UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS MATEMÁTICAS, DA NATUREZA E TECNOLOGIA DA INFORMAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA FERRAMENTA AUTOMATIZADA PARA SUPORTE À ANÁLISE DE PONTOS DE FUNÇÃO FORMANDO UMA *BASELINE* DE PROJETOS DE *SOFTWARE*

JULIO CESAR DE LEMOS

ORIENTADOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Matemáticas, da Natureza e Tecnologia da Informação, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA
2006

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS MATEMÁTICAS, DA NATUREZA E TECNOLOGIA DA INFORMAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA FERRAMENTA AUTOMATIZADA PARA SUPORTE À ANÁLISE DE PONTOS DE FUNÇÃO FORMANDO UMA *BASELINE* DE PROJETOS DE *SOFTWARE*

JULIO CESAR DE LEMOS

ORIENTADOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Matemáticas, da Natureza e Tecnologia da Informação, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA
2006

DESENVOLVIMENTO DE UMA FERRAMENTA AUTOMATIZADA PARA SUPORTE À ANÁLISE DE PONTOS DE FUNÇÃO FORMANDO UMA *BASELINE* DE PROJETOS DE *SOFTWARE*

AUTOR: JULIO CESAR DE LEMOS

ORIENTADOR: LUIZ EDUARDO GALVÃO MARTINS

Dissertação de Mestrado defendida em 08 de março de 2006, pela Banca Examinadora constituída dos Professores:

_____

Prof. Dr. Rafael Ferreira Alves

UNIMEP

_____

Prof. Dr. Ricardo Luís de Freitas

PUCCamp

_____

Prof. Dr. Luiz Eduardo Galvão Martins

UNIMEP

AGRADEÇO a Deus pelo amparo, *permitindo* o alcance de mais um degrau.

DEDICO à minha mãe Edy e ao meu pai Oscar que sempre me apoiaram.

OFEREÇO *à minha esposa Celma e ao meu filho Giovane, cujo apoio, carinho e companheirismo foram indispensáveis.*

AGRADECIMENTOS

Ao Prof. Dr. Luiz Eduardo Galvão Martins, pela orientação, ensinamentos, estímulo, exemplo e amizade dedicados nestes anos de convívio tão proveitosos. Agradeço as horas contínuas que dedicou, melhorando não apenas a qualidade deste trabalho, mas também a minha formação acadêmica e profissional. É impossível em espaço tão pequeno registrar a profunda gratidão que sinto.

Meu agradecimento à comissão julgadora, Drs. Luiz Eduardo Galvão Martins, Rafael Ferreira Alves e Ricardo Luís de Freitas. Obrigado por me conscientizar o quão pouco sabemos e que a prática da pesquisa científica envolve um aprendizado contínuo.

Finalmente, gostaria de prestar meus agradecimentos aos meus colegas de mestrado pela experiência obtida durante nossas conversas e discussões nos trabalhos. Agradeço também aos meus amigos e familiares no fornecimento de estímulo, desde o início até a conclusão do presente trabalho.

“Nada se torna real até ser experimentado,

mesmo um provérbio não significa nada para você

até sua vida poder ilustrá-lo.”

John Keats

(1795 - 1821)

RESUMO

Uma *baseline* de projetos de software fornece os itens necessários para predições de esforço, prazos e custos. Portanto, é um repositório de elementos básicos para um modelo efetivo de medição de software. A técnica de Análise de Pontos de Função (APF) foi proposta para ajudar a medir a funcionalidade de sistemas de software. É usada para medir o tamanho do software em termos funcionais. Porém, muito deste processo depende do julgamento do profissional que está medindo o software. Diferenças para o mesmo produto podem ocorrer na mesma empresa. Este trabalho tem como objetivo o desenvolvimento de uma ferramenta capaz de manter uma *baseline* de projetos de software particularizada por equipe de desenvolvimento. Devido à escolha da PF (Pontos de Função) como medida de tamanho de software, a ferramenta, além de manter a *baseline*, dará suporte à contagem dos Pontos de Função. Esta ferramenta terá como diferencial a captura de dados, através de interface amigável e ajuda ao usuário, minimizando diferenças no julgamento quando da contagem e o armazenamento dos dados dos projetos contados a fim de se obter uma *baseline*. A contribuição deste trabalho é proporcionar uma estrutura de armazenamento de informações sobre os dados dos projetos contados, formando uma *baseline* de projetos para uma equipe de desenvolvimento em particular. À medida que as informações forem armazenadas nesta estrutura, será possível trabalhar com esta base de informações, e realizar inferências estatísticas a fim de obter novos modelos de estimativas de desenvolvimento de projetos de software.

PALAVRAS-CHAVE: Análise de Pontos de Função, *baseline*, Projetos de software.

ABSTRACT

A baseline of software projects supplies the necessary items for predictions of effort, time and costs. Therefore, it is a repository of basic elements for an effective model of software measurement. The technique of Function Points Analysis (FPA) was proposed to help measure the functionality of software systems. It is used to measure the size of the software in functional terms. However, a lot of this process depends on the judgment of the professional who is measuring the software. Differences for the same product can happen in the same company. The goal of this work is to develop a tool capable of maintaining a baseline for software projects particularized by development team. Due to the choice of FP (Function Points) as measure of software size, the tool, besides maintaining the baseline, will give support to the counting of the Function Points. This tool will have as differential the capture of data, through a friendly interface and help to the user, minimizing differences in the judgment when counting and storing the data of the projects in order to obtain a baseline. The contribution of this work is to provide a structure of storage of information on the data of the counted projects, forming a baseline of projects for a development team. As the information is stored in this structure, it will be possible to work with this information base, and to accomplish statistical inferences in order to obtain new estimate models of development of software projects.

KEYWORDS: Function Points Analysis, *Baseline*, Software Projects.

ÍNDICE

FIGURAS

QUADROS

GLOSSÁRIO

AIE	**Arquivo de Interface Externa**
ALI	**Arquivo Lógico Interno**
APF	**Análise de Pontos de Função**
CE	**Consulta Externa**
CGS	**Características Gerais do Sistema**
DET	***Data Element Type* – Tipo de Dado**
EE	**Entrada Externa**
FA	**Fator de Ajuste**
FTR	***File Type Referenced* – Arquivo Referenciado**
IFPUG	**International Function Point Users Group – Grupo Internacional de Usuários de Pontos de Função**
NI	**Nível de Influência**
RET	***Record Element Type* – Tipo de Registro**
SE	**Saída Externa**

1. INTRODUÇÃO

Sistemas de Informação apoiados por Softwares são medidos por muitas razões: (a) indicar a qualidade do produto; (b) avaliar a produtividade das pessoas que produzem o produto; (c) avaliar os benefícios em termos de produtividade e qualidade derivados de novos métodos e ferramentas de software; (d) formar uma *baseline* para estimativas; (e) Ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional (Pressman, 1995). A Análise de Pontos de Função (APF) provê uma metodologia padronizada capaz de medir a funcionalidade do software sob o ponto de vista do usuário (ABRAN & ROBILLANRD, 1996) (IFPUG, 2000). Linhas de código (LOC) é uma métrica tradicional de tamanho de software. Diferentemente de Pontos de Função (PF), LOC mede o tamanho de um software em função do número de linhas de código ou milhares de linhas de código (KLOC). Existem vários problemas com o uso de LOC como unidade de medida de tamanho de software. O principal deles é a falta de um único padrão e da definição exata do que é linha de código. Jones (JONES, 2003) identificou onze principais variações de métodos de contagem de LOC.

Em se tratando de medição de Sistemas de Informação baseados em software ou projetos de software, há a necessidade de estabelecer o que comumente referenciamos como *baseline*. A *baseline* irá fornecer informações relevantes como nível de desempenho dos projetos baseados nos diversos níveis de produtividade e qualidade da organização (GARMUS & HERRON, 2001).

No contexto de Melhoria do Processo de Software, Capers Jones destaca o *estabelecimento* de uma *baseline* como marco inicial. Uma *baseline* provê uma base quantitativa de produtividade, cronograma, custos, qualidade e satisfação do usuário, necessária para futuros índices de melhoria (KAN, 2003).

Empresas interessadas em Melhoria do Processo de Software utilizam informações de sua *baseline* com propósitos de benchmark, a fim de fazer comparações de desempenho com companhias similares. Portanto, uma

baseline quantitativa é um pré-requisito para um efetivo Processo de Melhorias (KAN, 2003).

Quando se estabelece uma *baseline* de métricas, benefícios podem ser obtidos em nível estratégico, técnico e de projeto. A *baseline* consiste de dados compilados de projetos encerrados. Além das medidas orientadas à função (PF) ou tamanho (LOC), a *baseline* pode ser complementada com métricas de qualidade, como: defeitos, custo dos erros, erros de manutenção/manutenção total e esforço de manutenção/esforço de desenvolvimento, entre outras (PRESSMAN, 1995).

1.1. OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo apresentar e discutir aspectos de uma ferramenta de software que gerencia uma *baseline* particularizada por equipe de desenvolvimento. Reconhecendo a importância da Análise de Pontos de Função (APF) na identificação do tamanho funcional do projeto de software foi acrescida à ferramenta a metodologia de contagem e o armazenamento dos Pontos de Função.

1.2. JUSTIFICATIVA

O diferencial desta ferramenta em relação às demais existentes no mercado, é justamente a capacidade de gerenciamento de *baselines* particularizadas por organização, enquanto que as demais ferramentas gerenciam *baselines* genéricas, formadas por projetos coletados da indústria de software ao longo dos anos. A justificativa de se obter uma precisão maior nas estimativas face a uma *baseline* particularizada é apresentada ao longo do trabalho.

1.3. METODOLOGIA DE TRABALHO

A metodologia utilizada para a elaboração do trabalho é apresentada em quatro etapas, descritas a seguir.

a) Revisão bibliográfica e capacitação técnica: Verificou-se na literatura especializada, os principais tópicos referentes a *baseline* de projetos de software e Análise de Pontos de Função (APF) a fim de embasar e sedimentar o conhecimento. Uma capacitação técnica em UML (Unified Modeling Languange), OO (Orientação a Objetos) e JAVA, fez-se necessário para a elaboração da ferramenta.

b) Análise e modelagem: Utilizou-se de algumas atividades da Engenharia de Requisitos (ER) para levantamento dos requisitos fundamentais na fase de elicitação. Para a modelagem dos requisitos levantados, utilizou-se o diagrama de casos de uso da *UML.* A estrutura da *baseline* foi representada através do diagrama de classes e a lógica envolvida na contagem dos Pontos de Função foi expressa através dos diagramas de atividades.

c) Desenvolvimento da ferramenta: Os diagramas de classes foram implementados na linguagem JAVA acessando o SGBD (Sistema Gerenciador de Banco de Dados) MySQL. A escolha da linguagem de programação e do banco de dados, deveu-se ao fato de serem softwares livres.

d) Validação da ferramenta: Foram realizados vários testes unitários a fim de minimizar possíveis erros de manipulação (inclusão, alteração e exclusão) de dados. Na validação final, uma empresa da região de Campinas/SP disponibilizou a documentação e a contagem final dos Pontos de Função de um sistema já encerrado. Estes dados foram digitados na ferramenta e como resultado verificamos que foi possível armazenar na *baseline* vários itens da documentação entregue e a ferramenta realizou corretamente a contagem dos

Pontos de Função, comparando-se com o total dos Pontos de Função apresentado pela Empresa.

1.4. CONTRIBUIÇÃO DO TRABALHO

Este trabalho colabora no sentido de propor uma ferramenta capaz de formar e manter uma *baseline* de projetos de software particular por organização, além de oferecer uma funcionalidade extra, a contagem e o armazenamento de Pontos de Função, conforme normas estabelecidas pelo IFPUG (International Function Points Users Group) (IFPUG, 2000). É importante salientarmos sua relevância e contribuição face ao gerenciamento de uma *baseline* particularizada. Segundo a maioria dos autores pesquisados, uma *baseline* com projetos da própria organização tende a oferecer modelos de estimativas mais precisos por refletir dados da própria organização. Esta ferramenta não é inédita, existem várias no mercado, porém cabe salientar sua natureza livre. Cremos que a maior contribuição virá com a evolução da ferramenta – o Módulo de Estimativas. Neste novo módulo, pretende-se inferir nos dados históricos da organização, armazenados na *baseline*, e propor modelos de estimativas de desenvolvimento de software. Estes modelos poderão estimar o prazo, o custo e o esforço de desenvolvimento de software dado um número *n* de Pontos de Função. A base para tornar isto possível foi concretizada com a realização deste trabalho.

2. *BASELINE* DE PROJETOS DE SOFTWARE

Em se tratando de medição de sistemas de software ou projetos de software, há a necessidade de estabelecer o que comumente referenciamos como *baseline*. A *baseline* irá fornecer informações relevantes como nível de desempenho dos projetos baseados nos diversos níveis de produtividade e qualidade da organização (GARMUS & HERRON, 2001).

No contexto de Melhoria do Processo de Software, Capers Jones destaca o estabelecimento de uma *baseline* como marco inicial. Uma *baseline* provê uma base quantitativa de produtividade, cronograma, custos, qualidade e satisfação do usuário, necessária para futuros índices de melhoria (KAN,2003).

Empresas interessadas em Melhoria do Processo de Software utilizam informações de sua *baseline* com propósitos de *benchmark*, a fim de fazer comparações de desempenho com companhias similares. Portanto, uma *baseline* quantitativa é um pré-requisito para um efetivo Processo de Melhorias (KAN,2003).

Segundo a IEEE (IEEE Std. No. 610.12-12-1990), uma *baseline* é uma especificação ou produto que foi formalmente revisado e acordado, e que depois disso serve como base para futuros desenvolvimentos, e pode ser alterada somente através de um processo formal de controle de alterações.

No contexto da Engenharia de Software, uma *baseline* é um marco no desenvolvimento do sofware que é feito na entrega de um ou mais Itens de Configuração do Software (*SCIs – Software Configuration Items*) e a aprovação destes SCIs, que é obtida através de uma revisão técnica formal. Por exemplo, os elementos de uma especificação de projeto devem ser armazenados na *baseline*. Quando erros são encontrados em alguma especificação, todas as partes da especificação devem ser revisadas, corrigidas e acordadas, somente depois elas retornam para a *baseline* (PRESSMAN, 2001).

Pressman expressa sua preocupação em alterações realizadas no software que não são atualizadas na *baseline*. Toda melhoria realizada nos projetos de software deverá ser atualizada formalmente, desde as fases iniciais de especificação do projeto até uma manutenção corretiva. Para que a *baseline* seja a base para futuros desenvolvimentos, é necessário que ela reflita exatamente a realidade do desenvolvimento do software.

Quando se estabelece uma *baseline* de métricas, benefícios podem ser obtidos em nível estratégico, técnico e de projeto. A *baseline* consiste de dados compilados de projetos encerrados. Além das medidas orientadas à função (PF) ou tamanho (LOC), a *baseline* pode ser complementada com métricas de qualidade como defeitos, custo dos erros, erros de manutenção/manutenção total e esforço de manutenção/esforço de desenvolvimento, entre outras (PRESSMAN, 1995).

Armazenar informações referentes a projetos de software em uma *baseline* é uma prática comum em organizações que conduzem programas de métricas. Isto envolve a coleta e análise dos dados sobre numerosos projetos ou sistemas de software (GARMUS & HERRON, 1996). Uma *baseline* pode ser de dois tipos: (1) de sistemas ou aplicações (resumida) e (2) de sistemas finalizados (estendida).

a) *Baseline* de Sistemas ou Aplicações (resumida) é formada por dados coletados apenas com informações sobre métricas (tamanho funcional em PF, tamanho em LOCs, etc.), geralmente utilizada apenas para predição de esforço.

b) *Baseline* de Sistemas Finalizados (estendida) é formada principalmente por projetos desenvolvidos recentemente e projetos de melhoria. As informações neste tipo de *baseline* são mais detalhadas e poderão conter o tipo de entrega, tecnologia utilizada, plataforma de desenvolvimento e tipo de aplicação.

Uma *baseline* de projetos de software fornece os itens necessários para predições de esforço, prazos e custos. Portanto, é um repositório de elementos

básicos para um modelo efetivo de medição de software (GARMUS & HERRON, 2001).

2.1. ELEMENTOS BÁSICOS PARA MEDIÇÃO DE SOFTWARE

Os elementos básicos para um efetivo modelo de medição de software podem ser expressos em dois conceitos: quantitativo e qualitativo. Ambos se referem ao programa de métricas. Dados quantitativos incluem fatores como o tamanho funcional do software entregue, o esforço requerido para o desenvolvimento, o tempo requerido para produzir o software e o custo do projeto.

Níveis de produtividade de desempenho são calculados com base nesses dados quantitativos. Expressões comuns de níveis de desempenho incluem índices de produtividade e qualidade relativas ao tamanho. Quando o tamanho do projeto é medido em pontos de função, pode-se expressar a produtividade como resultado do número de pontos de função produzidos por pessoas-mês ou o número de horas por pontos de função. Esta medida de produtividade torna-se uma das muitas medidas que permite criar perfis de desempenho organizacional e comparar o desempenho em relação aos padrões da indústria (GARMUS & HERRON, 2001).

A comparação de desempenho de uma organização em relação a outras organizações ou padrões da indústria só é possível através de dados coletados de uma *baseline* própria (formada por dados que refletem a realidade daquela organização). A Figura 1 ilustra um modelo de medição de software proposto por (GARMUS & HERRON, 2001).

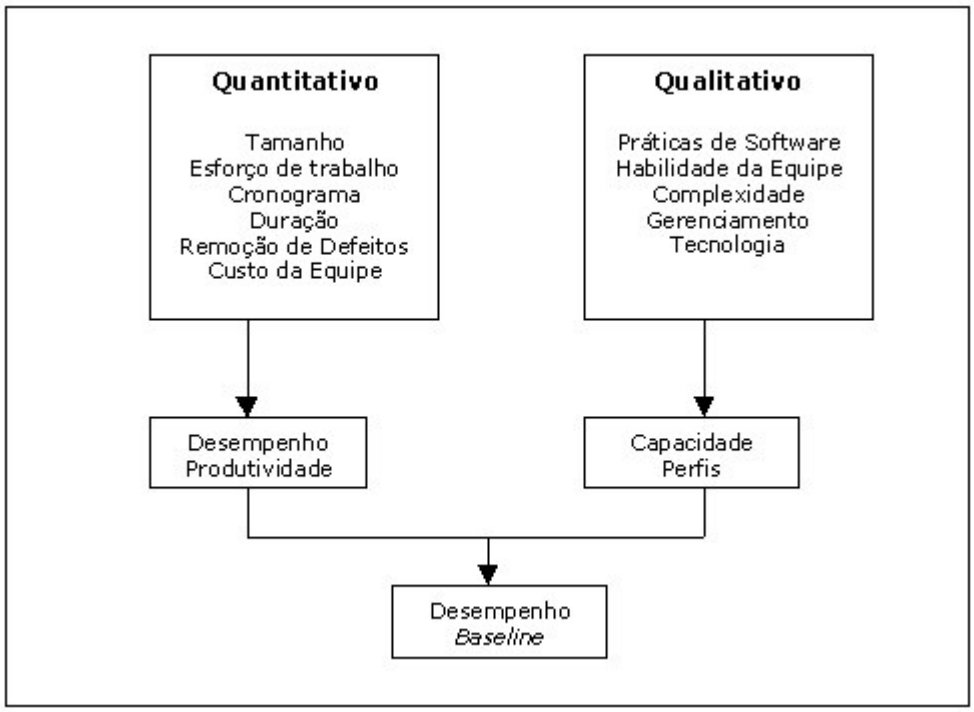


FIGURA 1– MODELO DE MEDIÇÃO DE SOFTWARE DE GARMUS & HERRON (2001)

O modelo apresentado na Figura 1 ilustra a formação de uma *baseline* formada por elementos quantitativos e qualitativos. Com as informações quantitativas é possível extrair níveis de produtividade e desempenho, como por exemplo, o número de PF produzidos por pessoa-mês (medida de produtividade). As informações qualitativas são expressas em termos das características utilizadas no processo de desenvolvimento de software e são referenciadas como fatores influenciadores. Portanto, conclui-se que as características qualitativas influenciam os níveis de desempenho e produtividade no desenvolvimento de software (GARMUS & HERRON, 2001).

Destaca-se a importância dos elementos básicos para medição de software também como variáveis utilizadas em modelos de análise de dados da Engenharia de Software (ES). Para a correta construção de modelos de estimativas, há de se considerar características comuns no desenvolvimento de software como linhas de código, experiência da equipe de desenvolvimento, esforço despendido, entre outras (BRIAND E OUTROS, 1992).

A abordagem típica utilizada para estimativas de desenvolvimento de software, impreterivelmente é baseada na intuição, ou no melhor caso, em médias de dados históricos. Porém a experiência isolada da equipe, ou de um indivíduo da equipe pode trazer relevantes distorções (HUMPHREY & SINGPURWALLA, 1991). Em seu trabalho os autores descrevem a importância de registrar a experiência da equipe de desenvolvimento para futuros estudos empíricos.

2.2. MODELOS DE ESTIMATIVAS ENCONTRADOS NA LITERATURA

ALBRECHT & GAFFNEY (1983) publicaram um estudo realizado com dados coletados de 24 aplicações desenvolvidas pelo Serviço de Processamento de Dados da IBM. Nestes dados coletados, foi realizada a contagem de pontos de função em cada um dos 24 projetos, constituindo desta forma uma *baseline*. Através desta *baseline* foi possível extrair, por projeto, a relação Pontos de Função *versus* Horas de Trabalho, a qual foi descrita como esforço *E*. Utilizando análise de regressão, o estudo foi capaz de predizer o esforço (em horas de trabalho) em função do volume de pontos de função de uma aplicação, ou seja, determinou-se a linha ajustada da variável dependente *E* (esforço) expressa em função da variável independente *PF* (pontos de função). A figura 2 ilustra o modelo de regressão obtido a partir dos dados coletados dos 24 projetos.

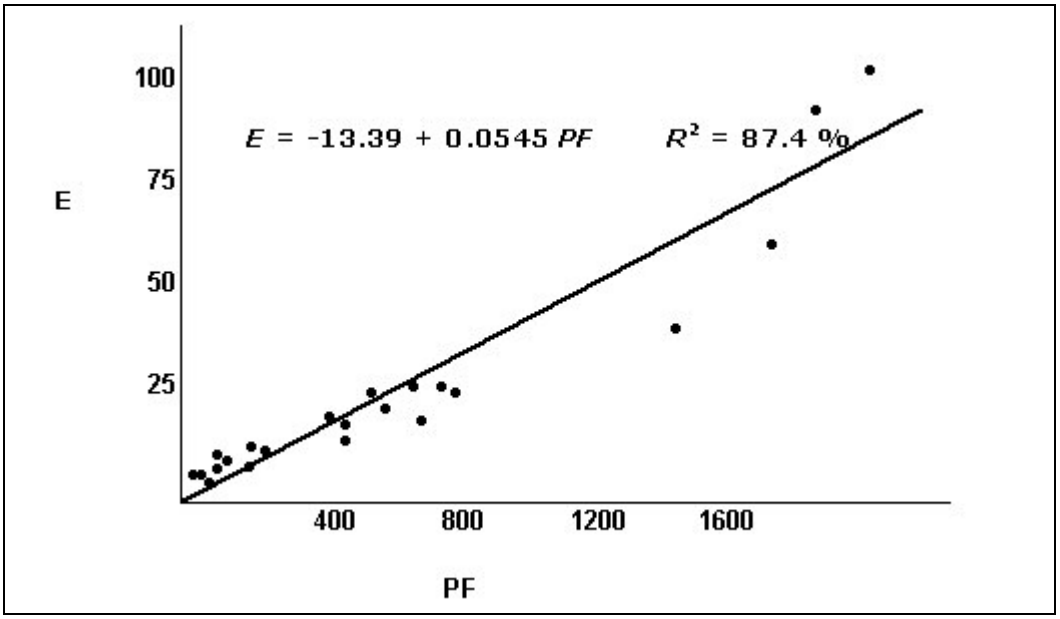


FIGURA 2 – MODELO DE REGRESSÃO DO ESFORÇO DE DESENVOLVIMENTO (E) VERSUS PONTOS DE FUNÇÃO (PF) USANDO DADOS DA IBM (O ESFORÇO É MEDIDO EM MILHARES DE HORAS DE TRABALHO) (MATSON E OUTROS, 1994).

Outro trabalho sobre modelo de estimativas de custo foi realizado por KEMERER (1987), onde, através de uma validação empírica, foram avaliados quatro modelos de estimativa de custo de software (SLIM, COCOMO, Pontos de Função e ESTIMACTS). Segundo seu trabalho um dos pontos fracos de ferramentas que utilizam LOC (linhas de código), como COCOMO e SLIM, é que para estimativas de esforço nas etapas iniciais do ciclo de vida dos projetos, é difícil mensurar o tamanho em LOC, necessitando para isto, lançar mão de tabelas de transformação de Pontos de Função em LOC.

Outra desvantagem na utilização do tamanho em LOC é a falta de padronização da contagem. Foi realizada uma comparação entre seis modelos de estimativas de tamanho de software gerados por diferentes equipes especializadas em contagem de linhas de código. Constatou-se resultados que variavam de 6622 a 36700 LOCs quando na realidade o software possuía 9177 LOCs (VERNER & TATE, 1992).

