Sistemas & Gestão 10 (2015), pp 180-193

SISTEMAS & GESTÃO
Revista Eletrônica

PROPOSTA METODOLÓGICA PARA ELABORAÇÃO ORÇAMENTÁRIA DE TESTES DE SOFTWARE

METHODOLOGICAL PROPOSAL FOR BUDGET PREPARATION IN SOFTWARE TESTING

Luciana Vago Matieli[a]; Fernando Oliveira de Araujo[a]

[a] **Universidade Federal Fluminense** (UFF) - Niterói, RJ, Brasil - Programa de Pós-Graduação em Sistemas de Gestão

Resumo

As metodologias empregadas no mercado referentes às práticas de precificação relacionadas à atividade de testes de software estão fortemente baseadas em empirismo, sem parâmetros ou critérios amplamente aceitos, o que não confere a devida credibilidade em termos de prazo e custo, tanto para empresas contratantes, quanto para terceiras. O presente estudo tem por objetivo propor uma metodologia de estimativa de teste de software, contribuindo para melhoria do referido processo. Em termos metodológicos, o estudo se ampara na literatura técnico-científica sobre o tema, além de entrevistas e consultas a especialistas técnicos (experts) de uma empresa terceira que prestam serviço a uma grande empresa brasileira do setor de telecomunicações, de modo a levantar teórica e empiricamente os principais parâmetros a serem considerados, bem como seus pesos relativos. Como conclusões, verifica-se a possibilidade de concepção do método proposto, representando contribuição relevante para o aumento da transparência dos serviços de teste e o estabelecimento de parâmetros mais fidedignos em termos de prazo e custo de projetos de teste de software.

Palavras-chave: teste de software; estimativa de teste de software; metodologia para estimativa de teste de software; qualidade de software.

Abstract

In the market, the methodologies relative of pricing practices related to the activity of software testing are strongly based on empiricism without parameters or widely accepted criteria, which does not give proper credibility in terms of time and cost for both contractors, as to third parties. This study aims to propose a methodology to estimate software testing, contributing to improvement of that process. In terms of methodology, the study relies on scientific and technical literature on the subject, interviews and consultations with technical experts of a third party, providing service to a large Brazilian company in the telecommunications industry in order to raise theoretically and empirically the main parameters to be considered, as well as their relative weights. As conclusions, there is the possibility of designing the proposed method, representing significant contribution to increasing the transparency of the testing and the establishment of more reliable parameters in terms of time and cost of software testing services projects.

Keywords: *software testing; estimation of software testing; methodology for estimation of software testing; software quality.*

1. INTRODUÇÃO

Empresas brasileiras de telecomunicações, usualmente, contratam empresas terceiras para realizarem o desenvolvimento e o teste dos seus sistemas, em especial sistemas *customer relationship management* (CRM) e de faturamento. 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. Com isso, a empresa terceira precisa gerar uma documentação denominada de proposta técnica contendo as atividades de teste que serão realizadas com seus respectivos prazos e custos.

A validação da referida proposta técnica de teste pela empresa contratante pode gerar dúvidas e questionamentos em relação aos custos apresentados pela empresa terceira, causando desgaste nesta interface empresa contratante e empresa terceira.

PROPPI / LATEC
DOI: 10.7177/sg.2015.v10.n1.a15

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

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 (sistema online[1] ou sistema batch[2]) 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.

O teste é uma atividade fundamental para garantir a qualidade do software. Segundo Liou (2010), 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.

Para Zhou *et al.* (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.

Segundo Nguyen *et al.* (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 de teste. De acordo com relatórios da indústria, teste de software consome cerca de 10-5% do esforço total do projeto e, em alguns projetos, este número pode chegar a 50%.

Para Lazic *et* 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.

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, e o maior desafio é fazer com que a empresa contratante entenda e aprove os valores descritos na proposta técnica de teste.

Especificamente, o problema da pesquisa está relacionado com a falta de definição de requisitos de teste de software, padronização dos requisitos de teste, entre empresa contratante e terceira para que se consiga fazer uma estimativa de teste mais precisa, de forma clara e transparente.

O estudo tem como pano de fundo uma das maiores empresas brasileiras de telecomunicações (contratante) e uma das mais qualificadas empresas multinacionais no desenvolvimento e teste de software (contratada). Os questionários foram aplicados em 15 funcionários desta empresa global de consultoria de gestão, serviços de tecnologia e outsourcing; com cerca de 7.000 colaboradores, atendem localmente organizações de todos os ramos de atividade no Brasil.

Em particular, o presente trabalho se propõe a perseguir respostas para o seguinte problema: seria possível desenvolver uma proposta metodológica capaz de contribuir para elaboração orçamentária de teste de software com o objetivo de melhorar a interface empresa contratante e terceira?

Em termos de estrutura, esse trabalho está organizado em cinco seções, a saber: a seção 2 está dedicada a oferecer um levantamento da literatura técnico-científica no sentido de contribuir para subsidiar a análise do problema e o encaminhamento de solução. A metodologia empregada no estudo está descrita na seção 3. Os resultados do estudo são discutidos na seção 4 e a seção 5 apresenta as conclusões e sugestões de estudos futuros.

