UNIVERSIDADE DO SUL DE SANTA CATARINA

YURI SILVA RODRIGUES

TÉCNICA DE ESTIMATIVA DAS ATIVIDADES DE TESTES DE SOFTWARE: ANÁLISE DE PONTOS DE TESTES

Florianópolis

2012

YURI SILVA RODRIGUES

TÉCNICA DE ESTIMATIVA DAS ATIVIDADES DE TESTES DE SOFTWARE: ANÁLISE DE PONTOS DE TESTES

Monografia apresentada ao Curso de Especialização em Engenharia de Projetos de Software da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Especialista em Engenharia de Projetos de Software.

Orientadora: Prof. Dra. Maria Inés Castiñeira.

Florianópolis

2012

YURI SILVA RODRIGUES

TÉCNICA DE ESTIMATIVA DAS ATIVIDADES DE TESTES DE SOFTWARE: ANÁLISE DE PONTOS DE TESTES

Esta monografia foi julgada adequada à obtenção do título de Especialista em Engenharia de Projetos de Software e aprovada em sua forma final pelo Curso de Especialização em Engenharia de Projetos de Software da Universidade do Sul de Santa Catarina.

Florianópolis, Março de 2012.

_____

Professora e orientadora Maria Inés Castiñeira, Dra.
Universidade do Sul de Santa Catarina

_____

Prof. Vera Regiane Niedersberg Schuhmacher, MSc.
Universidade do Sul de Santa Catarina

_____

Prof. Aran Tcholakian Morales, Dr.
Universidade do Sul de Santa Catarina

Dedico a concretização deste trabalho aos meus pais, Margarete da Silva Rodrigues e Clécio Rodrigues.

AGRADECIMENTOS

Agradeço aos colegas e professores do curso, foi um grande aprendizado dividir as experiências. À empresa que permitiu a realização deste trabalho. À professora Inês Castiñeira, por ter acreditado e apoiado na realização. Aos meus pais e minha namorada, que sempre me apoiaram e deram total apoio para realização deste objetivo.

RESUMO

Este trabalho consiste em um estudo da técnica Análise de Pontos de Testes (APT), a fim de aperfeiçoar a estimativa das atividades de testes na etapa de planejamento do desenvolvimento de software em uma empresa estudo de caso.

O estudo também apresenta as técnicas e níveis de testes, utilizados nas atividades de testes, bem como a técnica de estimativa de software, chamada, Análise de Pontos de Função, utilizada em conjunto com a Análise de Pontos de Teste. Foi pesquisada uma ferramenta de software para dar suporte à técnica de estimativa.

Também foram abordadas as atividades do fluxo de desenvolvimento da organização que são de responsabilidade ou necessitam da participação do analista de testes/testador.

Considerando um dos projetos já realizados pela empresa foi desenvolvido um exemplo, apresentando o passo a passo da técnica, na ferramenta estudada. Dessa forma foi possível analisar como a técnica APT poderia ser aplicada na empresa considerada.

Palavras-chave: Testes de software. Estimativas de Testes. Pontos de Testes. Estimativas de Software.

ABSTRACT

This work is a study of technical Test Points Analysis (TPA), in order to improve the estimate of the testing activities in the planning stage of software development in a business case study.

The study also shows the levels of testing techniques and used in testing activities, as well as a technique for estimating software call Analysis Points Function, used in conjunction with the analysis of test points. It was searched a software tool to support the estimation technique.

Also discussed were the activities of the flow development of the organization are the responsibility of participation or require testing of the analyst / tester.

Whereas one of the projects already undertaken by the company designed an example, showing step by step technique, the tool studied. It was thus possible to analyze how the APT technique could be applied in the company considered.

Keywords: Software testing. Estimates Test. Points Test. Estimates Software.

LISTA DE ILUSTRAÇÕES

Figura 1: Regra 10 de Myers. 21
Figura 2: Teste de estrutura. 22
Figura 3: Teste de caixa preta. 23
Figura 4: Elementos da contagem de pontos de função. 28
Figura 5: APT – Visão Geral. 31
Figura 6: PT – Visão Geral. 32
Figura 7: Fórmula PT. 32
Figura 8: PTD – Visão Geral. 33
Figura 9: Fórmula PTD. 33
Figura 10: Fórmula FDf. 36
Figura 11: Valor característica explícita. 37
Figura 12: Fórmula CE. 37
Figura 13: Fórmula CI. 38
Figura 14: Fórmula QRd. 38
Figura 15: Fórmula PTE. 38
Figura 16: HTP – Visão Geral. 39
Figura 17: Fórmula HTP. 39
Figura 18: THT – Visão Geral. 43
Figura 19: Fórmula THT. 43
Figura 20: Fórmula IPC. 44
Figura 21: Fluxo de desenvolvimento. 47
Figura 22: Subfluxo Preparação. 47
Figura 23: Subfluxo Planejamento. 48
Figura 24: Subfluxo Execução. 49
Figura 25: Subfluxo Encerramento. 52
Figura 26: Cadastro do projeto. 54
Figura 27: Cadastro PF. 55
Figura 28: Cadastro FDf. 55
Figura 29: Cadastro CE. 56
Figura 30: Cadastro CI. 56
Figura 31: Cadastro PTE. 57

Figura 32: Cadastro AT-01. 57
Figura 33: Cadastro AT-02. 58
Figura 34: Cadastro QET. 58
Figura 35: Cadastro IPC. 59
Figura 36: Totalizador. 59

LISTA DE QUADROS

Quadro 1: Valor Ue. 34
Quadro 2: Valor Uy. 34
Quadro 3: Classificação I. 35
Quadro 4: Valor I. 35
Quadro 5: Valor C. 36
Quadro 6: Valor ferramentas de teste. 40
Quadro 7: Valor teste de precedência. 41
Quadro 8: Valor documentação de teste. 41
Quadro 9: Valor ambiente de desenvolvimento. 42
Quadro 10: Valor ambiente de teste. 42
Quadro 11: Valor testware. 42
Quadro 12: Distribuição das horas de testes. 43
Quadro 13: Valor TE. 44
Quadro 14: Valor FG. 45
Quadro 15: Estimativas SGP. 54
Quadro 16: Horas realizadas SGP. 54
Quadro 17: Distribuição das horas de testes. 60

LISTA DE TABELAS

Tabela 1: Valor funções padronizadas. 36
Tabela 2: Valor QRd. 37
Tabela 3: Comparativo de horas. 60

LISTA DE SIGLAS

A - Aderência
APF – Análise de Pontos de Função
APT – Análise de Pontos de Testes
AT – Ambiente de Testes
C – Complexidade
CE – Característica Explícita
CI – Característica Implícita
EUA – Estados Unidos da América
F – Funcionalidade
FDf – Funções Dependentes
FG – Ferramentas de Gerenciamento
HP – Horas Previstas
HTP – Horas de Testes Primárias
I - Interface
IPC – Índice de Planejamento e Controle
NIST – *National Institute of Standards and Technology*
P – Performance
PERT – *Program Evaluation and Review Techinique*
PF – Pontos de Função
PDT – Plano de Testes
PT – Pontos de Testes
PTD – Pontos de Testes Dinâmicos
PTE – Pontos de Testes Estáticos
QET – Qualidade da Equipe Técnica
QRd – Qualidade Dinâmica
RRT – Relatório Resultado de Testes
S – Segurança
SGP – Sistema Gerenciador de Projetos
SGT – Sistema Gerenciador de Testes
TE – Tamanho da Equipe
THT – Total de Horas de Testes

U – Uniformidade

Ue – Importância do Usuário

Uy – Intensidade de Uso

XPTO – Nome fictício dado à empresa – estudo de caso

SUMÁRIO

1 INTRODUÇÃO 15
1.1 OBJETO DE PESQUISA 16
1.2 OBJETIVOS 16
1.2.1 Objetivo geral 17
1.2.2 Objetivos específicos 17
1.3 JUSTIFICATIVA 17
1.4 PROCEDIMENTOS METODOLÓGICOS 18
1.5 ESTRUTURA DO TRABALHO 18
2 REVISÃO DE LITERATURA 20
2.1 TESTE DE SOFTWARE 20
2.1.1 Técnicas de testes de software 21
2.1.1.1 Técnica de testes estruturais 22
2.1.1.2 Técnicas de testes funcionais 23
2.1.1.2.1 Técnica de testes funcional – regressão 24
2.1.2 Níveis de teste de software 24
2.1.2.1 Testes unitários 25
2.1.2.2 Testes de integração 25
2.1.2.3 Testes de sistemas 26
2.1.2.4 Testes de aceitação 26
3 TÉCNICA DE ESTIMATIVA DE SOFTWARE E DE TESTES 27
3.1 TÉCNICA DE ESTIMATIVA DE SOFTWARE 27
3.2 TÉCNICA DE ESTIMATIVA DE TESTE 29
3.2.1 Filosofia da técnica análise de pontos de testes (APT) 29
3.2.2 Total dos pontos de testes (PT) 32
3.2.2.1 Pontos de testes dinâmicos (PTD) 33
3.2.2.1.1 Pontos de função (PF) 34
3.2.2.1.2 Funções dependentes (FDf) 34
3.2.2.1.3 Qualidade dinâmica (QRd) 36
3.2.2.2 Pontos de testes estáticos (PTE) 38
3.2.3 Estimativa das horas de teste primária (HTP) 39
3.2.3.1 Qualificação da equipe de teste (QET) 39

3.2.3.2 Ambiente de teste (AT) 40
3.2.3.3 Ferramentas de teste 40
3.2.3.4 Testes de precedência 40
3.2.3.5 Documentação de teste 41
3.2.3.6 Ambiente de desenvolvimento 41
3.2.3.7 Ambiente de teste 42
3.2.3.8 Testware 42
3.2.4 Número total de horas de teste (THT) 43
3.2.4.1 Método para cálculo do IPC 44
3.2.4.2 Tamanho da equipe (TE) 44
3.2.4.3 Ferramentas de gerenciamento (FG) 45
4 ESTUDO DE CASO 46
4.1 EMPRESA ESTUDO DE CASO: XPTO 46
4.1.1 Fluxo de desenvolvimento 47
4.1.1.1 Subfluxo: Preparação 47
4.1.1.2 Subfluxo: Planejamento 48
4.1.1.3 Subfluxo: Execução 49
4.1.1.3.1 Elaborar plano de testes 50
4.1.1.3.2 Realizar testes 51
4.1.1.4 Subfluxo: Encerramento 52
4.2 FERRAMENTA PARA CALCULAR OS PONTOS DE TESTES 53
4.3 SISTEMA ESTUDO DE CASO – SGP 53
4.3.1 Utilizando a ferramenta (APT) 54
5 CONCLUSÃO 61
5.1 TRABALHOS FUTUROS 62
REFERÊNCIAS 63
ANEXO A – SEÇÕES DO PLANO DE TESTES 66
ANEXO B – PLANO DE TESTES DE CASO DE USO 68
ANEXO C – RELATÓRIO DE RESULTADO DE TESTES 69

1 INTRODUÇÃO

Nos dias atuais, uma das grandes dificuldades em gerenciar projetos de software é realizar as estimativas e saber o tamanho do que está sendo gerenciado. As empresas desenvolvedoras de software precisam construir seu produto com qualidade e no tempo acordado. Para isso, precisam conhecer o tamanho, o tempo, o esforço e o custo para desenvolver o seu software.

Para conhecer, é necessário estimar as atividades envolvidas no desenvolvimento de software, porém sabe-se que estimar não é uma tarefa fácil e muito menos assertiva. Às vezes as atividades são subestimadas ou superestimadas, o que geralmente acarretam em gastos desnecessários às empresas.

Com as estimativas mais precisas, fica mais fácil construir cronogramas, disponibilizar recursos e calcular os custos, facilitando o controle e o acompanhamento das atividades de testes no processo de desenvolvimento da organização. Segundo Tom DeMarco (2009, tradução nossa), não se pode controlar o que não se pode medir.

As estimativas são muito importantes para o planejamento das atividades do projeto. Com um planejamento mais coeso, o controle será simplificado.

Com o objetivo de tornar as estimativas das atividades de desenvolvimento mais assertivas, a empresa estudo de caso já utiliza uma técnica formal, conhecida como Análise de Pontos de Função, (VASQUEZ; SIMÕES, ALBERT, 2011), contudo, para as atividades de testes de software, não é aplicado nenhuma técnica formal. Dessa forma, o presente trabalho visa estudar a técnica Análise de Pontos de Testes (APT), (VEENENDAAL; DEKKERS, 1999), para estimar as atividades de testes, objetivando também as estimativas mais assertivas destas atividades.

De acordo com Veenendaal e Dekkers (1999), utilizando APT, é possível definir e calcular a importância relativa das funções do software, visando à utilização do tempo da equipe de testes da forma mais eficiente possível.

1.1 OBJETO DE PESQUISA

Este estudo tem como tema a técnica de mensuração das atividades de teste de software de uma empresa desenvolvedora de software da grande Florianópolis.

Sua problematização se dá porque no processo de desenvolvimento da organização, não é utilizada uma técnica formal para mensurar e estimar o tamanho e o esforço das atividades de testes. Hoje, é estimado apenas o tempo das atividades, porém nas estimativas não é utilizada nenhuma técnica. Utiliza-se como base para estimar o tempo das atividades, um cenário crítico, no qual se procede com uma estimativa por analogia, considerando projetos anteriores.

Fatores como a experiência e a produtividade da equipe, a complexidade dos casos de uso, o grau de automação dos testes, a qualidade da especificação dos requisitos, a qualidade do ambiente de testes, não são considerados nas estimativas, resultando muitas vezes em estimativas superestimadas ou subestimadas.

Por isso, faz-se necessário a utilização de uma técnica a fim de melhorar as estimativas, facilitando o gerenciamento das atividades de testes.

Considerando que a empresa utiliza a técnica de pontos de função (VASQUEZ; SIMÕES, ALBERT, 2011) para realizar as estimativas de software, questiona-se se a técnica de análise de pontos de testes é viável para estimar o esforço das atividades de testes.

1.2 OBJETIVOS

Serão apresentados a seguir como objetivo geral e objetivos específicos.

1.2.1 **Objetivo geral**

Estudar a técnica - Análise de Pontos de Testes - de mensuração do tamanho das atividades de testes, a fim de considerar a sua aplicação na empresa, estudo de caso.

1.2.2 **Objetivos específicos**

a) Estudar a técnica de mensuração do tamanho das atividades de testes de software: Análise de Pontos de Testes;
b) Pesquisar uma ferramenta *free* para calcular o tamanho das atividades de testes de software;
c) Estimar o esforço das atividades de testes de software de um projeto estudo de caso, com base no tamanho das atividades e na produtividade da equipe, utilizando a ferramenta escolhida.