Pontos de função, por ser independente da linguagem de programação, possibilitam a geração de modelos de estimativas de desenvolvimento e custo

já nas fases iniciais do ciclo de desenvolvimento. Gerentes podem utilizar estimativas de custo para aprovar ou rejeitar um projeto proposto ou gerenciar o processo de desenvolvimento (STRIKE & OUTROS , 2001).

Em seu estudo sobre o relacionamento entre tamanho de software versus esforço de desenvolvimento, FENTON & OUTROS (2004) utiliza pontos de função como métrica de tamanho de software. A utilização de um modelo causal com redes bayesiana ajuda gerentes de projetos a analisar o impacto no esforço de desenvolvimento de projetos de software, sob dois cenários: a) Diminuir a funcionalidade (pontos de função) sem perder a qualidade; b) Manter a funcionalidade, perdendo qualidade.

Uma das conclusões, segundo Kemerer, é a duvidosa precisão de um modelo de estimativa fora do seu ambiente original. Alguns dos modelos desenvolvidos em diferentes ambientes não trabalham bem como o esperado, necessitando de calibrações que colocam em dúvida sua precisão.

MATSON E OUTROS (1994) seguindo a linha de Kemerer, sugerem que uma *baseline* da própria organização refletiria um modelo mais ajustado que modelos compostos por dados coletados de outras organizações, mesmo sendo uma organização qualitativamente similar.

3. ANÁLISE DE PONTOS DE FUNÇÃO – VISÃO GLOBAL

3.1. HISTÓRICO

Com o passar dos anos várias melhorias foram implementadas desde sua publicação em 1979 (Quadro 1). As primeiras três (Albrecht 79, Albrecht 83 e GUIDE 85) se propuseram a estruturar esta técnica incluindo pesos e critérios de seleção, enquanto que as cinco versões da IFPUG (International Function Points Users Group) padronizaram regras e diretrizes para a identificação das funções e características gerais do sistema (IFPUG, 2000) (2000).

QUADRO 1 – LISTA DAS VERSÕES OFICIAIS DE PONTOS DE FUNÇÃO

1	Albrecht 79
2	Albrecht 83
3	GUIDE 85
4	IFPUG V.1 86
5	IFPUG V.2 88
6	IFPUG V.3 90
7	IFPUG V.4 94
8	IFPUG V.4.1 99

IFPUG

No início de 1986, um grupo de 12 pessoas se reuniu em Toronto para discutir experiências sobre o uso do APF; nascia o IFPUG (International Function Points Users Group – Grupo Internacional de Usuários de Pontos de Função).

Com a disseminação do uso da técnica, começaram a surgir outras aplicações. Verificou-se que a técnica de APF poderia ser usada para medir a produtividade de desenvolvimento, a qualidade de sistemas através do acompanhamento de índices de erros por Pontos de Função, etc. A possibilidade de usar o APF com benefícios gerenciais fez com que rapidamente aumentasse o número de usuários.

Hoje em dia o IFPUG é o órgão responsável pela padronização da técnica de contagem de Pontos de Função (BRAGA , 1996).

3.2. CONTAGEM DE PONTOS DE FUNÇÃO

3.2.1. VISÃO GLOBAL

Pontos de Função é uma métrica de tamanho de software que utiliza a funcionalidade lógica em termos do negócio (*Business*), sob a ótica do usuário (segundo A.J. Albrecht). Diferentemente de outras métricas, como por exemplo, modelos que utilizam LOC (Linhas de Código) como critério de medida de tamanho. Neste caso, o tamanho tem relação com a linguagem de programação (ABRAN & ROBILLANRD, 1993).

Muito tem se falado em Análise de Pontos de Função (APF), mas o que realmente significa analisar a funcionalidade de um software? Para sanar esta dúvida, é importante conceituar **Tamanho Funcional**.

Tamanho funcional é uma medida de tamanho de software, baseada em uma avaliação padronizada dos requisitos lógicos dos usuários.

Na indústria há atualmente várias maneiras de se medir tamanho funcional, a mais sedimentada são os Pontos de Função. Semelhante aos metros quadrados de uma casa, Pontos de Função são independentes dos métodos físicos, ferramentas ou linguagem de desenvolvimento utilizados para construir o software.

Um dos desafios na implementação da análise de Pontos de Função é tornar o método compreensível para todos 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), Pontos de Função forçam os desenvolvedores a pensarem em termos lógicos (DEKKERS, 1998).

A seguir será apresentada a técnica de contagem de Pontos de Função e seus conceitos. A base deste trabalho é o Manual de Práticas de Contagem do International Function Points Users Group (IFPUG), atualmente na versão 4.1.1.

A técnica divide a funcionalidade do software a ser contado em duas principais funções:

1. Funções de dados e
2. Funções de transações.

As funções de dados podem ser de dois tipos:

- Arquivos Lógicos Internos (ALI).
- Arquivos de Interface Externa (AIE).

As funções de transações são divididas em três tipos diferentes:

- Entradas Externas (EE).
- Saídas Externas (SE).
- Consultas Externas (CE).

Para um melhor entendimento sobre os conceitos utilizados na contagem de Pontos de Função, segue uma breve explicação sobre os cinco tipos de funções.

3.2.2. ARQUIVO LÓGICO INTERNO (ALI)

Um ALI é uma entidade lógica persistente, a respeito da qual dados serão mantidos. Os ALIs baseiam-se em requisitos lógicos dos usuários e são independentes da implementação ou meios de armazenamento, tais como tabelas ou banco de dados. Um ALI é contado com base em uma avaliação do número de campos de dados não recursivos do usuário e do número de tipos de elementos de registros lógicos nele contidos. Um ALI reside inteiramente dentro da fronteira da aplicação e é mantido através de Entradas Externas (EE) (IFPUG, 2000).

3.2.3. ARQUIVO DE INTERFACE EXTERNA (AIE)

Sob o ponto de vista do IFPUG (2000), um AIE é um grupo de dados ou informações de controle identificável pelo usuário, referenciado pela aplicação, mas mantida dentro das fronteiras de outra aplicação.

3.2.4. ENTRADA EXTERNA (EE)

Segundo a definição do IFPUG (2000) Entrada Externa é um processo elementar no qual dados atravessam a fronteira de fora para dentro. Tais dados podem vir de uma tela de entrada de dados, por via eletrônica ou através de um outro aplicativo. Os dados podem ser informações de controle ou informações do negócio. No caso dos dados serem informações do negócio, serão utilizados para manter um ou mais arquivos lógicos internos. Se os dados forem informações de controle, não será necessário que atualizem um arquivo lógico interno. Um exemplo de EE seria “Incluir empregado” em um aplicativo de recursos humanos.

3.2.5. SAÍDA EXTERNA (SE)

Segundo a definição do IFPUG (2000), uma Saída Externa (SE) é um processo elementar que gera dados ou informações de controle, enviados para fora da fronteira do aplicativo.

Portanto, SE é um processo elementar no qual dados derivados passam através da fronteira de dentro para fora. Os dados criam relatórios ou arquivos de saída, que são enviados a outros aplicativos. Esses relatórios e arquivos são criados a partir de um ou mais arquivos lógicos internos e/ou arquivos de interface externa.

3.2.6. CONSULTA EXTERNA (CE)

Segundo a definição do IFPUG (2000), uma Consulta Externa (CE) é um processo elementar constituído por uma combinação entrada-saída que resulta na recuperação de dados. O lado de saída não contém dados derivados. Nenhum arquivo lógico interno é mantido no processamento. A diferença entre uma consulta externa e uma saída externa é que na consulta externa não existem dados derivados, enquanto que na saída externa, esta condição é obrigatória.

O Processo de Contagem

O processo de cálculo do tamanho de um sistema envolve:

1. Identificar os Arquivos Lógicos Internos (ALI), Arquivos de Interface Externa (AIE), Entradas Externas (EE), Saídas Externas (SE) e Consultas Externas (CE).

2. Classificar cada função quanto à complexidade funcional relativa como: simples, média ou complexa.

3. Calcular os Pontos de Função Brutos através da aplicação dos pesos de acordo com a tabela específica.

4. Efetuar avaliação das 14 Características Gerais do Sistema calculando o fator de ajuste.

5. Calcular os Pontos de Função ajustados.

O diagrama a seguir ilustra os passos para se efetuar a contagem de Pontos de Função.

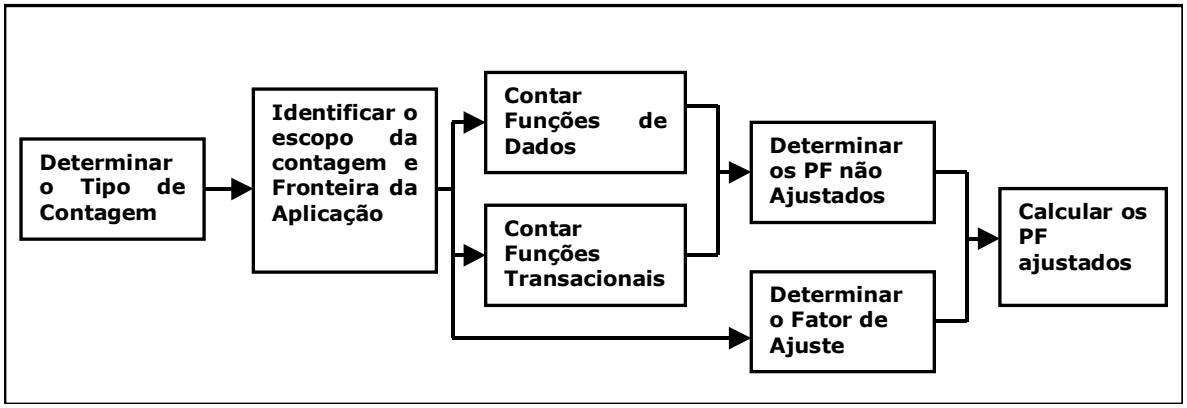


FIGURA 3 – ETAPAS PARA CONTAGEM DE PONTOS DE FUNÇÃO

As próximas seções abordarão cada uma das etapas mostradas na figura 3.

3.2.7. DETERMINAR O TIPO DE CONTAGEM

O primeiro passo no processo de contagem de Pontos de Função é identificar o tipo de contagem desejada, como mostrada a seguir:

- Contagem de projetos de desenvolvimento
- Contagem de projetos de manutenção
- Contagem de uma aplicação

Projeto de desenvolvimento

A contagem de Pontos de Função de projetos de desenvolvimento prevê a contabilização das funções identificadas no modelo lógico do sistema, permitindo uma estimativa dos recursos de tempo e pessoal, necessários ao desenvolvimento do sistema. Normalmente são feitas várias medições durante o desenvolvimento do projeto, inclusive na implantação do mesmo.

Projeto de manutenção

A aplicação de Pontos de Função em projetos de manutenção de sistemas consiste na medição das modificações que envolvem a inclusão, alteração ou

exclusão de funções. Após a instalação da funcionalidade resultante da manutenção, os cálculos dos Pontos de Função da aplicação devem ser refeitos para refletir as alterações efetuadas no sistema.

Contagem de uma aplicação

Trata-se da aplicação da técnica de Pontos de Função em um sistema já totalmente desenvolvido; é também referenciado como Pontos de Função instalados. Esta contagem provê a medida atual dos pontos de funções fornecidos ao usuário pela aplicação. Este número é calculado após o desenvolvimento do aplicativo e deve ser atualizado após qualquer manutenção que altere sua funcionalidade (BRAGA, 1996).

3.2.8. IDENTIFICAR O ESCOPO DA APLICAÇÃO E FRONTEIRA DA APLICAÇÃO

A fronteira de contagem separa o projeto ou aplicação que está sendo contado de aplicações externas, ou seja, outros sistemas da organização.

As fronteiras são utilizadas para estabelecer:

- O escopo do produto que está sendo medido.
- A propriedade do dado, requerida pela contagem, identificando se o dado pertence à aplicação que está sendo contada ou a outra aplicação.
- A propriedade das funções, identificando se a função pertence à aplicação que está sendo contada.

Regras

As seguintes regras devem ser aplicadas para o estabelecimento das fronteiras:

- A fronteira é determinada através do ponto de vista do usuário. O foco está no que pode ser compreendido e descrito pelo usuário.

- A fronteira entre aplicações relacionadas é baseada em funções empresariais separadas como vista pelos usuários e não por necessidades tecnológicas.

- Para projetos de manutenção, a fronteira inicial deve estar em conformidade com a fronteira já estabelecida para a aplicação ou aplicações que estão sendo modificadas.

3.2.9. CONTAR FUNÇÕES DE DADOS

Funções de dados representam a funcionalidade provida ao usuário através de dados internos ou externos à aplicação. Funções de dados são definidas como Arquivos Lógicos Internos (ALI) ou Arquivos de Interface Externa (AIE).

O termo arquivo, neste contexto, não tem o mesmo significado que o utilizado tradicionalmente em processamento de dados. Neste caso, arquivo, refere-se a um grupo de dados logicamente relacionados e não à implementação física deste grupo de dados (IFPUG, 2000).

Um ALI equivale, em um diagrama de fluxo de dados, a um depósito de dados, e em um modelo entidade relacionamento, a uma entidade (BRAGA, 1996). É importante que alguns termos utilizados no processo de contagem de Pontos de Função sejam identificados e definidos. Nas próximas linhas serão abordados alguns destes termos.

Informação de controle

É o dado usado pela aplicação para garantir a aderência da aplicação aos requisitos das funções do negócio estabelecidas pelo usuário (BRAGA, 1996).

É o dado que influencia um processo elementar da aplicação que está sendo contada. Ele especifica o que, quando, ou como o dado está sendo processado (IFPUG, 2000).

Identificado pelo usuário

Termo utilizado nas definições de ALI e AIE, se refere aos requisitos definidos para processos e/ou grupos de dados que são concordados e entendidos por ambos, usuário(s) e desenvolvedor(es) de software (IFPUG, 2000).

Dados derivados

Dados derivados são dados cujo processamento vai além da recuperação e edição direta de informações de arquivos lógicos internos ou arquivos de interface externa. São o resultado de algoritmos e/ou cálculos. Dados derivados ocorrem quando um ou mais elementos são combinados com uma fórmula, de modo a gerar ou derivar um ou mais elementos de dados adicionais.

Algoritmo

Um algoritmo é definido como um procedimento mecânico para executar um dado cálculo ou resolver um problema utilizando uma série de passos.

Mantido

Termo utilizado nas definições de ALI e AIE, se refere à habilidade de incluir, alterar e excluir dados através de processos elementares da aplicação (IFPUG, 2000).

Processo elementar

É a menor atividade que é significativa para os negócios do usuário. O processo elementar deve ser auto-suficiente, deixando a aplicação que está sendo contada em um estado consistente (BRAGA, 1996).

Um processo elementar é a menor unidade de atividade que é significativa para o(s) usuário(s). O processo elementar tem que ser autocontido e deixa o negócio da aplicação que está sendo contada em um estado consistente (IFPUG, 2000).

Fatores de complexidade

O número de ALI e AIE e suas complexidades relativas determinam a contribuição das funções do tipo de dados para a contagem de Pontos de Função Brutos (IFPUG, 2000).

Dado Elementar Referenciado

Segundo o IFPUG (2000), Dado Elementar Referenciado ou em inglês DET (Data Element Type), é um campo único e não repetido que é reconhecido pelo usuário. Será utilizada a sigla em inglês DET comumente utilizada na comunidade científica.

Registro Lógico Referenciado

Segundo o IFPUG (2000), Registro Lógico Referenciado ou em inglês RET (Record Element Type), é um subgrupo de elementos de dados, reconhecido pelo usuário, dentro de um ALI ou AIE . Existem dois tipos de subgrupos:

- Opcional
- Mandatório

Subgrupos opcionais são aqueles em que o usuário tem a opção de usar um ou nenhum dos subgrupos durante o processo elementar que adiciona ou cria uma instância do dado.

Subgrupos mandatórios são aqueles onde o usuário tem que usar pelo menos um subgrupo.

Será utilizada a sigla em inglês, RET comumente utilizada na comunidade científica.

Arquivo Lógico Interno (ALI)

É um grupo de dados logicamente relacionados, ou informações de controle, identificado pelo usuário e mantido dentro das fronteiras da aplicação. O intento

principal de um ALI é assegurar que dados serão mantidos por um ou mais processos elementares da aplicação que está sendo contada (IFPUG, 2000). Devemos identificar os ALIs, definir a complexidade de cada um e calcular a contribuição em termos de Pontos de Função.

Regras de identificação de ALI:

O IFPUG, através do IFPUG (2000), rege as seguintes duas regras que deverão ser aplicadas (ambas) para a identificação de ALIs.

- O grupo de dados ou informações de controle é lógico e identificável pelo usuário.
- O grupo de dados é mantido através de um processo elementar dentro das fronteiras da aplicação que está sendo contada.

São exemplos de ALI:

- Cadastro de clientes
- Cadastro de produtos
- Arquivo de dependentes
- Cadastro de funcionários
- Arquivo de controle de acesso à aplicação

Não são exemplos de ALI:

- Arquivos temporários
- Arquivos de trabalho
- Arquivos de classificação

- Arquivos incluídos por motivos de tecnologia
- Arquivos de índices alternativos

Complexidade de ALI

A complexidade de um ALI é calculada a partir da quantidade de Registros Lógicos Referenciados (RET) e da quantidade de Dados Elementares Referenciados (DET).

Regras para contagem de DET

Segundo o IFPUG (2000), as seguintes regras devem ser aplicadas na contagem de DETs:

- Considere um DET para cada campo único não repetido, reconhecido pelo usuário, mantido em ou recuperado de um ALI através de um processo elementar .
- Considere um DET para cada item de dado em um ALI que exista para atender a requisitos solicitados pelo usuário, envolvendo relacionamento com outro ALI.
- Considere as seguintes implementações técnicas como um DET, para um conjunto complexo de campos:
    - Campos que aparecem mais de uma vez em um ALI por causa de tecnologia ou técnica de implementação.
    - Campos repetitivos que são idênticos em formato e existem para permitir múltiplas ocorrências de um mesmo dado.

Regras para contagem de RET

Segundo o IFPUG (2000), as seguintes regras devem ser aplicadas na contagem de RET:

- Conte um RET para cada subgrupo de dados de um ALI independentemente de ser o subgrupo opcional ou mandatório.
- Caso não haja subgrupo de informações, conte um RET para cada ALI.

Definição de complexidade

O quadro a seguir demonstra a definição da complexidade de cada ALI.

QUADRO 2 – DEFINIÇÃO DA COMPLEXIDADE PARA ALI

	1 a 19 DET	**20 a 50 DET**	**51 ou mais DET**
1 RET	Simples	Simples	Média
2 a 5 RET	Simples	Média	Complexa
6 ou mais RET	Média	Complexa	Complexa

Contribuição em Pontos de Função Brutos

O quadro a seguir demonstra o número de Pontos de Função correspondentes ao tipo de ALI identificado.

QUADRO 3 – PONTOS DE FUNÇÃO BRUTOS VERSUS GRAU DE COMPLEXIDADE FUNCIONAL

Grau de Complexidade Funcional	**Pontos de Função Brutos**
Simples	7
Média	10
Complexa	15

Exemplo de contagem de ALIs

O quadro a seguir demonstra o cálculo para:

- 7 ALIs com grau simples
- 4 ALIs com grau médio
- 2 ALIs com grau complexo

QUADRO 4 – CÁLCULO DA COMPLEXIDADE

Tipo de Função	Complexidade funcional	Complexidade total
ALI	7 Simples X 7	49
	4 Média X 10	40
	2 Complexas X 15	30
	Total Pontos de Função Brutos	**119**

Arquivo de Interface Externa (AIE)

Um AIE é um grupo de dados logicamente relacionados ou informações de controle identificável pelo usuário e necessários à aplicação, mas mantidos fora das fronteiras da aplicação que está sendo contada. Isto significa que um AIE contado em uma aplicação deve estar classificado como ALI em outra aplicação (IFPUG, 2000).

Regras de identificação de AIE

Segundo IFPUG (2000), todas as regras seguintes devem ser aplicadas para identificação de um AIE:

- O grupo de dados ou informação de controle é um grupo de dados lógicos e identificados pelo usuário.
- O grupo de dados é referenciado e externo à aplicação que está sendo contada.
- O grupo de dados não é mantido (não sofre manutenção) pela aplicação que está sendo contada.
- O grupo de dados é contado como um ALI por outra aplicação.

Complexidade de AIE

A complexidade é calculada em função da quantidade de Registros Lógicos Referenciados (RET) e da quantidade de Dados Elementares Referenciados (DET). Devemos usar as regras de contagem para contar o número de RET e DET.

Regras para contagem de DET

Segundo o IFPUG (2000), as seguintes regras devem ser aplicadas na contagem de DET:

- Considere um DET para cada campo não repetido, reconhecido pelo usuário, mantido ou recuperado, presente em cada AIE.
- Considere um DET para cada item de dado em um AIE que exista para atender a requisitos solicitados pelo usuário, envolvendo relacionamento com outro ALI.
- Considere as seguintes implementações técnicas como um DET, para um conjunto completo de campos:
    - Campos que aparecem mais de uma vez por causa da tecnologia ou técnica de implementação.
    - Campos repetitivos que são idênticos em formato e existem para permitir múltiplas ocorrências de um mesmo dado.

Regras para contagem de RET

Segundo o IFPUG (2000) as seguintes regras devem ser aplicadas na contagem de RET:

- Conte um RET para cada subgrupo de dados de um AIE independentemente de ser o subgrupo opcional ou mandatório.
- Caso não haja subgrupo de informações, conte um RET para cada AIE.

Definição de complexidade

O quadro a seguir demonstra a definição da complexidade de cada AIE.

QUADRO 5 – DEFINIÇÃO DA COMPLEXIDADE PARA AIE

	1 a 19 DET	**20 a 50 DET**	**51 ou mais DET**
1 RET	Simples	Simples	Média
2 a 5 RET	Simples	Média	Complexa
6 ou mais RET	Média	Complexa	Complexa

Contribuição em Pontos de Função Brutos

O quadro a seguir demonstra o número de Pontos de Função correspondentes ao tipo de AIE identificado.

QUADRO 6 – PONTOS DE FUNÇÃO VERSUS COMPLEXIDADE FUNCIONAL

Grau de Complexidade Funcional	**Pontos de Função Brutos**
Simples	5
Média	7
Complexa	10

Exemplo de contagem de AIEs

O quadro a seguir demonstra o cálculo para:

- 5 AIEs com grau simples
- 4 AIEs com grau médio
- 3 AIEs com grau complexo

QUADRO 7 – CÁLCULO DA COMPLEXIDADE TOTAL

Tipo de Função	**Complexidade funcional**	**Complexidade total**
AIE	5 Simples X 5	25
	4 Média X 7	28
	3 Complexas X 10	30
	Total Pontos de Função Brutos	**83**

3.2.10. CONTAR FUNÇÕES TRANSACIONAIS

Funções do tipo transação representa a funcionalidade provida ao usuário pelo processamento de dados em uma aplicação (IFPUG, 2000). Funções do tipo transações são definidas como:

- Entradas Externas (EE)
- Saídas Externas (SE)
- Consultas Externas (CE)

ENTRADA EXTERNA (EE)

Uma Entrada Externa (EE) processa dados ou informações de controle que vêm de fora das fronteiras da aplicação que está sendo contada. A EE é por si só um processo elementar. Através de processo lógico específico os dados mantêm um ou mais ALIs, as informações de controle processadas podem ou não manter um ALI (IFPUG, 2000).

As EEs representam o fluxo de informações de fora para dentro da aplicação. Cada EE deve ser analisada segundo a quantidade de FTR e a quantidade de DETs. Com estas informações é calculada a complexidade de cada EE (Simples, Média ou Complexa) e conseqüentemente o número de Pontos de Função.

Normalmente operações de inclusão de registros em arquivos, alterações de registros e exclusão de registros são analisadas como Entradas Externas.

Regras de identificação de EE

Segundo o IFPUG, através de seu manual IFPUG (2000), todas as regras a seguir tem que ser aplicada em um processo elementar que está sendo contado como ocorrência única em uma Entrada Externa:

- Os dados ou informações de controle são recebidos de fora da fronteira da aplicação.

- Pelo menos um Arquivo Lógico Interno (ALI) é mantido se o dado entrar na fronteira da aplicação .

- Para o processo identificado, um dos seguintes três enunciados tem que ser aplicado:

    - Processamento lógico é único para o processo lógico realizado por outra Entrada Externa para a aplicação.

    - O conjunto de dados elementares identificado é diferente do conjunto de dados identificado por outra Entrada Externa para a aplicação.

    - Os Arquivos Lógicos Internos e Arquivos de Interface Externa referenciados são diferentes dos arquivos referenciados por outra Entrada Externa na aplicação.

Definição de regras de complexidade

Deve-se determinar a complexidade funcional de cada Entrada Externa (EE) baseando-se no número de Arquivos Lógicos Referenciados (FTR) e no número de Dados Elementares Referenciados (DET). Cada EE deve ter seu grau de complexidade identificado.

Definição de Arquivos Lógicos Referenciados

Segundo o IFPUG (2000), um Arquivo Lógico Referenciado, ou em inglês FTR (File Type Referenced) é:

- Um Arquivo Lógico Interno lido ou mantido por um tipo de função ou

- Um Arquivo de Interface Externa lido ou mantido por um tipo de função transacional.

Será utilizada a sigla em inglês FTR comumente utilizada na comunidade científica.

