em função da alta dispersão e variedade de registros encontrados fez-se necessário um refinamento da pesquisa, sendo necessário combinar algumas palavras "Software Testing" AND "Estimation" e "Software Test" AND "Estimation" com o intuito de restringir o universo da pesquisa, além de considerar apenas os artigos da área da ciência da computação.

Após a realização dessas buscas, os 25 registros em duplicidade encontrados nas bases SCOPUS e Web of Science foram devidamente descartados, permanecendo um total de 188 artigos a serem analisados, a partir dos títulos e dos resumos.

A partir da análise dos títulos e resumos dos 188 artigos, foram selecionados 7 para leitura e análise na íntegra, sendo considerados relevantes e aderentes com o estudo em questão. A seguir, o resultado obtido é mostrado, concentrando-se a apresentação nas principais referências para cada tópico levado em consideração. Os dados dos artigos estão ilustrados no Quadro 1.

Quadro 1. Artigos encontrados

Autor (Ano de Publicação)	**Resumo Artigo**
Christodoulakis D. & Panziou G. (1990)	Técnicas de estimação óptimas são aplicadas para prever o futuro comportamento dos sistemas de software.
JIANG, Li-Xin; HAN, Wan-Jiang; YAN, Chen-Chen & SHI, Bo-Ying (2012)	Neste trabalho é proposto um modelo de estimativa do tamanho do teste de função, com base em etapas de teste. O modelo é aplicável a teste de caixa preta. É realizada análise global do modelo COCOMO. Seus passos básicos incluem: casos de teste de design, etapas totais do teste de soma, definir os parâmetros do modelo com base no padrão de modelo, definir fator tamanho, e calcular o tamanho do teste.
Liou, Jing-Chiou (2010)	Neste trabalho, apresentamos um modelo paramétrico para estimar o teste de software, juntamente com um gráfico de teste para combinar os casos de teste com os requisitos e casos de teste de análise para auxiliar na produção de estimativas mais precisas.
Nguyen, Vu; Pham, Vu & Lam,Vu (2013)	Este artigo descreve um processo simples, nomeado de qEstimation, para estimar o tamanho e esforço das atividades de teste de software. O processo incorpora uma proposta de abordagem para medir o tamanho da o caso de teste com base em seus postos de controle, pré-condições e os dados de teste, bem como o tipo de teste.

Srinivasan, Krishnamoorthy & Fisher, Douglas (1995)	Este artigo descreve dois métodos de aprendizado de máquina, que que usamos para construir estimadores de esforço de desenvolvimento de software a partir de dados históricos.
Tronto, Iris Fabiana de Barcelos; Silva, José Demísio Simões da; & Sant'anna, Nilson (2008)	Este trabalho tem por objetivo oferecer métodos alternativos para aqueles que não acreditam em modelos de estimação, baseados em redes neurais artificiais e regressão.
Zhu Xiaochun; Zhou Bo; Hou LI; Chen Junbo & Chen Lu (2008)	Este artigo propõe uma abordagem baseada em experiência para estimar o esforço de execução de teste. A abordagem é caracterizada por um conjunto de testes como um vetor de 3 dimensões,que combina o número de processo de teste, a complexidade da execução do teste e a experiência dos testador.

Fonte: Os autores (2013)

De forma análoga as pesquisas das bases Scopus e Web of Science, foi feito o levantamento na base de dados SciELO.Org no realizado no período de 25 de Maio de 2014 a 28 de Maio de 2014, os resultados desta pesquisa evidenciaram muitos temas relacionados as três palavras-chave pesquisadas isoladamente "Software Testing"; "Software Test"; "Estimation", sendo necessário combinar algumas palavras "Software Testing" AND "Estimation" e "Software Test" AND "Estimation" com o intuito de restringir o universo da pesquisa.

A busca na base de dados SciELO.Org não retornou nenhuma publicação pertinente para o desenvolvimento da revisão bibliográfica, mesmo encontrando 19 artigos, nenhum corresponde ao tema central da pesquisa. A Figura 1 ilustra o resumo das pesquisas nas bases Scopus, Web of Science e SciELO.Org.