1.3 JUSTIFICATIVA

Esta pesquisa tem como expectativa estudar um método para medir e estimar as atividades de testes no processo de desenvolvimento da empresa XPTO, ajudando no planejamento, gerenciamento e controle dos projetos. Medir e estimar são atividades consideradas complexas, já que dependem de fatores internos e externos.

O estudo dessa técnica foi motivado a partir da problematização atual e pelo fato da organização já utilizar a técnica de pontos de função na mensuração de software. Os pontos de função são a base para técnica que será utilizada na mensuração dos pontos de testes.

1.4 PROCEDIMENTOS METODOLÓGICOS

O procedimento metodológico deste trabalho tem o tipo de pesquisa exploratória e descritiva. Segundo GIL (1991), uma pesquisa exploratória tem como objetivo proporcionar maior familiaridade com o problema. Já a pesquisa descritiva, tem como objetivo descrever as características de determinado fenômeno. Serve de apoio para observação, classificação e interpretação das variáveis.

A natureza deste trabalho é qualitativa, pois não serão usados métodos e técnicas estatísticas. “A interpretação dos fenômenos e a atribuição de significados são básicas no processo de pesquisa qualitativa.” (SILVA; MENEZES, 2005, p. 20).

Nesse trabalho foi selecionada uma técnica já existente para futura resolução do problema na mensuração e estimativas das atividades de testes da organização.

O método de procedimento selecionado é o monográfico com a técnica de pesquisa bibliográfica baseada em livros, artigos científicos e sites.

1.5 ESTRUTURA DO TRABALHO

O capítulo um aborda a introdução, problematização, objetivos, procedimentos metodológicos e justificativa deste trabalho.

A teoria de testes de software, técnicas e níveis de testes utilizados na empresa, são apresentadas no capítulo dois.

No capítulo três, é apresentado um resumo sobre a técnica de estimativa de software, Análise de Pontos de Função, e explicado a teoria da técnica de estimativa de teste, Análise de Pontos de Testes.

O capítulo quatro aborda o estudo de caso. É detalhada a empresa considerada, é descrito seu fluxo de desenvolvimento e a ferramenta pesquisada para automação dos pontos de testes. Também é apresentado o estudo de caso, comparando a técnica de estimativa utilizada atualmente pela empresa e a técnica proposta por este trabalho.

Por fim, no capítulo cinco, é apresentada a conclusão. São demonstrados os pontos fortes e fracos da ferramenta e da técnica proposta. Também são sugeridos temas para trabalhos futuros.

2 REVISÃO DE LITERATURA

Para o desenvolvimento deste trabalho foi realizada uma revisão de literatura na qual foram selecionadas algumas obras de autores com o intuito de embasar teoricamente esta pesquisa.

Nesta revisão, foram aprofundados os conceitos de teste de software, técnicas e níveis de testes. Entre as diferentes técnicas e níveis de testes de software, são abordados com maior profundidade os utilizados na empresa.

2.1 TESTE DE SOFTWARE

Atualmente há inúmeros estudos que comprovam que erros em software causam enormes prejuízos à economia.

Segundo o estudo do Instituto Nacional de Padrões e Tecnologia dos EUA os defeitos em software resultam em um custo anual de $59,5 bilhões à economia americana. De acordo com esse estudo mais de um terço do custo poderia ser evitado com melhorias nos testes de software (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 2002).

O teste de software é uma validação dos requisitos definidos para o software. O seu objetivo é garantir que as necessidades, requisitos dos clientes, estão sendo atendidos e diminuir a probabilidade da ocorrência de falhas quando o sistema estiver em produção.

Segundo Myers (2004, p. 1, tradução nossa), “teste de software é um processo, ou uma série de processos, projetado para certificar que o software faz o que foi projetado para fazer e que não faz nada não intencional”.[1]

De acordo com Sommerville (2003), um teste bem sucedido é aquele que faz com que o sistema opere incorretamente, expondo o defeito existente, isto é, demonstra a presença e não a ausência de defeitos do sistema.

---

[1] “Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended.”

O processo de testes deve contar com uma metodologia favorável ao processo de desenvolvimento, dispor de colaboradores qualificados, ambiente e ferramentas adequadas. “A metodologia de testes deve ser o documento básico para organizar a atividade de testar aplicações no contexto da empresa.” (RIOS; MOREIRA, 2006, p. 8).

O processo de teste deve iniciar paralelamente ao processo de desenvolvimento. Podendo verificar os documentos produzidos no inicio do projeto de desenvolvimento. Ainda neste âmbito, Myers (1979, apud Bastos et al. 2007, p. 19) estabeleceu a Regra 10 de Myers, na qual defende que quanto mais tarde for encontrado um erro, tende a ser mais caro a sua correção.

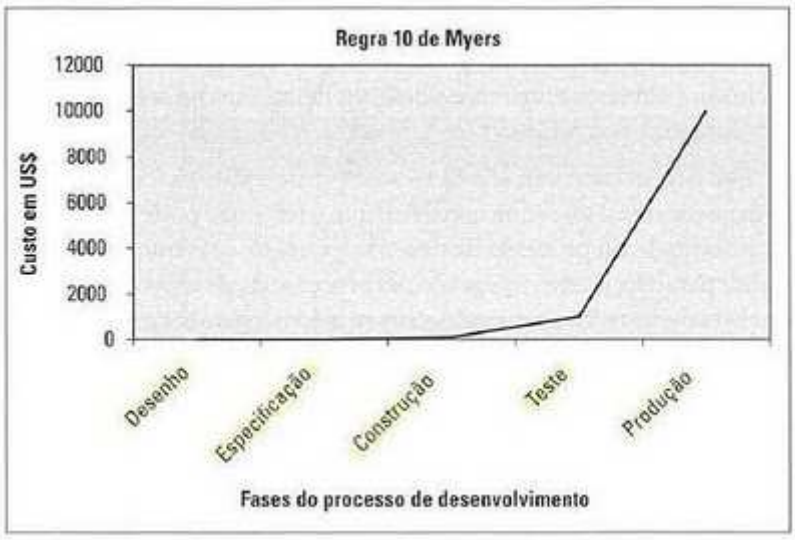

Figura 1: Regra 10 de Myers.
Fonte: Bastos et al. 2007, p. 19.

2.1.1 **Técnicas de testes de software**

Para Pressman (2001), o teste de software tem como objetivo projetar uma série de testes com alta probabilidade de encontrar o máximo de erro possível antes da sua entrega para o cliente. Para a execução destes testes são utilizadas as técnicas de testes de software. Estas técnicas fornecem uma orientação sistemática para concepção dos testes, a fim de exercitar a lógica interna, as entradas e saídas, o comportamento e o desempenho do software.

Alguns anos depois, Pressman (2006) escreveu que a técnica deve ser flexível para promover uma abordagem de teste sob medida. Ao mesmo tempo, deve ser rígida para gerar um planejamento razoável e acompanhamento gerencial, à medida que o projeto avança.

2.1.1.1 Técnica de testes estruturais

Os testes estruturais também conhecidos como testes caixa branca, devem garantir que o software seja estruturalmente sólido e que funcione no contexto técnico. “Visam avaliar as clausulas de código, a lógica interna do componente codificado, as configurações e outros elementos técnicos.” (RIOS; MOREIRA, 2006, p. 13).

Pressman (2006) explica que os testes estruturais só podem ser projetados após o projeto ao nível de componente (código fonte) existir. Os detalhes lógicos do software devem estar disponíveis.

Ainda nesta linha, Sommerville (2003), explana que estes testes são aplicados às unidades pequenas do programa, como sub-rotinas ou às operações relacionadas com um objeto.

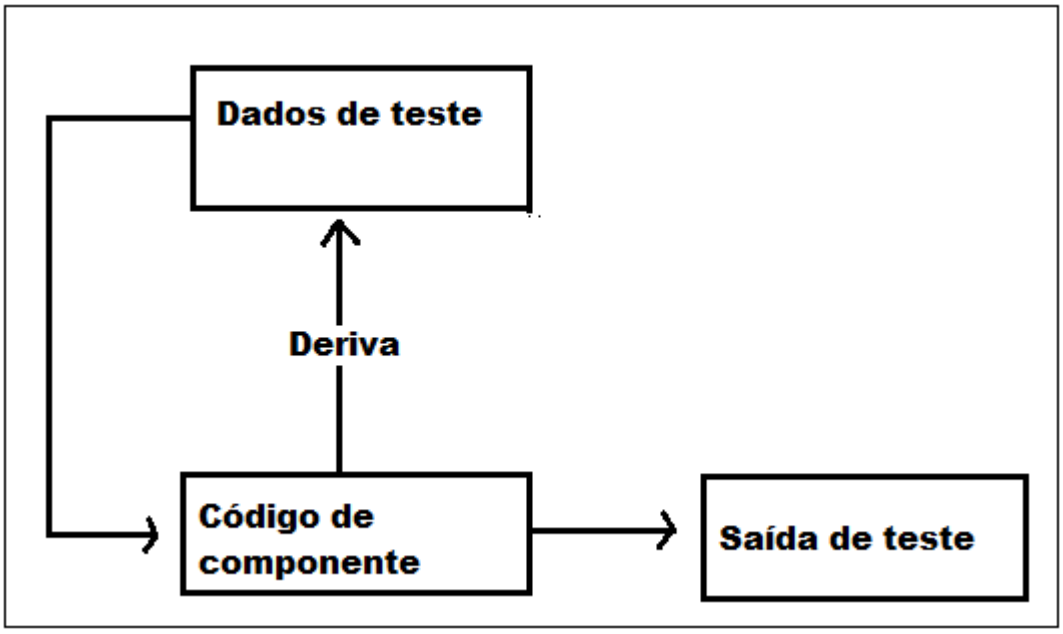

Figura 2: Teste de estrutura.
Fonte: Adaptado de Sommerville 2003, p. 382.

Estes testes são projetados para garantir que o software é estruturalmente robusto e não para garantir que esteja funcionalmente correto.

2.1.1.2 Técnicas de testes funcionais

Os testes funcionais, caixa preta, ao contrário dos testes estruturais, não estão preocupados como o processo ocorre, e sim com os resultados do processo.

Na opinião de Bastos et al. (2007, p. 57):

> Os testes funcionais são desenhados para garantir que os requisitos e as especificações do sistema tenham sido atendidos. O processo costuma envolver a criação das condições de teste para uso na avaliação da correção da aplicação.

Sommerville (2003), explica que o comportamento somente pode ser determinado estudando suas entradas e saídas relacionadas.

Estes testes objetivam verificar a funcionalidade e a aderência aos requisitos, do ponto de vista externo ou do usuário, sem se basear no código ou lógica interna do componente testado. (RIOS; MOREIRA, 2006, p. 13).

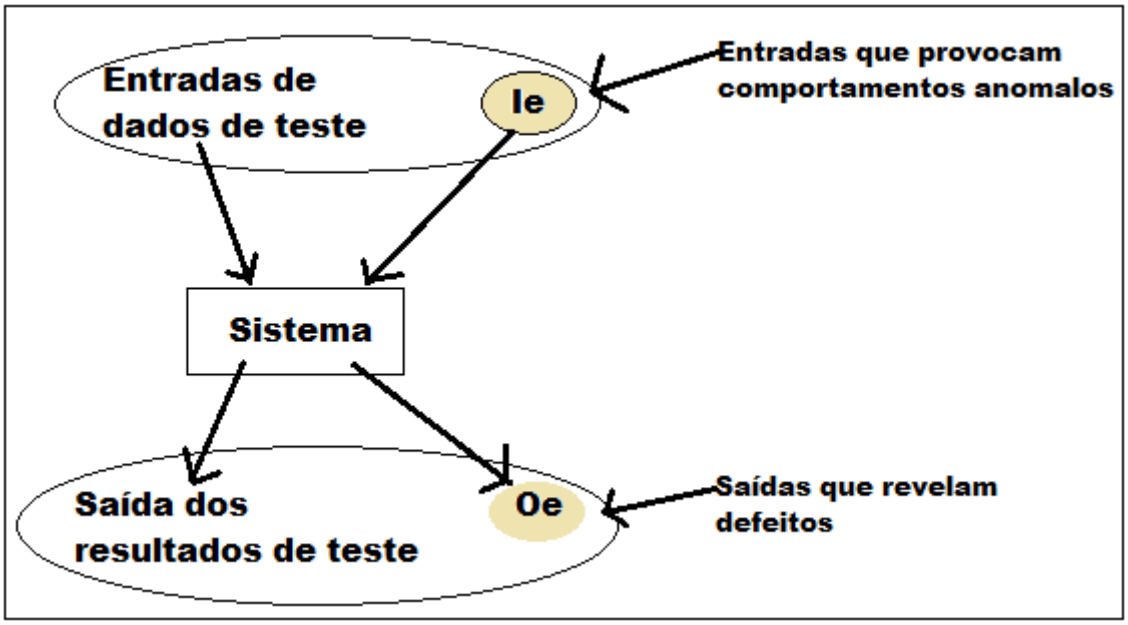

Figura 3: Teste de caixa preta.
Fonte: Adaptado de Sommerville, 2003, p. 378.

Para Black et al. (2008), esta técnica pode ser realizada em todos os níveis de testes.

2.1.1.2.1 Técnica de testes funcional – regressão

Seu objetivo é garantir que os defeitos encontrados foram corrigidos e que as correções não afetaram outras partes inalteradas do produto. Seus objetivos são:

- Verificar se as alterações realizadas geraram alguma inconsistência no aplicativo ou em outros sistemas;
- Verificar se as funções previamente testadas continuam funcionando após realização de mudanças;
- Determinar se a documentação do sistema permanece atual;
- Determinar se os dados e as condições de teste permanecem atuais.

O teste de regressão deve ser executado quando há o risco de que alterações no software possam ter afetado outras partes deste software.

2.1.2 **Níveis de teste de software**

Os níveis de testes de software representam em que fase do ciclo de desenvolvimento será aplicado um determinado tipo de teste. Os níveis de testes abordados neste trabalho são os executados na organização, testes unitários, integração, sistema e aceitação.

De acordo com Myers (1979, apud Bastos et al. 2007, p. 19), os testes unitários podem remover entre 30% e 50% dos defeitos dos programas. Já os testes de sistemas podem remover de 30% a 50% dos defeitos remanescentes.

2.1.2.1 Testes unitários

O teste unitário é o primeiro nível de teste. Estes testes devem focar o esforço na verificação da menor unidade do projeto, componente ou o módulo.

Segundo Pressman (2006), deve-se utilizar o projeto no nível de componente como guia. Os caminhos de controle são testados para descobrir erros dentro dos limites do módulo. A complexidade dos testes e dos erros descobertos é limitada pelo escopo restrito, estabelecido para o teste de unidade. Neste teste, é verificada a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente. Esse tipo de teste pode ser conduzido em paralelo para diversos componentes.

