Uma simplificação da técnica análise de pontos de função para estimar tamanho de aplicativos web[1]

Edilson José Davoglio Cândido

Orientadora: ***Profa. Dra. Rosely Sanches***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Outubro/2004

[1] Este trabalho contou com o apoio financeiro do CNPq.

A Comissão Julgadora:

Profa. Dra. Rosely Sanches _____

Profa. Dra. Ana Cristina Rouiller _____

Profa. Dra. Renata Pontin de Mattos Fortes _____

Agradecimentos

À minha orientadora, Profª. Drª. Rosely Sanches, pela dedicação, paciência e confiança depositada, que tanto contribuíram para o meu crescimento intelectual e pessoal.

Aos professores Dr. José Carlos Maldonado e Drª. Renata Pontin de Mattos Fortes pelos valiosos conselhos no exame de qualificação.

Ao Prof. Dr. Dorival Leão Pinto Junior por toda a paciência e disponibilidade para as explicações estatísticas.

Aos meus pais, por todo o apoio e incentivo.

Ao meu irmão Hélio, por ter me ajudado durante todo esse tempo em São Carlos.

À Ana, Beth e Laura, por toda a atenção dedicada.

A todos os meus amigos.

A Deus.

"What is not measurable make measurable"

Galileo Galilei

Resumo

A estimativa de tamanho de software é fundamental para determinar as estimativas de esforço e de tempo de desenvolvimento necessários à construção de um software, e os aplicativos desenvolvidos para web não são uma exceção a essa premissa. Neste trabalho é apresentada uma simplificação da contagem detalhada de pontos de função promovida pelo IFPUG (International Function Point Users Group) utilizando como princípio as idéias de simplificação sugeridas pela NESMA (Netherlands Software Metrics Association) para estimar sistemas de informação administrativos, além da apresentação dos requisitos de uma ferramenta para apoiar o processo de contagem dos pontos de função e determinar a produtividade. Em um estudo empírico foram analisadas vinte aplicações web, e as estimativas obtidas através do método simplificado ficaram bastante próximas das realizadas utilizando-se o método detalhado do IFPUG. Baseado nesses resultados, foi possível estabelecer um método de estimativa de tamanho simplificado para aplicativos web, levando em consideração as características de desenvolvimento da empresa estudada.

Abstract

Software size estimation is a key factor to determine the amount of time and effort needed to develop software systems, and the web applications are no exception. In this master dissertation a simplified way of the IFPUG (International Function Point Users Group) function points based on the simplification ideas suggested by NESMA (Netherlands Software Metrics Association) to estimate size of management information systems and the requisites of a software to support the counting and determine the team productivity are presented. In an empirical study, twenty web applications were analyzed. The estimates using the simplified method were close to the ones using the IFPUG detailed method. Based on the results, it was possible to establish a simplified method to estimate the size of web applications according to the development characteristics of the studied company.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO

1

Introdução

1.1 Contexto

Neste início de século XXI, fatores como a globalização da economia e a maior competitividade do mercado têm gerado inúmeros desafios para as empresas. No caso específico das empresas relacionadas com desenvolvimento de software, construir tais sistemas em tempo hábil, com custos razoáveis e qualidade adequada tornou-se fundamental.

Para que essas empresas possam alcançar tais objetivos, é preciso que elas tenham uma gestão efetiva de seus processos de software, focalizando: pessoas, produto, processo e projeto [56]. Sob a óptica do projeto, é necessário que ele seja planejado através de um conjunto de atividades dentre as quais as *estimativas* podem ser consideradas fundamentais, pelo fato de fornecerem um guia para as demais atividades [27].

As estimativas relacionam-se com a tentativa de determinar quanto esforço, recursos e tempo de desenvolvimento serão necessários para construir um software. Entretanto, a solução de tais problemas é complexa para ser considerada como um todo, o que leva a uma decomposição dos mesmos, no sentido de caracterizá-los novamente em um conjunto de problemas menores.

Dessa forma, o processo de estimativa é constituído de uma série de procedimentos, os quais são apresentados na figura 1.1:

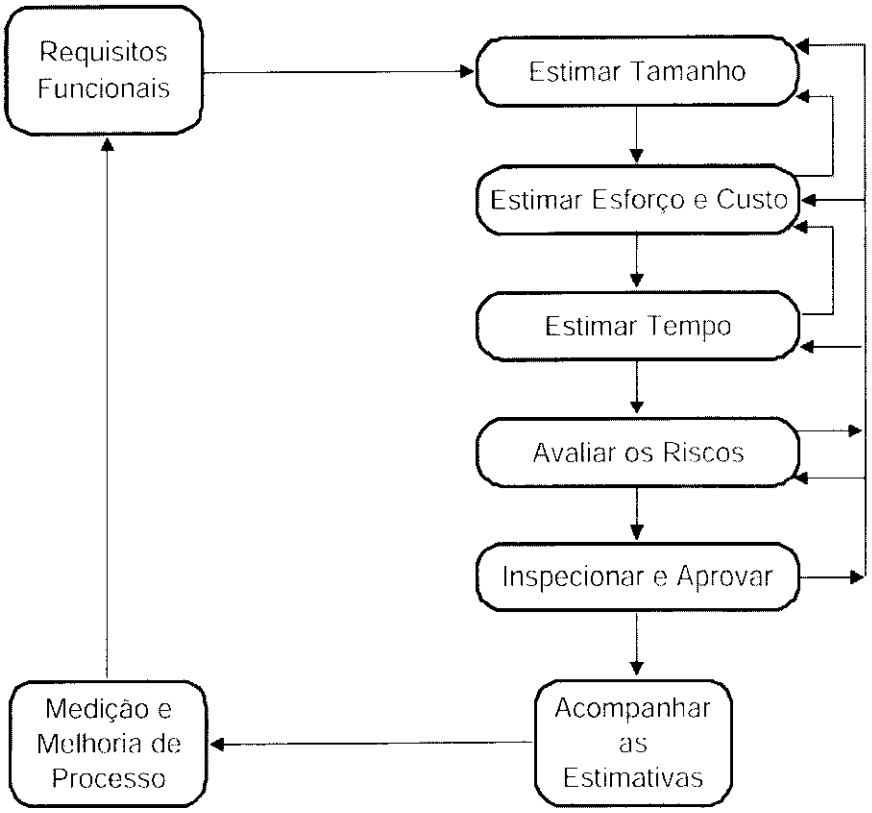


Figura 1.1: Procedimentos do Processo de Estimativa de Software [27]

A estimativa de tamanho (dimensionamento) do software é o primeiro procedimento do processo de estimativa, a partir do qual são feitos o planejamento do trabalho e as estimativas subseqüentes do esforço requerido [45]. Existem várias técnicas que determinam o tamanho de um software, e uma das mais conhecidas e utilizadas atualmente é a *análise de pontos de função*.

Proposta inicialmente por Albrecht [43], a análise de pontos de função quantifica o tamanho de um software através da avaliação das funcionalidades fornecidas, sob o ponto de vista do usuário. Atualmente, o *International Function Point Users Group* (IFPUG) e o *Netherlands Software Metrics Association* (NESMA) são as organizações mais reconhecidas na promoção e estímulo das estimativas de tamanho de software através dos pontos de função.

É amplamente aceito que as estimativas de tamanho de software são fundamentais para a determinação dos custos do projeto de software [8], [23], [34], [36] (no sentido de esforço e tempo de desenvolvimento necessários a construção de um software). Padrões de modelos de referência para qualidade do processo de software tais como o SW CMMI (áreas de processo relacionadas com planejamento e estimativas), o SPICE (processos fundamentais), e a Norma

ISO 12207 (processos organizacionais), também relatam a importância da elaboração de estimativas de tamanho de software como uma das atividades de planejamento do projeto. Os aplicativos web não são uma exceção a essas premissas.

Entretanto, segundo pesquisa do Ministério da Ciência e Tecnologia, no Brasil, apenas cerca de 29% das empresas realizam estimativas de tamanho de software [13] e apesar de não haver uma pesquisa para o quadro das empresas que desenvolvem exclusivamente aplicativos web, a análise do número citado em conjunto com o fato de que 45% das empresas pesquisadas desenvolvem aplicativos web [14], indica que essas empresas se coloquem em situação parecida.

Não há um estudo específico que identifique as causas da não realização de estimativas de tamanho, mas o fato da execução dos métodos para determinar a estimativa de tamanho não serem triviais pode ser uma das possíveis causas.

1.2 Objetivo

O desenvolvimento de um método simplificado para estimar tamanho de aplicativos web poderia colaborar para a mudança do quadro apresentado anteriormente. Assim, o objetivo deste trabalho é apresentar um método simples, de fácil execução, capaz de colaborar para a realização de estimativas de tamanho de software confiáveis, incentivar a utilização das estimativas de tamanho, e conseqüentemente, colaborar para a obtenção de qualidade no processo de desenvolvimento de software voltado para web.

1.3 Metodologia de Pesquisa

A pesquisa se caracteriza como sendo do *tipo exploratória*, pois tem como objetivo proporcionar uma visão geral sobre as estimativas de tamanho de software, e também apresentar uma hipótese na qual é feita a tentativa de simplificação de um método de cálculo de pontos de função para o domínio de aplicações web.

Os métodos de procedimento de pesquisa são *pesquisa bibliográfica* e *pesquisa estudo de caso*. A pesquisa bibliográfica, desenvolvida a partir de materiais já elaborados, tais como livros e artigos científicos, tem como finalidade apresentar o que foi produzido sobre estimativas de tamanho de software. A pesquisa estudo de caso, realizada utilizando dados de vinte aplicativos web de uma pequena empresa [15] de desenvolvimento de aplicativos web, objetivou investigar o comportamento dos métodos simplificados elaborados pela NESMA para estimativa de

tamanho de sistemas de informação administrativos, em um ambiente de desenvolvimento web.

Apesar dos métodos simplificados elaborados pela NESMA, que comprovadamente são eficientes para estimativas de tamanho de sistemas de informação administrativos, não terem apresentado para os aplicativos web resultados satisfatórios, auxiliaram na obtenção de um método simplificado, cujos resultados ficaram bastante próximos daqueles encontrados pelo método detalhado do IFPUG.

Além disso, foi efetuada a modelagem de uma ferramenta capaz não apenas de apoiar a contagem dos pontos de função, mas também determinar a produtividade de cada um dos funcionários e o tempo de desenvolvimento necessário para a construção de um aplicativo.

As técnicas de coleta de dados foram *observação simples*, *entrevistas informais* e *documentos*, cujo intuito foi compreender os requisitos e elaborar as contagens dos pontos de função de cada uma das aplicações.

1.4 Organização do Trabalho

Este trabalho está organizado em seis capítulos. Neste capítulo de introdução situa-se o contexto do trabalho e descrevem-se os principais objetivos.

No Capítulo 2 - *Estimativa de Tamanho de Software*, é apresentada uma descrição das estimativas de tamanho de software e uma visão geral das pesquisas relacionadas a esse tópico.

O método de contagem dos pontos de função promovido pelo IFPUG e as formas de contagem que a NESMA desenvolveu são apresentadas no Capítulo 3 - *Procedimento de Contagem dos Pontos de Função*.

No Capítulo 4 - *Estudo de Caso*, são mostrados os resultados obtidos através da contagem dos pontos de função dos aplicativos da empresa utilizando-se os métodos do IFPUG e NESMA, e a forma como o método simplificado foi obtido.

A modelagem de uma ferramenta capaz de apoiar o processo de contagem e a obtenção do esforço necessário para o desenvolvimento de cada aplicativo é descrita no Capítulo 5 - *Apoio Automatizado*.

No Capítulo 6 - *Conclusões e Trabalhos Futuros*, apresentam-se as conclusões e desdobramentos deste trabalho.

CAPÍTULO

2

Estimativa de Tamanho de Software

Neste capítulo apresenta-se uma visão geral da estimativa de tamanho de software e uma descrição dos pontos de função promovidos pelo IFPUG, NESMA e COSMIC, além do método Mark II [1].

2.1 Visão Geral

A estimativa de tamanho de software é considerada uma tarefa fundamental do gerenciamento de software. A partir dela, é possível predizer as estimativas de esforço e tempo de desenvolvimento, que auxiliam nas tomadas de decisões durante o processo de desenvolvimento do software [27], [45]. As estimativas baseadas em linhas de código (LOC - *Lines Of Code*) ou nas funcionalidades que o software apresenta são duas medidas bastante utilizadas para determinar o tamanho de um software.

As linhas de código são medidas diretas que podem ser facilmente contadas e manipuladas [54]. Há uma grande variedade de formas para se calcular as linhas de código. Jones [8] sugere onze possíveis variações para a contagem das linhas de código de um programa. Fenton et al. [32] apresenta algumas variações e as implicações que elas podem causar. Ainda segundo

[1] Caso não haja familiaridade com os termos da análise de pontos de função, recomenda-se a leitura do capítulo 3 - *Procedimento de Contagem dos Pontos de Função*, antes de se prosseguir neste capítulo.

Fenton et al., a definição mais aceita para se contar as LOC é a da *Hewlett-Packard*, na qual cada programa é considerado uma lista simples de arquivos, e os comentários e linhas em branco são removidos.

Algumas das pesquisas envolvendo linhas de código têm focado a medição de componentes para determinar previsões de tamanho de software. Dolado [1] relata os resultados da validação de um método baseado em componentes para o ambiente restrito de uma linguagem de quarta geração (Informix-4GL) utilizada em aplicações administrativas. O método baseado em componentes foi criado por Verner e Tate [2], e determina o tamanho de um sistema de software a partir do dimensionamento das diferentes partes do sistema.