2. REVISÃO DA LITERATURA

Segundo Furtado (2002: 24):

"A tecnologia da informação (TI) pode ser definida como todo recurso tecnológico e computacional destinada à coleta, manipulação, armazenamento e processamento de dados e/ou informações dentro de uma organização. Alternativamente, pode-se dizer que a tecnologia da

[1] Sistemas Online, no contexto de um site, significa estar disponível para acesso imediato a uma página de Internet, em tempo real. No contexto de outro sistema de informação, significa estar em plena operação, de acordo com as funções desempenhadas nessa rede ou sistema.

[2] Sistemas Batch ou sistemas em lote foram os primeiros sistemas multi-programáveis a serem implementados, caracterizando-se por programas armazenados em disco ou fita, que uma vez iniciados, exigem pouca ou nenhuma interação do usuário, processando de forma sequencial e contínua até o fim do serviço, quando então é devolvido o resultado final do processamento.

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

informação é o uso de recursos computacionais para desenvolvimento de sistemas de informação. Seus componentes essenciais são software e hardware." (Furtado, 2002:24).

Para Stair *et* Reynolds (2006), um sistema de informação é um tipo especializado de sistema que possui uma série de elementos interrelacionados com o propósito de coletar, manipular, armazenar e disseminar dados e informação.

Para Abreu *et al.* (2009), o CRM funciona como uma plataforma de gerenciamento do relacionamento com o cliente e, como ferramenta de TI, é bastante útil para organização e armazenagem de dados dos clientes, para estreitar e facilitar o contato da empresa com o cliente, para melhorar a produtividade por meio da automação da área de suporte e, principalmente, para aumentar a retenção de clientes.

Segundo Abreu *et al.* (2009), um sistema de faturamento executa todas as atividades relacionadas ao faturamento, sendo responsável pelo gerenciamento e faturamento das contas telefônicas dos clientes, e possibilita interfaces com outros sistemas com o objetivo de compartilhar informações das contas e relatórios financeiros.

Normalmente, um projeto de software é iniciado porque um cliente tem um problema e está disposto a pagar para resolvê-lo (Horstmann, 2008).

Segundo Cangussu *et al.* (2001), os gerentes de projeto estão sempre enfrentando problemas em controlar as fases do processo de desenvolvimento de software (SDP). Dois grandes desafios de gestão são gerar uma previsão exata (tempo) de conclusão e o custo de um projeto específico, principalmente no que diz respeito ao processo de teste de software (STP), uma subfase do SDP. Muitas técnicas têm sido propostas para resolver estes problemas com sucesso apenas parcial, ou seja, os modelos são disponíveis para fazer previsões iniciais ou para responder perguntas. Entretanto, não se observa, na literatura, nenhum modelo largamente aceito e/ou reconhecido para o STP para corrigir desvios.

A importância dos testes de software para a garantia de qualidade não pode ser subestimada. A estimativa de um módulo é importante para minimizar os custos e melhorar a eficácia do processo de teste de software. Para o autor, observa-se crescimento na demanda por qualidade de software nos últimos anos. Como consequência, problemas relacionados com os testes estão se tornando cada vez mais cruciais (Gondra, 2008).

De acordo com Myers *et al.* (1979), encontrar erros nos programas é a intenção do teste. Para Zhu *et al.* (2008), a qualidade de software torna-se cada vez mais importante em mercados competitivos atuais. Nesse contexto, observa-se que o teste é uma atividade amplamente utilizada para garantir a qualidade. Para os referidos autores, muitas organizações já estão criando equipes independentes de teste de software para realizar essas atividades.

Conforme apontam Tronto *et al.* (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, uma superestimativa pode não atender ao projeto e gerar orçamentos não competitivos.

Para facilitar o entendimento dos métodos/ técnicas de estimativas, é necessário entender alguns conceitos como requisito, caso de uso, cenário, cenário de teste, caso de teste.

De acordo com Hirama (2012), um requisito é uma declaração de algo que precisa ser implementado no sistema. Um requisito funcional é uma característica ou função do software. Um requisito não funcional é uma restrição ou comportamento esperado que se aplica a todo o sistema.

Segundo Larman (2002), um caso de uso é uma coleção de cenários relacionados de sucesso e fracasso, que descrevem um ator usando um sistema como meio para atingir um objetivo.

Para Martins (2007), o cenário é utilizado para análises dos fluxos de um caso de uso. Como tal, ele é uma instância de um caso de uso ou um caminho percorrido numa execução de caso de uso. Para Larman (2002), um cenário de teste é uma sequência específica de ações e interações entre atores e o sistema; é também chamado de instâncias de casos de uso. É uma história particular de uso de um sistema ou um caminho através do caso de uso; por exemplo, o cenário de efetuar com sucesso a compra de itens em dinheiro, ou cenário de não consumar a compra de itens por causa da recusa de uma autorização de crédito.

Um caso de teste é um conjunto de entradas, condições de execução e um critério de sucesso/ falha (Pezzè *et* Young, 2008).

Segundo Lopes *et* Nelson (2008), "a maioria dos modelos de estimativa aplicados ao desenvolvimento visa também estimar o esforço de teste dado à importância dessa atividade". O Quadro 1 consolida alguns dos métodos/ técnicas de estimativa de teste mais adotados pelo mercado.