Os testes unitários são executados para descobrir erros em cálculos aritméticos, condições limites e fluxo de dados.

Ainda Pressman (2006) defende que se as entradas e saídas de dados não ocorrerem adequadamente, todos os outros testes são discutíveis.

Neste nível, os testes são executados pelos programadores.

2.1.2.2 Testes de integração

Os testes de integração são executados pelo o analista de sistemas no ambiente de desenvolvimento após os testes unitários. “É uma técnica sistemática para construir a arquitetura do software enquanto ao mesmo tempo, conduz testes para descobrir erros associados às interfaces.” (Pressman 2006, p. 297).

Esses testes devem ser desenvolvidos a partir da especificação do sistema e começar assim que algumas versões de alguns componentes estejam prontas.

O foco dos testes de integração não está em “o quê” os componentes fazem, mas se eles se comunicam conforme especificado no desenho do sistema.

Para Sommerville (2003) a grande dificuldade nos testes de integração é encontrar erros descobertos durante o processo. Após a descoberta de algum erro,

as complexas interações entre os componentes do sistema dificultam a descoberta da origem do erro.

2.1.2.3 Testes de sistemas

Os testes do nível de sistema são executados pelos analistas de testes após todos os testes de integração. Deve ser realizado em um ambiente controlado. Neste nível, o analista de teste deve criar os casos de testes. Os casos de testes são criados com base nos requisitos e no protótipo do sistema. Estes testes são executados com o objetivo de validar se o sistema funciona conforme as regras de negócios.

De acordo com Myers (2004), este, é um teste fundamental no processo, porque em termos de produto, o número de erros cometidos, e da gravidade desses erros, esta etapa no ciclo de desenvolvimento geralmente é a mais propensa a erros.

2.1.2.4 Testes de aceitação

Os testes de aceitação são executados pelo o cliente juntamente com o analista de negócio no período de homologação do sistema no ambiente do cliente. É o último passo para implantação do sistema.

Segundo Perry (2006, p. 492, tradução nossa), “o objetivo do teste de aceitação é determinar em todo o ciclo de desenvolvimento que todos os aspectos do processo de desenvolvimento atendem às necessidades do usuário”.[2]

Este teste tem o objetivo de validar se o sistema está pronto para entrar em produção e executar as funções que se espera.

---

[2] “The objective of acceptance testing is to determine throughout the development cycle that all aspects of the development process meet user needs.”

3 TÉCNICA DE ESTIMATIVA DE SOFTWARE E DE TESTES

Este capítulo apresenta inicialmente a técnica de estimativa de pontos de função, para medição de software, e a seguir a técnica de pontos de teste, utilizada para mensurar as atividades de verificação de software.

3.1 TÉCNICA DE ESTIMATIVA DE SOFTWARE

Análise de Pontos de Função (APF) é a técnica adotada pela organização, para medição de software. Esse método é a base para a técnica de pontos de teste, objeto deste trabalho e que tem como finalidade estimar as atividades de testes.

Vasquez, Simões, Albert (2011, p. 31) esclarecem que:

> [...]é uma técnica de medição das funcionalidades fornecidas por um software do ponto de vista de seus usuários. Ponto de função (PF) é a sua unidade de medida, que tem por objetivo tornar a medição independente da tecnologia utilizada para a construção do software. Ou seja, a APF busca medir o que o software faz, e não como ele foi construído.

Sommerville (2003) explica que é mais adequado utilizar pontos de função em sistemas de processamento de dados dominados por operações de entradas e saídas. Um ponto de função não é uma característica única, mas uma combinação de características do programa.

No entendimento de Rios e Moreira (2006), a contagem dos pontos de função se baseia em dois tipos de funções: funções de dados e funções transacionais. É realizada com base em cinco tipos de componentes de software:

- Arquivos lógicos internos: é um grupo lógico de dados relacionados, identificável pelo usuário, ou informações de controle mantidas dentro da fronteira do aplicativo;
- Arquivos de interfaces externas: é um grupo de dados relacionados, identificável pelo usuário, ou informações de controle,

referenciados pelo aplicativo, porém mantidos dentro da fronteira de outro aplicativo;

- Entradas externas: é um processo elementar que manipula dados ou informações de controle que vem de fora da fronteira da aplicação;
- Saídas externas: é um processo elementar que gera dados ou informações de controle, enviados para fora da fronteira do aplicativo;
- Consultas externas: é um processo elementar constituído por uma combinação entrada-saída que resulta na recuperação de dados. O lado de saída não contém dados derivados. Nenhum arquivo lógico interno é mantido no processamento.

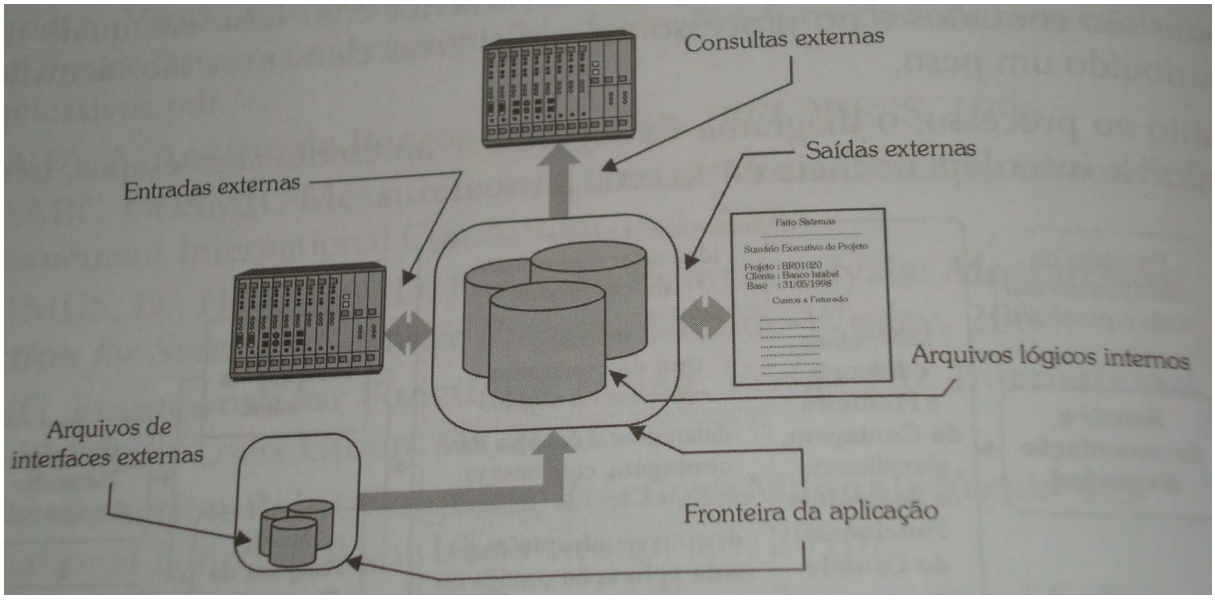

Figura 4: Elementos da contagem de pontos de função.
Fonte: Vasquez, Simões, Albert, 2011, p. 42.

Pontos de função é uma medida unicamente de tamanho funcional do sistema. Pontos não medem diretamente o esforço, o custo ou a produtividade. Com os pontos de função juntamente com outras variáveis é que se pode obter a produtividade, estimar custo e esforço. (VASQUEZ; SIMÕES; ALBERT, 2011, p. 32).

Ao realizar estimativas e medições dos pontos de função do projeto em cada fase do seu ciclo de vida é possível determinar se os requisitos funcionais cresceram ou diminuíram.

De acordo com Albrecht (1979), a mensuração baseada em funções tem provado ser um meio eficaz de comparar produtividade entre projetos.

3.2 TÉCNICA DE ESTIMATIVA DE TESTE

Com o tamanho do software em pontos de função podemos estimar o tamanho das atividades de testes. Para isso, utilizamos a técnica de Análise de Pontos de Testes (APT), desenvolvida por Martin Pol, Ruud Tennissen e Erik van Veenendaal.

Conforme Rios e Moreira (2006, p. 79), “a grande vantagem da análise de pontos de testes é usar como ponto de partida a medição do sistema em análise de pontos de função”. Para utilizar a APT se faz necessária a análise do sistema em pontos de função, caso contrário, a APT não irá funcionar.

Bastos et al.(2007, p. 227), lembram que:

> Toda técnica de medição e estimativa precisa considerar o ambiente ou o local onde está sendo usada. Ou seja, sempre é necessário adequar o modelo ao ambiente onde está sendo utilizado pela primeira vez, até que, com base num histórico de informações coletadas, seja possível obter resultados cada vez mais precisos.

Utilizando da técnica de Análise de Pontos de Testes (APT), consegue-se o tamanho das atividades de testes. Para conseguir as estimativas de tempo, são necessários acertos e calibragens. A tendência é que ao longo do tempo consiga-se chegar a resultados mais precisos, ainda de acordo com Bastos et al. (2007).

Veenendaal e Dekkers (1999), explicam que a técnica APT é para ser utilizada somente nas estimativas de testes caixa preta. Uma vez que os testes caixa branca já terão sido incluídos nas estimativas produzidas pela análise de pontos de função. A APF cobre o fator produtividade dos testes caixa branca, porém não cobre o fator dos testes de aceitação ou de sistema.

3.2.1 **Filosofia da técnica análise de pontos de testes (APT)**

A APT considera três elementos importantes: tamanho do sistema, a estratégia de teste e o nível de produtividade da equipe. Veenendaal e Dekkers (1999, tradução nossa), descrevem que os dois primeiros elementos em conjunto,

determinam o volume de trabalho de teste a ser realizado (expresso em pontos de teste). Se o número de pontos for multiplicado pela produtividade (quantidade de tempo necessário para realizar determinado volume de teste) teremos uma estimativa de teste em horas.

O tamanho do sistema para a APT é determinado principalmente pelo número de pontos de função que o sistema possui. Há alguns fatores que influenciam pouco ou não têm nenhuma influência na contagem dos pontos de função, contudo, nos testes são importantes. São eles:

- **Complexidade:** a tendência é que seja consumido mais horas de testes, conforme aumenta a complexidade da função;
- **Interface:** a dificuldade de testar o sistema aumenta de acordo com a quantidade de arquivos envolvidos em um caso de teste;
- **Uniformidade:** a reutilização do material de testes. Utiliza-se deste fator apenas quando há o aproveitamento do material de uma função para outra.

A estratégia de testes está relacionada com os requisitos de qualidade especificados para o sistema. As atividades de testes devem determinar se esses requisitos foram ou não satisfeitos. Durante a elaboração da estratégia de testes, juntamente com o cliente, deve-se determinar as características de qualidade e sua importância.

O grau de importância determina o rigor dos testes. Quanto maior for este grau, mais exigente e completo deve ser o teste, conseqüente, maior será o volume de trabalho. Na APT este volume é calculado com embasamento na estratégia de testes.

Já a produtividade na análise de pontos de testes é o tempo necessário para resolução de um ponto de testes, determinado pelo tamanho do sistema e pela estratégia de testes adotada.

Para Veenendaal e Dekkers (1999), a produtividade possui duas variáveis. Uma é a produtividade com base na experiência da equipe de testes (conhecimento e habilidade). A outra é o fator ambiental, ou seja, é o grau de influência do ambiente de testes na produtividade. O fator ambiental está relacionado com a disponibilidade do ferramental de testes, o conhecimento da equipe sobre o ambiente, a qualidade e a disponibilidade da base de testes.

Além desses fatores, Rios e Moreira (2006) definem que os testes também são afetados por esses fatores:

- **O grau de comprometimento dos usuários com os testes:** os resultados dos testes serão maiores conforme o envolvimento dos usuários nas atividades de teste e/ou homologação;
- **A qualidade do sistema testado (o ciclo de reincidência de defeitos):** quando há a existência de defeitos provenientes de fases anteriores (desenho ou projeto), mais custoso será testar o sistema;
- **O nível de cobertura almejado com os testes:** os requisitos dos sistemas é que vão estabelecer o nível de cobertura. Alguns sistemas demandam de testes de carga ou de desempenho, porém outros podem dispensar estes testes;
- **A qualidade da documentação do sistema e, em especial, dos requisitos:** os requisitos são a base do sistema. Havendo erros nos requisitos, teremos defeitos em outras fases. Os testes e as correções serão maiores.

Bastos et al. (2007) explanam que a utilização da APT no processo de teste, determinará uma unidade de medida própria, o que é justificado quando se tem uma atividade de teste independente, ligado com o tamanho do sistema em pontos de função.

Vale destacar, que algumas variáveis bem como seus valores foram definidos pelos criadores da técnica, Martin Pol, Ruud Tennissen e Erik van Veenendaal.

O processo da APT é ilustrado abaixo.

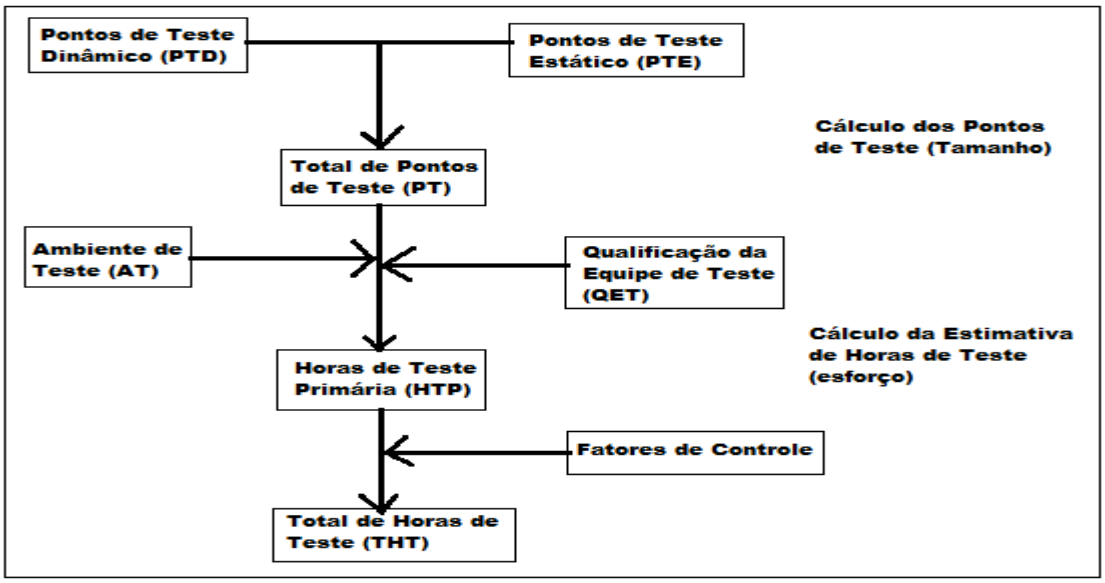

Figura 5: APT – Visão Geral.
Fonte: Adaptado de Bastos et al. 2007, p. 230.

