UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

DANIELE WOLFART

ESTIMATIVA DE TAMANHO DE SOFTWARE POR MEIO DA TÉCNICA DE ANÁLISE DE PONTOS DE FUNÇÃO

MONOGRAFIA DE ESPECIALIZAÇÃO

MEDIANEIRA
2012

DANIELE WOLFART

ESTIMATIVA DE TAMANHO DE SOFTWARE POR MEIO DA TÉCNICA DE ANÁLISE DE PONTOS DE FUNÇÃO

Monografia apresentada como requisito parcial à obtenção do título de Especialista na Pós Graduação em Engenharia de Software, da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Medianeira.

Orientador: Prof. M.Sc. Alan Gavioli

MEDIANEIRA

2012

Dedico este trabalho a minha família.

AGRADECIMENTOS

Agradeço a Deus pela família que me deste, pela minha saúde e pelas oportunidades me proporcionadas até hoje.

Agradeço aos meus pais pela minha vida e pela educação que me deram.

Agradeço em especial aos meus irmãos que foram fundamentais para que a conclusão deste trabalho tenha sido realizada.

Agradeço ao meu orientador, professor Alan Gavioli, pela ajuda proporcionada e pela compreensão e apoio para a finalização deste trabalho.

Ninguém vai bater mais forte do que a vida. Não importa como você bate, mas sim o quanto aguenta apanhar e continuar lutando, o quanto pode suportar e seguir em frente. É assim que se ganha.

"Rocky Balboa".

RESUMO

WOLFART, Daniele. Estimativa de tamanho de software por meio da técnica de Análise de Pontos de Função. 2012. 88. Monografia (Especialização em Engenharia de Software). Universidade Tecnológica Federal do Paraná, Medianeira, 2012.

Um dos maiores problemas encontrados no processo de desenvolvimento de sistemas é concluir projetos com qualidade, dentro dos prazos e orçamentos previstos. Tendo conhecimento do que e quanto precisa ser produzido aumenta indiscutivelmente o sucesso do gerenciamento e finalização com êxito dos produtos de *software*. Desta forma este estudo tem por objetivo definir o conceito de estimativa de *software* e sua aplicabilidade no processo de desenvolvimento, como sendo uma ferramenta de apoio ao planejamento e gerenciamento de sistemas visando o aumento da qualidade do produto final. Além de apresentar os tipos de estimativas disponíveis no processo de estimar sistema, o trabalho busca identificar quais são as principais técnicas de estimativa de tamanho de *software* disponíveis no mercado, sua aplicabilidade e suas principais vantagens, para auxiliar no conhecimento da dimensão dos produtos a serem desenvolvidos. O trabalho também apresenta a definição, o funcionamento e as vantagens da técnica de estimativa de tamanho de *software*: Análise de Pontos de Função. Por fim, é apresentado um estudo de caso para demonstrar a aplicabilidade desta técnica, através da obtenção da estimativa de tamanho de um sistema real.

Palavras-chave: Estimativa; Pontos de Função; Métricas.

ABSTRACT

WOLFART, Daniele. *Estimating software size using the technique of Function Point Analysis*. 2012. 88. Monografia (Especialização em Engenharia de Software). Universidade Tecnológica Federal do Paraná, Medianeira, 2012.

One of the bigger problem encountered in the process of system development is to complete projects with quality, on time and budgets provided. Having knowledge of what and how much needs to be produced arguably increases the success of the management and successful completion of the systems. Therefore, this study aims to define the concept of software estimation and its applicability in the development process, as a tool to support planning and management systems to increase the quality of the final product. Besides presenting the types of available estimates in the process of estimating system, the paper seeks to identify which are the main techniques for estimating size of software available on the market, its applicability and its main advantages, to assist in understanding the size of the systems to be developed. The paper also presents the definition, operation and advantages of the technique to estimate software size: Function Point Analysis. Finally, we present a case study to demonstrate the applicability of this technique, by obtaining the estimated size of a real system.

Keywords: *Estimate; Function Points; Metrics.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AIE	-	Arquivo de Interface Externa
ALI	-	Arquivo Lógico Interno
APF	-	Análise de Pontos de Função
AR	-	Arquivo Referenciado
BFPUG	-	*Brazilian Function Point Users Group*
CBP	-	Controle de Bens Patrimoniais
CE	-	Consulta Externa
CFPS	-	*Certified Function Point Specialist*
CGS	-	Características Gerais de Sistema
COSMIC	-	*Common Software Measurement International Consortium*
CPM	-	*Counting Practices* Manual / Manual de Práticas de Contagem
EE	-	Entrada Externa
FFP	-	*Full Function Points*
FSM	-	*Functional Size Measurement*
GIT	-	Grau de Influência Total
GUIDE	-	*Grupo de Usuários IBM*
IBM	-	*International Business Machines*
IEC	-	*International Engineering Consortium*
IFPUG	-	*International Function Point Users Group*
ISO	-	*International Organization for Standardization*
JTC1	-	*Joint Technical Committee One*
LOC	-	*Lines of Code*
PF	-	*Ponto de Função*
SC7	-	*Sub-Committee Seven*
SE	-	Saída Externa
TD	-	Tipo de Dado
TI	-	Tecnologia de Informação
TR	-	Tipo de Registro
UCP	-	*Use Case Points*
UKSMA	-	*United Kingdom Software Metrics Association*

VAF	-	Valor do Fator de Ajuste
XP	-	*Extreme Programming*
WG12	-	*Working Group 12*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, o sucesso das organizações de desenvolvimento de *software* no mercado globalizado e competitivo que é o da área de Tecnologia de Informação (TI), está diretamente ligado ao nível da qualidade do produto final, baseado na eficiência, entrega do produto no prazo estabelecido e dentro do orçamento previsto (MONTEIRO, 2005).

Uma das maiores dificuldades encontradas no gerenciamento de projetos de informática é saber a dimensão do que está sendo gerenciado. Muitas aplicações que a princípio parecem pequenas, ao longo do desenvolvimento mostram-se muitas vezes maiores do que o previsto inicialmente ou, em alguns casos, torna-se tão complexas e grandes, que se perde o controle. Devido a isso os mecanismos de acompanhamento e avaliação do progresso do processo, projeto e produto, geralmente baseados em informações qualitativas e quantitativas, coletadas através de medições, são recursos essenciais e cada vez mais importantes na gestão do desenvolvimento de *software* (SIMÕES, 2004; MONTEIRO, 2005).

Segundo Barcellos (2010), a medição de *software* é considerada uma das atividades mais importantes para a gerência e melhoria de processos e produtos de *software*, uma vez que ao serem coletadas e armazenadas, podem ser analisadas através de métodos e fornecem informações importantes para a tomada de decisão, servindo também como subsídios para a elaboração de planos mais realistas para os projetos.

Para o gerente do projeto, essas medidas, também conhecidas como métricas permitem um melhor entendimento do processo de produção e do controle do projeto de *software* e são de extrema importância para gerar o produto final com a máxima qualidade. Exatamente essa necessidade de medidas que retratem a eficiência do desenvolvimento e minimizem os fracassos de projetos, principalmente em relação a falhas no cronograma e em estimativas realizadas, que demandaram a realização de estudos na área de estimativas de *software* (MONTEIRO, 2005).

Os atos de medir e de estimar estão entre as partes mais importantes de um projeto de sistema bem-sucedido e alguns fatos como: a falta de maturidade, o desinteresse das empresas de desenvolvimento de sistemas e a baixa popularidade deste assunto entre os profissionais da área de informática são algumas das

principais causas para o insucesso e o alto custo dos sistemas de informação (GOMES, 2003).

Para Gomes (2003) o termo métrica de *software* refere-se à mensuração dos indicadores quantitativos do tamanho e complexidade de um sistema utilizados para comparar contra os desempenhos observados (qualitativos) no passado a fim de derivar previsões de desempenho futuro.

Segundo Hazan (2009) e Freire (2008), as estimativas de tamanho, esforço, prazo e custo constituem o primeiro tipo de análise quantitativa que deve ser executado para melhorar o planejamento e o acompanhamento de projetos de *software*, pois são as estimativas de má qualidade que constituem uma das principais causas dos projetos cancelados. Já as estimativas inadequadas podem levar a prazos não cumpridos, custos excessivos com horas extras e má qualidade do *software*.

Este trabalho abrangerá o estudo de estimativa de tamanho de *software*, que segundo Vazquez *et al.* (2010) é uma das propriedades mais importantes do desenvolvimento de *software* a ser medida, com foco para a técnica de Análise de Pontos de Função.

1.1 OBJETIVO GERAL

Demonstrar o funcionamento e a aplicabilidade da técnica de Análise de Pontos de Função como um meio de estimar tamanho de *software*.

1.2 OBJETIVOS ESPECÍFICOS

- Compreender os conceitos, os tipos e as técnicas de estimativa de *software*;

- Compreender a técnica de Análise de Pontos de Função, identificando sua aplicabilidade.
- Obter a estimativa de tamanho de um sistema aplicando a técnica de Análise de Pontos de Função.

1.3 JUSTIFICATIVA

Segundo Barcellos (2010), os avanços tecnológicos e a alta competitividade do mercado estão continuamente aumentando a demanda por *softwares* cada vez melhores e que sejam produzidos em projetos aderentes aos custos e prazos planejados. Porém, a tendência de projetos de TI, voltados a desenvolvimento de *software*, é apresentarem falha nas estimativas de tempo e esforço e principalmente custo, ou seja, projetos com atraso na entrega ou que estejam fora do orçamento inicial (TORRES, 2004; GERMANO, 2008). Segundo Santana; Gusmão (2010) esta característica já vem de longa data, apresentada por Pressman (2006) *apud* Hazan (2009) como um dos três sintomas da Crise de *Software*, por volta da década de 80, quando os números eram ainda piores (conforme Figura 1). Foi a partir deste momento que surgiu a preocupação em identificar as causas e obter melhores resultados nos projetos de desenvolvimento de *software*.

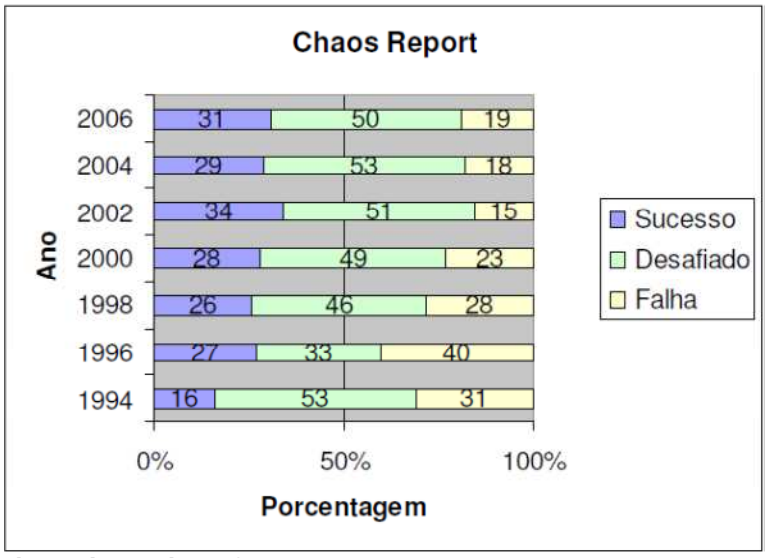

Figura 1 - Análise de Projetos de Softwares
Fonte: SANTANA e GUSMÃO (2010)

Para Torres (2004) essa característica apresentada por projetos de *softwares* de falharem em estimativas de tempo, esforço e custo são resultado de previsões

baseadas apenas em parâmetros qualitativos que acabam sendo mal gerenciados, e diz:

> *"Se compararmos com projetos de Engenharia Civil, seria como se soubéssemos o que deve ser feito em uma casa a ser construída (como será o acabamento e o piso, quais serão as cores das paredes, se será um sobrado ou uma casa térrea, se terá piscina ou não, se terá quartos e suítes, etc.), porém não saberíamos que tamanho terá cada cômodo e conseqüentemente, qual será o tamanho total em metros quadrados da casa construída. Somente saberíamos ao final do projeto qual o tamanho do software"* (TORRES, 2004).

A falta de um parâmetro quantitativo que diga o tamanho do *software* a ser construído gerou a necessidade de se estabelecer um sistema de medição para prover um mecanismo de quantificação do que acontece no processo de desenvolvimento de *software*, possibilitando a analogia com outros *softwares* anteriormente desenvolvidos (TORRES, 2004). Esta necessidade foi formalizada por De Marco (DEMARCO, 1982 *apud* SANTANA e GUSMÃO, 2010) através da frase: "*Não podemos controlar aquilo que não podemos medir*". Assim, baseado no tamanho de um *software* e no tempo que foi gasto para sua conclusão, é possível estimar mais precisamente quanto tempo será preciso para desenvolver um novo, com características ou cenários idênticos, daí a grande importância em se estudar e entender estimativa de *software*.

Para saber o custo de um projeto de *software* é preciso saber o esforço necessário para desenvolvê-lo e para determinar o esforço é preciso mensurar o tamanho do projeto de *software*. Desta forma, determinar o tamanho de um projeto de *software* é uma das primeiras e principais atividades relacionadas a estimativas a serem efetuadas durante o clico de vida do projeto (MACORATTI, 2005).

Além disso, a falta de previsibilidade de custo e prazo de projetos de *software* pode levar a conseqüências desastrosas, tais como: conflitos entre o gerente do projeto e a equipe, baixa estima da equipe, entrega de *software* de baixa qualidade, perda de imagem da organização, e até mesmo o cancelamento do projeto. Assim, torna-se importante o investimento na implantação de um processo de estimativas efetivo, visando à melhoria da previsibilidade de custo, prazo e esforço (HAZAN, 2009).

1.4 ESTRUTURA DO TRABALHO

Para facilitar a compreensão do estudo, esse trabalho está organizado em seis capítulos. O primeiro apresenta uma introdução ao conceito de métricas de *software* e porque houve a necessidade de se realizar estimativas de *software*. Também apresenta os objetivos gerais e específicos do presente trabalho, bem como a justificativa deste trabalho e uma breve apresentação do tema a ser tratado nos demais capítulos.

O capítulo II apresenta o conceito, os tipos e técnicas de estimativa de *software*.

No capitulo III será abordada a técnica de Análise de Pontos de Função.

No capítulo IV é feito um estudo de um sistema para a aplicação da técnica de APF na realização da estimativa de tamanho deste *software*.

O capítulo V apresenta os resultados e discussão da aplicação da técnica de contagem de PF.

No capítulo VI são relatadas as considerações finais sobre o desenvolvimento do presente trabalho e as sugestões para trabalhos futuros relacionados ao tema.

2 ESTIMATIVA DE SOFTWARE

Realizar estimativas é uma das atividades mais desafiadoras e importantes em um projeto de *software*. O planejamento e controle de um projeto não seria possível sem a realização de estimativas confiáveis, que apóiam principalmente as atividades de gestão de projetos de *software* (DEMARCO, 1991 *apud* MACORATTI, 2005; MONTEIRO, 2005).

O propósito principal de um processo de estimativa é disponibilizar informações que beneficiem o planejamento e o controle dos projetos de *software* o mais cedo possível durante seu ciclo de vida (VAZQUEZ *et al.*, 2010). Se bem elaboradas permitem a verificação da viabilidade do projeto, a elaboração de propostas técnicas e comerciais, a confecção de planos e cronogramas detalhados e o acompanhamento efetivo de projetos (MONTEIRO, 2005).

Silveira (2000) define o termo estimativa como sendo a medida aproximada de tamanho, esforço, duração e equipe necessária para o desenvolvimento de um projeto.

Antes de iniciar um projeto, deve-se estimar o trabalho a ser realizado (tamanho/esforço), os recursos necessários, o tempo de duração (cronograma) e, por fim, o custo do projeto. Apesar da realização de estimativa ser uma tarefa trivial, muitas vezes, um pouco arte, um pouco ciência, é uma atividade muito importante que não deve ser conduzida desordenadamente (CARVALHO *et al*., 2006).

Cada projeto de *software* tende a se constituir em uma experiência única devido a particularidades em relação aos seus requisitos, tecnologia empregada e a equipe de trabalho. Diante desta realidade, definir com exatidão o custo final e a data de conclusão de um projeto de *software* somente é possível quando o projeto estiver finalizado. Toda a atividade e cálculo feito durante o ciclo de vida do projeto são estimativas. Contudo as informações obtidas através de estimativas, mesmo não sendo totalmente precisas, são de grande valia para constituir uma coleção de dados históricos que pode ser usada para extrair indicadores de produtividade e qualidade cada vez mais próximos da realidade e cada vez mais confiáveis (VAZQUEZ *et al.*, 2010).

2.1 O PROCESSO DE ESTIMATIVA DE SOFTWARE

Segundo Vazquez *et al.* (2010), o processo de estimativa de um projeto de *software* envolve basicamente quatro atividades:

1. Estimar o tamanho do produto a ser gerado;
2. Estimar o esforço empregado na execução do projeto;
3. Estimar a duração do projeto;
4. Estimar o custo do projeto.

Determinar o tamanho do produto de *software* tem sido tomado como o ponto de partida para a realização de estimativas, pois com base nas estimativas de tamanho, é possível chegar à estimativa de esforço e, a partir dessas, fica mais fácil estimar recursos, tempo e custo (CARVALHO *et al.*, 2006). Porém, antes de iniciar o processo de estimativa de tamanho, é necessário decidir a unidade de medida a ser utilizada (VAZQUEZ *et al.*, 2010).

Após a definição da unidade de medida de tamanho a ser utilizado, o processo de estimativa conforme mostra a Figura 2, tem início com a estimativa de tamanho do projeto tendo como base a declaração do escopo do sistema. Como as estimativas devem ser realizadas no início do processo de desenvolvimento de *software*, então, o principal insumo ou artefato de entrada a ser utilizado é frequentemente um documento inicial de requisitos ou até mesmo um documento do próprio cliente (HAZAN, 2009; MACAROTTI, 2005).