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

Quadro 1. Alguns métodos/técnicas de estimativa de teste

1. Estimativas baseadas na regra *Ad-hoc*
Regra: o esforço estimado é definido pela gerência.
Pontos Positivos: não foram observados pontos de destaque na literatura.
Pontos Negativos: o esforço de teste não está baseado em nenhuma regra e nenhum prazo definitivo, e a precisão da estimativa depende da experiência, objetividade e percepção de quem realiza a estimativa.
2. Estimativas baseadas na regra 40-20-40
Regra: a regra 40-20-40 deve ser utilizada apenas como uma diretriz da engenharia de software na distribuição do esforço de um projeto de desenvolvimento de software. Esta regra recomenda que 40% do esforço seja reservado para a análise e projeto, 20% do esforço seja reservado para a codificação e 40% do esforço seja reservado para o teste.
Pontos Positivos: a regra é simples de ser aplicada e, em alguns projetos de teste, foi verificado que o estimado para o esforço despendido em testes ficou bem próximo do realizado.
Pontos Negativos: não considera pontos importantes dos testes como a cobertura e a complexidade, 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.
3. Estimativas baseadas na Porcentagem do tempo para o Desenvolvimento
Regra: 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.
Pontos Positivos: com base na prática profissional de 10 anos na área de teste de software, é possível destacar que, após a contagem do número de linhas de código ou pontos de função, a estimativa de teste é gerada com maior agilidade, aplicando-se um percentual.
Pontos Negativos: com base na prática profissional de 10 anos na área de teste de software, é possível destacar que, nesta técnica, não é levada em consideração a natureza do teste, ambiente, complexidade dos casos de teste, entre outras variáveis.
4. Estimativas baseadas em Pontos por Função
Regra: 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.
Pontos Positivos: pode gerar estimativas precisas desde que a análise de pontos de função também seja precisa.
Pontos Negativos: por ter um crescimento polinomial, em que número de casos de teste é igual a (Ponto de Função) elevado a 1.2, a estimativa de Capers Jones apresenta esforços diferentes dependendo da consideração feita. 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.
5. Estimativas a partir de Bases Históricas de Projetos de Teste
Regra: 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.
Pontos Positivos: histórico de projetos dentro da organização, consultas de projetos análogos para realizar estimativas mais consistentes, gerar dados estatísticos.

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

Pontos Negativos: 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.
6. Estimativas baseadas em Pontos por Casos de Teste (TCP – Test Case Points)
Regra: 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.
Pontos Positivos: uma vantagem observada nesta técnica é a ênfase dada à automatização dos testes, e a flexibilidade de estimar testes de software que sejam executados: só manualmente, só automatizados ou manual e automatizado.
Pontos Negativos: a técnica não estabelece nenhum critério para transformar os pontos obtidos por caso de teste em esforço de testes.
7. Estimativas baseadas em Análise de Pontos de Testes (APT)
Regra: 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.
Pontos Positivos: 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.
Pontos Negativos: a 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.
8. Estimativas baseadas em Pontos por Caso de Uso (UCP)
Regra: 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 Positivos: pontos por Caso de Uso – 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.
Pontos Negativos: 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.

Fonte: Elaborado a partir de Lopes *et* Nelson (2008)

O grande desafio de uma empresa contratante é escolher uma técnica de estimativa que atenda as suas necessidades e, principalmente, que consiga ter resultados confiáveis e transparentes. A técnica escolhida gera estimativas específicas que são importantes para mensurar todo esforço e custo que a etapa de teste de software demanda.

3. METODOLOGIA

A pesquisa está baseada em fontes primárias (investigação direta), e foram desenvolvidas as seguintes etapas:

- Definição da amostra a ser pesquisada: a amostra pesquisada constituiu-se de 15 funcionários com, no mínimo, 5 anos de experiência na área de telecomunicações e na área de teste de software; e, no mínimo, 3 anos de experiência na geração de orçamentos de teste de software. Estes funcionários trabalham em uma empresa que presta serviços de análise/desenvolvimento/teste de software para uma grande empresa brasileira de telecomunicações com mais de 7.000 funcionários no Brasil, que participaram de importantes e relevantes projetos

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

de telecomunicações, como por exemplo: Nono Dígito, Portabilidade, Consolidação SAP.

- Elaboração do Questionário de Levantamento de uma Proposta Orçamentária com Funcionários da Área de Teste de Software: para o desenvolvimento da etapa de coleta de dados primários, foi elaborado o questionário com apenas uma pergunta discursiva (aberta) que teve como objetivo capturar de um respondente qualificado sua perspectiva relacionada ao processo de desenvolvimento de proposta orçamentária, conforme apresentado no Apêndice I.
- Análise dos Dados: os dados do Questionário de Levantamento de uma Proposta Orçamentária com Funcionários da Área de Teste de Software, por conter uma pergunta aberta, podem gerar várias respostas, e as respostas podem ser totalmente diferentes. Para esta situação, foi feita uma análise semântica dos dados visando à sistematização e consolidação dos resultados:

- Os respondentes #1, #2, #4, #6, #7, #8 e #11 consideraram, no seu modelo massa de dados, complexidade versus tempo, execução, planejamento, gestão, reteste, e não consideraram mudança de tempo e escopo.

- Os respondentes #10 e #12 consideraram, no seu modelo complexidade versus tempo, execução, planejamento, gestão, reteste, e não consideraram massa de dados e mudança de tempo e escopo.

- O respondente #5 considerou, no seu modelo massa de dados, complexidade versus tempo, execução, planejamento, gestão, mudança de tempo e escopo, e não considerou reteste.

- O respondente #9 considerou, no seu modelo complexidade versus tempo, execução, planejamento, gestão, reteste, mudança de tempo e escopo, e não considerou massa de dados.