Regras para contagem de FTR

- Conte um FTR para cada ALI mantido.
- Conte um FTR para cada ALI ou AIE lido durante o processamento da EE.
- Conte apenas um FTR para cada ALI que é lido e mantido por uma EE.
- Conte um FTR caso haja acesso a arquivo de mensagens de erro.

Regras para contagem de DET

- Conte um DET para cada campo não recursivo, identificado pelo usuário, mantido em um ALI por uma EE.
- Conte um DET para cada campo pertencente a um ALI que não é digitado pelo usuário, mas é mantido por uma EE.
- Conte as seguintes técnicas de implementação físicas como um único DET para o grupo de campos:
    - Um campo Lógico que é armazenado fisicamente em múltiplos campos, mas é requerido pelo usuário como peça única de informação. (Por exemplo, campo data, desmembrado em DIA, MÊS e ANO).
    - Campos que aparecem mais de uma vez em um ALI por necessidade da técnica ou tecnologia de implementação devem ser contados apenas uma única vez.
    - Campos que indicam condição de erro durante o processamento ou confirmação de que o processo está completo.
    - Conte um único DET para linhas de comando ou teclas de função que provejam a ação a ser tomada pela EE.
    - Em tela de atualização só conte os campos que possam sofrer atualização.

- Em telas de exclusão conte somente os campos-chave.

- Um DET deve ser contado quando uma ou mais mensagens de erro informarem ao usuário, através de um campo ou área na tela, que uma EE não pode ser processada por erro de edição, erro de validação ou se ainda houver uma mensagem de confirmação.

Definição de complexidade

O quadro a seguir demonstra a definição da complexidade de cada EE.

QUADRO 8 – DEFINIÇÃO DA COMPLEXIDADE PARA EE

	1 a 4 DET	**5 a 15 DET**	**16 ou mais DET**
0 ou 1 FTR	Simples	Simples	Média
2 FTR	Simples	Média	Complexa
3 ou mais FTR	Média	Complexa	Complexa

Contribuição em Pontos de Função Brutos

O quadro a seguir demonstra o número de Pontos de Função correspondentes a cada tipo de função EE identificada.

QUADRO 9 – PONTOS DE FUNÇÃO BRUTOS POR GRAU DE COMPLEXIDADE FUNCIONAL

Grau de Complexidade Funcional	**Pontos de Função Brutos**
Simples	3
Média	4
Complexa	6

Exemplo de contagem de EEs

O quadro a seguir demonstra o cálculo para:

- 3 EEs com grau simples

- 2 EEs com grau médio

- 2 EEs com grau complexo

QUADRO 10 – CÁLCULO FINAL DE PONTOS DE FUNÇÃO BRUTOS PARA EE

Tipo de Função	Complexidade funcional	Complexidade total
EE	3 Simples X 3	9
	2 Média X 4	8
	2 Complexas X 6	12
	Total Pontos de Função Brutos	**29**

SAÍDA EXTERNA (SE)

Segundo o IFPUG (2000), uma Saída Externa (SE) é um processo elementar que gera dados ou informações de controle para fora das fronteiras da aplicação. A intenção primária de uma Saída Externa é a de apresentar informações ao usuário através de um processamento lógico que recupere dados ou informações de controle. O processamento lógico deverá conter pelo menos uma fórmula matemática ou cálculo, criando dados derivados, mantendo um ou mais Arquivos Lógicos Internos ou alterar o comportamento do sistema.

Cada SE deve ser analisada segundo a quantidade de Arquivos Lógicos Referenciados (FTR) e a quantidade de Dados Elementares Referenciados (DET). Com estas informações é calculada a complexidade de cada SE, que pode ser simples, média ou complexa.

Regras de identificação de SE

De acordo com o IFPUG (2000), todas as seguintes regras têm que ser aplicadas para o processo elementar, que está sendo contado como uma ocorrência única em uma Saída Externa:

- A função envia dados ou informações de controle externos à fronteira da aplicação.

- Para o processo identificado, um dos seguintes três enunciados tem que ser aplicado:

- O processamento lógico é único para o processo lógico realizado por outra Saída Externa para a aplicação.

- O conjunto de dados elementares identificado é diferente do conjunto de dados identificado por outra Saída Externa para a aplicação.

- Os Arquivos Lógicos Internos e Arquivos de Interface Externa referenciados são diferentes dos arquivos referenciados por outra Saída Externa na aplicação.

Segundo BRAGA (1996), para identificar as SE é necessário procurar dados ou informações de controle que estão sendo enviados para fora das fronteiras da aplicação. As regras são:

- Regras de contagem de dados

- Regras de processamento de informações de controle

Regras de contagem de dados

Todas as seguintes regras de contagem devem ser aplicadas às funções para serem consideradas como SE.

- O processo envia dados, ou informações de controle, para fora da fronteira da aplicação.

- O dado, ou informação de controle é enviado através de um processo elementar da aplicação.

- O processo é a menor unidade de atividade com significado para o usuário final no negócio.

- O processo é autocontido e deixa o negócio que a aplicação informatiza em estado consistente.

- Com relação ao processo identificado, uma das seguintes opções deve ser correta:

- A lógica de processamento é diferente de lógicas processadas em outras Saídas Externas da aplicação.

- Os dados elementares identificados são diferentes dos dados de outras Saídas Externas da aplicação.

Definição e regras de complexidade

Deve-se determinar a complexidade funcional relativa de cada SE baseando-se no número de Arquivos Lógicos Referenciados (FTR) e no número de Dados Elementares Referenciados (DET).

Regras, segundo IFPUG (2000) para contagem de FTR

- Conte um FTR para cada ALI ou AIE lido durante o processo elementar.

- Conte um FTR para cada ALI mantido durante o processamento do processo elementar.

- Conte somente um FTR para cada ALI mantido e lido durante o processo elementar.

Regras, segundo IFPUG (2000), para contagem de DET

- Conte um DET para cada campo não recursivo, reconhecido pelo usuário, que entra na fronteira da aplicação e é requerido para especificar quando, o quê e/ou como o dado está sendo recuperado ou gerado por um processo elementar.

- Conte um DET para cada campo não recursivo, reconhecido pelo usuário, que exista na fronteira.

Exemplos práticos de contagem, segundo BRAGA (1996)

- Conte um DET para cada sumário ou campo de total que apareça em uma SE.

- Não conte literais como DET. Exemplo: Cabeçalho de campos, nome de relatório.

- Não conte número de páginas ou campos automáticos do sistema.

- Conte as seguintes implementações físicas como um único DET:

    - Um campo Lógico que é armazenado como múltiplos campos, mas é requerido pelo usuário como uma única informação.

    - Cada tipo de “label” e cada tipo equivalente de número (valor) em um gráfico de saída.

    - Informação de texto que poderia ser uma única palavra, sentença ou frase.

Definição de complexidade

O quadro a seguir demonstra a definição da complexidade de cada SE.

QUADRO 11 – DEFINIÇÃO DE COMPLEXIDADE PARA SE

	1 a 5 DET	**6 a 19 DET**	**20 ou mais DET**
0 ou 1 FTR	Simples	Simples	Média
2 a 3 FTR	Simples	Média	Complexa
4 ou mais FTR	Média	Complexa	Complexa

Contribuição em Pontos de Função Brutos

O quadro a seguir demonstra o número de Pontos de Função correspondentes a cada tipo de função SE identificada.

QUADRO 12 – PONTOS DE FUNÇÃO BRUTOS VERSUS COMPLEXIDADE FUNCIONAL

Grau de Complexidade Funcional	**Pontos de Função Brutos**
Simples	4
Média	5
Complexa	7

Exemplo de contagem de SEs

O quadro a seguir demonstra o cálculo para:

- 3 SEs com grau simples
- 2 SEs com grau médio
- 2 SEs com grau complexo

QUADRO 13 – CONTAGEM TOTAL DE PONTOS DE FUNÇÃO BRUTOS PARA SE

Tipo de Função	Complexidade funcional	Complexidade total
SE	3 Simples X 4	12
	2 Média X 5	10
	2 Complexas X 7	14
	Total Pontos de Função Brutos	**36**

CONSULTA EXTERNA (CE)

Segundo o IFPUG (2000), uma Consulta Externa (SE) é um processo elementar que gera dados ou informações de controle para fora das fronteiras da aplicação. A intenção primária de uma Consulta Externa é a de apresentar informações ao usuário através de um processamento lógico que recupere dados ou informações de controle de um Arquivo Lógico Interno ou de um Arquivo de Interface Externa. O processamento lógico não contém fórmula matemática ou cálculo, nenhum dado é derivado. Nenhum Arquivo Lógico Interno é mantido durante o processamento, assim como o comportamento do sistema não é alterado.

Regras de identificação de CE

De acordo com o IFPUG (2000), todas as seguintes regras têm que ser aplicadas para o processo elementar que está sendo contado como uma ocorrência única em uma Consulta Externa:

- A função envia dados ou informações de controle externos à fronteira da aplicação.
- Para o processo identificado, um dos seguintes três enunciados tem que ser aplicado:
    - Processamento lógico é único para o processo lógico realizado por outra Consulta Externa para a aplicação.
    - O conjunto de dados elementares identificados é diferente do conjunto de dados identificado por outra Consulta Externa para a aplicação.
    - Os Arquivos Lógicos Internos e Arquivos de Interface Externa referenciados são diferentes dos arquivos referenciados por outra Consulta Externa na aplicação.

Segundo BRAGA (1996), as seguintes regras devem ser aplicadas às informações para serem consideradas como Consultas Externas (CE).

- Uma solicitação de consulta que “entre” na aplicação.
- Resultados de saída disponibilizados para fora das fronteiras da aplicação.
- Dados não recuperados (consultados).
- Os dados recuperados (mostrados) não contém dados derivados.
- A solicitação de consulta e os resultados fornecidos compõem um processo que é a menor unidade de atividade com significado para o negócio do usuário.
- O processo elementar é autocontido e mantém a aplicação, que está sendo contada, em um estado consistente.
- O processo não atualiza um Arquivo Lógico Interno (ALI).
- Para o processo identificado, uma das seguintes regras deve ser aplicada:

- A lógica do processo na parte de entrada ou na parte de saída é única quando comparada com outras lógicas de Consultas Externas da mesma aplicação.

- Os elementos de dados que compõem a parte de entrada ou a parte de saída são diferentes de outras consultas externas da mesma aplicação.

Definição e regras de complexidade

A determinação da complexidade funcional de cada CE é feita baseando-se no número de Arquivos Lógicos Referenciados (FTR) e no número de Dados Elementares Referenciados (DET) para a parte de entrada e para a parte de saída.

Use a maior das duas complexidades funcionais (entrada e saída) da consulta para externar a complexidade final da consulta e conseqüentemente o número referente de Pontos de Função.

REGRAS, segundo o IFPUG (2000), para contagem de FTR

Regra para a parte de entrada

- Conte a quantidade de FTRs na lógica de processamento da Consulta Externa.

Regra para a parte de saída

- Conte a quantidade de FTRs na lógica de processamento da Consulta Externa.

REGRAS, segundo o IFPUG (2000), para contagem de DET

Regra para a parte de entrada

- Conte um DET para cada campo não recursivo que aparece na parte de entrada de uma Consulta Externa.

- Conte um DET para cada campo que especifica o critério de seleção de dados.

- Conte as seguintes técnicas de implementação física como um único DET para todo o grupo de campos:

    - Campos utilizados para indicar que o processamento foi realizado com sucesso.

    - Campos que permitam a capacidade de especificar que a consulta externa deve ser executada.

      Exemplo: Conte como um DET o botão OK que deve ser apertado pelo usuário para efetivar a consulta.

- Conte um DET quando uma ou mais mensagens de erro informarem ao usuário que uma consulta não foi efetivada por erro de edição ou validação, ou se ainda uma mensagem de confirmação ocorrer.

Regras para a parte de saída

- Conte um DET cada campo não recursivo, identificado pelo usuário que aparece na parte de saída da consulta.

- Não conte literais como DETs (nome do relatório e cabeçalho de colunas, são exemplos).

- Não conte variáveis de paginação ou variáveis automáticas do sistema, tais como números de página.

- Conte as seguintes técnicas de implementação física como um único DET, para o inteiro conjunto de campos:

    - Um campo lógico que é armazenado fisicamente como múltiplos campos, mas é requerido pelo usuário como uma única informação.

- Campos, que por causa da tecnologia empregada no desenvolvimento do sistema, aparecem mais de uma vez em um Arquivo Lógico Interno (ALI).

Definição de complexidade

O quadro a seguir demonstra a definição da complexidade de cada CE para a parte de ENTRADA.

QUADRO 14 – DEFINIÇÃO DA COMPLEXIDADE PARA CE – ENTRADA

	1 a 4 DET	**5 a 15 DET**	**16 ou mais DET**
0 ou 1 FTR	Simples	Simples	Média
2 FTR	Simples	Média	Complexa
3 ou mais FTR	Média	Complexa	Complexa

O quadro a seguir demonstra a definição da complexidade de cada CE para a parte de SAÍDA.

QUADRO 15 – DEFINIÇÃO DA COMPLEXIDADE PARA CE – SAÍDA

	1 a 5 DET	**6 a 19 DET**	**20 ou mais DET**
0 ou 1 FTR	Simples	Simples	Média
2 a 3 FTR	Simples	Média	Complexa
4 ou mais FTR	Média	Complexa	Complexa

Contribuição em Pontos de Função Brutos

O quadro a seguir demonstra o número de Pontos de Função correspondentes a cada tipo de função CE identificada. Utilizar sempre a maior complexidade encontrada entre as partes de entrada e saída.

QUADRO 16 – PONTOS DE FUNÇÃO BRUTOS VERSUS COMPLEXIDADE

Grau de Complexidade Funcional	**Pontos de Função Brutos**
Simples	3
Média	4
Complexa	6

Exemplo de contagem de SEs

O quadro a seguir demonstra o cálculo para:

- 3 CEs com grau simples
- 2 CEs com grau médio
- 2 CEs com grau complexo

QUADRO 17 – CONTAGEM TOTAL DE PONTOS DE FUNÇÃO BRUTOS PARA CE

Tipo de Função	Complexidade funcional	Complexidade total
CE	3 Simples X 3	9
CE	2 Média X 4	8
CE	2 Complexas X 6	12
	Total Pontos de Função Brutos	**29**

3.2.11. DETERMINAR OS PONTOS DE FUNÇÃO NÃO AJUSTADOS (BRUTOS)

De posse da definição de cada função e sua complexidade relativa, devemos calcular os Pontos de Função Brutos, através do seguinte quadro:

QUADRO 18 – CÁLCULO DE PONTOS DE FUNÇÃO BRUTO POR TIPO DE FUNÇÃO

Tipo Função	Complexidade Funcional	Total por Complexidade	Total por Tipo
ALIs	Simples Média Complexa	X 7 = X 10 = X 15 =	
AIEs	Simples Média Complexa	X 5 = X 7 = X 10 =	
EEs	Simples Média Complexa	X 3 = X 4 = X 6 =	
SEs	Simples Média Complexa	X 4 = X 5 = X 7 =	
CEs	Simples Média Complexa	X 3 = X 4 = X 6 =	

Exemplo de cálculo de Pontos de Função Brutos

Considerando um sistema que possua:

QUADRO 19 – RESUMO DE COMPLEXIDADE POR TIPO DE FUNÇÃO

Arquivos Lógicos Internos	10 simples, 5 médio, 2 complexo
Arquivos de Interface Externa	3 simples, 2 médio, 0 complexo
Entradas Externas	13 simples, 7 média, 5 complexa
Saídas Externas	8 simples, 5 média, 3 complexa
Consultas Externas	9 simples, 5 média, 4 complexa

Atualizando o quadro mostrado a seguir, teríamos:

QUADRO 20 – QUADRO RESUMO TOTAL DE PONTOS DE FUNÇÃO BRUTOS

Tipo Função	**Complexidade Funcional**	**Complexidade Funcional**		**Total por Tipo**
ALIs	10	Simples	X 7 =	70
	5	Média	X 10 =	50
	2	Complexa	X 15 =	30
				150
AIEs	3	Simples	X 5 =	15
	2	Média	X 7 =	14
		Complexa	X 10 =	0
				29
EEs	13	Simples	X 3 =	39
	7	Média	X 4 =	28
	5	Complexa	X 6 =	30
				97
SEs	8	Simples	X 4 =	32
	5	Média	X 5 =	25
	3	Complexa	X 7 =	21
				78
CEs	9	Simples	X 3 =	27
	5	Média	X 4 =	20
	4	Complexa	X 6 =	24
				71
Total de Pontos de Função Brutos:				**425**

Neste exemplo o sistema possui 425 Pontos de Função Brutos.

3.2.12. DETERMINAR O FATOR DE AJUSTE

O cálculo do fator de ajuste é baseado em quatorze Características Gerais do Sistema (CGS). Cada característica está associada à descrição que auxilia na determinação do nível de influência (NI) de cada uma. Os níveis de influência variam de zero a cinco, respectivamente *nenhuma influência* até *influência forte (IFPUG, 2000)*.

As Características Gerais do Sistema podem influir no seu tamanho de –35% até +35%. O fator de ajuste é aplicado sobre os Pontos de Função Brutos para permitir o cálculo dos Pontos de Função ajustados. Os seja, o valor do fator de ajuste pode variar de 0,65 até 1,35 (IFPUG, 2000).

Processo de cálculo do fator de ajuste

Os seguintes passos devem ser executados para o cálculo do fator de ajuste:

1. Avaliar o impacto de cada uma das 14 CGSs no aplicativo que está sendo contado, atribuindo peso de 0 a 5 para cada característica.
2. Calcular o nível de influência através da soma dos pesos de cada uma das 14 características.
3. Calcular o fator de ajuste através da equação:

Fator-Ajuste = (NI*0,01)+0,65

Exemplo:

A soma dos pesos das 14 CGSs resultou em 61 (nível de influência=61), o fator de ajuste será:

Fator-Ajuste = (61*0,01)+0,65

Fator-Ajuste = 1,26

Características Gerais do Sistema (CGS)

As 14 CGSs são relacionadas a seguir:

1. Comunicação de Dados
2. Funções Distribuídas
3. Performance
4. Configuração do equipamento
5. Volume de transações
6. Entrada de dados on-line
7. Interface com o usuário
8. Atualização on-line
9. Processamento complexo
10. Reusabilidade
11. Facilidade de implementação
12. Facilidade operacional
13. Múltiplos locais
14. Facilidade de mudanças (flexibilidade)

Níveis de Influência

Cada Característica Geral do Sistema deve ser avaliada em termos de sua influência, usando uma escala de zero (0) a cinco (5), como mostrado no quadro a seguir:

QUADRO 21 – GRAUS DE INFLUÊNCIA

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

A seguir será mostrado um quadro com as quatorze características e seus respectivos níveis de influência, variando de 0 (zero) a 5 (cinco), e também uma breve explicação para cada nível de influência por característica.

QUADRO 22 – DETERMINAÇÃO DO NÍVEL DE INFLUÊNCIA

Grau	**Descrição**
	1. Comunicação de Dados
0	Aplicação puramente batch ou funciona em um micro stand-alone
1	Aplicação é batch, mas utiliza entrada de dados remota ou impressão remota.
2	Aplicação é batch, mas utiliza entrada de dados remota e impressão remota.
3	Aplicação inclui entrada de dados on-line e um processamento batch ou de um sistema de consulta.
4	Aplicação é mais do que uma entrada on-line, mas suporta apenas um tipo de protocolo de comunicação.
5	Aplicação é mais do que uma entrada on-line e suporta mais de um tipo de protocolo de comunicação.
	2. Funções Distribuídas
0	Aplicação não auxilia na transferência de dados ou funções entre os processadores da empresa.
1	Aplicação prepara dados para o usuário final utilizar em outro processador, tal como planilhas em PC

2	Aplicação prepara dados para a transferência, é transferido e então processado em outro equipamento da empresa (não pelo usuário final).
3	Processamento distribuído e a transferência de dados são *on-line* e apenas em uma direção.
4	Processamento distribuído e a transferência de dados são *on-line* e apenas em ambas as direções.
5	As funções de processamento são dinamicamente executadas no equipamento mais adequado.
3. Performance	
0	Nenhum requisito especial de performance foi solicitado pelo usuário.
1	Requisitos de performance e de desenho foram estabelecidos e revistos, mas nenhuma ação especial foi requerida.
2	Tempo de resposta e volume de processamento são itens críticos durante horários de pico de processamento. Nenhuma determinação especial para a utilização do processador foi estabelecida. A data limite para a disponibilidade de processamento é sempre o próximo dia útil.
3	Tempo de resposta e volume de processamento são itens críticos durante todo o horário comercial. Nenhuma determinação especial para a utilização do processador foi estabelecida. A data limite necessária para a comunicação com outros sistemas é um item importante.
4	Os requisitos de performance estabelecidos requerem tarefas de análise de performance na fase de análise e desenho da aplicação.
5	Além do descrito no item anterior, ferramentas de análise de performance foram usadas nas fases de desenho, desenvolvimento e/ou implementação para atingir os requisitos de performance estabelecidos pelo usuário.
4. Configuração do Equipamento	
0	Nenhuma restrição operacional explícita ou mesmo implícita foi incluída.
1	Existem restrições operacionais leves. Não é necessário esforço especial para resolver as restrições.
2	Algumas considerações de ajuste de performance e segurança são necessárias.
3	São necessárias especificações especiais de processador para um módulo específico da aplicação.

4	Restrições operacionais requerem cuidados especiais no processador central ou no processador dedicado.
5	Além das características do item anterior, há considerações especiais na distribuição do sistema e em seus componentes.
	5. Volume de Transações
0	Não estão previstos períodos de picos de volume de transação.
1	Estão previstos picos de transações mensalmente, trimestralmente, anualmente ou em certo período do ano.
2	São previstos picos semanais.
3	São previstos picos diários.
4	Altos volumes de transações foi estabelecido pelo usuário, ou o tempo de resposta necessário atinge nível alto o suficiente para requerer análise de performance na fase de desenho.
5	Além do descrito no item anterior, é necessário utilizar ferramentas de análise de performance nas fases de desenho, desenvolvimento e/ou implantação.
	6. Entrada de Dados on-line
0	Todas as transações são processadas em modo batch.
1	De 1% a 7% das transações são entradas de dados on-line.
2	De 8% a 15% das transações são entradas de dados on-line.
3	De 16% a 23% das transações são entradas de dados on-line.
4	De 24% a 30% das transações são entradas de dados on-line.
5	Mais de 30% das transações são entradas de dados on-line.
	7. Interface com o usuário
	• Auxílio à navegação (teclas de função, acesso direto e menus dinâmicos). • Menus. • Documentação e help (auxílio) on-line. • Movimento automático do cursor.

Grau	• Scrolling vertical e horizontal. • Impressão remota (através de transações on-line). • Teclas de função preestabelecidas. • Processos batch submetidos a partir de transações on-line através de Hardcopy. • Utilização de mouse. • Menus pop-up. • O menor número possível de telas para executar as funções de negócio. • Suporte bilíngüe (o suporte a duas línguas; conte como quatro itens). • Suporte multilíngüe (o suporte a mais de duas línguas; conte como seis itens). Determinar, segundo os itens acima, qual o grau de enquadramento.
0	Nenhum dos itens descritos.
1	De um a três dos itens descritos.
2	De quatro a cinco dos itens descritos.
3	Mais de cinco dos itens descritos, mas não há requisitos específicos do usuário quanto a amigabilidade do sistema.
4	Mais de cinco dos itens descritos, e foram estabelecidos requisitos quanto a amigabilidade fortes o suficiente para gerarem atividades específicas envolvendo fatores, tais como minimização da digitação, para mostrar inicialmente os valores utilizados com mais freqüência.
5	Mais de cinco dos itens descritos, e foram estabelecidos requisitos quanto a amigabilidade fortes o suficientes para quererem ferramentas e processos especiais para demonstrar antecipadamente que os objetivos foram alcançados.
8. Atualização On Line	
0	Nenhum
1	Atualização on-line de um a três Arquivos Lógicos Internos. O volume de atualização é baixo e a recuperação de dados é fácil.

2	Atualização on-line de mais de três Arquivos Lógicos Internos. O volume de atualização é baixo e a recuperação dos dados é fácil.
3	Atualização on-line na maioria dos Arquivos Lógicos Internos.
4	Em adição ao item anterior, é necessário proteção contra perda de dados que foi desenhada e programada no sistema.
5	Além do item anterior, altos volumes trazem considerações de custo no processo de recuperação. Processos para automatizar a recuperação. Processos para automatizar a recuperação foram incluídos minimizando a intervenção do operador.
9. Processamento complexo	
Grau	• Controle sensitivo (processamento especial de auditoria) e/ou processamento de segurança específica da aplicação. • Processamento lógico extensivo. • Processamento matemático extensivo. • Processamento gerando muitas exceções, resultando em transações incompletas que devem ser processadas novamente. Exemplo: transações de auto-atendimento bancário interrompidas por problemas de comunicação ou com dados incompletos. • Processamento complexo para manusear múltiplas possibilidades de entrada/saída. Exemplo: multimídia. Determinar, segundo os itens acima, qual o grau de enquadramento.
0	Nenhum dos itens descritos.
1	Apenas um dos itens descritos.
2	Dois dos itens descritos.
3	Três dos itens descritos.
4	Quatro dos itens descritos.
5	Todos os cinco itens descritos.
10. Reusabilidade	
0	Nenhuma preocupação com reutilização de código.