3.2.2 **Total dos pontos de testes (PT)**

A soma dos pontos de testes dinâmicos e pontos de testes estáticos resultam no número total de pontos de testes (PT), como visto na figura abaixo.

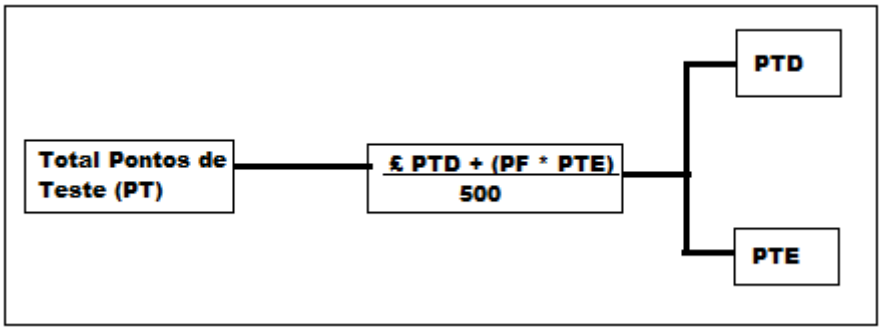

Figura 6: PT – Visão Geral.
Fonte: Adaptado de Bastos et al. 2007, p. 240.

A fórmula para se obter os PT é:

$$PT = \Sigma PTD + (PF * PTE) / 500$$

Figura 7: Fórmula PT.
Fonte: Rios e Moreira, 2006, p. 86.

Na qual:

ΣPTD é o somatório dos pontos de testes dinâmicos de todas as funções;

PF é o tamanho do sistema em pontos de função*;

PTE é o total dos pontos de testes estáticos;

* Rios e Moreira (2006) orientam que o sistema deve ter no mínimo 500 PF, ou seja, PF será o tamanho do sistema ou 500, caso tenha menos pontos de função.

A seguir encontra-se o detalhamento sobre como obter o valor das variáveis da função.

3.2.2.1 Pontos de testes dinâmicos (PTD)

Os PTD dependem dos Pontos de Função, Funções Dependentes (FDf) e Qualidade Dinâmica (QRd), como está ilustrado na figura 8 logo a seguir. Na falta de algum fator ou na dificuldade de mensurá-los, Bastos et al (2007), explicam que devem ser utilizados os valores médios mostrados em itálico.

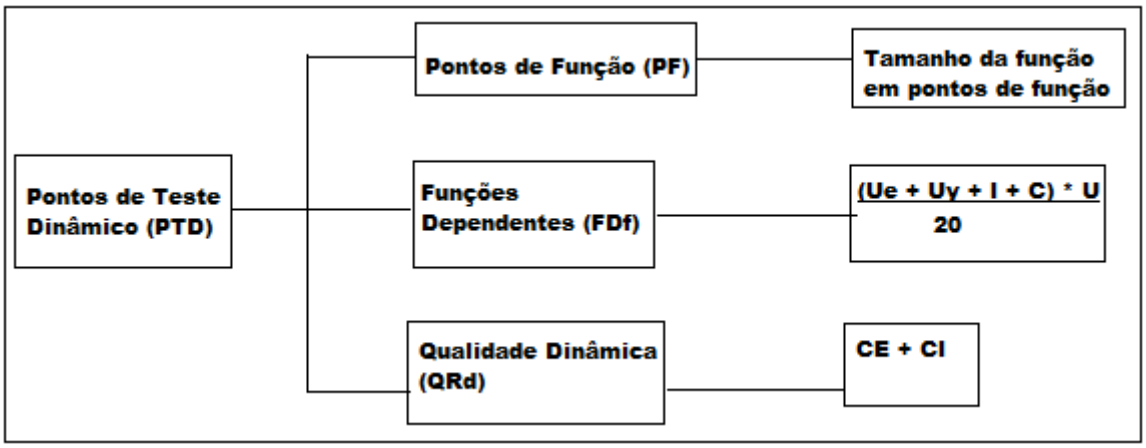

Figura 8: PTD – Visão Geral.
Fonte: Adaptado de Bastos et al. 2007, p. 232.

Os Pontos de Testes Dinâmicos são calculados através da formula:

$$PTD = PF * FDf * QRd$$

Figura 9: Fórmula PTD.
Fonte: Rios e Moreira, 2006, p. 92.

Onde:

PF é o número de pontos de função da função testada;

FDf é o total das funções dependentes;

QRd é o total das características explícitas e implícitas.

3.2.2.1.1 Pontos de função (PF)

As funções mensuradas em PF que devem ser consideradas em pontos de testes são: Entradas Externas, Saídas Externas e Consultas Externas.

3.2.2.1.2 Funções dependentes (FDf)

As Funções Dependentes mensuram a dependência que apresentam em relação às funções correspondentes mensuradas pela técnica de Pontos de Função. Deve-se analisar todas as funcionalidades e depois proceder com o somatório dos valores encontrados.

As FDf são compostas por cinco variáveis:

- **Importância do usuário (Ue):** utilizado para medir o grau de importância da funcionalidade para o usuário. Este valor pode ser obtido por entrevistas com os usuários ou com os desenvolvedores.

3	Baixa
6	*Normal*
12	Alta

Quadro 1: Valor Ue.
Fonte: Rios e Moreira, 2006, p. 88.

- **Intensidade de uso (Uy):** utilizado para levantar o nível de utilização da função pelo usuário em um intervalo de tempo (dia, semana, mês, etc). O usuário deverá informar a intensidade com que utiliza determinada função.

2	Baixa
4	*Normal*
8	Alta

Quadro 2: Valor Uy.
Fonte: Rios e Moreira, 2006, p. 88.

- **Interface (I):** utilizado para medir o relacionamento dos arquivos (data-sets), usados na função testada e a quantidade de funções que utilizam estes mesmos arquivos.
  A classificação da interface é mostrada a seguir:

Arquivos (data-sets)	**Funções**		
	1	2-5	>5
1	Baixa	Baixa	Normal
2-5	Baixa	Normal	Alta
>5	Normal	Alta	Alta

Quadro 3: Classificação I.
Fonte: Rios e Moreira, 2006, p. 88.

Caso a função não altere nenhum arquivo, deve-se atribuir o valor de interface baixo.

Após a definição do valor de relacionamento, deve-se verificar a classificação no quadro a seguir.

2	Baixa
4	*Normal*
8	Alta

Quadro 4: Valor I.
Fonte: Rios e Moreira, 2006, p. 89.

- **Complexidade (C):** a complexidade da função é determinada pelo seu código ou pelo código da parte do sistema que executa a função. A avaliação da complexidade depende do número de condições no código da função. No cálculo das condições, apenas o código de processamento deve ser considerado. Por exemplo: Um IF, conta uma vez, um IF, ELSE, conta duas vezes. As condições que são resultadas de verificações no banco de dados não contam, já que estão incluídas implicitamente na contagem de Pontos de Função.

3	Baixa – até 5 condições
6	*Normal – entre 6 e 11*
12	Alta – mais de 11

Quadro 5: Valor C.
Fonte: Rios e Moreira, 2006, p. 89.

- **Uniformidade (U):** utilizado para medir o grau de reutilização do material de testes. O seu valor varia entre 0,6 e 1,0. Utiliza-se o valor 1,0 quando não há a reutilização do material. Em casos específicos em que há a reutilização do material, seja por semelhança ou por se tratar de função-clone, deve-se utilizar o valor 0,6 (completa utilização).

O resultado final das FDf se dá através da fórmula:

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

Figura 10: Fórmula FDf.
Fonte: Rios e Moreira, 2006, p. 89.

Existem algumas funções padronizadas que já possuem uma fórmula própria de contagem, são elas: funções de mensagens de defeito, de ajuda e de estrutura de menu.

Tabela 1: Valor funções padronizadas.

Funções	**PF**	**Ue**	**Uy**	**I**	**C**	**U**	**FDf**
Mensagens de defeito	4	6	8	4	3	1	1,05
Telas de ajuda	4	6	8	4	3	1	1,05
Menus	4	6	8	4	3	1	1,05

Fonte: Rios e Moreira, 2006, p. 90.

3.2.2.1.3 Qualidade dinâmica (QRd)

As características de qualidade dinâmica medem a qualidade dos requisitos relativos à qualidade. Esta medida demonstra como a qualidade dos

requisitos pode afetar a qualidade dos testes. Neste contexto, a distinção é feita entre características explicitas (CE) e características implícitas (CI).

As CE mensuráveis são:

- **Funcionalidade (F);**
- **Performance (P);**
- **Segurança (S);**
- **Aderência e efetividade (A).**

A importância dos requisitos relativos a cada característica de qualidade é avaliado. Para cada característica têm-se os seguintes valores e pesos:

Tabela 2: Valor QRd.

CE	Peso	Sem Importância 0	Pouca Importância 3	Importância Normal 4	Muito Importante 5	Extremamente Importante 6
F	0,75					
P	0,10					
S	0,05					
A	0,10					

Fonte: Próprio autor, 2011.

Obtêm-se o valor de cada característica explícita pela fórmula:

Z = ((Valor atribuído conforme importância) * (Peso)) / 4

Figura 11: Valor característica explícita.
Fonte: Rios e Moreira, 2006, p.91.

O calculo final para se alcançar o valor das CE é:

CE = F.z + P.z + S.z + A.z

Figura 12: Fórmula CE.
Fonte: Rios e Moreira, 2006, p. 91.

Já as CI são utilizadas apenas quando há coleta de dados e/ou indicadores para futuras medições quanto à qualidade dos testes. De acordo com Rios e Moreira (2006), estes indicadores servirão para prover uma medida padrão para servir de comparação com o projeto em andamento.

A partir do pressuposto que para cada CE (funcionalidade, performance, segurança e aderência) existe uma CI associada, há a necessidade de somar-se 1 para cada uma delas, limitando-se a 4.

Portanto a fórmula para calcular as Características Implícitas é:

CI = n * 0,02 (n varia entre 0 e 4).

Figura 13: Fórmula CI.
Fonte: Rios e Moreira, 2006, p. 92.

Com os valores das CE e CI, têm-se o valor da Qualidade Dinâmica, que se dá pela fórmula:

QRd = CE + CI

Figura 14: Fórmula QRd.
Fonte: Rios e Moreira, 2006, p. 92.

3.2.2.2 Pontos de testes estáticos (PTE)

Os PTE diferentemente dos Pontos de Testes Dinâmicos que consideram cada uma das características isoladamente, levam em consideração, o sistema como um todo.

Caso a equipe de testes não adote o processo de revisão de documentação e código usando *checklists*, o valor dos PTE deve ser nulo.

Os PTE são baseados nas qualidades das características citadas anteriormente (funcionalidade, performance, segurança e aderência). Para cada característica deve-se utilizar um *checklist*. Cada *checklist* utilizado soma-se 16 pontos de testes.

A fórmula para calcular os PTE é:

PTE = 16 * n (onde n é o nº de *checklist* utilizado, podendo chegar a 4).

Figura 15: Fórmula PTE.
Fonte: Rios e Moreira, 2006, p. 93.

3.2.3 **Estimativa das horas de teste primária (HTP)**

O total de horas de testes primárias (HTP) depende do número total de pontos de testes (PT) e dos fatores: qualificação da equipe de teste (QET) e ambientes de teste (AT) como pode ser visto na figura abaixo. As horas primárias representam as fases de preparação, especificação, execução e conclusão.

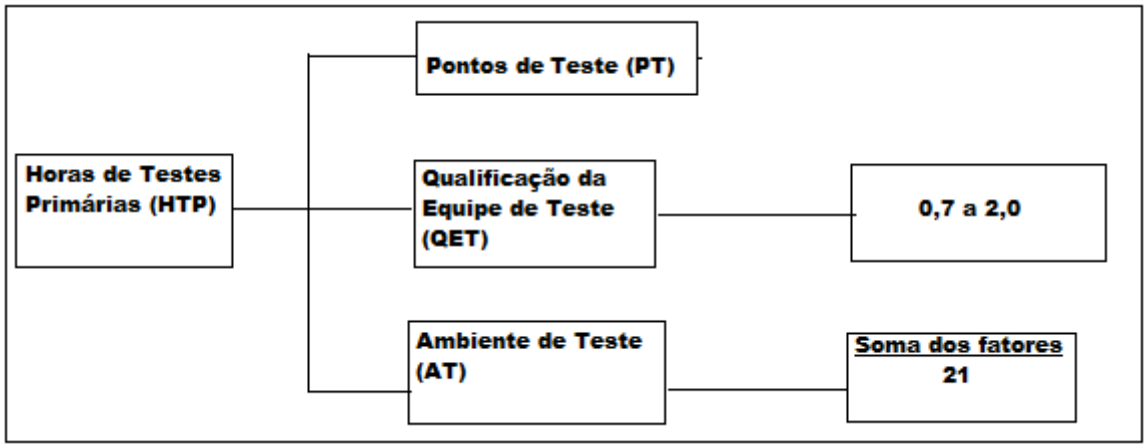

Figura 16: HTP – Visão Geral.
Fonte: Adaptado de Rios e Moreira. 2006, p. 95.

A fórmula para calcular as HTP:

HTP = PT * QET * AT

Figura 17: Fórmula HTP.
Fonte: Rios e Moreira, 2006, p. 97.

3.2.3.1 Qualificação da equipe de teste (QET)

A qualificação da equipe, geralmente, é determinada por uma base histórica. São levadas em consideração a experiência, qualificação e produtividade da equipe. O valor deve variar entre 0,7 e 2,0, onde quanto melhor for a equipe, menor será o QET.

3.2.3.2 Ambiente de teste (AT)

O número de horas de teste para cada ponto de teste além da QET, também é influenciado pelo ambiente de teste. Como no cálculo dos pontos de testes, caso as informações estejam insuficientes para permitir a avaliação, deve-se adotar os valores médios que estão em itálico.

O total do AT se dá pela soma de todos os fatores / 21.

3.2.3.3 Ferramentas de teste

Este quesito mede o grau de automatização dos testes. A disponibilidade de ferramental indica que algumas destas atividades podem ser executadas de forma automática, com isso, mais rapidamente.

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 fase de execução*
4	Não existe ferramenta de automação de teste

Quadro 6: Valor ferramentas de teste.
Fonte: Rios e Moreira, 2006, p. 95.

3.2.3.4 Testes de precedência

Este item mede a qualidade dos testes anteriores. Por exemplo, caso o teste de integração tenha sido bem executado, o teste seguinte (teste de sistema), terá resultados melhores.

2	Existe um plano para o teste precedente e a equipe está familiarizada com ele.
4	*Existe um plano para o teste precedente.*
8	Não existe um plano para o teste precedente.