- O respondente #3 considerou, no seu modelo massa de dados, execução, planejamento, gestão, mudança de tempo e escopo, e não considerou complexidade versus tempo e nem reteste.

- O respondente #13 considerou, no seu modelo massa de dados, execução, planejamento, gestão, reteste, e não considerou complexidade versus tempo e nem mudança de tempo e escopo.

- Com um apoio de uma planilha, foi possível mapear as variáveis que mais apareceram nas respostas dos funcionários e fariam parte do modelo proposto. As variáveis eleitas ao modelo foram: cenário de teste e complexidade, massa de dados, planejamento de teste, execução de teste, gestão de teste, reteste e mudança de tempo e escopo.

- Elaboração do Questionário de Levantamento dos Percentuais: Durante a definição do modelo, surgiram dúvidas com relação aos percentuais das horas de planejamento, reteste e gestão, sendo necessário retornar aos especialistas para verificar quais seriam os percentuais atribuídos pelos respondentes. O questionário de levantamento dos percentuais para a validação do modelo, apresentado no Apêndice I, foi proposto com essa finalidade.
- Período da investigação em Campo: Entre 15 de maio a 15 de julho de 2014 e, dentre os 15 profissionais que compunham a amostra desejável da pesquisa, foram obtidas 13 respostas.

A Figura 1 tem como objetivo de ilustrar a lógica utilizada para a aplicação da respectiva metodologia.

4. RESULTADOS DA PESQUISA

Conforme sinalizado, dentre os 15 especialistas, 13 responderam o questionário de levantamento de uma proposta orçamentária com funcionários da área de teste de software, que continha uma pergunta aberta.

4.1 MODELO PROPOSTO

O modelo proposto contempla os seguintes elementos: horas para as atividades de execução e, na execução, temos as variáveis do modelo, complexidade, massa de dados, planejamento, gestão, além das horas de reteste e cronograma.

As variáveis, execução, planejamento, gestão e reteste foram calculadas considerando os seguintes passos:

1. Considerar as respostas dos Respondentes;
2. Ordenar as respostas dos Respondentes;
3. Calcular a amplitude (R) entre a menor e a maior resposta;
4. Calcular o número de intervalos de classe (NIC);
5. Calcular os Intervalos de Classe (IC), conforme descrito na memória de cálculo das horas de execução de teste.

Para o cálculo das demais variáveis, o procedimento adotado é análogo.