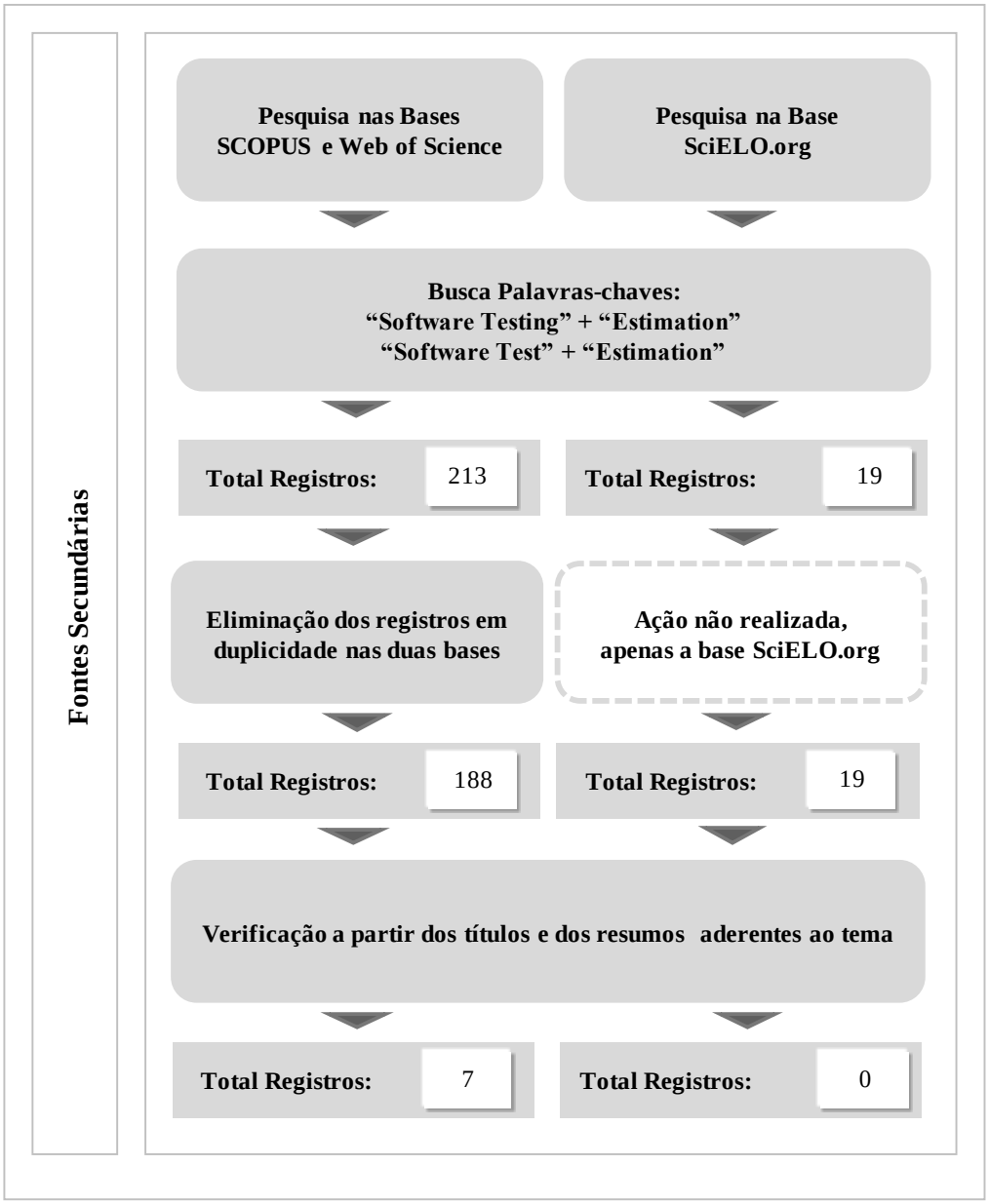

Figura 1. Pesquisas nas bases SCOPUS, Web of Science e SciElo.Org

Fonte: Os autores (2014)

3. RESULTADOS E DISCUSSÃO

Segundo Lopes & Nelson (2008), a maioria dos modelos de estimativa aplicados ao desenvolvimento visa também estimar o esforço de teste dado à importância dessa atividade. Desta forma foi possível identificar diferentes oito eixos de discussão sobre a estimativa de teste de software: (i) Estimativa gerada a partir da distribuição do esforço de um projeto de desenvolvimento de software; (ii) Estimativa gerada por uma porcentagem do tempo para o desenvolvimento; (iii) Estimativa gerada em pontos por função (PF); (iv) Estimativa gerada em pontos por caso de uso (UCP); (v) Estimativa

gerada em pontos por casos de teste (TCP); (vi) Estimativa gerada pela análise de pontos de testes (APT); (vii) Estimativa gerada a partir de bases históricas de projetos de teste; (viii) Estimativa gerada a partir da Simple Estimate Test (SET).