O estudo desenvolvido por Dolado foi baseado em 42 projetos. A avaliação dos valores encontrados para os componentes de menus foi boa. Já a avaliação dos componentes de módulos de entrada e relatórios não apresentou resultados tão satisfatórios. Apesar disso, o autor relata que encontrou resultados razoáveis.

Bielak [44] descreve a tentativa de se criar linhas de direção para melhorar as estimativas de tamanho de software, estudando como a contagem dos elementos de programação pode ajudar a predizer o tamanho do software. O autor e sua equipe buscaram extrair informações dos dados históricos de um projeto, visando permitir a estimativa de tamanho de trabalhos futuros.

Para isso, foi especulado como a presença ou ausência de diferentes elementos de programação afetava o tamanho do código. Em especial, considerou-se:

- elementos GUI (*Graphical User Interface*) em um componente.
- eventos e mudanças de estado geradas por uma janela, diálogo ou objeto.
- funções membro por componentes.
- componentes de sistema reutilizáveis.

O tamanho de um componente foi determinado utilizando a definição de lista de checagem desenvolvida pelo SEI (*Software Engineering Institute*), acrescida de algumas alterações. Os resultados da investigação mostraram que a contagem e soma desses elementos fornecem algumas informações que podem ser utilizadas para um aumento da confiança nas predições realizadas.

Entretanto, como o próprio autor relata, tal estudo não pretende generalizar a técnica ou aplicá-la em outros ambientes de desenvolvimento. Para os propósitos do pesquisador, as relações encontradas colaboraram para o desenvolvimento de discernimentos capazes de ajudá-lo a examinar futuras predições baseando-se em experiências passadas.

A utilização das LOC para estimativas de tamanho de software gera muita discussão. As críticas baseiam-se na argumentação de que as linhas de código são dependentes da linguagem de programação, e que seu uso na estimativa requer um nível de detalhes que pode ser difícil de alcançar antes que a análise e o projeto tenham sido completados [7], [36].

Já as estimativas de tamanho baseadas nas funcionalidades do aplicativo definem elementos que podem ser contados previamente, refletindo as características funcionais apresentadas pelo software. Os conceitos desse tipo de contagem foram propostos por Albrecht em 1979 [43]. Desde então, eles vêm sendo refinados, principalmente depois da criação de organizações como o IFPUG [42] e a NESMA [53], voltadas exclusivamente para o desenvolvimento da técnica.

No último ano, quatro métodos de estimativa de tamanho de software (IFPUG, NESMA, MarkII, COSMIC-Full Function Points) foram aprovados como padrão ISO para medição de tamanho funcional, sob as seguintes denominações:

- ISO\IEC 20926:2003 (IFPUG).
- ISO\IEC 24570:2003 (NESMA).
- ISO\IEC 20968:2003 (MarkII).
- ISO\IEC 19761:2003 (COSMIC-FFP).

A seguir, os quatro métodos utilizados para estimar o tamanho de um software a partir da identificação e contagem dos pontos de função são apresentados.

2.2 Pontos de Função Segundo o IFPUG

O IFPUG [42] é uma organização fundada em 1986, sem fins lucrativos, que promove a utilização da técnica análise de pontos de função para estimar tamanho de software. Ela mantém um manual de práticas de contagem, o CPM (*Counting Practices Manual*), atualmente na versão 4.1.1 [40], cujo intuito é a padronização do processo de contagem.

A organização oferece uma série de serviços, incluindo:

- Conferência Anual: congrega especialistas e praticantes da técnica análise de pontos de função para trocar experiências com relação aos últimos acontecimentos no campo das métricas de software, bem como promover e encorajar o uso de métricas de processo e de produto de software.

- Seminários e Workshops Educacionais: em conjunto com as conferências, são oferecidas uma variedade de oportunidades de treinamento para complementar as boas práticas de gerenciamento de software. Os *workshops* incluem tópicos relacionados com práticas de contagem dos pontos de função, técnicas de gerenciamento de projeto e estratégias de melhoria de processo.

- Certificação Profissional: através do programa CPFS (*Certified Function Point Specialist*), o IFPUG oferece certificação profissional para os praticantes da técnica análise de pontos de função.

- Comitês de Trabalho: os membros da organização apóiam os avanços das disciplinas de métricas de software participando de forma voluntária de uma variedade de comitês de trabalho. Esses comitês produzem informações vitais sobre os padrões de contagem e estudos do impacto de novas tecnologias no processo de mensuração de software.

Além do escritório nos Estados Unidos, existem representações oficiais do IFPUG em outros países, inclusive no Brasil, cujo trabalho é desempenhado pelo BFPUG (*Brazilian Function Point Users Group*) [3]. Esse grupo disponibiliza artigos, possui um fórum sobre a técnica, e permite a realização do exame de certificação aqui no Brasil. Além do grupo brasileiro, podem ser citadas as seguintes representações:

- ASMA (*Australian Software Metrics Association*) - Austrália
- AEMES (*Asociacion Espanola de Metricas de Software*) - Espanha
- DASMA (*Deutschsprachige Anwendergruppe fur Software Metrik and Aufwanddschatzung*) - Alemanha
- FiSMA (*Finland Software Metrics Association*) - Finlândia
- GUFPI (*Italian Users Group on Function Points*) - Itália
- JFPUG (*Japanese Function Point Users Group*) - Japão
- NASSCOM (*National Association of Software & Service Companies*) - Índia
- UKSMA (*United Kingdom Software Metrics Association*) - Reino Unido

As pesquisas baseadas nos pontos de função promovidos pelo IFPUG, que neste trabalho também são chamados pontos de função detalhados, envolvem críticas e descréditos em relação à técnica, argumentações sobre sua utilidade, tentativas de simplificação e criação de métodos específicos para uma determinada tecnologia de desenvolvimento de software.

Kitchenham [2] diz que os pontos de função não são as medidas simples e sem dificuldades que as pessoas imaginam que elas sejam. Kitchenham aponta como um dos problemas a transformação das complexidades simples, média ou alta em números, não sendo possível definir diferenças entre os valores, ou seja, dizer que 4 é o dobro da quantidade do valor 2. Outro ponto relaciona-se com o fator de ajuste, baseado em uma avaliação subjetiva de quatorze características. Segundo a pesquisadora, não há evidências de que elas melhorem qualquer modelo de previsão de tamanho de software envolvendo os pontos de função.

As análises de Lokan [46] ratificam os argumentos de Kitchenham [2]. Ele conclui, através de um estudo empírico, que as CGS (Características Gerais dos Sistemas) não devem ser utilizadas para determinar o fator de ajuste pelo fato de existirem muitas dúvidas sobre a forma como são determinadas.

Uma outra crítica baseia-se em resultados de estudos que mostram diferenças no valor das contagens quando as mesmas são efetuadas por diferentes pessoas. Os estudos apontam diferenças de 12% [10] e 30% [19].

A resposta às críticas foi dada por Furey [55], em um artigo no qual ele aponta razões para a utilização da técnica. Os pontos de função são perfeitos? Não. Eles são uma métrica útil? Na opinião de Furey, sim. Segundo ele, os pontos de função são consistentes, independentes de tecnologia, repetíveis, e ajudam a normalizar os dados, permitindo comparações.

O foco dos pontos de função é na funcionalidade que o usuário requer. Por exemplo, uma aplicação que consiste em 1.500 pontos de função permanecerá com o mesmo tamanho se escrita em Cobol, C++, ou Smalltalk. Muitas pessoas acham que os pontos de função são dependentes de tecnologia pelo fato do esforço envolvido na implementação de diferentes tecnologias variar. Mas o objetivo dos pontos de função é medir tamanho, não esforço.

O fato dos pontos de função basearem-se em requisitos funcionais permite que eles sejam estimados e contados prematuramente e mais precisamente do que as linhas de código. Outro fato citado por Furey [55] é que, às vezes, as pessoas colocam ênfase demais em uma métrica para fazer tudo, para ser chamada de "bala de prata". Deve-se lembrar que os pontos de função são bons para estimar a funcionalidade dos sistemas. Quando utilizados com outras métricas

são uma poderosa ferramenta para a gerência do desenvolvimento de software.

Pöstion et al. [37] relatam que após a adoção da técnica APF (Análise de Pontos de Função) no departamento de engenharia de programas e sistemas da Siemens da Áustria, foi possível realizar estimativas de esforço baseando-se em tabelas de transformação específicas para o domínio de negócio e a tecnologia utilizada. As análises são realizadas por um especialista na técnica em conjunto com o gerente de projeto, possibilitando uma boa revisão dos requisitos do usuário.

A própria Kitchenham [2] comenta que na maioria dos casos, se os pontos de função forem utilizados com cuidado em uma organização específica, provavelmente serão encontrados poucos problemas. E que seria um erro rejeitar os pontos e voltar às problemáticas linhas de código. Entretanto, ela destaque a necessidade de um refinamento da técnica.

A recente aprovação da técnica promovida pelo IFPUG como um padrão ISO reforça a utilidade da utilização da técnica. Todavia, o padrão considera os procedimentos elaborados pela técnica até a determinação dos pontos de função ajustados. A utilização das CGS tornou-se opcional a partir de 2002 como uma pré-condição para que a técnica se tornasse um padrão ISO. As CGS ainda são um dos pontos mais criticados da técnica devido à variação na interpretação das CGS, bem como a constatação de que algumas delas estão desatualizadas. Devido a isso, este estudo não considera as CGS. As análises e propostas baseiam-se nos pontos de função não ajustados.

Apesar das argumentações contra e a favor da técnica, as pesquisas não se resumiram a somente esses aspectos. Lokan [47] apresenta um estudo sobre as correlações encontradas entre os cinco elementos dos pontos de função (entradas, saídas, consultas, arquivos internos, arquivos externos). O conhecimento de relações entre os elementos pode significar que uma forma mais simples de dimensionar o tamanho é possível, sem a necessidade de utilização do processo completo dos pontos de função [35]. O emprego de listas de checagem sugere erros que podem ocorrer caso alguns tipos de elementos contribuam mais ou menos que o esperado para o total [29].

As correlações entre os cinco elementos dos pontos de função foram investigadas anteriormente por Kitchenham [26] e Jeffery [35]. As conclusões deles concordam em alguns aspectos, mas não em outros. Ambos encontraram entradas, consultas e arquivos internos como significativamente correlacionados entre si. Kitchenham [26] encontrou correlação significativa entre saídas e entradas, consultas e arquivos internos, e Jeffery [35] não. Ele encontrou algumas

correlações significativas envolvendo arquivos externos, e Kitchenham [26] não.

A discordância entre os dois trabalhos motivou Lokan [47] a tentar tornar mais clara a compreensão das relações entre os cinco elementos dos pontos de função. Os projetos analisados por ele vêm do repositório ISBSG (*International Software Benchmarking Standards Group*) [41]. Esse é um repositório público de dados a respeito de projetos de software já completos. Após a análise de 269 projetos foi observada concordância com os trabalhos de Kitchenham [26] e Jeffery [35], nos pontos em que esses trabalhos concordam entre si. A única exceção foi uma correlação muito fraca entre arquivos internos e externos. Nos pontos discordantes entre os trabalhos as observações da pesquisa tendem a concordar com Kitchenham [26].

Outras conclusões encontradas foram que a forma como as correlações se tornam mais fortes e significativas depende da linguagem de programação. As correlações são relativamente raras e fracas em projetos desenvolvidos utilizando linguagens de terceira geração, e mais fortes para projetos desenvolvidos em linguagens de quarta geração. O autor ressalta, todavia, que isso não é uma sugestão de que o comportamento das correlações é causado pela escolha da linguagem. É possível que outros fatores que influenciam a escolha da linguagem também tenham influência na correlação entre os elementos.

As correlações também variaram de acordo com o tipo de projeto. Elas são mais fortes em projetos de desenvolvimento utilizando linguagens de quarta geração, e mais fracas em projetos de manutenção utilizando linguagens de terceira geração.

As pesquisas sobre os pontos de função detalhados relacionam-se também com tentativas de simplificação e criação de métodos específicos para uma determinada tecnologia de desenvolvimento de software.

Kusumoto et al. [25] propuseram um método de dimensionamento de software baseado nos pontos de função detalhados e na ferramenta de análise de requisitos orientados a objetos chamada REQUARIO [31], desenvolvida na empresa Hitachi Ltda. Os objetivos da pesquisa eram dois: propor regras para a contagem dos pontos de função através da especificação de requisitos e desenvolver uma ferramenta baseada nessas regras para apoiar a contagem dos pontos de função.

Na ferramenta, os requisitos são definidos através de conceitos como "*story*", "*scenario*", "*scene*"e "*character*".

- *Story*: é um conjunto de *scenarios*. Representa o negócio como um todo utilizando um fluxo de exemplos.

- *Scenario*: cada fluxo de exemplos do *story* é chamado *scenario*. Um *scenario* consiste de todos os objetos (pessoa/produto) que aparecem no *story* e sua seqüência de comportamento (operação/processamento), mostrando quem envia e quem recebe uma mensagem, e o que é a mensagem. A seqüência de comportamento (os *scenarios*) pode ser dividida em partes, chamadas *scenes*.

- *Character*: é uma pessoa ou um objeto. Ex: em um sistema de biblioteca, os livros, a recepcionista, o usuário.