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

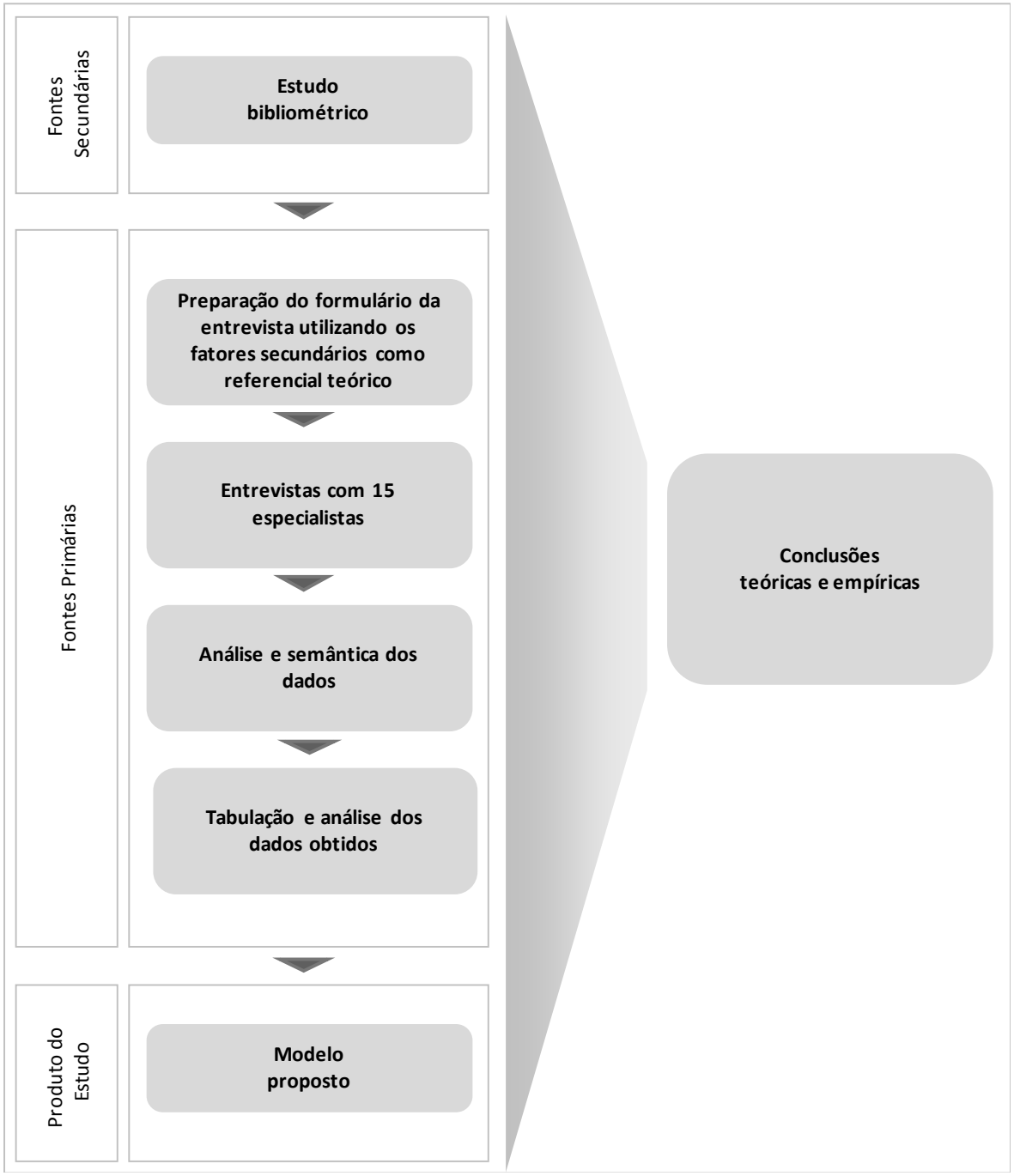

Figura 1. Metodologia aplicada no estudo

Fonte: Os autores (2014)

4.1.1 Execução do Teste

As horas de execução estão contempladas no modelo e foram calculadas a partir das respostas dos especialistas do questionário de levantamento dos percentuais, conforme ilustrado na Tabela 1.

A resposta do gerente, especialista #13, não está considerada na amostra original, pois já está contemplando faixas de valores. Para a execução, a resposta dada foi de 30 a 50% de acordo com a complexidade do projeto de teste.

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

Tabela 1. Memória de cálculo da execução de teste

Execução de Teste	Amostra Original	Respondentes (n)	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
		Respostas (R)	55	50	60	50	75	80	60	60	74	65	70	62
	Amostra Ordenada	Respondentes (n)	#2	#4	#1	#3	#7	#8	#12	#10	#11	#9	#5	#6
		Respostas (R)	50	50	55	60	60	60	62	65	70	74	75	80

Memória de Cálculo

1. Calculando a amplitude (R) entre a menor e a maior resposta:

   Equação 1 - Amplitude (R) da Execução de Teste

   R = Xmax – Xmin

   R = 30

2. Calculando o número de intervalos de classe:

   Equação 2 - Número de Intervalos de Classe (NIC) da Execução de Teste

   Nº de Intervalos de Classe (NIC) = n^0.5

   Nº de Intervalos de Classe (NIC) = 3.464102 ~ 3

3. Calculando os Intervalos de Classe (IC):

   Equação 3 - Intervalo de Classes (IC) da Execução de Teste

   Tamanho do IC = R/NIC

   Tamanho do IC = 10

	IC	Ocorrências
IC1	60 [50-60[	3
IC2	70 [60-70[	5
IC3	80 [70-80]	4

Considerando a maior ocorrência entre os especialistas na faixa de 60 a 70% e na experiência da pesquisadora, assume-se para o modelo proposto a faixa de 60 a 70% do tempo para a execução, conforme ilustrado na Figura 2.

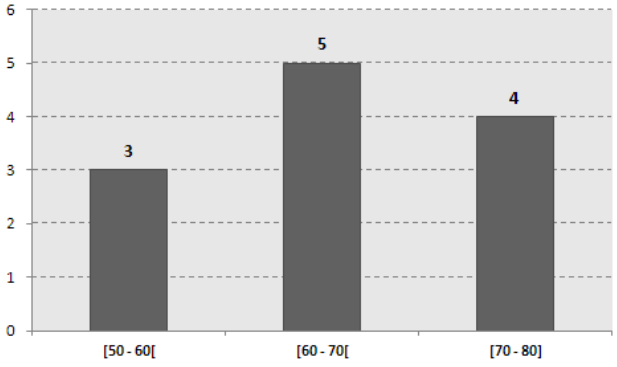

Figura 2. Faixas da execução de teste

Fonte: Os autores (2014)

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

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

4.1.2 Variáveis do Modelo

No modelo proposto, a variável do modelo será o caso de teste, ou seja, o menor "grão" a ser definido para posterior classificação da complexidade de cada um deles.

De acordo com Martins (2007), os casos de testes são elaborados a partir dos casos de uso. Os passos a serem seguidos são:

1. Para cada caso de uso, gerar uma lista de cenários possíveis;
2. Para cada cenário, identificar pelo menos um caso de teste e as condições válidas para "execução";
3. Para cada caso de teste, identificar os valores de entrada para o teste e o resultado esperado. A Figura 3 ilustra alguns exemplos de casos de teste:

Caso de teste: CT1

- *Cenário/condição: pedido colocado com sucesso.*
- *Entradas:*
  - *Código de cliente (V) e senha (V).*
  - *Código de produto (V) e quantidade (V).*
  - *Condição de pagamento (V).*
- *Resultado: pedido registrado com sucesso.*

Caso de teste: CT2

- *Cenário/condição: cliente inválido.*
- *Entradas:*
  - Código de cliente (I) e/ou senha (I).
- *Resultado: mensagem de erro de usuário/senha inválido.*

Figura 3. Exemplos de casos de teste

Fonte: Elaborado a partir de Martins (2007)

A Figura 4 ilustra a relação existente entre os conceitos de requisito, caso de uso, cenário de teste e caso de teste:

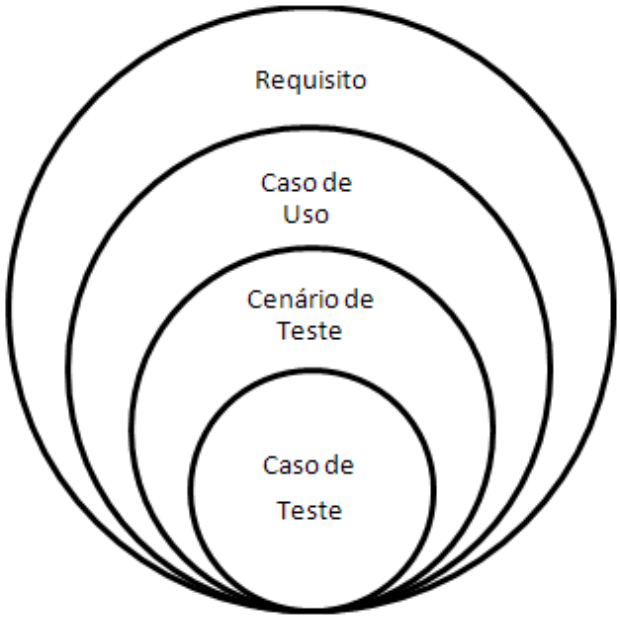

Figura 4. Conceitos de Teste de Software

Fonte: Os autores (2014)

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

Cabe ressaltar que este trabalho de definição e padronização de casos de teste para posterior classificação irá ajudar o usuário do método a escolher qual valor da faixa utilizar.

As massas de dados estão contempladas no modelo e são calculadas da mesma forma que a variável do modelo, caso de teste. Ou seja, primeiramente é preciso definir qual é o caso de teste referente à massa de dados, para posterior classificação quanto a sua complexidade.

4.1.3 Complexidade

Para classificação da complexidade do caso de teste, utilizam-se os critérios quantidade de processamentos e a quantidade de verificações a se fazer, conforme evidencia-se 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 interface, processamento ligeiramente demorado e mais de uma verificação.	2
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 (2014)

4.1.4 Planejamento

As horas de planejamento estão contempladas no modelo e foram calculadas a partir das respostas dos especialistas do questionário de levantamento dos percentuais.

A resposta do gerente, especialista #13, não está sendo considerada na amostra original, pois já está contemplando faixas de valores. Para o planejamento, a resposta dada foi de 15 a 30% de acordo com a complexidade do projeto de teste.

Considerando as maiores ocorrências entre os especialistas na faixa de 23 a 30% e na experiência da pesquisadora, assume-se que, para o modelo proposto, será preciso avaliar a sua complexidade do projeto de teste para a escolha da faixa do tempo para o planejamento.

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

4.1.5 Gestão

As horas de gestão estão contempladas no modelo e foram calculadas a partir das respostas dos especialistas do questionário de levantamento dos percentuais.

A resposta do gerente, especialista #13, não está sendo considerada na amostra original, pois já está contemplando faixas de valores. Para a gestão, a resposta dada foi de 20 a 40% de acordo com a complexidade do projeto de teste.

Considerando a maior ocorrência entre os especialistas na faixa de 17 a 20% e na experiência da pesquisadora, assume-se para o modelo proposto a faixa de 17 a 20% do tempo para a gestão.

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

4.1.6 Reteste

As horas de reteste estão contempladas no modelo e foram calculadas a partir das respostas dos especialistas do questionário de levantamento dos percentuais.

A resposta do analista de sistema sênior, especialista #3, não está sendo considerada na amostra original, pois ele não menciona a variável reteste no seu modelo.

A resposta do gerente, especialista #13, não está sendo considerada na amostra original, pois já está contemplando faixas de valores. Para o reteste, a resposta dada foi de 15 a 30% de acordo com a complexidade do projeto de teste.

Considerando a maior ocorrência entre os especialistas na faixa de 5 a 12% e na experiência da pesquisadora, assume-se para o modelo proposto a faixa de 5 a 12% do tempo para a gestão.

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

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

4.1.7 Cronograma

O cronograma é uma variável a mais a ser considerada no modelo, pois está relacionado ao tempo de execução do projeto de teste. O objetivo desta variável é incluir mais horas na estimativa caso o tempo de execução de todo o projeto de teste seja insuficiente de acordo com o cronograma proposto. Caso o cronograma proposto seja suficiente, não é feito o acréscimo de horas.

Conforme discutido previamente, as horas de execução estão contempladas no modelo e são calculadas multiplicando o total de casos de teste pela complexidade do caso de teste e multiplicando o total de casos de teste referente à massa de dados pela complexidade da massa de dados e posteriormente somando os dois resultados, conforme Equação 1:

$$EXE = (CT * x) + (CTM * c) \quad (1)$$

Em que,

EXE : horas de execução

CT : total de casos de teste

x : complexidade do caso de teste

CTM : total dos casos de teste referente à massa de dados

c : complexidade dos casos de teste referente à massa de dados

A título de entendimento da Equação 1, segue um exemplo ilustrado no Quadro 3.