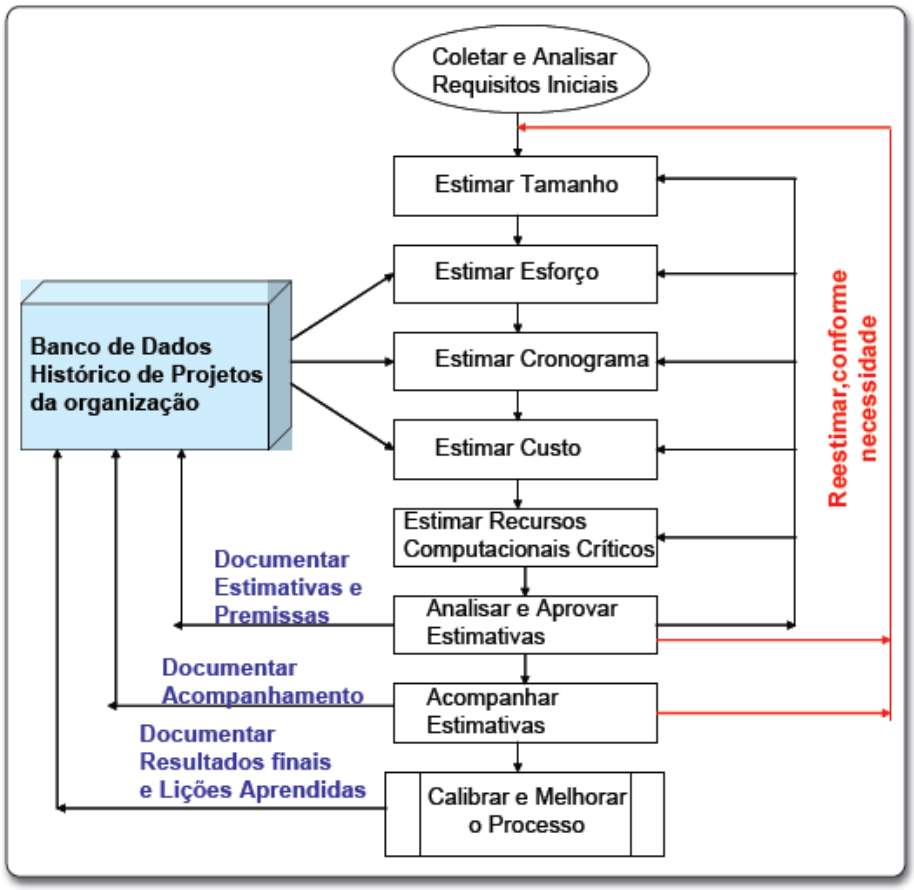

Figura 2 - Processo de Estimativas de Projetos de *Software*
Fonte: HAZAN (2009)

O responsável pelas estimativas deve analisar os requisitos para garantir a qualidade e então estimar o tamanho do projeto de *software* de acordo com a unidade de medida escolhida. O próximo passo é a derivação das estimativas de esforço, prazo (cronograma) e custo (orçamento). Para que as estimativas sejam completas e eficientes, devem ser feitas com base na estimativa de tamanho e levar em consideração as experiências e os dados históricos de projetos passados, os recursos disponíveis dentro e fora da organização, os dados de custo e os fatores de riscos que envolvem os projetos (HAZAN, 2009; VAZQUEZ *et al.*, 2010).

A cada etapa realizada, as estimativas obtidas devem passar por um processo de aprovação, antes de serem registradas na base histórica. Além de poderem ser utilizadas como fonte de informação para projetos futuros, servirá como insumos para as estimativas das etapas seguintes (VAZQUEZ *et al.*, 2010).

No decorrer do processo de desenvolvimento, as estimativas devem ser acompanhadas conforme o refinamento dos requisitos. Conforme mais informações forem sendo obtidas, as estimativas iniciais podem e devem ser refeitas e refinadas. Re-estimar deve ser uma atividade constante durante todo o ciclo de vida do desenvolvimento do *software*, principalmente se ocorrerem mudanças significativas nos requisitos funcionais ou não funcionais (HAZAN, 2009; MACORATTI, 2005).

Quando o projeto é concluído, devem-se documentar as medidas reais de tamanho, prazo, custo, esforço e recursos utilizados, assim como outros atributos relevantes do projeto, visando à coleta de dados para a melhoria do processo de estimativas (VAZQUEZ *et al.*, 2010).

2.2 TIPOS DE ESTIMATIVA

As seções seguintes abrangerão as principais formas de estimativas de *software*: tamanho, esforço, tempo e custo.

2.2.1 ESTIMATIVA DE TAMANHO

"*Estimativa de tamanho de software é um processo pelo qual uma pessoa ou um grupo de pessoas estima o tamanho de um produto de software*" (McPHEE, 1999 *apud* ANDRADE, 2004).

O tamanho do produto de *software* é o ponto de partida para a realização de estimativas, pois são as estimativas de tamanho que constituem o insumo principal para a maioria dos modelos usados para estimar custo, esforço e cronograma e conseqüentemente a precisão da estimativa de tamanho é fundamental (FIORINI *et al.,* 1998 *apud* CARVALHO *et al.*, 2006; STAA; HAZAN, 2005).

O tamanho do *software* significa a quantidade de trabalho a ser executado no desenvolvimento de um projeto em uma unidade de medida especificada. Cada projeto pode ser estimado de acordo com o tamanho físico (geralmente medidos de acordo com os requisitos do usuário); com base nas funções que o usuário obtém,

na complexidade do problema que o *software* irá resolver e no reuso do projeto, que mede o quanto o produto será copiado ou modificado a partir de outro produto existente (MONTEIRO, 2005; ANDRADE, 2004)

Segundo Ross (S.d.) *apud* Freire (2008), determinar o tamanho de um projeto de *software* é uma das primeiras e principais atividades relacionadas às estimativas a serem efetuadas durante o ciclo de vida do projeto, além de ser uma das atividades mais difíceis de serem realizadas em uma organização de desenvolvimento de *software*, principalmente quando é realizada no início do projeto

Quanto maior for o conhecimento e o detalhamento das informações sobre o projeto melhor serão os subsídios para a elaboração da estimativa de tamanho. As estimativas geralmente acontecem em fases iniciais de um projeto, quando o entendimento sobre o que deve ser desenvolvido ainda é incipiente. Por isso, em fases posteriores, a estimativa deve ser refeita quando mais informações sobre o projeto tornarem-se disponíveis (MONTEIRO, 2005)

As primeiras métricas de estimativa de tamanho de *software* surgiram em 1950/1960 e se basearam no tamanho físico de linhas de código como *o Lines of Code* (LOC). Esta métrica considera o *software* sob a perspectiva da estrutura interna e é aplicada nas fases finais do projeto (ANDRADE, 2004).

Atualmente, existem diversas métricas de estimativa de tamanho de projetos de *software*, e há dificuldade para selecionar a mais apropriada para o tamanho de um projeto de *software* de uma organização. As métricas mais conhecidas e utilizadas foram desenvolvidas com base nas funções de *software* tais como: *Function Point Analysis* (Análise por Pontos de Função); *COSMIC Full Function Point, Internet Points, Domino Points, Use Case Points* (Pontos de Caso de Uso) (ANDRADE, 2004; MONTEIRO, 2005). Essas métricas serão apresentadas no decorrer deste capítulo.

2.2.2 ESTIMATIVA DE ESFORÇO

O esforço para desenvolver um novo sistema pode ser definido como a quantidade de trabalho necessária para completar uma atividade ou outro elemento

de um projeto e geralmente é expresso como um total de horas, dias, meses ou semanas gastos por um grupo de pessoas na realização de suas atividades (FREIRE, 2008; HAZAN, 2009). Freire (2008) ressalta que o esforço não deve ser confundido com duração, que pode ser definida como o tempo necessário para completar uma atividade ou outro elemento de um projeto.

O esforço estimado para o projeto geralmente é obtido a partir do cálculo da estimativa de tamanho do produto. O esforço total de um projeto é calculado com base em seu processo de desenvolvimento, que envolve não somente a atividade de codificação do *software*, mas também de elaboração de documentos, implementação de protótipos, projeto do produto a ser entregue, revisão e teste do código (PETERS, 1999 *apud* MONTEIRO, 2005).

Segundo Gomes (2003) e Silveira (2000), uma técnica para auxiliar na melhor precisão da estimativa de esforço necessário para o desenvolvimento do produto, é realizar esta estimativa para o maior nível de detalhamento das fases do projeto, ou seja, se possível aplicada a cada tarefa do projeto para cada função de *software*.

Para estimar o esforço e prazo, é preciso que seja selecionada uma abordagem para a obtenção da estimativa (modelos paramétricos, analogia, julgamento de especialistas, modelo de estimativas baseado em atividades, relações simples de estimativas) (FREIRE, 2008). Segundo Vazquez *et al.* (2010), a melhor maneira de se chegar à estimativa de esforço a partir da estimativa de tamanho do projeto é utilizar-se de dados internos da própria organização, como o número de horas alocadas em projetos passados, conjuntamente com o tamanho do projeto, dimensionado em suas diversas fases.

A estimativa de esforço é de extrema importância, pois com ela já definida torna-se muito mais fácil a realização da estimativa de prazo de um projeto. Além disso, a falta de estimativas de esforço e do tempo necessário para executar um projeto pode tornar inviável o planejamento e o gerenciamento de um projeto de *software*, pois sem essas estimativas não se pode saber se o projeto está atrasado ou se seu custo foi maior do que o estimado (MONTEIRO, 2005).

2.2.3 ESTIMATIVA DE TEMPO

Com as estimativas de tamanho e esforço calculadas para um projeto de software podem-se, então, determinar as estimativas de prazo. Isto geralmente demanda estimativa de quantas pessoas estarão envolvidas no projeto, que atividades serão executadas, e quando essas atividades iniciarão e serão finalizadas. Tanto dados históricos quanto modelos de dados da indústria podem ser usados para predizer o número de pessoas necessárias para um tamanho de projeto e como o trabalho pode ser dividido dentro do prazo previsto. Para isso, geralmente são utilizadas ferramentas de planejamento de projeto que contenham cronograma (MONTEIRO, 2005).

A estimativa de prazo de maneira simplificada pode ser dada pela razão entre o esforço previsto (geralmente em número de horas trabalhadas) e a quantidade de recursos alocada na execução do projeto, conforme equação (1) (FREIRE, 2008; MONTEIRO, 2005):

$$\text{Prazo} = \frac{\text{Esforço}}{\text{Quantidade de Recursos}} \quad (1)$$

Porém, Gomes (2003) ressalta a importância de ao estimar o tempo de duração do projeto tomar cuidado com a relação tempo/pessoas, pois se um projeto for estimado em dez pessoas-mês e há cinco pessoas disponíveis, não necessariamente ele levará dois meses para ser concluído. É necessário realizar análise da produtividade da equipe, distribuição das tarefas, verificarem a disponibilidade dos integrantes, entre outras atividades geralmente sob responsabilidade do gerente de projeto e em sua maioria controladas por cronogramas. Outro exemplo que gerentes devem tomar cuidado é com a duplicação do número de pessoas em um projeto, que não reduz necessariamente a duração do projeto pela metade (muito pelo contrário, colocar mais pessoas num projeto em andamento apenas retardará ainda mais o processo, uma vez que estas pessoas deverão receber treinamento adequado e "aprender" todo o projeto desde seu início até a fase atual, o que consome muito tempo).

Segundo Bassi (2009), estimar duração corresponde precisamente em dizer quanto tempo a equipe levará para concluir o trabalho. Deve-se tomar cuidado, pois para a mesma tarefa, o tempo para a sua realização pode variar, por exemplo, conforme o tamanho da equipe e a disponibilidade de seus membros, pois esses fatores influem na velocidade de desenvolvimento.

2.2.4 ESTIMATIVA DE CUSTO

O objetivo das estimativas de custo é calcular de maneira antecipada todo e qualquer custo que esteja associado ao sistema, tais como recursos necessários humanos, tecnológicos, burocráticos e de infra-estrutura, ao longo de todo processo de desenvolvimento do produto: fases de construção, instalação, operação, melhoria e manutenção (LEITE, 2006; GOMES, 2003; SIMOES, 2004).

Segundo Freire (2008), em projetos de *software*, o custo é geralmente proporcional ao esforço despendido para sua construção, onde o trabalho humano é o principal recurso a ser consumido. Conseqüentemente, o custo é com freqüência associado a homens-mês ou homens-hora.

Porém, existem muitos fatores a ser considerados quando se está estimando o custo total de um projeto de *software*, como por exemplo: horas trabalhadas (salários e encargos); aquisição ou aluguel de *hardware* e *software*; viagens com reuniões e testes no cliente; contas telefônicas (ligações de longas distâncias, vídeo-conferência, linhas dedicadas para testes, etc.), treinamentos: cursos, livros e manuais, infraestrutura: salas de trabalho, energia aquecimento/refrigeração, redes e Internet, entre outros custos (FREIRE, 2008; LEITE, 2006).

Segundo Monteiro (2005), uma forma de obter o custo total das horas trabalhadas é pelo produto da estimativa de esforço do projeto (em horas) e valor de uma hora trabalhada (R$ por hora). Um custo total mais preciso pode ser obtido por meio do valor das horas trabalhadas específicas de cada recurso utilizado no projeto (recursos técnicos, recursos de suporte, gerente de projeto, *etc*.), ou por meio da

determinação do percentual de esforço total do projeto a ser realizado em cada etapa do processo pelos recursos do projeto.

2.3 TÉCNICAS DE ESTIMATIVAS DE TAMANHO

Tendo em vista que o foco deste trabalho está voltado à análise de estimativas de tamanho de *software*, as seções seguintes apresentarão alguns dos principais métodos de medição funcionais desenvolvidos e muito utilizados ao longo da história de desenvolvimento de *software*, como: Linhas de Código (LOC), Mark II, COSMIC-FFP, *Use Case Points (UCP)* e Análise de Pontos de Função.

2.3.1 Linhas de Código

A indústria de *software* surgiu por volta da década de 40 e seus 10 primeiros anos, entre 1947 e 1957, representaram a fabricação de *software* muito pequena, raro eram os programas que ultrapassavam 1000 declarações em seu código fonte. Todos estes eram escritos em linguagens de baixo nível, seja *assembly*[1] ou linguagem de máquina (SANTANA; GUSMÃO, 2010).

As primeiras tentativas de se medir tamanho de *software*, em cerca de 1950, utilizou a técnica de mensuração por Linhas de Código, muito utilizada até meados da década de 70. Ela consiste em definir que o tamanho de um sistema deve ser medido pela contagem do número de linhas do código fonte do seu programa. Essa métrica considera o *software* sob a perspectiva de sua estrutura interna e é aplicada nas fases finais do projeto, uma vez que a quantidade exata de linhas de código só é conhecida após a conclusão do sistema (MACORATTI, 2005; MONTEIRO, 2005).

As linhas de código apresentam um resultado imediato da atividade de desenvolvimento do produto final, porém, não considera diretamente, diversas outras atividades envolvidas, como: requisitos, arquitetura, testes e outras. Alguns autores

[1] *Assembly* é uma linguagem de programação, provavelmente a primeira da história, surgida na década de 50, quando os computadores ainda usavam válvulas. A idéia do *Assembly* é usar um comando em substituição a cada instrução de máquina (MORIMOTO, 2005)

consideram que estas atividades estão implicitamente consideradas, ou seja, para se produzir certo número de LOC, livre de defeitos, utilizando certo processo de *software*, é necessário especificar os requisitos, projetar e modelar a arquitetura e testar o código, por exemplo (LEITE, 2009b).

Apesar da métrica de LOC ser muito simples e ser muito fácil automatizar sua implementação, percebe-se que é uma métrica muito frágil e imprecisa, por apresentar algumas desvantagens, como (MONTEIRO, 2005; MACORATTI, 2005; HAZAN, 2009):

- Definição de linha de código ser subjetiva, sem padrões para contagem, não estando claro se deve incluir ou não na contagem linhas em branco, linhas de comentário, declaração de dados e linhas de instruções, *etc*;
- É uma medida técnica, sem significado para o usuário;
- Não é consistente, pois algumas linhas de código são mais trabalhosas que outras;
- Dependência do grau de reuso;
- O tamanho de um *software* em linhas de código varia de uma linguagem de programação para outra;
- Apresenta problemas de definição para linguagens não procedurais;
- Penaliza programas pequenos e bem projetados;
- Ser bastante complicado e subjetivo aplicar esta métrica em um processo de estimativas cujo insumo é um documento inicial de requisitos.

2.3.2 Padrão ISO de Medição Funcional

Medição funcional é o termo referente a métodos que dimensionam o tamanho do *software* a partir das funções requisitadas pelo usuário. No começo da década de 90 diante do grande número de métodos de Medição Funcional de tamanho (*Functional Size Measurement* - FSM), surgidos a partir da Análise de Pontos de Função proposto por *Albrecht* e com o objetivo de acabar com a inconsistência existente entre esses métodos e criar um método mais rigoroso de medição funcional, grupos de trabalho de métricas de *software* da Austrália, Reino

Unido, Holanda e Estados Unidos formaram um grupo de trabalho denominado WG12 (*Working Group* 12), subordinado ao SC7 – *Sub-Committee Seven* do JTC1 – *Joint Technical Committee One* estabelecido pela ISO – *International Organization for Standardization* em conjunto com o IEC – *International Engineering Consortium* (VAZQUEZ *et al.*, 2010; ASMA, 1999).

Como resultado dos trabalhos do WG12, foi estabelecido um conjunto de padrões internacional chamado de norma 14143 (ISO/IEC 14143). A norma ISO/IEC 14143 foi estabelecida para garantir que todos os métodos de medição funcional sejam baseados em conceitos semelhantes, e possam ser testados para assegurar o comportamento similar e da forma esperada pelo método, dependendo dos domínios funcionais a que se aplicam (VAZQUEZ *et al.*, 2010).

No fim de 2002 a técnica de Análise de Pontos de Função mantida pelo IFPUG, foi aprovado como um método de medição funcional dentro dos padrões exigidos pela norma ISO/IEC 14143 sob a denominação ISO/IEC 20926:2002. Porém, essa técnica é contemplada apenas até a determinação dos pontos de função não ajustados. As características gerais do manual de contagem do IFPUG, que são utilizadas para estabelecer o fator de ajuste, possuem requisitos tecnológicos e de qualidade o que tornam essa parte do processo excluída do padrão de medição funcional da ISO.

Segundo VAZQUEZ *et al.* (2010), atualmente são cinco os métodos de medição funcional que são aderentes ao ISO/IEC 14143:

- IFPUG CPM 4.3 (ISO/IEC 20926);
- NESMA CPM 2.1 (ISO/IEC 24570);
- Mark II (ou MK II) CPM 1.3.1 (ISO/IEC 20968);
- COSMIC-FFP *Measurement Manual* 2.2 (ISO/IEC 19761);
- FISMA 1.1 (ISO/IEC 29881).

2.3.3 Análise de Pontos de Função

A Análise de Pontos de Função (APF) é uma técnica para medir o desenvolvimento do *software* do ponto de vista do usuário, pela quantificação das

funcionalidades a ele oferecidas. A unidade de medida desta técnica é Ponto de Função (VAZQUEZ *et al.*, 2010).