1	Código reutilizado foi usado somente dentro da aplicação.
2	Menos de 10% da aplicação foi projetada prevendo utilização posterior do código por outra aplicação.
3	10% ou mais da aplicação foi projetada prevendo utilização posterior do código por outra aplicação.
4	A aplicação foi especificamente projetada e/ou documentada para ter seu código facilmente reutilizado por outra aplicação e a aplicação é customizada pelo usuário em nível de código-fonte.
5	A aplicação foi especificamente projetada e/ou documentada para ter seu código facilmente reutilizado por outra aplicação e a aplicação é customizada para uso através de parâmetros que podem ser alterados pelo usuário.
11. Facilidade de Implantação	
0	Nenhuma consideração especial foi estabelecida pelo usuário e nenhum procedimento especial é requerido na implantação.
1	Nenhuma consideração especial foi estabelecida pelo usuário, mas procedimentos especiais são necessários na implantação.
2	Requerimentos de conversão e implantação foram estabelecidos pelo usuário e roteiros de conversão e implantação foram providos e testados. O impacto da conversão no projeto não foi considerado importante.
3	Requerimentos de conversão e implantação foram estabelecidos pelo usuário e roteiros de conversão e implantação foram providos e testados. O impacto da conversão no projeto é considerado importante.
4	Além do item 2, conversão automática e ferramentas de implantação foram providas e testadas.
5	Além do item 3, conversão automática e ferramentas de implantação foram providas e testadas.
12. Facilidade Operacional	
0	Nenhuma consideração de operação, além do processo normal de salva foi estabelecido pelo usuário.
1 - 4	Verifique quais das seguintes afirmativas podem ser identificadas na aplicação. Selecione as que forem aplicadas. Cada item vale um ponto, exceto se definido explicitamente. • Foram desenvolvidos processos de inicialização, salva e recuperação, mas a intervenção do operador é necessária.

	• Foram estabelecidos processos de inicialização, salva e recuperação, e nenhuma intervenção do operador é necessária (conte como dois itens). • A aplicação minimiza a necessidade de montar fitas magnéticas. • A aplicação minimiza a necessidade de manuseio de papel.
5	A aplicação foi desenhada para trabalhar sem operador; nenhuma intervenção do operador é necessária para operar o sistema além de executar e encerrar a aplicação. A aplicação possui rotinas automáticas para recuperação em casos de erro.
13. Múltiplos Locais	
0	Os requisitos do usuário não consideram a necessidade de instalação em mais de um local.
1	A necessidade de múltiplos locais foi considerada no projeto, e a aplicação foi desenhada para operar apenas sobre o mesmo ambiente de software e hardware.
2	A necessidade de múltiplos locais foi considerada no projeto, e a aplicação está preparada para trabalhar apenas em ambientes similares de software e hardware.
3	A necessidade de múltiplos locais foi considerada no projeto, e a aplicação está preparada para trabalhar sob diferentes ambientes de hardware e/ou software.
4	Plano de documentação e manutenção foram providos e testados para suportar a aplicação em múltiplos locais; além disso, os itens 1 ou 2 caracterizam a aplicação.
5	Plano de documentação e manutenção foram providos e testados para suportar a aplicação em múltiplos locais; além disso, o item 3 caracteriza a aplicação.
14. Facilidade de Mudanças (Flexibilidade)	
	• Estão disponíveis facilidades como consultas e relatórios flexíveis para atender necessidades simples (conte como 1 item). • Estão disponíveis facilidades como consultas e relatórios flexíveis para atender necessidades de complexidade média (conte como 2 itens). • Estão disponíveis facilidades como consultas e relatórios flexíveis para atender necessidades complexas (conte como 3 itens). • Dados de controle são armazenados em tabelas que são mantidas

Grau	pelo usuário através de processos on-line, mas mudanças são tornadas efetivas somente no dia seguinte. • Dados de controle são armazenados em tabelas que são mantidas pelo usuário através de processos on-line, mas mudanças tem efeito imediatamente (conte como 2 itens). Determinar, segundo os itens acima, qual o grau de enquadramento.
0	Nenhum dos itens descritos.
1	Um dos itens descritos.
2	Dois dos itens descritos.
3	Três dos itens descritos.
4	Quatro dos itens descritos.
5	Todos os cinco itens descritos.

3.2.13. CALCULAR OS PONTOS DE FUNÇÃO AJUSTADOS

Cálculo de Pontos de Função para Projetos de Desenvolvimento

O cálculo dos Pontos de Função de um projeto de desenvolvimento consiste de três componentes da funcionalidade:

- Funcionalidade da aplicação.
- Funcionalidade de conversão.
- Fator de ajuste da aplicação.

Funcionalidade da aplicação

A funcionalidade da aplicação consiste das funções obtidas depois da instalação do software para satisfazer as necessidades do negócio do usuário.

Funcionalidade de conversão

A funcionalidade de conversão consiste das funções providas para converter dados ou necessidades específicas de conversão manifestadas pelo usuário.

Fator de ajuste da aplicação

O fator de ajuste da aplicação é determinado pelas 14 características gerais do sistema.

Fórmula de cálculo

A seguinte fórmula é utilizada na contagem de Pontos de Função para projeto de desenvolvimento.

DFP = (UFP + CFP) * VAF

Onde:

DFP é o número de Pontos de Função de desenvolvimento

UFP é o número de Pontos de Função Brutos

CFP é o número de Pontos de Função adicionados pelo processo de conversão

VAF é o fator de ajuste

Exemplo

Dada a contagem de Pontos de Função Brutos (UFP) do projeto de desenvolvimento de 107 e a contribuição da conversão em Pontos de Função Brutos (CFP) de 7, considerando um fator de ajuste (VAF) de 1.26, encontrar o número de Pontos de Função do projeto de desenvolvimento.

Aplicando a fórmula, obtém-se:

DFP = (UFP + CFP) * VAF

DFP = (107 + 7) * 1.26

DFP = 144 Pontos de Função

Cálculo de Pontos de Função de Projeto de Manutenção

O cálculo dos Pontos de Função de um projeto de manutenção envolve o cálculo de três componentes de funcionalidade.

- Funcionalidade da aplicação
- Funcionalidade de conversão
- Aplicação do Fator de Ajuste

Funcionalidade da aplicação

A funcionalidade da aplicação é composta por:

- Pontos de Função que são adicionados através da funcionalidade que é incluída no sistema.
- Pontos de Função que são contados, pois representam funcionalidade que sofreu alteração.
- Pontos de Função que são contados, pois representam a funcionalidade que foi excluída durante a manutenção.

Funcionalidade de conversão

Refletem os Pontos de Função entregues através da funcionalidade de conversão solicitada pelo usuário.

Fator de ajuste da aplicação

São considerados:

- Fator de ajuste da aplicação antes do início da manutenção.
- Fator de ajuste da aplicação após o término da manutenção.

Fórmula de cálculo

$$EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB)$$

Onde:

EFP Pontos de Função do projeto em manutenção.

ADD Pontos de Função Brutos que foram incluídos na aplicação pelo projeto de manutenção. Refletem as funções que foram adicionadas à aplicação.

CHGA Pontos de Função que foram alterados na aplicação pelo projeto de manutenção. Refletem as funções que sofreram alteração. Este número representa as funções após as modificações.

CFP São os Pontos de Função que foram adicionados pelo processo de conversão.

VAFA Fator de ajuste da aplicação depois do projeto de manutenção.

DEL Pontos de Função Brutos que foram excluídos da aplicação pelo projeto de manutenção. Refletem as funções que foram excluídas da aplicação.

VAFB Fator de ajuste da aplicação antes do projeto de manutenção.

É importante salientar que os Pontos de Função excluídos (DEL) são somados no projeto de manutenção, já que efetivamente há um esforço envolvendo sua exclusão.

Exemplo

O exemplo é composto por:

Funcionalidade Incluída: 7 Pontos de Função

Funcionalidade alterada: 14 Pontos de Função

Funcionalidade excluída: 3 Pontos de Função

Considerando que o fator de ajuste da aplicação permaneceu com o mesmo índice de 1,26, teríamos:

EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB)

EFP = [(7 + 14 + 0) * 1,26] + (3 * 1,26)

EFP = 30,24 Pontos de Função

Cálculo de uma aplicação

Quando um projeto de manutenção é encerrado, as contagens de Pontos de Função da aplicação devem ser atualizadas para refletir as manutenções realizadas. A funcionalidade da aplicação pode ser alterada de várias maneiras:

- A adição de novas funções aumenta o tamanho da aplicação.
- A alteração de funções pode aumentar, diminuir, ou não alterar o tamanho de uma aplicação.
- A exclusão de funções diminui o tamanho do sistema.
- A alteração do fator de ajuste pode aumentar, diminuir ou não alterar o tamanho de uma aplicação.

Nota: A funcionalidade de conversão não afeta a contagem dos Pontos de Função de uma aplicação.

A seguinte fórmula é utilizada para o cálculo dos Pontos de Função de uma aplicação após um projeto de manutenção:

$$APF = [(UFPB + ADD + CHGA) – (CHGB + DEL)] * VAFA$$

Onde:

APF São os Pontos de Função ajustados da aplicação.

UFPB São os Pontos de Função Brutos da aplicação antes do projeto de manutenção.

ADD São os Pontos de Função Brutos que foram adicionados pelo projeto de manutenção.

CHGA São os Pontos de Função Brutos correspondentes às funções que sofreram alteração durante o projeto de manutenção. Este número reflete as funções depois da manutenção.

CHGB São os Pontos de Função Brutos correspondentes às funções que sofreram alteração durante o projeto de manutenção. Este número reflete as funções antes da manutenção.

DEL São os Pontos de Função Brutos correspondentes às funções que foram excluídas da aplicação pelo projeto de manutenção.

VAFA É o fator de ajuste da aplicação verificado depois do projeto de manutenção.

Exemplo

A contagem do tamanho do aplicativo após o projeto de manutenção é mostrada a seguir:

APF = [(UFPB + ADD + CHGA) – (CHGB + DEL)] * VAFA

APF = [(107 + 7 + 14) – (14 + 3)] * 1,26

APF = 139,86

APF = 140 Pontos de Função

Observações finais

O projeto de desenvolvimento tinha um total de 107 Pontos de Função Brutos, sete Pontos de Função providos pelas funções de conversão de dados, resultando em um total de 144 Pontos de Função ajustados.

Após um projeto de manutenção, o tamanho do sistema pode permanecer igual, aumentar ou mesmo diminuir.

3.3. VANTAGENS E DESVANTAGENS DO USO DE ANÁLISE DE PONTOS DE FUNÇÃO

Vantagens

Existe uma coerência entre IFPUG (2000) e BRAGA (1996) em relação às vantagens da utilização da técnica de Análise por Ponto de Função que são descritas a seguir:

- Independência de tecnologia. Um conjunto de funções disponibilizadas por um sistema a seus usuários independente da tecnologia, possibilitando a comparação de produtividade entre ambientes de desenvolvimento.
- Provê uma unidade padrão de medida de software. Esta unidade permite a criação de métricas na área de produtividade e qualidade de software. Com o aumento das atividades na área de Informática, é reforçada a necessidade de se acompanhar a produtividade das equipes de

desenvolvimento, produtividade na manutenção de sistemas e ainda um programa de monitoração da qualidade do software.

- Provê um veículo para estimativas de desenvolvimento de sistemas. Com o crescente aumento da complexidade dos sistemas e conseqüentemente dos prazos de desenvolvimento, é necessário estimar, cada vez mais cedo, os prazos para cada fase do ciclo de desenvolvimento de sistemas.

- É simples o suficiente, para minimizar o esforço gasto com as medidas.

- É consistente através de diferentes projetos e utilizado por diferentes empresas, permitindo que as empresas troquem informações visando a melhoria dos seus processos e produtos.

- É compreensivo pelo pessoal não técnico, visando facilitar seu entendimento pelos usuários finais.

- Utilizável desde o início do ciclo de desenvolvimento de um sistema, onde ainda não se têm todas as informações necessárias, mas já se sabe o tamanho funcional.

- Uma ferramenta para determinar o tamanho de um pacote a ser adquirido através da contagem dos Pontos de Função que ele possui.

- Uma ferramenta de auxílio para determinar os benefícios que um pacote operacional pode oferecer a uma empresa, através da contagem dos Pontos de Função que refletem suas necessidades.

- Um veículo para medição de estimativas de custo e recursos requeridos para o desenvolvimento e manutenção de software.

- Um fator de normalização na comparação entre softwares.

- Acompanhamento da qualidade e produtividade visando à otimização do processo de desenvolvimento de sistemas.

- Uma ferramenta para auxiliar a decisão entre a compra de um pacote ou o desenvolvimento do aplicativo na empresa.

Em seu trabalho sobre a aplicação da técnica de Pontos de Função para analisar o portfólio de software de empresas de grande porte, Capers Jones (JONES, 1999), justifica a escolha da APF como métrica de tamanho de software por ser independente de tecnologia e melhor que a métrica LOC ("*Lines Of Code"*), que segundo sua ótica, torna-se impraticável em linguagens modernas de programação como Visual Basic da Microsoft e ABAP/4 da SAP. No relatório divulgado por Capers Jones, é apresentado um estudo sobre o tamanho das características do SAP R/3 ("*System, Applications, and Products"*), onde os seguintes tópicos são apresentados:

- Uso de Pontos de Função para avaliar aspectos de retorno de investimento do SAP

- Uso de Pontos de Função para customizar características do SAP

- Uso de Pontos de Função para estimar aplicações na linguagem ABAP/4 da SAP

Devido ao aparecimento de novos paradigmas de desenvolvimento de software, como o RUP *(Rational* Unified *Process)* e métodos ágeis, como o XP *(eXtreme Program)*, onde os requisitos são incrementais, torna-se difícil estimar o esforço de desenvolvimento nas fases iniciais. Análise de Pontos de função é a melhor alternativa para determinar o tamanho do software nas fases iniciais de projeto, por ser independente de linguagem de programação e padronizado (MOHAGHEGHI & OUTROS, 2005).

Desvantagens

A seguir, algumas desvantagens, segundo a ótica de alguns autores.

Pontos de Função têm falhas fundamentais em sua construção (KITCHENHAM, 1998). Por exemplo, existe a possibilidade de afirmar que um

programa é maior que o outro, quando na verdade não é. Se a análise de Pontos de Função for utilizada como *benchmarking* em empresa, como base para contrato de suporte ou desenvolvimento entre diferentes companhias, certamente haverá uma grande margem de risco na estimativa do tamanho funcional, que acarretará sérios problemas. A autora acima mencionada cita que várias empresas utilizam o tamanho funcional do software como entrada para modelos de estimativas de tempo e esforço e neste contexto o risco é maior ainda. A principal falha diagnosticada foi relativa a “problemas de construção”. Sob sua ótica, a construção da Análise de Pontos de Função de Albrecht envolve classificação de entradas, saídas, arquivos lógicos, interface de sistemas e consultas como simples, média e complexa. Isto significa que as contagens de escala absoluta são reduzidas a medidas de escala ordinal, e não há como aplicar outras transformações que não estejam no intervalo (simples, médio ou complexo).

Outro ponto fraco identificado na Contagem de Pontos de Função é referente à subjetividade. Kemerer, em sua pesquisa identificou uma diferença de 12% na contagem de Pontos de Função do mesmo produto realizado por pessoas diferentes (KEMERER, 1993).

Na visão de FELTON & PFLEEGER (1997), existem vários problemas com a medição de Pontos de Função, e enfatizam que usuários da técnica têm que conhecer tais limitações. Os autores citam alguns problemas que serão relatados a seguir:

Problemas com subjetividade no fator tecnologia

O fator de ajuste tem uma abrangência que vai de 0,65 a 1,35, portanto a contagem de Pontos de Função Brutos pode ser alterada por mais ou menos 35%. Então, a incerteza inerente à subjetividade devido ao grau dos sub-fatores de ajuste pode ter um significativo efeito no valor final dos Pontos de Função.

Problemas com contagem dupla

Segundo investigação Symons (1998), existe a possibilidade de contagem de complexidade interna em duas situações: a) no peso dado a entradas na contagem de Pontos de Função Brutos; b) no fator de ajuste da complexidade.

Problemas com valores intuitivos por parte do contador de Pontos de Função

Complementando o item anterior, a classificação dada aos sub-fatores leva ao fator de ajuste da complexidade, valores que são intuitivos por parte do contador. Albrecht incluiu o fator de ajuste da complexidade como um meio de melhorar a predição de recursos, acentuando tanto a complexidade interna do sistema quanto a complexidade associada com a funcionalidade na visão do usuário. Ele diz que estas duas complexidades deveriam ser medidas separadamente.

Problemas com precisão

Assim como em KEMERER (1993), foi realizada uma investigação empírica a respeito do uso de Pontos de Função como métrica. O estudo sugeriu que Pontos de Função não é uma métrica indicada porque existe correlação entre seus elementos constituintes (KITCHENHAM & KANSALA, 1993). Isto significa que sob a ótica da estatística, o Fator de Ajuste não é significante para melhorar estimativas.

Problemas com uso de Pontos de Função nas etapas iniciais do ciclo de vida

O cálculo de Pontos de Função requer uma total especificação do sistema de software, a documentação de requisitos do usuário não é suficiente.

Problemas com mudanças de requisitos

Em comparações realizadas com contagem de Pontos de Função geradas no início da especificação com a contagem obtida na entrega do sistema, verificou-se um incremento de 400 a 2000% (KEMERER, 1993). Isto acontece

devido principalmente ao refinamento dos requisitos conforme o progresso do desenvolvimento do sistema de software.

Problemas em distinguir itens especificados

Para minimizar o problema de julgamento de *experts* , algumas organizações (como por exemplo, o IFPUG) publicam regras detalhadas de contagem, porém o problema da subjetividade ainda perdura.

Problemas com dependência de tecnologia

Segundo pesquisas que utilizaram Pontos de Função em estimativas de funcionalidade de desenvolvimento de sistemas usando linguagem de quarta-geração, foi identificado uma acurácia melhorada para itens de entrada nas contagens de Ponto de Função, quando utilizado este tipo de linguagem.

Problemas com domínio da aplicação

Análise de Pontos de Função tem sido largamente utilizada em aplicações de processamento de dados, domínio do qual foi originalmente desenvolvido, porém seu uso em aplicações científicas e de tempo-real geram controvérsias.

Problemas com peso no quesito subjetividade

Os pesos escolhidos para calcular Pontos de Função Brutos foram determinados subjetivamente por experiência da IBM. Estes valores podem não ser apropriados para outros ambientes de desenvolvimento.

Conclusão

Embora a literatura apresente uma série de problemas com relação à Análise de Pontos de Função, ela é amplamente utilizada e já está sedimentada através de uma metodologia padronizada com usuários no mundo todo, razão esta para a adotarmos como métrica de tamanho funcional de software

4. TRABALHOS CORRELATOS

4.1. CHECKMARK - UMA FERRAMENTA DE ESTIMATIVA DE SOFTWARE BASEADA EM APF

Pensando em amenizar problemas decorrentes de estimativas de custo e esforço envolvendo projetos de software, foi realizado um trabalho que procurou minimizar o impacto causado por estimativas baseadas em KLOC, onde há a necessidade de se determinar a tecnologia de desenvolvimento (linguagem de baixo nível, quarta-geração etc.). A ferramenta selecionada para tal experimento foi a "CHECKMARK" (desenvolvida por SPR Inc. CHECKMARK, que é registrada como CHECKPOINT nos EUA), que possui duas características: A ferramenta utiliza Análise de Pontos de Função (APF) ao invés de SLOC (Source Lines Of Code, ou em português, Linhas de Código Fonte); e são considerados fatores especiais presentes em desenvolvimento de software (FUJIWARA E OUTROS, 1998).

Razões para adotar APF

A organização onde o estudo foi aplicado possuía várias linguagens de programação e ferramentas. Por ser independente da técnica de implementação física e ser concebida sob a ótica do usuário, APF foi escolhida para determinar o tamanho funcional do software. Outro ponto a considerar é que a APF utilizada foi uma variação da APF de Albretch, a "Feature Point Analysis" adaptada pela SPR (Software Productivity Research Inc.). Feature point analysis possui as mesmas características da APF, além de incorporar mais uma característica: a medida de complexidade de "algoritmo". A justificativa para a utilização desta variação da APF foi o tipo de desenvolvimento de software, o CAD (Computer Aided Design, ou em português, Desenho Auxiliado por Computador).

Funções da Ferramenta CHECKMARK

Esta ferramenta provê três funções: “estimativa”, ”medição” e “avaliação”. O trabalho foi realizado com a função “estimativa”. Esta função tem em torno de 6.000 itens de dados que compõe sua *BASELINE*.

A ferramenta possui quatorze entradas para estimar o esforço. São eles:

QUADRO 23 – ENTRADAS PARA ESTIMAR O ESFORÇO

1. Descrição do Projeto	Data de início, data final, etc.
2.Classificação do Projeto	Novo ou de manutenção, projeto grande ou pequeno, programa interno ou produto comercial e domínio do projeto
3. Metas do Projeto	Condições de estimativa como estimativa padrão, restrição na qualidade, ou restrição no cronograma, etc.
4. Complexidade do Projeto	Completa complexidade do projeto na visão de algoritmo, programa, e dados do software
5. Tamanho funcional	Contagem de Pontos de Função
6. Código Fonte	Linguagem de programação de cada código classificado como novo, alteração, reuso e deletado.
7. Custo do Projeto	Média de salários, etc.
8. Seleção de Tarefas	Divisão de fases e tarefas
9. Restrição de Desenvolvimento	Restrição na duração ou pessoal de desenvolvimento
10. Fatores Pessoais	Experiência no desenvolvimento de métodos e linguagens
11. Fatores de Tecnologia	Condições nas ferramentas ou hardware
12. Fatores do Processo	Desenvolvimento de métodos ou

	métodos de garantia de qualidade, etc.
13. Fatores do Ambiente	Condições de localizações de desenvolvimento, etc.
14. Fatores Especiais	Riscos, etc.

O estudo concluiu que a ferramenta para estimativa de projetos de software tem uma acurácia satisfatória. Foi identificado um coeficiente de correlação de 0,89 entre o Esforço Estimado e o Esforço Medido numa base de 51 projetos de software.

4.2. ESTIMATIVA- FERRAMENTA DA SERPRO BASEADA EM APF

Visão Geral

Segundo CANDÉAS & LOPES (1999), o ESTIMATIVA compõe um plano estratégico de medições do SERPRO (Serviço Federal de Processamento de Dados) e automatiza a técnica de Análise de Pontos de Função auxiliando os líderes de projeto nas pontuações, além de oferecer estimativas de prazo para um projeto a ser desenvolvido de acordo com a produtividade da equipe.

O SERPRO, empresa pública de Tecnologia de Informação, vinculada ao Ministério da Fazenda, é constituído de várias Superintendências de Negócio que desenvolvem sistemas para atender os clientes. Dentre as Superintendências, a SUNAT (Superintendência de Negócios – Administração Tributária) iniciou, em 1998, o processo de implantação de um plano de medições adotando a métrica de Análise de Pontos de Função (APF) baseando-se no padrão internacional estabelecido pelo IFPUG (Internacional Function Points Users Group), conferindo confiabilidade ao processo. Quando utilizada em combinação com outras medidas, a APF pode ser utilizada para determinar, por exemplo, o custo, a taxa de produção e manutenção, ou até a qualidade do sistema do ponto de vista do usuário.

Objetivo do sistema

O ESTIMATIVA visa automatizar o trabalho de medição do Processo de Desenvolvimento de Sistemas minimizando o esforço dos líderes de projeto ao interagir com o Sistema, obtendo, de maneira simples e dinâmica, uma compreensão intuitiva do porte do projeto em termos de Pontos de Função. Seu objetivo não se restringe apenas à pontuação de sistemas mas também oferece a possibilidade de fazer estimativas de prazo para um projeto a ser desenvolvido. Após quantificar um projeto, o Sistema estima o esforço de produtividade de acordo com as plataformas e as equipes de desenvolvimento alocadas. Além de outras funcionalidades inerentes à técnica de APF, o Sistema se propõe a integrar as informações dos projetos para, a partir de uma base histórica (*BASELINE*), cruzar informações propiciando análises comparativas com outros sistemas internos a fim de se obter subsídios para o fornecimento de estimativas mais próximas da realidade da Empresa.

