UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA WEB DE SUPORTE À ANÁLISE POR PONTOS DE TESTE

JOÃO RICARDO RODRIGUES

BLUMENAU
2010

2010/2-16

JOÃO RICARDO RODRIGUES

FERRAMENTA WEB DE SUPORTE À ANÁLISE POR PONTOS DE TESTE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Everaldo Artur Grahl - Orientador

BLUMENAU
2010

2010/1-16

FERRAMENTA WEB DE SUPORTE À ANÁLISE POR PONTOS DE TESTE

Por

JOÃO RICARDO RODRIGUES

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Everaldo Artur Grahl, Mestre – Orientador – FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB

Blumenau, 13 de dezembro de 2010

Dedico este trabalho aos meus pais, que estiveram sempre presentes, e aos meus amigos, em especial aos que me ajudaram com este trabalho e me apoiaram durante a realização do mesmo.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela vida e pelas oportunidades concedidas a mim durante ela, entre elas a de estar realizando este curso.

Agradeço também aos meus colegas da Senior Sistemas que colaboraram para a execução de um estudo de caso utilizando um de seus projetos. Obrigado a todos que me participaram respondendo perguntas, fornecendo dados e me auxiliando onde fosse preciso.

Agradeço aos colegas de faculdade, principalmente aos mais próximos, por tudo que passamos durante este curso.

Agradecimentos especiais à minha família e amigos, por todo o apoio dado a mim durante a realização deste trabalho. Obrigado pela força concedida, pelas cobranças feitas, pelas palavras de motivação nos meus momentos de desnorteio e também por toda a ajuda com as muitas dúvidas que iam surgindo.

Finalmente, agradeço também ao meu orientador Everaldo Artur Grahl que me forneceu ajuda, dicas e idéias e me orientou durante o desenvolvimento deste trabalho.

O insucesso é apenas uma oportunidade para começar de novo com mais inteligência.

Henry Ford

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta *web* de suporte à métrica de análise por pontos de teste. Esta métrica é usada para estimar o tempo exigido para a execução da fase de teste de um projeto de *software*. A ferramenta permite a estimativa de horas de teste exigidas por um projeto, e a emissão de relatórios e gráficos contendo os resultados obtidos. É mostrado também um estudo de caso aplicado em uma *software house* de Blumenau, para avaliar a precisão desta métrica.

Palavras-chave: Testes de *software*. Estimativas. Análise por pontos de teste.

ABSTRACT

This article presents the development of a web tool which supports test point analysis metrics. These metrics are used to estimate the time required for the execution of a project's test phase. The tool allows the estimation of the testing hours required by the project, and the emission of reports and charts containing these results. A case study was applied at a software house from Blumenau, to measure the metrics' precision.

Key-words: Software testing. Estimation. Test point analysis.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

BFPUG – *Brazilian Function Point Users Group*

CASE – *Computer Aided Software Engineering*

E – Efetividade e aderência

F - Funcionalidade

FPA – *Function Point Analysis*

FURB – Fundação Universidade Regional de Blumenau

GWT – Google *Web Toolkit*

HTP – Horas de Teste Primárias

HTT – Horas de Teste Totais

IBM – *International Business Machines*

IFPUG – *International Function Point Users Group*

MER – Modelo de Entidade e Relacionamento

MySQL – *My Structured Query Language*

P – Performance

PF – Pontos de Função

PTD – Pontos de Teste Dinâmicos

PTE – Pontos de Teste Estáticos

RF – Requisito Funcional

RNF – Requisito Não-Funcional

S - Segurança

TPA – *Test Point Analysis*

UCP – Use Case Points

SUMÁRIO

1 INTRODUÇÃO

Um projeto de software é algo que deve ser muito bem planejado e analisado. Uma das atitudes que podem ser tomadas a favor do planejamento de um projeto, para se ter uma melhor noção de como o mesmo deve ser gerenciado, é fazer estimativas e medições. A partir da análise de experiências passadas podem ser extraídos importantes dados e, com estes, ter uma base para melhor gerir o atual projeto. Dependendo da quantidade de projetos que estiverem no histórico de uma equipe podem ser levantadas também estatísticas em relação ao seu rendimento.

No entanto, não há a certeza de que os resultados obtidos estarão absolutamente corretos ou que tudo sairá como o planejado. O sucesso vai depender mais de quantas e quais foram as experiências tomadas como referência. As estimativas não podem prever mas podem deduzir, dando alguma certeza da probabilidade de determinado evento ocorrer no futuro, em função da medição de eventos passados semelhantes (MARCO, 1989, p. 7). O seu papel é sugerir, baseado em dados elencados, quanto tempo e esforço será necessário para a atividade analisada.

Entre as técnicas de estimativa que podem ser utilizadas em projetos de software estão os *Use Case Points* (UCP) e a *Function Point Analysis* (FPA). O UCP consiste em medir o tamanho e esforço de uma implementação baseado nos seus casos de uso. Vários fatores são levados em consideração, como o nível de complexidade dos atores e a quantidade de transações realizadas. Já a FPA serve para calcular o tamanho funcional de uma aplicação, o que também auxilia a estimar o custo e a duração de um projeto. Este cálculo é realizado baseando-se em aspectos como arquivos lógicos, entradas e saídas de um software (ALEXANDER, 2004). A *Test Point Analysis* (TPA) é uma técnica derivada da FPA com o objetivo de obter o tamanho em pontos de teste de uma aplicação e de suas funções (VEENENDAAL; DEKKERS, 1999, p. 4). Desta maneira é possível deduzir o tempo que deve ser dedicado à fase de testes do projeto baseado em algumas características suas e da equipe responsável pelo mesmo.

Considerando a pouca disponibilidade de ferramentas pesquisadas com suporte a TPA, viu-se a necessidade de se disponibilizar uma ferramenta dedicada a este método de estimativa.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho foi desenvolver um protótipo de ferramenta web para suporte a TPA. Como objetivo específico do trabalho, tem-se aplicar a teoria de TPA em um caso real de uma *software house*, a fim de avaliar sua precisão.

1.2 ESTRUTURA DO TRABALHO

O primeiro capítulo apresenta uma introdução ao trabalho e ao tema escolhido, mencionando as motivações para a escolha deste assunto. Também são vistos os objetivos do trabalho e a sua estrutura.

O segundo capítulo se trata da fundamentação teórica do trabalho. Nele são apresentados alguns conceitos sobre testes de *software*. Em seguida é explicado como funciona a análise por pontos de função, que é a base para o assunto seguinte. Este vem a ser a análise por pontos de teste, cuja parte do capítulo mostra onde ela pode ser usada, quais dados são necessários para o seu cálculo e como ele é realizado. No final deste capítulo são vistos trabalhos correlatos à proposta.

No terceiro capítulo é abordado todo o desenvolvimento do trabalho. São elencadas as ferramentas utilizadas na elaboração do protótipo e mostrados os diagramas de classes, casos de uso, seqüência e atividades, bem como o modelo de entidade e relacionamento (MER). São também demonstradas a aplicabilidade da técnica estudada e a operacionalidade do protótipo desenvolvido através de um estudo de caso onde são mostradas as telas da ferramenta.

O quarto capítulo traz as considerações finais a respeito do trabalho e relaciona sugestões para trabalhos futuros baseados neste tema.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é explicado o que são testes de *software*. Em seqüência é detalhada a métrica de Análise por Pontos de Função, sendo que esta é um pré-requisito para a estimativa por pontos de teste. A seguir são vistos conceitos de estimativas de testes utilizando TPA, seguida por uma explanação de como é feito o cálculo de pontos de teste através esta técnica. Finalmente, são analisados dois trabalhos correlatos referentes ao assunto.

2.1 TESTE DE SOFTWARE

O teste de *software* é um processo que faz parte do ciclo de desenvolvimento de um sistema e tem como objetivo avaliar a qualidade do *software* ou possibilitar melhorias revelando defeitos (PEZZÉ, YOUNG; 2008, p. 39). Nos primórdios do desenvolvimento de *software*, a atividade de testes era vista como a simples tarefa de navegar pelo código corrigindo problemas e era executada pelos próprios desenvolvedores. A partir de 1957, o conceito de testes de *software* conseguiu ampliar seus valores e se tornou um processo de detecção de erros, mas ainda era encarado como um processo que ocorria apenas no final do processo de desenvolvimento (BARTIÉ, 2002, p. 3).

Atualmente os processos de testes são compreendidos como um conjunto completo de atividades interdependentes que têm por objetivo a qualidade do *software*, ocorrendo ao longo do desenvolvimento e da evolução dos sistemas de software, desde o início da engenharia de requisitos até a entrega e subseqüente evolução (PEZZÉ, YOUNG; 2008, p. 63).

> "Embora freqüentemente se ouça falar de uma 'fase' de teste em desenvolvimento de software, como se teste fosse uma atividade distinta que ocorre em um ponto particular do desenvolvimento, não se deve confundir essa etapa intensa de testes com o processo de teste e análise como um todo, assim como não se deve confundir a compilação do programa com a programação. (PEZZÉ, YOUNG; 2008, p. 63).

Pode-se perceber que, no que diz respeito a desenvolvimento de sistemas, as organizações estão aumentando sua dependência tecnológica de forma rápida e constante, o que se trata de uma tendência natural. Isso reflete na tecnologia buscada para a redução de custos, para a ampliação das formas de atuação da organização, para o desenvolvimento de sistemas cada vez mais complexos e para se manter relevante em um mercado de trabalho

cada vez mais competitivo. Neste cenário, todos os aspectos envolvidos no processo de desenvolvimento têm um nível crescente de complexidade. Com isso os riscos de falhas e mau funcionamento aumentam tanto quanto a complexidade do ambiente, tornando-se mais difícil produzir sistemas com um nível de qualidade desejável. Esta é uma das maiores justificativas para a implantação de um processo de garantia da qualidade de *software* como elemento fundamental na sobrevivência em um mercado cada vez mais exigente (BARTIÉ, 2002, p. 5).

Conforme mencionado anteriormente, o processo de testes tem o objetivo de encontrar erros no sistema que está sendo desenvolvido, de forma a corrigir problemas que possam existir no *software*. No entanto, esta atividade não é feita de forma equívoca ou não-categorizada. O teste de *software* é planejado de modo que diversos aspectos do sistema possam ser analisados. É fundamental que o levantamento dos testes seja feito de forma categorizada para que se aumente a eficiência da detecção do maior número possível de cenários de teste, ou seja, situações que possam ser testadas com a intenção de detectar erros (BARTIÉ, 2002, p. 112).

2.1.1 Planejamento e estimativas

Em projetos de software, uma prática que deve ser bem especificada é o seu planejamento. É no planejamento de projetos que é analisado como ele será executado. Planejamento é a distribuição racional no tempo dos recursos disponíveis para a realização de alguma atividade. Planejar é, de modo geral, decidir antecipadamente o que deve ser feito e quando deve ser feito, ou seja, é traçar uma linha de ação (ROSA, 1997, p. 20).

Para auxiliar no planejamento do projeto existem métricas que se baseiam em características específicas do sistema a ser desenvolvido para estimar uma quantidade de tempo para a duração de algumas fases do projeto. A idéia de que certos aspectos de um projeto devem ser mensurados durante seu planejamento é muito defendida. Tais medidas ajudam a assegurar que o que foi projetado seja executado conforme o definido, no tempo estipulado.

> Um projeto é controlado no sentido que é garantir um mínimo de surpresas durante seu desenvolvimento. O projeto mais bem controlado não é aquele que faz a melhor ou a maior parte do trabalho, mas aquele que melhor convive com as suas previsões. Quando existem desvios naquilo que o projeto deveria originalmente oferecer, tais desvios são secundários e *detectados no início*. (MARCO, 1989, p. 5, grifo do autor).

Por se encontrar entre as fases do projeto, uma que pode e certamente deve ser planejada é a etapa de testes. Bastos et al. (2007, p. 112) afirmam que "O primeiro passo para a execução de um teste bem-feito é o seu planejamento", mencionando que um documento que permite elaborar o planejamento de testes é o plano de teste, onde são definidos o seu nível de cobertura e sua abordagem.

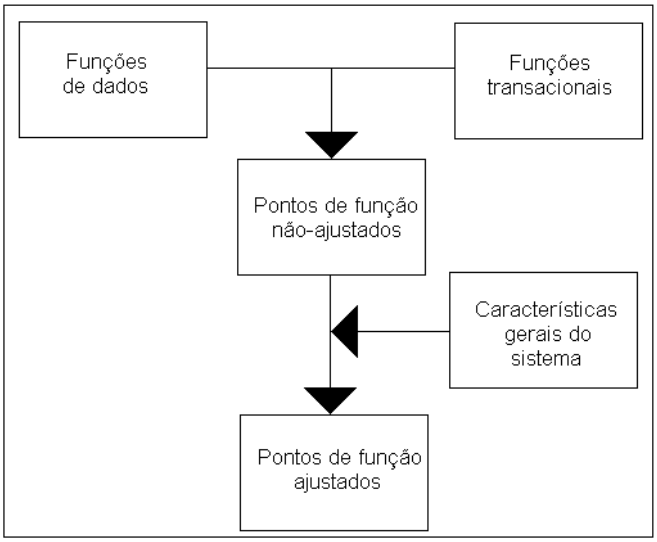
Para mensurar os testes pode ser utilizada a análise por pontos de teste, que é o tema principal deste trabalho. Além disso, a análise por pontos de função é uma técnica para medir o tamanho de um sistema e pode ser usada durante o planejamento da fase de desenvolvimento. Esta técnica é um pré-requisito para a análise por pontos de teste e será vista detalhadamente na próxima seção.

2.2 ANÁLISE POR PONTOS DE FUNÇÃO

A análise por pontos de função ou *Function Point Analysis* (FPA) é uma métrica proposta em 1979 por Allan Dalbrecht, pesquisador da IBM, sendo revista e ganhando uma nova versão em 1984. A entidade responsável pela definição das regras desta métrica é o *International Function Point Users Group* (IFPUG), tendo como representante brasileiro o *Brazilian Function Point Users Group* (BFPUG). Esta métrica é utilizada por diversas empresas de grande porte para dimensionar a funcionalidade dos seus sistemas de informação, podendo ser usada também no mercado como indicador de base para formação e preço e estimativas de prazo para projetos de software (SOUZA, 2006, p. 37). Provou-se, através do seu uso e aperfeiçoamento, que a análise por pontos de função é uma técnica valiosa e precisa para estimar o tamanho de um sistema, assim como documentar e enumerar suas capacidades (HELLER, 2010).

A obtenção dos pontos de função de um projeto se dá pelo cálculo de cinco funções principais, as quais se dividem em funções de dados e funções transacionais. Este primeiro grupo consiste em arquivos lógicos internos e os arquivos de interface externos. Já o outro

grupo contém entradas externas, saídas externas e consultas externas (ALEXANDER, 2004). Na figura 1 pode ser visto um esquema que representa todos os passos do cálculo de pontos de função. Estes passos são explicados nos textos a seguir.



Fonte: adaptado de Fatto Consultoria (2001).

Figura 1 - Cálculo de pontos de função

Arquivos lógicos internos representam os dados que são armazenados e alterados pela aplicação em questão. Arquivos de interface externos são dados que não são mantidos pela aplicação, mas que serão utilizados ou referenciados por ela. Entradas externas tratam-se dos processos que controlam dados e informações de fora da aplicação com a intenção de alterar o comportamento do sistema ou realizar manutenção de arquivos. Uma saída externa é um processo que envia dados ou controla informações com o objetivo de fornecer ao usuário dados obtidos através de operações matemáticas ou atualizações de arquivos, por exemplo. Já as consultas externas recuperam dados existentes em arquivos para fornecê-los ao usuário (ALEXANDER, 2004).

Cada um destes itens contribui para a obtenção dos pontos de função dependendo do seu nível de complexidade. Tomando as saídas externas como exemplo, é verificada a quantidade de tipos de arquivo referenciados pela aplicação em relação aos tipos de elementos de dados. Baseado nesta verificação, este item recebe uma qualificação que pode ser alta, média ou baixa complexidade. Uma aplicação com uma quantidade menor de tipos de arquivos referenciados tende a ter complexidade baixa. Por outro lado, aplicações com uma grande quantidade de tipos de arquivo referenciados e vários tipos de elementos de dados são classificadas como tendo complexidade alta (HELLER, 2000).

Este tipo de verificação é realizado para cada um dos itens mencionados. Quanto maior a complexidade de um item, mais pontos de função ele vale. A soma dos valores obtidos com

as cinco verificações equivale à quantidade de pontos de função não-ajustados (ALEXANDER, 2004).

Em seguida deve ser calculado o valor do fator de ajuste. Este valor é obtido através da avaliação de catorze características gerais do sistema. Estas devem ser avaliadas segundo sua influência; quanto maior o grau de influência, maior seu valor para o fator de ajuste. As características gerais do sistema são (SOUZA, 2006, p. 49):

a) comunicação de dados: representa o grau da comunicação direta da aplicação com o processador;
b) processamento de dados distribuído: representa o grau de transferência de dados entre os componentes da aplicação;
c) desempenho: representa a performance da aplicação, ou seja, o tempo de resposta;
d) utilização de equipamento: representa as restrições de recursos computacionais e as considerações do projeto para que a configuração do equipamento não fique sobrecarregada;
e) taxa de transações: representa a periodicidade das transações de negócio da que influenciam no desenvolvimento da aplicação;
f) entrada de dados *on-line*: representa a entrada de funções e dados de controle da aplicação disponibilizada de forma *on-line* ou interativa;
g) eficiência do usuário final: representa as considerações de fatores humanos, usabilidade e facilidade de uso da aplicação;
h) atualização *on-line*: representa a quantidade de arquivos lógicos internos que são atualizados *on-line*;
i) complexidade de processamento: representa a quantidade de processamentos lógicos e matemáticos executados pela aplicação;
j) reusabilidade: representa a capacidade de reutilização do código da aplicação em outras aplicações;
k) facilidade de instalação: representa a conversão de ambientes anteriores e a facilidade de instalação da aplicação e implantação de dados;
l) facilidade operacional: representa a presença de aspectos como inicialização, recuperação e *backup*;
m) múltiplos locais: representa a capacidade da aplicação de ser utilizada em múltiplos locais e organizações;
n) facilidade de mudanças: representa o grau de desenvolvimento da aplicação em relação a facilidades de modificação da lógica de processamento e estrutura de

dados, considerando também a capacidade de manipulação de consultas de diferentes complexidades.