Quadro 7: Valor teste de precedência.
Fonte: Rios e Moreira, 2006, p. 96.

3.2.3.5 Documentação de teste

Esta variável mede a qualidade da documentação do sistema, na qual o teste irá se basear.

3	Durante o desenvolvimento do sistema são usados padrões de documentação e templates. Há revisões periódicas da documentação
6	*Durante o desenvolvimento do sistema são usados padrões de documentação e templates. Não são comuns revisões periódicas da documentação*
12	A documentação não segue nenhum padrão e nem templates são utilizados

Quadro 8: Valor documentação de teste.
Fonte: Rios e Moreira, 2006, p. 96.

3.2.3.6 Ambiente de desenvolvimento

Esta variável mede a natureza do ambiente no qual o sistema foi desenvolvido.

2	O sistema foi desenvolvido usando uma linguagem de 4º geração integrada a um sistema de gerência de banco de dados
4	*O sistema foi desenvolvido usando uma combinação de linguagem de 4º e 3º geração*
8	O sistema foi desenvolvido em linguagem de 3º geração como COBOL, PASCAL, C++, Delphi, ASP, HTML, etc.

Quadro 9: Valor ambiente de desenvolvimento.
Fonte: Rios e Moreira, 2006, p. 96.

3.2.3.7 Ambiente de teste

Esta variável mede a estrutura do ambiente no qual o sistema foi testado.

1	O ambiente de teste já foi utilizado inúmeras vezes
2	*O ambiente de teste é similar ao que já vinha sendo utilizado anteriormente*
4	O ambiente de teste é completamente novo e experimental

Quadro 10: Valor ambiente de teste.
Fonte: Rios e Moreira, 2006, p. 97.

3.2.3.8 Testware

Esta variável mede a reutilização e a disponibilidade do material de testes. O termo testware, é utilizado em referência a todos os artefatos de testes.

1	Existem materiais de testes, tais como base de dados, tabelas, casos de testes e outros, que poderão ser reutilizados
2	*Existem apenas tabelas e bases de dados disponíveis para reutilização*
4	Não existe testware disponível

Quadro 11: Valor testware.
Fonte: Rios e Moreira, 2006, p. 97.

3.2.4 **Número total de horas de teste (THT)**

O número total de horas final se dá com a correção do total de horas primárias, incluindo as atividades de planejamento e controle, também denominado de índice de planejamento e controle (IPC) como ilustrado abaixo.

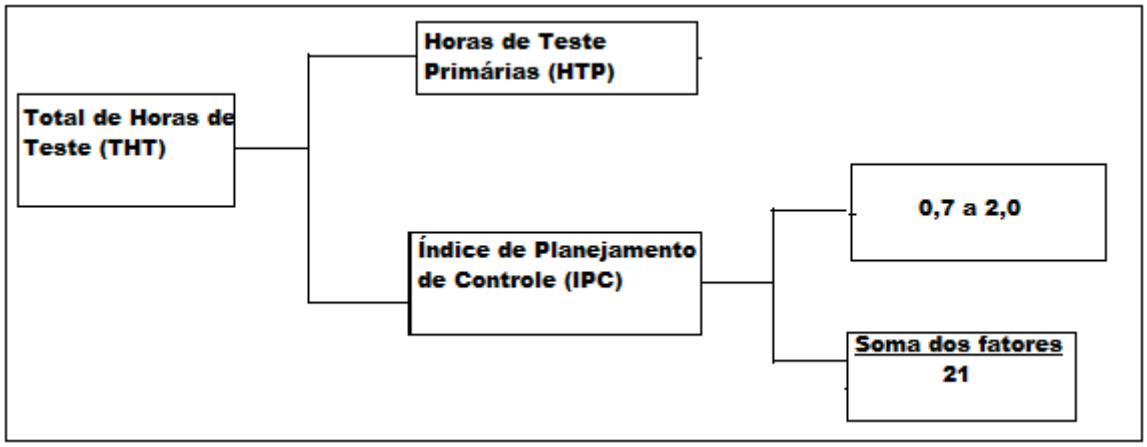

Figura 18: THT – Visão Geral.
Fonte: Adaptado de Rios e Moreira. 2006, p. 97.

O percentual do IPC depende dos fatores abaixo:

- Tamanho da equipe (TE);
- Ferramentas de gerenciamento (FG)

Com o valor do IPC, a fórmula para calcular o THT é:

THT = HTP * IPC

Figura 19: Fórmula THT.
Fonte: Rios e Moreira, 2006, p. 98.

Com o total de horas de testes calculado, deve-se distribuir as horas entre as fases do projeto da seguinte forma:

Preparação	10%
Especificação	40%
Execução	45%
Transição	5%

Quadro 12: Distribuição das horas de testes.
Fonte: Veenendaal e Dekkers, 1999.

Conforme Rios e Moreira (2006), os valores podem ser alterados quando forem coletados os valores históricos, ou houver a necessidade desta alteração, priorizando alguma etapa do processo de testes.

3.2.4.1 Método para cálculo do IPC

O IPC é calculado através da soma dos fatores TE e FG. O resultado dará um valor percentual que será acrescido ao total de horas primárias resultando no total de horas de testes.

A fórmula para cálculo do IPC é:

IPC = 1 + (TE + FG)

Figura 20: Fórmula IPC.
Fonte: Rios e Moreira, 2006, p. 98.

3.2.4.2 Tamanho da equipe (TE)

O tamanho da equipe corresponde ao número de pessoas que compõem a equipe de testes.

0,03	Entre 3 e 4 pessoas (inclusive)
0,06	*Entre 5 e 10 pessoas (inclusive)*
0,12	11 ou mais pessoas

Quadro 13: Valor TE.
Fonte: Rios e Moreira, 2006, p. 98.

3.2.4.3 Ferramentas de gerenciamento (FG)

O ferramental de gerenciamento corresponde o nível em que os recursos estão automatizados nas fases de planejamento e controle.

0,02	Existem ferramentas de registro de tempo e de gerência de defeitos, além de ferramentas de gerência de configuração
0,04	*Apenas uma das ferramentas citadas acima está disponível*
0,08	Não existem ferramentas disponíveis

Quadro 14: Valor FG.
Fonte: Rios e Moreira, 2006, p. 98.

4 ESTUDO DE CASO

Este capítulo contextualiza a organização, aborda a ferramenta para automação dos pontos de testes e o estudo de caso, realizado no sistema SGP.

4.1 EMPRESA ESTUDO DE CASO: XPTO

Para designar a organização estudo de caso será utilizado um nome de fantasia (XPTO) visando resguardar a imagem desta entidade. A XPTO é uma empresa de cunho tecnológico situada em Florianópolis / SC.

De acordo com seu site institucional, é uma das maiores empresas de sistemas de gestão do país. Está no mercado há 21 anos desenvolvendo soluções corporativas para segmentos específicos de negócios.

A empresa possui mais de 1200 clientes no Brasil e no exterior, e mantém alianças com os mais respeitados fornecedores mundiais de tecnologia e convênios com instituições de ensino, adotando uma política de capacitação constante de sua equipe e desenvolvendo projetos de pesquisa e inovação.

O resultado são soluções de alta qualidade e tecnologia com foco no cliente, que por meio da automação, integração e padronização das rotinas de trabalho, prestam agilidade aos processos de negócio, otimizam tempo e economizam recursos.

A seguir, está detalhado o fluxo de desenvolvimento da organização.

4.1.1 **Fluxo de desenvolvimento**

O fluxo de desenvolvimento possui cinco subfluxos: Monitoramento, Preparação, Planejamento, Execução e Encerramento.

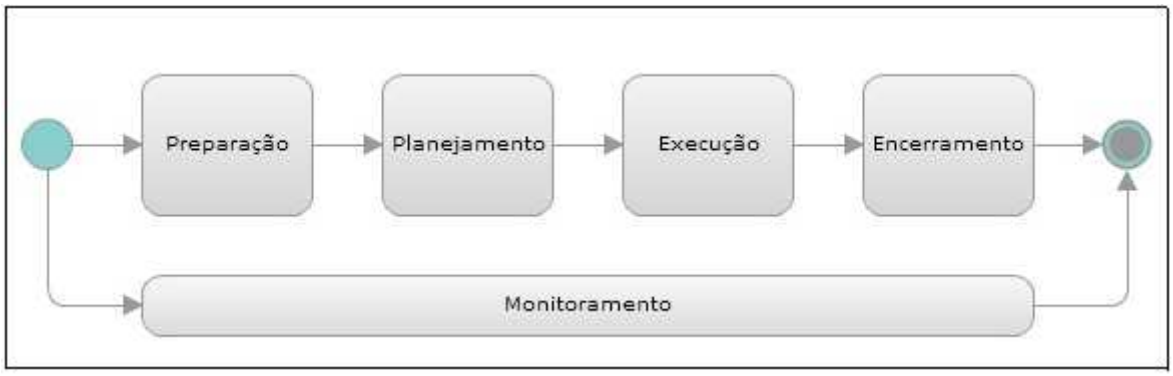

Figura 21: Fluxo de desenvolvimento.
Fonte: Ferramenta wiki da empresa, 2011.

Nesse trabalho é abordado apenas às atividades de testes, que estão relacionadas com os subfluxos: Preparação, Planejamento, Execução e Encerramento.

4.1.1.1 Subfluxo: Preparação

Segundo a wiki da empresa, esse subfluxo tem o objetivo de preparar os artefatos e o ambiente a fim de disponibilizar as ferramentas necessárias para o início do projeto.

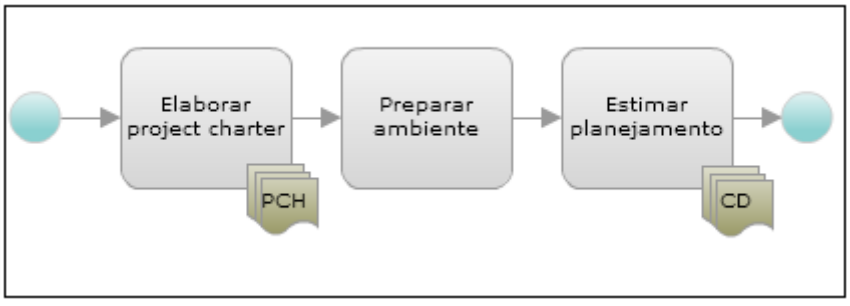

Figura 22: Subfluxo Preparação.
Fonte: Ferramenta wiki da empresa, 2011.

Na fase de Preparação o analista de testes participa apenas da etapa: Estimar planejamento. Nessa fase é solicitada ao analista de testes a estimativa das atividades de testes. As atividades de testes são: elaboração do plano de testes, testes por caso de uso, teste de sistema e geração do relatório de resultado de testes.

A estimativa é feita com base no protótipo e nas especificações dos casos de uso.

Atualmente, essa estimativa é feita por analogia. É utilizado um cenário crítico de uma experiência passada como parâmetro.

O analista de testes procede com a estimativa utilizando a técnica PERT, pessimista, provável e otimista, e a envia para a assistente de projeto.

4.1.1.2 Subfluxo: Planejamento

Como descrito na wiki da empresa, o seu objetivo é descrever os processos necessários e suficientes para referenciar as atividades de levantamento, análise e gerenciamento dos requisitos, garantindo que o cliente e a equipe do projeto mantenham comum entendimento dos requisitos a serem implementados.

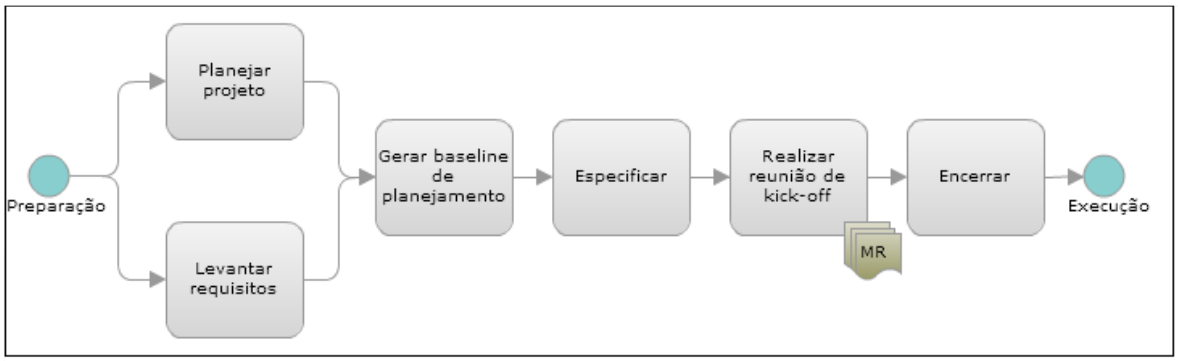

Figura 23: Subfluxo Planejamento.
Fonte: Ferramenta wiki da empresa, 2011.

Nesse subfluxo, o analista de testes participa juntamente com outros envolvidos da reunião de *kick-off*.

Nessa reunião o analista de negócio apresenta as principais funcionalidades do projeto, cliente, restrições e riscos conhecidos até o momento.

A assistente de projeto apresenta as estimativas do projeto e verifica após a apresentação do analista de negócio se todos os envolvidos estão comprometidos com suas atividades e estimativas. Este comprometimento é oficializado na memória de reunião.

4.1.1.3 Subfluxo: Execução

De acordo com a wiki da empresa, o subfluxo execução tem como objetivo descrever os processos necessários e suficientes para nortear as atividades de execução.

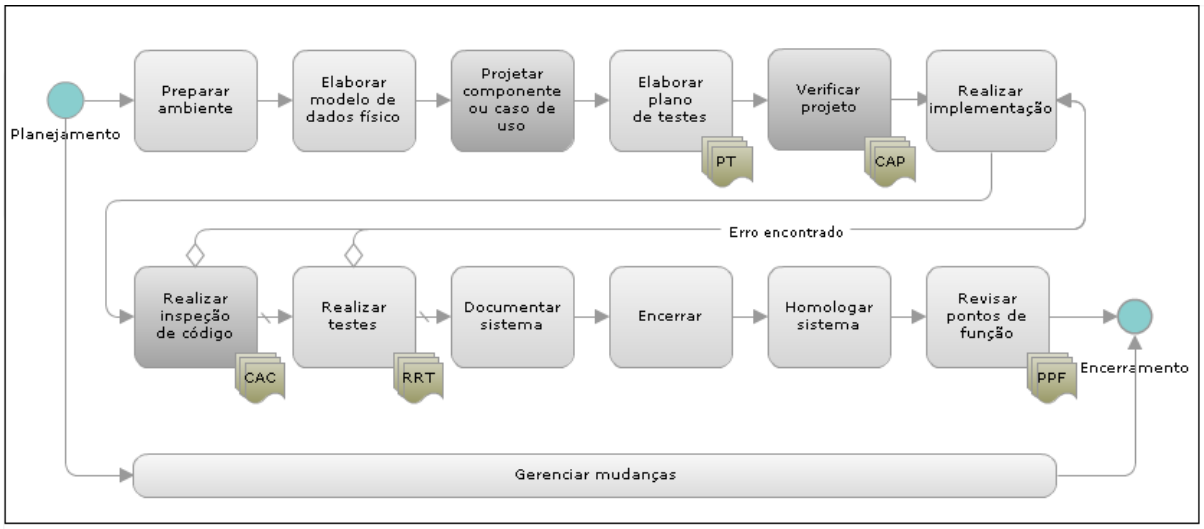

Figura 24: Subfluxo Execução.
Fonte: Ferramenta wiki da empresa, 2011.