Nível macro do ESTIMATIVA

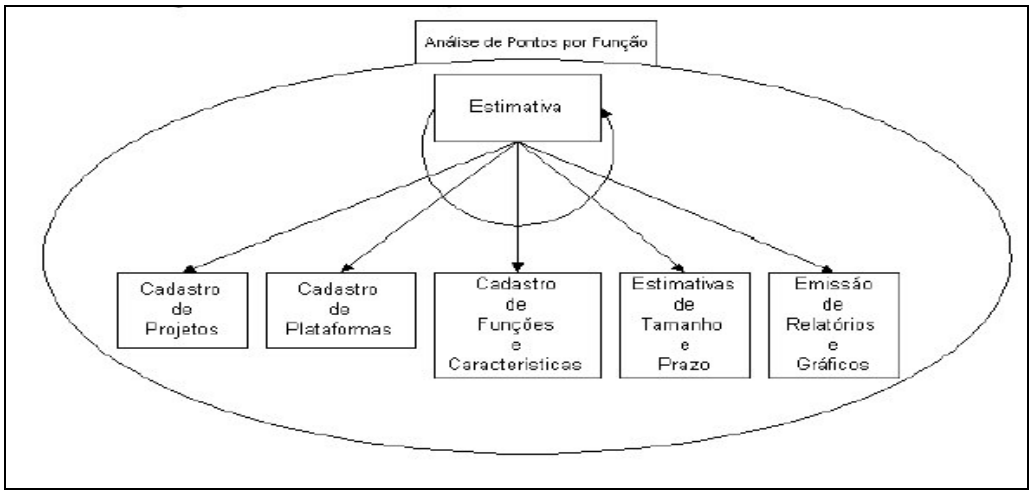


FIGURA 4 – NÍVEL MACRO DO ESTIMATIVA

De uma maneira global, as funções realizadas pelo ESTIMATIVA são descritas a seguir:

- **Cadastro de Projetos:** É o processo de catalogação dos projetos a serem pontuados. Também é possível importar projetos da base histórica (*BASELINE*)

- **Cadastro de Plataformas:** Diz respeito à linguagem com qual serão desenvolvidas as funções associadas, além da quantidade de desenvolvedores envolvidos e o nível de experiência na linguagem definida.

- **Cadastro de Funções e Características:** Neste módulo é feita a entrada nas Funções e Características pertinente à plataforma selecionada, podendo incluir, alterar ou excluir funções, além de definir o nível de influência para cada uma das 14 características contempladas pela técnica de APF.

- **Estimativa de Tamanho e Prazo:** Efetua o cálculo dos Pontos de Função a partir das funções e características cadastradas, fornecendo estimativas de horas de desenvolvimento de acordo com a linguagem e equipe alocados para as plataformas. As estimativas estão sendo calculadas tomando como base a tabela da SPR (Software Productivity Research), que utiliza o conceito de Níveis de Linguagem.

- **Emissão de Relatórios e Gráficos:** Permite a geração de relatórios analíticos, bem como geração de gráficos com uma melhor visualização dos dados.

4.3. GEMETRICS – FERRAMENTA CASE PARA GERENCIAMENTO DE PROJETOS

Esta ferramenta é o resultado de um esforço entre as universidades Vale do Itajaí (Itajaí/SC) e Federal de Santa Catarina (Florianópolis/SC) no sentido de integrar em um único aplicativo de software, características das ferramentas CASE (*Computer-Aided Software Engineering)* de gerenciamento de projetos e métricas de software existentes CBComp2001 (2001).

Objetivo

A ferramenta CASE GEMETRICS tem como objetivo: disponibilizar ao gerente de projeto: estimativas de esforço; custo de um projeto de software; definição de uma estrutura de divisão de trabalho; planejamento de uma programação viável de projeto e acompanhamento de projetos em base contínua. Além disso, o gerente pode usar a ferramenta para compilar métricas, que por fim oferecerão uma indicação da produtividade no desenvolvimento de software e da qualidade do produto.

Métricas de Software

Análise de Pontos de Função foi a métrica adotada para medir o tamanho do software e a partir do tamanho poder estimar esforço e custo. Este módulo do GEMETRICS segue as regras estipuladas pelo IFPUG. A escolha da métrica de Pontos de Função, segundo seus idealizadores, deve-se ao fato dos Pontos de Função serem a única medida independente de plataforma ou linguagem, além de se tratar de um padrão mundialmente reconhecido, existente a cerca de 15 anos. A figura a seguir ilustra o módulo em que é realizada a contagem de Pontos de Função.

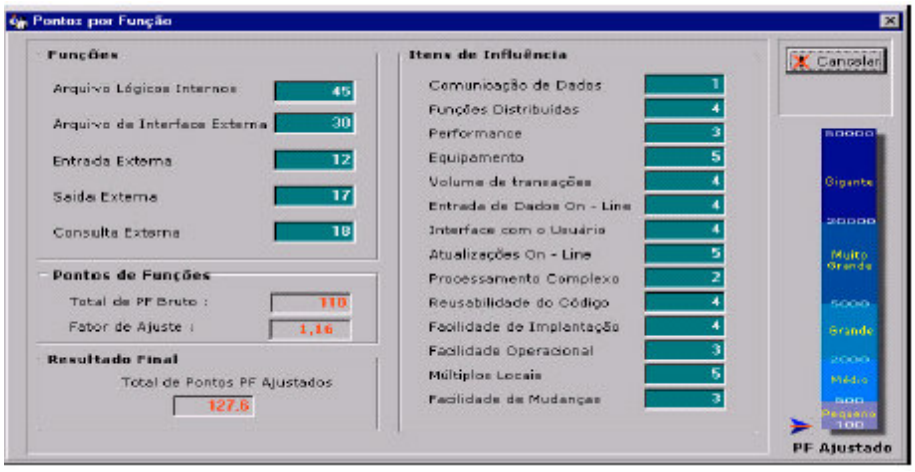


FIGURA 5 – MÓDULO ANÁLISE DE PONTOS DE FUNÇÃO

4.4. FPRECORD LITE VERSÃO 2.00

Este software tem por objetivo fazer a contagem de Pontos de Função (PF). Ele não trabalha com estimativas a partir dos PF encontrados. A ferramenta de Contagem de Pontos de Função foi desenvolvida por Chis Pty Ltd 1999,2000 e pode ser encontrada no endereço www.chispl.com. A tela mostrada na figura a seguir exibe a interface gráfica utilizada na obtenção dos dados para a Contagem dos Pontos de Função.

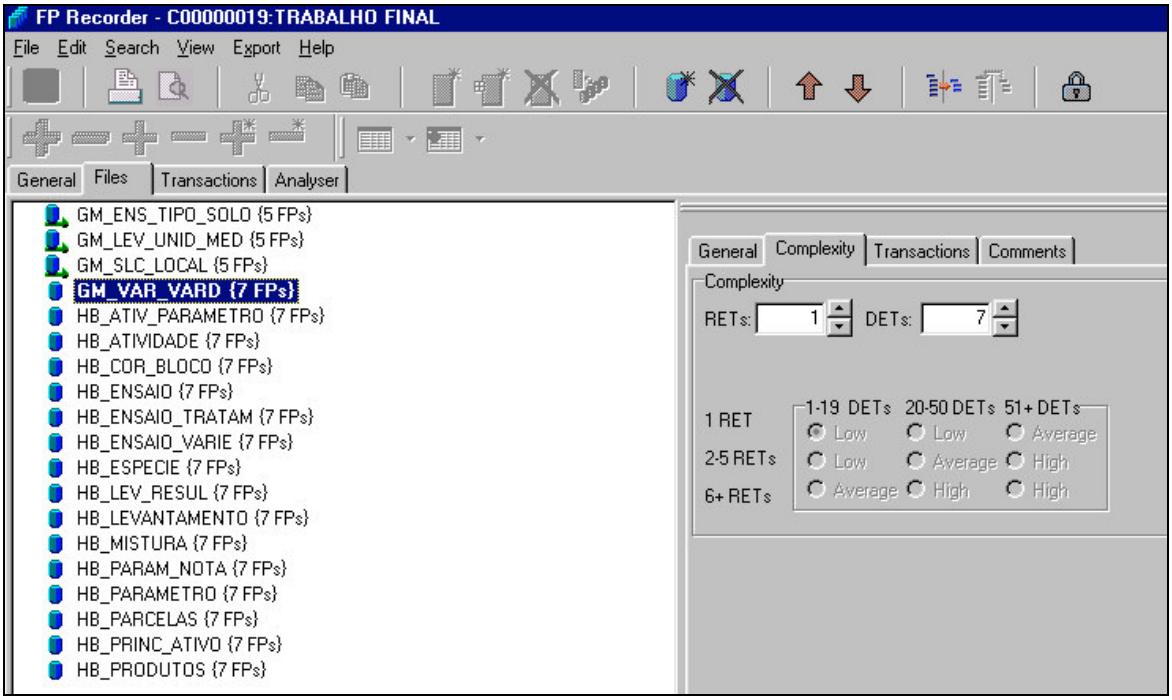


FIGURA 6 – INTERFACE GRÁFICA DO FPRECORD LITE.

4.5. SPR KNOWLEDGEPLAN - FERRAMENTA DE ESTIMATIVA E PLANEJAMENTO DE SOFTWARE

A SPR KnowledgePLAN é uma poderosa ferramenta de estimativa de projetos de software. Desenvolvida em ambiente visual, possui uma interface intuitiva e conta com a ajuda de *wizards,* ou assistentes, que ajudam os usuários a construírem estimativas. Sua *knowledge base*, ou base de conhecimento é

composta por mais de 8.000 projetos de software que cobrem a maioria dos ambientes computacionais SPR (2005).

Objetivo

O principal objetivo da ferramenta é gerar estimativas de custo, cronograma de desenvolvimento e prazo de entrega de projetos de software.

Características

A ferramenta, por possuir uma *baseline* formada por mais de 8.000 projetos de software, oferece aos usuários um assistente que através de interface amigável, questiona os usuários no sentido de refinar os projetos da base de dados que mais se assemelham às características da organização. Basicamente os assistentes solicitam uma nota que oscila de 0 a 5 (0 para “não se aplica”, até 5 para “totalmente satisfeito”) para cada uma das características qualitativas e quantitativas de sua base de dados, como mostradas a seguir:

- Nível de experiência do time, subdividido em experiência com a linguagem de programação, com a análise e com o hardware;
- Nível de uso de metodologias e ferramentas de desenvolvimento;
- Nível da qualidade de desenvolvimento, aplicado ao ciclo de vida do projeto;
- Nível da arquitetura computacional da empresa, como a utilização de técnica de OO (Orientação a Objetos) aplicada à análise e/ou ao desenvolvimento de software, utilização da arquitetura cliente-servidor;
- Nível de familiaridade com a arquitetura computacional da organização, entre outras.

A ferramenta ainda oferece três metodologias para encontrar o tamanho do software, são elas: Tamanho por Analogia, onde projetos análogos são selecionados da base de conhecimento; Tamanho por Componente, usado quando o desenvolvedor especifica o número aproximado de componentes de

software, como as telas e relatórios; e Tamanho por Métrica, onde a ferramenta oferece suporte à contagem de pontos de função, como mostrado na figura a seguir.

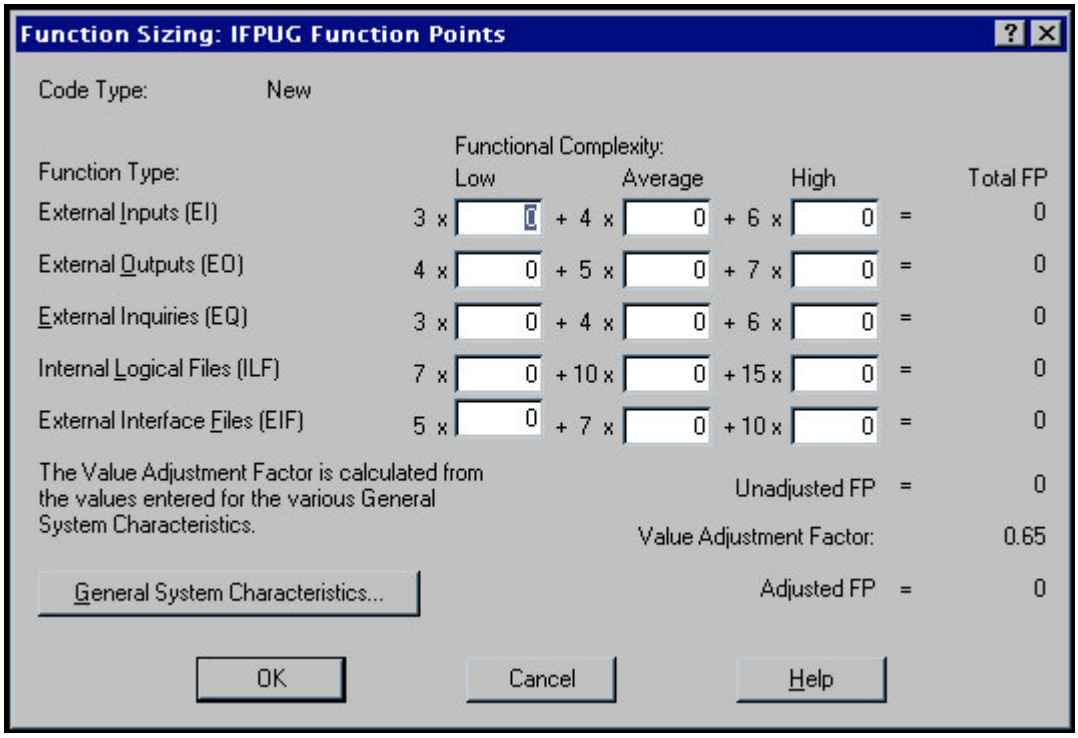


FIGURA 7 – TELA DE SUPORTE A CONTAGEM DE PONTOS DE FUNÇÃO DO KNOWLEDGEPLAN – SPR (2005).

O produto final da ferramenta é um cronograma com as principais atividades do ciclo de vida do projeto, com opção de comunicação com as principais ferramentas de gerenciamento de projetos, entre elas o *MSProject* e o *Artemis Views,* como exibido na figura a seguir.

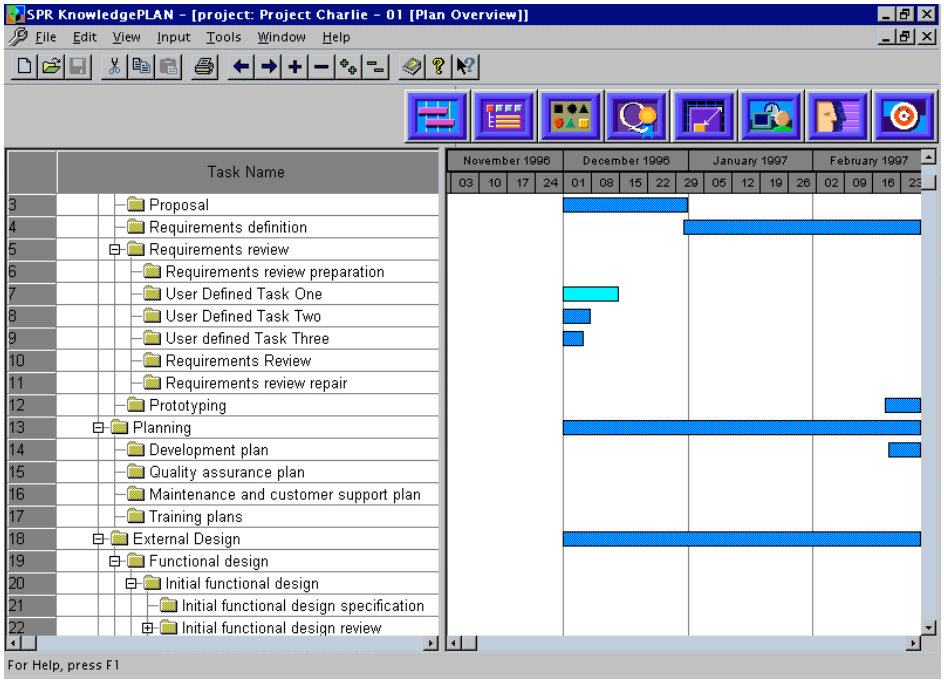


FIGURA 8 – TELA DE CRONOGRAMA DO KNOWLEDGEPLAN – SPR (2005).

Existe um módulo comercializado separadamente, o KnowledgePLAN 3.0, que permite que os usuários customizem a base de conhecimento para refletir: índices internos de produtividade; tamanhos dos softwares entregues; metodologia utilizada e níveis de recursos. Bibliotecas de customização da base de conhecimento podem ser criadas e compartilhadas pela organização.

4.6. CONCLUSÃO

Várias são as ferramentas de estimativas de desenvolvimento de software encontradas no mercado e a maioria delas gerenciam *baselines* genéricas e utilizam APF como métrica de tamanho de software. A ferramenta proposta neste trabalho também utiliza APF e diferencia-se por gerenciar *baseline* particularizada por equipe de desenvolvimento e ser de natureza livre. Na seção 4.5 verificamos que a ferramenta KnowledgePLAN tem um módulo que se preocupa em customizar a base de conhecimento de acordo com a organização, reforçando o propósito desta dissertação. Como aspecto negativo

da ferramenta proposta em relação às apresentadas, citamos a ausência do módulo responsável por gerar estimativas de desenvolvimento de software.

5. DESENVOLVIMENTO DA FERRAMENTA

Reconhecendo a importância de pontos de função como métrica de tamanho funcional de software e de uma *baseline* particularizada por equipe de desenvolvimento de projetos de software, discorridos respectivamente nos capítulos 2 e 3, decidiu-se desenvolver uma ferramenta capaz de gerenciar uma *baseline* de projetos, além de oferecer suporte à contagem e armazenamento dos pontos de função das aplicações, segundo metodologia do IFPUG.

No processo de elicitação dos requisitos, a fonte principal foi a revisão bibliográfica a respeito de *baselines*. Procurou-se incorporar à ferramenta, as características qualitativas e quantitativas identificadas pelos vários autores pesquisados no capítulo 2. Quanto à metodologia de contagem dos pontos de função, o manual prático de contagem IFPUG (2000) forneceu as regras e o método de contagem.

A ferramenta foi implementada em JAVA e utiliza como SGBD (Sistema Gerenciador de Banco de Dados) o MySQL. Tanto na escolha da linguagem de desenvolvimento quanto do SGBD, levou-se em consideração a questão do uso livre, pois pretende-se que a ferramenta seja distribuída livremente entre as equipes de desenvolvimento (várias empresas).

5.1. MODELAGEM DOS REQUISITOS

Para expressar as características e funcionalidades da ferramenta, foram utilizados vários diagramas da UML (*Unified Modeling Languange)*, como mostrado a seguir.

5.1.1. MODELAGEM DE CASOS DE USO

O modelo de casos de uso apresentado na figura 9 identifica as principais funcionalidades da ferramenta. Os casos de uso referentes aos projetos

(cadastrar ambiente, cadastrar pessoas, cadastrar, alterar e excluir projetos) referem-se a funcionalidades atribuídas ao gerenciamento da *baseline*, enquanto que os casos de uso Contar, Alterar e Consultar PF do projeto, dizem respeito à APF. Neste modelo de casos de uso temos uma visão geral da ferramenta.

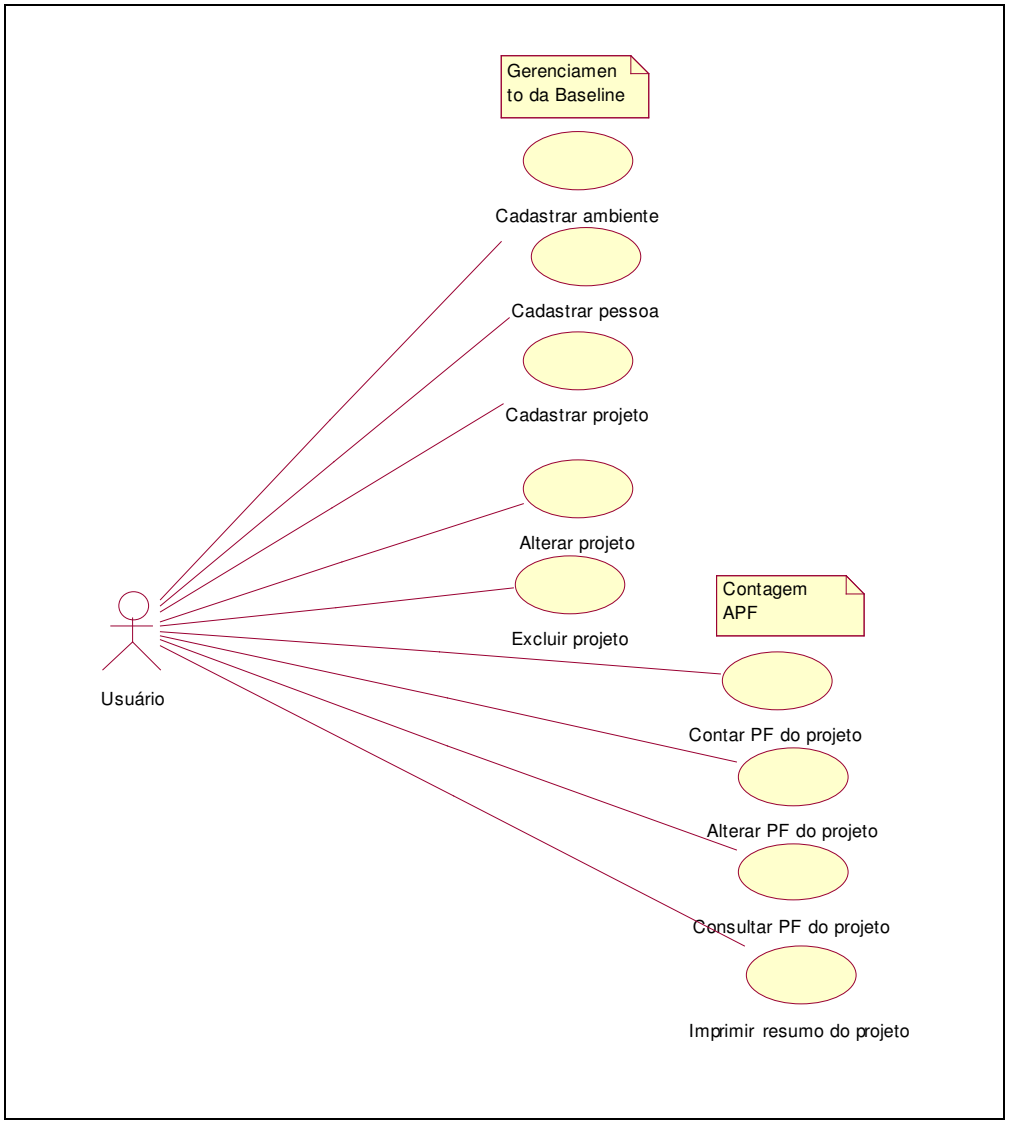


FIGURA 9 – REQUISITOS FUNCIONAIS IMPLEMENTADOS NA FERRAMENTA

Para expressar o gerenciamento da *baseline*, foi utilizado o modelo de casos de uso a seguir.

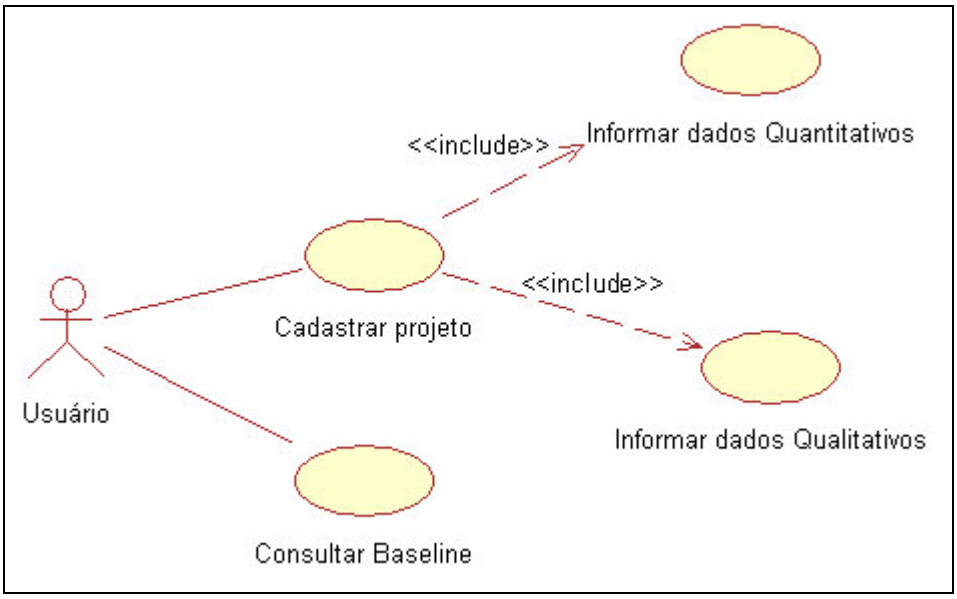


Figura 10 – Gerenciamento da *baseline*

As informações necessárias para o gerenciamento da *baseline* concentram-se basicamente no caso de uso cadastrar projeto. São tratados os dados quantitativos e qualitativos no referido caso de uso. Os dados quantitativos do projeto refletem aspectos como: duração do projeto; tamanho funcional; esforço de desenvolvimento; entre outros. Os dados qualitativos refletem características como a habilidade da equipe e o ambiente tecnológico.

Para expressar o método de contagem dos pontos de função, foi utilizado o modelo de casos de uso mostrado na figura 11. Os casos de uso refletem as macro etapas da contagem de pontos de função:

a. Contar as funções de dados e transacionais, a fim de obter os pontos de função brutos;

b. Avaliar as quatorze características gerais que serão aplicadas aos pontos de função brutos, obtendo assim os pontos de função ajustados.