Após a soma do nível da influência de todas as características gerais do sistema, é obtido o valor final do ajuste através da fórmula mostrada no quadro 1. A multiplicação deste valor pelos pontos de função não ajustados resulta na quantidade de pontos de função ajustados, conforme indica o quadro 2 (SOUZA, 2006, p. 50).

VFA = (NIT x 0,01) + 0,65
onde: VFA = valor final do ajuste NIT = nível de influência total

Fonte: adaptado de Souza (2006, p. 50).

Quadro 1 – Fórmula para obtenção do valor final do ajuste

PFA = PFNA * VFA
onde: PFA = pontos de função ajustados PFNA = pontos de função não ajustados VFA = valor final do ajuste

Fonte: adaptado de Souza (2006, p. 51).

Quadro 2 – Fórmula para obtenção da quantidade de pontos de função ajustados

2.3 ANÁLISE POR PONTOS DE TESTE

A TPA é uma técnica para estimar o tempo exigido para a fase de testes de um sistema ou projeto. Esta técnica pode ser utilizada para preparar uma estimativa para a realização de testes de sistema[1] e testes de aceitação[2], cobrindo testes de caixa-preta[3]. Os testes de caixa-branca já são analisados através da FPA (VEENENDAAL; DEKKERS, 1999, p. 1), cujo valor é uma pré-condição para o cálculo dos pontos de teste. Ressalta-se que toda técnica de medição e estimativa precisa considerar o ambiente ou local onde está sendo usada. Sempre é necessário adequar o modelo ao ambiente onde ele está sendo usado pela primeira vez até que, com base num histórico de informações coletadas, seja possível obter resultados cada vez

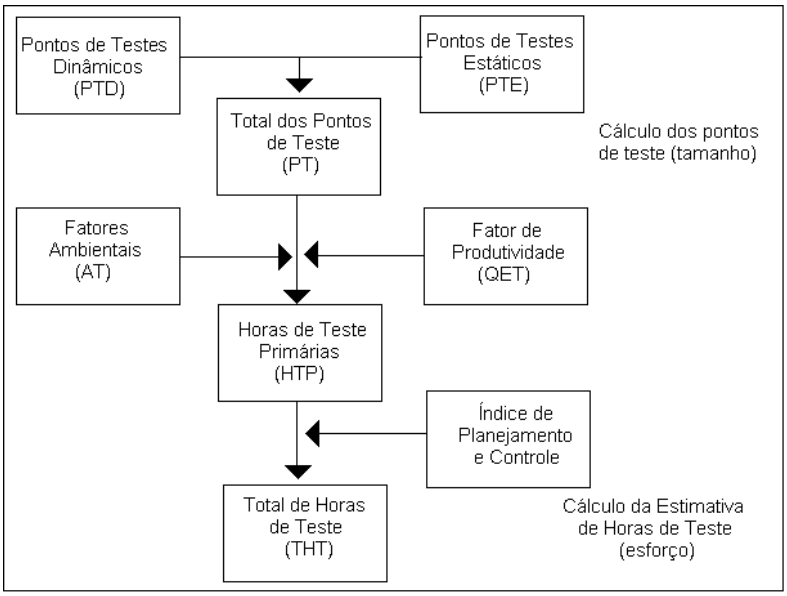
[1] Teste de sistema é o teste que verifica a completude do produto em relação a uma especificação de requisitos (PEZZÉ, YOUNG; 2008, p. 444).

[2] O teste de aceitação verifica a adequação de um sistema às necessidades do usuário, bem como a sua utilidade (PEZZÉ, YOUNG; 2008, p. 444).

[3] Testes de caixa-preta são orientados a entrada e saída. Estes testes consistem em fornecer valores de entrada, analisar os valores de saída e baseando-se em critérios de sucesso e falha avaliar o funcionamento do sistema sem analisar o seu código fonte (PEZZÉ; YOUNG, 2008, p. 172).

mais precisos (BASTOS et al., 2007, p. 227).

Uma visão geral do cálculo dos pontos de teste pode ser visto na figura 2, que descreve os passos nos quais o cálculo dos pontos e horas de teste se divide. Estes passos são explicados em detalhes nos textos a seguir.



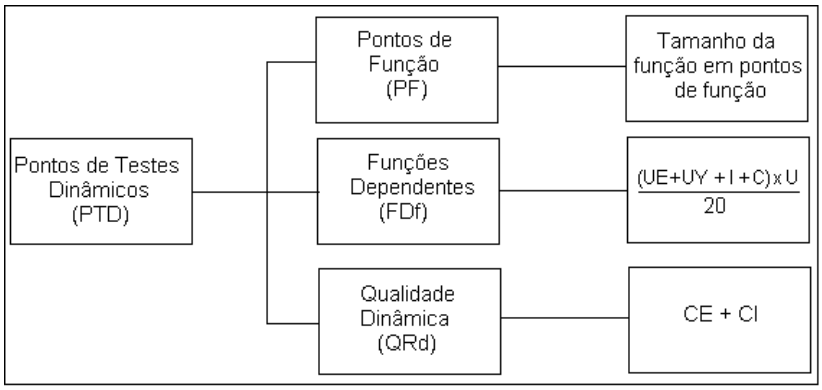
Fonte: adaptado de Bastos et al. (2007, p. 230).

Figura 2 - Cálculo de pontos de teste

Para calcular o tamanho do processo de teste deve ser encontrado o total de pontos de teste. Este valor é subdividido entre dois valores: os pontos de testes dinâmicos (calculados com base nas funções do sistema, conforme tratadas pela análise de pontos de função) e os pontos de testes estáticos (que levam em consideração o sistema como um todo, ao contrário dos pontos de testes dinâmicos que consideram cada uma das características isoladamente) (BASTOS et al., 2007, p. 231).

2.3.1 Pontos de testes dinâmicos

Os pontos de testes dinâmicos podem ser calculados através do esquema mostrado na figura 3. Este esquema divide o cálculo de pontos de testes dinâmicos em três itens – pontos de função, funções dependentes e qualidade dinâmica – sendo que estes itens são explicados detalhadamente nos textos a seguir.



Fonte: adaptado de Bastos et al. (2007, p. 232).

Figura 3 - Cálculo de pontos de testes dinâmicos

Primeiramente é preciso o valor dos pontos de função do processo, que pode ser obtido através da técnica de FPA vista na seção anterior. Isto porque cada funcionalidade medida em pontos de função tem uma contrapartida na medição dos pontos de teste. As funções utilizadas na análise por pontos de função que também são tratadas em pontos de teste são as entradas externas, as saídas externas e as consultas externas (BASTOS et al., 2007, p. 231).

Em seguida deve ser calculado o valor das funções dependentes, que é constituído pelos fatores a seguir. Estes devem ser avaliados e, para cada um deles, deve ser selecionada uma única qualificação (importância alta, média ou baixa, por exemplo), sendo que cada uma dessas contribui para o cálculo com um valor específico, conforme tabelas a seguir. Caso não haja certeza sobre a qualificação a ser selecionada, existe um valor padrão destacado em negrito na relação a seguir que deve ser escolhido (VEENENDAAL; DEKKERS, 1999, p. 5). Os fatores a serem avaliados e suas respectivas qualificações são:

a) importância do usuário (do inglês *User importancE* - UE): mede o grau de importância que o usuário dá à função a ser testada de acordo com o quadro 3;

V	Quali
3	Baixa
6	**Nor**
1	Alta

Fonte: adaptado de Bastos et al. (2007, p. 233).

Quadro 3 – Valores e qualificações da importância do usuário

b) intensidade de uso (do inglês *Use intensitY* - UY): mede o nível de utilização da função por intervalo de tempo de acordo com o quadro 4;

V	Quali
2	Baixa
4	**Nor**
8	Alta

Fonte: adaptado de Bastos et al. (2007, p. 233).

Quadro 4 – Valores e qualificações da intensidade de uso

c) uniformidade (do inglês *Uniformity* - U): mede o nível de reutilização do material

de teste. Este valor varia entre 0,6 e 1,0. Normalmente o valor utilizado 1,0, exceto pelos casos onde o material de teste possa ser aproveitado de uma função para outra. Nestes casos, o valor escolhido será menor;

d) interface (do inglês *Interface* - I): mede o nível de inter-relacionamento entre os *data-sets* (arquivos) e as funções que estão sendo medidas de acordo com o quadro 5. Leva-se em consideração o número de *data-sets* afetados pela função medida e o número de funções que afetam um *data-set* específico.

Arq	Funções		
	1	2	A
1	B	B	N
2	B	N	A
A	N	A	A

Fonte: adaptado de Bastos et al. (2007, p. 234).

Quadro 5 – Relação entre arquivos e funções para qualificação da interface

O número de *data-sets* acessados por uma função já deve ser definido no cálculo dos seus pontos de função. Se nenhum arquivo for modificado pela função, sua interface é considerada baixa. Caso contrário, ela é classificada de acordo com o quadro 6:

V	Quali
2	Baixa
4	**Nor**
8	Alta

Fonte: adaptado de Bastos et al. (2007, p. 234).

Quadro 6 – Valores e qualificações da interface

e) complexidade (do inglês *Complexity* - C): mede a complexidade do algoritmo do programa ou da função a ser avaliada. Este item é medido pela quantidade de comandos de condição (como *if* e *case*) existentes no algoritmo de acordo com o quadro 7.

V	Qualificação
3	Baixa – até 5 condições
6	**Normal – entre 6 e 11**
1	Alta – mais de 11 condições

Fonte: adaptado de Bastos et al. (2007, p. 235).

Quadro 7 – Valores e qualificações da complexidade

Após a avaliação destes cinco fatores, o valor total das funções dependentes pode ser obtido através da fórmula mostrada no quadro 8.

FDf = [(Ue + Uy + I + C) / 20] x U

onde:

FDf = Valor das funções dependentes para a função avaliada
Ue = Valor da importância do usuário
Uy = Valor da intensidade de uso
I = Valor da interface
C = Valor da complexidade
U = Valor da uniformidade

Fonte: adaptado de Bastos et al. (2007, p. 235).

Quadro 8 – Fórmula para obtenção do valor das funções dependentes de uma função

As características da qualidade dinâmica medem como a qualidade dos requisitos pode afetar a qualidade dos testes. Para esta medição são consideradas medidas explícitas e medidas implícitas. As características explícitas são as seguintes:

a) funcionalidade: avalia se o que foi implementado é o que foi especificado pelo usuário, se o sistema faz o que deveria fazer e se não faz o que não deveria fazer (BASTOS et al., 2007, p. 238);
b) performance: verifica o tempo de resposta, confiabilidade e escalabilidade de um sistema operando sob uma carga de trabalho definida (MEIER et al., 2007, p. 15);
c) segurança: identifica vulnerabilidades de segurança e erros no sistema que dificilmente são descobertos durante as fases de *design* e implementação (MICHAEL, RADOSEVICH; 2009).
d) aderência e efetividade: diz respeito aos testes de alto nível, ou seja, testes de sistema e testes de aceitação. Seu foco costuma ser a parte visível de um objeto ou função (BASTOS et al., 2007, p. 238).

Cada um destes itens deve receber um valor de acordo com o quadro 9. Normalmente é escolhido o valor em negrito em caso de incerteza sobre a classificação de um item.

V	Qualificação
0	A qualidade dos requisitos não é importante para o resultado dos testes.
3	A qualidade dos requisitos não é importante, mas precisa ser considerada para o resultado dos testes.
4	**A qualidade dos requisitos tem importância média para o resultado**
5	A qualidade dos requisitos é muito importante.
6	A qualidade dos requisitos é extremamente importante.

Fonte: adaptado de Bastos et al. (2007, p. 237).

Quadro 9 – Valores e qualificações das características explícitas

Após classificar cada uma das características, é preciso multiplicar os valores definidos pelos pesos de cada característica (funcionalidade: peso 0,75, performance: peso 0,10, segurança: peso 0,05, efetividade e aderência: peso 0,10) e em seguida dividi-los por quatro, conforme a fórmula apresentada no quadro 10. Por exemplo, utilizando o item de funcionalidade a fórmula seria:

F = (V x 0,75) / 4

onde:

F = valor resultante para o item de funcionalidade

V = valor atribuído ao item de funcionalidade conforme o quadro

Fonte: adaptado de Bastos et al. (2007, p. 237).

Quadro 10 – Exemplo de fórmula para cálculo de uma característica explícita

O valor total as medidas explícitas é obtido pela soma dos quatro valores obtidos através da avaliação destas medidas, vide quadro 11.

CE = (F x a) + (P x b) + (S x c) + (A x d)

onde:

CE = valor total das características explícitas

F = valor atribuído ao item de funcionalidade

P = valor atribuído ao item de performance

S = valor atribuído ao item de segurança

A = valor atribuído ao item de aderência e efetividade

a = ajuste do valor de funcionalidade (peso e divisão por 4)

b = ajuste do valor de performance (peso e divisão por 4)

c = ajuste do valor de segurança (peso e divisão por 4)

d = ajuste do valor de aderência e efetividade (peso e divisão por 4)

Fonte: adaptado de Bastos et al. (2007, p. 237).

Quadro 11 – Fórmula para cálculo do valor total das características explícitas

As medidas implícitas são utilizadas quando há coleta de dados e/ou indicadores para futuras medições quanto à qualidade dos testes. Um teste de performance, por exemplo, pode ser medido através da coleta de dados para futuro processamento. Sempre que houver indicadores que possam ser utilizados para avaliar uma das quatro características explícitas citadas acima, pode existir uma característica implícita associada, que adiciona 0,02 ao valor. O valor total das características implícitas é encontrado através da fórmula mostrada no quadro 12. (BASTOS et al., 2007, p. 239)

CI = n x 0,02

onde:

CI = valor total das características implícitas

N = número de características com medições (0 a 4)

Fonte: adaptado de Bastos et al. (2007, p. 239).

Quadro 12 – Fórmula para cálculo das características implícitas

A soma das medidas explícitas e implícitas resulta no valor da qualidade dinâmica, conforme indica o quadro 13. Somando este valor à quantidade de pontos de função e à fórmula das funções dependentes (quadro 8) são obtidos os pontos de testes dinâmicos (quadro 14).

QD = CE + CI

onde:

QD = valor total da qualidade dinâmica

CE = valor total das características explícitas

CI = valor total das características implícitas

Fonte: adaptado de Bastos et al. (2007, p. 239).

Quadro 13 – Fórmula para cálculo da qualidade dinâmica

PTDf = PFf x FDf x QD

onde:

PTDf = pontos de teste dinâmicos da função testada
PFf = pontos de função da função testada
FDf = valor total das funções dependentes da função testada
QD = valor total da qualidade dinâmica

Fonte: adaptado de Bastos et al. (2007, p. 239).

Quadro 14 – Fórmula para cálculo dos pontos de testes dinâmicos de uma função

2.3.2 Pontos de testes estáticos

Os pontos de testes estáticos só devem ser considerados quando a equipe de teste adotar processos de revisão de documentação e de códigos usando *checklists*, caso contrário devem ser descartados e terão valor nulo (BASTOS et al., 2007, p. 239). Já se uma característica de qualidade (funcionalidade, performance, segurança e aderência) for testada através de um *checklist*, ela adiciona dezesseis pontos à contagem (quadro 15).

PTE = n x 16

onde:

PTE = pontos de testes estáticos
N = número de características que utilizam checklists (0 a 4)

Fonte: adaptado de Bastos et al. (2007, p. 240).

Quadro 15 – Fórmula para cálculo dos pontos de testes estáticos do sistema

Calculando o número de pontos de testes estáticos e dinâmicos, é possível obter a quantidade total de pontos de teste através da fórmula mostrada no quadro 16. Para o cálculo deve-se considerar que o sistema tem no mínimo quinhentos pontos de função. Caso o sistema a ser testado tenha menos pontos de função, o valor a ser usado para o cálculo deve ser quinhentos (BASTOS et al., 2007, p. 241).

PT = PTD + (PF x PTE) / 500

onde:

PT = total de pontos de teste
PTD = soma dos pontos de teste de todas as funções
PF = tamanho total do sistema em pontos de função
PTE = total dos pontos de teste estático

Fonte: adaptado de Bastos et al. (2007, p. 240).

Quadro 16 – Fórmula para cálculo do total de pontos de testes do sistema

2.3.3 Horas de teste primárias

Com a obtenção do total de pontos de teste, que representam o volume das atividades de testes primárias, é possível multiplicá-lo por um fator de produtividade e um fator de ambiente de teste para se chegar à quantidade de horas de teste primárias (VEENENDAAL; DEKKERS, 1999, p. 11).

O fator de produtividade representa a qualificação da equipe de teste baseado no histórico da mesma. Este valor vai de 0,7 a 2,0 e indica o número de horas necessárias para cada ponto de teste. Quanto mais qualificada for a equipe, menor será este valor (BASTOS et al., 2007, p. 242).

Os fatores ambientais seguem o mesmo padrão das funções dependentes vistas anteriormente. Os itens têm três qualificações, cada uma adicionando um valor específico para o cálculo, dentre as quais uma única deve ser escolhida. O valor de todos os itens verificados deve ser somado e depois dividido por 21. A diferença é que, ao invés de serem simplesmente classificados como de importância baixa, média ou alta, o significado de cada classificação varia de acordo com o fator que está sendo avaliado. Em caso de dúvida sobre qual valor melhor reflete a situação de um fator, deve ser usado o valor em negrito. A seguir são listados estes fatores, bem como as descrições de suas classificações e quantos pontos elas valem:

a) existência de ferramentas de teste automatizado[4], que deve ser classificada de acordo com o quadro 17;

V	Qualificação
1	Existe uma ferramenta de automação para as fases de especificação e execução dos testes.
2	**Existe uma ferramenta de automação para as fases de especificação ou para a fase de execução.**
4	Não existe ferramenta de automação de teste.

Fonte: adaptado de Bastos et al. (2007, p. 243).

Quadro 17 – Valores e qualificações da existência de ferramentas de teste automatizado

[4] Testes automatizados são os que o testador determina um ambiente onde os testes serão executados e em seguida inicia a série de testes, que consistem em uma seqüencia de ações executadas através de uma ferramenta (FOURNIER, 2009, p. 171).

b) existência de plano de testes de precedência[5], que deve ser classificada de acordo com o quadro 18;

V	Qualificação
2	Existe um plano para o teste precedente e a equipe está familiarizada com ele, assim como com os respectivos casos de teste e resultados de
4	**Existe um plano para o teste precedente.**
8	Não existe um plano para o teste precedente.

Fonte: adaptado de Bastos et al. (2007, p. 244).