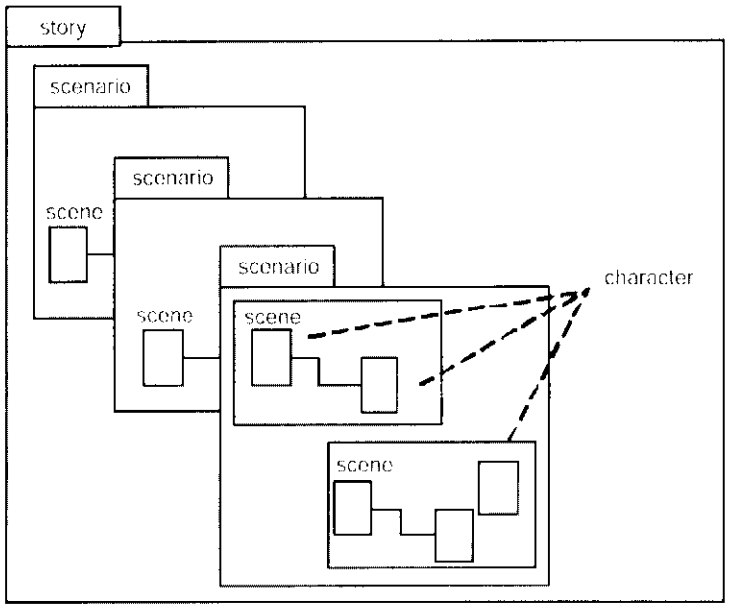


Figura 2.1: Estrutura da Requario [31]

As regras de identificação para as funções de dados basciam-se nos *characters* que representam arquivos ou base de dados. Caso os dados de um *character* possam ser renovados (inclusão, alteração, exclusão) ele é um ALI (Arquivo Lógico Interno). Caso contrário, é um AIE (Arquivo de Interface Externa). Os DETs (*data element type*) são contados através do número de atributos e um RET (*record element type*) é contado para cada *character*.

As funções transacionais são obtidas a partir das *scenes*. Há um conjunto de regras para determinar as entradas, saídas e consultas. Por exemplo, numa *scene* onde um *character* A envia uma mensagem para um *character B*, sendo que A não é um ALI/AIE e B é, se os dados de B são modificados, a função transacional é uma saída externa. Caso contrário, é uma consulta externa.

Após especificar as regras de contagem, uma ferramenta foi desenvolvida utilizando a linguagem C++. Os dados de entrada da ferramenta vêm do arquivo *story* da REQUARIO [31]. A ferramenta foi aplicada a um software da empresa Hitachi e o resultado foi comparado com a contagem efetuada por um especialista da técnica análise de pontos de função. As contagens são mostradas na tabela 2.1.

	Especialista	Ferramenta
Funções de Dados	67	67
Funções Transacionais	117	86
Total de Pontos de Função	184	153

Tabela 2.1: Comparação entre as contagens de um especialista e da ferramenta [25]

Para as funções de dados, os resultados foram idênticos. Já para as funções transacionais houve diferenças. Os autores afirmam que existe a necessidade de um maior detalhamento das regras para determinação das funções transacionais e conseqüentemente, para que os resultados encontrados pela ferramenta e pelo especialista sejam iguais.

Os pesquisadores Kusumoto et al. [24] também examinam a possibilidade de medir os pontos de função de forma automática, a partir do código fonte de um programa escrito na linguagem de programação Java. Primeiro, foram estabelecidas regras para identificação e contagem das funções de dados e transacionais de programas orientados a objetos. Após isso, uma ferramenta utilizando tais regras foi desenvolvida em Java. Por fim, houve uma comparação entre os pontos de função determinados pela ferramenta e os contados por um especialista da técnica análise de pontos de função.

A identificação e contagem dos tipos de funções basearam-se na utilização de informações coletadas dinamicamente durante a execução do programa através de um conjunto de casos de teste elaborados. O conjunto de regras para identificar as funções de dados estabelece o seguinte:

- dentre as classes selecionadas durante a execução do programa, aquelas cujos métodos possuam argumentos são consideradas ALI. As demais são consideradas AIE. Os DETs são definidos como o número de variáveis simples (*int*, *boolean*) e os RETs baseiam-se nas variáveis definidas como tipos de classe.

Para as funções transacionais, deve-se considerar que o método responsável por atualizar ou

referenciar os dados na classe pode ser utilizado para obter a função. Duas regras são estabelecidas:

- regra 1: se uma classe definida pelo usuário é entregue como argumento para uma classe função de dados, ela é uma entrada externa.

- regra 2: antes do valor de retorno ser entregue para a classe de fronteira (classe na qual os métodos são chamados inevitavelmente quando o usuário entra com dados. Exemplo: classes GUI ou classes Java Servlet) algumas classes modificam esse valor. Essas classes são consideradas saídas externas. As funções que não satisfazem as regras 1 ou 2 são consideradas consultas externas.

O número de DETs para as entradas externas é determinado pelo total de argumentos dos métodos chamados pelas classes de fronteira. Para as saídas e consultas, são considerados os valores de retorno dos métodos chamados pelas classes de fronteira. Caso o valor de retorno seja uma classe, conta-se o número de variáveis definidas na classe. O FTR (*file type referenced*) é o número de funções de dados que é referenciado pela função transacional.

Uma vez definidas as regras, foi desenvolvida uma ferramenta para programas escritos na linguagem Java. A ferramenta inclui componentes para:

- análise de sintaxe: analisa o programa e armazena as informações de sintaxe no arquivo de informações de sintaxe.

- execução: executa os casos de teste, armazenando as informações no arquivo de *log* de execução.

- cálculo dos pontos de função: calcula os pontos de função baseando-se nos dados da sintaxe e *log* de execução.

O formato da informação de sintaxe é mostrado pela figura 2.2:

C significa o rótulo do nome da classe e CV significa o rótulo do nome, tipo e *flag* (DETflag) de cada variável de classe. M é o rótulo do nome do método definido na classe e MV é o rótulo do nome, tipo, e *flag* de cada variável de classe definida no método.

A ferramenta foi avaliada num estudo de caso com um aplicativo web. Os resultados das contagens determinadas pela ferramenta e pelo especialista são apresentados na tabela 2.2

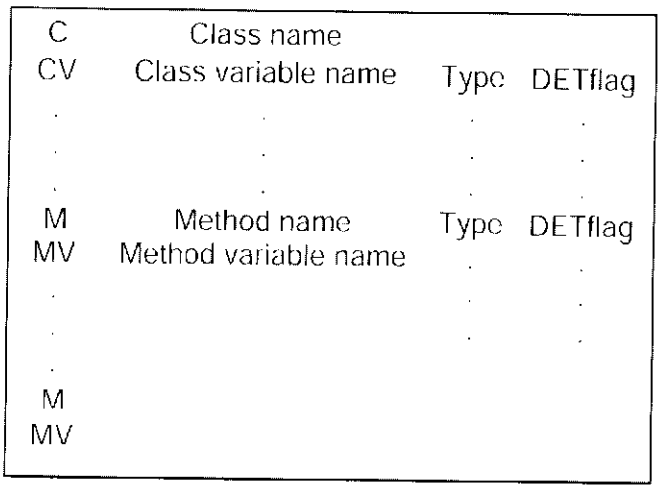


Figura 2.2: Formato do Arquivo de Informação de Sintaxe [24]

		Ferramenta	Especialista
Funções de Dados	ALI	11	4
	AIE	1	7
Funções Transacionais	EE	9	6
	SE	13	14
	CE	0	0
Total de Pontos de Função		174	170

Tabela 2.2: Comparação entre as contagens da ferramenta e de um especialista [24]

O estudo de caso mostrou que a ferramenta encontrou resultados parecidos com os do especialista na identificação do número de funções de dados e transacionais. Entretanto, ela foi incapaz de determinar uma classificação correta das funções encontradas. Isso demonstrou a necessidade de regras mais refinadas para a determinação das funções transacionais, cujo maior problema foi a contagem de funções repetidas.

Uemura et al. [28] apresentam regras detalhadas de medição de pontos de função para especificações de projetos utilizando UML, além do desenvolvimento de uma ferramenta de medição de pontos de função. Como entrada da ferramenta são utilizadas as especificações projetadas pela *Rational Rose*.

Os pontos de função são calculados utilizando diagramas de seqüência e diagramas de classe. A fronteira da contagem é determinada pelos tipos de objetos que aparecem no diagrama de seqüência, ou seja, objetos ator estão fora da fronteira e os outros objetos estão dentro da fronteira.

Uma função de dados é identificada e classificada através da utilização de três passos:

1) Selecionar candidatos à função de dados: os objetos que possuem atributos e trocam dados com objetos que não sejam atores, são candidatos a funções de dados.

2) Determinar o tipo da função: objetos que tem operações as quais mudam os atributos de outros objetos na troca de dados são classificados como ALI. Os outros são classificados como AIE.

3) Determinar a complexidade das funções de dados: os DETs são encontrados contando-se os atributos de uma classe. Caso a classe seja derivada de outra classe, o número de atributos da classe base também deve ser adicionado. Os RETs não podem ser contados a partir dos diagramas de classe ou seqüência. Entretanto, os autores afirmam que segundo suas experiências, quase sempre o número de RETs é igual a 1.

Na contagem das funções transacionais, toda mensagem trocada por um objeto especificado como função de dados é candidata à função transacional, desde que ela possua um argumento.

As funções transacionais são determinadas através de dois passos:

1) Selecionar candidatos à função transacional: listar a seqüência de mensagens nas quais a primeira é enviada pelo objeto ator e a última é recebida pelo objeto não ator no diagrama de seqüência.

2) Determinar o tipo da função transacional: para cada seqüência listada no passo 1, utilizando os padrões a seguir, identificar a função transacional e sua complexidade.

   a) Padrão 1: um objeto ator envia mensagem para uma função de dados. É uma entrada externa.

   b) Padrão 2: uma função de dados envia mensagem para um objeto ator. Caso os argumentos da mensagem sejam iguais aos atributos da função de dados, considera-se uma consulta externa. Caso contrário, uma saída externa.

   c) Padrão 3: um objeto ator envia mensagem para uma função de dados, e uma mensagem é retornada da função de dados para o objeto ator. Caso os argumentos da mensagem 2 sejam iguais aos atributos da função de dados, consideram-se duas consultas externas. Caso contrário, duas saídas externas.

d) Padrão 4: um objeto ator envia mensagem para uma função de dados, e a função de dados envia mensagem para outro objeto ator. Nesse caso, são divididas em duas funções transacionais, utilizando os padrões 1 e 2 para classificá-las.

Uma ferramenta foi implementada em C++. A entrada da ferramenta são os diagramas de classe e de seqüência da *Rational Rose*. Dos diagramas são extraídas as funções, posteriormente avaliadas e calculadas.

A ferramenta foi aplicada em três sistemas: de compras, de vendas e de controle de estoque. Os pontos de função calculados pela ferramenta foram comparados com os calculados por um especialista. Os resultados são mostrados na tabela 2.3.

	Especialista			Ferramenta		
	compras	vendas	controle de estoque	compras	vendas	controle de estoque
Funções de Dados	14	29	26	14	29	26
Funções Transacionais	18	63	36	16	50	33
Total	32	92	62	30	79	59

Tabela 2.3: Resultados das Medições [28]

Os valores encontrados para as funções de dados foram idênticos. Já para as funções transacionais houve diferenças. Essas diferenças foram causadas pelo fato de em vários diagramas de seqüência a ferramenta identificar uma única consulta externa, e o especialista considerar duas consultas externas. Os autores observaram a necessidade de descrições mais detalhadas nos diagramas de seqüência para que a ferramenta possa identificar corretamente as funções transacionais.

Outra pesquisa relacionada com medição de pontos de função a partir dos requisitos expressos em UML é a de Tavares et al. [20]. Seu trabalho foi parte de um esforço cujo objetivo era a obtenção do nível 2 do CMM (*Capability Maturity Model*) pelo SERPRO (Serviço Federal de Processamento de Dados).

Tavares et al. [20] determinam os pontos de função através das informações obtidas do diagrama de classes. As seguintes regras são propostas:

a) Regras para as fronteiras:

1) Considere cada ator humano como um usuário do sistema

2) Considere cada ator não humano, o qual não é um sistema desenvolvido apenas para fornecer funcionalidades para o sistema medido, como uma aplicação externa.

b) Regras para classes:

3) Selecione toda classe como um arquivo lógico. As exceções são aquelas que fazem parte de um relacionamento de agregação/composição.

4) Atributos de classes são candidatos a DETs para arquivos e transações pelos quais são lidos e/ou mantidos.

5) Candidatos a RETs são determinados pelos subgrupos dos arquivos e pelas regras de relacionamento de associação, agregação/composição e herança.

c) Regras para relacionamentos de associação:

6) Selecione a classe associação como um arquivo lógico, considerando para a contagem dos DETs os atributos da classe associação e os atributos que são chaves primárias das classes associadas.

7) Selecione as classes associadas como arquivos lógicos.

d) Regras para relacionamentos de agregação/composição:

8) Uma classe agregada à outra classe é candidata a RET para o arquivo relacionado à classe agregada.

e) Regras para relacionamentos de herança:

9) Uma classe abstrata não é candidata a arquivo lógico. É candidata a RET em cada classe que herda suas propriedades.

10) Subclasses de uma classe concreta são candidatas a arquivo lógico ou RET daquela classe. Se as subclasses não são contadas como arquivo lógico, são RET da superclasse.

f) Candidatos adicionais a arquivos:

11) Se os casos de uso fizeram uso implícito de arquivos lógicos que não estejam representados no diagrama de classes, estes arquivos devem ser incluídos no conjunto de arquivos.

g) Mapeamento das funções transacionais:

12) Selecione todo caso de uso que possui uma relação direta com um ator. Esse caso de uso é candidato a uma ou várias transações.

13) Selecione todo caso de uso que estende um caso de uso selecionado na regra 12. A extensão pode incluir interação com um usuário ou uma aplicação externa.

14) Cada classe mantida e/ou lida por um caso de uso conta como um tipo de FTR para a transação associada, se e somente se, a classe foi identificada como um arquivo.

As regras de 12 a 14 devem ser utilizadas em conjunto com as regras do manual do IFPUG para identificação de entradas externas, saídas externas e consultas externas.