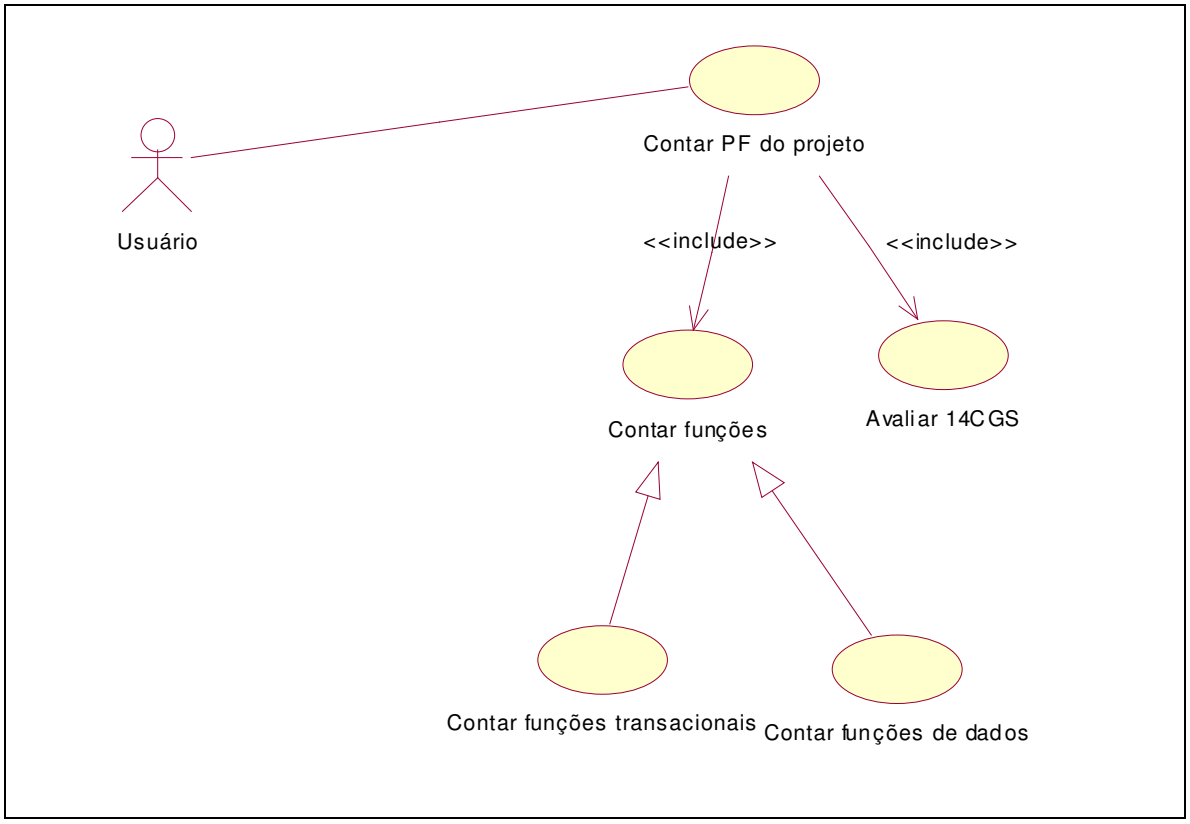


FIGURA 11 – CASO DE USO CONTAR PONTOS DE FUNÇÃO

5.1.2. MODELAGEM DE ATIVIDADES

Para expressar a lógica envolvida na APF, vários diagramas de atividades serviram de apóio. A figura 12 representa a lógica principal para determinar os pontos de função de um projeto. Basicamente a lógica consiste em:

a. Contar as funções de dados e transacionais, obtendo os pontos de função brutos;

b. Determinar as quatorze características gerais do sistema;

c. Calcular o Fator de Ajuste através das quatorze características gerais do sistema;

d. Aplicar o fator de ajuste aos pontos de função brutos, obtendo os pontos de função ajustados.

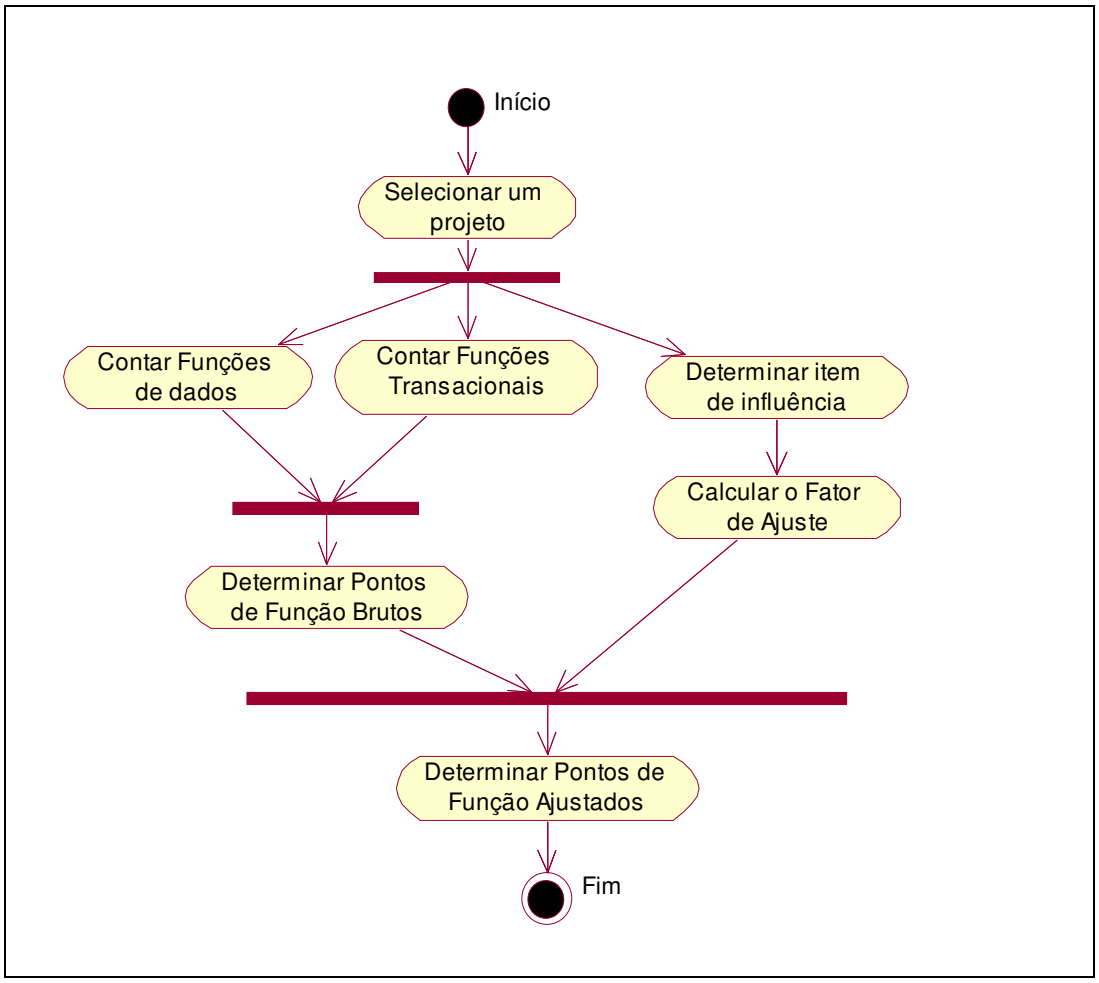


FIGURA 12 – DIAGRAMA DE ATIVIDADES CONTAR PF DO PROJETO

Para expressar a lógica de contagem das funções de dados, foi utilizado o diagrama de atividades expresso na figura 13. As etapas da contagem são descritas a seguir:

a. Identificar uma função de dados.

b. Identificar um agrupamento lógico de dados que corresponda a um arquivo.

c. Identificar o tipo de função, arquivo lógico interno ou arquivo de interface externa.

d. Contar os registros lógicos e os itens de dados.

e. Determinar, segundo a tabela do IFPUG, a complexidade da função de dados.

f. Determinar a quantidade de pontos de função brutos relacionada com a complexidade encontrada no item acima.

g. Armazenar os pontos de função brutos encontrados

h. Realizar o procedimento até identificar todas as funções de dados.

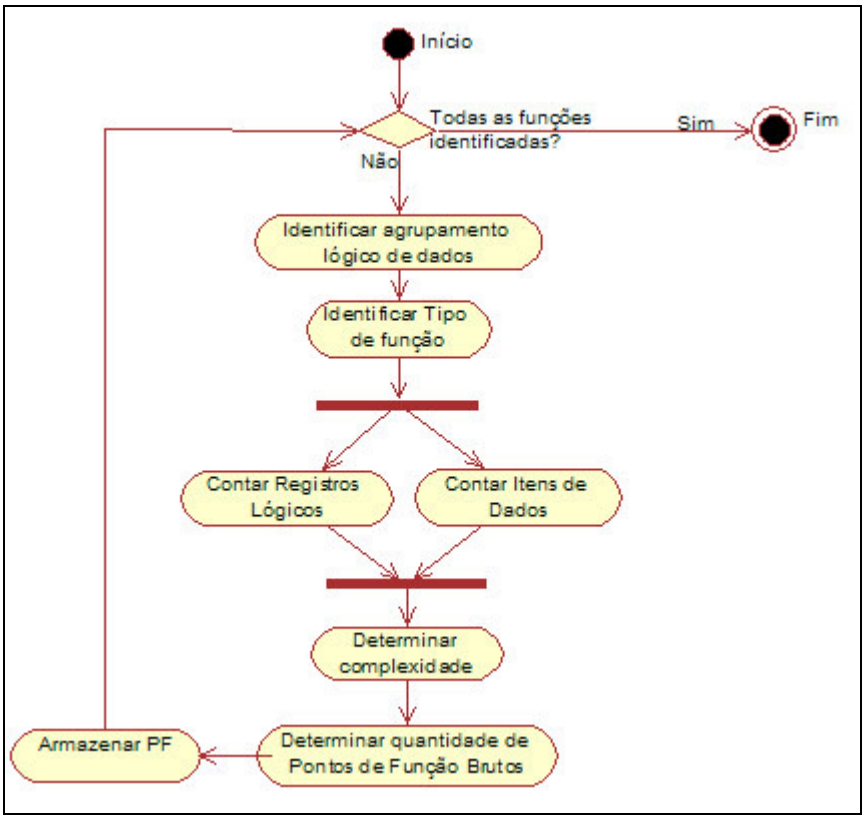


FIGURA 13 – DIAGRAMA DE ATIVIDADES CONTAR FUNÇÕES DE DADOS.

A lógica de contagem das funções transacionais está expressa na figura 14. O procedimento, semelhante à contagem das funções de dados, é descrito a seguir:

a. Identificar uma função transacional.

b. Identificar um processo elementar que corresponda a uma transação.

c. Identificar o tipo de função, entrada externa, saída externa ou consulta externa.

d. Contar os itens de dados e os arquivos lógicos referenciados.

e. Determinar, segundo a tabela do IFPUG, a complexidade da função transacional.

f. Determinar a quantidade de pontos de função brutos relacionada com a complexidade encontrada no item acima.

g. Armazenar os pontos de função brutos encontrados.

h. Realizar o procedimento até identificar todas as funções transacionais.

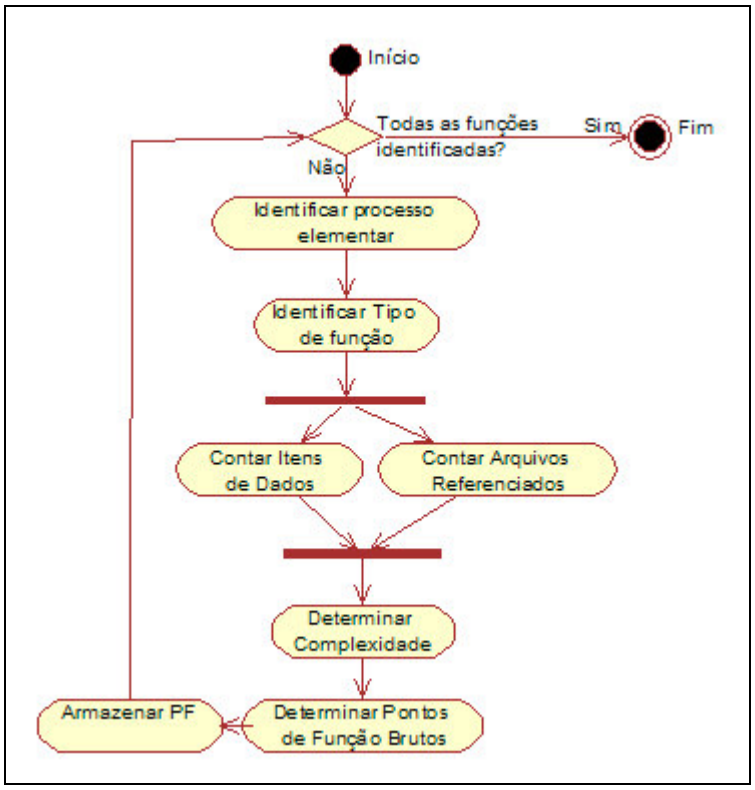


FIGURA 14 – DIAGRAMA DE ATIVIDADES CONTAR FUNÇÕES TRANSACIONAIS.

A lógica para determinar as quatorze características gerais do sistema (CGS) é expressa na figura 15 e obedece aos seguintes procedimentos:

a. Selecionar uma das 14 CGS.

b. Atribuir para a característica selecionada, uma nota (peso) que varia de zero (0) a cinco (5).

c. Armazenar a nota dada à característica.

d. Realizar o procedimento para as 14 CGS.

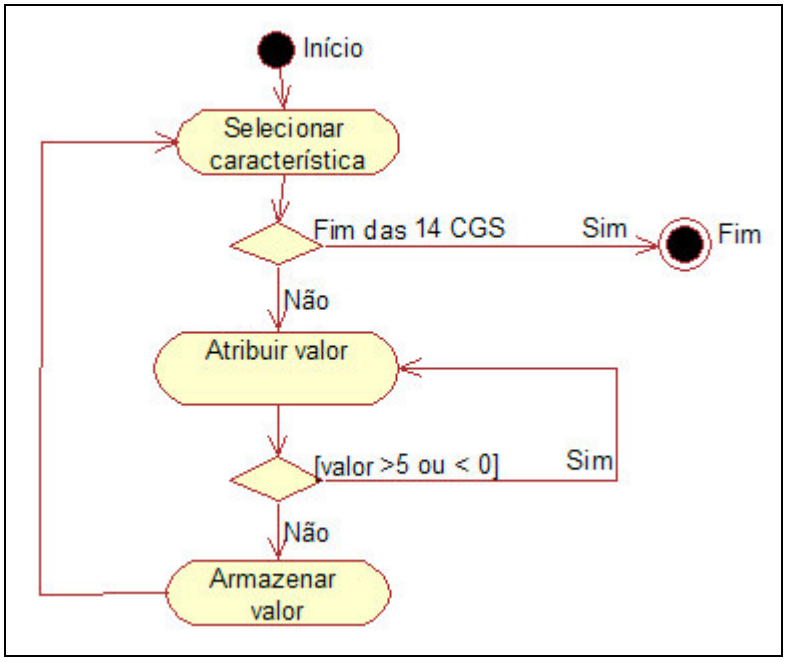


FIGURA 15 – DIAGRAMA DE ATIVIDADES DETERMINAR AS 14 CGS.

5.2. ESTRUTURA DA *BASELINE*

O gerenciamento da *baseline* foi representado pelo Diagrama de Classes apresentado na figura 16. A escolha deste diagrama deveu-se à natureza estática representada pela *baseline*, ou seja, o armazenamento das

informações que compõem uma *baseline*. A seguir serão apresentadas as principais classes.

Na classe Ambiente, estão representadas as informações referentes ao ambiente computacional da organização. O ambiente será formado por um Sistema Operacional, Sistema Gerenciador de Banco de Dados e a Linguagem de programação. Após estabelecer os Ambientes, o usuário poderá associar um a vários projetos da *baseline*.

A classe Pessoa é responsável por representar o perfil da equipe. Possui os atributos nome, atribuição (líder de equipe, analista de sistemas, programador, etc.) e experiência do profissional em anos. Após estabelecer as pessoas envolvidas, o usuário poderá associar pessoas a projetos.

A classe Projeto representa as características qualitativas e quantitativas, tais como datas de início e término do projeto, número de pessoas-mês, custo do projeto, Linhas de Código (LOC) e Pontos de Função (PF). As classes Funções de Dados, Funções Transacionais e Item de Influência dão suporte à contagem de Pontos de Função.

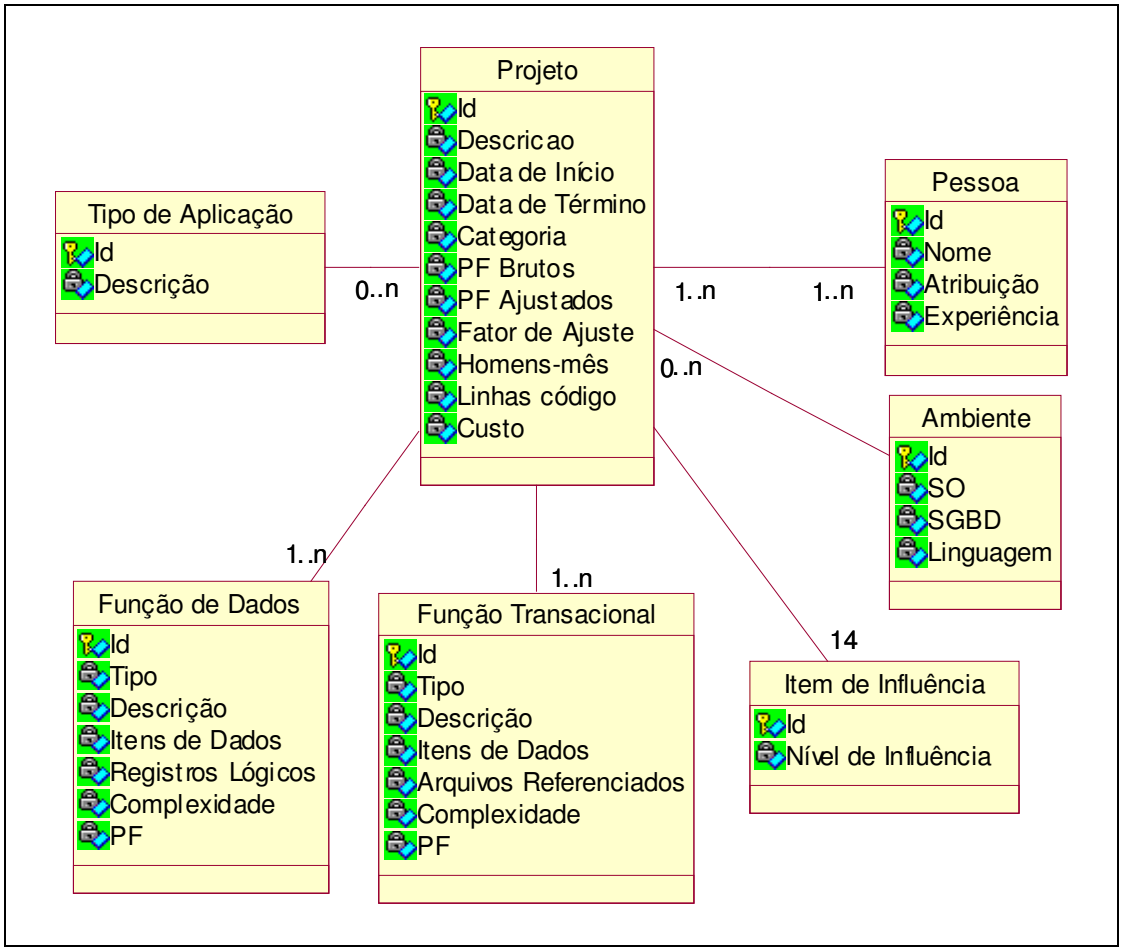


FIGURA 16 – ESTRUTURA DE ARMAZENAMENTO DA BASELINE.

5.3. ARQUITETURA DA FERRAMENTA

Novamente utilizamos a UML para representar as classes envolvidas na arquitetura da ferramenta. Estas classes refletem a implementação dos diagramas anteriormente apresentados. Devido ao número elevado de classes utilizadas, serão apresentadas as partes principais que compõem as funcionalidades da ferramenta. A figura 17 exibe uma visão da ferramenta como um todo. Observa-se que a maioria das classes são acessadas através da classe principal “MenuApf”, verificada pelo relacionamento de agregação.

Esta classe representa o menu principal do sistema, responsável pelas chamadas das funcionalidade do sistema.

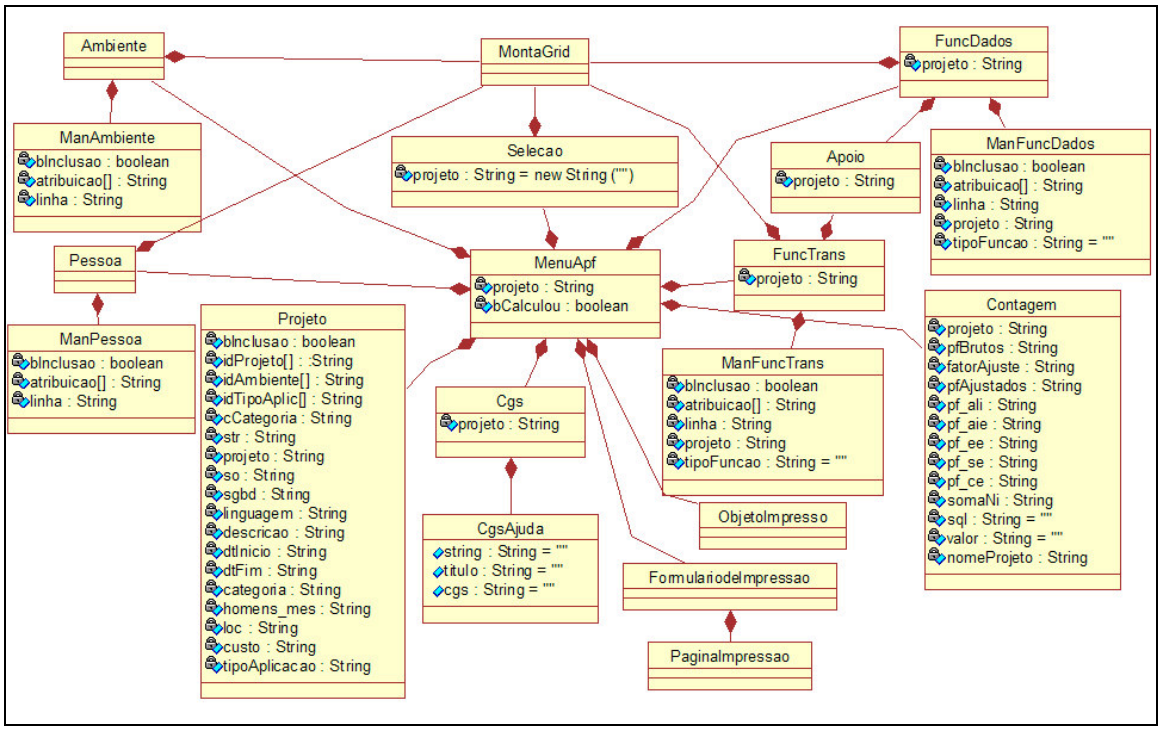


FIGURA 17 – VISÃO GERAL DA FERRAMENTA

As duas figuras a seguir, ilustram as duas principais funcionalidades do sistema, a figura 18 exibe a arquitetura interna que permite o gerenciamento da *baseline*. A figura 19 mostra a arquitetura para suportar a contagem dos pontos de função. A seguir serão apresentadas as principais classes envolvidas na arquitetura interna da ferramenta segundo a visão de gerenciamento da *baseline*.

a. Acesso: Classe responsável pela camada de interface entre a aplicação e o SGBD.

b. Pessoa/ManPessoa: Classe responsável pelo gerenciamento dos atributos das pessoas envolvidas no desenvolvimento dos projetos.

c. Ambiente/ManAmbiente: É a classe responsável pelo gerenciamento dos possíveis ambientes de desenvolvimento de sistemas, constituídos de linguagem de programação, sistema operacional e SGBD.

d. Projeto: é a classe principal de gerenciamento da *baseline*. Responsável por agrupar e gerenciar as informações quantitativas e qualitativas dos projetos.