Quadro 3. Cálculo das complexidades versus pesos

Caso de Teste	Complexidade	Quantidade de CT * Peso Complexidade
Caso de Teste 1	Baixa	1*1
Caso de Teste 2	Média	1*2
Caso de Teste 3	Média	1*2
Caso de Teste 4	Alta	1*3
Caso de Teste 5	Muito Alta	1*4
Total		12

Fonte: Os autores (2014)

Adicionalmente, após a definição dos casos de testes versus as suas complexidades, a equipe do projeto deve, considerando-se a razoabilidade da faixa proposta para o modelo, atribuir um valor apropriado para fins de cálculo, a depender da complexidade do projeto, conforme Equação 2:

$$EXE = \text{horas de execução equivale a x\% da faixa} \quad (2)$$

Em que,

EXE : horas de execução

Para as horas de planejamento, a equipe de projeto deve, também, considerar a razoabilidade da faixa proposta para o modelo e atribuir um valor apropriado para fins de cálculo, de acordo com a complexidade do projeto, conforme Equação 3:

$$PLN = \text{horas de planejamento equivale a x\% da faixa} \quad (3)$$

Em que,

PLN : *horas de planejamento*

Para as horas de gestão, a equipe de projeto deve, também, considerar a razoabilidade da faixa proposta para o modelo e atribuir um valor apropriado para fins de cálculo, de acordo com a complexidade do projeto, conforme Equação 4:

$$MGM = \text{horas de gestão equivale a x\% da faixa} \quad (4)$$

Em que,

MGM : *horas de gestão*

Para as horas de reteste, a equipe de projeto deve também, considerar a razoabilidade da faixa proposta para o modelo e atribuir um valor apropriado para fins de cálculo, de acordo com a complexidade do projeto. Este valor deve ser aplicado sobre o somatório das horas de execução, planejamento e gestão, pois em algumas situações o reteste é simplesmente reexecutar o cenário, mas em outras o reteste é gerado por conta de um cenário planejado incorretamente, mas para todas as situações a gestão sobre a re-execução se faz necessária. A fórmula do reteste ilustrada na Equação 5:

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

*RT = (EXE + PLN + MGM) * x% da faixa do reteste* (5)

Em que,

RT : *horas de reteste*

Para o cronograma suficiente, aplica-se o peso igual a 1 e, para o cronograma insuficiente, aplica-se o peso igual a 2, dobrando assim as horas do projeto de teste.

O modelo proposto, chamado de *Simple Estimate Test* (SET) apresenta o seguinte cálculo, conforme Equação 6:

*SET = ((EXE + PLN + MGM) + RT) * C* (6)

Em que,

SET : *simple estimate test*

EXE : horas de execução

PLN : horas de planejamento

MGM : horas de gestão

RT : horas de reteste

C : cronograma

5. CONCLUSÕES

O presente trabalho representa uma contribuição para elaboração orçamentária de teste de software com o objetivo de melhorar a interface empresa contratante e empresa terceira, pois ofereceu um método útil para definir, padronizar requisitos de teste, esforços e custos.

O método é útil, pois evidencia as regras para a realização de estimativas de teste de software, evitando questionamentos entre empresa contratante e terceira.

O modelo 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.

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 das vezes sem nenhum embasamento técnico-científico.

O estudo sugere que, com o desenvolvimento de uma proposta metodológica, é possível definir os casos de teste e casos de teste de massa de dados que são multiplicados por complexidades classificadas como baixa, média, alta e muito alta e, a partir deste resultado, é possível concluir se o teste é muito complexo, complexo, médio ou pouco complexo, e aplicar as faixas para cálculo do Simple Estimate Test (SET), gerando uma estimativa de teste mais precisa, de forma clara e transparente.

Cabe ressaltar que as premissas/variáveis que foram consideradas no modelo proposto são casos de teste referentes à massa de dados, complexidade, faixa de 60 a 70% para definição das horas de execução, faixa de 23 a 30% para a definição das horas de planejamento, faixa de 17 a 20% para a definição para horas de gestão, faixa de 5 a 12% para a definição das horas de reteste, e variável cronograma, que define se o tempo para a realização do projeto é suficiente ou não.

Além da equipe de teste de software, a equipe de gestão de projetos de software também pode se beneficiar do modelo proposto, pois, considerando uma base existente, fica mais simples de entender o custo do projeto de teste e, consequentemente, o custo de cada um dos projetos para justificar junto a uma diretoria.

Os principais pontos de aproximação entre os apontamentos da literatura e a experiência de especialistas no que concerne à estimativa de testes de software em termos de escopo são estimativas baseadas nas definições dos casos de uso e casos de testes. Em alguns métodos e/ou técnicas, fica claro que deve ser definido um requisito, um "grão", para que a estimativa seja gerada a partir dele.

Os principais pontos de afastamento entre os apontamentos da literatura e a experiência de especialistas no que concerne à estimativa de testes de software em termos de escopo são na literatura consultada. Não foram encontrados métodos ou técnicas que mencionassem os percentuais aplicados para cada macro atividade de teste de software como, por exemplo execução, planejamento e gestão.

Nesse sentido, o presente trabalho representa uma contribuição para o estreitamento dessa lacuna teórica evidenciada.

Como limitação do modelo proposto, destaca-se que não foi aplicado em um projeto real. Assim, recomenda-se que o modelo seja aplicado em projetos empresariais reais.

Como sugestão de pesquisas futuras, recomenda-se a aplicação do modelo apresentado em outras empresas do setor de telecomunicações, bem como a extrapolação do modelo para empresas de outros setores.

A avaliação do grau de satisfação dos funcionários das empresas contratante e terceira após a implantação do Simple Estimate Test (SET) seria também uma sugestão para pesquisas futuras, já que um dos propósitos do estudo foi propor algo capaz de contribuir na definição dos requisitos

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

de teste, padronização dos requisitos, e transparência dos valores estimados para posteriores consultas, para melhorar o relacionamento entre empresa contratante e terceira.