Além dos trabalhos apresentados acima, há também aqueles abordando o paradigma orientado a objetos. Whitmire [1] considera em suas pesquisas cada uma das classes como um arquivo lógico e os métodos enviados através da fronteira da aplicação como funções transacionais. Ele rejeita a existência de arquivos de interface externa.

Schooneveldt [51] classifica as classes como arquivos lógicos e os serviços entregues como funções transacionais. Por sua vez, Teologlou [39] propôs os pontos objeto (PoPs - *Predictive Object Points*), cujo intuito é mensurar a funcionalidade (comportamento do objeto), o nível de complexidade adicionada ao software pela quantidade de comunicação entre os objetos do sistema, e a quantidade de reuso através de herança.

Fetcke et al. [17] propuseram o mapeamento do método de Jacobson et al. [21] para o modelo de pontos de função. Eles tentam demonstrar que os pontos de função são independentes da tecnologia utilizada para a implementação e que podem ser utilizados para o paradigma orientado a objetos.

O mapeamento é formulado como um conjunto de regras e procedimentos que facilitam a contagem dos pontos de função no método OO-Jacobson, também conhecido como OOSE (*Object Oriented Software Engineering*).

O modelo OOSE é dividido em três processos consecutivos: análise, construção e teste. O foco desse trabalho é na fase de análise, que é dividida em análise de requisitos e análise de robustez.

A análise de requisitos é expressa em termos de casos de uso e complementada pelo modelo de domínio de objetos. A análise de robustez estrutura o modelo de casos de uso em um modelo de análise (objetos).

Um conjunto de 15 regras foi definido para possibilitar o mapeamento. São elas:

Atores, usuários e aplicações externas:

1 Admitir cada ator humano como um usuário do sistema.

2 Admitir cada ator não humano que seja um sistema separado não designado para prover funcionalidades somente para o sistema sob consideração, como uma aplicação externa.

3 Rejeitar um ator não humano que seja parte de um sistema básico, tal como um sistema de base de dados relacional ou uma impressora.

Funções transacionais:

4 Selecionar cada caso de uso que tenha uma relação direta com um ator aceito pelas regras 1 ou 2. Esse caso de uso será candidato para uma ou várias transações.

5 Selecionar cada caso de uso que estenda um caso de uso selecionado pela regra 4. A extensão pode incluir interação com um usuário ou uma aplicação externa.

6 Nenhum outro caso de uso será selecionado.

Arquivos:

a) objetos tipados

7 a) Selecionar cada objeto do tipo entidade como um candidato a arquivo lógico, a menos que as regras de 9 a 11 se apliquem.

8 a) Nenhum outro objeto será selecionado.

b) objetos não tipados

7 b) Selecionar cada objeto do domínio como um candidato a arquivo lógico, a menos que as regras de 9 a 11 se apliquem.

8 b) Nenhum outro objeto será selecionado.

Relacionamentos de Agregações

9) Um objeto de domínio ou entidade que é parte de outro objeto não é um candidato a arquivo lógico, mas é candidato a RET.

Relacionamentos de Herança

10) Um objeto abstrato não é candidato a arquivo lógico. É candidato a RET para cada objeto que herda suas propriedades.

11) Subobjetos de objetos concretos são candidatos a arquivos lógicos ou RETs. Se esses subobjetos não são contados como arquivos lógicos, eles são subgrupos opcionais do arquivo relacionado ao seu super-objeto.

12) Se os casos de uso fazem uso implícito de arquivos lógicos que não são representados no modelo de objetos, esses arquivos tem de ser incluídos no conjunto de arquivos.

Fatores de Ponderação

13) Atributos de objetos são candidatos a DET para os arquivos e transações pelos quais são lidos ou mantidos.

14) Candidatos a RET são determinados pelos subgrupos de arquivos e pelas regras de 9 a 11.

15) Cada objeto lido ou mantido por um caso de uso é contado como um FTR para as transações associadas a ele, se e somente se, o objeto tiver sido identificado como arquivo no passo 3.

As regras acima devem ser utilizadas em conjunto com o manual do IFPUG, para identificação e avaliação das funções de dados e transacionais. Essas regras de mapeamento, quando comparadas com outros métodos, apresentam como vantagem o fato de serem baseadas no padrão promovido pelo IFPUG, que é largamente utilizado. Como desvantagens pode-se citar o fato das regras não se aplicarem a outros métodos de modelagem orientada a objetos.

Outro trabalho relacionado com orientação a objetos é o de Caldiera et. al [18]. Nele, é definida uma adaptação dos pontos de função detalhados, chamada pontos de função orientados a objetos, para permitir a medição de projetos de desenvolvimento orientados a objetos.

Caldiera et al. [18] tentam mapear os conceitos de pontos de função para conceitos orientados a objetos, cuja representação utiliza as notações do método OMT [48]. A contagem considera os dados e métodos de uma classe e os conceitos de agregação e herança.

O primeiro passo é a contagem dos arquivos lógicos. Foram criadas quatro formas de se contar os arquivos, definidas pelas diferentes escolhas de como lidar com agregações e generalizações/especializações:

1) Classe Simples: conte cada classe como um arquivo lógico, sem se importar com relacionamentos de agregação ou herança.

2) Agregações: conte cada agregação como um arquivo lógico.

3) Generalização/Especialização: dada uma hierarquia de herança, conte como arquivo lógico cada coleção de classes composta pela superclasse e a subclasse.

4) Misto: combinação das opções 2 e 3.

Nos arquivos lógicos, atributos (*int, string*) são contados como DETs. Uma associação de valor simples também é contada como DET. As associações multivaloradas são contadas como RET.

Os métodos de cada classe do sistema são analisados. Os abstratos não são contados. Os concretos são contados apenas uma vez. A complexidade dos métodos é baseada na tabela de entradas externas do manual do IFPUG [40].

O processo elaborado foi aplicado em oito softwares de telecomunicações, escritos na linguagem C++. Encontrou-se uma grande correlação entre as quatro formas de se identificar os arquivos lógicos, sugerindo que elas são transformações lineares umas das outras e que a escolha de uma ou outra não interfere de forma significativa na estimativa do tamanho do software.

Os autores citam também a eliminação das ambigüidades dos pontos de função detalhados e o fato de se levar em conta, de uma melhor forma, as características do ambiente orientado a objetos.

Já Ram [33] afirma que os pontos de função detalhados não podem ser utilizados para medir a funcionalidade de sistemas orientados a objetos. Em seu trabalho, sugere um procedimento de contagem para medir a funcionalidade de um sistema orientado a objetos durante a fase de projeto a partir da perspectiva de quem desenvolve o software. O principal objetivo é utilizar todas as informações disponíveis durante a fase de projeto orientado a objetos para estimar os OODFP (*Object Oriented Design Function Points*).

O método OODFP é adaptado dos pontos de função detalhados. Nele, um arquivo lógico é uma coleção de elementos de dados que são visíveis a todos os métodos de uma classe. As funções transacionais são os métodos das classes.

Um arquivo lógico dentro da fronteira da aplicação é considerado um ALI. Os que estão fora da fronteira são AIEs. O número de DETs baseia-se nos tipos de dados (*int, char*) e os RETs nas relações (agregação, polimorfismo).

Os métodos são considerados funções transacionais. A complexidade é baseada na tabela de complexidade das entradas externas do método detalhado do IFPUG [40]. Algumas regras devem ser seguidas:

1) Dados e métodos devem ser considerados como entidades simples.

2) Toda agregação implica na contagem de um RET.

3) Polimorfismo é considerado pela definição de uma associação multivalorada e é contado como um FTR.

4) Herança é considerada para estimar a classe derivada, mas não o método derivado.

O método foi comparado com o trabalho de Caldiera [18]. Os cálculos dos aplicativos utilizados foram convertidos em linhas de código [9] e quando comparados com o número de linhas de código dos aplicativos prontos, mostraram um resultado mais eficiente que o método de Caldiera [18].

Por fim, Ruhe [50] apresenta os web objects, um método de estimativa de tamanho para projetos de desenvolvimento web que utiliza para as estimativas de esforço um modelo chamado WEBMO.

Nesse estudo o método é aplicado no contexto de uma pequena empresa australiana de desenvolvimento de software para web. Modelos de estimativa de esforço baseados nos web objects são comparados com modelos baseados nos pontos de função, utilizando OLS (*ordinary least squares regression*).

Testados em dados de vinte aplicações, os web objects apresentaram melhores resultados do que modelos utilizando os pontos de função, quando os dois foram comparados com os resultados encontrados pela estimativa de um método informal baseado em julgamentos de um especialista, que é utilizado pela empresa. Segundo Ruhe [50], a maior limitação dos pontos de função é sua aplicabilidade restrita a sistemas administrativos.

2.3 Pontos de Função Segundo a NESMA

A NESMA [53] foi fundada em 1989, e é o maior grupo de usuários da técnica análise de pontos de função na Europa. A organização mantém seu próprio manual de práticas e contagem, atualmente na versão 2.0. Ela tem como objetivos coletar, manter e desenvolver conhecimento

relacionado com a técnica de pontos de função, promover uma padronização do método, e aumentar a sua utilização.

A NESMA reconhece três tipos de contagem de pontos de função: Detalhada, Estimada, e Indicativa. A descrição desses três tipos de contagem é feita no capítulo 3 (Procedimento de Contagem dos Pontos de Função).

2.4 Mark II

O MarkII ou Mk II foi criado por Charles Symons na década de 80, inspirado na proposta de Albrecht. O método foi desenvolvido pela consultoria KPMG entre 1985 e 1986 como um método proprietário, tornando-se de domínio público posteriormente. Atualmente, a organização responsável por manter o padrão da técnica é a UKSMA (*United Kingdom Software Metrics Association*).

O método MkII é aplicado nos países do Reino Unido e apresenta algumas diferenças quando comparado com o do IFPUG. O MkII considera todos os requisitos como transações lógicas, constituídas de uma entrada, algum tipo de processamento e um componente de saída. Além disso o método produz estimativas muito maiores do que as do IFPUG quando grandes projetos são contados [11].

2.5 COSMIC-FFP

O COSMIC (*Common Software Measurement International Consortium*) foi criado no ano de 1998 com o objetivo de desenvolver um novo método de medição funcional de tamanho de software. O grupo revisou métodos de estimativas de tamanho de software já existentes (IFPUG, MarkII, NESMA) e tentou se basear nas melhores características deles para criar um novo método.

Esse novo método foi publicado em Novembro de 1999 como a versão 2.0 do COSMIC-FFP (COSMIC *Full Function Points*) [12]. O manual está disponível (em inglês, francês, japonês e espanhol) desde então, para acesso público. O COSMIC-FFP é desenvolvido para ser aplicado em softwares dos seguintes domínios [12]:

- Aplicações de sistemas de informação, que tipicamente são necessários para apoiar a administração de negócios, tais como bancos, seguradoras e softwares de contabilidade.

- Aplicações de tempo real, cuja tarefa é acompanhar e controlar eventos que estejam acontecendo em tempo real. O método ainda não é capaz de estimar o tamanho de softwares que tenham como característica algoritmos matemáticos complexos, ou variáveis de processo contínuo tais como imagens de vídeo, encontradas em softwares de jogos de computador.

Rollo [57] afirma que o método COSMIC-FFP é o mais flexível para estimar o tamanho funcional de uma aplicação web. Entretanto, um resultado empírico que apoiasse tal afirmação não foi apresentado.

As outras pesquisas sobre o COSMIC-FFP têm se relacionado com sua aplicabilidade na estimativa de tamanho de softwares de tempo real. Diab et al. [22] propõem uma formalização do método COSMIC-FFP para a linguagem ROOM (*Real-Time Object Oriented Modeling*). Os benefícios dessa formalização são eliminar variações que podem levar a diferentes contagens da mesma especificação, dependendo da interpretação feita pelo avaliador, e automatizar o método para especificações ROOM. Além disso, busca-se prover uma caracterização clara e não ambígua dos conceitos do método, tornando o COSMIC-FFP útil para outras notações orientadas a objeto, como UML.

Raman [30] baseia-se no fato de não existir um modelo de lógica fuzzy para o COSMIC-FFP, para desenvolver um modelo de estimativa de esforço para o Cosmic-FFP através de lógica fuzzy, utilizando o modelo de conjunto fuzzy e o modelo de regressão linear fuzzy.

Por fim, Bootsma [38] apresenta um estudo no qual é descrito como os *Full Functions Points* permitiram estimativas mais consistentes e precisas dos projetos de software de tempo real.

2.6 Considerações Finais

Neste capítulo foi apresentada uma visão geral sobre a estimativa de tamanho de software. No capítulo a seguir, os métodos de contagem dos pontos de função são descritos.

CAPÍTULO

3

Procedimento de Contagem dos Pontos de Função

Neste capítulo são apresentadas as etapas do procedimento de contagem dos pontos de função mantido pelo IFPUG e também as formas de contagem que a NESMA preconiza.

3.1 Contagem dos Pontos de Função Segundo o IFPUG

O procedimento de contagem dos pontos de função elaborado pelo IFPUG está descrito no manual de práticas de contagem, o CPM (*Counting Practices Manual*), atualmente na versão 4.1.1 [40]. As etapas desse procedimento podem ser visualizadas na figura 3.1.