3.1 Estimativa gerada a partir da distribuição do esforço de um projeto de desenvolvimento de software

Para Patel *et alli* (2001) tradicionalmente, a estimativa de esforço para testes era uma percentagem do resto das fases do ciclo de vida de desenvolvimento. Esta abordagem para a estimativa é mais propenso a erros e acarreta um risco maior de atrasar os prazos de lançamento do produto.

Segundo Pressman (2005) a distribuição recomendada de esforço em todo o processo de software é muitas vezes referida como a regra 40-20-40. Esta regra recomenda que 40% do esforço é reservado para a análise e projeto, 20% do esforço é reservado para a codificação e 40% do esforço é reservado para o teste. Essa distribuição do esforço deve ser utilizada apenas como uma diretriz e varia de acordo com a natureza de cada projeto. A regra é bem simples de ser aplicada, conforme ilustrado na Figura 2.

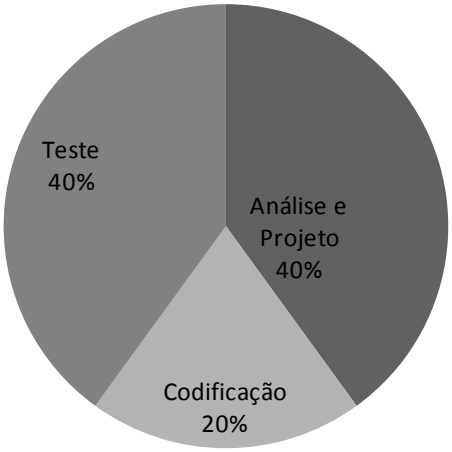

Figura 2. Distribuição do Esforço (40-20-40)
Fonte: Pressman (2005)

Para Matieli (2014) aplicando um percentual para estimar o esforço de teste, os pontos importantes como a cobertura e a complexidade dos testes não são considerados, e podem ter sistemas que exigem

uma complexidade muito maior nos testes. Nestes casos o esforço de teste estimado por esta regra tende a ser bem menor do que realmente deverá ser realizado.

Hoje a regra 40-20-40 está sob ataque. Alguns acreditam que mais de 40% do esforço global deve ser gasto durante a análise e design. Por outro lado, alguns defensores do desenvolvimento ágil argumentam que menos tempo deve ser gasto "na frente" e que uma equipe deve se mover rapidamente para a construção.

3.2 Estimativa gerada baseada na porcentagem do tempo para o desenvolvimento

Segundo Lopes & Nelson (2008), nesta técnica o esforço estimado de teste é baseado no valor estimado do tempo/esforço do desenvolvimento (número de linhas de códigos ou pontos por função), ou seja, é feita a contagem do número de linhas de código ou de pontos por função e aplica-se neste resultado um percentual para estimar o tempo do teste. Segundo Matieli (2014) um ponto positivo desta técnica seria que após a contagem do número de linhas de código ou pontos de função aplicando-se um percentual a estimativa de teste é gerada com maior agilidade. Todavia nesta técnica não é levada em consideração a natureza do teste, ambiente, complexidade dos casos de teste, entre outras variáveis.

Para Nageswaran (2001) este método não é baseado em nenhum princípio científico ou técnico. Atrasos nos cronogramas podem variar 50-75% do tempo estimado. Este método está longe de ser o mais utilizado.

3.3 Estimativa gerada em pontos por função (PF)

Para Lopes & Nelson (2008), o esforço estimado de caso de teste é determinado pela estimativa de ponto de função segundo Capers Jones. Utiliza-se a fórmula: número de casos de teste = (ponto de função) elevado a 1.2. Pode gerar estimativas precisas desde que a análise de pontos de função também seja precisa. A estimativa de Capers Jones apresenta esforços diferentes dependendo da consideração feita por ter um crescimento polinomial, onde número de casos de teste é igual a (ponto de função) elevado a 1.2. O número de casos de testes cresce à medida que cresce o tamanho do sistema. Portanto, ao considerarmos a estimativa para cada caso de uso e depois somarmos o resultado obtido de todos os

casos de uso, o resultado do esforço será bem menor do que quando consideramos o tamanho total do projeto. Quanto maior o tamanho em pontos de função, maior será o esforço despendido. Outro ponto é que esta técnica não oferece a estimativa de teste em horas e também não considera nenhuma característica do projeto de software.

Segundo Mastorakis *et alli* (2009) a desvantagem da utilização desta técnica é que requisitos são exigidos com antecedência.