Quadro 18 – Valores e qualificações da existência de plano de testes de precedência

c) existência de um padrão de documentação de teste, que deve ser classificada de acordo com o quadro 19;

V	Qualificação
3	Durante o desenvolvimento do sistema, são utilizados padrões de documentação e *templates*. Acontecem revisões periódicas da documentação
6	**Durante o desenvolvimento do sistema, são utilizados padrões de documentação e *templates*. Seu uso não é verificado de maneira**
1	A documentação não segue nenhum padrão nem *templates* são usados.

Fonte: adaptado de Bastos et al. (2007, p. 244).

Quadro 19 – Valores e qualificações da existência de um padrão de documentação de teste

d) geração da linguagem utilizada no ambiente de desenvolvimento, que deve ser classificada de acordo com o quadro 20;

V	Qualificação
2	O sistema foi desenvolvido com uma linguagem de quarta geração, integrada a um sistema de gerenciamento de banco de dados.
4	**O sistema foi desenvolvido com uma combinação de linguagens de terceira e quarta gerações.**
8	O sistema foi desenvolvido em linguagem de terceira geração (Cobol, Pascal, C++, Delphi, etc)

Fonte: adaptado de Bastos et al. (2007, p. 244).

Quadro 20 – Valores e qualificações da geração da linguagem utilizada

e) histórico de utilização do ambiente de testes, que deve ser classificado de acordo com o quadro 21;

V	Qualificação
1	O ambiente de teste já foi utilizado inúmeras vezes.
2	**O ambiente de teste é similar ao que já vinha sido usado**
4	O ambiente de teste é completamente novo e experimental.

Fonte: adaptado de Bastos et al. (2007, p. 244).

Quadro 21 – Valores e qualificações do histórico de utilização do ambiente de testes

[5] Testes de precedência são testes que, para serem executados, dependem da execução de testes anteriores. Para cada etapa do processo de teste, a atividade imediatamente anterior deve produzir bons resultados para que a atividade seguinte seja bem executada. A melhor maneira de conduzir este trabalho é a partir de um plano de teste (BASTOS et al., 2007, p. 243).

f) disponibilidade de *testware*[6], que deve ser classificada de acordo com o quadro 22.

V	Qualificação
1	Existem materiais de teste, tais como bases de dados, tabelas, casos de teste, e outros, que poderão ser reutilizados.
2	**Existem apenas tabelas e bases de dados disponíveis para**
4	Não existe *testware* disponível.

Fonte: adaptado de Bastos et al. (2007, p. 244).

Quadro 22 – Valores e qualificações da disponibilidade de testware

As horas de teste primárias são obtidas através da fórmula mostrada no quadro 23.

HTP = PT x QET x AT

onde:

HTP = horas de teste primárias
PT = pontos de teste
QET = qualificação da equipe de teste
AT = fatores de ambiente de teste (soma de todos os fatores dividida por 21)

Fonte: adaptado de Bastos et al. (2007, p. 245).

Quadro 23 – Fórmula para cálculo das horas de teste primárias

Por fim, o número total de horas de teste é obtido multiplicando-se as horas de teste primárias por um índice de planejamento e controle. Este índice é um percentual que representa estas duas atividades e seu valor varia de acordo com o tamanho da equipe e da disponibilidade de ferramentas de gerenciamento (BASTOS et al., 2007, p. 246). Os valores que podem ser acrescentados de acordo com o tamanho da equipe são mostrados no quadro 24 e os valores referentes às ferramentas de gerenciamento são mostrados no quadro 25. O valor em negrito deve ser utilizado em caso de incerteza sobre como qualificar o item.

V	Qualificação
0	A equipe de qualidade possui de 3 a 4 técnicos.
0	**A equipe de qualidade possui de 5 a 10 técnicos.**
0	A equipe de qualidade possui 11 ou mais técnicos.

Fonte: adaptado de Bastos et al. (2007, p. 246).

Quadro 24 – Valores e qualificações do tamanho da equipe

V	Qualificação
0	Existem ferramentas de registro de tempo, gerenciamento de testes, gerenciamento de defeitos e gerenciamento de configuração.
0	**Apenas uma dessas ferramentas está disponível.**
0	Não existem ferramentas disponíveis.

Fonte: adaptado de Bastos et al. (2007, p. 246).

Quadro 25 – Valores e qualificações das ferramentas de gerenciamento

No quadro 26 é mostrada a fórmula para obtenção deste índice e no quadro 27 está a fórmula para o cálculo das horas de teste totais.

[6] *Testware* é todo material utilizado para a realização de testes. O *testware* inclui os casos de teste, *scripts* de teste, dados para a realização dos testes, entre outros itens (RICE, 2010).

IPC = 1 + TE + FG

onde:

IPC = índice de planejamento e controle
TE = valor correspondente ao tamanho da equipe
FG = valor correspondente às ferramentas de gerenciamento

Fonte: adaptado de Bastos et al. (2007, p. 246).

Quadro 26 – Fórmula para obtenção do índice de planejamento e controle

HTT = HTP x IPC

onde:

HTT = horas de teste totais
HTP = horas de teste primárias
IPC = índice de planejamento e controle

Fonte: adaptado de Bastos et al. (2007, p. 245).

Quadro 27 – Fórmula para cálculo das horas de teste totais

Para a avaliação da precisão da técnica de análise por pontos de teste, foram consideradas outras fontes com pesquisas relacionadas a este assunto. Campos e Birnfeld (2010) afirmam que é importante "analisar e avaliar a TPA, perante a realidade do projeto no qual ela será aplicada, pois há pré-requisitos que caso não tenhamos, poderá ocasionar um mau uso de tal técnica e acabar resultando em mais problemas, do que soluções com a sua aplicação", também enfatizando que estimativas só têm maior valor se houverem bases históricas disponíveis que contribuam para ajustes que possam ser necessários no modelo. Um estudo feito com a técnica aplicando-a a seis projetos distintos chegou a resultados significantemente maiores que o tempo de testes realizados nestes projetos. A técnica estimou valores de mais de dez mil pontos de testes em dois destes projetos e obteve um esforço estimado superior a 350% mais alto em relação à base histórica (LOPES; NELSON, 2008, p. 4). Já Meisen, autor do Test Manager que foi estudado como trabalho correlato, utilizou esta ferramenta para aplicar a TPA a dois sistemas. O primeiro deles é um sistema da empresa onde trabalhava, considerando também apenas algumas funções para a análise. A métrica estimou 17 horas para o teste, enquanto o teste foi executado em 13 horas, ou seja, a métrica estimou um tempo de testes superior ao realizado (MEISEN, 2005, p. 67). A segunda aplicação da técnica por este autor foi realizada no Laboratório de Qualidade de *Software* da Universidade Regional de Blumenau, onde foi analisado um sistema que segundo a TPA necessitaria de 58,05 horas de teste. O processo de testes durou 64,44 horas para este sistema, mostrando que o valor estimado desta vez foi inferior ao realizado (MEISEN, 2005, p. 69). Estes últimos casos mostram como o tempo estimado pode oscilar tanto para menos quanto para mais horas em relação ao tempo efetivamente investido no processo de testes e também como estes resultados podem diferir das horas reais tanto em números pequenos como em grandes.

2.4 TRABALHOS CORRELATOS

Foram encontradas duas ferramentas que possuem uma idéia semelhante à deste trabalho. A primeira a ser analisada é o Test Manager, um TCC feito na FURB baseado em TPA (MEISEN, 2005). A segunda é o Testimation, uma aplicação web para a realização de cálculo de estimativas (HORNE; HARRIS; HORNE, 2006). Após a análise destas duas ferramentas é mostrado um quadro comparativo com as características das mesmas.

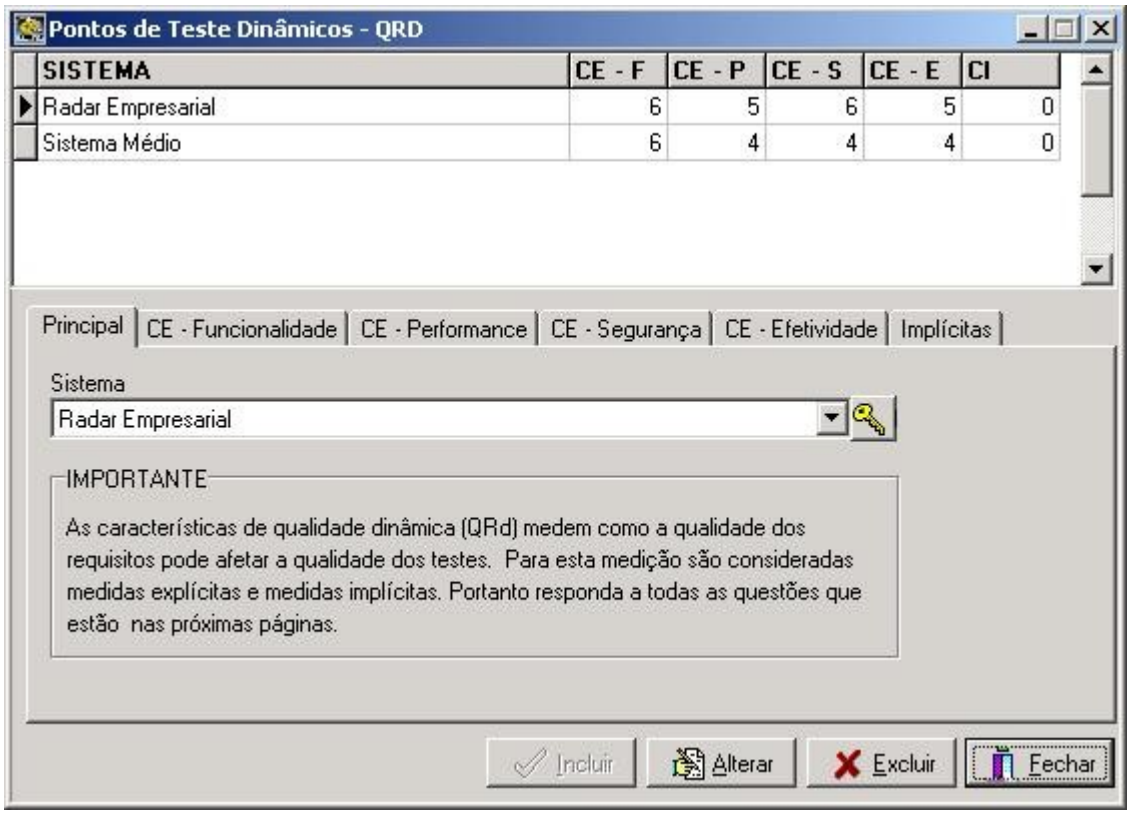
2.4.1 Test Manager

O Test Manager é uma ferramenta *desktop* para auxiliar na realização de estimativas quanto ao tempo necessário para execução do processo de testes, baseando-se na técnica de análise por pontos de teste (MEISEN, 2005, p. 13).

É possível cadastrar um sistema, seus módulos e funções, assim como os recursos a serem utilizados (pessoas que realizarão os testes, por exemplo). Ao cadastrar um módulo devem ser informados seus pontos de função e ao cadastrar uma função, os fatores para o cálculo dos pontos de testes dinâmicos, vistos na secção 2.3.1 deste trabalho.

Através de um menu é possível acessar telas específicas para cada um dos cálculos que compõem o FPA. Nestas telas devem ser informados os dados necessários que não são fornecidos no cadastro de projetos. Existe também uma opção específica para calcular e mostrar os resultados finais (pontos de teste obtidos e horas de teste estimadas), que podem ser visualizados em um relatório.

A figura 4 mostra a tela de cálculo do valor da qualidade dinâmica do Test Manager.



Fonte: Meisen (2005, p. 60).

Figura 4 – Tela de cálculo do valor da qualidade dinâmica do Test Manager

Nesta tela podem ser vistos os valores atuais das características de funcionalidade, performance, segurança e aderência e efetividade (aqui representada apenas como "efetividade") do sistema, bem como suas características implícitas. Os valores são obtidos através das questões presentes nas abas em que esta tela se divide, as quais devem ser respondidas pelo usuário.

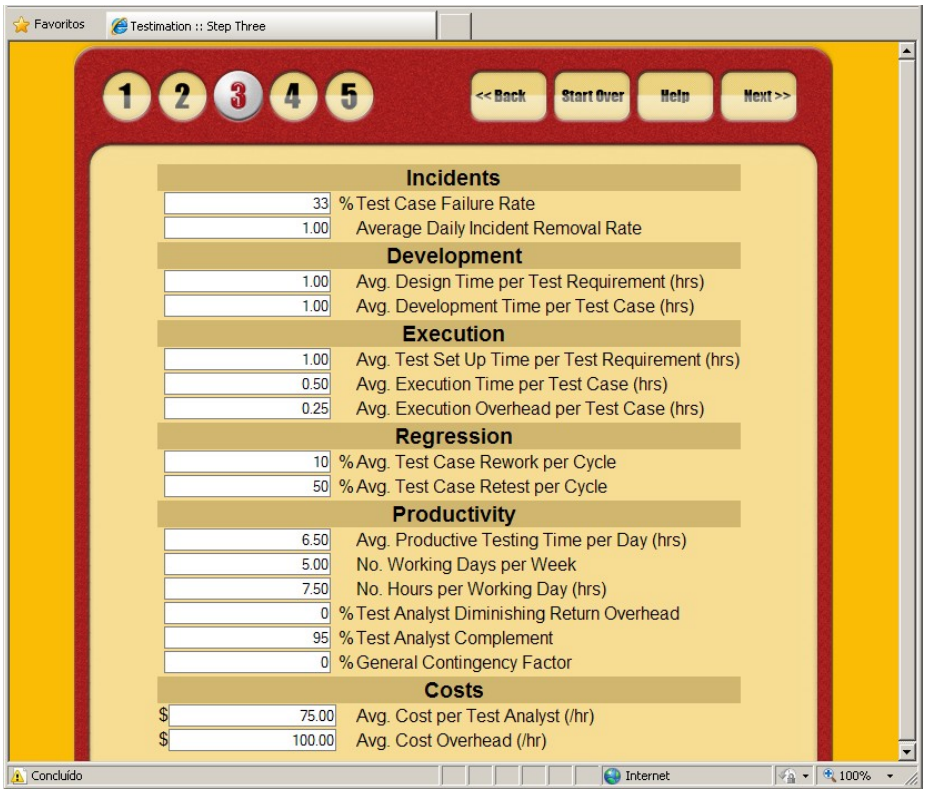
2.4.2 Testimation

O Testimation é uma ferramenta web que tem por objetivo receber dados de entrada e, com base nesses, fornecer uma estimativa de esforço para testes. No entanto, a técnica utilizada não é a TPA e sim uma métrica própria que baseia seus cálculos em informações diferentes das da TPA. Entre estas estão a quantidade de pessoas que irão trabalhar em cada função, o grau de experiência dessas, a data inicial da fase de testes do projeto e quantos dias por semana serão trabalhados no mesmo (HORNE; HARRIS; HORNE, 2006).

A ferramenta possui uma interface para entrada de dados dividida por páginas, simples e de fácil utilização. A primeira página é uma introdução ao *site*, onde podem ser preenchidos

o nome do projeto, *e-mail* do responsável e outros dados similares para o início da estimativa. Da segunda à quarta página são exibidos campos para que o usuário informe os valores exigidos para o cálculo. Na quinta e última página são informados os dados resultantes. Esses são a quantidade de horas estimadas para as fases de testes do projeto (*design*, desenvolvimento e execução de testes), o tempo que deve ser dedicado aos testes como um todo e uma data final para os mesmos, considerando a data inicial e quantidade de dias de trabalho informados pelo usuário. A ferramenta disponibiliza ainda a possibilidade de emissão de um relatório para impressão contendo estes resultados.

A figura 5 mostra uma das telas do Testimation onde o usuário deve fornecer dados. O seu cálculo se divide em cinco passos e esta tela representa o terceiro passo, conforme pode ser visto pela parte superior da figura. Os dados a serem fornecidos diferem dos exigidos pela TPA porque este sistema utiliza uma métrica diferente.



Fonte: Horne, Harris e Horne (2006).

Figura 5 – Tela de fornecimento de dados para análise do Testimation

No quadro 28 é apresentada uma comparação entre as ferramentas descritas quanto a suas as características e funcionalidades. Ambas as ferramentas realizam estimativas de teste, porém cada uma utiliza uma métrica diferente. Enquanto o Test Manager faz uso da TPA, Testimation usa um sistema diferente, elaborado pelos seus desenvolvedores. O Test Manager permite também o cadastro dos sistemas a serem avaliados, bem como as funções a serem avaliadas e as pessoas que irão executar os testes. Já o Testimation não salva estes cadastros;

as informações são fornecidas pelo usuário e o resultado da análise é mostrado logo em seguida, sem possibilidade de armazenamento dos dados informados. Os dois sistemas permitem que os resultados da estimativa possam ser visíveis em forma de relatório, o qual pode também ser impresso. O Testimation é uma aplicação *web* que deve ser acessada via *browser* de internet, enquanto o Test Manager é um aplicativo *desktop,* ou seja, não depende de um *browser* de internet para ser utilizado. Por fim, nenhum dos trabalhos estudados possibilita a emissão dos resultados em gráficos ou o cálculo de pontos de função, o qual é um pré-requisito para a análise por pontos de teste.

CARACTERÍSTICAS	TEST MANAGER (Autor: Fábio C. Meisen)	TESTIMATION (Autores: Horne, Harris e Horne)
realiza estimativas de teste	X	X
cálculo de FPA		
cálculo de TPA	X	
gerenciador de sistemas e seus dados	X	
aplicação web		X
resultados das estimativas visíveis em formato de relatório	X	X
resultados das estimativas visíveis em formato de gráfico		

Quadro 28 – Comparação entre Test Manager e Testimation

3 DESENVOLVIMENTO DA FERRAMENTA

Este capítulo relata aspectos referentes ao desenvolvimento da ferramenta. Estes incluem os seus requisitos, detalhes sobre a sua especificação e implementação, sendo feita também uma apresentação da ferramenta. Além disso será demonstrado o estudo de caso realizado para este trabalho e as conclusões obtidas a partir do mesmo.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A definição dos requisitos se baseou na proposta de implementar um protótipo de ferramenta que dê suporte à TPA com apoio a usuários que desconheçam esta métrica. A função da ferramenta é coletar dados informados pelo gerente de projeto a respeito do projeto que deve ser analisado. A partir destes dados deve então ser estimado o tempo necessário para a execução completa da fase de testes do projeto. Outro recurso que a ferramenta deve disponibilizar é um sistema de auxílio que, além de ter uma página de ajuda com informações específicas sobre a TPA, forneça informações para o usuário enquanto os dados são preenchidos. Estas informações irão explicar ao usuário que valores devem ser informados por ele de acordo com o campo selecionado no momento, detalhando também qual a influência daquele campo no valor final da estimativa.