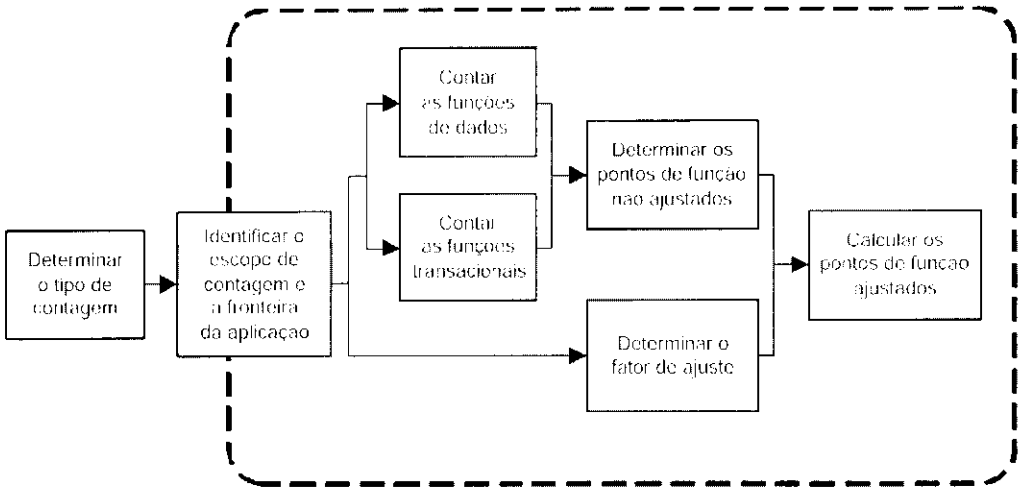


Figura 3.1: Fluxograma do Procedimento de Contagem dos Pontos de Função [40]

Como apresentado na figura 3.1, o procedimento para o cálculo dos pontos de função segundo o IFPUG possui sete etapas, que são descritas nas próximas subseções.

3.1.1 Determinar o Tipo de Contagem

A primeira etapa do procedimento de contagem dos pontos de função elaborado pelo IFPUG consiste em determinar o tipo de contagem que será utilizada. A contagem pode estar relacionada tanto com um projeto de desenvolvimento quanto com um produto já desenvolvido.

Dessa forma, é possível efetuar a contagem de um software que será desenvolvido, de um software já existente, ou das atividades de manutenção da aplicação, classificando o tipo de contagem em uma das seguintes formas:

(1) Projeto de Desenvolvimento

A contagem dos pontos de função de um projeto de desenvolvimento mede a funcionalidade solicitada e entregue ao usuário pela nova aplicação. A identificação das funcionalidades é baseada no processo de definição dos requisitos, que deve utilizar uma terminologia capaz de representar de forma clara para todos os envolvidos no processo as necessidades do negócio.

(2) Projeto de Manutenção

A contagem dos pontos de função de um projeto de manutenção mede as modificações de uma aplicação já existente que adicionam, alteram, ou excluem funcionalidades dessa aplicação.

(3) Aplicação

A contagem dos pontos de função de uma aplicação é associada com um software instalado. Ela é efetuada ao final da contagem dos pontos de função de um projeto de desenvolvimento e também todas as vezes que o projeto de manutenção altera as funcionalidades da aplicação.

3.1.2 Identificar o Escopo de Contagem e a Fronteira da Aplicação

O escopo de contagem define as funcionalidades que serão incluídas em uma determinada contagem dos pontos de função. Ele estabelece se uma ou mais aplicações serão contadas e também identifica quais funções de cada uma das aplicações serão consideradas para fornecer respostas relevantes ao propósito da contagem.

O escopo de contagem:

(1) de um projeto de desenvolvimento: inclui todas as funções que serão desenvolvidas pelas atividades de projeto.

(2) de um projeto de manutenção: inclui todas as funções que estão sendo adicionadas, alteradas ou excluídas.

(3) de uma aplicação: pode incluir, dependendo do propósito (por exemplo, fornecer um pacote de software como solução), apenas as funções que estão sendo utilizadas pelo usuário ou todas as funções entregues.

Já a fronteira da aplicação indica o limite lógico entre o usuário e a aplicação que está sendo medida, identificando as funções internas e externas a ela. O termo usuário se refere a qualquer pessoa ou aplicação que esteja interagindo com a aplicação através do envio ou recebimento de dados.

O manual do IFPUG estabelece algumas regras que devem ser consideradas para a identificação da fronteira da aplicação. São elas:

- A fronteira é determinada com base na visão do usuário. O foco deve estar no que ele pode entender e descrever.
- A fronteira entre aplicações deve ser baseada na separação das funções conforme os processos do negócio, não em considerações tecnológicas.

- A fronteira estabelecida no início do projeto deve estar de acordo com a fronteira estabelecida para a aplicação que será modificada em um projeto de manutenção.

A figura 3.2 mostra um exemplo das fronteiras entre uma aplicação de recursos humanos, e uma outra, responsável pela folha de pagamentos, mantida por outro departamento.

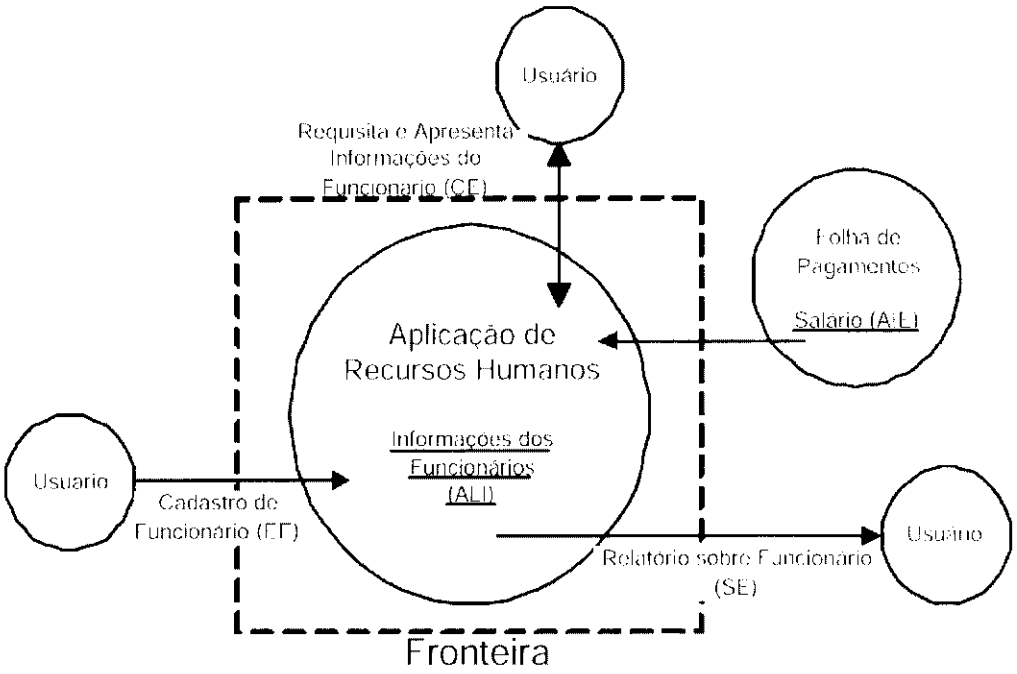


Figura 3.2: Exemplo de Definição de Fronteira [40]

A partir da definição da fronteira entre a aplicação de recursos de humanos e as outras aplicações, é possível determinar quais são as funções de dados e transacionais (apresentadas nas duas próximas subseções) da aplicação e quais interagem com ela, mas são de outras aplicações.

3.1.3 Contar as Funções de Dados

As funções de dados representam as funcionalidades fornecidas pelo sistema ao usuário, para atender suas necessidades de dados. Essas funções podem ser classificadas em Arquivos Lógicos Internos (ALIs) e Arquivos de Interface Externa (AIEs).

Antes de apresentar a definição de ALI e AIE, é importante relatar o significado de alguns termos na Análise de Pontos de Função (APF):

- Usuário: é qualquer pessoa que especifica requisitos funcionais ou interage com o sistema a qualquer momento.

- Arquivo (não significa um arquivo do sistema operacional): refere-se a um grupo de dados logicamente relacionados e reconhecidos pelo usuário. É possível que um arquivo no sentido da APF possa estar mapeado em um ou mais arquivos físicos do sistema operacional ou em tabelas do banco de dados.

- Processo Elementar: é a menor unidade de atividade significativa para o usuário(s). Exemplo: um usuário requer que um novo colaborador seja adicionado à aplicação Funcionários. Para o usuário, a definição de colaborador inclui salário e informações sobre os dependentes. Portanto, na perspectiva do usuário a menor unidade de atividade é adicionar um novo funcionário. Caso apenas uma das partes seja adicionada, seja ela salário ou dependentes, tal fato não se constituiria numa atividade reconhecida como um processo elementar.

  Este processo elementar deve ainda ser completo em si mesmo e deixar a aplicação que está sendo contada em um estado consistente. Exemplo: o requisito do usuário para adicionar um colaborador requer incluir as informações sobre salário e dependentes. Se essas informações não são incluídas, um colaborador não pode ser criado. Adicionar apenas uma das informações deixa o processo em um estado inconsistente. Entretanto, se ambas as informações são incluídas, o processo elementar se torna completo e a aplicação fica em um estado consistente.

- Mantido: é a habilidade de modificar dados através de um processo elementar. Exemplo: incluir, alterar, excluir, atualizar, e criar dados.

- Informação de Controle: é um dado que influencia um processo elementar da aplicação que está sendo contada. Ela especifica o que, quando, ou como o dado deve ser processado.

- Identificável pelo usuário: refere-se aos requisitos definidos para processos e grupos de dados, acordados e entendidos pelos usuários e desenvolvedores.

Uma vez descrito o significado desses termos, segue a definição de ALI e AIE:

- Arquivo Lógico Interno (ALI): grupo de dados logicamente relacionados ou informações de controle, identificáveis pelo usuário, e mantidos dentro da fronteira da aplicação que está sendo contada. Sua intenção principal é armazenar dados mantidos através de um ou mais processos da aplicação que está sendo contada.

São exemplos de ALI:

- tabelas que armazenam dados mantidos pela aplicação.
- arquivos de *help*, desde que mantidos pela aplicação.
- arquivos mantidos não só pela aplicação, mas também por outra aplicação.

Não são exemplos de ALI:

- arquivos de *backup*.
- arquivos de índice.
- operações de visões.

- Arquivo de Interface Externa (AIE): grupo de dados logicamente relacionados ou informações de controle, identificáveis pelo usuário, e mantidos fora da aplicação que está sendo contada. Sua intenção primária é armazenar dados referenciados através de um ou mais processos elementares que estão dentro da fronteira da aplicação sendo contada.

  São exemplos de AIE:

  - tabelas de banco de dados, desde que referenciadas e externas à aplicação.
  - arquivos de *help*, desde que mantidos por outra aplicação.

  Não são exemplos de AIE:

  - conversão de dados, ou seja, dados que são recebidos de outra aplicação para manter um ALI.
  - dados que são apenas formatados, para a utilização em uma outra aplicação.

A diferença básica entre um ALI e um AIE é que o ALI é mantido pela aplicação que está sendo contada, enquanto o AIE não é mantido. Compreendida essa diferença, a próxima etapa do processo de contagem das funções de dados é a identificação dos ALIs e AIEs. Após isso, a complexidade funcional de cada ALI e AIE é determinada através da identificação e contagem dos DETs (*data element type*) e RETs (*record element type*).

Um DET é um campo único, não repetido, e reconhecido pelo usuário. As seguintes regras devem ser aplicadas para a identificação de um DET:

- Conte um DET para cada campo único reconhecido pelo usuário, não repetido, e mantido ou recuperado de um ALI ou AIE através de um processo elementar.
- Quando duas aplicações mantêm e/ou referenciam o mesmo ALI ou AIE, mas cada aplicação mantém ou referencia DETs separados, deve-se contar apenas os DETs que estão sendo utilizados em cada uma das aplicações cujo ALI ou AIE está sendo dimensionado.
- Conte um DET para cada campo requerido pelo usuário para estabelecer um relacionamento com outro ALI ou AIE.

Um RET é um subgrupo de DETs, reconhecido pelo usuário e que faz parte de um ALI ou AIE. Existem dois tipos de subgrupos:

- subgrupos opcionais: são aqueles que o usuário tem a opção de utilizar ou não durante um processo elementar que adiciona ou cria uma instância de dados.
- subgrupos obrigatórios: são aqueles que o usuário requer que seja utilizado durante um processo elementar que adiciona ou cria uma instância de dados.

  Como exemplo, pode ser apresentada uma situação na qual o colaborador pode ser horista ou mensalista. O usuário determina que o colaborador seja ou horista ou mensalista, e que cada um dos tipos possa possuir informações sobre dependentes. Nesse caso, haveria três subgrupos de RETs, que seriam classificados como: Colaborador Horista (obrigatório), Colaborador Mensalista (obrigatório), e Dependente (opcional).

  As seguintes regras devem ser aplicadas para a identificação de um RET:

  - Conte um RET para cada subgrupo obrigatório ou opcional de um ALI ou AIE.
  - Caso não exista subgrupos, conte o ALI ou o AIE como um RET.

Por fim, a complexidade funcional é convertida em pontos de função não ajustados. Esse processo pode ser visualizado na figura 3.3.

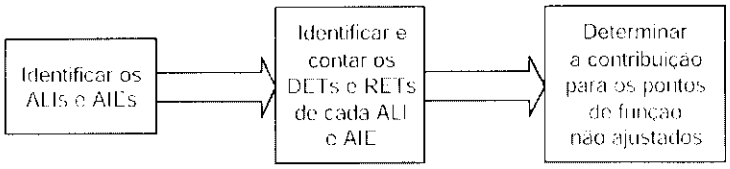


Figura 3.3: Resumo do Processo de Contagem de um ALI e de um AIE [40]

As próximas duas subseções apresentam em detalhes como os ALIs e os AIEs são identificados e classificados quanto a sua complexidade funcional.

Identificação de Arquivo Lógico Interno (ALI)

Um ALI pode ser identificado utilizando as seguintes regras:

- O grupo de dados ou informação de controle é logicamente relacionado e identificável pelo usuário.
- O grupo de dados é mantido dentro da fronteira da aplicação que está sendo contada.