6. REFERÊNCIAS

Artigos de Periódicos online

Cangussu, J. W., Decarlo, R. A. e Mathur, A. P. (2001), "A State Model for the Software Test Process with Automated Parameter Identification, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics", Vol.2, pp. 706-711.

Gondra, I. (2008), "Applying Machine Learning to Software Fault-Proneness Prediction", *The Journal of Systems and Software*, Vol.81, pp.186–195.

Lazic, L. e Mastorakis N. (2008), "Cost Effective Software Test Metrics", *WSEAS Transactions on Computers*, Vol.7, No.6, pp. 599-619.

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

Nguyen, V., PHAM, V., e LAM, V. (2013), "qEstimation: A Process for Estimating Size and Effort of Software Testing", *International Conference on Software and Systems Process*, pp. 20-28.

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*, Vol.81, pp. 356–367.

Zhou, B., Okamura, H., e Dohi T. (2013), "Brief Contributions: Enhancing Performance of Random Testing through Markov Chain Monte Carlo Methods", *IEEE Transactions On Computers*, Vol. 62, No. 1, pp.186-192.

Zhu, X., Zhou, B., Hou, L., Chen, J. & Chen, L. (2008), "An Experience-Based Approach for Test Execution Effort Estimation", The 9th International Conference for Young Computer Scientists, China.

Livro

Furtado, V. (2002), Tecnologia e Gestão da Informação na Segurança Pública, 1 ed., Garamond, Fortaleza, CE.

Hirama, K. (2012), Engenharia de Software, Elsevier, São Paulo, SP.

Horstmann, C. (2008), Conceitos de computação com Java, 5 ed., Bookman, São Paulo, SP.

Larman, C. (2002), Utilizando UML e Padrões, 3 ed., Bookman, São Paulo, SP.

Martins, J. C. C. (2007), Técnicas Para Gerenciamento de Projetos de Software, 1 ed., Bransport, Rio de Janeiro, RJ.

Myers, G. J., Badgett, T. e Sandler, C. (1979), The Art of Software Testing, 3 ed., John Wiley & Sons, Inc, USA.

Pezzè, M. e Young, M. (2008), Teste e Análise de Software: Processos, Princípios e Técnicas, 1 ed., Bookman, São Paulo, SP.

Stair, R. M. e Reynolds, G. W. (2006), Princípios de Sistemas de Informação: uma abordagem gerencial. 2 ed.,Thomson, São Paulo, SP.

Simpósios e Congressos

Abreu, I. de M., Jones, G. D. C., Boas, A. A. V. e Ribeiro, K. C. de S. (2009), "Análise do Fluxo de Informação entre Sistemas de Cadastro e Faturamento: Um estudo de Caso de uma empresa de Telecomunicações", artigo apresentado no ENEGEP 2009: Encontro Nacional de Engenharia de Produção, Salvador, BA, 06 a 09 de outubro, 2009, disponível em: http://www.abepro.org.br/biblioteca/enegep2009_TN_STP_109_724_14626.pdf (Acesso em 15 de fevereiro de 2014).

Lopes, F. A. e Nelson, M. A. V. (2008), "Análise das Técnicas de Estimativas de Esforço para o Processo de Teste de Software", artigo apresentado no EBTS 2008: III Encontro Brasileiro de Teste de Software, Recipe, PE, 17 a 18 de outubro, 2008, disponível em: http://ebts2008.cesar.org.br/artigos/EBTS2008Analise_das_Tecnicas_Estimativas_de_Esforco.pdf. (Acesso em 15 de fevereiro de 2013).

Revista Eletrônica Sistemas & Gestão
Volume 10, Número 1, 2015, pp. 180-193
DOI: 10.7177/sg.2015.v10.n1.a15

APÊNDICE I

Questionário de Levantamento de uma Proposta Orçamentária com Funcionários da Área de Teste de Software

1. Pergunta Discursiva

Com base na sua experiência na área de Teste de Software, imagina que foi contratado pela sua empresa para criar uma metodologia capaz de gerar uma proposta orçamentária de teste. Imagina que possui alguns elementos como recursos, por exemplo, requisitos de teste, caso de uso, cenário de teste, caso de teste, complexidades, entre outros. Favor descrevê-la no campo abaixo.

Nota: Favor descrever uma proposta orçamentária de teste de software.

Questionário de Levantamento dos Percentuais para a Validação do Modelo.

1. Contextualização e Esclarecimentos

Prezado especialista, mais uma vez conto com a sua participação neste estudo. Pedimos a gentileza de que preencha individualmente o presente questionário, que não tomará mais do que 5 minutos do seu tempo.

Resguardamos a confidencialidade dos dados que terão finalidade acadêmica e desde já agradecemos a sua gentileza e nos colocamos à disposição para quaisquer dúvidas.

1. Caso o seu modelo calculasse as horas de execução percentualmente, qual seria o percentual proposto?

- Resposta:_____

2. Caso o seu modelo calculasse as horas de planejamento percentualmente, qual seria o percentual proposto?

- Resposta:_____

3. Caso o seu modelo calculasse as horas de gestão percentualmente, qual seria o percentual proposto?

- Resposta:_____

4. Caso o seu modelo calculasse as horas de reteste percentualmente, qual seria o percentual proposto?

- Resposta:_____