A técnica da Análise de Pontos de Função surgiu na International Business Machines (IBM), no início da década de 1970, como alternativa às métricas baseadas em linhas de código. Na época, *Allan Albrecht* foi encarregado de medir a produtividade de vários projetos de *software* desenvolvidos por uma unidade da IBM. Esses projetos tinham sido desenvolvidos em várias linguagens de programação distintas, tornando inviável uma análise conjunta de produtividade usando a métrica de linhas de código. Isso motivou a busca por uma medida que fosse independente da linguagem de programação utilizada, criando então a técnica de Análise de Pontos de Função. Os conceitos desta técnica foram introduzidos por *Allan J. Albrecth*, em uma conferência do *GUIDE* – Grupo de Usuários IBM, em 1979 (VAZQUEZ *et al.*, 2010; MACORATI, 2005).

A técnica foi refinada por Albert em 1984, a partir de então houve um aumento considerável na sua utilização, o que levou a necessidade de definir um padrão para aplicação da técnica. Com este objetivo foi criado em 1986 o *International Function Point Users Group* (IFPUG) que passou a ser responsável, entre outros, pela elaboração, manutenção e divulgação de um manual de práticas de contagem (COM – *Counting Practices Manual*), além de realizar treinamentos e manter um programa de certificação para profissionais especializados em aplicar a técnica de APF (CFPS – *Certified Function Point Specialist*). Essa organização foi originada em Toronto e hoje se encontra nos Estados Unidos contendo pouco mais de 3000 afiliados de vinte países diferentes (VAZQUEZ *et al.*, 2010; PARO, 2005; SANTANA; GUSMÃO, 2010).

Em 1990 foi lançada a primeira versão do Manual de Práticas de Contagem com o objetivo de padronizar a técnica. Em Fevereiro de 2012, o manual está na versão 4.3 (VAZQUEZ *et al.*, 2010; IFPUG, 2011).

Segundo Macoratti (2005) a APF é a técnica mais usada para estimativa de tamanho de *software*. Em 1998, foi constituído o BFPUG – *Brazilian Function Point Users Group* – o representante do IFPUG no Brasil.

A APF tem como objetivo medir o tamanho do projeto de *software* a partir do ponto de vista do usuário do *software*, levando em conta basicamente as

características do sistema do ponto de vista da sua fronteira com o usuário independente da metodologia de desenvolvimento, da plataforma e da tecnologia utilizadas no desenvolvimento da aplicação. Ou seja, a APF busca medir o que o *software* faz, e não como ele foi construído (MACORATI, 2005; HAZAN, 2009; VAZQUEZ *et al.*, 2010).

É importante destacar que pontos de função não medem diretamente esforço, produtividade ou custo, pois é exclusivamente uma medida de tamanho funcional do *software*. Este tamanho em conjunto com outras variáveis é que pode ser usado para derivar produtividade, estimar esforço e custo (VAZQUEZ *et al.*, 2010). O capítulo 3 apresentará maiores informações a respeito desta técnica.

2.3.4 Mark II

A técnica Mark II, ou simplesmente MK II, foi criada por *Charles Symons*, inspirado na proposta de *Albrecht*, como um método proprietário da empresa KPMG nos anos de 1985/86. Visa melhorar a escala de tamanho funcional, contando mais apuradamente a complexidade de processamento interno de Sistemas de Informações Gerenciais (VAZQUEZ *et al.*, 2010; PARO, 2005).

Uma das diferenças principais entre APF e MK II é que a primeira técnica conta “Arquivos Lógicos” uma vez para cada parte de *software* sendo mensurada, enquanto que a MK II conta “tipos de entidade” toda vez que elas são referenciadas em cada transação lógica. Segundo o manual do MK II, as duas métricas pontuam similarmente os projetos com até 400 pontos de função. Ultrapassando esse valor, a tendência é que a métrica MK II pontue valores maiores que a APF (PARO, 2005).

Segundo Vazquez *et al.* (2010) a disseminação da técnica foi dificultada inicialmente, pelo fato de ser um método proprietário de uma empresa de consultoria (a KPMG). Mas atualmente o método é de domínio público e a associação responsável por manter o padrão da técnica é a associação de métricas do Reino Unido – UKSMA (*United Kingdom Software Metrics Association*), que também possui

ações e objetivos similares ao do IFPUG. No entanto, é um método de aplicação restrita no Reino Unido.

2.3.5 COSMIC FFP

O método COSMIC *Full Function Points* (COSMIC – FFP) é uma das abordagens de medição funcional de tamanho de *software* mais atuais. Foi proposta em 1997 com o nome de *Full Function Points* v.1, como uma adaptação da métrica FPA/IFPUG, para sistemas *real time*. Em 1999 o grupo *Common Software Measurement International Consortium* – COSMIC propôs a abordagem COSMIC – FFP – V.2.1 (Cosmic Ponto de função cheio – V.2.1) como uma métrica totalmente independente de outras métricas, projetada para trabalhar tanto com aplicações de negócio como com *softwares* em tempo real (CALAZANS, 2003; MONTEIRO, 2005).

Essa técnica foi criada para medir o tamanho de um *software* baseado em suas funcionalidades. Foi proposta para tratar tanto requisitos de *softwares* embarcados como de tempo real e surgiu como uma alternativa de mensuração mais exata (de forma a não gerar dúvidas, não sendo ambígua), com independência de domínio e propondo diferentes medidas para diferentes propósitos (considerando a visão do usuário e do desenvolvedor) (CALAZANS, 2003; MONTEIRO, 2005).

A métrica COSMIC possui várias características que a diferenciam das métricas já existentes, tais como (CALAZANS, 2003):

- A possibilidade de aplicar a visão do usuário final e do desenvolvedor (todas as métricas desconsideram a visão do desenvolvedor e em muitos processos a visão do desenvolvedor é responsável por uma mensuração mais acurada do produto);
- A identificação de camadas (considerando que a maioria das tecnologias existentes trabalha com este paradigma);
- A flexibilidade (utilizando a abordagem FPA uma EE – Entrada Externa - pode ter no máximo de 3 a 6 pontos, no COSMIC depende de quantos movimento efetua);

- E a aplicabilidade de forma mais simples e menos ambígua.

Segundo Cappelli (2009), a vantagem de utilização da técnica de COSMIC - FFP é o fato de ser possível realizar estimativas de tamanho, em fases iniciais de projeto, porém apresenta como grande ponto negativo o fato de possuir pouco relato de experiência prática usando essa técnica.

2.3.6 UCP

A técnica de estimativa por Pontos de Caso de Uso foi proposta em 1993 por *Gustav Karner*, da *Objectory* (atual, *Rational Software*) para medir o tamanho de projetos de *software* orientados a objeto, com base nas especificações de Casos de uso. Uma técnica baseada em dois métodos muito utilizados: o mecanismo de Análise de Pontos de Função e na metodologia conhecida como *Mk II*, uma adaptação da técnica de PFs, bastante utilizada na Inglaterra. A forma de lançar uma estimativa é o principal diferencial da métrica por Casos de Uso: o método trata de estimar o tamanho de um sistema de acordo com o modo como os usuários o utilizarão, a complexidade de ações requerida por cada tipo de usuário e uma análise em alto nível dos passos necessários para a realização de cada tarefa, em um nível muito mais abstrato que a técnica de Pontos de Função (FREIRE, 2003b; MACORATI, 2005; MONTEIRO, 2005; SOUSA, 2009).

A técnica de UCP explora o modelo e a descrição de casos de uso, substitui algumas características técnicas propostas pela APF, cria os fatores ambientais e propõe uma estimativa de produtividade. O processo de contagem da técnica é composto basicamente por seis etapas (PARO, 2005; MONTEIRO, 2005):

- Classificação dos atores;
- Classificação dos casos de uso;
- Cálculo dos pontos de casos de uso não justados;
- Cálculo dos fatores técnicos;
- Cálculo dos fatores ambientais; e
- Cálculo dos pontos de caso de uso.

A UCP tem contribuído para diminuir algumas dificuldades encontradas pelo mercado em relação à resistência de adoção de métodos de estimativas, porque é uma técnica simples, fácil de usar e rápida de aplicar, quando se têm as informações necessárias para realizar as estimativas. No entanto, a UCP conta com algumas desvantagens: somente pode ser aplicado em projetos de *software* cuja especificação tenha sido expressa em casos de uso. Sendo assim, sua eficácia depende de uma padronização de todos os casos de uso de uma instituição, pois um dos passos para a obtenção das métricas é a contagem de transações por caso de uso (MONTEIRO, 2005; HAZAN, 2009).

Além disso, como não existe um padrão único para a escrita de uma especificação de caso de uso, diferentes estilos na escrita do caso de uso ou na sua granularidade podem levar a resultados diferentes na medição por PCU. Assim, a métrica se torna subjetiva. E ainda, devido ao processo de medição do PCU ser baseado em casos de uso, o método não pode ser empregado antes de concluída a análise de requisitos do projeto. Na maioria das vezes existe a necessidade de se obter uma estimativa antes da finalização desta etapa (HAZAN, 2009; PARO, 2005). Segundo Paro (2005), a técnica de UCP está sendo pouco utilizado se comparado com a APF.

Segundo Aguiar (2003) se comparar a utilização de Pontos por Caso de Uso com Pontos de Função para realização de estimativas de tamanho de software, ele afirma que é mais conveniente utilizar Pontos de Função, devido:

- Os Pontos de Função são mantidos por uma organização internacional sem fins lucrativos, o IFPUG, desde 1986;
- Os Pontos de Função possuem suporte no Brasil, o BFPUG, além de empresas especializadas;
- O IFPUG mantém um programa mundial de certificação profissional em Pontos de Função, o qual confere aos aprovados o título de *Certified Function Point Specialist* (CFPS), programa esse realizado no Brasil pelo BFPUG;
- Os Pontos de Função são padronizados internacionalmente pela ISO, através da norma ISO/IEC 20926, possibilitando a uniformidade na aplicação;

- Os Pontos de Função modelam os requisitos a um nível de abstração mais elevado e independente dos artefatos do que os UCP, podendo ser utilizados por organizações que utilizem qualquer forma de representação dos requisitos, casos de uso ou outras;
- A existência de grande acervo de dados sobre Pontos de Função armazenados por diversas organizações possibilita a realização de estudos e comparações;
- A Utilização dos Pontos de Função em contratos e licitações é uma realidade no Brasil, tendo surgido a partir da iniciativa de organizações governamentais e rapidamente alcançado o mercado em geral.

2.3.7 Planning Poker

Projetos cujo desenvolvimento utilizem metodologias ágeis também necessitam de estimativas, sejam elas de tamanho, custo, esforço ou duração. Nestes casos também é necessário mensurar o tamanho de cada tarefa com uma boa precisão, e, conseqüentemente, conseguir prever quando determinado recurso deverá ser concluído. Segundo Cunha (2009), a metodologia de trabalho mais utilizada por equipes que empregam metodologias ágeis (ex: *Scrum* ou *Extreme Programming* - *XP*) no desenvolvimento de *software* é: o *Planning Poker*.

O *Planning poker* é uma técnica de estimativa criada por *James Grenning* em 2002, cuja contribuição decisiva para difundir essa técnica entre os praticantes de métodos ágeis foi de *Mike Cohn*. Esta técnica usa o conhecimento dos desenvolvedores para chegar às estimativas. As suas regras estimulam a interação entre os membros da equipe e permitem que todos expressem as suas opiniões, ao mesmo tempo em que participam do planejamento do projeto e aumentam o seu entendimento sobre o sistema que irão desenvolver. Para chegar a estimativas realistas e razoáveis, as próprias pessoas que participarão da implementação usam a estratégia de divisão e conquista junto com o dimensionamento por analogia (BASSI, 2009).

De forma simplificada, o funcionamento do *Planning Poker* é: uma técnica de estimativa baseada no consenso de toda a equipe, onde é utilizado um conjunto de cartas com valores específicos (na maioria das vezes esses valores seguem a Seqüência de *Fibonacci*[2]) que representam a escala (ou unidade de medida adotada) e é praticado como se fosse um jogo. Um participante apresenta a tarefa ou estória para o time, e, após uma breve discussão, cada um escolhe uma carta e coloca virada para baixo sobre uma mesa. Quando todas as cartas estiverem lançadas, elas são viradas e caso não haja consenso nos valores escolhidos, as diferenças são discutidas de forma breve, e uma nova rodada acontece até que haja a convergência (CUNHA, 2009).

Se no desenvolvimento de *software* utilizando metodologias tradicionais é necessário uma unidade de medida padrão para realizar a estimativa de *software*, utilizando metodologias ágeis isso não é diferente. Segundo BASSI (2009) para começar a usar *Planning poker*, é preciso definir alguns conceitos e fazer alguns preparativos sendo que o mais importante para a estimativa de tamanho é a definição de uma escala. Esta escala irá medir a unidade de medida padrão adotada. Boas escalas podem ser criadas usando números de *Fibonacci* (1, 2, 3, 5, 8, 13,...), potências de 2 (1, 2, 4, 8, 16, 32,...) ou (1, 5, 10, 20, 40, 80,...), pois são seqüências exponenciais que irão absorver parte do erro associado às estimativas de funcionalidades grandes.

Existem duas medidas de tamanho que podem ser utilizadas no desenvolvimento de *software*: pontos de estória (*story points*) e dia ideal (*ideal time*). Os pontos são medidas abstratas com tamanho definido pela equipe. Um dia ideal é o quanto seria produzido em um dia de trabalho se 100% do tempo fosse dedicado a uma atividade, sem interrupções ou distrações. Para este caso, a equipe precisa determinar a porcentagem de seu tempo que ela dedica ao projeto. Este percentual servirá para fazer a conversão entre dias ideais e dias de trabalho (BASSI, 2009; CUNHA, 2009b).

Bassi (2009) também afirma que os pontos e os dias ideais fornecem estimativas de tamanho, que são medidas sobre o volume de trabalho, mas que

[2] Sequência de *Fibonacci* é uma seqüência numérica criada pelo matemático Leonardo Pisa, que tem a seguinte lei de formação: cada elemento, a partir do terceiro, é obtido somando-se os dois anteriores (RIBEIRO, 2008).

além dessas, é interessante obter estimativas de duração, como por exemplo, horas de trabalho ou dias de trabalho, que prevêem a quantidade de tempo necessária para cumprir a estimativa de tamanho. Contudo, é importante que medidas de tamanho e duração fiquem separadas, pois a velocidade da equipe pode variar durante o projeto. Assim, quando a velocidade de desenvolvimento mudar, basta identificar a nova velocidade e derivar as novas estimativas de duração.

3 ANÁLISE DE PONTOS DE FUNÇÃO

Segundo *Albrecht* (1983) *apud* Macoratti (2005) a técnica de Análise de Pontos de Função foi criada com o intuito de medir a complexidade do produto pela quantificação de funcionalidade expressa pela visão que o usuário tem do *software*. O modelo mede o que é o sistema, o seu tamanho funcional e não como este será, além, de medir a relação do sistema com usuários e outros sistemas. A técnica pode ser aplicada para qualquer *software*, independente da tecnologia usada e mede uma aplicação pelas funções desempenhadas para/e por solicitação do usuário final.

A APF é uma das métricas de estimativa de tamanho mais sedimentadas no mercado e que proporciona resultados cada vez mais precisos à medida que artefatos da fase de análise e projeto são gerados (MACORATTI, 2005). Pois, esta técnica permite uma contagem indicativa no início do projeto, quando não se conhece os detalhes do modelo de dados; podendo ser definida na fase de projeto quando se têm um maior conhecimento das funções do *software*, gerando uma estimativa; até o término do desenvolvimento quando se efetua uma contagem detalhada com base no conhecimento das funções levantadas durante todo o processo de desenvolvimento do *software* (IFPUG, 1999 *apud* MACORATTI, 2005).

Segundo Santana e Gusmão (2010), devido à grande aceitação e padronização do método de estimativa de APF, o mesmo acabou tornando-se uma norma ISO (ISO 20926) em 2002. Além disso, segundo publicações da Fatto (2011) a Análise de Pontos de Função é atualmente um instrumento utilizado por profissionais da área de sistemas e em empresas de todos os portes e segmentos da economia brasileira. No início o foco principal de sua aplicação eram as estimativas. Atualmente além de ser uma ferramenta importante em estimativas, tem sido aplicada cada vez com mais sucesso como unidade de medição de contratos de desenvolvimento de *software* e como ferramenta na gerência de projetos. Boa parte deste sucesso é caracterizado como resultado da comunidade bem ativa de usuários de pontos de função no Brasil – por meio do BFPUG.

3.1 OBJETIVOS DA APF

A Análise de Pontos de Função é um método-padrão para a medição do desenvolvimento de *software*, visando estabelecer uma medida de tamanho do *software* em Pontos de Função, com base na funcionalidade a ser implementada, sob o ponto de vista do usuário. Seus principais objetivos são (HAZAN, 2009; SANTANA; GUSMAO, 2010, MACORATI, 2005):

- Medir funcionalidades do sistema requisitadas e recebidas pelo usuário;
- Medir o tamanho do desenvolvimento de *software* e da manutenção de forma independente de tecnologia usada para o desenvolvimento;
- Prover uma unidade de medição normalizada entre projetos e organizações.
- Prover uma métrica de medição para apoiar a análise de produtividade e qualidade;
- Prover um fator de normalização para comparação de *software*.

Embora o objetivo principal da Análise de Pontos de Função é ser usada para medir o tamanho de um projeto de *software*, a APF pode ser usada para (NESMA, 2009; MONTEIRO, 2005):

- Descrever o escopo de um sistema e medir o seu tamanho funcional, independentemente da tecnologia que será usada no sistema;
- Derivar a produtividade e métricas (indicadores) de desempenho do processo, estimativa das necessidades de recursos e auxiliar no gerenciamento de projetos;
- Avaliar os fatores em um ambiente de desenvolvimento que influenciam a produtividade e oferecer uma base para melhorar os processos de desenvolvimento e manutenção;
- Determinar o escopo e tamanho da melhoria em um sistema e auxiliar na gestão de suas mudanças;
- Fornecer suporte para planejamento e controle de projetos de *software*;
- Apoiar a análise de produtividade e qualidade do processo de desenvolvimento de sistemas;

- Servir-se como ferramenta de comunicação com os usuários;
- Realizar estudos de *benchmark*[3];
- Facilitar a negociação nos contratos de *software*; e
- Medir artefatos de um sistema.