Após identificar e contar os DETs e RETs, deve-se determinar a complexidade do ALI utilizando a tabela 3.1:

	1 a 19 DET	20 a 50 DET	51 ou mais DET
1 RET	Baixa	Baixa	Média
2 a 5 RET	Baixa	Média	Alta
6 ou mais RET	Média	Alta	Alta

Tabela 3.1: Identificação da Complexidade Funcional de um ALI [40]

A complexidade funcional é convertida em pontos de função não ajustados utilizando a tabela 3.2:

Complexidade Funcional	Pontos de Função Não Ajustados
Baixa	7
Média	10
Alta	15

Tabela 3.2: Contagem dos Pontos de Função Não Ajustados - ALI [40]

Identificação de Arquivo de Interface Externa (AIE)

Um AIE pode ser identificado utilizando as seguintes regras:

- O grupo de dados ou informação de controle é logicamente relacionado e identificável pelo usuário.
- O grupo de dados é referenciado e externo à aplicação que está sendo contada.
- O grupo de dados não é mantido pela aplicação que está sendo contada.
- O grupo de dados é mantido em um ALI de outra aplicação.

Para determinar a complexidade funcional de um AIE - assim como na identificação dos ALIs - deve-se identificar e contar o número de DETs e RETs.

Após identificar e contar os DETs e RETs, deve-se determinar a complexidade do AIE utilizando a tabela 3.3:

	1 a 19 DET	20 a 50 DET	51 ou mais DET
1 RET	Baixa	Baixa	Média
2 a 5 RET	Baixa	Média	Alta
6 ou mais RET	Média	Alta	Alta

Tabela 3.3: Identificação da Complexidade Funcional de um AIE [40]

A complexidade funcional é convertida em pontos de função não ajustados utilizando a tabela 3.4:

Complexidade Funcional	Pontos de Função Não Ajustados
Baixa	5
Média	7
Alta	10

Tabela 3.4: Contagem dos Pontos de Função Não Ajustados - AIE [40]

3.1.4 Contar as Funções Transacionais

As funções transacionais representam as funcionalidades de processamento de dados fornecidas pela aplicação ao usuário. Elas são classificadas em Entradas Externas (EEs), Saídas Externas (SEs) ou Consultas Externas (CEs).

Para entender a definição de EE, SE e CE, é preciso compreender o significado dos seguintes termos utilizados na APF: usuário, arquivo, processo elementar, mantido, informação de controle, identificável pelo usuário, e lógica de processamento.

Dos termos apresentados, com exceção de lógica de processamento, todos já foram descritos na seção anterior, que fala da identificação dos ALIs e AIEs. Porém, visando facilitar a consulta dos mesmos, são apresentados novamente a seguir, junto com a definição de lógica de processamento.

- Usuário: é qualquer pessoa ou coisa que especifica requisitos funcionais ou interage com o sistema a qualquer momento.

- Arquivo (não significa um arquivo do sistema operacional): refere-se a um grupo de dados logicamente relacionados e reconhecidos pelo usuário. É possível que um arquivo no sentido da APF possa estar mapeado em um ou mais arquivos físicos do sistema operacional ou em tabelas do banco de dados.

- Processo Elementar: é a menor unidade de atividade significativa para o usuário(s). Exemplo: um usuário requer que um novo colaborador (colocar nota de rodapé) seja adicionado à aplicação Funcionários. Para o usuário, a definição de colaborador inclui salário e informações sobre os dependentes. Portanto, na perspectiva do usuário a menor unidade de atividade é adicionar um novo funcionário. Caso apenas uma das partes seja adicionada, seja ela salário ou dependentes, tal fato não se constituiria numa atividade reconhecida como um processo elementar.

  Este processo elementar deve ainda ser completo em si mesmo e deixar a aplicação que está sendo contada em um estado consistente. Exemplo: o requisito do usuário para adicionar um colaborador requer incluir as informações sobre salário e dependentes. Se essas informações não são incluídas, um colaborador não pode ser criado. Adicionar apenas uma das informações deixa o processo em um estado inconsistente. Entretanto, se ambas as informações são incluídas, o processo elementar se torna completo e a aplicação fica em um estado consistente.

- Mantido: é a habilidade de modificar dados através de um processo elementar. Exemplos: incluir, alterar, excluir, atualizar, e criar dados.

- Informação de Controle: é um dado que influencia um processo elementar da aplicação

que está sendo contada. Ela especifica o que, quando, ou como o dado deve ser processado.

- Identificável pelo usuário: refere-se aos requisitos definidos para processos e grupos de dados, acordados e entendidos pelos usuários e desenvolvedores.

- Lógica de Processamento: requisitos que são especificamente solicitados pelo usuário para completar um processo elementar. Esses requisitos podem incluir ações nas quais:

  1. Validações são realizadas.
     Ex: Quando um novo funcionário é incluído, a lógica de processamento verifica se o CPF do funcionário é válido.
  2. Fórmulas matemáticas e cálculos são realizados.
     Ex: Um relatório que além de apresentar todos os funcionários, calcula campos que apresentem o total de funcionários, o total de funcionários horistas, e o total de funcionários mensalistas.
  3. Valores equivalentes são convertidos.
     Ex: Conversão de moeda. Através da recuperação de valores de tabelas (não existem cálculos), valores em reais são covertidos para dólares ou euros.
  4. Dados são filtrados e selecionados utilizando-se critérios para comparar múltiplos conjuntos de dados.
     Ex: Para gerar uma lista de funcionários cujas férias vencem no próximo mês, o processo elementar compara as datas de vencimento das férias de cada funcionário.
  5. Condições são analisadas para se determinar quais são aplicáveis.
     Ex: Quando um novo funcionário é incluído e existe a possibilidade dele ser pago por horas ou por mês de trabalho.
  6. Um ou mais ALIs são atualizados.
     Ex: Quando um funcionário é incluído, o processo elementar atualiza o ALI Funcionários, para manter os dados do funcionário.
  7. Um ou mais ALIs ou AIEs são referenciados.
     Ex: Ao incluir um colaborador, o AIE CEPs é referenciado para determinar se o CEP informado está correto.

8. Dados ou informações de controle são recuperados.

   Ex: Para visualizar uma lista de pagamentos, as informações sobre pagamentos são recuperadas do sistema.

9. Dados derivados são criados através da transformação de dados existentes em novos dados.

   Ex: Para determinar o número de registro de um funcionário (LUSI01) os seguintes dados são concatenados:

   - as duas primeiras letras do nome (LU para LUCIANA)
   - as duas primeiras letras do sobrenome (SI para SILVA)
   - um número de controle (iniciando em 01)

10. O comportamento do sistema é alterado.

    Ex: O comportamento do sistema é alterado para que os pagamentos sejam efetuados aos funcionários semanalmente ao invés de mensalmente.

11. Informações são apresentadas.

    Ex: Uma lista de funcionários é apresentada ao usuário.

12. Existe a capacidade de aceitar dados ou informações de controle que entram pela fronteira da aplicação.

    Ex: Um usuário entra com várias informações para incluir um agendamento de recebimento.

13. Dados são ordenados ou organizados.

    Ex: O usuário solicita uma lista de nomes em ordem alfabética.

A tabela 3.5 exibe um resumo das lógicas de processamento que podem ser executadas pelas funções transacionais:

Tipo de Lógica de Processamento	Função Transacional		
	EE	SE	CE
1- Validações são realizadas	p	p	p
2- Fórmulas matemáticas e cálculos são realizados	p	o*	n
3- Valores equivalentes são convertidos	p	p	p
4- Dados são filtrados e selecionados utilizando-se critérios para comparar múltiplos conjuntos de dados	p	p	p
5- Condições são analisadas para se determinar quais são aplicáveis	p	p	p
6- Um ou mais ALIs são atualizados	o*	o*	n
7- Um ou mais ALIs ou AIEs são referenciados	p	p	o
8- Dados ou informações de controle são recuperados	p	p	o
9- Dados derivados são criados através da transformação de dados existentes em novos dados	p	p	o
10- O comportamento do sistema é alterado	o*	o*	n
11- Informações são apresentadas	p	o	o
12- Existe a capacidade de aceitar dados ou informações de controle que entram pela fronteira da aplicação	o	p	p
13- Dados são ordenados ou organizados	p	p	p

Tabela 3.5: Resumo das Lógicas de Processamento utilizadas pelas EEs, SEs, e CEs [40]

Legenda:

o é obrigatório que a função execute essa lógica de processamento.

o* é obrigatório que a função execute ao menos uma das lógicas identificadas como o*.

p a função pode executar a lógica de processamento. Entretanto, não existe uma obrigação.

n a função não pode executar a lógica de processamento.

Após a definição desses termos, segue a definição de cada uma das funções transacionais:

- Entrada Externa: processo elementar que processa dados ou informações de controle que vêm de fora da fronteira da aplicação. Sua principal intenção é manter um ou mais arquivos lógicos internos e/ou alterar o comportamento do sistema.

São exemplos de EE:

- inclusão, alteração ou exclusão de um cliente num cadastro de uma loja qualquer (nesse caso, são 3 EEs).
- importação de dados de clientes cadastrados em outro sistema (dados de conversão).

Não são exemplos de EE:

- menus.
- telas de *login*.

- Saída Externa: processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. Sua principal intenção é apresentar informação ao usuário através de lógica de processamento que não seja apenas uma simples recuperação de dados ou informações de controle. Seu processamento deve conter cálculo, criar dados derivados, manter um arquivo lógico interno, ou alterar o comportamento do sistema.

  São exemplos de SE:

  - relatórios que apresentam um campo com a totalização dos dados (existe cálculo).
  - informações em formato gráfico.

  Não são exemplos de SE:

  - telas de ajuda (*help*).
  - listas de seleção (*drop-downs*).

- Consulta Externa: processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. Sua principal intenção é apresentar informações ao usuário através da recuperação de dados ou informações de controle de um ALI ou AIE. A lógica de processamento não contém fórmulas matemáticas ou cálculos, não cria dados derivados e não mantém ALIs durante o processamento.

  São exemplos de CE:

  - telas de ajuda (*help*).
  - listas de seleção (*drop-downs*), desde que recuperem dados de um ALI.

Não são exemplos de CE:

- relatórios que contenham cálculo ou gerem dados derivados.
- listas de seleção (*drop-downs*) que possuam dados estáticos.

O processo de contagem das funções transacionais começa pela identificação das EEs, SEs e CEs. A seguir, é determinada a complexidade funcional de cada função transacional com base na identificação e contagem dos DETs (*data element type*) e FTRs (*file type referenced*).

Um DET é um campo único, não repetido, e reconhecido pelo usuário. A definição de DET é a mesma já vista na identificação das funções de dados. Entretanto as regras utilizadas para identificar os DETs de cada uma das funções transacionais são diferentes das utilizadas para as funções de dados.

Um FTR é um ALI lido ou mantido pela função transacional, ou um AIE lido pela função transacional. Cada função transacional possui regras específicas para a identificação dos FTRs.

Após a identificação e contagem dos DETs e FTRs, é possível determinar a complexidade funcional de cada EE, SE, e CE. Após isso, a complexidade funcional é convertida em pontos de função não ajustados, como mostra a figura 3.4.

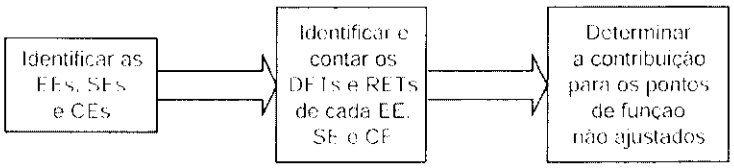


Figura 3.4: Resumo do Processo de Contagem das EEs, SEs e CEs [40]

As próximas subseções apresentam em detalhes como as EEs, SEs, e CEs são identificadas e classificadas quanto a sua complexidade funcional.

Identificação de Entrada Externa (EE)

Para cada processo elementar cuja principal intenção seja manter um ou mais ALIs ou alterar o comportamento do sistema, as seguintes regras devem ser aplicadas para a identificação de uma única Entrada Externa:

- Os dados ou informações de controle são recebidos de fora da fronteira da aplicação.
- Ao menos um ALI é mantido caso os dados que entram pela fronteira não sejam informações de controle que alterem o comportamento do sistema.

- Para o processo identificado, ao menos uma das três regras abaixo deve ser satisfeita:
  - A lógica de processamento é única, quando comparada com outras entradas externas da aplicação.
  - O conjunto de tipos de dados identificado é diferente de outros conjuntos identificados em outras entradas externas.
  - Os ALIs e AIEs referenciados são diferentes dos arquivos referenciados por outras entradas externas da aplicação.

A contagem dos FTRs e DETs das entradas externas identificadas baseiam-se nas seguintes regras:

1. Contagem de FTRs:
   (a) Conte um FTR para cada ALI mantido.
   (b) Conte um FTR para cada ALI ou AIE lido durante o processamento.
   (c) Conte um FTR para cada ALI que é tanto mantido quanto referenciado.
2. Contagem de DETs:
   (a) Conte um DET para cada campo único reconhecido pelo usuário, não repetido, que entra ou sai pela fronteira da aplicação e é necessário à conclusão do processo.
   (b) Não conte campos que são recuperados ou derivados pelo sistema e armazenados em um ALI durante o processo elementar caso os campos não atravessem a fronteira da aplicação.
   (c) Conte um DET para a capacidade de enviar mensagens de resposta do sistema para fora da fronteira da aplicação, indicando erros durante o processamento, ou confirmações de que o processamento foi concluído.
   (d) Conte um DET para a capacidade de especificar uma ação a ser tomada. Caso existam múltiplos métodos de se invocar o mesmo processo lógico, apenas um DET deverá ser contado.