Alguns requisitos foram baseados na análise dos trabalhos correlatos. Constatou-se que é interessante salvar os dados informados pelo usuário para que eles possam ser reutilizados no futuro sem a necessidade de serem informados novamente. O trabalho foi desenvolvido tendo em mente um ambiente *web* que utiliza a TPA porque o aplicativo verificado que utiliza a TPA é um aplicativo *desktop* e o aplicativo *web* analisado não utiliza esta métrica. Nenhum dos trabalhos correlatos apresenta os resultados em forma de gráfico, então este requisito foi adicionado como uma funcionalidade exclusiva da ferramenta desenvolvida neste trabalho.

O protótipo a ser desenvolvido deverá:

a) permitir ao usuário cadastrar seu projeto de software (Requisito Funcional - RF);
b) permitir ao usuário cadastrar as funções de um projeto (RF);
c) ter uma interface para que o usuário possa informar os dados exigidos para o cálculo dos pontos de testes dinâmicos (RF);

d) ter uma interface para que o usuário possa informar os dados exigidos para o cálculo dos pontos de testes estáticos (RF);
e) ter uma interface para que o usuário possa informar os dados exigidos para o cálculo da quantidade de horas de teste (RF);
f) calcular os pontos de teste do projeto baseado nos valores cadastrados no sistema e informados pelo usuário (RF);
g) converter o valor de pontos de teste calculado para quantidade de horas, segundo os valores informados pelo usuário (RF);
h) permitir que os resultados possam ser visualizados em forma de um gráfico, indicando detalhes sobre o resultado do cálculo – por exemplo, como cada uma das partes do cálculo (pontos de testes estáticos e dinâmicos, etc.) compõem o tempo total (RF);
i) permitir que os resultados possam ser visualizados em forma de arquivo de texto, indicando detalhes sobre o resultado do cálculo – por exemplo os resultados parciais de cada uma das etapas do cálculo total (quantidade de pontos de testes estáticos, funções dependentes, qualidade dinâmica, etc.) (RF);
j) permitir que sejam emitidos relatórios de projeto incluindo as informações cadastradas do mesmo, bem como uma lista de estatísticas (tempo de duração do projeto e relação entre horas de trabalho e pessoas alocadas, por exemplo) (RF);
k) permitir que os gráficos, arquivos de texto e relatórios emitidos sejam impressos (RF);
l) permitir ao usuário cadastrar um *username* e uma senha para poder efetuar *login* (RF);
m) ser desenvolvido na linguagem de programação Java (Requisito Não Funcional - RNF);
n) ser desenvolvido utilizando banco de dados MySQL (RNF);
o) realizar controle de autenticação de usuários. Por ser uma ferramenta web, o usuário que quiser utilizá-la deverá realizar *login* para poder acessar seus projetos cadastrados e fazer cálculos a partir deles (RNF);
p) mostrar uma pequena descrição (*hint*) ao lado dos campos a serem preenchidos, para informar o usuário do significado de tais campos, sua influência no cálculo e como eles devem ser preenchidos (RNF);
q) possuir uma documentação de ajuda ao usuário sobre a TPA, explicando com detalhes em que consiste o cálculo da mesma (RNF);

r) ser compatível com Internet Explorer (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção são apresentadas as atividades referentes a fase de especificação da ferramenta. Estas incluem os diagramas de casos de uso, de seqüência, de classes e um Modelo de Entidade e Relacionamento (MER). Estas atividades serão demonstradas com detalhes nos itens a seguir. Os diagramas foram elaborados com a ferramenta *Computer Aided Software Engineering* (CASE) Enterprise Architect e o MER foi elaborado com a ferramenta CASE Power Designer.

3.2.1 Diagrama de casos de uso

O diagrama de casos de uso especifica as interações efetuadas entre o sistema e os atores que fazem uso do sistema. Foram identificados nove casos de uso que representam os requisitos funcionais da ferramenta desenvolvida e um ator, o gerente de projeto, que é o usuário do sistema que fornecerá os dados para a análise. Este ator poderia ser também um testador, desde que ele seja responsável por fornecer os dados para as estimativas. A figura 6 mostra os casos de uso, o ator e a relação entre eles.

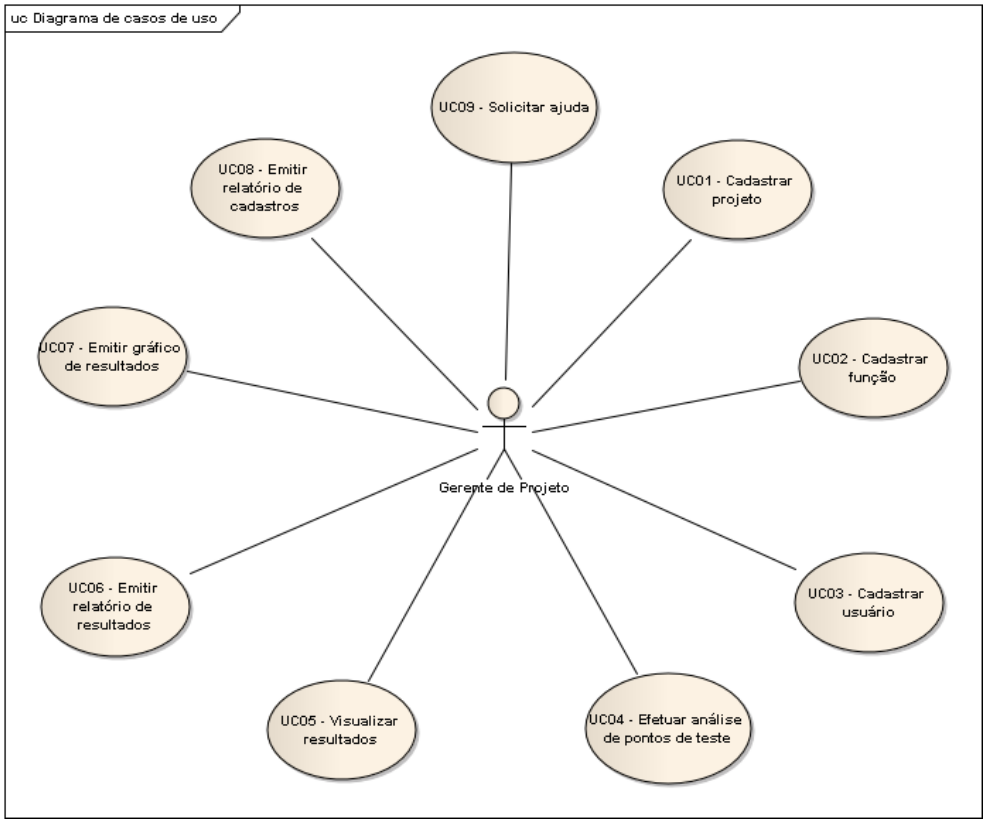


Figura 6 – Diagrama de casos de uso

No que diz respeito a descrições, os casos de uso podem ser definidos como:

a) UC01 - Cadastrar projeto: permite ao gerente de projeto cadastrar o projeto do qual será realizada a análise por pontos de teste, informando um nome para o projeto, uma descrição e o nome do gerente responsável;

b) UC02 – Cadastrar função: permite ao gerente de projeto cadastrar as funções de um projeto já cadastrado no sistema, informando um nome para a função, uma descrição e os dados necessários para a análise (pontos de função, importância do usuário, intensidade de uso, interface (em relação a arquivos), interface (em relação a funções), complexidade e uniformidade);

c) UC03 - Cadastrar usuário: permite ao gerente de projeto cadastrar um usuário no sistema, com *login* e senha, ter acesso às funcionalidades da ferramenta;

d) UC04 – Efetuar análise de pontos de teste: permite ao gerente de projeto efetuar o cálculo dos pontos e horas de teste utilizando o sistema. Para isso, ele deve fornecer os dados necessários para o cálculo dos pontos de teste dinâmicos, pontos de teste estáticos, horas de teste primárias e horas de teste totais. Os pontos de teste dinâmicos consistem na influência que os requisitos têm para o resultado dos testes considerando a funcionalidade, a performance, a segurança e a aderência e efetividade da ferramenta, bem como quantas destas características possuem coleta

de dados para processamentos futuros. Os pontos de teste estáticos são definidos pela quantidade de características do sistema (dentre funcionalidade, performance, segurança e aderência e efetividade) que sofrem revisão de documentação e de código através da utilização de *checklists*. Os dados exigidos para as horas de teste primárias são a qualificação da equipe de teste e a avaliação do ambiente de teste considerando as ferramentas de teste, a existência de testes de precedência, a existência de documentação de teste, o ambiente de desenvolvimento, o ambiente de teste e o *testware*. As horas de teste totais são obtidas através de fatores de ajuste informados pelo gerente e baseados na quantidade de técnicos da equipe de qualidade e na existência de ferramentas de gerenciamento;

e) UC05 - Visualizar resultados: permite ao gerente de projeto visualizar o resultado final da análise em pontos de teste e em horas de teste;
f) UC06 - Emitir relatório de resultados: permite ao gerente de projeto emitir um relatório contendo o resultado da análise em pontos de teste e horas de teste, bem como informações parciais obtidas durante a análise e os dados do projeto cadastrado;
g) UC07 - Emitir gráfico de resultados: permite ao gerente de projeto emitir um gráfico contendo o resultado da análise por pontos de teste e informações parciais obtidas durante a análise em forma de gráfico;
h) UC08 - Emitir relatório de cadastros: permite ao gerente de projeto emitir um relatório listando todos os projetos e funções cadastrados por ele no sistema;
i) UC09 - Solicitar ajuda: permite ao gerente de projeto acessar um menu contendo informações sobre a utilização da ferramenta e sobre análise por pontos de teste.

3.2.2 Diagrama de classes

Na figura 7 é apresentado o diagrama de classes do protótipo implementado. Este diagrama mostra as classes mais importantes para o sistema, seus métodos e atributos.

A principal classe do sistema é denominada `TestLight`. É ela a responsável por controlar que páginas são mostradas ao usuário, sendo que as operações a serem realizadas (cadastros, cálculos, salvar e coletar dados no banco) são realizadas pelas classes correspondentes a cada página. Esta classe contém o menu principal que dispõe os botões

através dos quais podem ser acessadas as páginas e funcionalidades da ferramenta. Como esta classe interage com as outras com a finalidade de instanciar uma página e esta é instanciada apenas uma vez quando for chamada, a relação entre estas classes foi estabelecida como sendo 1 para 1.

Na parte superior esquerda do diagrama são vistas as classes correspondentes às páginas de ajuda (`PaginaAjuda`), informações sobre o sistema (`PaginaSobre`), principal (`PaginaHome`) e de início de sessão (`PaginaLogin`). Estas são instanciadas pela classe `TestLight` sempre que a opção correspondente do menu principal for selecionada. A classe principal também se relaciona com a classe `Usuario` representando a instância do usuário que iniciou sessão no sistema através de *login*.

A seguir podem ser identificadas as classes `PaginaCadastroUsuario`, `PaginaCadastroProjeto` e `PaginaCadastroFuncao`, todas ligadas à classe `TestLight` que instancia estas páginas e as mostra na tela. A classe da página de cadastro de usuários pode criar instâncias de usuários e por este motivo está ligada à classe `Usuario`. Da mesma maneira as classes das páginas de cadastro de projeto e função estão ligadas às classes `Projeto` e `Função`, respectivamente.

Cada instância da classe `Projeto` contém uma instância da classe `Funcao` para cada função cadastrada no projeto e uma única instância da classe `CalculoTPA`. Esta classe controla todos os cálculos realizados durante a análise, armazena os valores necessários para estes cálculos e possui funções para realizar cada um dos três passos nos quais a ferramenta divide a análise. A classe `Funcao` também mantém dados utilizados no cálculo, mas apenas os referentes às funções em si, e não ao projeto como um todo.

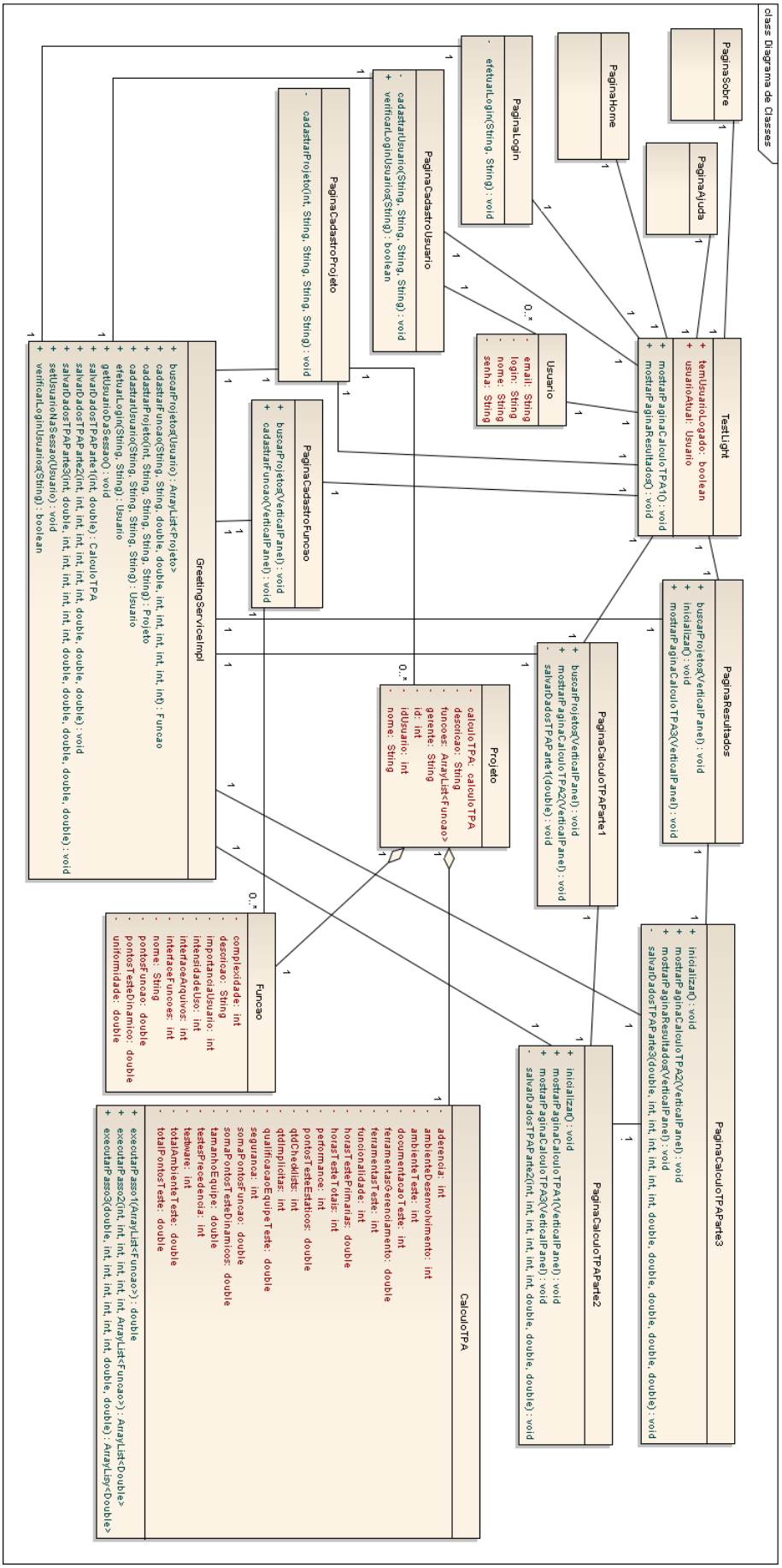


Figura 7 – Diagrama de classes

Na parte superior direita do diagrama se encontram as quatro classes correspondentes às páginas mostradas ao usuário, onde é feita a análise propriamente dita. Nota-se que as classes `PaginaCalculoTPAParte1` e `PaginaResultados` estão ligadas à classe `TestLight`, ou seja, podem ser acessadas pelo menu principal, mas as classes `PaginaCalculoTPAParte2` e `PaginaCalculoTPAParte3` não. Estas classes só são acessadas durante a análise, conforme os passos são seguidos, e por isso não estão ligadas à classe `TestLight`. Ressalta-se também que, embora as funções que efetuam a transição entre telas não foram incluídas no diagrama, estas classes referentes ao cálculo listam as funções que mudam as páginas entre elas. A classe `PaginaCalculoTPAParte2`, por exemplo, mostra funções denominadas `MostrarCalculoTPAParte1` e `MostrarCalculoTPAParte3`. Isto foi feito apenas para melhor identificar, no diagrama, quais classes podem seguir para quais classes dentro do cálculo do TPA.

Na parte inferior do diagrama encontra-se a classe `GreetingServiceImpl`, que faz as operações envolvendo o servidor de dados. As funções listadas nesta classe são as interações que a ferramenta realiza com o banco para armazenamento ou recuperação de informações. Ela possui ligações com as classes das páginas de *login*, cadastros e cálculos para salvar ou carregar estes dados sempre que for necessário.

3.2.3 Diagrama de seqüência

O diagrama de seqüência mostra a ordem da troca de mensagens entre objetos resultante dos processos executados pelo aplicativo. No diagrama que pode ser visto na figura 8 é representado o caso de uso UC04 – Efetuar análise de pontos de teste, ou seja, os procedimentos do cálculo de pontos de teste executado pela ferramenta.