Na Execução, as atividades do analista de testes são: Elaborar plano de testes e Realizar testes.

4.1.1.3.1 Elaborar plano de testes

O plano de testes é um artefato do processo de testes. É uma maneira de planejar a execução de testes manuais em software. Conforme apresentado no anexo A, o plano de testes tem as seguintes seções:

- Quadro de revisões: informa o número da revisão, autor, descrição e data;
- Objetivo dos testes: relata o objetivo da atividade de testes;
- Escopo dos testes: informa o que será e o que não será testado;
- Ambiente de testes: descreve o software e hardware que serão utilizados nos testes do sistema;
- Ferramentas: informa as ferramentas utilizadas no processo de testes;
- Referências a documentos importantes: descreve quais são os artefatos importante para construção do plano de testes;
- Padrão de nomenclatura dos casos de testes: informa os padrões adotados pela empresa na confecção dos casos de testes;
- Casos de testes: roteiro de testes criado a partir do documento de especificação de requisitos de software e do protótipo do sistema.

Os casos de testes são importantes no momento da execução dos testes, pois o analista de testes consegue realizar uma sequência de passos de forma prática, sem a necessidade de consultar todos os documentos de especificação de requisitos no momento dos testes, podendo focar apenas na sua execução.

No caso de testes são informados as pré-condições, fluxo-principal, fluxo alternativos, pós-condições, conforme o anexo B.

Além do que é especificado pelo cliente (requisitos funcionais, requisitos não funcionais, regras de negócio), o caso de testes deve conter também procedimentos que testam a eficiência e a corretude do sistema.

4.1.1.3.2 Realizar testes

O objetivo desta atividade é a realização dos testes por parte do analista de testes, no nível teste de sistema, de acordo com os casos de testes criados na atividade: “Elaborar plano de testes”, e dos testes dos desenvolvedores, nos níveis de unidade e de integração.

Para registrar a execução desta atividade, é utilizada uma ferramenta própria da empresa, que neste trabalho está sendo chamada de SGT – Sistema Gerenciador de Testes.

De acordo com o wiki da empresa, o SGT foi concebido com o objetivo de aprimorar a execução dos testes, agilizar o processo de testes e obter estatísticas sobre os testes efetuados.

O SGT também é utilizado na seleção dos casos de testes por parte do analista de teste, porém, tanto o analista de teste quanto o desenvolvedor, o utilizam para registrar os testes que obtiverem o resultado esperado e os que obtiverem um resultado diferente do esperado, defeito.

Caso seja constatada a necessidade de revisar ou melhorar o caso de testes, ainda durante a sua execução, e tal revisão possa ser feita naquele momento, então o caso de testes deve ser analisado, efetuando-se as alterações necessárias. Este item pode ser repetido conforme apropriado.

Caso haja algum defeito nos testes do analista de testes, este é encaminhado ao desenvolvedor para correção. Após a correção, o desenvolvedor providencia a geração de versão e avisa o analista de testes para reaplicação do teste que resultou em defeito.

Após as reaplicações dos testes por caso de uso e estes não apresentam mais defeitos, é executado o teste de regressão em todos os casos de uso.

Ao final das correções e do teste de regressão, o analista de teste gera a partir do SGT, o RRT – Relatório de Resultado de Testes. É este relatório que é enviado ao cliente com o resultado de todos os testes efetuados no sistema.

De acordo com o anexo C, este relatório possui as seguintes seções:

- Quadro de revisões: informa o número da revisão, autor, descrição e data;

- Plano de testes: utilizado para rastrear o plano de testes com o relatório de resultado;
- Aplicação dos testes: descreve todos os testes feitos no sistema. Informa o testador, caso de uso, caso de teste, descrição do teste, situação, descrição da ocorrência e resolução da correção.

4.1.1.4 Subfluxo: Encerramento

Conforme a wiki da empresa, o seu objetivo é elaborar os documentos com as informações finais do projeto, como produtividade, lições aprendidas, desvios, dados de auditorias e tabulações de treinamentos. Informações utilizadas, na reunião de encerramento, para discussão dos pontos fortes e fracos do desenvolvimento do projeto.

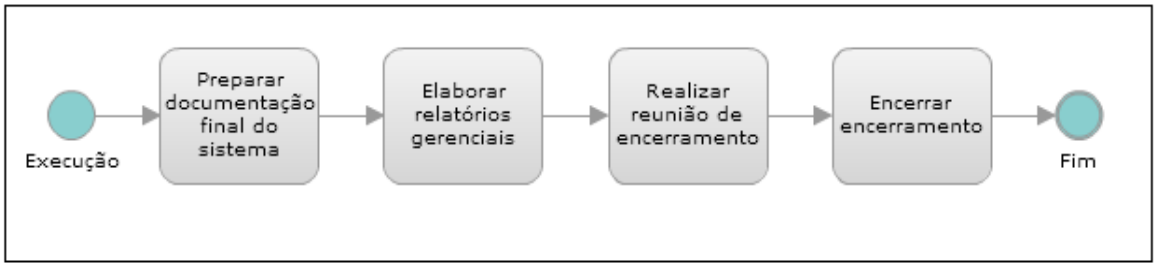

Figura 25: Subfluxo Encerramento.
Fonte: Ferramenta wiki da empresa, 2011.

No subfluxo Encerramento, o analista de testes participa da reunião de encerramento, juntamente com todos os envolvidos no projeto.

Nessa reunião, são apresentadas e discutidas as informações e os números finais coletados, as lições aprendidas, os pontos fortes e fracos do projeto. A partir desta apresentação, a equipe deve participar e adicionar observações no relatório final.

4.2 FERRAMENTA PARA CALCULAR OS PONTOS DE TESTES

A ferramenta pesquisada e selecionada para automatizar o cálculo de pontos de testes é a APT. Esta ferramenta está disponível em: http://www.testadores.com/index.php/web-links/40-programas/293-analise-de-pontos-de-teste-v103.

No site para download, há um tutorial de como instalar e utilizar a ferramenta.

A ferramenta é totalmente *free* e disponibilizada nos idiomas português e inglês. É uma ferramenta bastante intuitiva e possui tópicos de ajuda entre as suas funcionalidades.

No estudo de caso será mostrado o funcionamento da ferramenta.

4.3 SISTEMA ESTUDO DE CASO – SGP

Para designar o sistema estudo de caso, também será utilizado um nome fantasia, neste caso SGP – Sistema Gerenciador de Projetos.

Esta subseção apresenta o estudo de caso baseado no sistema SGP, da empresa XPTO.

O sistema SGP tem como objetivo possibilitar o acompanhamento da execução dos contratos de estudos e projetos de obras Rodoviárias, elaborados por empresas de consultoria, por intermédio do armazenamento das informações técnicas e seus artefatos, além de permitir a comunicação entre os atores envolvidos no que se referem às revisões, aprovações e pareceres nas várias etapas do projeto.

Para proceder com as estimativas no subfluxo “Preparação”, etapa “Estimar Planejamento” da empresa XPTO, o analista de testes procedeu com as estimativas por analogia, técnica utilizada pela empresa, baseada em um cenário crítico anterior, no qual leva em consideração somente a quantidade de casos de uso do sistema.

O sistema possui 28 casos de uso.

A seguir segue as estimativas em horas com a técnica utilizada atualmente no processo da empresa:

Fase	Otimista	Provável	Pessimista
Especificação (plano de teste)	20	24	30
Execução (testes)	110	135	150
Relatório final	03	04	05

Quadro 15: Estimativas SGP.
Fonte: Próprio autor, 2012.

As atividades do fluxo de testes foram realizadas em:

Fase	Horas
Especificação (plano de teste)	22
Execução (testes)	81
Relatório	04

Quadro 16: Horas realizadas SGP.
Fonte: Próprio autor, 2012.

Como pode ser visto nos quadros a cima, as fases utilizadas no fluxo de testes da empresa, não são as mesmas sugeridas pelos autores.

4.3.1 **Utilizando a ferramenta (APT)**

A ferramenta será utilizada para estimar o projeto SGP.

Primeiramente, se faz necessário cadastrar o projeto.

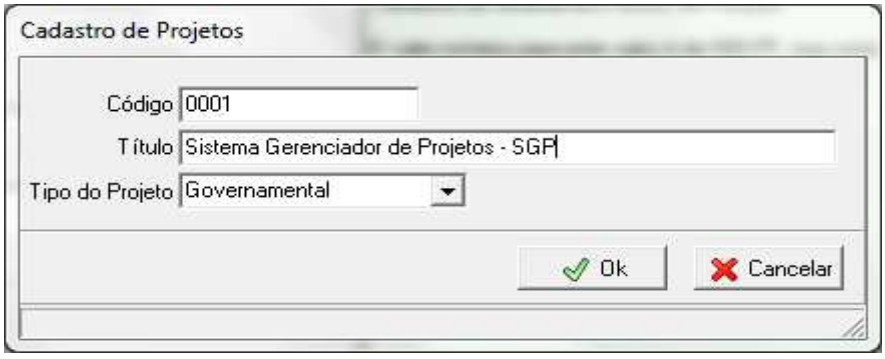

Figura 26: Cadastro do projeto.
Fonte: Próprio autor, 2012.

Após o cadastro do projeto, o sistema apresenta o cadastro dos Pontos de Testes Dinâmicos (Pontos de Função, Funções Dependentes e Qualidade Dinâmica). É possível configurar o modo de trabalho, entre *Wizard* e Geral.

Na primeira aba são informados os pontos de função:

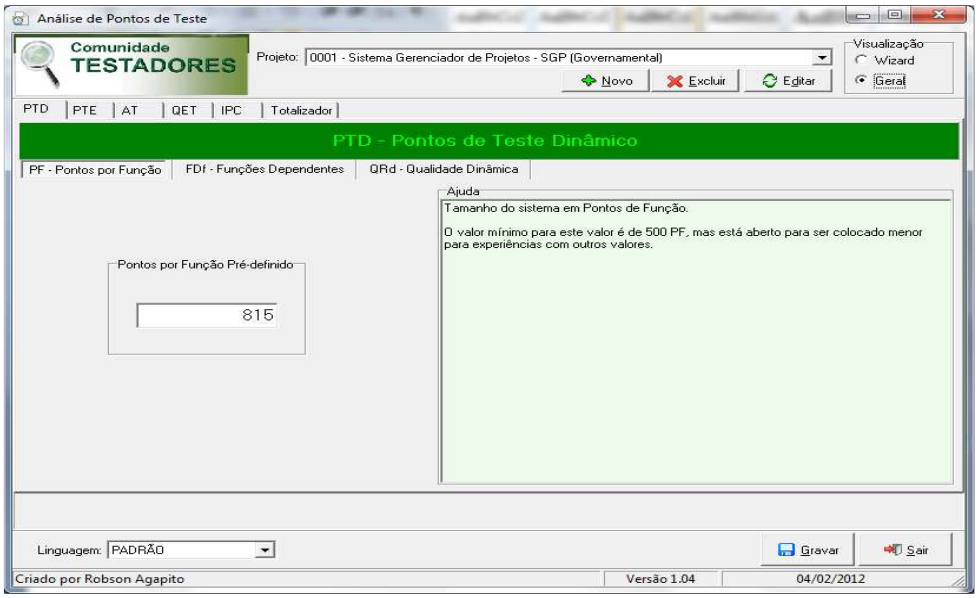

Figura 27: Cadastro PF.
Fonte: Próprio autor, 2012.

Na segunda aba são cadastradas as funções dependentes. No projeto SGP, o sistema foi dividido em 158 funções dependentes. Esta divisão foi baseada no documento de pontos de função que apresenta todas as funções do sistema, a complexidade e a quantidade de arquivos lógicos das funções (interface).

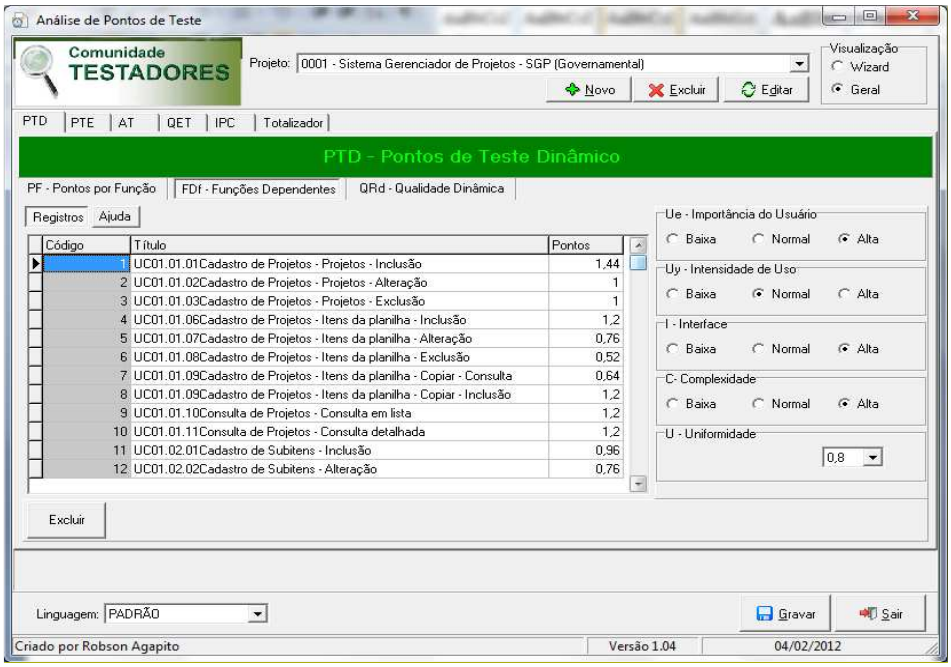

Figura 28: Cadastro FDf.
Fonte: Próprio autor, 2012.

A terceira aba é dividida em características explícitas e implícitas, que somadas, resultam na qualidade dinâmica.