Segundo Hazan (2001b), na prática a técnica Análise de Pontos por Função tem demonstrado grande utilidade na garantia de que a especificação de requisitos esteja bem elaborada e completa. Uma contagem de Pontos de Função é realizada usando uma descrição suficientemente formal das necessidades do negócio na linguagem do usuário. Assim, a contagem pode servir também como uma revisão dos requisitos do sistema com o usuário. Além disso, quando usada em combinação com outras medidas, a APF pode ser usada para determinar (MACORATTI, 2005):

- O nível de produtividade da equipe;
- Esforço de desenvolvimento de *software*;
- O custo de *software*;
- Taxa de produção de *software*;
- Taxa de manutenção de *software*.

Além destes objetivos, Macoratti (2005) afirma que o processo de contagem de Pontos de Função deve ser:

- Simples para minimizar o trabalho adicional do processo de medição;
- Conciso para permitir consistência, ao longo do tempo, dos projetos, e entre os usuários da técnica.

3.2 ALGUMAS RAZÕES PARA UTILIZAR APF

A APF surgiu para quebrar o paradigma de mensuração do tamanho do produto pela quantidade de linhas de código, bem como a dependência de tecnologia a ser empregada e difundiu-se no mercado proporcionando um processo

[3] *Benchmarking* é um processo ou técnica de gestão através do qual as empresas ou organizações avaliam o desempenho dos seus processos, sistemas e procedimentos de gestão comparando-o com os melhores desempenhos encontrados noutras organizações (NUNES, 2008).

maduro para avaliar o tamanho lógico do *software* com base em requisitos funcionais dos usuários. O foco da metodologia da APF compreende na mensuração da aplicação pelo que ela faz, pela perspectiva do usuário e não como a aplicação foi construída. A contagem dos pontos por função é baseada na avaliação dos requisitos do usuário, sendo que ela representa exclusivamente o tamanho funcional da aplicação. O ponto por função (unidade de medida da APF) por sua vez serve de base para que com outras variáveis possa ser calculado o esforço, prazo e o custo (FABIAN, 2008; MACORATTI, 2005).

Na parte de projetos a APF pode servir de suporte na análise de produtividade e qualidade, em conjunto com métricas de esforço, defeitos e custo. A utilização da APF em projetos de *software* apresenta benefícios como (SIMOES, 2004; MACORATTI, 2005; PARO, 2005; FABIAN, 2008):

- A estimativa é feita em função da visão do usuário;
- Facilidade de aprendizagem e aplicação da técnica;
- Permite dimensionar o tamanho de um *software* a ser desenvolvimento;
- Permite realizar estimativas de custo, cronograma e recursos para o desenvolvimento e manutenção de *softwares*;
- Independência de tecnologia;
- Pode ser usada como uma ferramenta de auxílio gerencial, pois possibilita a coleta de dados para obtenção de diversos indicadores de acompanhamento;
- Pode ser usada como uma ferramenta para auxiliar a decisão entre a compra de um pacote ou o desenvolvimento do aplicativo da empresa;
- Representa uma forma de normalização para comparação de *software*, ou ainda comparação de produtividade na utilização de tecnologias diferentes;
- Pode medir o tamanho em qualquer fase do ciclo de vida do desenvolvimento de sistema;
- Promove *benchmarking* entre várias organizações;
- Manutenção de dados históricos;
- Complementa o gerenciamento de requisitos, ao analisar a qualidade do levantamento de requisitos;

- Uma forma para fundamentar a negociação de contratos, estabelecendo uma unidade tangível para o cliente, o ponto por função, estabelecendo um preço por esta unidade;
- Possui certificação ISO que é uma garantia consistente de que esta metodologia é eficaz e totalmente madura para ser adotada pelas empresas.

Segundo Aguiar (2003) *apud* Macoratti (2005), dentre as principais razões para a utilização da APF como métrica estão os seguintes motivos:

- O fato dos Pontos de Função serem mantidos por uma organização internacional sem fins lucrativos, o IFPUG , desde 1986;
- Os PF possuírem suporte no Brasil através do *chapter* – BFPUG;
- Os PF serem padronizados pela ISO através da norma ISO/IEC 20296;
- Existir um grande acervo de informações sobre PF armazenados em diversas organizações o que permite estudos e comparações;
- Os PF modelarem os requisitos a um nível de abstração mais alto e independente dos artefatos e poderem ser usados por organizações que usam qualquer forma de representação de requisitos;
- Os PF serem usados em contratos e licitações no Brasil em organizações governamentais e pelo mercado em geral.

Segundo Paro (2005), a APF também apresenta algumas desvantagens, entre elas: a necessidade de um bom nível de experiência no assunto para efetuar uma contagem acurada; a incapacidade de automação do processo de contagem (contagem manual); a necessidade significativa de um nível de detalhe de informações do *software* para uma medição mais confiável (entradas, saídas, consultas, registros, bases, *etc*). Além disso, Dekkers (1998) aponta um dos desafios na implementação da análise de Pontos de Função: tornar o método compreensível para os desenvolvedores. Devido ao fato de serem baseados em requisitos funcionais dos usuários (o que o *software* faz), independentemente da implementação física (como o *software* faz o que faz), pontos de função forçam os desenvolvedores a pensarem em termos lógicos.

3.3 DEFINIÇÃO DE TERMOS

Antes de iniciar o estudo sobre a técnica de APF e qual seu processo de contagem de Pontos de Função, é importante reavaliar alguns conceitos muito utilizados quando se trata de Tecnologia de Informação, mas que no processo de contagem da APF causam algumas confusões, porque sua utilização genérica na TI frequentemente possui um significado diferente daquele utilizado na contagem de PF (DEKKERS, 1998). Os termos a seguir são usados tanto na metodologia de pontos de função, quanto na tecnologia da informação, porém com características particulares:

- Usuário
- Lógico
- Aplicativo (sistema)
- Projeto
- Arquivo

3.3.1 Usuário

O termo *usuário* tem um significado mais amplo para a análise de pontos de função do que o usual. Em tecnologia da informação, refere-se a uma pessoa “física” que utiliza o *software*. Na terminologia de Pontos de Função, usuário é qualquer pessoa ou coisa que interaja (envia ou receba dados) com a aplicação. Exemplos de usuário: pessoa física, outra aplicação, um *hardware*, um ator em um caso de uso, agentes governamentais (exigências regulatórias do governo normalmente abrangem boa parte dos requisitos de um *software*), gestores do negócio que o *software* vai atender (na medida em que quem especifica os requisitos do *software*, com ele também interage) (VAZQUEZ *et al.*, 2010).

Essa diferença no significado pode fazer com que algumas funções contáveis sejam esquecidas pelos desenvolvedores, por não parecem requisitos definidos por ou para um usuário “físico". Além disso, se a contagem de pontos de função fosse

baseada apenas no conceito de usuário como sendo uma pessoa, não seria possível medir sistemas que não têm interface com o usuário final. Porém, a APF pode ser aplicada também nestas situações (DEKKERS, 1998; VAZQUEZ *et al.*, 2010).

3.3.2 Lógico

O termo *lógico* em tecnologia da informação geralmente refere-se a *layouts* de bases de dados ou modelos lógicos de dados. Porém quando o termo lógico é utilizado na contagem de pontos de função, refere-se aos requisitos conceituais ou funcionais dos usuários, excluindo a implementação física ou os requisitos de projeto (DEKKERS, 1998).

Segundo Dekkers (1998) os requisitos lógicos dos usuários são aqueles que um usuário experiente no assunto identificaria como requisitos do *software*. Os requisitos lógicos ou funcionais dos usuários descrevem o que o *software* deve fazer, sem dizer como o *software* executará as funções. Os pontos de função medem o tamanho desses requisitos funcionais dos usuários, apenas. Considerações sobre o projeto e a qualidade, embora importantes para a construção do *software*, não são parte do tamanho lógico do *software*, não sendo, portanto, contados em pontos de função. Todas as contagens de pontos de função são realizadas a partir da perspectiva lógica do usuário, e os contadores iniciantes precisam pensar logicamente ao contar pontos de função.

3.3.3 Aplicativo

Os termos *aplicativo* e *sistema* geralmente são utilizados de forma intercambiável em processamento de dados e estão diretamente ligados à divisão física do *software*. Alguns exemplos de como os aplicativos ou sistemas podem ser divididos pelos desenvolvedores, são (DEKKERS, 1998):

- Com base nas funções executadas em modo *batch* ou *on-line*. Às vezes, cada modalidade de implementação física de um único conjunto de funções cooperativas pode ser considerada um sistema separado pelos desenvolvedores: por exemplo, o sistema *batch* de contas a receber e o sistema *on-line* de contas a receber;
- Com base na plataforma física na qual um subconjunto das funções (ou subfunções) reside: por exemplo, o sistema de folha de pagamento *mainframe* e o sistema de folha de pagamento PC;
- Com base nos pacotes físicos que compreendem um conjunto de funções: por exemplo, o aplicativo de banco de dados *Access* ou o aplicativo de entrada de dados.

No contexto da contagem de pontos de função, o termo aplicativo é definido pelo IFPUG como uma coleção coesa de dados e procedimentos automatizados que suportam um objetivo de negócio, podendo consistir de um ou mais componentes, módulos ou subsistemas. Frequentemente, o termo *aplicação* é utilizado como sinônimo para sistema, sistema aplicativo ou sistema de informação. A aplicação deve ser definida segundo a visão do usuário e de acordo com considerações de negócio e não com base em questões técnicas como arquitetura do *software* ou plataforma tecnológica (VAZQUEZ *et al.*, 2010).

Estas diferenças na definição e utilização da palavra aplicativo podem resultar em excesso de contagem, como por exemplo: baseado no primeiro exemplo citado acima, o aplicativo ter seus PF contados como dois, uma contagem *on-line* e outra *batch*, porém se levar em consideração o significado de aplicativo na contagem de pontos de função os “sistemas” *batch* e *on-line* seriam um único “aplicativo” (DEKKERS, 1998).

Segundo DEKKERS (1998), é importante lembrar que na contagem de pontos de função, um aplicativo representa a forma segundo a qual um usuário vê *logicamente o sistema,* enquanto na tecnologia da informação um aplicativo geralmente representa como *o sistema é fisicamente implementado.*

3.3.4 Projeto

Segundo Dekkers (1998) o termo *projeto* quando utilizado no desenvolvimento de sistemas, pode descrever:

- Escopo de trabalho que inclui a melhoria ou o desenvolvimento de vários aplicativos de *software*;
- O escopo de trabalho incluindo consertos/manutenção de funções existentes, além da melhoria de outras funções em um único aplicativo de *software*;
- Acertos no *software* em operação ou upgrades no *software* existente;
- Qualquer combinação dos itens acima.

Porém, na contagem de pontos de função, projeto refere-se ao produto do trabalho relacionado ao desenvolvimento ou melhoria de um único aplicativo (sistema). *O* que é ‘projeto’ em termos de TI pode conter diversos ‘projetos de pontos de função’ e, dessa forma, necessitar de diversas contagens de PF. O Manual de Práticas de Contagem do IFPUG define projeto de desenvolvimento e de melhoria como (DEKKERS, 1998):

- Desenvolvimento: A especificação, construção, teste e entrega de um novo sistema de informações.
- Melhoria: A modificação de um aplicativo existente.

3.3.5 Arquivo

O termo *arquivo*, em TI geralmente lembra o processamento de *mainframe*, orientado a transações, sendo usado como sinônimo de *dataset*. Termos relacionados, como 'arquivos de pesquisa’, 'arquivos de saída’, 'arquivos *batch'*, 'arquivos *excel'* e 'arquivos de transação' ainda são muito usados hoje em dia (DEKKERS, 1998).

Na contagem de pontos de função, arquivo é um agrupamento lógico de dados requerido pelos usuários. O CPM define arquivo como (DEKKERS, 1998):

- Arquivo: Para os tipos de funções de dados, um grupo de dados logicamente relacionados, não a implementação física desse grupo de dados.

A seguir, alguns exemplos de arquivos físicos/*datasets* em TI que não seriam Arquivos (entidades) em pontos de função (DEKKERS, 1998):

- Um *dataset* de entrada poderia conter transações que causariam atualizações em cadastros no aplicativo. (Para PF seria contado como uma ou mais Entradas Externas, uma vez que é esse o requisito lógico do usuário. O local físico de armazenamento desses processos por acaso é um *dataset*).
- Um arquivo de saída poderia conter a versão eletrônica de diversos relatórios ou grupos de dados (por exemplo, vários formulários diferentes de imposto de renda poderiam estar todos gravados em uma única fita física de saída). Em PF isto seria contado com várias Saídas Externas (uma para cada um dos requisitos funcionais distintos do usuário). Para os usuários, não importa se há uma ou várias fitas físicas - desde que eles recebam a funcionalidade.

A confusão pode surgir porque um Arquivo Lógico Interno (ALI) e um Arquivo Lógico Externo (AIE) em termos de pontos de função referem-se a uma entidade persistente de dados, não um arquivo físico ou *dataset*. Vale ressaltar que arquivo em pontos de função refere-se a um agrupamento lógico de dados relacionados (DEKKERS, 1998).

3.4 PROCESSO DE CONTAGEM

O processo de contagem dos pontos de função pode ser dividido em sete etapas, conforme ilustra a Figura 3 (HAZAN, 2001b):

- Determinar o tipo de contagem de pontos de função;
- Identificar o escopo de contagem e a fronteira da aplicação;
- Contagem das funções de dados;
- Contagem das funções transacionais;

- Determinar a contagem de pontos de função não ajustados;
- Determinar o valor do fator de ajuste;
- Calcular os pontos de função ajustados.

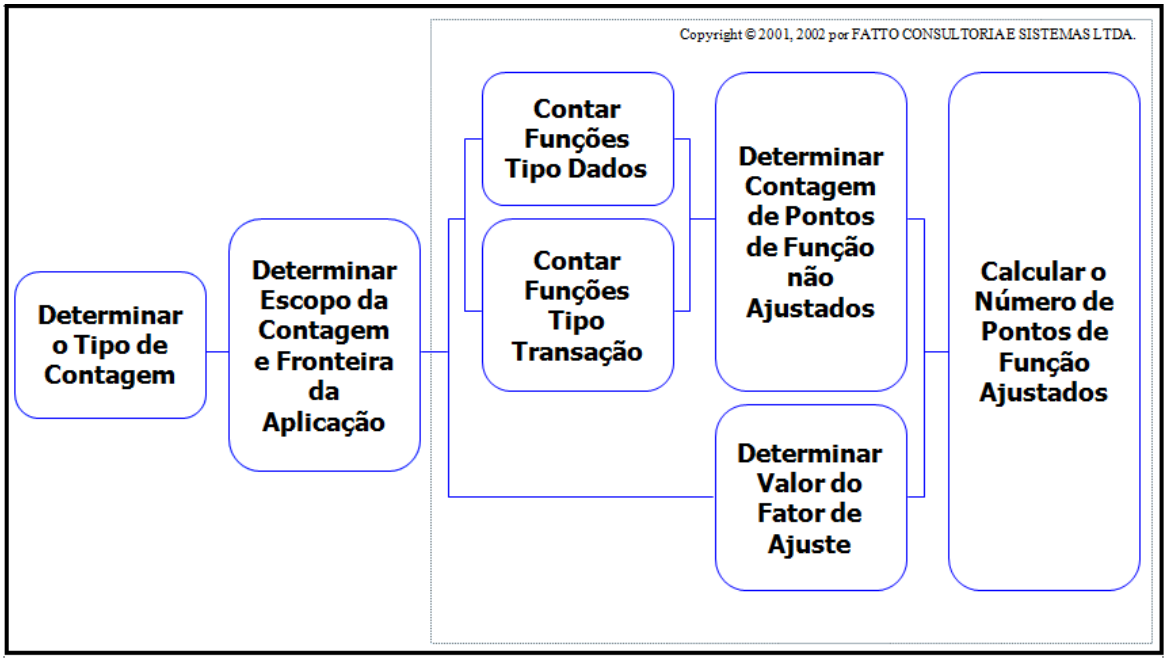

Figura 3 - Processo de Contagem de Pontos de Função
Fonte: FATTO (2011)

3.4.1 TIPO DE CONTAGEM

O primeiro passo a ser seguido para a contagem de PF de um projeto de software é determinar o tipo de contagem (O que de fato deve ser contado?). Neste passo os responsáveis pela medição devem estabelecer o tipo de contagem que será usado para medir o projeto de software. O tipo de contagem não altera a aplicação das regras de contagem, mas pode influenciar no Cálculo do tamanho no final na contagem. Antes de iniciar uma contagem de pontos de função deve-se identificar qual o tipo de contagem que se vai efetuar dentro dos três (3) definidos pelo CPM do IFPUG (CAMPOS, 2010; IFPUG, 1999 *apud* MACORATTI, 2005b).

Os três tipos de contagem podem ser classificados dentro de dois grupos de contagem. Um deles é a contagem da aplicação existente onde se podem contar todas as funcionalidades oferecidas pela aplicação ou contar somente um subgrupo

das existentes (visando atender a um propósito). E, existe o outro grupo que é o que envolve a criação ou manutenção de funcionalidades que estão no escopo de um projeto; seja ele de desenvolvimento ou de melhoria (CAMPOS, 2010b).

Os três tipos de contagem possíveis são:

- **Contagem de pontos de função de projeto de desenvolvimento**: Projeto de desenvolvimento é aquele que tem como objetivo desenvolver e fornecer a primeira versão de uma aplicação, é a criação de uma aplicação. O número de pontos por função de um projeto de desenvolvimento mede a funcionalidade do projeto a ser entregue. Neste caso, a contagem não compreende apenas o software em si, mas também outras ações a serem feitas durante a sua execução também devem ser contadas, como uma migração de base de dados, por exemplo. O tamanho funcional deste tipo de projeto é uma medida de funcionalidade que é oferecida aos usuários com a criação da aplicação medida a partir das funções fornecidas ao usuário na primeira instalação da aplicação (quando o projeto é concluído) (CAMPOS, 2010b; FABIAN, 2008).

É importante destacar que qualquer contagem realizada antes do término de um projeto é na verdade uma estimativa da funcionalidade que será entregue ao seu final. Conforme os requisitos vão ficando mais claros durante o projeto e as funções vão sendo desenvolvidas, é bastante natural encontrar funcionalidades não medidas no planejamento do projeto (VAZQUEZ *et al.*, 2010).