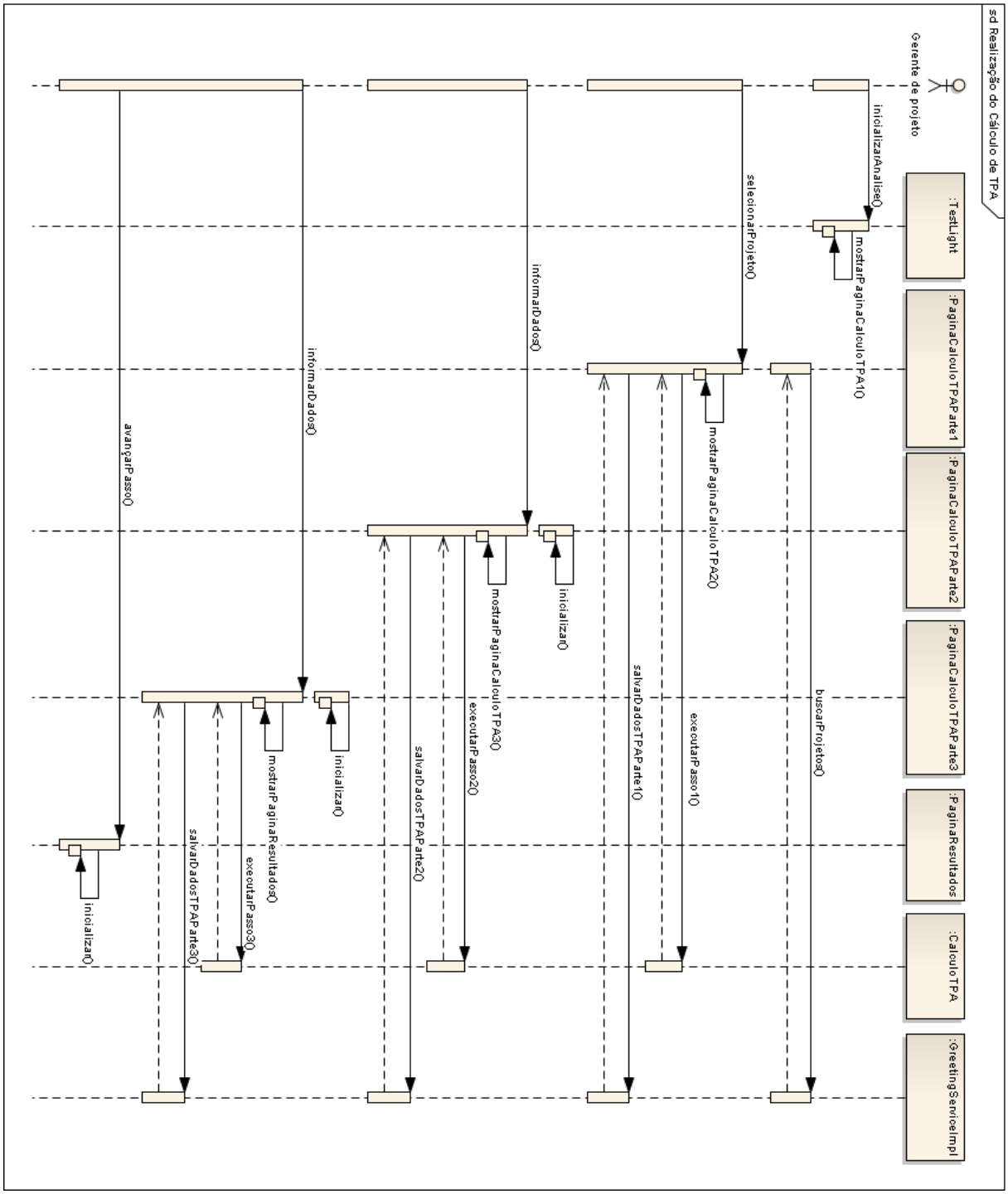


Figura 8 – Diagrama de seqüência

O ator representado no diagrama é o gerente de projeto, que é quem faz os cadastros na ferramenta e efetua a análise através dela. O primeiro objeto mostrado no diagrama leva o nome da ferramenta, TestLight, por ser a classe principal do sistema, controlando tudo o que é feito. Os objetos denominados PaginaCalculoTPAParte1, PaginaCalculoTPAParte2, PaginaCalculoTPAParte3 e PaginaResultados representam a interface que é apresentada ao usuário para que ele interaja com a ferramenta. O próximo objeto representa a classe CalculoTPA, que no sistema está contida pela classe de projetos. Esta classe é a responsável

por guardar todos os dados referentes ao cálculo de TPA de um projeto e também é ela que realiza os cálculos que são exigidos pela análise. O último objeto faz referência ao servidor que armazena as informações em banco de dados.

Quando o gerente de projeto clica no botão “Cálculo de Pontos de Teste” do menu, o sistema se encarrega de mostrar para ele a página onde pode ser iniciada a análise por pontos de teste, conhecida internamente como “Página 1”. Este nome faz referência ao fato da aplicação dividir a análise em quatro passos e este ser o primeiro deles. A classe PaginaCalculoTPAParte1 então busca todos os projetos cadastrados no servidor para que eles sejam informados na tela. O gerente de projeto seleciona o projeto que deve ser analisado e avança para o próximo passo. Neste momento, a função executarPasso1() realiza os primeiros cálculos que devem ser feitos utilizando os dados já fornecidos durante o cadastro das funções do projeto selecionado e a função salvarDadosTPAParte1() armazena no banco de dados os resultados obtidos nestes cálculos.

Para o segundo passo devem ser informados os valores necessários para o cálculo do total de pontos de teste. A função inicializar() é a responsável por mostrar os dados já cadastrados na tela. Se o gerente de projeto já efetuou a análise deste projeto anteriormente, os dados informados durante esta análise são coletados no servidor e trazidos para a tela para que eles possam ser revistos ou alterados. Após o gerente de projeto informar os dados exigidos pela página do segundo passo, denominada “Página 2”, ele deve avançar para a próxima tela. O mesmo processo realizado na mudança de página anterior é então realizado, com a diferença que os dados a serem salvos no servidor são os que o gerente informou na página 2.

O passo 3 pode ser designado como uma nova iteração do passo 2; apenas os valores a serem informados são diferentes. Em seguida é realizada a última parte do cálculo, os dados são armazenados no banco e é mostrada a página 4, também conhecida como “Página de Resultados”. Esta página mostra ao usuário os resultados dos cálculos efetuados.

3.2.4 Modelos de entidade e relacionamento

A figura 10 apresenta o MER conceitual do protótipo desenvolvido, o qual ilustra as entidades utilizadas na ferramenta, seus atributos e os relacionamentos entre elas. A tabela Projeto armazena os dados principais dos projetos cadastrados, os quais também contem funções. Os dados destas são guardados na tabela Função. Cada projeto pode possuir várias

funções mas cada função está associada a apenas um projeto, por isso a relação entre estas tabelas é de uma para muitas. Todos os valores relacionados ao cálculo dos pontos e horas de teste são armazenados na tabela CalculoTPA, que possui uma relação de uma para uma com a tabela Projeto. A tabela Usuario possui os usuários cadastrados e, como cada usuário pode efetuar o cadastro de diversos projetos no sistema, a relação entre estas tabelas é de uma para muitas.

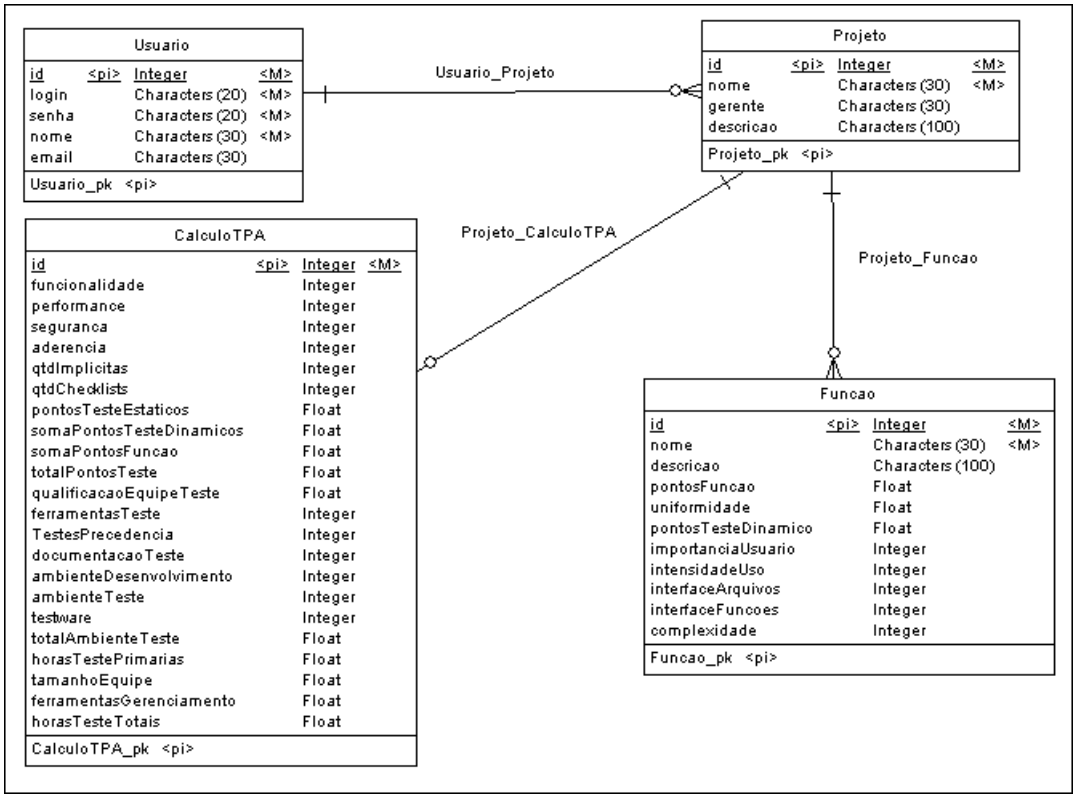


Figura 10 – MER conceitual

A figura 11 apresenta o modelo físico do protótipo desenvolvido, o qual mostra a modelagem física do banco de dados utilizado na ferramenta. As relações entre as tabelas desta modelagem são semelhantes a do MER conceitual, explicadas no parágrafo anterior.

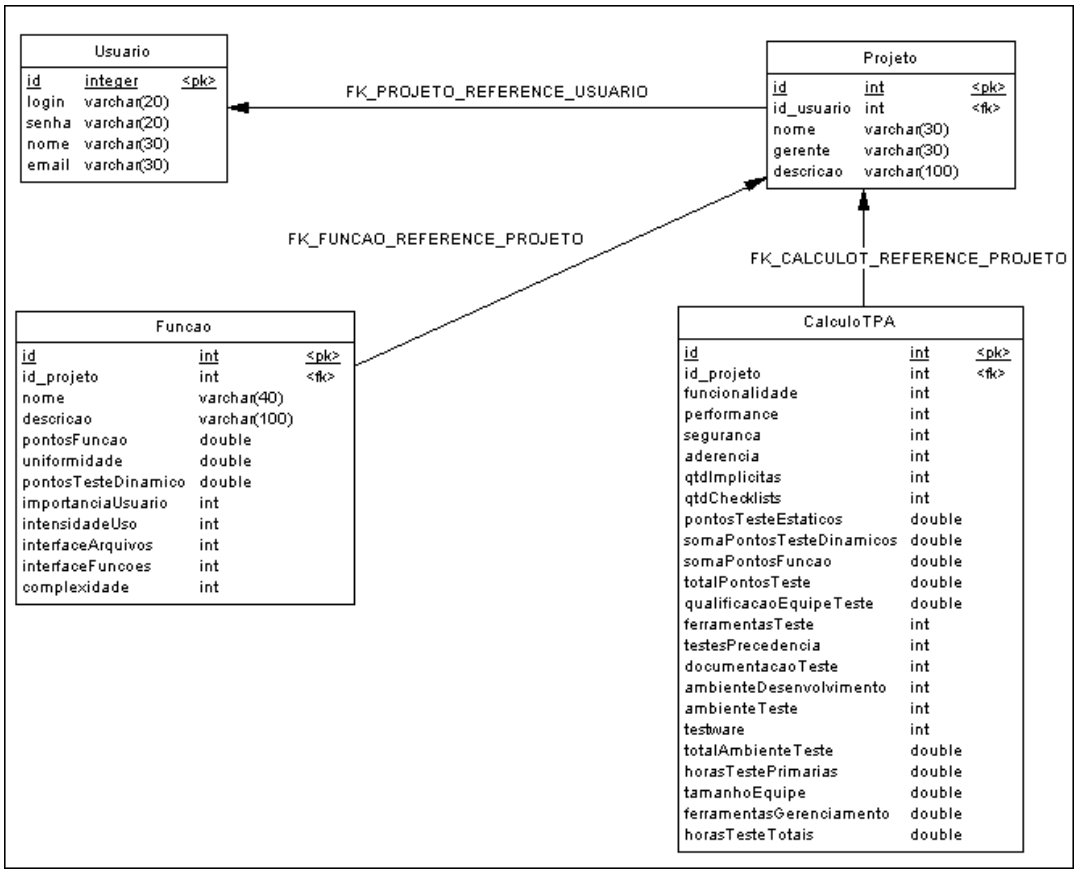


Figura 11 – MER físico

3.3 IMPLEMENTAÇÃO

Este capítulo apresenta informações referentes ao desenvolvimento do trabalho. Estas incluem as técnicas e ferramentas utilizadas, uma demonstração da operacionalidade da implementação auxiliada por um estudo de caso e os resultados obtidos.

3.3.1 Técnicas e ferramentas utilizadas

A ferramenta deste trabalho foi desenvolvida na linguagem de programação Java. Esta linguagem foi escolhida por ser adequada à proposta de elaboração de uma ferramenta para a *web* e possuir recursos que facilitam o desenvolvimento deste tipo de aplicativo. Em adição aos componentes padrão da linguagem foi utilizado o *Google Web Toolkit* (GWT), um

framework desenvolvido pelo Google para a criação de aplicativos em JavaScript com bom desempenho. Outros componentes usados foram o Jasper Reports e o JFreeChart para a geração dos relatórios e gráficos, respectivamente. Para a implementação da ferramenta foi utilizado o ambiente de desenvolvimento Eclipse.

O banco de dados utilizado é o MySQL. A ferramenta utilizada faz uso do modelo computacional cliente-servidor, realizando a interação com o usuário e a coleta de informações no cliente e o armazenamento dos dados no banco que se localiza no servidor. O servidor usado para a demonstração da aplicação em ambiente cliente-servidor é o Apache Tomcat.

A seguir serão mostrados alguns trechos de código da ferramenta a fim de demonstrar parte de sua implementação. Os trechos mostrados serão os que efetuam os cálculos para a obtenção do resultado final da análise de um projeto.

```
public ArrayList<Double> executarPasso2(int funcionalidade, int performance, int seguranca, int aderencia,
                                        int implicitas, int qtdChecklists, ArrayList<Funcao> funcoes) {
    this.funcionalidade = funcionalidade;
    this.performance = performance;
    this.seguranca = seguranca;
    this.aderencia = aderencia;
    this.qtdImplicitas = implicitas;
    this.qtdChecklists = qtdChecklists;

    double caracteristicasExplicitas = acharCaracteristicasExplicitas();
    double qualidadeDinamica = caracteristicasExplicitas + (qtdImplicitas * 0.02);

    Iterator<Funcao> iterator = funcoes.iterator();
    Funcao atual = null;
    somaPontosTesteDinamicos = 0;

    while (iterator.hasNext()) {
        atual = iterator.next();
        somaPontosTesteDinamicos += atual.getPontosTesteDinamico(qualidadeDinamica);
    }

    pontosTesteEstaticos = this.qtdChecklists * 16;

    if (somaPontosFuncao >= 500) {
        totalPontosTeste = somaPontosTesteDinamicos + ((somaPontosFuncao * pontosTesteEstaticos) / 500);
    } else {
        totalPontosTeste = somaPontosTesteDinamicos + ((500 * pontosTesteEstaticos) / 500);
    }

    ArrayList<Double> retorno = new ArrayList<Double>();
    retorno.add(pontosTesteEstaticos);
    retorno.add(somaPontosTesteDinamicos);
    retorno.add(totalPontosTeste);

    return retorno;
}
```

Figura 12 – Código do cálculo do total de pontos de teste

A figura 12 mostra um trecho de código pertencente à classe `CalculoTPA`. Este código é parte da função responsável por calcular o total de pontos de teste de um projeto, conforme mostrado no quadro 16. Os argumentos recebidos pela função são os valores que o usuário informou na tela da ferramenta que exige estes dados e as funções cadastradas no projeto que está sendo analisado. Inicialmente é chamada uma função auxiliar onde é obtido o

valor das características explícitas de acordo com a fórmula do quadro 11. Este valor é multiplicado pelas características implícitas, conforme fórmula vista no quadro 12, para que seja encontrado o valor da qualidade dinâmica. Em seguida as funções do projeto são verificadas uma por uma e seus pontos de teste dinâmico são adicionados à variável `somaPontosTesteDinamicos`. Desta maneira é encontrada a quantidade total de pontos de teste dinâmicos do projeto. Em seguida é mostrada a verificação da quantidade de pontos de função do projeto, representada pela variável `somaPontosFuncao`, para a realização do cálculo do total de pontos de teste. Como o mínimo de pontos de função que deve ser considerado para este cálculo é quinhentos, se o projeto tiver uma quantidade menor de pontos de função o valor utilizado será de quinhentos pontos. A função retorna um *ArrayList* com três valores obtidos nesta rotina, os quais serão então salvos em banco de dados pela classe que chamou a função.

```
public ArrayList<Double> executarPasso3(double qualificacaoEquipe, int ferramentas, int precedencia,
                                        int documentacao, int ambDesenvolvimento, int ambTeste, int testware,
                                        double tamanhoEquipe, double ferramentasGerenciamento) {
    ferramentasTeste = ferramentas;
    testesPrecedencia = precedencia;
    documentacaoTeste = documentacao;
    ambienteDesenvolvimento = ambDesenvolvimento;
    ambienteTeste = ambTeste;
    this.testware = testware;
    qualificacaoEquipeTeste = qualificacaoEquipe;
    this.tamanhoEquipe = tamanhoEquipe;
    this.ferramentasGerenciamento = ferramentasGerenciamento;

    totalAmbienteTeste = acharTotalAmbienteTeste();
    horasTestePrimarias = totalPontosTeste * qualificacaoEquipeTeste * totalAmbienteTeste;
    horasTesteTotais = horasTestePrimarias * (1 + this.tamanhoEquipe + this.ferramentasGerenciamento);

    ArrayList<Double> retorno = new ArrayList<Double>();
    retorno.add(totalAmbienteTeste);
    retorno.add(horasTestePrimarias);
    retorno.add(horasTesteTotais);

    return retorno;
}
```

Figura 13 – Código do cálculo do total de horas de teste

A rotina de cálculo das horas de teste primárias e totais é apresentada na figura 13. Novamente os parâmetros recebidos são os dados informados pelo usuário em uma das telas responsáveis por coletar informações para a análise. Uma função auxiliar é chamada para a obtenção do valor total do ambiente de teste conforme a fórmula mostrada no quadro 23. Com este valor podem ser obtidas as horas de teste primárias e totais. Os três valores obtidos nesta rotina são também retornados em um *ArrayList* para serem armazenados no banco.

3.3.2 Operacionalidade da implementação

A seguir será demonstrada a operacionalidade da ferramenta desenvolvida através da apresentação de um estudo de caso com um exemplo prático de utilização da análise por pontos de teste.