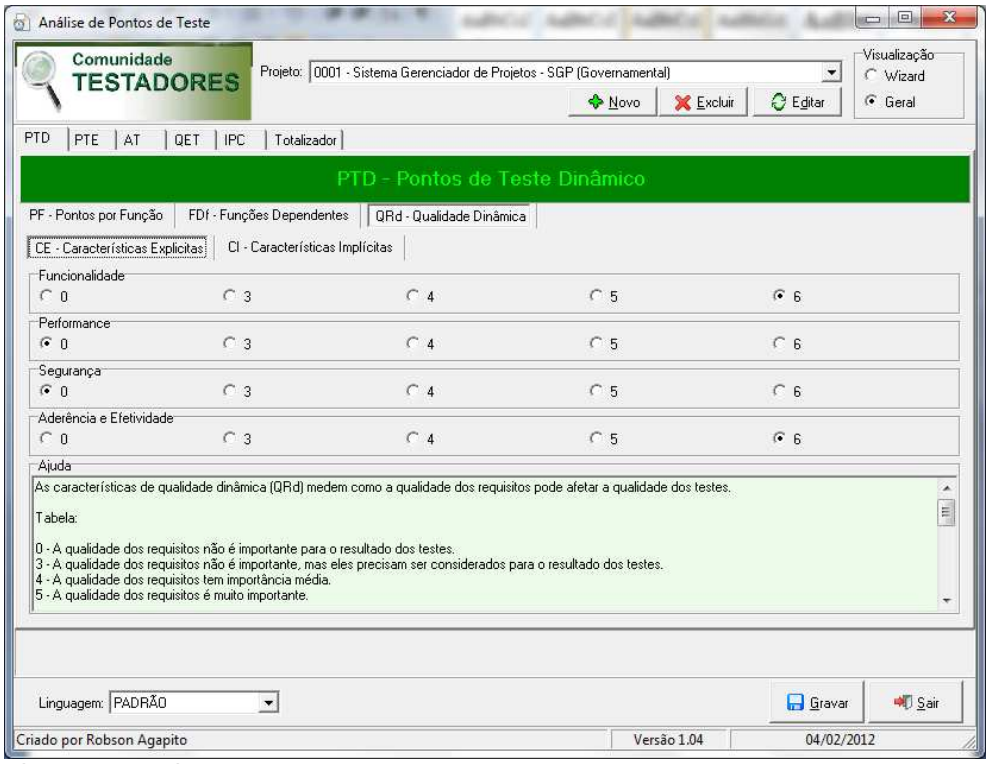

Figura 29: Cadastro CE.
Fonte: Próprio autor, 2012.

Para este projeto são extremantes importantes as características explícitas dos requisitos: Funcionalidade e Aderência.

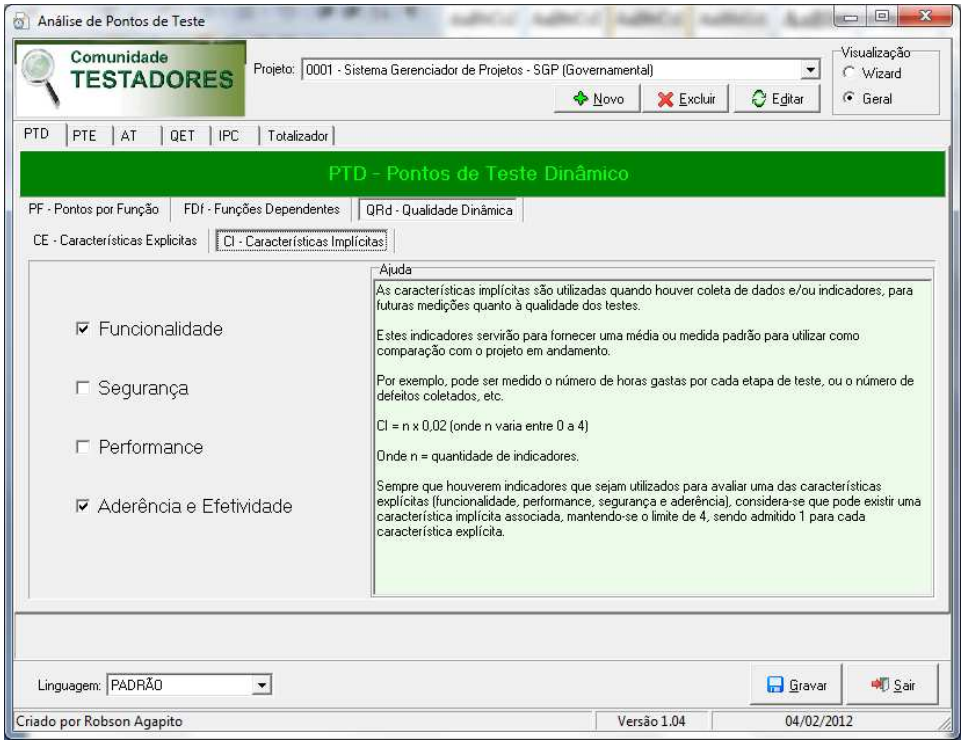

Figura 30: Cadastro CI.
Fonte: Próprio autor, 2012.

Para as características implícitas, também são coletados apenas os indicadores de Funcionalidade e Aderência.

O terceiro cadastro é dos Pontos de Testes Estáticos.

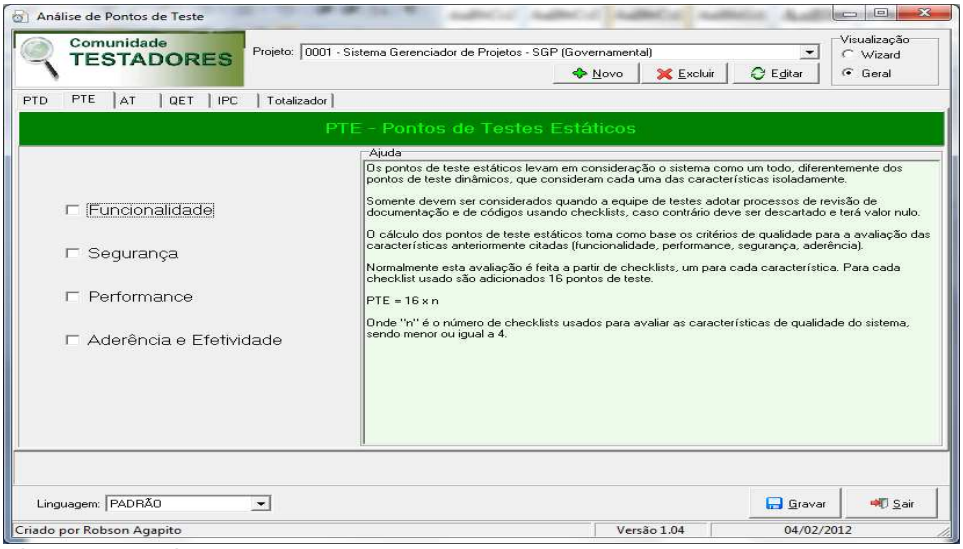

Figura 31: Cadastro PTE.
Fonte: Próprio autor, 2012.

Neste projeto não foi utilizado nenhum *checklist*, por isso não foi marcado nenhuma opção.

Com estes cadastros temos o Total de Pontos de Testes.

A partir do quarto cadastro, têm-se o cálculo da estimativa de horas de testes.

O quarto cadastro é utilizado para informar o Ambiente de Testes. Este cadastro está dividido em duas abas.

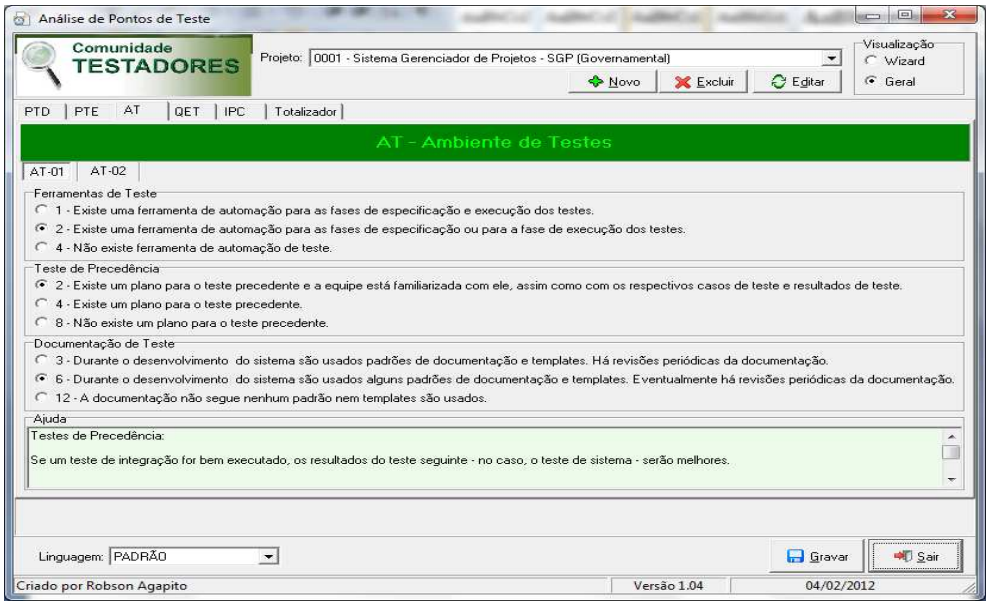

Figura 32: Cadastro AT-01.
Fonte: Próprio autor, 2012.

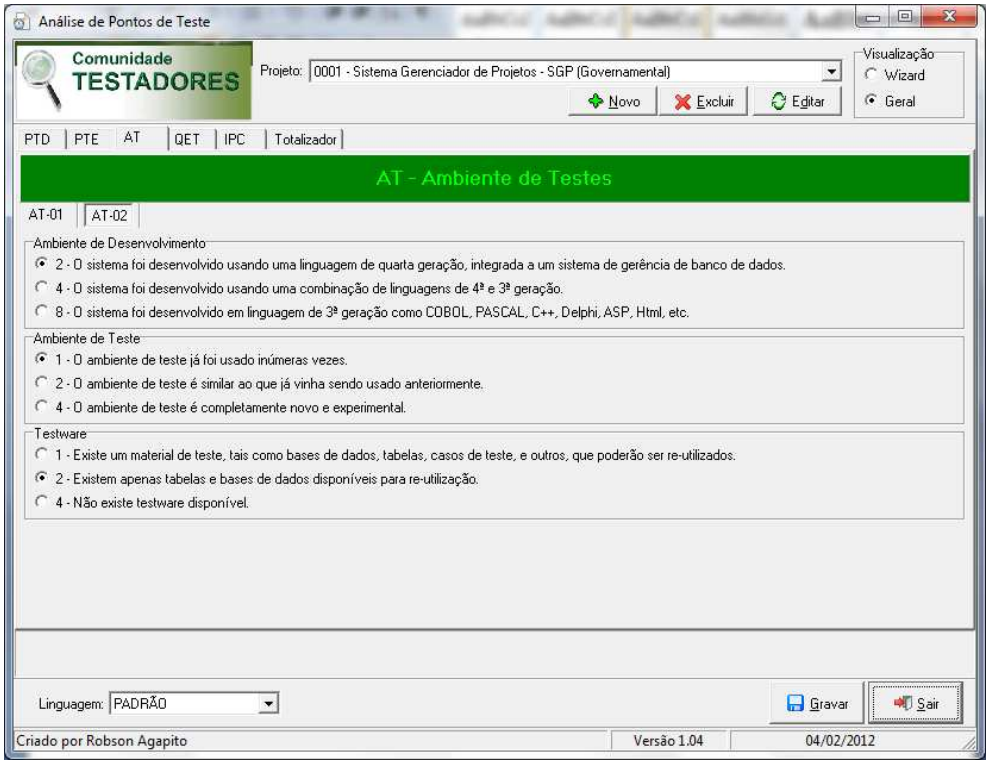

Figura 33: Cadastro AT-02.
Fonte: Próprio autor, 2012.

No quinto cadastro, é informada a qualificação da equipe de testes.

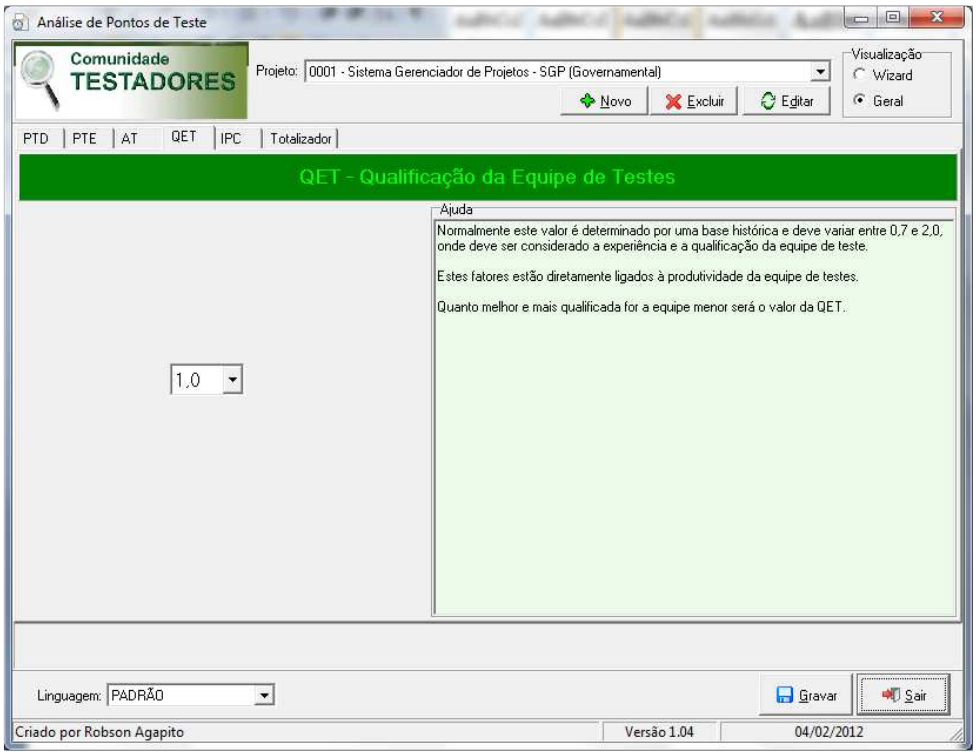

Figura 34: Cadastro QET.
Fonte: Próprio autor, 2012.

Ao final do quinto cadastro, tem-se a estimativa das Horas de Testes Primárias.

No sexto cadastro, é informado o Índice de Planejamento e Controle.