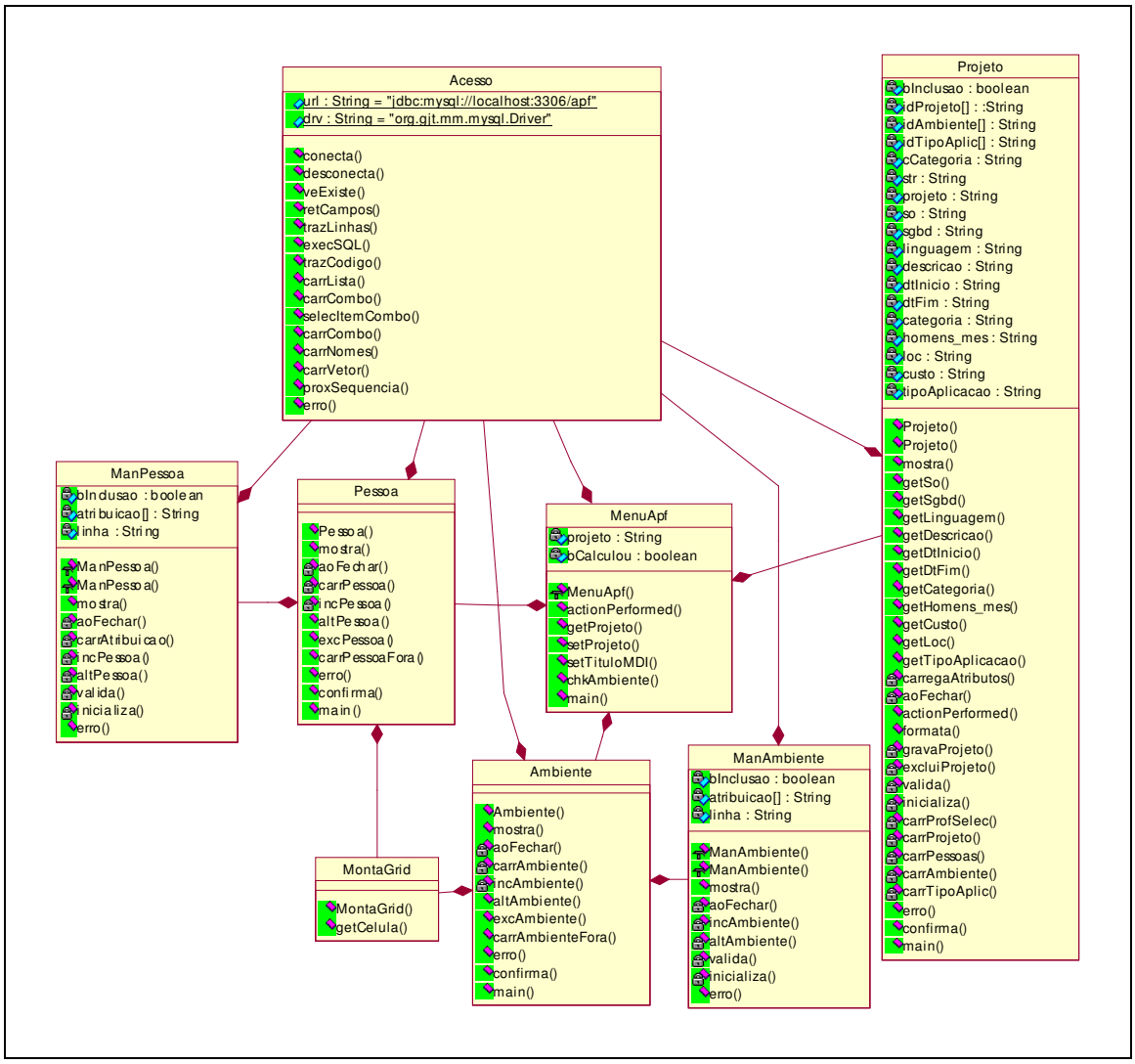


FIGURA 18 – VISÃO DA BASELINE

Para suportar a APF, que é uma funcionalidade acrescida à ferramenta, foram implementadas as classes listadas a seguir:

a. Acesso: Classe responsável pela camada de interface entre a aplicação e o SGBD.

b. FuncDados/ManFuncDados: Classes responsáveis pela manutenção dos pontos de função relativos à função de dados.

c. FuncTrans/ManFuncTran: Classes responsáveis pela manutenção dos pontos de função relativos às funções transacionais.

d. Cgs: Esta classe implementa as 14 Características Gerais do Sistema.

e. CgsAjuda: Classe responsável pela ajuda de contexto ao usuário no que diz respeito às 14 Características Gerais do Sistema.

f. Contagem: A referida classe implementa a metodologia de Análise de Pontos de Função. A classe recebe como parâmetro o código do projeto, e a partir dele, executa todos os cálculos previstos na metodologia. Após os cálculos, a classe atualiza a *baseline*, informando os PF brutos, PF ajustados e o fator de ajuste do projeto calculado.

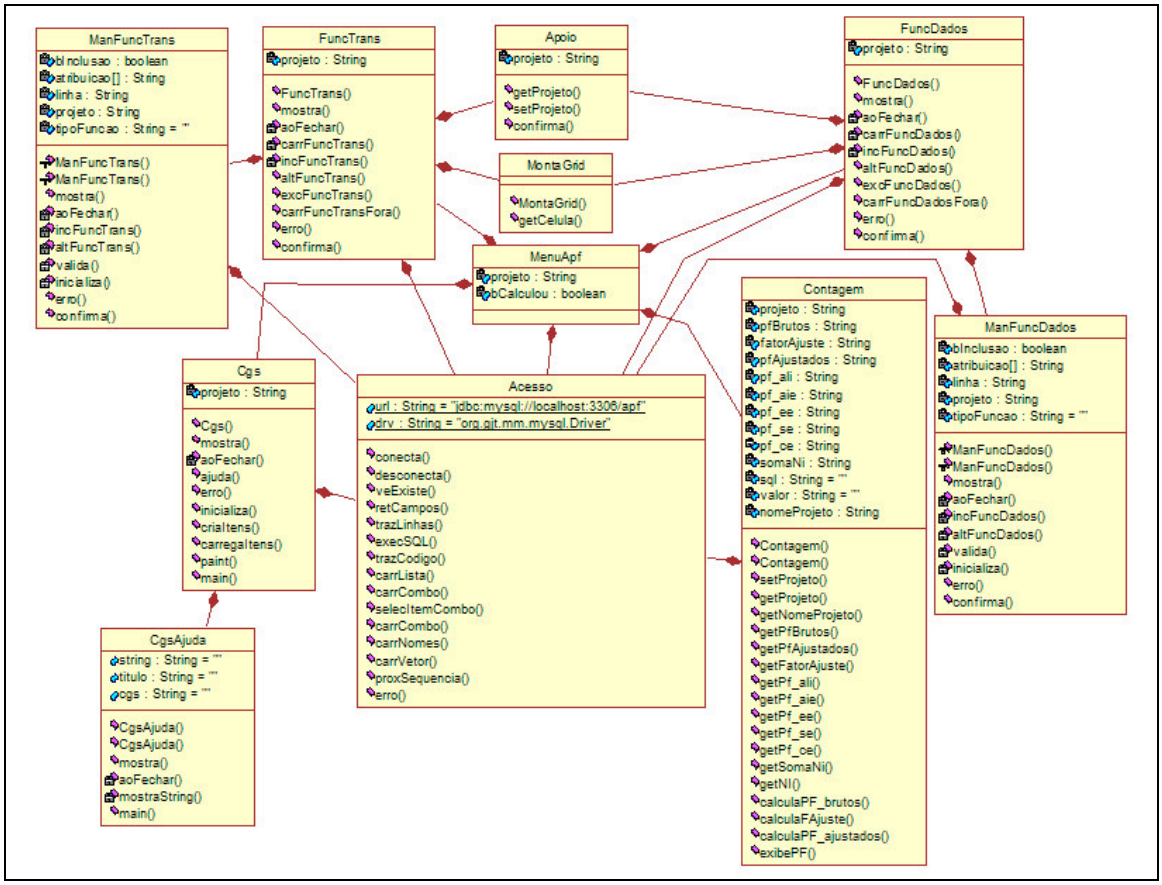


FIGURA 19 – VISÃO DA ANÁLISE DE PONTOS DE FUNÇÃO

5.4. INTERFACE COM O USUÁRIO

No desenvolvimento do sistema, procurou-se estabelecer uma interface amigável com o usuário, tornando seu uso intuitivo. A seguir serão apresentadas algumas telas da ferramenta.

A figura 20 mostra o cadastramento das pessoas que formam a equipe de desenvolvimento. Na tela exibida é possível informar o nome do colaborador, sua atribuição, ou seja, seu papel na equipe (coordenador, analista de sistemas, programador, etc), e sua experiência profissional em anos.

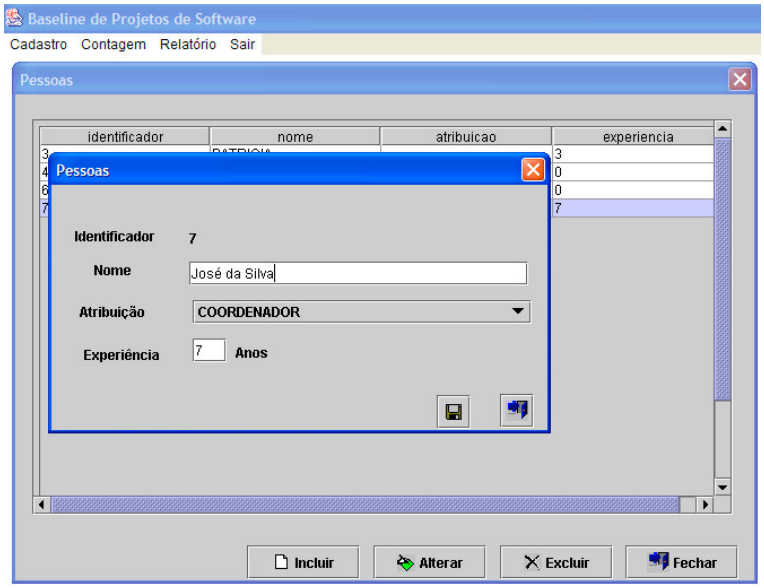


FIGURA 20 – INTERFACE PESSOAS

O ambiente operacional da equipe de desenvolvimento de sistema é apresentado na figura 21. Esta tela possibilita o cadastro das várias combinações existentes entre sistema operacional, SGBD e linguagem de programação.

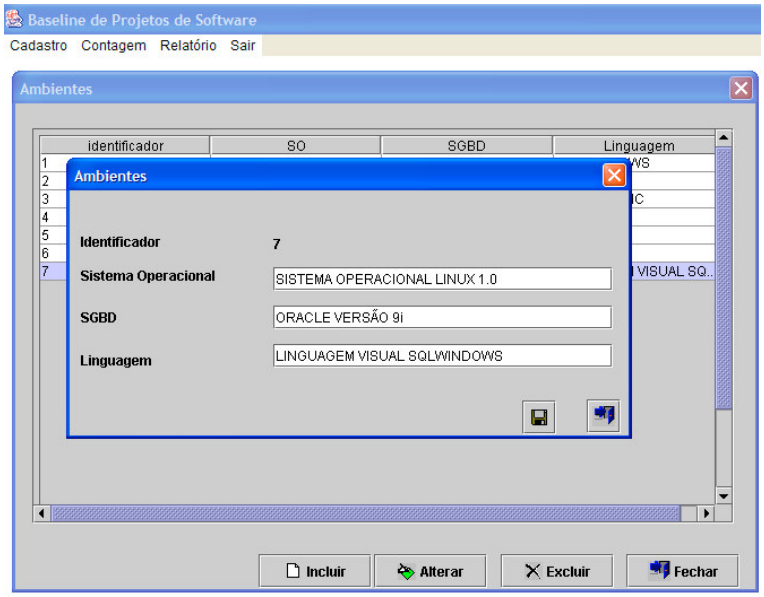


FIGURA 21 – INTERFACE AMBIENTE

O gerenciamento da *baseline* é realizado através da tela de cadastro de projetos apresentada na figura 22. Na referida tela, cada projeto recebe uma identificação única (código) e todos os dados relativos a ele são informados. Pode-se observar que um projeto está inserido nos contextos relacionados a seguir: tipo de aplicação (comercial, científica, COTs, etc), ambiente operacional de desenvolvimento, pessoas envolvidas em seu desenvolvimento, períodos de inicio e término, esforço despendido em homem-mês, custo real do projeto e tamanho em linhas de código (LOCs). Todas estas informações, relacionadas a todos os projetos, constituem a *baseline* de projetos de software mostrada neste trabalho.

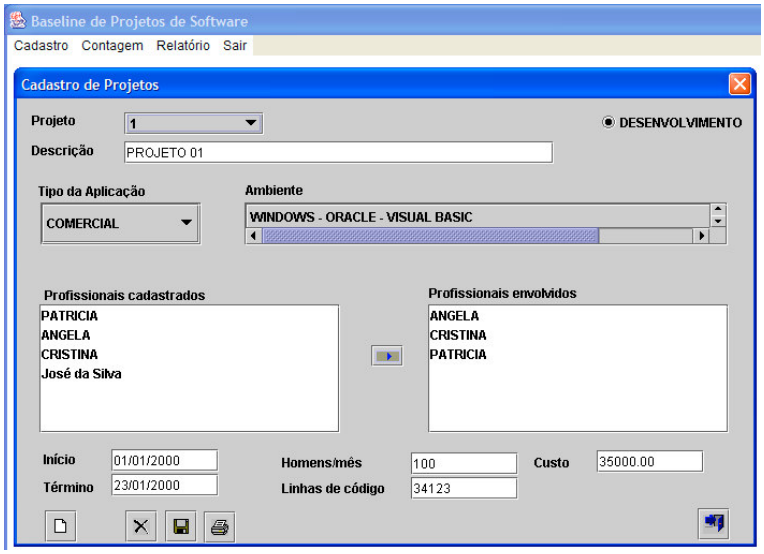


FIGURA 22 – INTERFACE PROJETOS

Para dar suporte à análise de pontos de função, as telas a seguir foram desenvolvidas. As figuras 23, 24, 25 e 26 indicam respectivamente o registro das funções de dados (ALI e AIE), o registro das funções transacionais (EE, SE, CE), o registro das 14 características gerais do sistema e a obtenção de todas as informações a respeito do projeto, inclusive a APF, em forma de relatório.

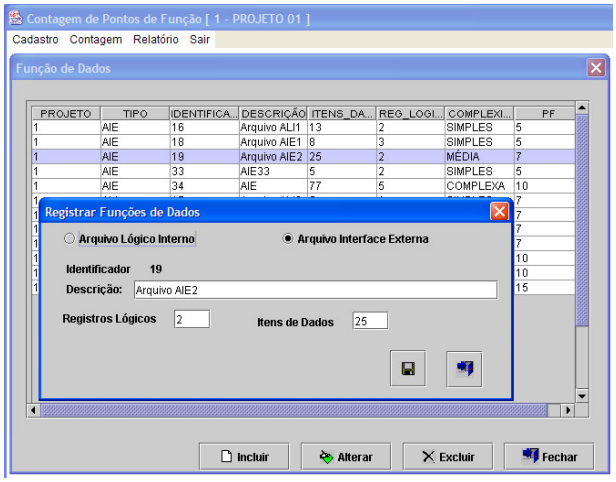


FIGURA 23 – INTERFACE REGISTRAR FUNÇÕES DE DADOS

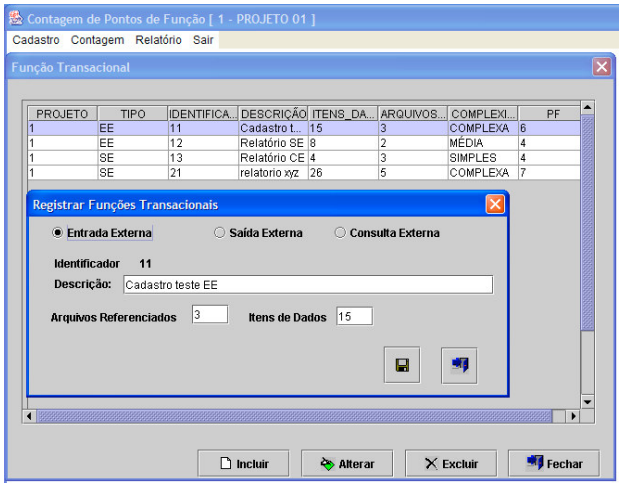
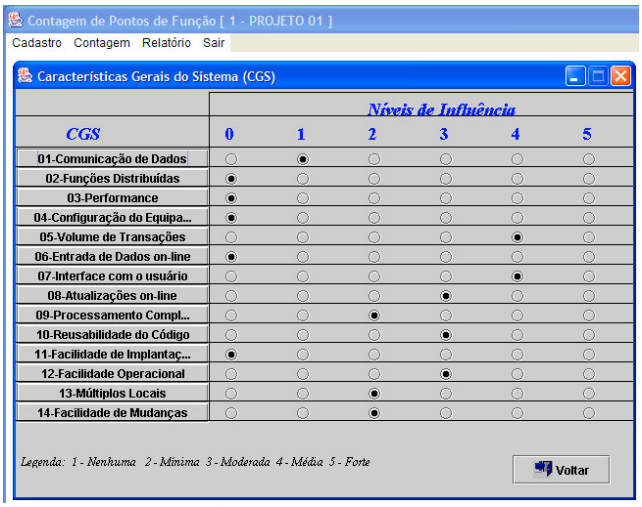


FIGURA 24 – INTERFACE REGISTRAR FUNÇÕES TRANSACIONAIS



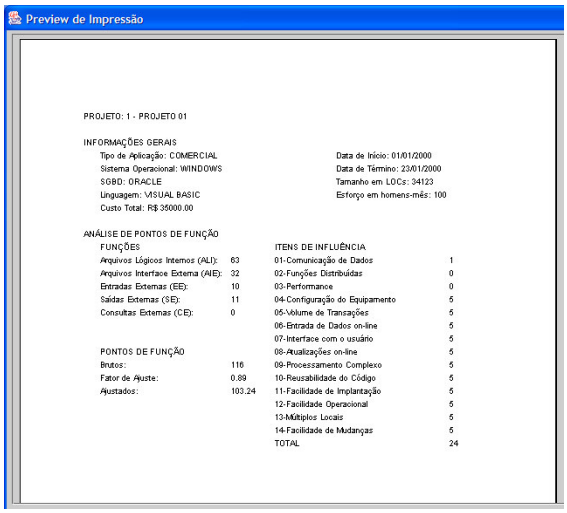


FIGURA 25 – INTERFACE REGISTRAR CARACTERÍSTICAS GERAIS DO SISTEMA

FIGURA 26 – SUMÁRIO DO PROJETO

A ferramenta, concluída, encontra-se em fase de testes de aceitação. Pretende-se disponibilizá-la ao público interessado em métricas de software, para que haja uma efetiva contribuição para melhoria do aplicativo.

6. VALIDAÇÃO DA FERRAMENTA

A ferramenta foi testada com um projeto real de uma empresa de desenvolvimento de software da região de Campinas/SP. Através de vários contatos com a empresa, o projeto, que serviu para o teste da ferramenta foi disponibilizado. O material entregue foi a documentação de um sistema já desenvolvido e uma planilha eletrônica com macros que executam a contagem dos pontos de função.

A empresa foi escolhida por já adotar pontos de função como métrica de tamanho de software. Portanto, com a documentação disponibilizada e a planilha com a contagem dos pontos de função já efetuada, foi possível inserir os dados do projeto na ferramenta e assim comparar os resultados obtidos pela ferramenta com os obtidos pela planilha. Nos próximos parágrafos serão apresentados o projeto e as etapas do teste realizado.

6.1. CARACTERÍSTICAS QUANTITATIVAS E QUALITATIVAS DO PROJETO

O projeto de software disponibilizado é responsável pela gestão de ensaios agronômicos produzidos por uma empresa do setor sucroalcooleiro. Este software denominado Banco de Dados Agronômicos, foi desenvolvido utilizando a linguagem de programação Visual Basic e armazena as informações no Sistema Gerenciador de Banco de Dados ORACLE. O projeto contou com a atuação de dois profissionais, um analista de sistemas e um programador. A empresa não soube informar a quantidade de LOC's, Homens/mês e o custo final do projeto.

6.1.1. CARACTERÍSTICAS GERAIS DO SISTEMA

Com base nos levantamentos dos requisitos efetuados, a empresa constatou os seguintes níveis de influência, conforme mostrado no quadro a seguir:

QUADRO 24 – DEFINIÇÃO DOS NÍVEIS DE INFLUÊNCIA

Características Gerais do Sistema	Nível de Influência
Comunicação de Dados	4
Funções Distribuídas	4
Performance	4
Configuração do Equipamento	1
Volume de Transações	4
Entrada de Dados On-Line	5
Interface com o usuário	5
Atualização On-Line	3
Processamento Complexo	4
Reusabilidade	2
Facilidade de implantação	4
Facilidade Operacional	1
Múltiplos Locais	1
Facilidade de mudanças	2

6.1.2. ITENS DE CONTAGEM DOS PONTOS DE FUNÇÃO

A planilha fornecida contém todas as informações referentes aos Arquivos Lógicos Internos (ALI), Arquivos de Interface Externos (AIE), Entradas Externas (EE), Saídas externas (SE) e consultas externas (CE) necessárias para a contagem dos pontos de função.

6.2. METODOLOGIA APLICADA AO TESTE

Com base nas informações contidas na documentação do sistema, foi criado um projeto na ferramenta contendo informações qualitativas e quantitativas como mostrado na figura a seguir.

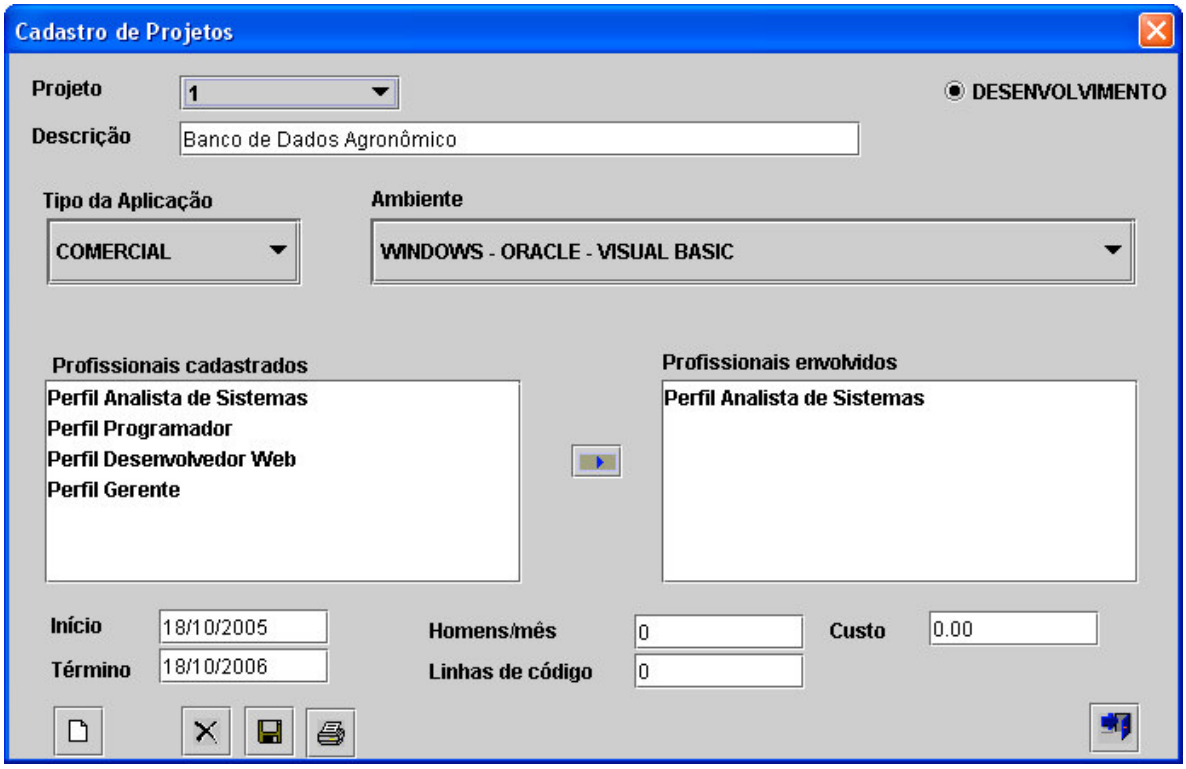


FIGURA 27 – CADASTRO DO PROJETO

Os dados necessários à contagem dos pontos de função foram transcritos da planilha fornecida pela empresa para as telas da ferramenta que efetuam a Análise dos Pontos de Função. A figura 28 ilustra as principais funções cadastradas.

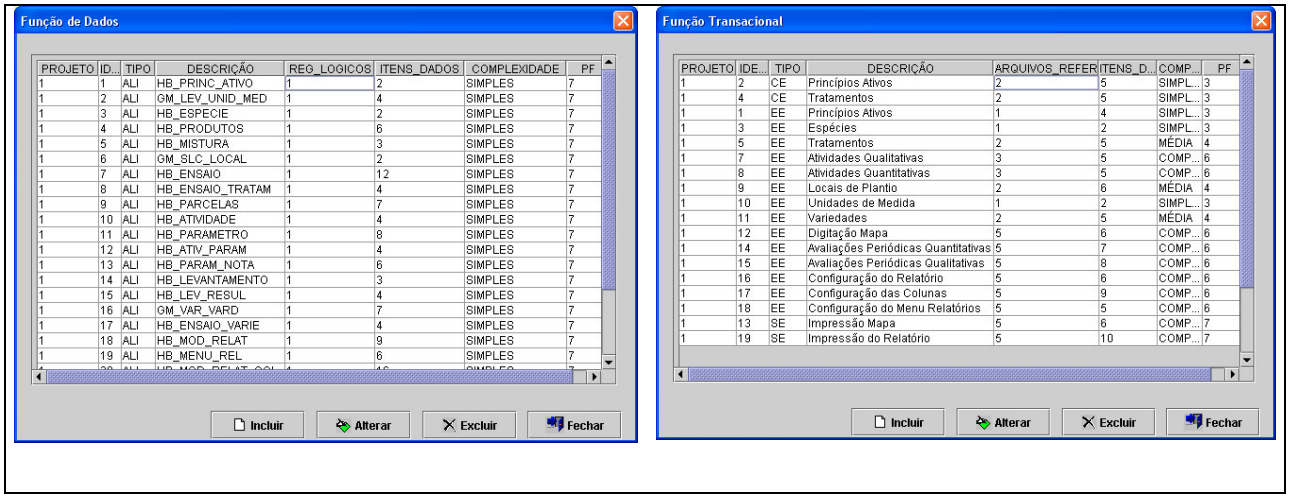


FIGURA 28 – CADASTRO DAS FUNÇÕES DE DADOS E TRANSACIONAIS

Após inserir todas as informações na ferramenta, foi gerado, através do menu “Contagem”, opção “Realizar contagem de Pontos de Função (APF)”, o relatório final com as informações referentes ao projeto, conforme figura a seguir.