3.4 Estimativas gerada em pontos por caso de uso (UCP)

Para Aranha & Borba (2007), a estimativa gerada pelos pontos de caso de uso (UCP) é uma extensão da estimativa gerada pelo ponto de função (PF) e estima o tamanho de um sistema com base nas especificações de caso de uso. Ambos UCP e PF consideram a complexidade do desenvolvimento de um sistema.

Segundo Lopes & Nelson (2008), o esforço estimado para teste é encontrado multiplicando o UCP ajustado com um fator de conversão. Este fator de conversão denota o número de homens/hora que representa esforço de teste requerido para uma combinação de linguagem e tecnologia. Este fator de conversão é determinado pela organização para as dadas combinações. Pontos por caso de uso ou use case points (UCP) foi desenvolvido baseado em pontos por função e a filosofia na qual se baseiam os dois métodos é a mesma, ou seja, as funcionalidades percebidas pelos usuários são a base para estimar o tamanho do software. A inexistência de padrões universais para a construção de casos de uso dificulta a comparação entre projetos de diferentes organizações. Sendo assim se os critérios utilizados para construir os casos de usos forem muito diversificados não há como garantir que os casos de usos estarão medindo a mesma coisa.

3.5 Estimativa gerada em pontos por casos de teste (TCP)

Para Patel *et alli* (2001), a estimativa baseadas em pontos por casos de teste é considerada uma das estimativas para testes funcionais mais exatas por enfatizar fatores que determinam a complexidade do ciclo de testes como um todo. Esta técnica combina quatro fases do processo de teste: a geração dos casos de testes, a implementação dos scripts para os testes automatizados, a execução dos testes manuais e a execução dos testes automatizados. Ela pode ser utilizada também em processos em que se aplicam uma ou mais fases do processo de teste.

Ainda segundo Patel *et alli* (2001) como dito acima, a análise TCP gera esforços de teste para as atividades de teste separados, isto é essencial porque os projetos de teste tendem a cair sob quatro diferentes modelos. Embora, na prática, a maioria dos projetos de teste é uma combinação dos quatro modelos de execução, conforme ilustrado na Figura 3.