- **Contagem de pontos de função de projeto de melhoria**: O projeto de melhoria, diferente do projeto de desenvolvimento, parte de um *software* já produzido e tem por objetivo efetuar e entregar uma manutenção adaptativa de uma aplicação já existente. O número de pontos de função de um projeto de melhoria mede as funções adicionadas, modificadas ou eliminadas do sistema pelo projeto, e também as eventuais funções de conversão de dados. Neste caso, o tamanho funcional de um projeto de melhoria é uma medida das funcionalidades que são oferecidas aos usuários com a implantação do respectivo projeto (CAMPOS, 2010b; VAZQUEZ *et al.*, 2010).
- **Contagem de pontos de função de aplicação**: A contagem de pontos de função de uma aplicação, também conhecida por pontos de função instalados

ou *baseline*, refere-se a uma aplicação já instalada e mede a funcionalidade fornecida ao usuário pela aplicação instalada. O tamanho da aplicação deve iniciar quando da finalização do projeto de desenvolvimento e, deve ser atualizada toda a vez que um projeto de melhoria alterar as funções da aplicação. Esta contagem fornece uma medida das funções que a aplicação oferece atualmente ao usuário (VAZQUEZ *et al.*, 2010; CAMPOS, 2010b).

3.4.2 FRONTEIRA DA APLICAÇÃO E O ESCOPO DA CONTAGEM

O segundo passo do Processo de Contagem é a identificação da fronteira da aplicação e o escopo da contagem. Para realizar uma contagem de pontos de função é necessário definir determinados contextos, entre eles identificar qual é o escopo para o tipo de contagem definida. ***O escopo da contagem*** não altera a aplicação das regras de contagem, mas influencia no tamanho da contagem e deve ser identificado depois que foi definido o tipo de contagem de pontos de função a ser realizada (CAMPOS, 2010c).

O objetivo da identificação do escopo de contagem consiste em definir quais funções serão incluídas na contagem: se abrange um ou mais sistemas, se abrange apenas uma parte do sistema e se compreenderá todas as funcionalidades; apenas as funcionalidades em uso pelo usuário ou funcionalidades específicas (FABIAN, 2008; VASQUEZ *et al.*, 2010).

Para Vazquez *et al.* (2010) o conceito de escopo é mais utilizado para a contagem de projetos de melhoria. E é neste tipo de contagem onde ocorrem mais erros, pois se realiza a contagem de funções da aplicação sem que elas tenham sido alteradas pela melhoria.

Quanto à atividade de estabelecer a ***fronteira da aplicação***, esta tem como objetivo delimitar onde começa e onde termina a medição dos pontos por função. A fronteira separa a aplicação que esta sendo contada das aplicações externas indicando o limite entre a aplicação e os demais usuários. A fronteira é definida estabelecendo uma interface conceitual (um limite lógico) entre a aplicação que esta sendo contada o usuário e as outras aplicações (MACORATTI, 2005b).

Para identificar a fronteira do processo de contagem, devem ser adotadas as seguintes regras (MACORATTI, 2005b; VAZQUEZ *et al.*, 2010):

- Deve ser considerado o ponto de vista do usuário, ou seja, o que o usuário pode entender e descrever como função da aplicação.
- A fronteira entre aplicações relacionadas deve considerar a funcionalidade das aplicações em termos das funções de negócio identificadas pelo usuário, e não sob o ponto de vista das interfaces necessárias.
- Em projetos de melhoria, a fronteira estabelecida no início do projeto deve estar de acordo com a fronteira já estabelecida para a aplicação sendo modificada.

Segundo Vazquez *et al.* (2010), as seguintes dicas auxiliam na identificação da fronteira da aplicação:

- Obter uma documentação do fluxo de dados no sistema e desenhar uma fronteira em volta para destacar quais partes são internas e externas à aplicação
- Identificar áreas funcionais como entidades e processos;
- Verificar como o grupo de dados são mantidos;
- Verificar como a aplicação é gerenciada; se é desenvolvida ou mantida na sua totalidade por uma equipe distinta.
- Comparar os critérios utilizados em outras métricas como esforço, duração, custo e defeitos. As fronteiras para efeito da análise de pontos de função e as outras métricas devem ser as mesmas.
- Verificar se o *software* possui ordens de serviço específicas e independentes.

A fronteira da aplicação é determinada com base na visão do usuário, está baseada no funcional como pode ou é visto pelo usuário da aplicação, não em considerações técnicas. Ela atua como uma membrana imaginária por onde os dados, processados por transações atravessam, entrando e saindo da aplicação (CAMPOS, 2010d).

Segundo Vazquez *et al.* (2010), a etapa de estabelecer a fronteira da aplicação é considerada uma das mais importantes do processo de contagem. Se a fronteira de aplicação não estiver bem definida, há risco do restante da contagem

não refletir o real tamanho da aplicação, comprometendo todo o trabalho. Além disso, o posicionamento incorreto da fronteira pode alterar a perspectiva da contagem de uma visão lógica (princípio da análise de pontos de função) para uma visão física, gerando conseqüências, como as seguintes:

- Contagem duplicada da mesma transação por várias "aplicações", uma vez que cada uma delas contribui com um subprocessamento para a transação do negócio.
- Contagem de funções de transferência de dados entre plataformas/ "aplicações". Por exemplo, a carga diária de uma tabela do servidor para uma aplicação cliente poderia ser contada como uma Saída Externa (servidor) e uma Entrada Externa (cliente), além de dois arquivos (a versão servidor e cliente da tabela). No entanto, o requisito poderia ser simplesmente o de validar as transações contra essa tabela.
- Dificuldade na determinação do tipo de transação. Como uma "aplicação" provê somente parte do processamento de uma transação do negócio, esse processamento pode não satisfazer as regras do IFPUG para determinação do tipo da transação.
- Duplicidade na contagem de arquivos. Por exemplo, um mesmo arquivo poderia ser contado como um Arquivo Lógico Interno para uma "aplicação" e um Arquivo de Interface Externa para outra "aplicação".

3.4.3 CONTAGEM DAS FUNÇÕES DO TIPO DADOS

A terceira etapa do processo de contagem é realizar a contagem das funções do tipo dados, que representam as funcionalidades fornecidas ao usuário para atender os requisitos de armazenamento de dados internos e externos à aplicação. São funções do tipo dados: Arquivos Lógicos Internos e Arquivos de Interface Externa (HAZAN, 2001b; VAZQUEZ *et al.*, 2010). Segundo Hazan (2001b), ambas as funções de tipos de dados são grupos de dados logicamente relacionados ou informações de controle identificadas pelo usuário. O que difere essas funções é o

local de armazenamento de seus dados perante a aplicação a ser contada, pois um ALI é mantido dentro da fronteira da aplicação, enquanto que o AIE é apenas referenciado pela aplicação, mas mantido dentro da fronteira de outra aplicação. Portanto:

- **Arquivo Lógico Interno (ALI)**: É um grupo de dados ou informações de controle logicamente relacionados e identificados pelo usuário que são mantidos (adicionados, alterados ou excluídos) através de um ou mais processos elementares da aplicação sendo contada.
- **Arquivo de Interface Externa (AIE):** É um grupo de dados ou informações de controle logicamente relacionados e identificados pelo usuário que são referenciados (lidos) por meio de um ou mais processos elementares dentro da fronteira da aplicação sendo contada. O AIE de uma aplicação é obrigatoriamente um ALI de outra aplicação.

Um grupo de dados logicamente relacionado refere-se a dados relacionados em um nível que o usuário consegue perceber como sendo de extrema importância para permitir que a aplicação realize uma atividade definida (IFPUG *apud* MACORATTI, 2005b). Já as informações de controle, são dados usados pela aplicação para garantir total conformidade com os requisitos das funções do negócio definidas pelo usuário. Elas especificam o que, quando e como os dados devem ser processados e influenciam um processo elementar da aplicação sendo contada (MACORATTI, 2005b; VAZQUEZ *et al.*, 2010).

Um processo elementar é a menor unidade de alguma atividade significativa para o usuário final. Ele deve ser autocontido e deixar a aplicação em um estado consistente. Em resumo, é uma transação completa (VAZQUEZ *et al.*, 2010).

3.4.3.1 Determinação da Complexidade

Para a etapa de contagem de Funções do tipo dado o primeiro passo a ser realizado é a identificação dos arquivos lógicos internos e arquivos de interface externa. Após a identificação ser realizada, cada uma dessas funções de dados

deve ser classificada segundo sua complexidade funcional baseada no número de Tipo de Dados (TD) e no número de Tipo de Registros (TR) (HAZAN, 2001b; VAZQUEZ *et al.*, 2010).

A complexidade funcional está dividida em três tipos: alta, média e baixa conforme a Figura 4, apresentada a seguir (IFPUG *apud* FABIAN, 2008):

	Tipos de dados			
Tipos de registros		< 20	20 – 50	> 50
	1	Baixa	Baixa	Média
	2 – 5	Baixa	Média	Alta
	> 5	Média	Alta	Alta

Figura 4 - Complexidade funcional de ALI e AIE
Fonte: IFPUG(s.d) *apud* FABIAN (2008)

3.4.3.2 Regras de Contagem de Tipos de Dados

Um tipo de Dado significa ser um campo único, não repetitivo e reconhecido pelo usuário. Representa um campo de dados que formula uma ocorrência de informação completa. Para contagem de um tipo de dado devem ser observadas as seguintes regras (VAZQUEZ *et al.*, 2010; FABIAN, 2008):

- Conte um tipo de dado para cada campo único reconhecido pelo usuário e não repetido, utilizado por um ALI ou AIE por meio de um processo elementar;
- Quando duas aplicações mantêm ou referenciam o mesmo ALI ou AIE conte apenas os campos utilizados pela aplicação sendo contada;
- Deverá ser considerado um tipo de dado para cada campo solicitado pela aplicação para estabelecer um relacionamento com outro ALI ou AIE.

3.4.3.3 Regras de Contagem de Tipos de Registros

Um tipo de registro representa um subgrupo de dados reconhecido pelo usuário, dentro de um ALI ou AIE. Existem dois tipos de subgrupo que podem ser identificados como registros (VAZQUEZ *et al.*, 2010; MACORATTI, 2005b):

- Opcionais: São subgrupos de dados que o usuário tem a opção de não informar no processo elementar que cria ou adiciona dados ao arquivo.
- Obrigatórios: São subgrupos de dados que o usuário deve utilizar pelo menos uma vez durante o processo elementar que cria ou adiciona dados ao arquivo.

As regras que devem ser utilizadas para determinar o número de tipos de registros tanto em ALI como em AIE são (VAZQUEZ *et al.*, 2010; FABIAN, 2008):

- Deve ser considerado um tipo de registro para cada subgrupos de dados de um ALI ou AIE (Cada função de dado tem um subgrupo de TD a ser contado com um TR);
- Conte um TR adicional para cada um dos seguintes subgrupos lógicos de TDs que contém mais de um TD: Entidade associativa com atributos não chave; Subtipo (outro que não o primeiro subtipo); e entidade atributiva em um relacionamento que não seja mandatório.
- Caso não identificado nenhum subgrupo de dados, deve ser considerado um tipo de registro para cada ALI ou AIE.

3.4.3.4 Determinação da Contribuição

Após a determinação da complexidade dos arquivos (ALI e AIE), deve-se calcular sua contribuição de acordo com a Figura 5:

TIPO DE FUNÇÃO	Baixa	Média	Alta
ALI	7 PF	10 PF	15 PF
AIE	5 PF	7 PF	10 PF

Figura 5 - Contribuição do ponto por função das funções do tipo dado
Fonte: IFPUG(S.d.) *apud* FABIAN (2008)

3.4.4 CONTAGEM DAS FUNÇÕES DO TIPO TRANSAÇÃO

A próxima etapa do Processo de Contagem é a contabilização das funções do tipo Transação que por sua vez representam as funcionalidades de processamento de dados do sistema fornecidas para o usuário. São funções do tipo transação (HAZAN, 2001b; VAZQUEZ *et al.*, 2010):

- ***Entradas Externas (EEs):*** são processos elementares (transações) que processam dados ou informações de controle que entram pela fronteira da aplicação. O objetivo principal de uma EE é manter um ou mais ALIs e/ou alterar o comportamento do sistema. Ex: Incluir Cliente; Alterar Funcionários; Excluir Pedido.
- ***Saídas Externas (SEs)***: são processos elementares que enviam dados ou informações de controle para fora da fronteira da aplicação. Seu objetivo é apresentar informações ao usuário que foram recuperadas através de um processamento lógico e não apenas uma simples recuperação de dados. Uma SE deve envolver cálculos ou criação de dados derivados ou também, pode manter um ALI ou alterar o comportamento do sistema. Ex: Relatório do total de vendas por funcionário, tela de *Login* (com criptografia).
- ***Consulta Externa (CE):*** assim como uma SE, é um processo elementar que envia dados (ou informações de controle) para fora da fronteira da aplicação. Porém, seu objetivo é apresentar informação para o usuário, por meio apenas de uma recuperação das informações, sem a realização de nenhum cálculo nem a criação de dados derivados. Nenhum ALI é mantido

durante sua realização, nem o comportamento do sistema é alterado. Ex: Consulta de Clientes, *Drop-Downs* desde que recuperem dados de arquivos lógicos (ALIs e/ou AIEs). Os *Drop-Downs* estáticos, com valores codificados diretamente no programa-fonte, não são contados.

3.4.4.1 Determinação da Complexidade

Assim como nas funções do tipo dado, deve-se também classificar as funções do tipo transação quanto a sua complexidade funcional na aplicação. Cada EE, SE e CE deve ser classificada com relação a sua complexidade funcional em alta, média ou baixa, de acordo com o número de arquivo referenciado (AR) e quanto ao número de tipo de dado (IFPUG *apud* FABIAN, 2008).

Concluído o levantamento do número de arquivos referenciados e tipos de dados de cada EE, SE e CE, deve-se classificá-los conforme as figuras 6 e 7:

Arquivos referenciados	Tipos de dados		
	< 5	5 – 15	> 15
< 2	Baixa	Baixa	Média
2	Baixa	Média	Alta
> 2	Média	Alta	Alta

Figura 6 - Complexidade para entradas externas
Fonte: IFPUG(S.d.) *apud* FABIAN (2008).

Arquivos referenciados	Tipos de dados		
	< 6	6 – 19	> 19
< 2	Baixa	Baixa	Média
2 – 3	Baixa	Média	Alta
> 3	Média	Alta	Alta

Figura 7 - Complexidade para saídas externas e consultas externas
Fonte: IFPUG(S.d.) *apud* FABIAN (2008).

3.4.4.2 Regras de Contagem para Arquivo Referenciado

Um arquivo referenciado refere-se a um ALI lido ou atualizado pela função do tipo transação ou então pode ser um AIE lido pela função do tipo transação (VAZQUEZ *et al.*, 2010; FABIAN, 2008). As regras para contagem de um arquivo referenciado são:

- Contar um arquivo referenciado para cada ALI mantido (não aplicável em para transações do tipo CE);
- Contar apenas um arquivo referenciado para cada ALI que é mantido e lido (não aplicável em transações do tipo CE);
- Contar um arquivo referenciado para cada ALI ou AIE lido durante o processamento. Mesmo que o ALI e/ou AIE processar mais de um tipo de registro, deverá ser contado apenas uma vez.

É importante ressaltar que no processo de contagem para arquivos referenciados não poderão ser contados outros tipos de arquivos que não sejam ALI ou AIE. Além disso, não é válido contar o mesmo arquivo mais de uma vez, mesmo que a transação faça várias leituras ou atualizações nele.

3.4.4.3 Regras de Contagem de Tipos de Dados

Conforme vimos anteriormente, um tipo de dado é um campo único, não repetido e reconhecido pelo usuário. As regras para contagem de tipos de dados para funções transacionais são (FABIAN, 2008; VAZQUEZ *et al.*, 2010):

- Contar um tipo de dado para cada campo não repetido e reconhecido pelo usuário, que atravessa a fronteira de aplicação (entrando e/ou saindo) e utilizado no processo. Independente de quantas vezes o campo entra ou sai da fronteira da aplicação, ele deve ser contado apenas uma vez;

- Contar um único tipo de dado quando uma mensagem é enviada para fora da fronteira da aplicação, como tratamento de exceções, mensagem de término de processamento, confirmação da conclusão de uma transação;
- Contar um tipo de dado para a forma de acionar uma ação a ser tomada. Mesmo que hajam várias formas de ativar o mesmo processo, deve ser contado apenas um tipo de dado.

Não são considerados tipos de dados:

- Dados Literais;
- Variáveis de paginação ou campos automáticos gerados pela aplicação;
- Auxílios de navegação.
- Atributos gerados dentro da fronteira da aplicação por uma função transacional e salvos em um ALI sem sair da fronteira.
- Atributos recuperados de um AIE ou ALI para participar no processamento sem sair da fronteira.

3.4.4.4 Determinação da Contribuição

Após a conclusão da etapa de determinação da complexidade das funções do tipo transação, deve ser calculada sua contribuição utilizando os seguintes critérios:

Tipo de função	Baixa	Média	Alta
EE	3 PF	4 PF	6 PF
SE	4 PF	5 PF	7 PF
CE	3 PF	4 PF	6 PF

Figura 8 - Contribuição dos pontos por função das funções de transação
Fonte: IFPUG *apud* FABIAN(2008)

3.4.5 CONTAGEM DE PONTOS DE FUNÇÃO NÃO AJUSTADOS

Uma vez contadas às funções de dados e as funções transacionais, a próxima etapa do processo de contagem é calcular os PFs não ajustados de uma aplicação. Os pontos de função não ajustados medem os requisitos específicos do usuário, permitindo assim apontar um requisito (um relatório, um gráfico, uma transação de entrada de dados, etc) e dizer o seu valor em PF (VAZQUEZ *et al.*, 2010).

Segundo o IFPUG a contagem do tamanho funcional, mesmo de pontos de função não ajustados é feita com base no cálculo a partir dos três tipos de contagem: projeto de desenvolvimento, de melhoria e aplicação (VAZQUEZ *et al.*, 2010). Porém, Hazan (2001b) apresenta uma maneira simples para o cálculo dos pontos de função não ajustados que pode ser feita da seguinte forma:

- Para cada um dos cinco tipos de função (ALI, AIE, EE, SE e CE), são computados os totais de pontos de função (NPFi), segundo a equação (2):

$$\mathbf{NPFi} = \sum_{j=1}^{3} NCi,j * Ci,j \qquad (2)$$

Onde NCi,j representa o número funções do tipo i (i variando de 1 a 5, segundo os tipos de função existentes: ALI, AIE, EE, SE e CE) que foram classificados na complexidade j (j variando de 1 a 3, segundo os valores de complexidade: simples, média e complexa).

Ci,j = valor da contribuição da complexidade j no cálculo dos pontos da função i, de acordo com as figuras 5 e 8 apresentadas anteriormente.

- O total de pontos de função não ajustados (PFNA) é dado pelo somatório dos pontos das tabelas de função, conforme a equação (3):

$$\mathbf{PFNA} = \sum_{i=1}^{5} NPFi \qquad (3)$$

Sendo que i varia de 1 a 5, segundo os tipos de função existentes (ALI, AIE, EE, SE e CE).