Para o estudo de caso foi utilizado um projeto da empresa Senior Sistemas S/A. de Blumenau. Um dos sistemas desenvolvidos pela Senior Sistemas é o Ronda Acesso e Segurança, um sistema integrado de segurança que permite controlar a entrada e saída de pessoas e de veículos, monitorar e tratar alarmes, gerenciar políticas de acesso e realizar monitoramento através de circuito fechado de televisão. O sistema possui integração com diversos dispositivos de *hardware* – de leitoras de crachá a catracas – para realizar o controle de acesso do ambiente que está sendo monitorado.

O projeto utilizado no estudo de caso, intitulado Projeto Dubai, teve como objetivo a integração do sistema Ronda Acesso e Segurança com um novo dispositivo, que até então não possuía integração com o sistema. Este equipamento passaria então a realizar as funções de validação de acesso de pessoas, controle de ativos, gerenciamento de dados necessários para a realização de suas operações e execução de comandos próprios do dispositivo ou do sistema.

O projeto possui muitos recursos e o cálculo dos seus pontos de função exigiria muito tempo, por isso foi realizada a análise por pontos de teste apenas do recurso de validação de acesso de pessoas. Este recurso compreende três funções internas do sistema, as quais serão utilizadas como referência para o cálculo dos pontos de teste deste recurso do projeto. Como a análise por pontos de teste permite a sua execução em apenas parte de um projeto, não sendo obrigatório o uso do projeto inteiro, será utilizado apenas este recurso e o tempo de execução das funções correspondentes.

Para a obtenção dos dados necessários para a realização da análise por pontos de teste foram consultados seis colaboradores que atuaram no projeto Dubai: o gerente do projeto Marco Alan Rotta, o analista de sistemas Clóvis Fabiano de Souza, o analista de qualidade Leandro da Cunha, o arquiteto de software Márcio Jasinski e dois dos programadores que implementaram as funções mencionadas anteriormente, Kleiton Stiven Finger e Franklyn Christian Gehrke.

As três funções selecionadas para a análise serão denominadas neste trabalho como “Validar acesso de pessoas”, “Validar acesso no dispositivo” e “Validar acesso no gerenciador de dados”. Juntas, elas são responsáveis por efetuar a entrada ou saída de uma

pessoa através do dispositivo e verificar se a pessoa pode fazer isso considerando as políticas de acesso definidas. Como as três funções realizam a mesma operação, os dados coletados para a análise consideraram todas estas funções como compartilhando da mesma avaliação.

A figura 14 mostra a tela principal da ferramenta quando ela é aberta. Esta tela possibilita acesso a todas as funcionalidades da ferramenta através do menu superior.

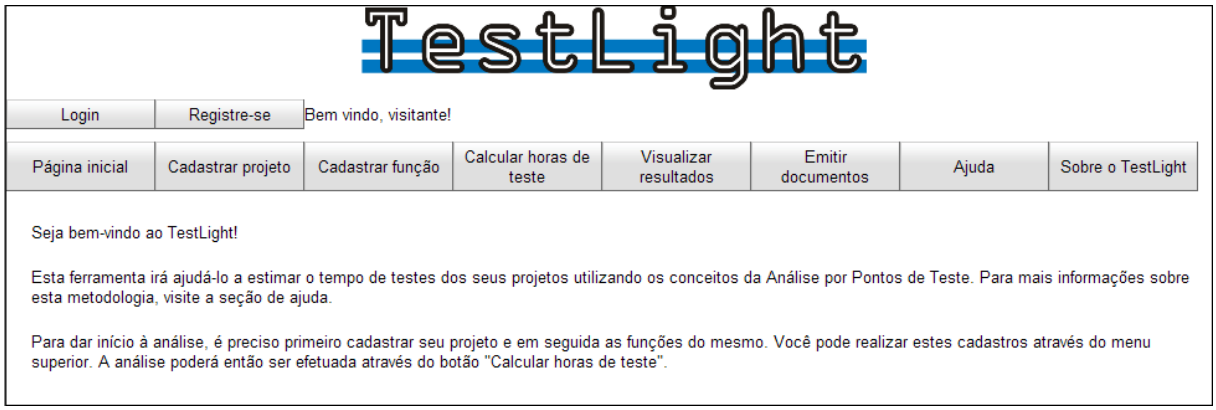


Figura 14 – Tela principal

A figura 15 mostra a tela de ajuda do sistema, a qual possui uma descrição da análise que é realizada pela ferramenta com o intuito de tirar dúvidas e esclarecer com detalhes como esta métrica funciona.

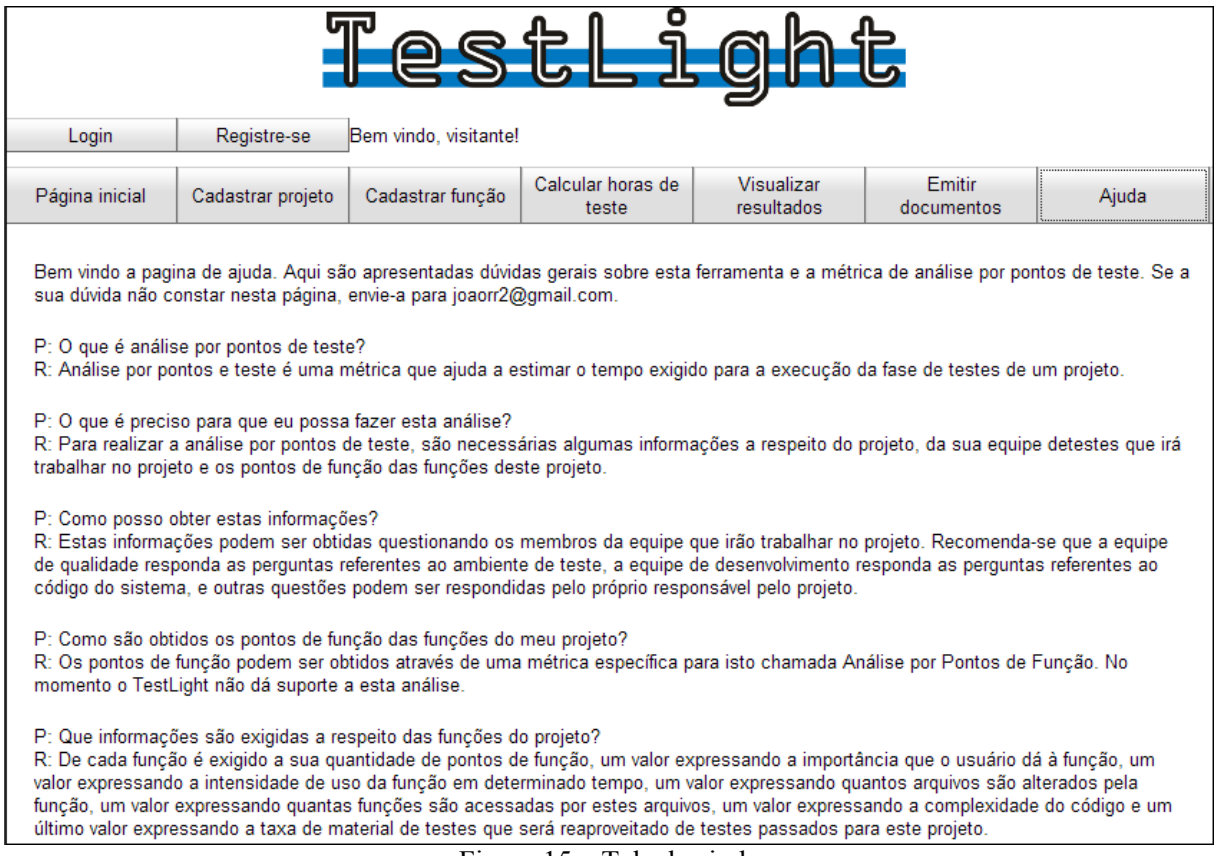


Figura 15 – Tela de ajuda

A figura 16 mostra a tela de *login*, onde o usuário pode entrar com seus dados

cadastrais para ter acesso aos seus projetos e funções cadastradas e também às outras funcionalidades do sistema.

Figura 16 – Tela de *login*

A figura 17 mostra a tela de cadastro de projetos, que é o ponto inicial para a realização da análise por pontos de teste utilizando esta ferramenta. Devem ser informados o nome do projeto, o gerente de projeto responsável e uma descrição. Na imagem a seguir é mostrado como exemplo o cadastro do projeto Dubai.

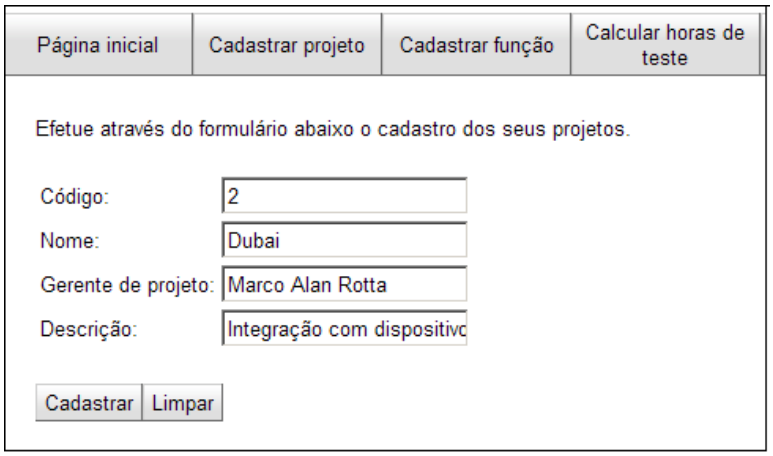


Figura 17 – Tela de cadastro de projetos

A figura 18 mostra a tela de cadastro de funções. Esta tela deve ser acessada após o cadastro de projetos para que sejam registradas as funções do mesmo que serão analisadas. Nesta tela são informados, além dos dados básicos (um nome, o projeto correspondente e uma descrição) as primeiras informações necessárias para a análise. Para cada função cadastrada é exigida a sua quantidade de pontos de função e os conceitos das funções dependentes. Para auxiliar o usuário no preenchimento destes dados e simplificar esta tarefa de modo que ele não precise realizar consultas para entender o que significa cada um destes conceitos e valores, a tela conta com uma ajuda de contexto. Sempre que um dos campos recebe foco, o lado direito da tela descreve exatamente o que ele representa, que valores devem ser fornecidos nele e o que significa cada um destes valores. A figura 18 mostra esta ajuda

descrevendo o campo da intensidade de uso. Ressalta-se que uma lista completa dos campos e valores a serem informados nesta tela, bem como nas outras telas que são vistas a seguir, pode ser encontrada na seção 2.3.1 do trabalho.

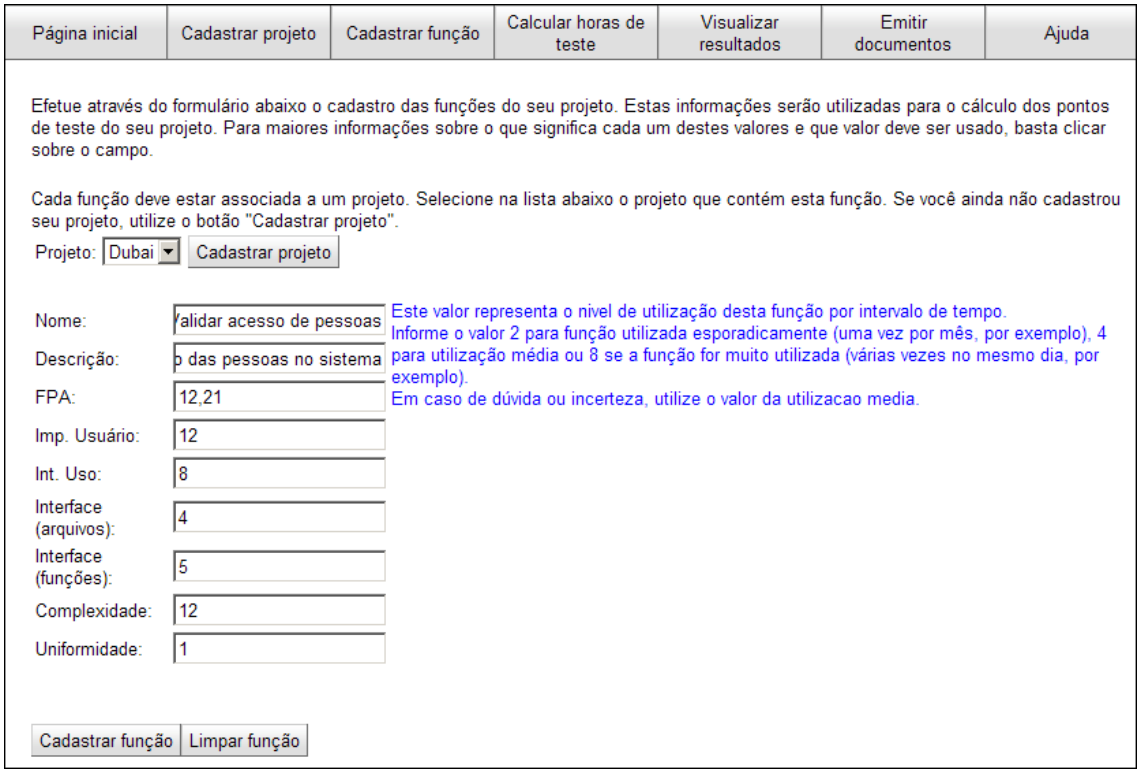


Figura 18 – Tela de cadastro de funções

Conforme mencionado anteriormente nesta seção, as funções analisadas neste estudo de caso serão três funções que juntas realizam a validação de acesso de pessoas. Para a obtenção das respostas referentes às funções dependentes que devem ser preenchidas nesta tela, foram consultados os programadores Kleiton Stiven Finger e Franklyn Christian Gehrke e o analista de qualidade Leandro da Cunha. De acordo com o que foi informado por eles, a importância que o usuário dá a estas funções em específico é alta (valor 12), pois a validação de acesso trata-se da principal finalidade do dispositivo. A intensidade de uso também é alta (valor 8) porque estes acessos costumam ser realizados por várias pessoas em diversas horas do dia. A complexidade das funções é alta (valor 12) pois existem muitas condições que devem ser verificadas considerando todas as regras de acesso que o dispositivo pode considerar. A complexidade da interface pode ser classificada como normal (valor 4) porque a quantidade de arquivos alterados pelas funções é baixa, apesar destes arquivos serem acessados por muitos recursos do sistema. O nível de uniformidade é 1,0, ou seja, não havia nenhum material de teste para ser reutilizado nos testes deste projeto. O analista de sistemas Clóvis Fabiano de Souza auxiliou na obtenção dos pontos de função das três funções analisadas. Os valores obtidos foram de 12,21 pontos de função para as funções “Validar

acesso de pessoas” e “Validar acesso no gerenciador de dados” e 119,88 pontos para a função “Validar acesso no dispositivo”.

A partir destes valores, pode-se obter os resultados para o valor das funções dependentes de cada função (quadro 29). Os cálculos para a obtenção deste valor são efetuados mais tarde em uma outra operação do sistema, e isto pode ser evidenciado através do trecho de código mostrado na figura 12.

```
FDf = [(Ue + Uy + I + C) / 20] * U
FDf = [(12 + 8 + 4 + 12) / 20] * 1,0
FDf = 1,8
```

Quadro 29 – Cálculo do valor das funções dependentes para o projeto Dubai

Depois de cadastradas as funções, deve ser acessada a opção “Cálculo de Pontos de Teste” do menu superior, através da qual poderá ser iniciada a análise propriamente dita. Esta opção abre a tela que é mostrada na figura 19, onde deve ser selecionado o projeto que sofrerá a análise por pontos de teste. A tela ressalta ainda que o pré-requisito para a análise ser feita é que haja um projeto cadastrado com funções, disponibilizando um botão para acesso ao cadastro de projetos caso o usuário tenha aberto esta tela ser ter feito o cadastro. A tela também mostra que a ferramenta divide o processo de análise em quatro passos e que este é o primeiro passo. A análise pode ser prosseguida através do botão “Próximo passo”.

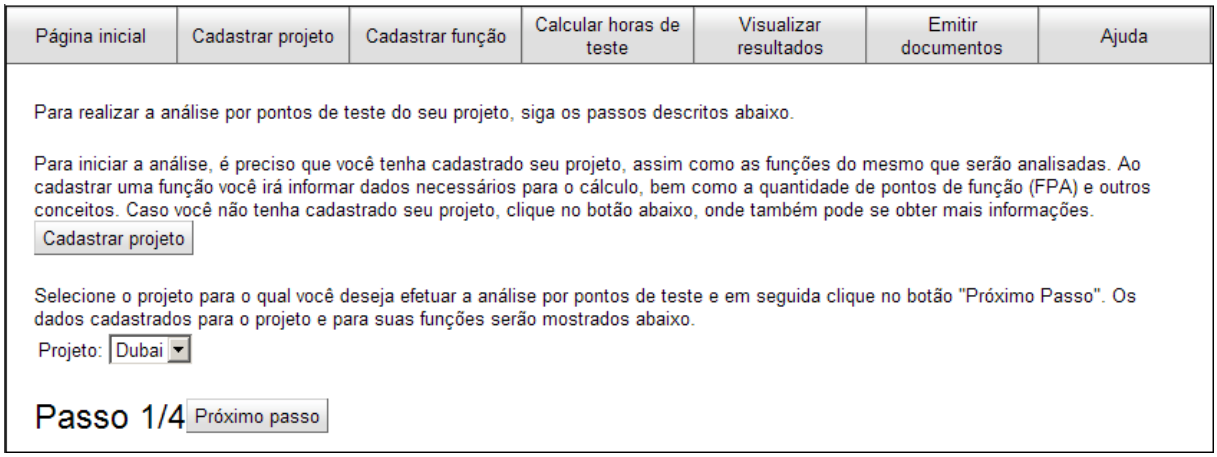


Figura 19 – Tela de cálculo das horas de teste (passo 1)

A figura 20 mostra a tela que pede as informações referentes à qualidade dinâmica, finalizando os pontos de teste dinâmicos, e referentes aos pontos de teste estáticos. Assim como a tela anterior, esta tela também possui ajuda de contexto que descreve exatamente o que significa cada um dos campos e com que valores eles devem ser preenchidos. É mostrado ao usuário, ainda, que este é o segundo dos quatro passos que constituem o cálculo dos pontos de teste finais.