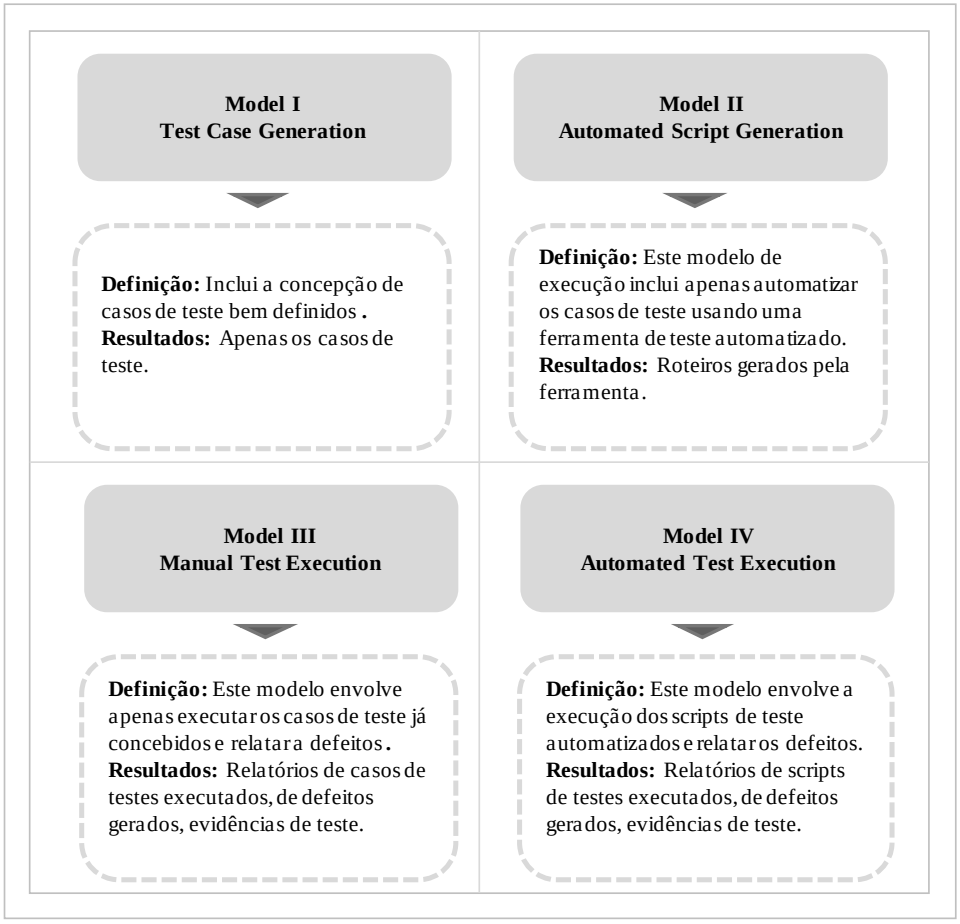

Figura 3. Modelos de execução de teste

Fonte: Adaptado de Patel *et alli* (2001)

Uma vantagem observada nesta técnica é a ênfase dada a automatização dos testes, e a flexibilidade de estimar testes de software que sejam executados: só manualmente, só automatizados ou manual e automatizado. A técnica não estabelece nenhum critério para transformar os pontos obtidos por caso de teste em esforço de testes.

3.6 Estimativa gerada em análise de pontos de testes (APT)

Segundo Souza *et alli* (2010) dentre as técnicas de estimativa de esforço para teste de software, a análise de pontos de teste (APT) é considerada na literatura como a mais consistente. Mas, apesar dos

autores da técnica afirmarem que a mesma foi utilizada por algumas organizações e obtiveram bons resultados, não foram encontrados relatos detalhados ou medições que comprovassem tais resultados.

Para Veenendaal & Dekkers (1999) o esforço estimado é para definir, desenvolver e executar testes funcionais, baseados na complexidade do desenvolvimento de software (derivado a partir da técnica de análise de pontos de função), considerando também a estratégia de testes e a produtividade.

Ainda segundo Veenendaal & Dekkers (1999) o maior benefício da APT está em conseguir reunir de forma sistemática fatores que influenciam o esforço específico de uma das etapas do processo de desenvolvimento, produzindo resultados mais precisos. Além disso, permite avaliar o esforço de teste por atividade e leva em consideração as etapas de planejamento e controle que são essenciais para o sucesso de qualquer projeto e devem ser consideradas em técnicas de estimativas de esforço.

Esta técnica APT pode ser considerada complexa e de difícil utilização e interpretação. São diversas variáveis como o tamanho do sistema em pontos de função, considerando também as características de qualidade a serem testadas dinamicamente, a produtividade da equipe de testes e a estratégia de testes dinâmicos ou estáticos sendo que seus valores são definidos através da análise dos diversos fatores que as compõem (Lopes & Nelson, 2008).

Segundo Bastos *et alli* (2007) a APT considera como base da sua análise, o tamanho do sistema a ser testado a partir das informações coletadas junto à equipe de desenvolvimento. Na definição e no cálculo do número de horas de teste, o tamanho do sistema em pontos de função é a base de cálculo inicial utilizada nestas estimativas. Os testes são também afetados pelos seguintes fatores:

a. O grau de complexidade do processo de teste. Com isto queremos dizer que sistemas muito complexos tendem a consumir mais horas de teste do que aqueles mais simples, como era de se esperar.
b. O nível de qualidade que se pretende alcançar com os testes. Evidentemente que a exigência de níveis de qualidade muito altos vão demandar um esforço maior de teste. Existem casos de sistemas financeiros, por exemplo, que devem estar isentos de defeito, ou possuir uma previsão de taxa de defeitos perto de zero ao entrarem em produção. Nestes casos o esforço de teste tenderá a ser maior.
c. O grau de envolvimento dos usuários com os testes. Quanto maior for o envolvimento dos usuários na atividade de teste ou homologação, tanto melhores serão os resultados alcançados e, muitas vezes, menor será o esforço de testar.