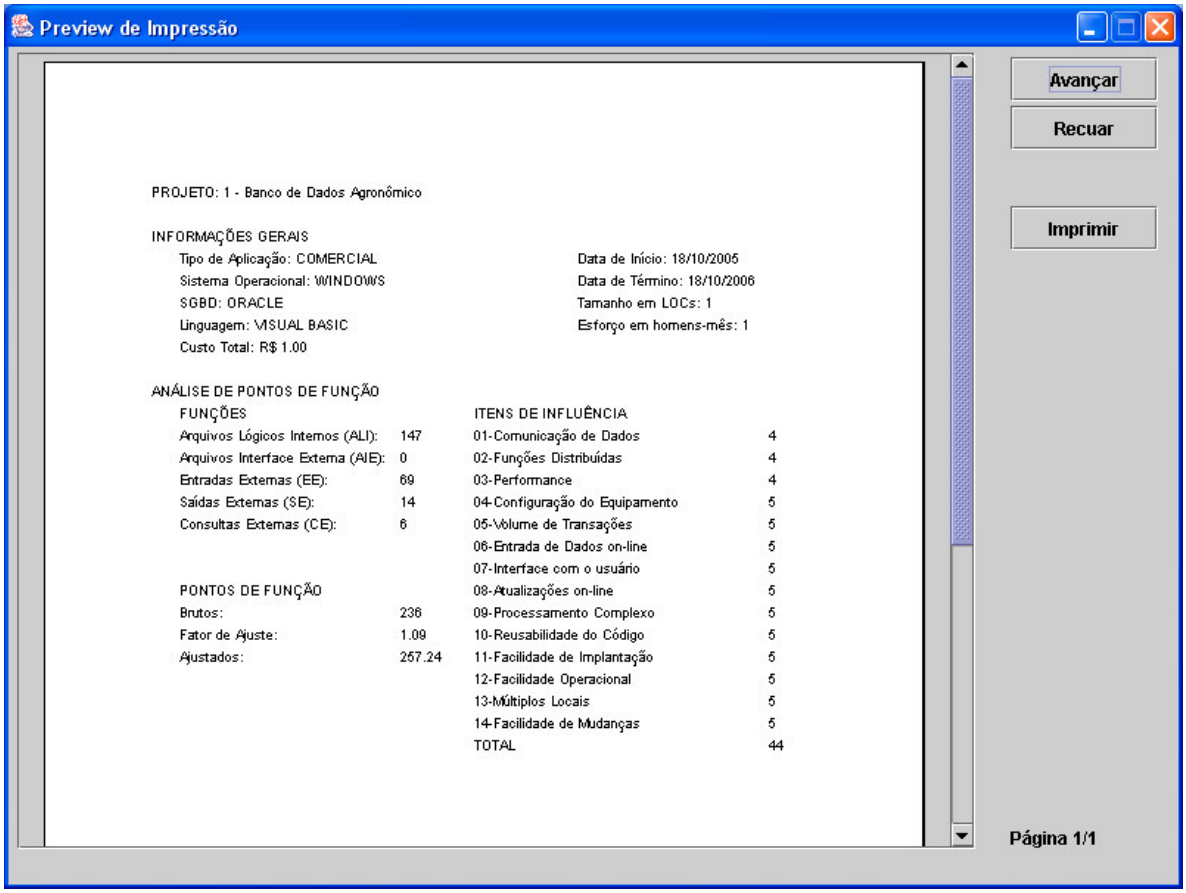
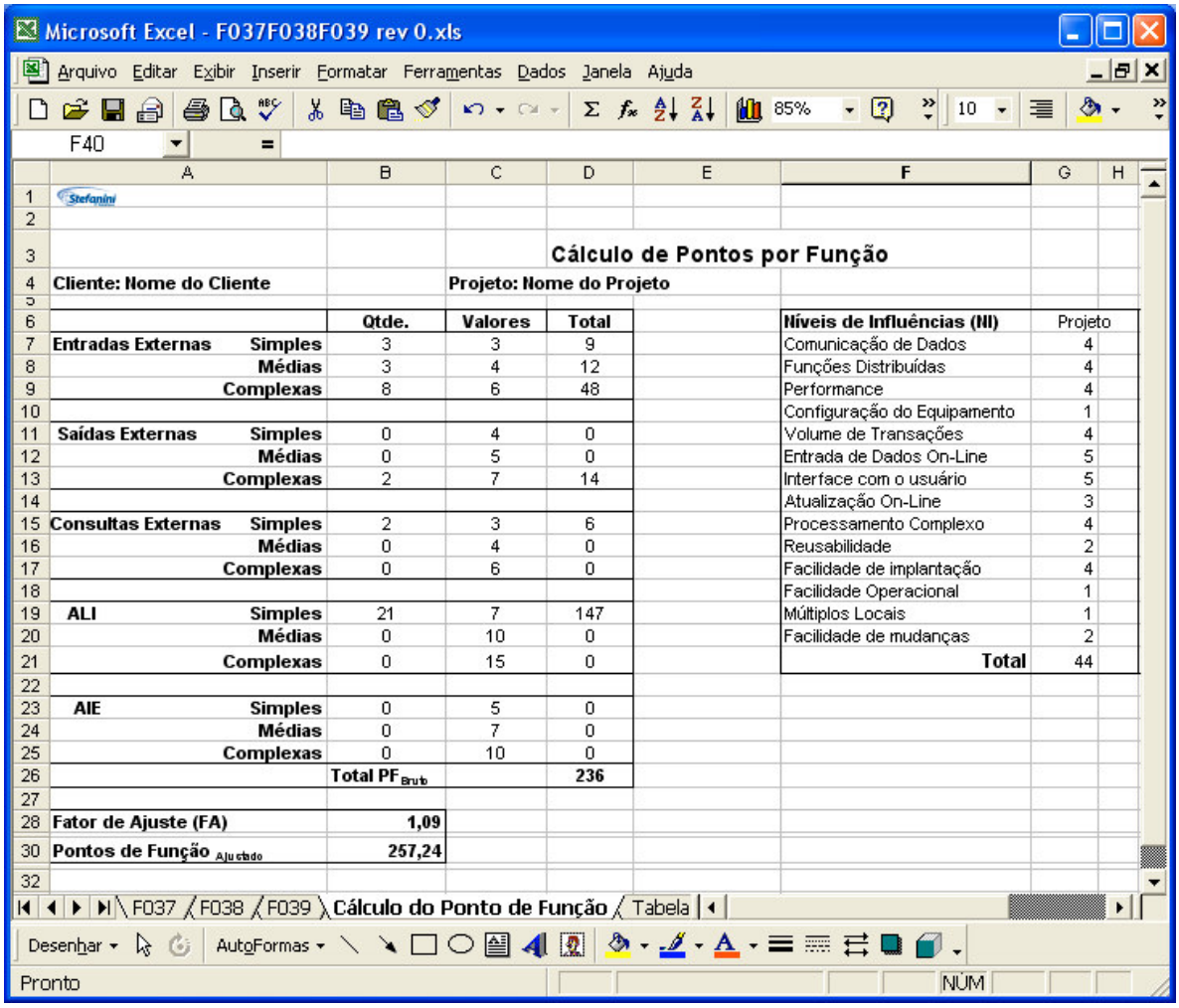


FIGURA 29 – RELATÓRIO FINAL DO PROJETO CADASTRADO

A última etapa do teste consistiu em comparar os resultados obtidos pela planilha que executa a contagem dos pontos de função fornecida pela empresa, com os resultados apresentados pela ferramenta. A figura 30 mostra os resultados da planilha.



Microsoft Excel - F037F038F039 rev 0.xls

Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda

F40

	A	B	C	D	E	F	G	H
1	Stefanini							
2								
3				Cálculo de Pontos por Função				
4	Cliente: Nome do Cliente		Projeto: Nome do Projeto					
5								
6		Qtde.	Valores	Total		Níveis de Influências (NI)	Projeto	
7	Entradas Externas Simples	3	3	9		Comunicação de Dados	4	
8	Médias	3	4	12		Funções Distribuídas	4	
9	Complexas	8	6	48		Performance	4	
10						Configuração do Equipamento	1	
11	Saídas Externas Simples	0	4	0		Volume de Transações	4	
12	Médias	0	5	0		Entrada de Dados On-Line	5	
13	Complexas	2	7	14		Interface com o usuário	5	
14						Atualização On-Line	3	
15	Consultas Externas Simples	2	3	6		Processamento Complexo	4	
16	Médias	0	4	0		Reusabilidade	2	
17	Complexas	0	6	0		Facilidade de implantação	4	
18						Facilidade Operacional	1	
19	ALI Simples	21	7	147		Múltiplos Locais	1	
20	Médias	0	10	0		Facilidade de mudanças	2	
21	Complexas	0	15	0		Total	44	
22								
23	AIE Simples	0	5	0				
24	Médias	0	7	0				
25	Complexas	0	10	0				
26		Total PF_{Bruto}		236				
27								
28	Fator de Ajuste (FA)	1,09						
30	Pontos de Função $_{Ajustado}$	257,24						
32								

F037 F038 F039 Cálculo do Ponto de Função Tabela

Desenhar AutoFormas

Pronto NUM

FIGURA 30 – PLANILHA FONECIDA PELA EMPRESA – RESULTADO APF

6.3. CONCLUSÕES

O teste da ferramenta mostrou-se satisfatório pois permitiu que a maioria dos requisitos levantados na documentação fornecida, fossem incorporados à *baseline*, como por exemplo o ambiente computacional, os profissionais envolvidos e o tipo da aplicação. Outro ponto positivo foi a confirmação da correta execução dos cálculos relativos à contagem dos pontos de função, através da comparação dos resultados gerados pela ferramenta, frente aos apresentados pela planilha exibida na figura 30.

Destaca-se, porém como ponto negativo a realização dos testes não ser executado diretamente pela empresa contatada. Das empresas pesquisadas, a escolhida permitiu apenas a divulgação do projeto, mas não permitiu a instalação e execução da ferramenta em seu local de trabalho devido ao acúmulo de atividades de seus colaboradores.

7. DISCUSSÃO DOS RESULTADOS

Com o desenvolvimento dos atuais módulos da ferramenta, formou-se uma estrutura capaz de armazenar as características quantitativas e qualitativas dos projetos de software e a ferramenta proposta gerencia eficazmente os projetos de uma equipe de desenvolvimento em particular. As equipes de desenvolvimento poderão se beneficiar no que se refere à organização dos projetos. Atualizando a *baseline*, através da ferramenta, com seus respectivos projetos, o usuário obterá um histórico dos desenvolvimentos já realizados, e poderá efetuar pesquisas na base de dados. Outro benefício oferecido pela ferramenta é a contagem dos pontos de função. Através de interface amigável, a ferramenta conduz o usuário a todas as etapas da metodologia de contagem dos pontos de função, até a obtenção do tamanho funcional do projeto.

Para os futuros módulos da ferramenta pretende-se, a partir das informações cadastradas na *baseline*, fazer estimativas. Um dos estudos possíveis seria em relação à análise de regressão. Estabelecendo os pontos de função dos projetos da *baseline* como sendo a variável independente, poderemos utilizar como variável dependente o Esforço (estimativa de esforço requerido dado um número *n* de PF), ou o Custo (estimativa de custo, dado um número *n* de PF), e assim por diante com todas as características presentes na *baseline*.

Este será um dos diferenciais da ferramenta, os modelos de estimativas serão gerados a partir de informações que refletem a realidade de uma organização em particular, diferentemente da maioria das ferramentas que trabalham com estimativas (como visto em seções anteriores) que geram modelos de estimativas baseados em *baseline*s genéricas.

A seguir será apresentado um exemplo de extração e tratamento dos dados coletados da *baseline* mantida com a ferramenta proposta. O quadro a seguir exibe uma possível extração das informações existentes na *baseline*.

QUADRO 25 – INFORMAÇÕES EXTRAÍDAS DA BASELINE - FONTE: MARTINS E CORRÊA (2002)

Projeto	Pessoas	LOC	Custo Total (R$)	Liguagem	Esforço (pm)	Prazo (Meses)	PF	PFA
NUTRI (Sistema de Nutrição)	9	5215	3969	Delphi	40,5	4,5	288	221,76
SR	9	9738	5292	Clipper	54	6	175	113,75
SCR (Sistema de Custo do Refeitório)	10	4417	4410	Delphi	45	4,5	174	149,64
SIAD (Sistema de Análise Descritiva)	4	18629	7840	Delphi	80	20	171	147,06
Projeto para Loja de Confecções [1]	3	1300	588	C++	6	2	135	116,1
Projeto para Loja de Confecções [2]	3	747	588	C++	6	2	83	67,23
Slocadora (Sistema de Locadora)	1	8368	6000	Clipper	12	12	193	154,4
PCL (Projeto Cooperativa de Logistas)	1	4597	3000	Clipper	6	6	139	90,35
PNAD	2	3285	2352	Delphi	24	12	212	137,8
SIAB	10	7481	3920	Delphi	40	4	482	318,12
SISMOB	10	7491	3430	Delphi	35	3,5	155	105,4
Sistema de Reservas de Laboratório	9	9869	3969	Clipper	40,5	4,5	175	133
Projeto para Fábrica de Lajes	1	5083	3000	Clipper	6	6	205	139,4
Projeto para Loja de Auto-Peças	1	10214	4000	Clipper	8	8	211	143,48
Projeto para Editora	1	7277	6000	Clipper	12	12	191	137,52

Com as informações extraídas da *baseline*, é possível gerar estimativas. As estimativas serão obtidas através de equações, utilizando-se as informações disponibilizadas pela *baseline* como variáveis dependentes e independentes. Poderemos, por exemplo, estimar a variável dependente, custo total, em função da variável independente, tamanho em linhas de código (LOC). A figura 31 ilustra a estimativa custo total em função de linhas de código.

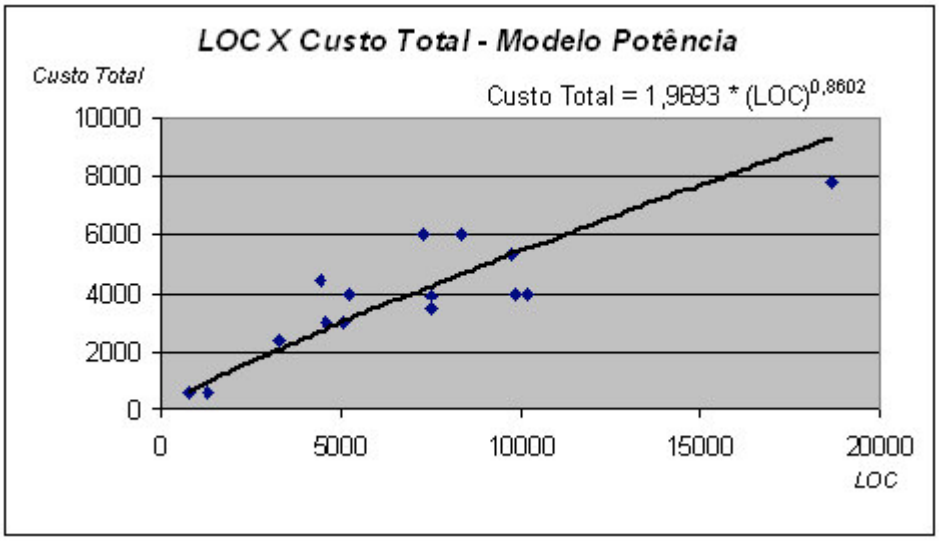


FIGURA 31 – CUSTO TOTAL EM FUNÇÃO DE LINHAS DE CÓDIGO (LOC) A PARTIR DOS 15 PROJETOS ANALISADOS

Na figura 32 é apresentada outra estimativa, pretende-se obter o prazo estimado, a partir do custo total.

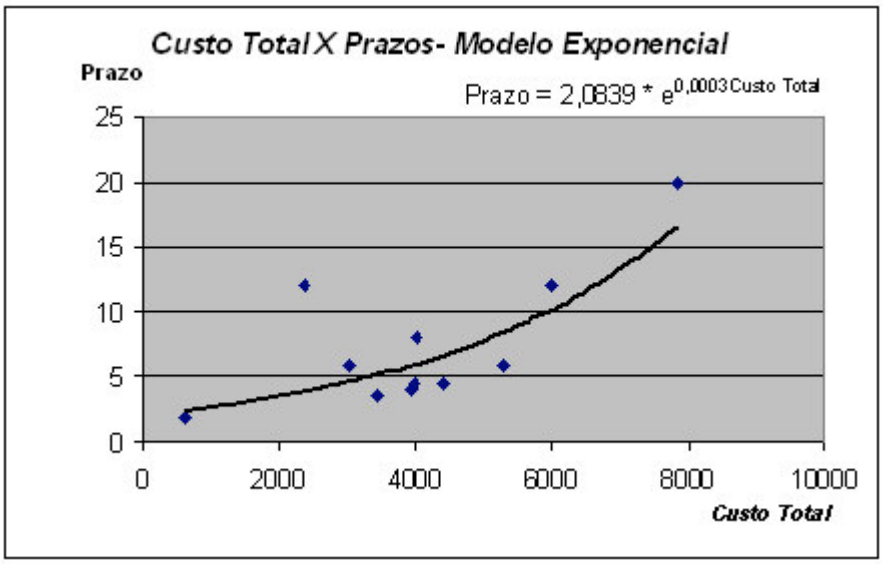


FIGURA 32 – PRAZO EM FUNÇÃO DO CUSTO TOTAL A PARTIR DOS 15 PROJETOS ANALISADOS

Pretende-se portanto que, no futuro, a ferramenta sofra atualizações que possibilitem a geração de modelos de estimativas (regressões lineares, exponenciais, logarítmicas, etc) automaticamente, a partir dos dados da *baseline*.

8. CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de uma ferramenta capaz de manter uma *baseline* de projetos de software particularizada por equipe de desenvolvimento. Para alcançar o objetivo proposto, foram realizadas revisões bibliográficas sobre *baselines* de projetos de software e métricas de tamanho de software. Através destas pesquisas, necessárias para o embasamento deste trabalho, evidenciou-se a importância do tamanho funcional como métrica de tamanho de software. Reconhecendo tal importância, foi acrescido à ferramenta, a metodologia de contagem de pontos de função, segundo normas e padrões estabelecidos pelo IFPUG.

No capítulo 2 destacou-se os conceitos, sob a ótica de vários autores, referentes à *baselines* de projetos de software, assim como as características qualitativas e quantitativas que as compõem. Uma das características da baseline é o tamanho do software. Devido a vários aspectos discorridos no capítulo 3, optou-se como métrica de tamanho de software, a análise de pontos de função. O capítulo 3 descreve a metodologia de contagem de pontos de função incorporada à ferramenta. As diversas ferramentas correlatas, existentes no mercado, são brevemente demonstradas no capítulo 4. A metodologia empregada no desenvolvimento e algumas das interfaces da ferramenta são apresentadas no capítulo 5.

Segundo estudos realizados, existem evidências de que *baselines* particularizadas tendem a produzir estimativas mais precisas comparadas a *baselines* genéricas, quando usada para tais propósitos. Destaca-se portanto a importância da ferramenta, que é justamente gerenciar *baselines* particularizadas por equipe de desenvolvimento. Através da ferramenta, é possível manter o histórico dos projetos desenvolvidos por equipe ou organização, diferentemente de outras ferramentas que gerenciam *baselines*

genéricas.

O tamanho funcional representa um aspecto importante no contexto da baseline. Pode-se utilizar pontos de função para estimar outros valores, como prazo, custo e esforço. Utilizando-se por exemplo análise de regressão, pode-se obter uma equação que expresse a estimativa de custo de um novo projeto, em função de seu tamanho funcional, ou o prazo de execução versus PF, ou esforço versus PF, enfim, pode-se cruzar pontos de função com quaisquer características presentes na *baseline*.

Pretende-se como melhorias futuras da ferramenta, que os modelos de estimativas sejam gerados automaticamente a partir dos dados mantidos na *baseline*. Estes modelos de estimativas refletirão dados particularizados da organização.

Referências

ABRAN, Alain & ROBILLANRD, Pierre N. – **Reability of Function Points Productivity Model For Enhancement Projects (A Field Study)** – 1063-6773/93 - 1993 IEEE.

ALBRECHT, A.J. & GAFFNEY, J.E. – **Software function, source lines of code, and development effort prediction: A software science validation** – IEEE Trans. Software Eng. vol. SE-9, no 6,pp639-648 – jun 1983.

BRAGA, Antônio – **Análise de Pontos de Função** – Rio de Janeiro: Infobook, 1996.

BRIAND, Lionel C., BASILI, Vitor R., THOMAS, William M. – **A pattern recognition approach for software engineering data analysis** – IEEE Trans. Software Eng. vol. 18 , no 11, - November 1992.

CANDÉAS, Alcione J. & LOPES, Cleuton C. C. – **ESTIMATIVA: Uma Ferramenta para Agilizar o Dimensionamento de Projetos no SERPRO com base na metodologia de Análise de Pontos por Função** – XIII Simpósio Brasileiro de Engenharia de Software – Caderno de Ferramentas – Florianópolis, 1999. Disponível em <www.serpro.gov.br>. Acesso em: 6 abr. 2003.

DEKKERS, Carol A. – **Pontos de Função – O que é um ponto de função?** – QAI Journal, Dezembro de 1998 – QUALITY PLUS TECHNOLOGIES – Disponível em: <www.qualityplustech.com>. Acesso em: 28 Fev. 2003.

FELTON, Norman E. & PFLEEGER, Shari Lawrence – **Software Metrics: A Rigorous & Practical Approach** – PWS Publishing Company, 1997.

FENTON, Norman & MARSH, William & NEIL, Martin & CATES Patrick & FOREY, Simon & TAILOR, Manesh – **Making Resource Decision for Software Projects** – 26th International Conference on Software Engineering - 2004 (ICSE'04) – 0270-5257/04 IEEE, 2004.

FUJIWARA, Fumiko & GOTO, Takushi & ARAKI, Sadao - **Examples of Applying Software Estimate Tool** - O-8186-8368-6/98 IEEE, 1998.

GARMUS D., HERRON D. – **Function Point Analysis – Measurement Practices for Successful Software Projects** – Addison-Wesley information technology series, 2001.

GARMUS D., HERRON D. – **Measuring the Software Process: A Practical guide to Functional Measurements** – Prentice-Hall PTR, 1996.

HUMPHREY, Watts S. & SINGPURWALLA, Nozer D. – **Predicting (individual) software productivity** – IEEE TRANSACTIONS ON SOFWARE ENGINEERING, VOL 17, NR.2 – FEBRUARY, 1991.

IFPUG – International Function Points Users Group – **Counting Practices Manual** – Release 4.1.1. April, 2000.

JONES, Capers – **Function points: A new way of looking at tools** – Software Productivity Research – August, 1994. Disponível em: <www.spr.com/articles>. Acesso em: 14 Fev. 2003.

JONES, Capers – **Software Portfolio Analysis with Function Point Metrics** – Software Productivity Research - June 18, 1999 – Disponível Em:<www.spr.com/articles>. Acesso em: 16 Abr. 2003.

KAN, Stephen H. –**Metrics and Models in Software Quality Engineering – Second Edition** – Addison-Wesley, 2003.

KEMERER, Chris F. – **Reliability of Function Points Measurement: A Field Experiment** – Communications of the ACM, Vol.36, Nr2, February 1993.

KITCHENHAM, B.A. & KANSALA, K. – **Inter-item Correlations among Function Points** – Proceedings of the IEEE Software Metrics Symposium, Baltimore, MD, IEEE Computer Society Press, 1993.

KITCHENHAM, Barbara – **The Problem with Function Points** – Counterpoint-IEEE SOFTWARE, 1998.

MARTINS, L. E. G. e CORRÊA, A. M. C. J. – **Análise Comparativa entre Modelos de Estimativas de Variáveis de Projeto de Software: Análise de**

Pontos de Função, COCOMO e Modelos Estatísticos - Relatório Científico Final - FAP-UNIMEP, Setembro/2002.

MATSON, Jack E. & BARRET, Bruce E. & MELLICHAMP, Joseph M. – **Software Development Cost Estimation Using Function Points** – IEEE TRANSACTIONS ON SOFWARE ENGINEERING, VOL 20, NR.4 – APRIL, 1994.

MOHAGHEGHI, Parastoo & ANDA, Bente & CONRADI, Reidar – **Effort Estimation of Uses Cases for Incremental Large-Scale Software Development** – ACM 1-58113-963-2 – May, 2005.

PRESSMAN, Roger S. – **Engenharia de Software** - MAKRON Books, 1995.

PRESSMAN, Roger S. – **Software Engineering: A Practitioner's Approach – Fifth Edition** - McGraw-Hill Companies, 2001.

SPR – Software Productivity Resource – **KnowledgePLAN - a guide**. Disponível em:<www.spr.com>. Acesso em: 10 Out. 2005.

STRIKE, Kevin & EMAM, Khaled El & MADHAVJI, Nazim – **Software cost estimation with Incomplete data** – IEEE TRANSACTIONS ON SOFWARE ENGINEERING, VOL 27, NR.10 – OCTOBER, 2001.

SYMONS, Charles R. – **Function Point Analysis: Difficulties and Improvements** - IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14, NO. I, JANUARY 1988.

VERNER, June and TATE Graham. – **A Software Size Model** - IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 18, NO. 4, APRIL 1992.