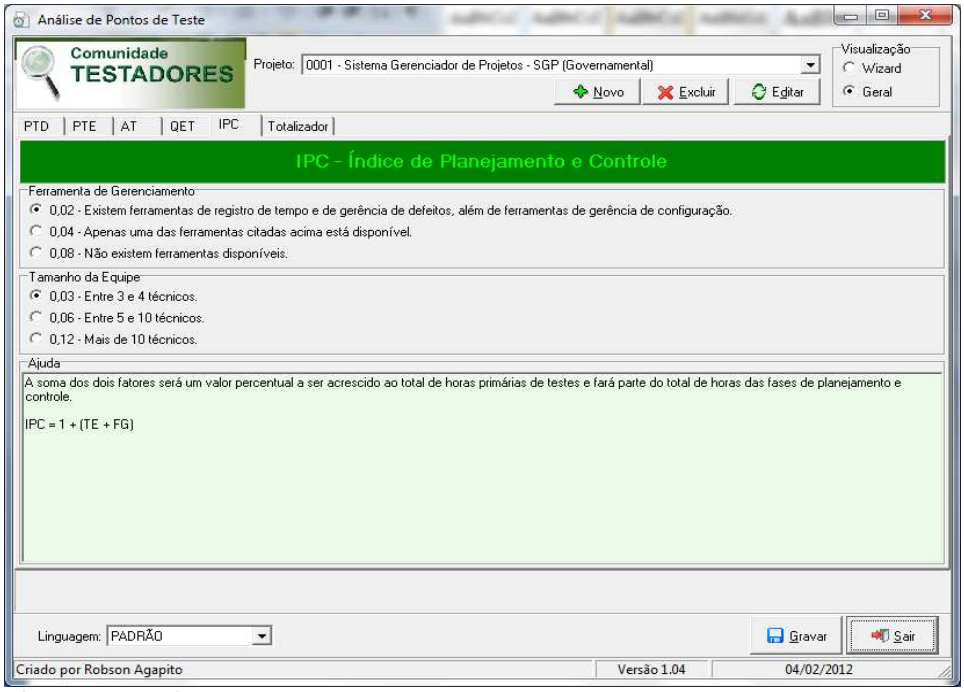

Figura 35: Cadastro IPC.
Fonte: Próprio autor, 2012.

Com o total de horas de testes primárias e com a correção do IPC, se obtém o total de horas de testes.

Na ferramenta APT, a última aba, Totalizador, apresenta todos os cálculos.

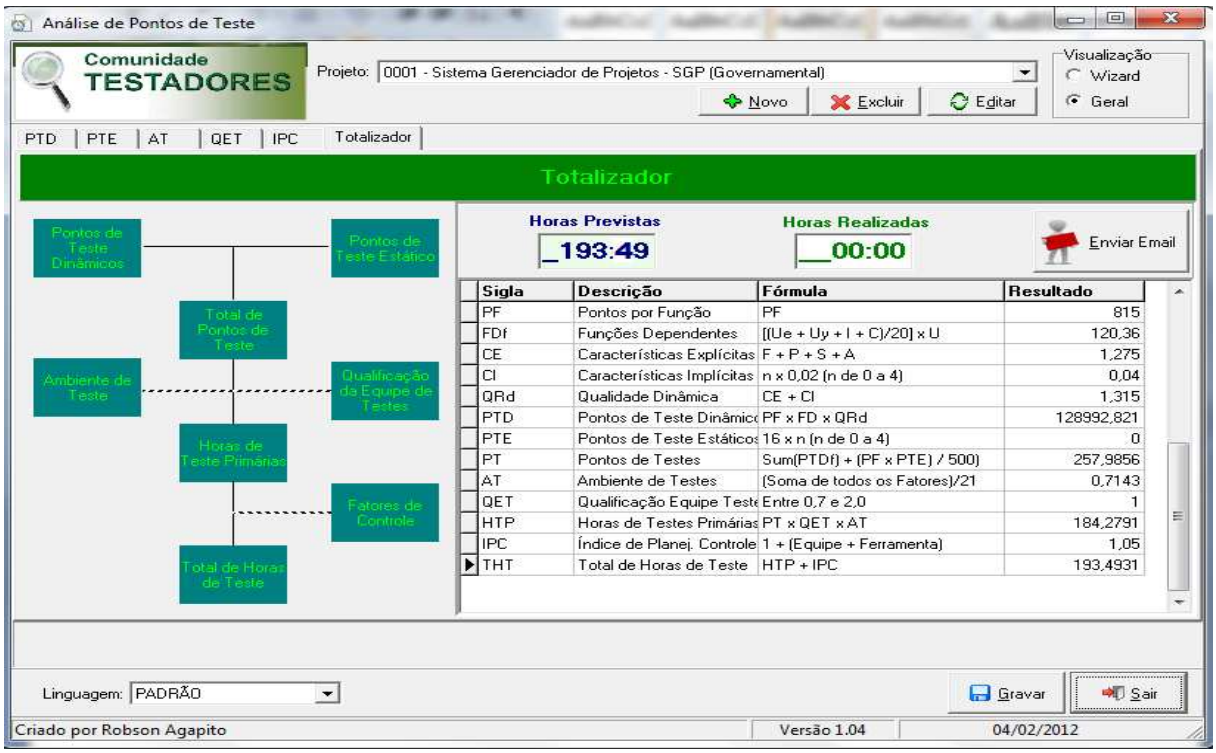

Figura 36: Totalizador.
Fonte: Próprio autor, 2012.

Como pode ser visto na figura a cima, o projeto SGP tem a estimativa das atividades de testes em 193:49 horas.

Distribuindo este total de horas entre as atividades do processo de testes sugerido pelos autores da metodologia de Análise de Pontos de Teste obtém-se:

Preparação	10% - 19:35 horas
Especificação	40% - 77:40 horas
Execução	45% - 88:00 horas
Transição	5% - 8:34 horas

Quadro 17: Distribuição das horas de testes.
Fonte: Próprio autor, 2012.

Comparando os valores obtidos pela estimativa APT e com a técnica por analogia, pode-se ver que a estimativa empírica foi superestimada. Com a técnica APT, a estimativa da tarefa Execução, chegou muito próximo do tempo realizado. A tabela a seguir apresenta os valores calculados para cada fase com a técnica empírica (otimista, provável e pessimista), a técnica APT e os valores realizados.

Tabela 3: Comparativo de horas.

Fase	Otimista	Provável	Pessimista	Realizados	APT
Especificação (plano de teste)	20	24	30	22	77:40
Execução (testes)	110	135	150	81	88:00
Relatório final	03	04	05	04	--
Total	133	163	185	107	165:40

Fonte: Elaboração do autor, 2012.

Com relação à etapa Especificação, que com a ATP ficou muito elevado, acredita-se que com calibragens e uma base histórica, pode-se chegar a um valor mais próximo. Sabe-se que atualmente a especificação dos casos de testes é muito superficial, geralmente são especificados apenas os cenários de testes, não validando nem as regras de negócio, por isso, resultando em um valor muito pequeno com a técnica por analogia e também com as horas efetivamente realizadas.

Já as etapas de Preparação e Transição, não são etapas executadas pelos analistas de testes, por isso não se têm as horas estimadas e nem realizadas.

5 CONCLUSÃO

Deve-se levar em conta que estimar não é uma tarefa fácil, porém, as estimativas das atividades de testes são essenciais para uma boa gerência e controle do projeto. De pouco adianta o desenvolvimento do software estar bem gerenciado e dentro do prazo, se na etapa de testes, as atividades atrasam, se tornando o gargalo do ciclo de desenvolvimento.

Para Rios e Moreira (2006, p.79), “o processo de estimar muitas vezes está intimamente ligado ao nível de produtividade interno da organização e se faz necessário à criação de uma base de informações com um histórico de medições para que se chegue a resultados mais precisos”.

De acordo com Eliza e Lagares (2011), a atividade de estimar sempre deve ser realizada e melhorada, para que após sucessivas melhorias, se possa obter o nível de exatidão esperado.

Como foi visto, a empresa XPTO possui um fluxo de desenvolvimento bem definido, contudo, as atividades de testes ainda não são estimadas com a precisão exigida para um processo de teste bem definido. A técnica utilizada atualmente, como pôde ser visto no estudo de caso, é útil para alguns cenários críticos, mas não possui precisão suficiente. Com isso, foi estudada a técnica APT, que é uma técnica bem definida e voltada para a área de testes de software.

Uma das considerações a serem feitas sobre o uso da APT é que esta requer os valores de muitas variáveis, sobre os quais somente pessoas em contato com o cliente poderão responder, por exemplo, importância do usuário e intensidade de uso das funções.

Através deste estudo, pode-se perceber que a técnica APT não é tão simples de se utilizar, contudo, com o auxílio da ferramenta, fica mais fácil o seu uso. Também é fato, que para estimar as atividades com a APT demandará mais esforço do analista de testes do que utilizar a estimativa por analogia. O analista de testes precisará ter um contato maior com o analista responsável pelo sistema, a fim de elencar as variáveis que dependem do cliente (importância para o cliente, intensidade de uso, entre outras).

Por ser uma técnica baseada em pontos de função, e a empresa já utilizar os pontos de função para medir seus sistemas, o pesquisador acredita que a APT pode ser adotada para estimar as atividades de testes.

Contudo, igualmente a toda mudança na cultura da empresa, é necessário avaliar com cuidado sua aplicação para não causar mais problemas do que soluções.

Para utilização da APT, o projeto precisa estar medido em Pontos de Função. É recomendada apenas para projeto de médio e grande porte, já que os autores indicam o uso somente para projetos com mais de 500 pontos de função.

No entendimento de Bastos et al. (2007), com a utilização da técnica, o processo de teste tem uma medida adequada para determinar seu tamanho, o que é justificado quando o teste é tratado como uma atividade independente, mantendo-se a ligação com o tamanho do sistema em pontos de função.

5.1 TRABALHOS FUTUROS

Para trabalhos futuros, sugere-se continuar com a pesquisa elencando ferramentas que auxiliem na utilização da técnica de APT, assim como realizar um estudo mais aprofundado aplicando a técnica de forma mais sistemática em outros sistemas.

Sugere-se também, o estudo de outra técnica que possa ser utilizada em sistemas de pequeno porte, já que a APT é recomendado para sistemas de médio e grande porte, como foi visto neste trabalho.

REFERÊNCIAS

ALBRECHT, Allan J. **Medindo a produtividade do desenvolvimento de aplicativos.** 1979. Disponível em: <http://www.fattocs.com.br/artigos/MeasuringAplicationDevelopmentProductivity.pdf>. Acesso em: 20 set. 2011.

BASTOS, Aderson et al. **Base de conhecimento em teste de software.** 2. ed. São Paulo: Martins, 2007. 264 p.

BLACK, Rex et al. ***Foundations of software testing:*** *ISTQB certification*. 2. ed. Londres: Cengage Learning, 2008. 258 p.

DEMARCO, Tom. ***Software Engineering:*** An Idea Whose Time Has Come and Gone? IEEE Software, New York, p.95-96, 2009. Julho/Agosto. Disponível em: <http://www.computer.org/cms/Computer.org/ComputingNow/homepage/2009/0709/rW_SO_Viewpoints.pdf>. Acesso em: 05 dez. 2011.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa.** 3. ed. São Paulo: Atlas, 1991.

JAVA MAGAZINE: **Estimativa x Teste de Software**. Rio de Janeiro: Devmedia, n. 92, abr. 2011. Mensal.

MYERS, Glenford J. ***The art of software testing*****.** 2. ed. New Jersey: John Wiley & Sons, Inc., 2004. 234 p.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Software Errors Cost U.S. Economy $59.5 Billion Annually**. Publicado em 28 de Junho de 2002. Disponível em <http://www.abeacha.com/NIST_press_release_bugs_cost.htm>. Acessado em 23 de outubro de 2011.

PERRY, William E.. ***Effective methods for software testing*****.** 3. ed. Indianápolis: Wiley Publishing, Inc., 2006. 973 p.

PRESSMAN, Roger S.. ***Software engineering*****:** a practitioner's approach**.** 5. ed. New York: Mcgraw-hill, 2001. 860 p.

PRESSMAN, Roger S.. **Engenharia de software.** 6. ed. São Paulo: Mcgraw-hill, 2006. 720 p.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de software.** 2. ed. Rio da Janeiro: Alta Books, 2006. 232 p.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da pesquisa e elaboração de dissertação.** 4. ed. Florianópolis: Ufsc, 2005. 138 p.

SOMMERVILLE, Ian. **Engenharia de Software.** 6. ed. São Paulo: Addison Wesley, 2003. 592 p.

VASQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado. **Análise de pontos de função:** medição, estimativas e gerenciamento de projetos de software. 11. ed. São Paulo: Érica, 2011. 232 p.

VEENENDAAL, Erik P. W. M. van; DEKKERS, Ton. Testpointanalysis: a method for test estimation. In: KUSTERS R., A COWDEROY, F. Heemstra; VEENENDAAL E. van (eds), ***Project Control for Software Quality***. Maastricht, The Netherlands: Shaker Publishing BV, 1999. Disponível em: <http://www.erikvanveenendaal.nl/UK/files/Testpointanalysis%20a %20method%20for%20test%20estimation.pdf>. Acesso em: 20 set. 2011.

ANEXOS

ANEXO A – Seções do Plano de Testes

1. Quadro de revisões

Revisão	Autor	Descrição	Data

2. Objetivos dos testes

O Plano de Teste para os casos de uso do sistema **SGP – Sistema Gerenciador de Projetos** tem os seguintes objetivos:

3. Escopo dos testes

4. Ambiente de testes

Os testes serão realizados no(s) seguinte(s) ambiente(s):

4.1 Hardware

4.2 Software

4.3 Ferramentas

As seguintes ferramentas serão empregadas para este projeto:

Item	Ferramentas

5. Referências a documentos relevantes

Número de ordem	Tipo do material	Referência bibliográfica
1		
2		
3		
4		

6. Padrão de nomenclatura dos casos de teste

Os casos de testes serão nomeados da seguinte forma:

.

ANEXO B – Plano de Testes de Caso de Uso

Plano de Teste de Caso de Uso

7.1 SGP

7.1.1 UCXXX - Nome do Caso de Uso

Caso de Teste ***Nome do Caso de Teste***	**Descrição**
	PRÉ-CONDIÇÕES:
	FLUXO-PRINCIPAL:
	FLUXO-ALTERNATIVO 1:
	FLUXO-ALTERNATIVO 2:
	PÓS-CONDIÇÕES

ANEXO C – Relatório de Resultado de Testes

1. Quadro de revisões

Revisão	Autor	Descrição	Data

2. Plano de testes

O presente relatório de resultado do testes de caso de uso foi baseado no plano de teste para os casos de uso do sistema **SGP – Sistema Gerenciador de Projetos** (PDT-XPTO-SGP.2011.0001 - Plano de Testes – SGP – Revisão 01).

3. Aplicação dos testes

Ocorrência dos testes:
Nome do Testador:
Data da execução:

Caso de Uso	Caso de Teste	Descrição do teste	Situação P(problema), S(sucesso), N(não realizado)	Descrição da Ocorrência	Diagnóstico