Após identificar e contar os DETs e FTRs, deve-se determinar a complexidade funcional da EE utilizando a tabela 3.6:

A complexidade funcional é convertida em pontos de função não ajustados utilizando a tabela 3.7:

	1 a 4 DETs	5 a 15 DETs	16 ou mais DETs
1 FTR	Baixa	Baixa	Média
2 FTRs	Baixa	Média	Alta
3 ou mais FTRs	Média	Alta	Alta

Tabela 3.6: Identificação da Complexidade Funcional - EE [40]

Complexidade Funcional	Pontos de Função Não Ajustados
Baixa	3
Média	4
Alta	6

Tabela 3.7: Contagem dos Pontos de Função Não Ajustados - EE [40]

Identificação de Saída Externa (SE)

Para cada processo elementar cuja principal intenção seja apresentar informações ao usuário, as seguintes regras devem ser aplicadas para a identificação de uma única Saída Externa:

- A função envia dados ou informações de controle para fora da fronteira da aplicação.
- Para o processo identificado, ao menos uma das três regras abaixo deve ser satisfeita:

  1. A lógica de processamento é única, quando comparada com outras saídas externas da aplicação.
  2. O conjunto de tipos de dados identificado é diferente de outros conjuntos identificados em outras saídas externas.
  3. Os ALIs e AIEs referenciados são diferentes dos arquivos referenciados por outras saídas externas da aplicação.

  Além disso:

  - A lógica de processamento contém ao menos uma fórmula matemática ou cálculo.
  - A lógica de processamento cria dados derivados.
  - A lógica de processamento mantém ao menos um ALI.
  - A lógica de processamento altera o comportamento do sistema.

A contagem dos FTRs e DETs das saídas externas identificadas baseiam-se nas seguintes regras:

1. Contagem de FTRs:

   (a) Conte um FTR para cada ALI ou AIE lido durante o processo.

   (b) Conte um FTR para cada ALI mantido durante o processo.

   (c) Conte um FTR para cada ALI que é tanto mantido quanto referenciado durante o processo.

2. Contagem de DETs (essas regras também se aplicam para identificação dos DETs das consultas externas):

   (a) Conte um DET para cada campo único reconhecido pelo usuário, não repetido, que entra pela fronteira da aplicação e é requerido para especificar quando, o que e/ou como os dados são recuperados ou gerados.

   (b) Conte um DET para cada campo único reconhecido pelo usuário, não repetido, que sai pela fronteira da aplicação.

   (c) Caso um DET entre e saia através da fronteira da aplicação, deve ser contado uma única vez.

   (d) Conte um DET para a capacidade de enviar mensagens de resposta do sistema para fora da fronteira da aplicação, indicando erros durante o processamento, ou confirmações de que o processamento foi concluído.

   (e) Conte um DET para a capacidade de especificar uma ação a ser tomada. Caso existam múltiplos métodos para invocar o mesmo processo lógico, apenas um DET deverá ser contado.

   (f) Não conte campos que são recuperados ou derivados pelo sistema e armazenados em um ALI durante o processo elementar caso os campos não atravessem a fronteira da aplicação.

   (g) Não conte literais como DETs (títulos, cabeçalhos de relatórios ou telas).

   (h) Não conte variáveis de paginação (comandos de paginação tais como anterior ou próximo).

Após identificar e contar os DETs e FTRs, deve-se determinar a complexidade funcional da SE utilizando a tabela 3.8:

	1 a 5 DETs	6 a 19 DETs	20 ou mais DETs
0 ou 1 FTR	Baixa	Baixa	Média
2 a 3 FTRs	Baixa	Média	Alta
4 ou mais FTRs	Média	Alta	Alta

Tabela 3.8: Identificação da Complexidade Funcional - SE [40]

A complexidade funcional é convertida em pontos de função não ajustados utilizando a tabela 3.9:

Complexidade Funcional	Pontos de Função Não Ajustados
Baixa	4
Média	5
Alta	7

Tabela 3.9: Contagem dos Pontos de Função Não Ajustados - SE [40]

Identificação de Consulta Externa (CE)

Para cada processo elementar cuja principal intenção seja apresentar informações ao usuário, as seguintes regras devem ser aplicadas para a identificação de uma única Consulta Externa:

- A função envia dados ou informações de controle para fora da fronteira da aplicação.
- Para o processo identificado, ao menos uma das três regras abaixo deve ser satisfeita:
    1. A lógica de processamento é única, quando comparada com outras consultas externas da aplicação.
    2. O conjunto de tipos de dados identificado é diferente de outros conjuntos identificados em outras consultas externas.
    3. Os ALIs e AIEs referenciados são diferentes dos arquivos referenciados por outras consultas externas da aplicação.

Além disso:

- A lógica de processamento recupera dados ou informações de controle de um ALI ou AIE.
- A lógica de processamento não contém fórmulas matemáticas ou cálculos.
- A lógica de processamento não cria dados derivados.
- A lógica de processamento não mantém ALIs.
- A lógica de processamento não altera o comportamento do sistema.

A contagem dos FTRs e DETs das consultas externas identificadas baseiam-se nas seguintes regras:

1. Contagem de FTRs:

   (a) Conte um FTR para cada ALI ou AIE lido durante o processo.

2. Contagem de DETs [1]:

   (a) Conte um DET para cada campo único reconhecido pelo usuário, não repetido, que entra pela fronteira da aplicação e é requerido para especificar quando, o que e/ou como os dados são recuperados ou gerados.

   (b) Conte um DET para cada campo único reconhecido pelo usuário, não repetido, que sai pela fronteira da aplicação.

   (c) Caso um DET entre e saia através da fronteira da aplicação, deve ser contado uma única vez.

   (d) Conte um DET para a capacidade de enviar mensagens de resposta do sistema para fora da fronteira da aplicação, indicando erros durante o processamento, ou confirmações de que o processamento foi concluído.

   (e) Conte um DET para a capacidade de especificar uma ação a ser tomada. Caso existam múltiplos métodos para invocar o mesmo processo lógico, apenas um DET deverá ser contado.

   (f) Não conte campos que são recuperados ou derivados pelo sistema e armazenados em um ALI durante o processo elementar caso os campos não atravessem a fronteira da aplicação.

[1] é idêntica a descrita para as saídas externas

(g) Não conte literais como DETs (títulos, cabeçalhos de relatórios ou telas).

(h) Não conte variáveis de paginação (comandos de paginação tais como anterior ou próximo).

Após identificar e contar os DETs e FTRs, deve-se determinar a complexidade funcional da CE utilizando a tabela 3.10:

	1 a 5 DETs	6 a 19 DETs	20 ou mais DETs
1 FTR	Baixa	Baixa	Média
2 a 3 FTRs	Baixa	Média	Alta
4 ou mais FTRs	Média	Alta	Alta

Tabela 3.10: Identificação da Complexidade Funcional - CE [40]

A complexidade funcional é convertida em pontos de função não ajustados utilizando a tabela 3.11:

Complexidade Funcional	Pontos de Função Não Ajustados
Baixa	3
Média	4
Alta	6

Tabela 3.11: Contagem dos Pontos de Função Não Ajustados - CE [40]

3.1.5 Determinar os Pontos de Função Não Ajustados

Uma vez identificadas e classificadas as funções de dados (ALIs e AIEs) e transacionais (EEs, SEs, CEs), basta somar todas as complexidades funcionais convertidas em pontos de função para obter os pontos de função não ajustados. A tabela 3.12 representa o cálculo dos pontos de função não ajustados.

Tipo de Função	Complexidade Funcional	Total Complexidade	Total por Tipo de Função
ALIs	_____ Baixa	x 7 = _____	
	_____ Média	x 10 = _____	
	_____ Alta	x 15 = _____	
			_____
AIEs	_____ Baixa	x 5 = _____	
	_____ Média	x 7 = _____	
	_____ Alta	x 10 = _____	
			_____
EEs	_____ Baixa	x 3 = _____	
	_____ Média	x 4 = _____	
	_____ Alta	x 6 = _____	
			_____
SEs	_____ Baixa	x 4 = _____	
	_____ Média	x 5 = _____	
	_____ Alta	x 7 = _____	
			_____
CEs	_____ Baixa	x 3 = _____	
	_____ Média	x 4 = _____	
	_____ Alta	x 6 = _____	
			_____
Total dos Pontos de Função Não Ajustados			_____

Tabela 3.12: Cálculo dos Pontos de Função Não Ajustados [40]

3.1.6 Determinar o Fator de Ajuste

O fator de ajuste, como o próprio nome diz, ajusta a contagem dos pontos de função não ajustados, produzindo uma variação na contagem de mais ou menos 35 %. Ele é baseado nas 14 Características Gerais dos Sistemas (CGS), responsáveis por avaliar as funcionalidades que afetam a aplicação de uma maneira geral.

Características Gerais dos Sistemas	
1. Comunicação de Dados	8. Atualização *On-Line*
2. Processamento de Dados Distribuído	9. Processamento Complexo
3. Performance	10. Reusabilidade
4. Utilização do Equipamento	11. Facilidade de Instalação
5. Volume de Transações	12. Facilidade de Operação
6. Entrada de Dados *On-Line*	13. Múltiplos Locais
7. Eficiência do Usuário Final	14. Facilidade de Mudanças

Tabela 3.13: Características Gerais dos Sistemas [40]

As CGS possuem um nível de influência (ni) que pode variar entre um e cinco. Os níveis estão representados na tabela 3.14.

	Níveis de Influência
0	Nenhuma Influência
1	Influência Mínima
2	Influência Moderada
3	Influência Média
4	Influência Significativa
5	Grande Influência

Tabela 3.14: Níveis de Influência das CGS [40]

O nível de influência é obtido a partir de um questionário. Para cada uma das características gerais dos sistemas existem cinco opções, determinando o nível de influência mais adequado para o sistema.

A seguir, cada uma das características gerais dos sistemas é detalhada, descrevendo as opções que podem ser escolhidas para determinar o nível de influência de cada uma delas.

1- Comunicação de Dados

Descreve o nível de comunicação direta entre a aplicação e o processador. Os dados ou informações de controle utilizados pela aplicação são enviados ou recebidos por meio de recursos de comunicação. Os terminais conectados localmente à unidade de controle são considerados recursos de comunicação, e todos os *links* de comunicação de dados requerem algum tipo de

protocolo [2].

Pontue como segue:

0 A aplicação é puramente *batch* ou uma estação de trabalho isolada.

1 A aplicação é puramente *batch*, mas possui entradas de dados ou impressão remota.

2 A aplicação é *batch*, mas possui entrada de dados e impressão remota.

3 A aplicação possui entrada de dados *on-line*, *front-end* de teleprocessamento para um processamento batch ou sistema de consulta.

4 A aplicação é mais que um *front-end*, mas suporta apenas um tipo de protocolo de comunicação.

5 A aplicação é mais que um *front-end*, e suporta mais de um tipo de protocolo de comunicação.

2- Processamento de Dados Distribuído

Descreve em que nível a aplicação transfere dados entre seus componentes. Funções ou dados distribuídos dentro da fronteira são características da aplicação.

Pontue como segue:

0 A aplicação não auxilia na transferência de dados ou processamento de funções entre os componentes do sistema.

1 A aplicação prepara dados para processamento em outro componente do sistema, tal como uma planilha eletrônica ou um banco de dados.

2 Dados são preparados para transferência. Então, são transferidos e processados em outro componente do sistema (não para processamento pelo usuário final).

3 Processamento distribuído e transferência de dados são feitos *on-line* e em apenas uma direção.

4 Processamento distribuído e transferência de dados são feitos *on-line* e em ambas as direções.

5 Os processamentos de funções são executados dinamicamente no componente mais apropriado do sistema.

[2] conjunto de convenções que permite a transferência ou intercâmbio de informações entre dois sistemas ou dispositivos.

3- Performance

Descreve em que nível o tempo de resposta e taxa de transações influenciam o desenvolvimento da aplicação. Os objetivos estabelecidos ou aprovados pelo usuário, em termos de tempo de resposta ou taxa de transações, influenciam (ou irão influenciar) o projeto, desenvolvimento, instalação e suporte da aplicação.

Pontue como segue:

0 Nenhum requisito especial de performance foi especificado pelo usuário.

1 Requisitos de performance e de projeto foram estabelecidos e revisados, mas nenhuma ação em especial foi requerida.

2 O tempo de resposta é crítico durante os horários de pico. Nenhum projeto especial para a utilização de CPU é requerido. O limite para o processamento é o dia de trabalho seguinte.

3 O tempo de resposta é crítico durante todas as horas de trabalho. Nenhum projeto especial para a utilização de CPU é requerido. O limite de processamento é crítico.

4 Adicionalmente, requisitos especificados pelo usuário são exigentes o bastante para que tarefas de análise de performance sejam necessárias na fase de projeto.

5 Adicionalmente, ferramentas de análise de performance devem ser utilizadas nas fases de projeto, desenvolvimento, e/ou implementação para que os requisitos de performance do usuário sejam atendidos.

4- Utilização do Equipamento

Descreve em que nível as restrições de recursos de computação influenciam no desenvolvimento da aplicação. Uma configuração operacional altamente utilizada, necessitando de considerações especiais de projeto, é uma característica da aplicação.

Pontue como segue:

0 Nenhuma restrição operacional explícita ou implícita é incluída.

1 Existem restrições operacionais, mas são menos restritivas que uma aplicação típica. Não há esforço necessário ao atendimento dessas restrições.

2 Algumas considerações de sincronismo ou segurança são incluídas.

3 Requisitos específicos de processador para uma parte específica da aplicação são incluídos.

4 Restrições operacionais estabelecidas necessitam de um processador dedicado ou grande utilização do processador central.