3.4.6 DETERMINAR O VALOR DO FATOR DE AJUSTE

A técnica de Análise por Pontos de Função considera que outros fatores afetam o tamanho funcional de um sistema. Fatores estes, que estão relacionados com características da aplicação. No cálculo dos PF brutos não é levada em conta a tecnologia usada nem os requisitos não funcionais do sistema. Por este motivo, esta etapa tem por objetivo estabelecer um fator de ajuste para a soma dos pontos de função não-ajustados. Ele ajusta os pontos de função em +/- 35% de acordo com a influência de 14 características definidas pelo IFPUG (MACORATTI, 2005b; FABIAN, 2008).

Para adequar-se à norma do padrão ISO/IEC de medição funcional que não compreende esta fase, o IFPUG tornou este passo opcional na APF (FABIAN, 2008).

Para a determinação do Valor do Fator de Ajuste (VAF) deve-se considerar 14 Características Gerais de Sistema (CGS) determinadas pelo IFPUG. Para cada característica é preciso determinar um nível de influência na aplicação de acordo com a Figura 9 (FABIAN, 2008).

Grau	Descrição
0	Nenhuma influência
1	Influência mínima
2	Influência moderada
3	Influência média
4	Influência significante
5	Influência forte

Figura 9 - Grau de Influencia para as CGS
Fonte: HAZAN (2001b)

As características gerais de sistema para determinação do valor do fator de ajuste dos pontos por função não-ajustados são (FABIAN, 2008; VAZQUEZ *et al.*, 2010; HAZAN, 2001b):

- ***Comunicação de dados***: Refere-se ao nível em que a aplicação comunica-se diretamente com o processador. Os dados são enviados e recebidos por meio de recursos de comunicação, como terminais conectados localmente à unidade de controle e protocolo de comunicação. Descrever se a aplicação utiliza protocolos diferentes para recebimento/envio das informações do sistema;
- ***Processamento distribuído***: Representa o nível de transferência de dados que a aplicação faz entre seus componentes dentro da fronteira de aplicação;
- ***Performance***: Refere-se ao nível de tempo de resposta e taxa de transações que influenciam o desenvolvimento da aplicação. A pergunta que deve ser avaliada para essa CGS é: *Quão rápida deve ser a aplicação e o quanto isto influencia o projeto?*;
- ***Configuração altamente utilizada***: Refere-se ao nível que restrições de recursos computacionais influenciam no desenvolvimento da aplicação. A questão a ser avaliada para essa CGS é: *O quanto a infraestrutura influencia o projeto?*;
- ***Volume de transações***: Descreve o nível que o alto volume de transações influencia o projeto, desenvolvimento, instalação e suporte da aplicação;
- ***Entrada de dados on-line***: Descreve em que nível são efetuadas entradas de dados na aplicação através de transações interativas;
- ***Eficiência do usuário final***: Refere-se ao nível de considerações sobre fatores humanos e facilidade de uso pelo usuário final influenciam o desenvolvimento da aplicação. As funções interativas fornecidas pela aplicação enfatizam um projeto para o aumento da eficiência do usuário final, tais como:
  - Auxílio à navegação (teclas de função, acesso direto e menus dinâmicos);

- Menus Documentação e *help on-line*;
- Movimento automático do cursor;
- Movimento horizontal e vertical de tela;
- Impressão remota (via transações *on-line*);
- Teclas de função preestabelecidas;
- Processos *batch* submetidos a partir de transações *on-line*;
- Utilização intensa de campos com vídeo reverso, intensificados, sublinhados, coloridos e outros indicadores;
- Impressão da documentação das transações *on-line* através de *hard copy;*
- Utilização de *mouse*;
- Menus *pop-up*;
- O menor número possível de telas para executar as funções de negócio;
- Suporte bilíngüe;
- Suporte multilíngüe.

- ***Atualização on-line***: Refere-se ao nível em que os arquivos lógicos internos são atualizados de forma *on-line*;
- ***Complexidade de processamento***: Refere-se ao nível em que o processamento lógico e/ou matemático influencia o desenvolvimento da aplicação;
- ***Reusabilidade***: Descreve em que nível a aplicação e seu código foram projetados, desenvolvidos e suportados para serem utilizados em outras aplicações;
- ***Facilidade de instalação***: Refere-se ao nível em que a conversão de ambientes preexistentes influencia o desenvolvimento da aplicação. Um plano e/ou ferramentas de conversão e instalação foram fornecidos e testados durante a fase de teste do sistema;
- ***Facilidade de operação***: Mede o nível em que a aplicação atende a alguns aspectos operacionais como inicialização, segurança e recuperação. A aplicação minimiza a necessidade de atividades manuais como montagem de fitas, manipulação de papel e intervenção manual do operador;

- ***Múltiplos locais:*** Refere-se ao nível em que a aplicação foi projetada, desenvolvida e suportada para diferentes ambientes de *hardware* e *software*;
- ***Facilidade de mudanças:*** Refere-se ao nível em que a aplicação foi desenvolvida para facilitar a mudança de sua lógica e/ou estrutura de dados. As seguintes características podem ser válidas para a aplicação:
    - São fornecidos mecanismos de consulta flexível, para atender necessidades simples; por exemplo, lógica de e/ou aplicada a apenas um arquivo lógico;
    - São fornecidos mecanismos de consulta flexível, para atender necessidades de complexidade média; por exemplo, lógica de e/ou aplicada a mais de um arquivo lógico;
    - São fornecidos mecanismos de consulta flexível, para atender necessidades de complexidade alta; por exemplo, lógica de e/ou combinadas em um ou mais arquivos lógicos;
    - Dados de controle de negócio são armazenados em tabelas que são mantidas pelo usuário através de processos on-line, mas mudanças têm efeitos somente no dia seguinte;
    - Dados de controle de negócio são armazenados em tabelas que são mantidas pelo usuário através de processos on-line, as mudanças têm efeito imediatamente.

Após atribuir um valor de influência para cada uma das 14 características gerais do sistema, estes valores devem então ser somados o que resultará no Grau de Influência Total (GIT), conforme a equação (4) (MACORATTI, 2005b):

$$\mathbf{GIT} = \sum_{i=1}^{14} GI_i \qquad (4)$$

O valor do fator de ajuste é calculado de acordo com a equação (5):

$$\mathbf{VFA =} (GIT * 0{,}01) + 0{,}65 \qquad (5)$$

Se o fator de ajuste de valor é igual a 1,00, a influência total das características gerais do sistema é neutra. Nesta situação, a contagem dos pontos

de função ajustados equivale a contagem de pontos de função não ajustados (MACORATTI, 2005b).

3.4.7 CALCULAR OS PONTOS DE FUNÇÃO AJUSTADOS

A última etapa da APF é calcular os pontos de função ajustados, o qual em posse dos pontos por função não-ajustados e o fator de ajuste definido, é aplicada uma forma matemática, específica para cada tipo de contagem: projeto de desenvolvimento, de melhoria e aplicação. Nesta etapa são usados três tipos de fórmulas matemáticas diferentes para chegar ao resultado final, apresentadas a seguir como descritas no manual do IFPUG, dependendo do tipo de cálculo que se deseja realizar (VAZQUEZ *et al.*, 2010).

3.4.7.1 Projeto de Desenvolvimento

Antes de iniciar o cálculo do número de Pontos de Função Ajustados de um projeto de Desenvolvimento é necessário compreender dois componentes fundamentais:

- Funcionalidade da aplicação requisitada pelo usuário para o projeto: Compreendem as funções usadas depois da instalação do sistema. Elas existem para satisfazer as necessidades de saída do negócio do usuário (IFPUG, 1999 *apud* MACORATTI, 2005b; VAZQUEZ *et al.*, 2010);
- Funcionalidade de conversão incluídas pelos usuários como requisitos: Compreendem funcionalidades disponíveis somente na instalação do sistema. Elas existem para converter dados ou proporcionar outros requisitos estabelecidos pelo usuário e necessários à conversão. Após a instalação estas funções são descartadas por não serem mais necessárias (IFPUG, 1999 *apud* MACORATTI, 2005b; VAZQUEZ *et al.*, 2010).

A fórmula a ser aplicada nos projetos de desenvolvimento é apresentada pela equação (6) (VAZQUEZ *et al.*, 2010):

$$\textbf{DFP} = (UFP + CFP) \quad (6)$$

Onde:

- DFP: Representa o tamanho (Número de pontos por função) do projeto de desenvolvimento;
- UFP: Número de pontos de função não-ajustados das funções disponíveis após a instalação da aplicação, exceto as funções de conversão (contagem final do projeto, o realizado - entregue);
- CFP: Número de pontos de função das funções de conversão;

Para obter os pontos de função ajustados, basta então multiplicar o resultado pelo fator de ajuste (VAF = Valor do fator de ajuste), conforme a equação (7) (FABIAN, 2008):

$$\textbf{DFP} = (UFP + CFP) \times VAF \quad (7)$$

Segundo Fabian (2008) a fórmula aplicada para estimativa em projetos não é complexa, o que determinará o sucesso da estimativa está realmente nas fases iniciais do processo de contagem.

3.4.7.2 Projeto de Melhoria

Segundo o IFPUG o conceito de melhoria envolve apenas manutenções evolutivas na aplicação, ou seja, alterações feitas na aplicação para atender aos novos requisitos de negócio do usuário. Não devem ser levadas em conta manutenções corretivas e preventivas (VAZQUEZ *et al.*, 2010).

A diretriz básica para considerar que uma função do tipo dado (ALI ou AIE) foi alterada é quando ela foi modificada em sua estrutura com alguma inclusão, alteração ou exclusão de campos ou atributos. Neste caso deve-se contar a funcionalidade toda no projeto de melhoria, não apenas os campos afetados pela manutenção (VAZQUEZ *et al.*, 2010).

Uma função do tipo transação é considerada alterada quando há alteração em um dos seguintes itens (VAZQUEZ *et al.*, 2010; MACORATTI, 2005b):

- Tipos de dados – Se houve inclusão, alteração ou exclusão da função.
- Arquivos referenciados – Se foram incluídos, excluídos ou alterados da função.
- Lógica de processamento – Se qualquer lógica for incluída, alterada ou excluída.

Os componentes para o cálculo dos pontos de função de um projeto de melhoria são:

- Funcionalidade da aplicação requisitada pelo usuário para o projeto: funções adicionadas, alteradas ou excluídas da aplicação pelo projeto de melhoria (VAZQUEZ *et al.*, 2010).
- Funcionalidade de conversão: Consiste dos pontos de função entregues por causa de qualquer funcionalidade de conversão requerida pelo usuário (IFPUG, 1999 *apud* MACORATTI, 2005b).

A fórmula a ser aplicada nos projetos de melhoria é apresentada pela equação (8) (VAZQUEZ *et al.*, 2010):

$$\mathbf{EFP} = (ADD + CHGA + CFP + DEL) \quad (8)$$

Os termos da fórmula são:

- EFP: Número de pontos de função do projeto de melhoria;
- ADD: Número de pontos de função não-ajustados das novas funções;
- CHGA: Número de pontos de função não-ajustados das funções modificadas, levando em consideração o funcionamento após a alteração;
- CFP: Número de pontos de função não-ajustados de funções de conversão;
- DEL: Número de pontos de função não-ajustados das funções excluídas da aplicação.

Para obter o valor dos pontos de função ajustados, a fórmula a ser aplicada adiciona o valor do fator de ajuste antes do início do projeto (VAFB) e o valor do fator

de ajuste depois da conclusão do projeto de melhoria (VAFA) conforme mostra a equação (9) (MACORATTI, 2005b):

$$\textbf{EFP} = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB) \quad (9)$$

3.4.7.3 Aplicação

Para aplicações existem duas fórmulas matemáticas para calcular o número de pontos de função. Uma para a primeira contagem dos pontos de função da aplicação, onde são contadas todas as funcionalidades requeridas pelo usuário de uma aplicação instalada. As funções da conversão de dados não devem ser computadas no tamanho da aplicação entregue, pois elas existirão somente para o processo de implantação do aplicativo. A segunda para recalcular o seu tamanho após um projeto de melhoria ter alterado suas funcionalidades (VAZQUEZ *et al.*, 2010; FABIAN, 2008).

A primeira fórmula é apresentada pela equação (10):

$$\textbf{AFP} = ADD \quad (10)$$

Onde:

- AFP: Tamanho da Aplicação;
- ADD: Número de pontos de função não-ajustados das funções.

Para o valor dos pontos de função ajustados, deve-se utilizar a equação (11):

$$\textbf{AFP} = ADD * VAF \quad (11)$$

Onde:

- VAF: Valor do fator de ajuste da aplicação.

Após o projeto de melhoria, deve-se aplicar a fórmula apresentada pela equação (12):

$$\textbf{AFPA} = (AFPB + ADD + CHGA) – (CHGB + DEL) \quad (12)$$

Os termos da fórmula são:

- AFPA: Número de pontos de função ajustados da aplicação;
- AFPB: Número de pontos de função não-ajustados antes do projeto de melhoria;

- ADD: Número de pontos de função não-ajustados das funções incluídas no projeto;
- CHGA: Número de pontos de função não-ajustados das funções alteradas depois do término do projeto;
- CHGB: Número de pontos de função não-ajustados das funções alteradas antes do início do projeto;
- DEL: Número de pontos de função não-ajustados das funções excluídas pelo projeto;

Para obter os pontos de função ajustados, deve-se multiplicar o resultado pelo VAFA (Valor do fator de ajuste depois do projeto), conforme a equação (13):

$$\textbf{AFPA} = [(AFPB + ADD + CHGA) – (CHGB + DEL)] \times VAFA \quad (13)$$

Quando um projeto de melhoria é concluído, o número de pontos de função da aplicação deve ser atualizado para refletir as modificações na aplicação (VAZQUEZ *et al.*, 2010).

4 ESTUDO DE CASO

4.1 LOCAL DO ESTUDO

O estudo foi aplicado na empresa Logic Tecnologia da Informação (Logic TI) localizada na cidade de Foz do Iguaçu. Inserida no Espaço de Desenvolvimento Empresarial do Parque tecnológico Itaipu em Foz do Iguaçu - PR, a Logic TI surgiu no início de 2005, com o objetivo de desenvolver o parque de *software* regional, criando uma empresa de desenvolvimento de *software* de qualidade, aplicando modelos de qualidade consagrados, as mais modernas tecnologias e programa contínuo de treinamento.

A empresa atua no modelo de Fábrica de *Software*, buscando prover, principalmente a clientes com extensa estrutura de Tecnologia da Informação, o serviço de desenvolvimento e manutenção de *software* sob encomenda, ou seja, que apresentam características específicas e totalmente adaptadas às necessidades e modelo de negócio do cliente, organizando, controlando, agregando valor, minimizando custos e maximizando lucros.

Para empresas de médio e grande porte, com setores próprios de Informática, a Logic TI busca permitir a essas empresas, reduzir custos de contratação e suprir a deficiência de mão-de-obra qualificada e especializada, terceirizando o processo de desenvolvimento de software como um todo ou em suas fases de Especificação, Projeto, Implementação, Testes e Manutenção individualmente. O contrato de serviço firmado entre as empresas contratante e a Logic TI em geral é realizado em Pontos de Função, pois ao medir o tamanho do *software* a empresa Logic TI consegue determinar custos e estimar prazo de entrega do produto final.

4.2 TIPO DE PESQUISA OU TÉCNICAS DE PESQUISA

O trabalho proposto seguiu a metodologia de realização de uma aplicação prática posterior à análise de técnicas de medição de tamanho de software, em que se optou pela Análise de Pontos de Função. Neste contexto, o estudo de caso consistiu em estimar o tamanho de um projeto de desenvolvimento de *software* a ser construído pela empresa Logic TI para um de seus clientes.

4.3 COLETA DOS DADOS

O projeto utilizado como *case* foi o projeto de desenvolvimento de um Sistema de Controle de Bens Patrimoniais (CBP) para a empresa *X*[4]*.* O escopo do projeto foi definido em: O CBP será um sistema criado para registrar os Bens Patrimoniais e automatizar as movimentações, transferências e o inventário dos mesmos.

O processo do sistema de controle de Bens Patrimoniais consiste do Administrador do sistema cadastrar os Bens Patrimoniais com suas informações específicas adicionais, como: classificação e área que este patrimônio pertence e quem será seu usuário e responsável.

O sistema permitirá controlar tanto as movimentações, como as transferências dos Bens Patrimoniais. A movimentação consistirá na troca de usuário, área e localização, já na transferência podem ser alterados os dados do usuário, responsável, área e a localização do Patrimônio.

O sistema também terá a funcionalidade de realizar o inventário dos bens patrimoniais de determinado escritório, área, classificação e/ou por usuário e nacionalidade do usuário. O sistema disponibilizará a relação de bens que estão sob responsabilidade de um determinado usuário, para que ele possa visualizá-las, verificar e informar quais bens patrimoniais ainda estão e, os que já não estão mais

[4] Empresa X – Por questões legais, o real nome da empresa cliente foi omitido e será considerado o nome X.

sob sua responsabilidade. Além disso, o usuário ainda tem a opção de informar o estado dos bens que ainda lhe pertencem e justificar a falta dos demais.

O CBP deverá interagir com outros sistemas já existentes na empresa X. A interface será realizada com:

- Sistema RHI – Sistema que disponibilizará as informações/lista de colaboradores ativos na empresa X, para que no CBP seja possível definir responsável e/ou o usuário dos Bens Patrimoniais;
- Sistema TGI – Sistema que disponibilizará as informações/lista das áreas (divisões departamentais) da empresa X, para que no CBP seja possível informar a qual área pertence cada Bem Patrimonial.