d. As interfaces que as funções que estão sendo testadas têm com os arquivos. Quanto maior o número de arquivos envolvidos em um caso de teste, por exemplo, mais difícil será testar o sistema ou software. Se o caso de teste, por exemplo, faz a manutenção apenas de um arquivo, muito mais fácil será testá-lo e avaliar os resultados alcançados.
e. A qualidade do sistema que está sendo testado (o ciclo de reincidência de defeitos). Quando o sistema tem defeitos de desenvolvimento ou de projeto, certamente, o trabalho de testá-lo será muito mais oneroso. Neste caso vale também aquela máxima que os custos dos defeitos crescem em progressão geométrica quanto mais tarde forem sendo encontrados. Defeitos de produção são muito mais caros que defeitos de projeto.
f. O nível de cobertura esperado com os testes. Os requisitos do sistema normalmente vão estabelecer o nível de cobertura exigido pelos testes. Por exemplo, para alguns sistemas os testes de carga ou de desempenho são imprescindíveis, enquanto que para outros, onde o nível de acesso é muito baixo, que esses tipos de testes podem até ser dispensados.
g. A experiência e a produtividade da equipe de testes (medidos através de indicadores históricos). Evidentemente que a qualidade da equipe de teste e da sua gestão estão ligados ao esforço despendido para a execução dos testes. Existem organizações que utilizam bases históricas para estimar e avaliar o seu processo de teste. Estas informações servem de base também para medir a produtividade da equipe.
h. O grau de automação dos testes. As ferramentas de automação permitem níveis maiores de produtividade, pois facilitam a repetição de testes já executados, tantas vezes quantas forem necessárias, além de facilitar a documentação dos casos de teste.
i. A qualidade do ambiente de teste, inclusive a sua capacidade de simular o ambiente de produção. O ambiente de teste deve estar bastante próximo ou ser igual ao ambiente de produção, pois com isso evitam-se muitos problemas provenientes da compatibilidade entre ambientes. Isto será especialmente importante para os testes de estresse, carga e desempenho.
j. A qualidade da documentação do sistema e, especialmente, dos requisitos. Como os requisitos são a base para o desenvolvimento do sistema, teremos um processo em cadeia, pois o teste apenas indica a ocorrência de defeitos, porém não resolve o problema decorrente de um projeto mal feito. Neste caso os custos de correção tendem a ser muito altos e o esforço de teste também será muito maior. A utilização da análise de pontos de teste vai permitir que o processo de testes tenha uma medida própria para determinar o seu tamanho, o que se justifica quando o teste for tratado como

uma atividade independente, mantendo-se a ligação com o tamanho do sistema em pontos de função.

3.7 Estimativa gerada a partir de bases históricas de projetos de teste

Segundo Lopes & Nelson (2008), o esforço estimado através de uma base histórica é baseado na coleta das informações armazenadas no banco de dados dos projetos, onde os requisitos de negócio são as informações básicas para as estimativas. Para que o processo de estimar o esforço seja realmente consistente, é necessário que os registros históricos de dados sejam extremamente organizados e sistemáticos para que os números produzidos tenham a maior exatidão possível.

Para Matieli (2014) esta técnica é positiva, pois gera um histórico de projetos dentro da organização, sendo possível realizar consultas de projetos análogos para realizar estimativas mais consistentes, gerar dados estatísticos. Porém, para que o processo de estimar esforço seja realmente consistente, é necessário que os registros históricos de dados sejam extremamente organizados e sistemáticos para que os números produzidos tenham a maior exatidão possível.

A geração de bases históricas das estimativas dos projetos é uma fonte para a elaboração de indicadores (SOUZA *et alli*, 2010).

3.8 Estimativa gerada a partir da Simple Estimate Test (SET)

Segundo Matieli (2014), o Simple Estimate Test (SET) foi proposto de acordo com a combinação da literatura técnico-científica, com a contribuição de respostas provenientes de investigação empírica aplicada junto a especialistas da área de teste de software.

Ainda para Matieli (2014), a técnica pode ser utilizada para estimar testes funcionais levando em consideração variáveis como: a definição dos casos de teste, complexidades dos casos de teste e suas respectivas classificações, horas de planejamento dos casos de teste, horas de execução dos casos de teste, horas de reteste, horas de gestão do teste, além da variável cronograma. Estas variáveis para a composição do modelo, seus respectivos pesos e percentuais foram definidas após uma análise semântica das respostas, a um questionário, aplicado de especialistas de uma grande empresa multinacional do setor de telecomunicações. Conforme ilustrado na Figura 4.