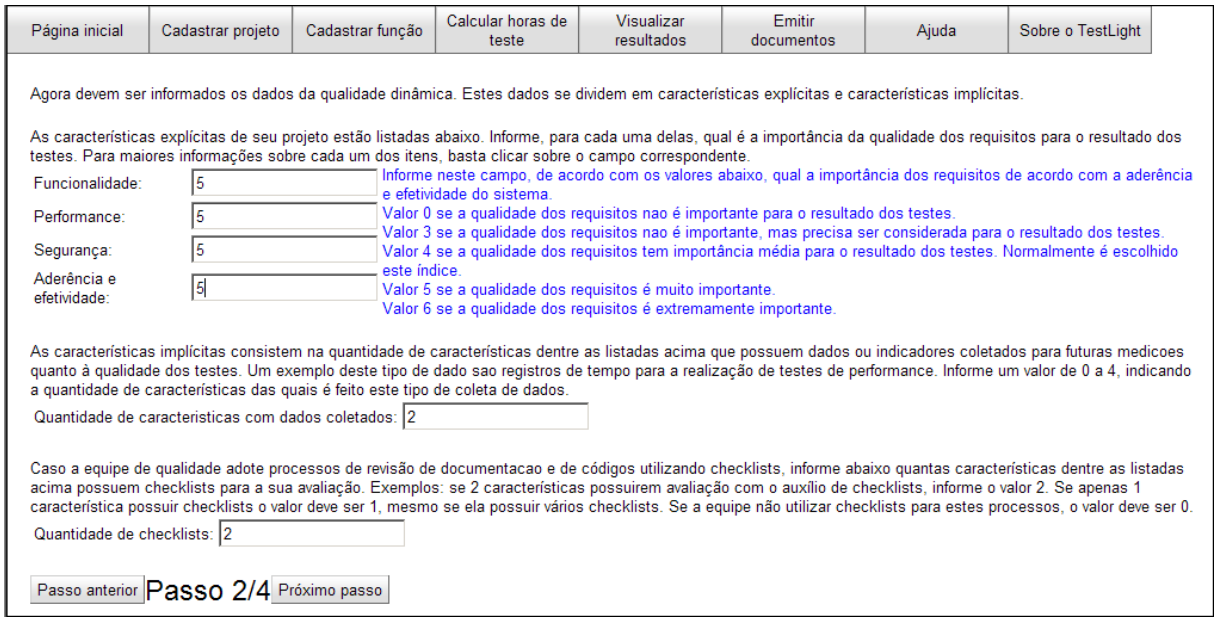


Figura 20 – Tela de cálculo das horas de teste (passo 2)

Segundo o analista de qualidade Leandro da Cunha, no projeto Dubai todos os quatro quesitos da qualidade dinâmica podem ter a qualidade dos requisitos classificada como muito importante (valor 5). Quanto às características implícitas, são duas as características que possuem coleta de indicadores para futuros processamentos: funcionalidade e performance. A respeito dos pontos de teste estático, o número de quesitos nos quais se utilizam *checklists* para a avaliação das características de qualidade são dois: funcionalidade e aderência e efetividade. A partir destes valores, podem ser obtidos os resultados ajustados de cada característica (quadro 30), o valor características implícitas (quadro 31) e os pontos de teste estáticos (quadro 32).

Valor de uma característica = valor atribuído * peso / 4 F = 5 * 0,75 / 4 = 0,9375 P = 5 * 0,10 / 4 = 0,125 S = 5 * 0,05 / 4 = 0,0625 A = 5 * 0,10 / 4 = 0,125

Quadro 30 – Cálculo das características da qualidade dinâmica para o projeto Dubai

CI = funções que possuem coleta de dados * 2 CI = 2 * 0,02 CI = 0,04

Quadro 31 – Cálculo do valor das características implícitas para o projeto Dubai

PTE = funções que utilizam *checklists* * 16 PTE = 2 * 16 PTE = 32

Quadro 32 – Cálculo dos pontos de teste estáticos para o projeto Dubai

Com a obtenção dos valores citados, também podem ser calculados os valores das características explícitas (quadro 33) e da qualidade dinâmica (quadro 34).

```
CE = F + P + S + A
CE = 0,9375 + 0,125 + 0,0625 + 0,125
CE = 1,25
```

Quadro 33 – Cálculo do valor das características explícitas para o projeto Dubai

```
QRD = CE + CI
QRD = 1,25 + 0,04
QRD = 1,29
```

Quadro 34 – Cálculo valor da qualidade dinâmica para o projeto Dubai

Os pontos de teste dinâmicos de cada função podem então ser obtidos através do cálculo mostrado no quadro 35.

```
PTDf = PFf * PDf * QRD
PTD(função 1): 12,21 * 1,8 * 1,29 = 28,35162
PTD(função 2): 12,21 * 1,8 * 1,29 = 28,35162
PTD(função 3): 119,88 * 1,8 * 1,29 = 278,36136
```

Quadro 35 – Cálculo dos pontos de teste dinâmicos para as funções do projeto Dubai

Somando os pontos de teste dinâmicos de todas as funções é obtido o total de pontos de teste dinâmicos do projeto, conforme mostra o quadro 36.

```
PTD = PTD(função 1) + PTD(função 2) + PTD(função 3)
PTD = 28,35162 + 28,35162 + 278,36136
PTD = 335,0646
```

Quadro 36 – Cálculo do total de pontos de teste dinâmicos do projeto Dubai

A soma dos pontos de função de cada uma das funções é vista no quadro 37. Os pontos de teste totais do projeto são encontrados com a execução do cálculo visto no quadro 38. Como o total de pontos de função do projeto é inferior a quinhentos, será utilizado o valor de quinhentos pontos neste cálculo.

```
PF = PF(função 1) + PF(função 2) + PF(função 3)
PF = 12,21 + 12,21 + 119,88
PF = 144,3
```

Quadro 37 – Cálculo do total de pontos de função do projeto Dubai

```
PT = PTD + (PF * PTE) / 500
PT = 335,0646 + (500 * 32) / 500
PT = 367,0646
```

Quadro 38 – Cálculo do total de pontos de teste totais do projeto Dubai

A figura 21 mostra a tela que corresponde ao terceiro passo do processo de análise. Nela deve ser feita a avaliação do ambiente de teste, da qualificação da equipe de teste e dos fatores de controle. Estes dados irão se basear nos pontos de teste calculados no passo anterior e a partir deles obter as horas de teste estimadas para o projeto. Esta tela possui também a ajuda de contexto presente nas telas anteriores e um botão para que o usuário possa seguir para o último passo.

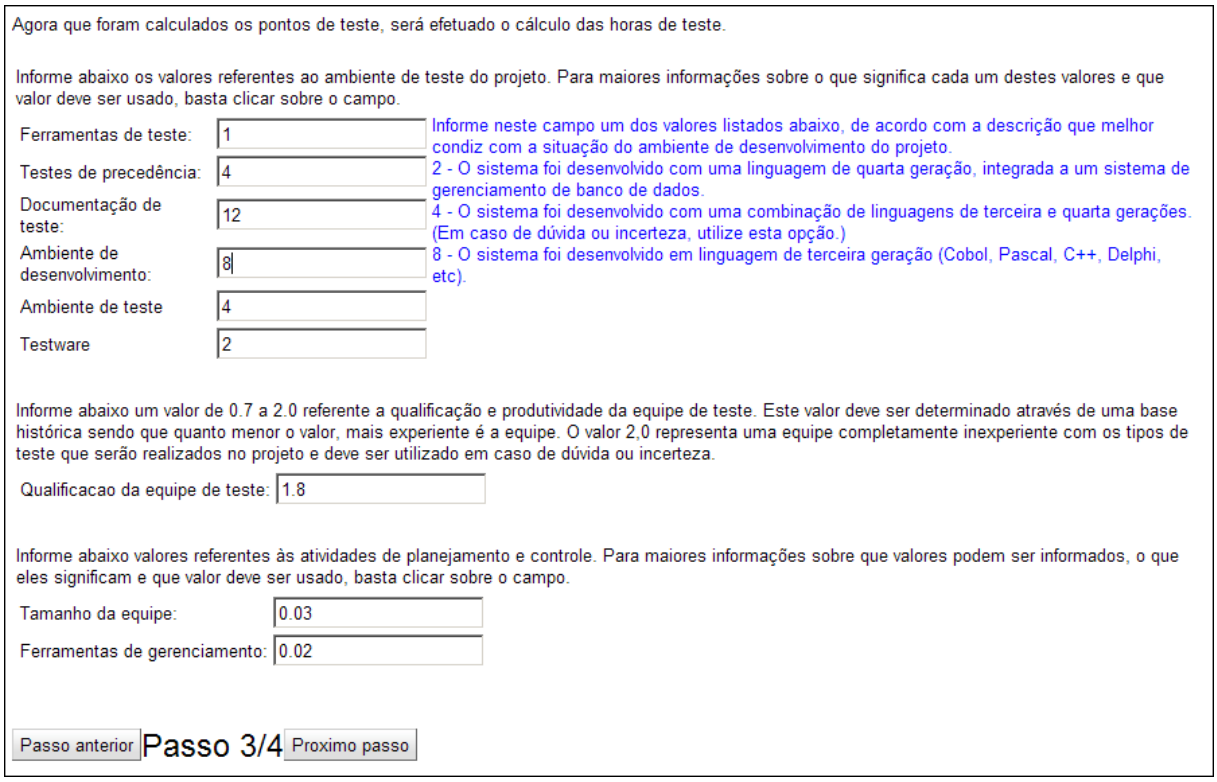


Figura 21 – Tela de cálculo das horas de teste (passo 3)

Para a obtenção das informações a respeito do ambiente de teste e dos fatores de qualidade do projeto foi consultado mais uma vez o analista de qualidade Leandro da Cunha. Foi visto que existe uma ferramenta de testes para as fases de especificação e execução dos testes (valor 1), existe um plano para o teste precedente (valor 4), a documentação de teste não segue nenhum padrão nem utiliza *templates* (valor 12), o sistema foi desenvolvido com linguagens de terceira geração (valor 8), o ambiente de teste era completamente novo e não havia sido utilizado nenhuma vez (valor 4), e existiam apenas tabelas e bases de dados que poderiam ser reutilizados como *testware* (valor 2). A qualificação da equipe de teste ficou como 1,8. Este valor foi obtido partindo do princípio que havia quatro membros na equipe de teste: um classificado como experiente, dois com pouca experiência e um sem familiaridade com os testes a serem realizados. Para o índice de planejamento e controle, foi visto que existem todas as ferramentas de gerenciamento exigidas pela métrica e existiam dois técnicos na equipe de qualidade (valor 0,02). A métrica assume como o número mínimo de técnicos sendo três, dando o valor 0,03 para esta situação, então será utilizado este valor para o caso da quantidade de técnicos ser menor que três. Com estes valores e a qualificação da equipe de teste, podem ser obtidos os valores do ambiente de teste (quadro 39), do índice de planejamento e controle (quadro 40) e das horas de teste primárias (quadro 41).

```
AT = soma de todos os fatores / 21
AT = (1 + 4 + 12 + 8 + 4 + 2) / 21
AT = 1,4761
```

Quadro 39 – Cálculo do valor de ambiente de teste para o projeto Dubai

```
IPC = 1 + tamanho da equipe + ferramentas de gerenciamento
IPC = 1 + 0,03 + 0,02
IPC = 1,05
```

Quadro 40 – Cálculo do índice de planejamento e controle do projeto Dubai

```
HTP = PT * QET * AT
HTP = 367,0646 * 1,8 * 1,4761
HTP = 975,2833
```

Quadro 41 – Cálculo das horas de teste primárias do projeto Dubai

O valor das horas de teste totais pode ser obtido através do cálculo do quadro 42.

```
HTT = HTP * IPC
HTT = 849,4829 * 1,05
HTT = 1024,0475
```

Quadro 42 – Cálculo do total de horas de teste totais do projeto Dubai

O quarto e último passo do processo de cálculo dos pontos e horas de teste consiste na tela que exibe os resultados ao usuário. São exibidos os valores obtidos em sua forma mais simples, ou seja, apenas os pontos de teste e as horas estimadas. Em seguida são mostrados botões para a emissão de relatórios e gráficos que exibem os resultados com detalhes e valores parciais obtidos durante a análise. Há também um botão que leva à tela de cadastro de projetos, caso o usuário queira fazer ajustes no projeto que acabou de ser analisado. A tela de exibição de resultados pode também ser acessada diretamente pelo menu superior da ferramenta, através do botão "Visualização de Resultados". Isto permite que sejam exibidos os valores calculados para projetos que já foram submetidos à análise sem ter que refazer todos os seus passos. Se a tela for aberta através do menu superior, será exibida uma lista de projetos cadastrados para que o usuário possa selecionar dentre eles qual deve ter seus resultados exibidos na tela. Na figura 22 é possível ver o resultado da análise realizada no projeto Dubai em pontos e horas de teste.

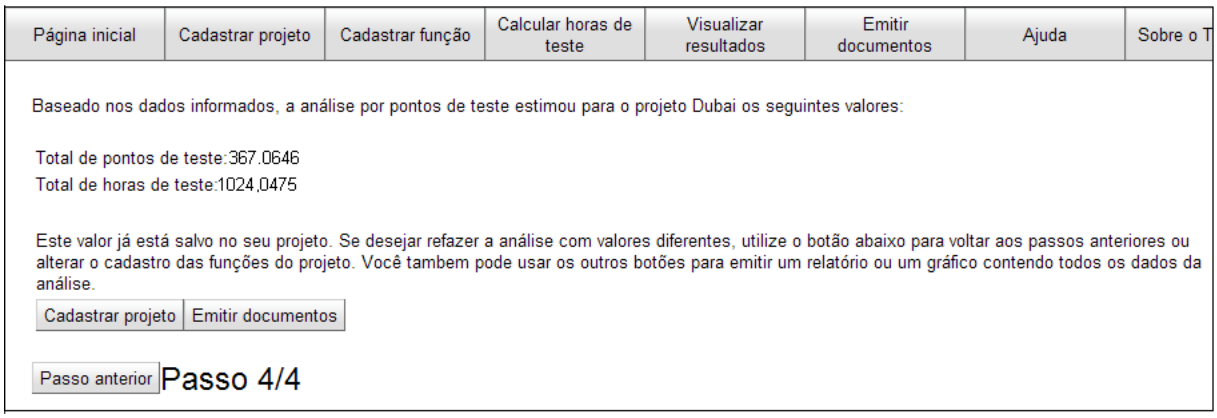


Figura 22 – Tela de cálculo das horas de teste (passo 4)

O sistema possibilita ainda a emissão de um relatório ou um gráfico contendo os resultados obtidos. É possível gerar também um relatório contendo todos os projetos cadastrados no sistema. A figura 23 mostra a tela de emissão de documentos, que contém uma descrição de cada um dos itens que podem ser gerados, e a figura 24 mostra o relatório da análise por pontos de teste, que lista os valores informados e obtidos durante o processo de análise.

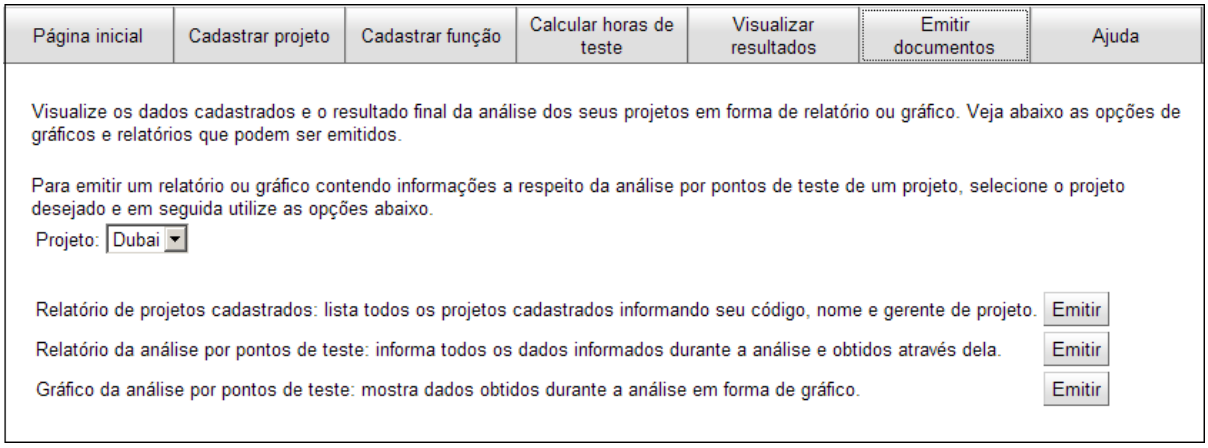


Figura 23 – Tela de emissão de documentos

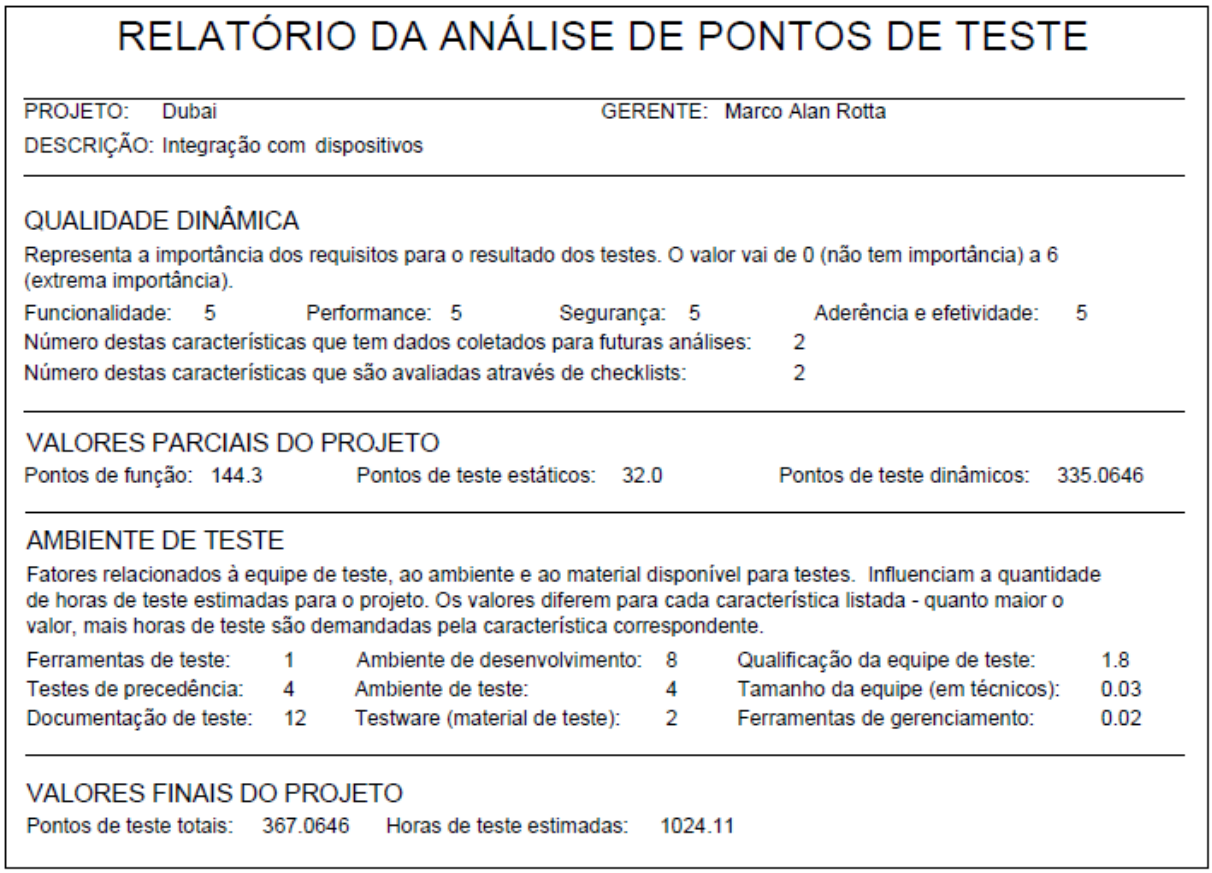


Figura 24 – Relatório da análise por pontos de teste

3.4 RESULTADOS E DISCUSSÃO

Aplicando a métrica estudada ao projeto Dubai, estimou-se que o tempo exigido para a realização dos testes das três funções selecionadas será aproximadamente de 1024 horas. O tempo total de testes efetuados no projeto (incluindo todas as suas catorze funções) durante sua execução foi de 976,5 horas, das quais estima-se que aproximadamente 209 destas horas foram dedicadas ao teste das três funções selecionadas. Ou seja, a quantidade de horas estimadas pela técnica de TPA teve um acréscimo de 390% em relação ao tempo de testes que as funções realmente tiveram.