Os requisitos preliminares que descrevem o escopo do projeto CBP utilizado no processo de contagem de Pontos de Função foram divididos em duas categorias: Funcionais e Não Funcionais, sendo que os requisitos funcionais são:

- **RF-01 Manter o cadastro de Classificação dos Bens Patrimoniais:** O sistema deverá manter interface para inclusão, exclusão, alteração e consulta de Classificações dos Bens Patrimoniais. A consulta das classificações deverá estar ordenada por Nome;
- **RF-02 Manter cadastro de Terceirizado**: O sistema mantém interface para inclusão, exclusão, alteração e consulta de Terceirizados. Os Terceirizados são utilizados no cadastro de Bem Patrimonial e nas movimentações dos Patrimônios;
- **RF-03 Manter cadastro de Bens Patrimoniais**: O sistema deve permitir a inclusão, atualização, exclusão e consulta de formulários de cadastro de Bem Patrimonial. Nas consultas de Bens Patrimoniais, os mesmos devem ser agrupados por Responsável e Status e serão ordenados por número do BPM;
- **RF-04 Consultar Colaborador do RHI**: O sistema obtém automaticamente a lista de Colaboradores ativos, gerada pelo sistema RHI, a fim de definir quem será o responsável e/ou usuário do Bem Patrimonial. Na consulta, o sistema disponibiliza filtro por Nome e/ou Matrícula. O sistema deverá recuperar os campos: Nome, Matrícula, Área e Ramal;
- **RF-05 Consultar Área do TGI**: O sistema obtém automaticamente a lista de Áreas, gerada pelo sistema TGI, a fim de definir a Área da Localização do

Bem Patrimonial. Na consulta, o sistema disponibiliza filtro por Descrição e/ou Sigla e mostra uma lista com os mesmos campos;

- **RF-06 Consultar Arquivos em Disco**: O sistema mantém interface para consulta de arquivos armazenados em disco, para que o ator escolha qual arquivo poderá ser anexado ao Bem Patrimonial ou a Transferência;
- **RF-07 Associar anexos de TBPM (Transferência de Bens Patrimoniais Móvel)**: O sistema deverá manter interface para que o administrador faça a consulta **[RF-06],** associação e dissociação de arquivos ao Bem Patrimonial ou à Transferência de Bens Patrimoniais**.** Será feito o *upload* dos arquivos associados à solicitação para um servidor e, posteriormente, esses arquivos ficarão disponíveis para *download*. São dados dos Anexos: Nome do Anexo, Caminho Físico;
- **RF-08 Gerar impressão da consulta de Bens Patrimoniais:** O sistema disponibiliza a opção de impressão para a consulta de Bens Patrimoniais;
- **RF-09 Disponibilizar Consulta de Movimentações dos Bens Patrimoniais**: O sistema deverá manter interface para consulta dos Bens Patrimoniais, e assim poder visualizar suas movimentações. Os Bens Patrimoniais serão agrupados por Responsável e Status. Ao selecionar um Bem Patrimonial e acionar a visualização do mesmo, o sistema deverá exibir as movimentações do mesmo;
- **RF-10 Manter registro de Movimentações de Bens Patrimoniais**: O sistema permite o registro de movimentação e consulta de formulários de registro de movimentações de bens patrimoniais.
- **RF-11 Associar Bem Patrimonial ao Formulário de Registro de Movimentação de Bens Patrimoniais**: O sistema deverá manter interface para associação e dissociação de Bens Patrimoniais ao Formulário Registro de Movimentação de Bens Patrimoniais.
- **RF-12 Disponibilizar Consulta de Transferências dos Bens Patrimoniais**: O sistema deverá manter interface para consulta das transferências dos bens patrimoniais. As transferências serão agrupadas por Código da Transferência e Data.

- **RF-13 Manter registro de Transferência de Bens Patrimoniais:** O sistema permite o registro de transferências e consulta de formulários de registro de Transferências de bens patrimoniais.
- **RF-14 Associar Bem Patrimonial ao Formulário de Registro de Transferência de Bens Patrimoniais**: O sistema deverá manter interface para associação e dissociação de Bens Patrimoniais ao Formulário Registro de Transferência de Bens Patrimoniais.
- **RF-15 Disponibilizar Consulta de Inventários de Bens Patrimoniais:** O sistema deverá manter interface para consulta dos inventários de Bens Patrimoniais, trazendo somente os inventários solicitados para o usuário logado. Os inventários serão agrupados por Solicitante e Data da Solicitação e serão ordenados por Número do Inventário.
- **RF-16 Registrar Inventário de Bens Patrimoniais:** O sistema permite o registro de inventário e consulta de formulários de registro de inventários de bens patrimoniais.
- **RF-17 Disponibilizar Consulta de Solicitações de Inventário:** O sistema deverá manter interface para consulta das solicitações de inventários de Bens Patrimoniais. As solicitações são ordenadas por Número do Inventário.
- **RF-18 Solicitar Inventário de Bens Patrimoniais:** O sistema permite solicitar inventários de bens patrimoniais. O sistema gera inventário por colaboradores, envia uma notificação a cada colaborador e bloqueia as ações a serem tomadas com algum Bem Patrimonial envolvido neste Inventário.
- **RF-19 Associar Usuário ao Formulário de Solicitação de Inventários de Bens Patrimoniais:** O sistema mantém interface para associação e dissociação de Usuários ao Formulário de Solicitação de Inventários de Bens Patrimoniais.
- **RF-20 Disponibilizar Visualização da Solicitação de Inventário de Bens Patrimoniais:** O sistema mantém interface para visualização da Solicitação de Inventário de Bens Patrimoniais e permitirá consultar os inventários já finalizados.
- **RF-21 Permitir Notificar Inventário Pendente ao usuário:** Quando o Administrador/Gestor visualiza suas solicitações de inventário e a presença

de inventários pendentes, o sistema permite enviar uma notificação ao usuário que tenha inventário pendente, para que este preencha o mais rápido possível seu inventário.

- **RF-22 Disponibilizar opção para justificar situação do Bem Patrimonial no Inventário:** Quando o Usuário altera a situação de um item do seu inventário para uma situação diferente de “Indefinido”, o sistema disponibiliza uma opção para o usuário justificar a falta do patrimônio ou mesmo informar as condições do patrimônio caso este esteja em sua posse.
- **RF-23 Gerar impressão dos dados do inventário por Usuário:** O sistema disponibiliza a opção de impressão do detalhe dos dados do inventário de um Usuário.
- **RF-24 Gerar impressão dos dados do registro de Transferência:** O sistema disponibiliza a opção de impressão dos dados do registro de Transferência.
- **RF-25 Permitir Cancelar Inventário Pendente ao usuário:** Quando o Administrador/Gestor visualiza suas solicitações de inventário e a presença de inventários pendentes, o sistema permite cancelar um inventário pendente, para que seja permitido realizar ações (como Movimentação e Transferência) dos Bens Patrimoniais envolvidos no Inventário corrente.
- **RF-26 Registrar Histórico do Bem Patrimonial:** Sempre que um Bem Patrimonial é cadastrado, movimentado ou transferido, o sistema registra histórico, armazenando a data, hora, nome do Bem Patrimonial, área, nome do responsável e usuário, e o status.
- **RF-27 Notificar Envio de Inventário aos usuários:** Quando o Administrador solicita um novo inventário, o sistema notifica, via correio eletrônico, todos os usuários dos Bens Patrimoniais indicados na Solicitação, que o Inventário encontra-se disponível para sua análise e preenchimento.

Além destes requisitos funcionais, foram identificados os seguintes requisitos não funcionais:

- **RNF-01** - O sistema deverá apresentar a interface tanto no idioma português, quanto espanhol. A definição será realizada no momento do login;

- **RNF-02** - O sistema deverá manter um padrão de desenvolvimento de código, o qual deverá estar todo comentado.

Além dos requisitos identificados tem-se também a estrutura dos dados identificada para o sistema de Controle de Bens Patrimonais, conforme ilustra a Figura 10.

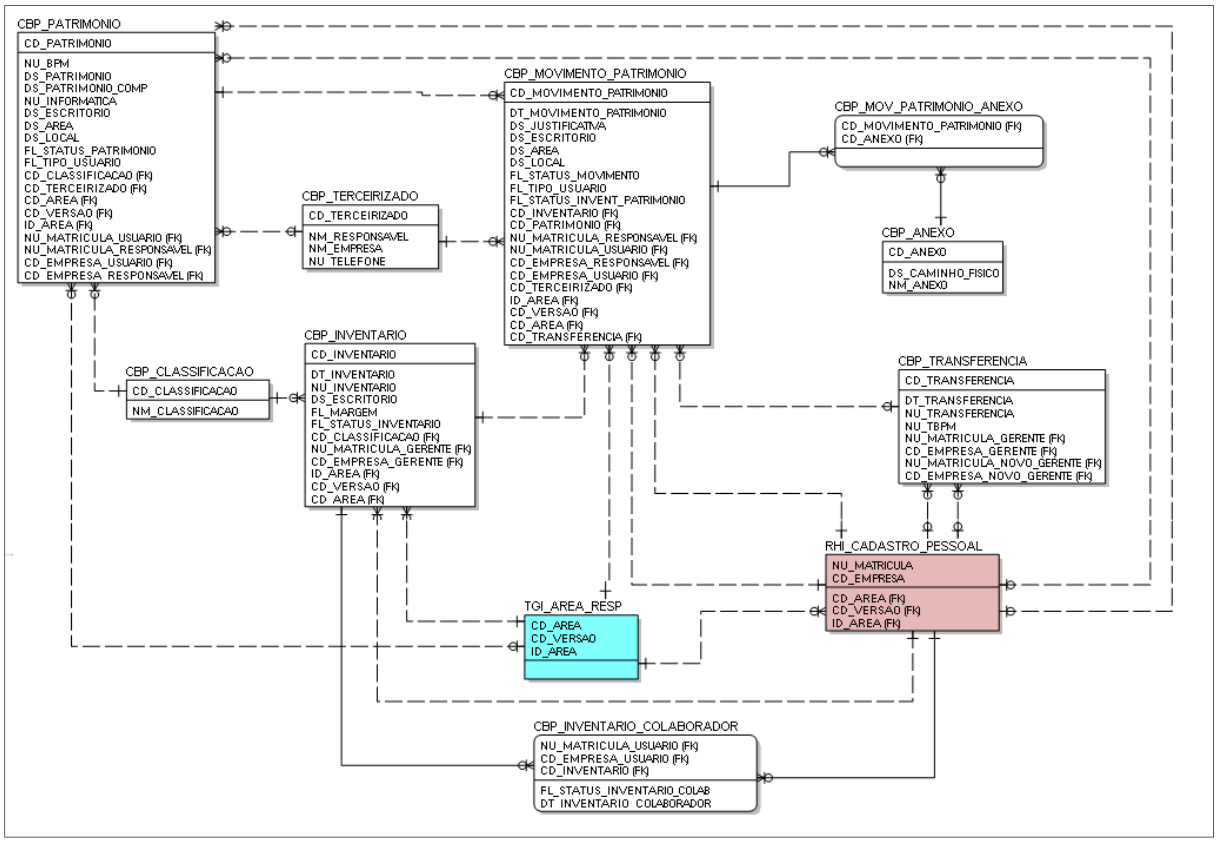

Figura 10 - Modelo de Dados do sistema CBP
Fonte: LOGIC TI (2011)

4.4 ANÁLISE DOS DADOS

Os dados utilizados no presente estudo de caso foram obtidos por meio de procedimentos internos (reuniões, análise de documentos da empresa e por meio de protótipos de tela) da empresa Logic TI. A empresa segue a metodologia do Processo Unificado de *Software* e a coleta de dados foi realizada durante a fase de concepção do sistema de Controle de Bens Patrimoniais.

5 RESULTADOS E DISCUSSÃO

Seguindo a metodologia de contagem de Pontos de Função, a estimativa de tamanho realizada no presente estudo de caso foi de um projeto de Desenvolvimento. Com base no escopo do projeto apresentado anteriormente no capítulo 4, foram identificadas as funções de dados e a quantidade de Tipo de Dados e Tipo de Registro conforme a Tabela 1:

Tabela 1 - Funções de Dados do sistema CBP

Descrição da Função	Tipo de Função de Dados	Tipo de Dado	Tipo de Registro
AREA (TGI – Tabela: Área)	AIE	3	1
COLABORADOR (RHI – Tabela: Cadastro Pessoal e Ramal)	AIE	4	2
PATRIMÔNIO (CBP - Tabela: Patrimônio, Classificação)	ALI	16	2
TERCEIRIZADO (CBP – Tabela: Terceirizado)	ALI	4	1
HISTÓRICO (CBP – Tabela: Movimento_Patrimonio, Anexo)	ALI	18	2
INVENTARIO (CBP – Tabela: Inventario)	ALI	11	2
TRANSFERENCIA (CBP – Tabela: Inventario)	ALI	5	1

Depois de identificadas as funções de dados, seguiu-se para a etapa de determinação da complexidade e da contribuição destas funções para a contagem de Pontos de Função deste projeto segundo as regras descritas nas Figuras 4 e 5 do capítulo 3. Para as funções levantadas segundo a Tabela 1, tem-se a contribuição apresentada na Tabela 2.

Tabela 2 - Contribuição das Funções de Dados

Descrição da Função	Tipo de Função de Dados	Complexidade	Tamanho (PF)
AREA	AIE	Baixa	5
COLABORADOR	AIE	Baixa	5
PATRIMÔNIO	ALI	Baixa	7
TERCEIRIZADO	ALI	Baixa	7
HISTÓRICO	ALI	Baixa	7
INVENTARIO	ALI	Baixa	7
TRANSFERENCIA	ALI	Baixa	7

Além das funções de dados, foram identificadas para o sistema CBP as funções transacionais e suas respectivas informações quanto ao tipo de função, quantidade de dados e quantidade de arquivos referenciados. Para a contagem da quantidade de dados de cada função foram utilizados os protótipos de tela de cada funcionalidade do sistema CBP. Como exemplo, tem-se a transação de Inserir Bem Patrimonial e sua respectiva tela (protótipo) conforme ilustra a figura 11; para esta funcionalidade tem-se 16 tipos de dados.

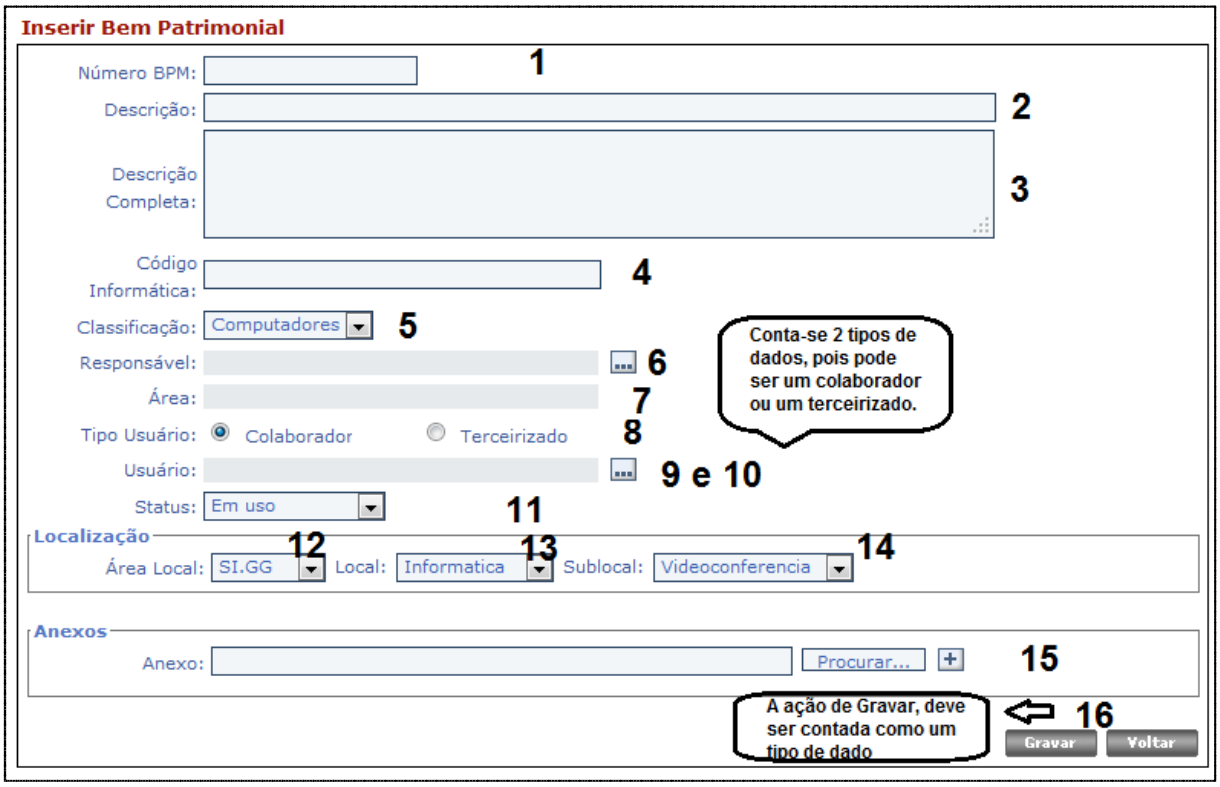

Figura 11 – Tela de Cadastro de Bem Patrimonial
Fonte: LOGIC TI (2011)

A tabela 3 apresenta as demais funções transacionais identificadas e suas respectivas informações necessárias para o processo de contagem.

Tabela 3 - Funções Transacionais do sistema CBP

Descrição da Função	Tipo de Função de Dados	Tipo de Dado	Tipo de Registro
Cadastrar Terceirizado (Tabela: Terceirizado)	EE	4	1
Excluir Terceirizado (Tabela: Terceirizado)	EE	3	1
Editar Terceirizado (Tabela: Terceirizado)	EE	3	1

Descrição da Função	Tipo de Função de Dados	Tipo de Dado	Tipo de Registro
Visualizar Detalhe Terceirizado (Tabela: Terceirizado)	CE	4	1
Listar Terceirizado (Tabela: Terceirizado)	CE	4	1
Cadastrar Patrimônio (Tabela: Área, Colaborador, Patrimônio, Anexo, Terceirizado)	EE	16	5
Excluir Patrimônio (Tabela: Patrimônio, Anexo)	EE	4	2
Editar Patrimônio (Tabela: Área, Colaborador, Patrimônio, Anexo, Terceirizado)	EE	15	5
Visualizar Detalhe Patrimônio (Tabela: Área, Colaborador, Patrimônio, Anexo, Terceirizado, Histórico)	CE	23	5
Listar Patrimônio (Tabela: Área, Colaborador, Patrimônio, Terceirizado)	CE	12	4
Relatório Patrimônio (Tabela: Área, Colaborador, Patrimônio, Terceirizado)	SE	10	4
Registrar Movimentação (Tabela: Área, Colaborador, Patrimônio, Histórico)	EE	19	4
Visualizar Detalhe Movimentação (Tabela: Área, Colaborador, Patrimônio, Histórico)	CE	18	4
Listar Movimentação (Tabela: Área, Colaborador, Patrimônio, Terceirizado)	CE	12	4
Registrar Transferência (Tabela: Área, Colaborador, Patrimônio, Histórico, Transferência, Anexo)	EE	20	5
Visualizar Detalhe Transferência (Tabela: Área, Colaborador, Patrimônio, Histórico, Transferência, Anexo)	CE	19	5
Listar Transferência (Tabela: Colaborador, Patrimônio, Transferência, Terceirizado)	CE	11	4
Relatório Transferência (Tabela: Área, Colaborador, Patrimônio, Histórico, Transferência, Anexo)	SE	19	5
Registrar Inventário (Tabela: Área, Colaborador, Patrimônio, Histórico, Inventário)	EE	10	5
Visualizar Detalhe Inventário (Tabela: Área, Colaborador, Patrimônio, Histórico, Inventário)	EE	9	5
Listar Inventário (Tabela: Colaborador, Patrimônio, Inventário, Terceirizado)	CE	9	4
Relatório Inventário (Tabela: Área, Colaborador, Patrimônio, Histórico, Inventário)	SE	9	5

Descrição da Função	Tipo de Função de Dados	Tipo de Dado	Tipo de Registro
Solicitar Inventário (Tabela:Área,Colaborador,Patrimônio, Inventário)	EE	8	4
Visualizar Detalhe Solicitação (Tabela: Área, Colaborador, Patrimônio, Histórico, Inventário)	CE	9	5
Listar Solicitações (Tabela: Área, Colaborador, Patrimônio, Inventário)	CE	14	4
Listar Colaboradores (RHI) (Tabela: Cadastro)	CE	2	1
Listar Áreas (TGI) (Tabela: Área)	CE	2	1

Para as funções transacionais identificadas para o projeto de Controle de Bens Patrimoniais, a complexidade e a contribuição destas funções segundo as regras da contagem descritas nas figuras 6, 7 e 8, são as apresentadas na Tabela 4.