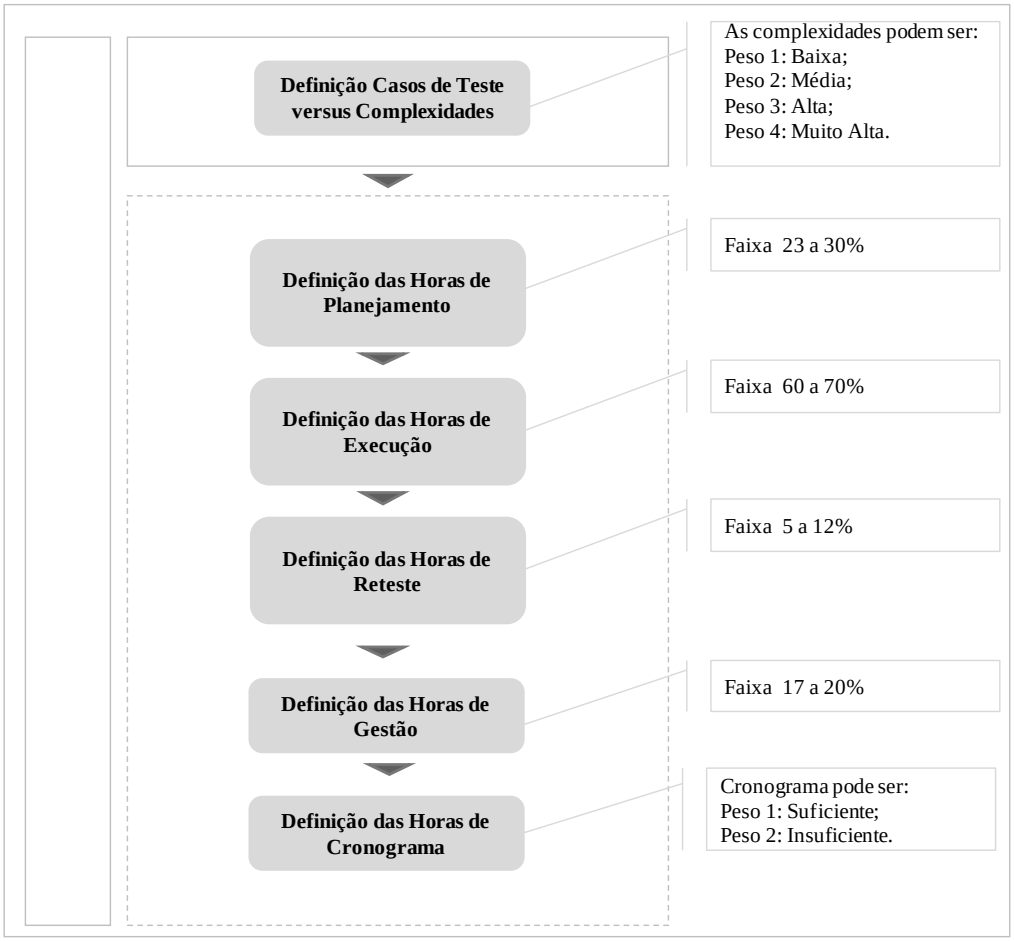

Figura 4. SET – Simple Estimate Test

Fonte: Os autores (2014)

A opção pela faixa de valores em detrimento de valores fixos, fez-se presente, tendo em vista que projetos de distintas complexidades demandam tempos distintos para a referida atividade.

Para estimar utilizando a técnica é desejável que o profissional tenha conhecimento do funcionamento do sistema. Observa-se no SET que a incorporação da razoabilidade do profissional no que concerne ao projeto e às especificidades do ambiente organizacional, além dos conhecimentos técnicos na definição e classificação dos casos testes, são requisitos determinantes para uma boa estimativa das complexidades indicadas no Quadro 2.

Quadro 2. Critérios de classificação das complexidades

Complexidade	Definição	Peso
Baixa	Pouca interação com o usuário com o programa, nem muito tempo de processamento para gerar um dado.	1
Média	Interação razoável do usuário com o programa, normalmente na mesma	2

	interface, processamento ligeiramente demorado e mais de uma verificação.	
Alta	Mais de uma interação distinta do usuário com o programa, normalmente em interfaces diferentes, o processamento e a validação dos resultados são demorados.	3
Muito Alta	Usuário precisa interagir bastante com o programa em mais de duas interfaces e validar um número grande (cinco ou mais) de resultados.	4

Fonte: Os autores (2013)

A partir da definição e classificação dos casos de testes, são calculadas as horas de planejamento, execução, reteste, gestão e cronograma, chegando a valor final de estimativa em horas.

Para alguns críticos esta técnica pode ser considerada de difícil utilização e interpretação, pois é preciso entender funcionalmente a complexidade do projeto para que se possa identificar dentro de cada uma das faixas o valor mais apropriado e assim chegar a um valor final em horas. Outro ponto importante é que a técnica proposta precisa ser aplicada em projetos empresariais reais (MATIELI, 2014).