Segundo o analista de qualidade Leandro da Cunha, o projeto Dubai necessitaria de uma quantidade maior de tempo para a execução dos testes em relação ao tempo que foi dedicado a esta fase do projeto. Por restrições de tempo, a fase de testes do projeto não pôde ser completamente executada. Se houvesse mais tempo hábil para esta etapa, os testes seriam considerados mais maduros e cobririam mais áreas que supostamente não foram atingidas durante o desenvolvimento do projeto. Em outras palavras, as horas estimadas pela métrica seriam consideradas suficientes para que possivelmente todos os testes necessários para as funções fossem executados.

Apesar da declaração dada pelo analista de qualidade a respeito da precisão da métrica no projeto estudado, com base nesta primeira verificação é possível questionar a precisão da análise por pontos de teste. Isto porque alguns dos valores utilizados nesta métrica estão sujeitos à subjetividade por parte de quem fornece as informações exigidas pelos cálculos. Enquanto as perguntas referentes ao ambiente de teste e aos fatores de controle são precisas e descrevem exatamente as situações que podem ser usadas como resposta, outras perguntas como as usadas para a obtenção dos pontos de teste dinâmicos (importância do usuário e intensidade de uso) são subjetivos e suas respostas podem variar dependendo do ponto de vista de quem fornece os dados para as respostas. Os valores da qualidade dinâmica (funcionalidade, performance, segurança e aderência e efetividade) e a qualificação da equipe de teste também podem ser considerados subjetivos, sendo que a pessoa que deve avaliar estes itens para dar uma resposta pode acabar expressando uma opinião pessoal, a qual pode variar pessoa para pessoa. Além disso, o conceito de *checklists* para os pontos de teste estáticos vai depender do que é entendido por "*checklist* utilizado na avaliação das características de qualidade do sistema", já que este conceito não é muito claro e a própria TPA não aprofunda a explicação acerca dele.

Logo, pode-se notar que o valor final da estimativa tende a variar significativamente se as possibilidades de mudança nos valores correspondentes a estas questões forem consideradas. Baseado nesta avaliação, foram feitas algumas simulações no estudo de caso realizado através da alteração de alguns dos valores considerados subjetivos a fim de verificar que influência estas mudanças teriam no resultado final da análise.

A primeira simulação realizada foi baseada na subjetividade dos valores da importância do usuário e da intensidade de uso das funções avaliadas. No projeto Dubai, estas três funções tiveram estas características classificadas como altas. Foi considerado para esta simulação as classificações delas como normais ao invés de altas, diminuindo um pouco a influência destes valores na análise. Refazendo-se todos os cálculos da TPA tendo apenas estes valores alterados em relação ao estudo de caso original, o valor total de horas de teste obtido foi de 764,4131 horas.

A segunda simulação realizada considerou a possível subjetividade dos campos da qualidade dinâmica (funcionalidade, performance, segurança e aderência e efetividade). No estudo de caso realizado, os requisitos de todos estes itens foram apontados como muito importantes para o resultado dos testes. Supondo que estes requisitos possuam uma importância apenas considerada normal para o resultado dos testes, ao refazer a análise com estes valores alterados a quantidade de horas de teste finais obtidas foi de 842,9189 horas.

Uma última simulação realizada foi através da junção das duas simulações anteriores, ou seja, considerando tanto as alterações nos valores da importância do usuário e intensidade de uso como as mudanças nos valores da qualidade dinâmica. Supondo que todos estes valores possam sofrer mudanças devido as interpretações que podem ser feitas sobre eles, e realizando uma nova análise considerando todas as variações propostas, o resultado final obtido foi de 633,49 horas de teste.

Não foram realizadas mais simulações com o TPA envolvendo o projeto Dubai pois os outros fatores que possam vir a ter uma interpretação diferente de pessoa para pessoa (definição de *checklists* utilizados e qualificação da equipe de teste) foram considerados precisos e bem definidos para o projeto em questão. A equipe de qualidade dificilmente poderia ser classificada como mais experiente (valor inferior a 1,8) pela baixa quantidade de membros na equipe com experiência nos testes do projeto em relação à equipe toda. Por outro lado, a equipe também não poderia ser classificada como menos experiente (valor superior a 1,8) também pela presença destes membros experientes, que fazem com que a equipe não possa ser considerada completamente despreparada para os testes a serem executados. O número de *checklists* utilizados se baseou nos únicos *checklists* que são usados nos processos

executados pela equipe que trabalhou no projeto, então seus valores também não foram alterados para a realização de simulações com a TPA.

As simulações realizadas envolvendo diferentes valores com a intenção de representar as possíveis variações de resposta às quais a análise está sujeita mostram como os resultados podem variar baseado na subjetividade das questões a serem respondidas. A primeira simulação resultou em uma diferença de aproximadamente 260 horas em relação ao resultado obtido no estudo de caso. Já a segunda simulação diferiu do resultado original em cerca de 181 horas e a terceira simulação, que é uma junção das alterações feitas nas outras duas, indicou uma possível diferença de quase 391 horas em relação ao valor obtido na análise original. O quadro 43 lista todas as simulações realizadas e os resultados obtidos através delas, mostrando para cada simulação a quantidade de horas de teste resultantes e quantas horas diferem deste valor em relação ao estimado pela técnica de TPA executada no projeto Dubai.

Item analisado	Horas de teste	Variação (em horas)
Valor estimado pela TPA	1024,11	-
Simulação 1 (imp. usuário e int. uso alteradas)	764,41	-259,70
Simulação 2 (qualidade dinâmica alterada)	842,92	-181,19
Simulação 3 (simulação 1 + simulação 2)	633,49	-390,62
Tempo real de testes do projeto Dubai	209,00	-815,11

Quadro 43 – Resultados das simulações realizadas

Os valores totais de horas de teste obtidos são altos em relação ao tempo de teste do projeto, o que indica que ou a precisão da métrica pode nem sempre se adequar à realidade ou os testes que foram de fato executados no projeto Dubai não foram suficientes em relação ao tempo de testes estimado pela técnica. Esta segunda opção pode ser a mais correta se for considerado o que foi dito pelo analista de qualidade Leandro da Cunha no início desta seção. Ainda assim a diferença de tempo entre o estimado pela TPA e o executado nos testes foi de mais de 800 horas, o que é uma quantidade de tempo considerável. A última simulação sugeriu uma diferença de 424,5 horas, o que é um valor mais aceitável para uma diferença de tempo, mas não é o valor que foi estimado pela técnica durante o estudo de caso realizado. Segundo o analista de qualidade Leandro da Cunha, é possível que o número de horas necessárias para a execução dos testes que não foram aplicados no projeto seja esta quantidade de horas sugerida pela técnica, ou seja, mais de 800 horas.

Para a avaliação da precisão da técnica, foram consideradas também pesquisas realizadas por outros autores, as quais foram mencionadas na seção 2.3.3 deste trabalho. Através delas é possível verificar o quanto o resultado da análise resultado pode variar dependendo de que valores são fornecidos, de quem os fornece e dos fatores subjetivos

analisados nas suposições feitas anteriormente.

Em suma, pode-se concluir que a técnica de análise por pontos de teste pode ser utilizada para a estimativa do tempo de testes de um projeto ou sistema se os itens a serem avaliados forem mais objetivos e não tão influenciados pela opinião pessoal ou pelo modo que cada avaliador vê algumas questões exigidas pela métrica. Apesar destes detalhes a técnica pode se mostrar útil para a realização de estimativas, desde que possam ser obtidas as respostas para as perguntas feitas e as classificações para os itens que devem ser avaliados.

Baseando-se na quantidade de Pontos de Função (PF) de um sistema, Campos e Birnfeld (2010) sugerem ainda que "o TPA é para projetos grandes, pois o PF inicial para se utilizar o TPA é de 500 PF (o que equivale para muitas empresas em mais de 2000 horas no projeto) e não é qualquer projeto que tem este tamanho".

Em comparação com os trabalhos correlatos, a qual pode ser vista no quadro 44, o protótipo desenvolvido dispõe dos recursos de cadastro e gerenciamento de projetos e de exibição de resultados em forma de relatório, os quais estão disponíveis também nos trabalhos analisados. Os valores que devem ser informados pelo usuário são descritos com detalhes durante o processo de preenchimento dos dados, o que pode ser comparado ao trabalho analisado Testimation, que descreve cada campo apenas através do seu nome, sem detalhar no que consiste cada item a ser informado. A ferramenta desenvolvida suporta ainda a emissão de gráficos contendo os resultados obtidos, característica que não está presente nos trabalhos analisados.

O protótipo não realiza o processo de cálculo de pontos de função, exigido como pré-requisito para o cálculo dos pontos de teste. Este item pode ser considerado uma limitação da ferramenta.

CARACTERÍSTICAS	TEST MANAGER (Autor: Fábio C. Meisen)	TESTIMATION (Autores: Horne, Harris e Horne)	TESTLIGHT (Autor: João R. Rodrigues)
realiza estimativas de teste	X	X	X
cálculo de FPA			
cálculo de TPA	X		X
gerenciador de sistemas e seus dados	X		X
aplicação web		X	X
resultados das estimativas visíveis em formato de relatório	X	X	X
resultados das estimativas visíveis em formato de gráfico			X

Quadro 44 – Comparação entre Test Manager, Testimation e TestLight

4 CONCLUSÕES

A técnica de análise por pontos de teste foi criada para auxiliar as empresas desenvolvedoras de *software* na estimativa do tempo exigido pela fase de execução de testes de um sistema ou projeto. Através desta estimativa é possível fazer um melhor planejamento das fases do projeto como um todo. Como este planejamento deve considerar fatores como o tempo de duração destas fases, os prazos do projeto, a alocação dos membros das equipes e a distribuição de tarefas, todo auxílio pode ser relevante.

O protótipo desenvolvido atingiu o objetivo de ser uma ferramenta *web* com suporte à análise por pontos de teste. O requisito de permitir o cadastro de projetos e funções, bem como o de realizar a estimativa das horas de teste dos projetos cadastrados baseando-se nos dados informados nos cadastros e em um processo de análise da ferramenta também foram atingidos. Também foi possível implementar na ferramenta o sistema de auxílio planejado, com informações correspondentes ao campo que está sendo preenchido no momento, além de uma página de ajuda complementar.

Foi realizada uma aplicação do protótipo desenvolvido na empresa Senior Sistemas S/A utilizando projeto Dubai para avaliar a precisão da ferramenta e utilidade da técnica. A funcionalidade da ferramenta foi comprovada através desta aplicação. Concluiu-se que a técnica pode ser considerada útil para a estimativa das horas de teste necessárias para um projeto, desde que alguns fatores do cálculo fossem bem compreendidos devido ao seu grau de subjetividade.

No estudo de caso foi possível verificar um acréscimo de aproximadamente 815 horas no tempo de teste utilizado nas três funções selecionadas. Foram feitas simulações com a técnica considerando os itens sujeitos a interpretações subjetivas e opiniões pessoais por parte do avaliador a fim de verificar a variação de resultados que a métrica poderia resultar e assim analisar sua precisão. Com estas simulações, no caso específico do projeto Dubai, constatado uma possível variação de cerca de 390 horas no resultado final.

As ferramentas utilizadas para a elaboração deste trabalho e para o desenvolvimento do protótipo foram consideradas adequadas para os seus propósitos e de fácil utilização. As operações realizadas com banco de dados, a utilização de GWT em sistema cliente-servidor e a geração de relatórios e gráficos proporcionaram as principais dificuldades durante a etapa de implementação do protótipo, entre elas a falta de experiência do autor deste trabalho com estes recursos.

Durante a elaboração do trabalho notou-se a escassez de material sobre o assunto específico de análise por pontos de teste. Apesar disto de ter dificultado a pesquisa feita sobre o funcionamento desta técnica, acredita-se que este fator colaborou também para que o estudo realizado seja considerado relevante, não apenas no âmbito da engenharia de software mas também no que diz respeito ao desenvolvimento de sistemas que seguem uma metodologia definida. Afinal o processo de testes é cada vez mais aplicado nas empresas desenvolvedoras de *software* para que o produto desenvolvido possa atingir um nível de qualidade.

Através da utilização da ferramenta desenvolvida neste trabalho a aplicação prática da TPA torna-se mais simples. Sem uma ferramenta para guiar o usuário no processo da análise e realizar os cálculos exigidos pela métrica, pode ser difícil realizar a utilização prática desta técnica, principalmente considerando o tempo e esforço gasto para a realização de todos os cálculos necessários para a obtenção dos valores.

Para trabalhos futuros sugere-se a implementação de um processo de cálculo de pontos de função semelhante ao realizado por esta ferramenta para pontos de teste. Este processo complementaria a TPA, sendo que uma exigência desta é a obtenção dos pontos de função das funções a serem analisadas. Desta maneira, o processo de obtenção de valores para a análise por pontos de teste estaria completamente coberto por uma única ferramenta.

REFERÊNCIAS BIBLIOGRÁFICAS

ALEXANDER, Alvin J. **How to determine your software application size using function point analysis.** Estados Unidos, 2004. Disponível em: <www.devdaily.com/FunctionPoints>. Acesso em: 10 ago. 2010.

BARTIÉ, Alexandre. **Garantia da qualidade de software.** 2. ed. Rio de Janeiro; Elsevier, 2002.

BASTOS, Anderson et al. **Base de conhecimento em teste de software**. 2. ed. São Paulo: Martins, 2007.

CAMPOS, Fabrício F.; BIRNFELD, Karine. **Conversando sobre teste de software**. [S. l.], 2010. Disponível em < http://dftestes.gershon.info/index.php/Cap %C3%ADtulo_9:_Estimativas_de_Testes_(APT)>. Acesso em: 19 nov. 2010.

FOURNIER, Greg. **Essential software testing**: a use-case approach. Estados Unidos: Taylor & Francis Group, 2009.

HELLER, Roger. **An introduction to function point analysis**. Estados Unidos, 2000. Disponível em: <http://www.qpmg.com/fp-intro.htm>. Acesso em: 10 ago. 2010.

HORNE, Jeff; HARRIS, David; HORNE, Gary. **Testimation.com**. Nova Zelândia, 2006. Disponível em: <www.testimation.com>. Acesso em: 24 fev. 2010.

LOPES, Fernanda A.; NELSON, Maria A. V. Análise das técnicas de estimativas de esforço para o processo de teste de software. In: ENCONTRO BRASILEIRO DE TESTES DE SOFTWARE, 3., 2008, Recife. **Anais...** Recife: [S. n.], 2008. p. 1-7. Disponível em: < http://ebts2008.cesar.org.br/artigos/EBTS2008-Analise_das_Tecnicas_Estimativas_de_Esforco.pdf>. Acesso em: 19 nov. 2010.

MARCO, Tom. **Controle de projetos de software**: gerenciamento, avaliação & estimativa. Rio de Janeiro: Campus, 1989.

MEIER, J. D. et al. **Performance testing guide for web applications.** Estados Unidos, 2007. Disponível em: <http://perftestingguide.codeplex.com/releases/view/6690#DownloadId=17955>. Acesso em: 06 nov. 2010.

MEISEN, Fábio C. **Ferramenta de apoio à métrica de análise por pontos de teste.** 2005. 87 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MICHAEL, C. C.; RADOSEVICH, Will. **Black box security testing tools.** Estados Unidos, 2009. Disponível em: <https://buildsecurityin.us-cert.gov/bsi/articles/tools/black-box/261-BSI.html>. Acesso em: 06 nov. 2010.

PEZZÉ, Mauro; YOUNG, Michal. **Teste e análise de software**: processos, princípios e técnicas. Porto Alegre: Bookman, 2008.

RICE, Randy. **Test estimation based on testware**. Estados Unidos, 2010. Disponível em: <http://riceconsulting.com/home/index.php/Testing-Metrics/test-estimation-based-on-testware.html>. Acesso em: 31 maio 2010.

ROSA, Edson. **Software de apoio a etapa de testes, utilizando técnicas de caixa preta**. 1997. 67 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SOUZA, Augusto. **Análise de pontos de função estendida:** métrica de software baseada na abordagem das dimensões tecnológica e ambiental/contextual. 2006. 151 f. Dissertação (Mestrado em Modelagem Computacional) – Centro de Pós-graduação e Pesquisa Visconde do Cairú, Salvador.

VEENENDAAL, Erik P. W. M.; DEKKERS, Tom. Testpointanalysis: a method for test estimation. In: EUROPEAN SOFTWARE CONTROL AND METRICS CONFERENCE, SOFTWARE CERTIFICATION PROGRAMME IN EUROPE, 10th, 2nd, 1999, Herstmonceux Castle. **Proceedings...** Netherlands: Shaker Publishing BV, 1999. p. 47-60. Disponível em: <www.testexpert.com.br/files/TPA%20-%20Test%20Point%20Analysis.pdf >. Acesso em: 24 fev. 2010.