Tabela 4 - Contribuição Funções de Transação

Descrição da Função	Tipo de Função de Dados	Complexidade	Tamanho (PF)
Cadastrar Terceirizado	EE	Baixa	3
Excluir Terceirizado	EE	Baixa	3
Editar Terceirizado	EE	Baixa	3
Visualizar Detalhe Terceirizado	CE	Baixa	3
Listar Terceirizado	CE	Baixa	3
Cadastrar Patrimônio	EE	Alta	6
Excluir Patrimônio	EE	Baixa	3
Editar Patrimônio	EE	Alta	6
Visualizar Detalhe Patrimônio	CE	Alta	6
Listar Patrimônio	CE	Alta	6
Relatório Patrimônio	SE	Alta	7
Registrar Movimentação	EE	Alta	6
Visualizar Detalhe Movimentação	CE	Alta	6
Visualizar Detalhe Transferência	CE	Alta	6
Listar Transferência	CE	Alta	6
Relatório Transferência	SE	Alta	7
Registrar Inventário	EE	Alta	6
Visualizar Detalhe Inventário	EE	Alta	6
Listar Inventário	CE	Alta	6
Relatório Inventário	SE	Alta	7
Solicitar Inventário	EE	Alta	6
Visualizar Detalhe Solicitação	CE	Alta	6
Listar Solicitações	CE	Alta	6
Listar Colaboradores (RHI)	CE	Baixa	3
Listar Áreas (TGI)	CE	Baixa	3

Totalizando o tamanho em Pontos de Função não ajustados das funcionalidades descritas na Tabela 2 e 4 (funções de dados e transacionais) através da fórmula: **DFP=(UFP+CFP),** obteve-se:

DFP = (186 + 0)

DFP = **186** pontos de função não ajustados

Obs: O projeto de Controle de Bens Patrimoniais não possui funções de conversão de dados.

Quadro 1 – Aplicação da fórmula de Contagem de PF não ajustado para o Projeto de Desenvolvimento CBP

Neste estudo aplicou-se também o fator de ajuste baseado nos seguintes níveis de influencia para cada uma das Características Gerais do Sistema(Tabela 5):

Tabela 5 - Influência das CGS no sistema CBP

Características Gerais do Sistema	Influência
01 - Comunicação de Dados	5
02 - Processamento Distribuído	5
03 - Performance	1
04 - Configuração Altamente Utilizada	1
05 - Volume de Transações	0
06 - Entrada de Dados On-line	5
07 - Eficiência do Usuário Final	4
08 - Atualização On-Line	4
09 – Complexidade de Processamento	1
10 - Reusabilidade	4
11 - Facilidade de Instalação	0
12 - Facilidade de Operação	4
13 - Múltiplos Locais	2
14 - Facilidade de Mudança	2

Fonte: LOGIC TI (2011)

O somatório das influências das CGS foi de 38 pontos. Assim sendo, o valor do fator de ajuste (VFA) para a Contagem final dos pontos de função encontrado foi:

VFA = (38 * 0,01) + 0,65

VFA = 1,03

Quadro 2 – Aplicação da Fórmula de Fator de Ajuste para o projeto CBP

Seguindo a última etapa do processo de contagem de pontos de função de um projeto, tem-se que calcular pontos de função ajustados; baseado nas

informações coletadas até o momento e seguindo a fórmula padrão de contagem de Pontos de Função ajustados de um projeto de desenvolvimento, obteve-se:

DFP = (186 + 0) x 1,03
DFP = 191,58 *pontos de função.*

Quadro 3 - Aplicação da fórmula de Contagem de PF Ajustado para o Projeto de Desenvolvimento CBP

Desta forma, segundo o processo de contagem de Pontos de Função apresentado neste trabalho e considerando o arredondamento (para mais após passar de 0,5 ponto) dos pontos de função, o projeto do sistema de Controle de Bens Patrimoniais foi estimado no tamanho de *Cento e Noventa e Dois (192) pontos de função*.

Esta estimativa de tamanho do sistema de Controle de Bens Patrimoniais realizada na fase inicial do Projeto é utilizada no contrato de serviço realizada entre as partes (Empresa desenvolvedora de sistemas Logic TI e sua cliente, empresa X), ou seja, a contratante X contrata o desenvolvimento de 192 pontos de função de um sistema (no caso o CBP) e em contrapartida a empresa Logic TI precisa desenvolver e entregar o sistema equivalente a 192 pontos de função. Além disso, com base nesta estimativa de tamanho a empresa Logic TI realiza o restante do procedimento de estimar *software* para assim acordar com a contratante o tempo de entrega e o custo deste projeto; bem como realiza o planejamento do processo de desenvolvimento deste sistema e utiliza estas informações como métricas/indicadores para projetos futuros.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÕES

Com base no presente estudo, conclui-se que a necessidade de gerenciar projetos e entregar o produto final com qualidade é atualmente o maior objetivo do processo de desenvolvimento de *software*. Para que isto seja possível torna-se obrigatório o melhor planejamento das atividades do processo de desenvolvimento de sistemas. Porém, só podemos planejar o que nos é conhecido, portanto a fase inicial (de concepção) dos projetos está ganhando força cada vez maior perante o mercado.

Este trabalho mostra que o planejamento eficiente consiste em estimar o tempo de desenvolvimento, o custo do projeto, bem como o esforço necessário para sua conclusão e isto somente é possível sabendo o quanto será preciso produzir, ou seja, conhecendo o tamanho do projeto baseado em uma unidade de medida.

Como o planejamento nasce junto com o início do projeto, é neste momento que o primeiro passo para o bom planejamento deve ser tomado, através da estimativa de tamanho do *software*. O real tamanho de um sistema somente é conhecido ao fim do projeto, porém através da estimativa é possível dimensionar o tamanho e realizar o planejamento e gerenciamento necessário.

Entre as técnicas de estimar tamanho de *software* vistas no presente trabalho percebe-se que a APF é a técnica mais madura e abrangente do mercado. Ela não apresenta restrições quanto à tecnologia, paradigma de desenvolvimento ou artefatos a serem utilizados, pois se baseia somente na visão do usuário e tem-se apresentado fortemente no mercado como meio de formalizar contratações, além de possuir órgãos mantenedores por diversos países do mundo, inclusive no Brasil através do BFPUG com forte atuação no mercado e por possuir certificação ISO que garante sua eficácia e maturidade.

O estudo de caso comprovou que o procedimento de contagem de pontos de função (unidade de medida do *software* da APF) é simples, prático e completo, bastando seguir as regras de contagem.

6.2 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Como sugestão de trabalho futuro, pode-se realizar um estudo aprofundado sobre a técnica de Contagem de Pontos por Casos de Uso para Projetos Orientados a Objetos, ou então realizar um comparativo entre a APF e UCP, as duas principais técnicas de estimativa de tamanho utilizadas hoje no mercado, visando identificar as vantagens de cada uma delas ou sua aplicabilidade para diferentes paradigmas de desenvolvimento de *software*.

REFERÊNCIAS

AGUIAR, M. **Pontos de Função ou Pontos por Caso de Uso? Como Estimar Projetos Orientados a Objetos**, 2003. Disponivel em: <http://www.bfpug.com.br/Artigos/UCP/Aguiar-Pontos_de_Funcao_ou_Pontos_por_Caso_de_Uso.pdf>. Acesso em: 31 Outubro 2011.

ANDRADE, E. L. P. D. **Pontos de Caso de Uso e Pontos de Função na Gestão de Estimativa de Tamanho de Projetos de Software Orientados a Objeto**, 2004. Disponivel em: <http://www.bfpug.com.br/>. Acesso em: 03 Março 2012.

ASMA. **O Padrão ISO de Medição Funcional de Tamanho**, 1999. Disponivel em: <http://www.bfpug.com.br/>. Acesso em: 06 Novembro 2011.

BARCELLOS, M. P. Medicao de Software - Um importante pilar da melhoria de processos de software. **Engenharia de Software Magazine**, n. 24, p. 64, 2010. ISSN 1983127-7.

BASSI, D. Estimativas Agéis com Planning Poker. **Engenharia de Software Magazine**, n. 9, p. 60, 2009. ISSN 1983127-7.

CALAZANS, A. T. S. **A utilização do COSMIC Full Function Points para estimativa de tamanho de software**, 2003. Disponivel em: <http://www.angelicatoffano.pro.br/upload_arquivos/pt/Artigo%20ASSE%202005_2.pdf>. Acesso em: 23 Outubro 2011.

CAMPOS, C. **Tipos de Contagem**, 2010. Disponivel em: <http://carloscamposinfo.com/cjec/?p=97>. Acesso em: 11 Janeiro 2012.

CAMPOS, C. **Tipo de Contagem**, 2010. Disponivel em: <http://carloscamposinfo.com/mundoapf/?p=163>. Acesso em: 23 Janeiro 2012b.

CAMPOS, C. **Escopo da Contagem**, 2010. Disponivel em: <http://carloscamposinfo.com/mundoapf/?p=192>. Acesso em: 02 Fevereiro 2012c.

CAMPOS, C. **Fronteira da Aplicação**, 2010. Disponivel em: <http://carloscamposinfo.com/mundoapf/?p=180>. Acesso em: 02 Fevereiro 2012d.

CAPPELLI, C. et al. **Pesquisa em Estimativas em Projetos de Modelagem de Processos**, 2009. Disponivel em: <http://www.seer.unirio.br/index.php/monografiasppgi/article/viewFile/504/717>. Acesso em: 23 Outubro 2011.

CARVALHO, V. A. D.; ARANTES, L. O.; FALBO, R. D. A. **EstimaODE:** Apoio a Estimativas de Tamanho e Esforço no Ambiente de Desenvolvimento de Software ODE, 2006. Disponivel em: <http://paa1.googlecode.com/svn-history/r106/trunk/seminario/EstimaODE.pdf>. Acesso em: 26 Agosto 2011.

CUNHA, D. **Mais Agilidade em suas estimativas com o Planning Poker**, 2009. Disponivel em: <http://www.brasiltech.net/agilez/2009/12/13/mais-agilidade-em-suas-estimativas-com-o-planning-poker/>. Acesso em: 24 Outubro 2011.

CUNHA, D. **Estimando pelo tamanho e não pela duração**, 2009. Disponivel em: <http://www.brasiltech.net/agilez/2009/09/22/estimando-pelo-tamanho-e-no-pela-durao/>. Acesso em: 24 Outubro 2011b.

DEKKERS, C. A. **Desmistificando Pontos de Função:** Entendendo a Terminologia, 1998. Disponivel em: <http://www.bfpug.com.br/Artigos/Desmistificando%20Pontos%20de%20Fun%C3%A7%C3%A3o.pdf>. Acesso em: 03 Novembro 2011.

FABIAN, C. M. **SICLA-APF Ferramenta de Análise de Pontos Por Função**, 2008. Disponivel em: <http://www.google.com.br/url?sa=t&rct=j&q=Ivan%2BMecenas%2BFPA&source=web&cd=7&ved=0CEwQFjAG&url=http%3A%2F%2Ftconline.feevale.br%2Ftc%2Ffiles%2F0001_1535.doc&ei=Fcq-Tp9LzO6CB4mblLkH&usg=AFQjCNELrjl8vuX968aLb5wUNVeeyLqhTw&sig2=-SgMZCYMyiIy9-kRUx1UFw&cad=r>. Acesso em: 12 Novembro 2011.

FATTO. www.fattocs.com.br. **Fatto Cs**, 2011. Disponivel em: <www.fattocs.com.br/download/fpa-m2-sw.ppt>. Acesso em: 25 Fevereiro 2011.

FREIRE, Y. M. A. **TUCP-M – Pontos de Casos de Uso Técnicos para Manutenção de Software**, 2008. Disponivel em: <https://uol01.unifor.br/oul/conteudosite/F1066345165/Dissertacao.pdf>. Acesso em: 15 Agosto 2011.

FREIRE, H. **Calculando Estimativas:** o Método de Pontos de Caso de Uso, 2003. Disponivel em: <http://pt.scribd.com/doc/4484908/Pontos-de-Caso-de-Uso>. Acesso em: 30 Outubro 2011b.

GERMANO, V. H. **Por que Projetos de software falham?**, 07 Outubro 2008. Disponivel em: <http://malditacomedia.blogspot.com/2008/10/por-que-projetos-de-software-falham.html>. Acesso em: 16 Agosto 2011.

GOMES, A. E. **Métricas e Estimativas de Software:** O ínicio de um rally de regularidade, 2003. Disponivel em: <http://www.linhadecodigo.com.br/artigo/102/M%C3%A9tricas-e-Estimativas-de-SoftwareO-in%C3%ADcio-de-um-rally-de-regularidade.aspx>. Acesso em: 15 Agosto 2011.

HAZAN, C. Analise de Pontos de Função - Uma aplicação nas estimativas de tamanho de Projetos de Software. **Engenharia de Software Magazine**, n. 02, p. 60, 2009. ISSN 1983127-7.

HAZAN, C. **Análise de Pontos de Função**, 2001. Disponivel em: <www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/APF.pdf>. Acesso em: 02 Fevereiro 2012b.

IFPUG. **International Fuction Point Users Group**, 2011. Disponivel em: <http://www.ifpug.org/>. Acesso em: 06 Novembro 2011.

LEITE, J. C. **Custos do Software**, 2006. Disponivel em: <http://www.dimap.ufrn.br/~jair/ES/slides/Custos.pdf>. Acesso em: 31 Agosto 2011.

LEITE, J. C. **Engenharia de Software**, 2009. Disponivel em: <http://engenhariadesoftware.blogspot.com/2007/04/mtricas.html>. Acesso em: 22 Outubro 2011b.

MACORATTI, J. C. **Estimativas de tamanho de software e APF**, 2005. Disponivel em: <http://www.macoratti.net/net_est1.htm>. Acesso em: 15 Agosto 2011.

MACORATTI, J. C. **Análise de Pontos por Função - O Processo de contagem**, 2005. Disponivel em: <http://www.macoratti.net/apf_pcta.htm>. Acesso em: 10 Fevereiro 2012b.

MARCIO, S. **Estimando Projetos de TI:** Arte ou Ciência, 2000. Disponivel em: <http://www.bfpug.com.br/Artigos/EstimandoProjetosDeTI.htm>. Acesso em: 27 Agosto 2011.

MONTEIRO, T. C. **Pontos de Caso de Uso Técnicos (TUCP):** Uma Extensão da UCP, 2005. Disponivel em: <www.cipedya.com/web/FileDownload.aspx?IDFile=156445>. Acesso em: 15 Agosto 2011.

MORIMOTO, C. E. Assembly. **www.hardware.com.br**, 2005. Disponivel em: <http://www.hardware.com.br/termos/assembly>. Acesso em: 26 Fevereiro 2012.

NESMA. **Análise de Pontos de Função para Melhoria de Software**, 2009. Disponivel em: <http://www.portaisgoverno.pe.gov.br/c/document_library/get_file?uuid=066903b6-39e9-44c4-833f-e7155a1c68c9&groupId=335215>. Acesso em: 06 Novembro 2011.

NUNES, P. **Conceito de Benchmarking**, 2008. Disponivel em: <http://www.knoow.net/cienceconempr/gestao/benchmarking.htm>. Acesso em: 26 Fevereiro 2012.

PARO, C. J. **Medidas de tamanho de desenvolvimento e de melhorias de software**, 2005. Disponivel em: <http://www.bfpug.com.br/Artigos/Medidas%20de%20tamanho%20de%20desenvolvimento%20e%20de%20melhorias%20de%20software.doc>. Acesso em: 30 Outubro 2011.

RIBEIRO, T. **Sequência de Fibonacci**, 2008. Disponivel em: <http://www.infoescola.com/matematica/sequencia-de-fibonacci/>. Acesso em: 26 Fevereiro 2012.

SANTANA, C.; GUSMÃO, C. Uso de Análise de Pontos de Função em Ambientes Ágeis. **Engenharia de Software Magazine**, n. 20, p. 60, 2010. ISSN 1983127-7.

SILVEIRA, M. **Estimando Projetos de TI: Arte ou Ciência**, 2000. Disponivel em: <http://www.bfpug.com.br/Artigos/EstimandoProjetosDeTI.htm>. Acesso em: 2012 Janeiro 26.

SIMÕES, C. **A Difícil Arte de Estimar Tempo para Implementação de Sistemas de informação**, 2004. Disponivel em: <www.bfpug.com.br/Artigos/Monografia%20-%20Métricas%20v3.doc>. Acesso em: 16 Agosto 2011.

SOUSA, D. J. D. **Uma abordagem sobre custo de software**, 2009. Disponivel em: <http://www.webartigos.com/artigos/uma-abordagem-sobre-custo-de-software/65926/>. Acesso em: 31 Outubro 2011.

STAA, A. V.; HAZAN, C. **Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software**, 2005. Disponivel em: <http://www.dbd.puc-rio.br/depto_informatica/05_04_hazan.pdf>. Acesso em: 20 Setembro 2011.

TORRES, M. B. **Análise de Pontos de Função: Estimativa Qualitativa x Estimativa Quantitativa**, 12 Fevereiro 2004. Disponivel em: <http://www.bfpug.com.br/Artigos/APF%20-%20Qualitativo%20x%20Quantitativo.pdf>. Acesso em: 16 Agosto 2011.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de Pontos de Função - Medição, Estimativas e Gerenciamento de Projetos de Software**. 9. ed. São Paulo: Érica, 2010.