4. CONCLUSÃO E SUGESTÕES FUTURAS

Durante o estudo, observou-se que existem algumas técnicas existentes na literatura, porém, nenhuma é dominante, melhor ou mais recomendável, por não existir, nem na literatura, nem na prática de grandes organizações, uma técnica dominante, melhor ou mais recomendável, na prática as empresas inventam os seus métodos, na maioria sem nenhum embasamento técnico-científico.

Nesse sentido, o presente trabalho oferece um panorama sumarizado das principais técnicas desenvolvidas para o enfrentamento das questões inerentes ao evidente empirismo nas práticas mercadológicas, representando um relevante insumo, tanto para pesquisadores, quanto para profissionais que atuam ou desejam atuar na area.

Como sugestões para estudos futuros, recomenda-se a seleção de algumas técnicas existentes na literatura e sua aplicação em projetos corporativos reais, para fazer uma avaliação comparativa no que concerne ao grau de acuracidade de cada teste para situações específicas.

REFERÊNCIAS

Aranha, E. & Borba, P. An estimation model for test execution effort, 2007. Disponível em: <http://www.academia.edu/8455490/An_Estimation_Model_for_Test_Execution_Effort>. Acessado em: 22 de janeiro 2015.

Bastos, A.; Rios, E.; Cristalli, R. & Moreira, T. Base de Conhecimento em Teste de Software. Martins, São Paulo, 2007.

Lazic, L. & Mastorakis N. (2008). Cost Effective Software Test Metrics. Wseas Transactions on Computers, Servia, v.7, n.6, p. 599-619.

Liou, J. (2010). On Software Test Estimate and Requirement Tracking. 19th International Conference on Software Engineering and Data Engineering 2010, USA, p. 57-62.

Lopes, F. A. & Nelson, M. A.V. Análise das Técnicas de Estimativas de Esforço para o Processo de Teste de Software, 2008. Disponível em: <http://ebts2008.cesar.org.br/artigos/EBTS2008-Analise_das_Tecnicas_Estimativas_de_Esforco.pdf>. Acessado em: 15 de fevereiro 2013.

Martins, J. C. C.. Técnicas Para Gerenciamento de Projetos de Software. Bransport, Rio de janeiro, 2007.

Matieli, L. V. Proposta metodológica para elaboração orçamentária de testes de software. Dissertação (Mestrado em Sistemas de Gestão). Niterói: Universidade Federal Fluminense, 2014. Orientador: Prof. Fernando Oliveira de Araujo.

Nageswaran S. Test Effort Estimation Using Use Case Points (UCP), 2001. Disponível em: <http://www.bfpug.com.br/Artigos/UCP/Nageswaran-Test_Effort_Estimation_Using_UCP.pdf>Acessado em 06/01/2015.

Mastorakis, N.; Mladenov, V. & Kontargyri, V. T.. Proceedings of the European Computing Conference. Springer, Grécia, 2009.

Nguyen, V.; Pham, V.; & Lam, V. (2013). qEstimation: A Process for Estimating Size and Effort of Software Testing. International Conference on Software and Systems Process, USA, p. 20-28.

Patel, N.; Govindrajan, M.; Maharana, S. & Ramdas, S. Test Case Point Analysis - Cognizant Technology Solutions, 2001. Disponível em: <www.stickyminds.com/getfile.asp?ot=XML&id=2566&fn=XUS373692file1.pdf >. Acessado em 06/01/2015.

Pressman, R. S. Software Enginnering: a practitioner's approach. McGraw-Hill, USA, 2005.

Souza, P. P. de; Barbosa, M. W. & Silva, A. R. da, Estimativa de Esforço de Teste no Auxílio da Garantia da Qualidade de Software, 2010 Disponível em: < http://www.spinsp.org.br/artigos/Revista_PBQP_2_Edi%C3%A7%C3%A3o.pdf>. Acessado em 07/01/2015.

Tronto, I. F. de B.; Silva, J. D. S. da; & Sant'anna, N. (2008). An Investigation of Artificial Neural Networks Based Prediction Systems in Software Project Management. The Journal of Systems and Software, v.81, p. 356–367.

Veenendaal, E.V. & Dekkers, T. Test point analysis: a method for test estimation. Project Control for Software Quality, Editors, Rob Kusters, Adrian Cowderoy, Fred Heemstra and Erik van Veenendaal. Shaker Publishing, 1999.

Zhou, B.; Okamura, H.; & Dohi T. (2013). Brief Contributions: Enhancing Performance of Random Testing through Markov Chain Monte Carlo Methods, *IEEE Transactions On Computers*, v. 62, n. 1, p.186-192.