5 Adicionalmente, existem limitações nos componentes distribuídos da aplicação.

5- Volume de Transações

Descreve em que nível as transações do negócio influenciam o projeto, desenvolvimento, instalação e suporte da aplicação.

Pontue como segue:

0 Não é antecipado nenhum período de pico de transações.

1 São antecipados períodos de pico de processamento (ex: mensal, quinzenal, periódico, anual).

2 Picos de transação semanais são previstos.

3 Picos de transação diários são previstos.

4 Altas taxas de transação definidas pelo usuário nos requisitos da aplicação ou níveis de serviço acordados são altos o bastante para requererem tarefas de análise de performance na fase de projeto.

5 Além do que está descrito no item 4, existem requisitos de ferramentas de análise de performance nas fases de projeto, desenvolvimento e/ou instalação.

6- Entrada de Dados *On-Line*

Descreve em que níveis são efetuadas entradas de dados na aplicação por meio de transações interativas.

Pontue como segue:

0 Todas as transações são processadas em lote.

1 De 1% a 7% das transações são entradas de dados interativos.

2 De 8% a 15% das transações são entradas de dados interativos.

3 De 16% a 23% das transações são entradas de dados interativos.

4 De 24% a 30% das transações são entradas de dados interativos.

5 Mais de 30% das transações são entradas de dados interativos.

7- Eficiência do Usuário Final

Descreve em que nível considerações sobre fatores humanos e facilidade de uso pelo usuário influenciam o desenvolvimento da aplicação. As funções interativas enfatizam um projeto para o aumento da eficiência do usuário final. O projeto inclui:

- auxílio para navegação (por exemplo: teclas de função, saltos, menus gerados dinamicamente).
- menus.
- ajuda *on-line* e documentação.
- movimentação automática do cursor.
- paginação.
- impressão remota por meio de transações *on-line*.
- teclas de função pré-definidas.
- tarefas em lote submetidas a transações *on-line*.
- seleção feita por posicionamento de cursor em tela de dados.
- uso intensivo de vídeo reverso, brilho, cores e outros indicadores.
- documentação impressa das transações.
- interface de mouse.
- janelas *pop-up*.
- utilização de número mínimo de telas para executar uma função do negócio.
- suporte a dois idiomas (conte como quatro itens).
- suporte a mais de dois idiomas (conte como seis itens).

Pontue como segue:

0 nenhum dos itens acima descritos.

1 de um a três dos itens acima descritos.

2 de quatro a cinco dos itens acima descritos.

3 seis ou mais itens acima descritos, mas não existem requisitos específicos do usuário associados à eficiência.

4 seis ou mais dos itens acima descritos, e requisitos estabelecidos sobre a eficiência para o usuário final são fortes o bastante para necessitarem de tarefas de projeto que incluam fatores humanos (por exemplo: minimizar o número de toques no teclado, maximizar padrões e uso de modelos).

5 seis ou mais dos itens acima descritos, e requisitos estabelecidos sobre a eficiência para o usuário final são fortes o bastante para necessitarem do uso de ferramentas e processos especiais para demonstrar que os objetivos foram alcançados.

8- Atualização *On-Line*

Descreve em que nível os arquivos lógicos internos da aplicação são atualizados de forma *on-line*.

Pontue como segue:

0 Não há atualização *on-line*.

1 Há atualização *on-line* de 1 a 3 arquivos. O volume de atualização é pequeno e a recuperação é fácil.

2 Há atualização *on-line* de 4 ou mais arquivos. O volume de atualização é pequeno e a recuperação é fácil.

3 A atualização da maioria dos arquivos internos é *on-line*.

4 Adicionalmente, a proteção contra a perda de dados é essencial e foi especialmente projetada e programada no sistema.

5 Adicionalmente, o alto volume de processamento torna necessária a análise do custo do processo de recuperação. São incluídos procedimentos altamente automatizados com um mínimo de intervenção do operador.

9- Processamento Complexo

Descreve em que nível o processamento lógico influencia o desenvolvimento da aplicação. Os seguintes componentes estão presentes:

- Controle sensível (por exemplo: processamento especial de auditoria) e/ou processamento de segurança específico da aplicação.
- Processamento lógico extensivo.
- Processamento matemático extensivo.
- Muito processamento de exceção resultando em transações incompletas que devem ser processadas novamente (por exemplo: transações incompletas em ATM[3] causadas por problemas de interrupção de teleprocessamento, falta de dados ou falhas nas validações).
- Processamento complexo para manipular múltiplas possibilidades de entrada e saída, (por exemplo: multimídia, ou independência de dispositivo).

Pontue como segue:

0 Nenhum dos itens acima.

1 Qualquer um dos itens acima.

2 Quaisquer dois dos itens acima.

3 Quaisquer três dos itens acima.

4 Quaisquer quatro dos itens acima.

5 Todos os cinco itens acima.

[3] *Automated Teller Machine*

10- Reusabilidade

Descreve em que nível a aplicação e seu código foram especificamente projetados, desenvolvidos e suportados para serem utilizados em outras aplicações.

Pontue como segue:

0 Não há código reutilizável.

1 Código reutilizável é utilizado na aplicação.

2 Menos de dez por cento da aplicação levou em consideração as necessidades de mais de um usuário.

3 Dez por cento ou mais da aplicação levou em consideração as necessidades de mais de um usuário.

4 A aplicação foi especificamente empacotada e/ou documentada para fácil reutilização. Ela é customizada pelo usuário no código.

5 A aplicação foi especificamente empacotada e/ou documentada para fácil reutilização. Ela é customizada pelo usuário por meio de manutenção de parâmetros.

11- Facilidade de Instalação

Descreve em que nível a conversão de ambientes preexistentes influencia o desenvolvimento da aplicação. Um plano de conversão e instalação e/ou ferramentas de conversão foram fornecidas e testadas durante a fase de teste do sistema.

Pontue como segue:

0 Nenhuma consideração especial foi definida pelo usuário, e não é requerido nenhum *setup* para a instalação.

1 Nenhuma consideração especial foi definida pelo usuário, mas é necessário *setup* para a instalação.

2 Requisitos de instalação e conversão foram definidos pelo usuário, e guias de conversão e instalação foram fornecidos e testados. O impacto da conversão sobre o projeto não é considerado importante.

3 Requisitos de instalação e conversão foram definidos pelo usuário, e guias de conversão e instalação foram fornecidos e testados. O impacto da conversão sobre o projeto é considerado importante.

4 Além do item 2, ferramentas de instalação e conversão automáticas foram fornecidas e testadas.

5 Além do item 3, ferramentas de instalação e conversão automáticas foram fornecidas e testadas.

12- Facilidade de Operação

Descreve em que nível a aplicação atende a alguns aspectos operacionais tais como: inicialização, *backup* e recuperação. A aplicação minimiza a necessidade de atividades manuais, como montagem de fitas, manipulação de papel e intervenção manual pelo operador.

Pontue como segue:

0 Além dos procedimentos de *backup* normais, nenhuma consideração operacional especial foi definida pelo usuário.

1 - 4 Um, alguns ou todos os seguintes itens se aplicam para a aplicação. Cada item tem valor de um ponto, com exceção daqueles cujo valor é citado.

- Procedimentos de inicialização, *backup* e recuperação foram fornecidos, mas é necessária a intervenção do operador.
- Procedimentos de inicialização, *backup* e recuperação foram fornecidos, e não é necessária a intervenção do operador (conte como dois itens).
- A aplicação minimiza a necessidade de montagem de fitas.
- A aplicação minimiza a necessidade de manipulação de papel.

5 A aplicação é projetada para operação não-assistida, isto é, não é necessária a intervenção do operador para operar o sistema, que não seja a inicialização e término da aplicação. A recuperação automática de erro é uma característica da aplicação.

13- Múltiplos Locais

Descreve em que nível a aplicação foi desenvolvida para múltiplos locais e organizações. A aplicação foi especificamente projetada, desenvolvida e suportada para instalação em múltiplos locais por múltiplas organizações.

Pontue como segue:

0 Os requisitos do usuário não consideram a necessidade de mais de um usuário/local de instalação.

1 Necessidades de múltiplos locais foram consideradas no projeto, e a aplicação foi projetada para operar apenas em ambientes idênticos de hardware e de software.

2 Necessidades de múltiplos locais foram consideradas no projeto, e a aplicação foi projetada para operar em ambientes de hardware e de software similares.

3 Necessidades de múltiplos locais foram consideradas no projeto, e a aplicação foi projetada para operar em ambientes diferentes de hardware e de software.

4 Adicionalmente aos itens 1 ou 2, plano de suporte e documentação são fornecidos e testados para apoiar a aplicação em múltiplos locais.

5 Adicionalmente ao item 3, plano de suporte e documentação são fornecidos e testados para apoiar a aplicação em múltiplos locais.

14- Facilidade de Mudanças

Descreve em que nível a aplicação foi desenvolvida para facilitar a alteração de sua lógica de processamento ou estrutura de dados.

As seguintes características podem ser válidas para a aplicação:

- São fornecidos mecanismos de consulta flexível, que permitem a manipulação de pedidos simples; por exemplo, lógica de e/ou aplicada a apenas um arquivo lógico (conte como um item).

- São fornecidos mecanismos de consulta flexível, que permitem a manipulação de pedidos de média complexidade; por exemplo, lógica de e/ou aplicada a mais de um arquivo lógico (conte como dois itens).

- São fornecidos mecanismos de consulta flexível, que permitem a manipulação de pedidos complexos; por exemplo, lógica de e/ou combinadas em um ou mais arquivos lógicos (conte como três itens).

- Dados de controle do negócio são mantidos pelo usuário por meio de processos interativos, mas as alterações só tem efeito no próximo dia útil (conte como um item).

- Dados de controle do negócio são mantidos pelo usuário por meio de processos interativos, e as alterações têm efeito imediato (conte como dois itens).

Pontue como segue:

0 Nenhum dos itens acima.

1 Qualquer um dos itens acima.

2 Quaisquer dois dos itens acima.

3 Quaisquer três dos itens acima.

4 Quaisquer quatro dos itens acima.

5 Todos os cinco itens acima.

Após avaliar cada uma das CGS e determinar o nível de influência, o fator de ajuste é calculado com a seguinte fórmula:

$$\text{Fator de Ajuste} = (Ni * 0,01) + 0,65$$

3.1.7 Calcular os Pontos de Função Ajustados

A última etapa do procedimento de contagem dos pontos de função inclui as fórmulas para o cálculo dos três tipos de contagem: projeto de desenvolvimento, projeto de manutenção, e aplicação.

Projeto de Desenvolvimento

O cálculo dos pontos de função de um projeto de desenvolvimento envolve a análise de três componentes:

- Funcionalidade da aplicação requisitada pelo usuário: são as funções utilizadas após a instalação do software para satisfazer as necessidades do usuário.

- Funcionalidade de conversão requisitada pelo usuário: são as funções disponíveis somente no momento da instalação da aplicação para converter dados ou fornecer outros requisitos especificados pelo usuário. Após a instalação essas funções são descartadas.

- Valor do fator de ajuste da aplicação: o fator de ajuste é determinado pela avaliação das 14 características gerais dos sistemas, e é responsável pela complexidade funcional da aplicação.

A seguinte fórmula é utilizada para calcular os pontos de função de um projeto de desenvolvimento:

$$DFP = (UFP - CFP) * VAF$$

Onde
DFP é a contagem dos pontos de função de um projeto de desenvolvimento.
UFP é a contagem dos pontos de função não ajustados para as funções que estarão disponíveis depois da instalação.
CFP são os pontos de função não ajustados adicionados pelas funções de conversão.
VAF é o valor do fator de ajuste.

Projeto de Manutenção

Existem quatro tipos de modificações que podem ser efetuadas durante a fase de manutenção[54]:

- Correção: Mesmo com as melhores garantias de qualidade, é provável que o cliente descubra defeitos no software. Manutenção corretiva modifica o software para corrigir defeitos.

- Adaptação: Com o tempo, o ambiente original (por exemplo, CPU, sistema operacional, regras do negócio, características externas do produto) para o qual o software foi desenvolvido provavelmente se modificará. Manutenção adaptativa resulta em modificações no software para acomodar mudanças no seu ambiente externo.

- Aperfeiçoamento: À medida que o software é usado, o cliente/usuário vai reconhecer funções adicionais que trarão benefício. Manutenção perfectiva aprimora o software além dos requisitos funcionais originais.

- Prevenção: O software de computador se deteriora devido a modificações, e, por causa disso, a manutenção preventiva, freqüentemente chamada reengenharia de software, deve ser implementada para permitir ao software servir às necessidades de seus usuários finais. Em essência, a manutenção preventiva faz modificações nos programas de computador, de modo que eles possam ser mais facilmente corrigidos, adaptados e melhorados.

No conceito de projeto de manutenção do IFPUG estão envolvidas apenas as manutenções prefectivas.

O cálculo dos pontos de função de um projeto de manutenção envolve a análise de três componentes:

- Funcionalidade da aplicação requisitada pelo usuário: consiste das funções incluídas, alteradas ou excluídas durante o projeto de manutenção.

- Funcionalidade de conversão requisitada pelo usuário: são as funções disponíveis somente no momento da instalação da aplicação para converter dados ou fornecer outros requisitos especificados pelo usuário. Após a instalação essas funções são descartadas.

- Valor do fator de ajuste: existem dois fatores de ajuste:

  - o fator de ajuste da aplicação antes que a manutenção se inicie.

  - o fator de ajuste da aplicação depois que o projeto de manutenção é concluído.

A seguinte fórmula é utilizada para calcular os pontos de função de um projeto de manutenção:

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

Onde

EFP é a contagem dos pontos de função de um projeto de manutenção.

ADD é a contagem dos pontos de função não ajustados das funções que serão incluídas pelo projeto de manutenção.

CHGA é a contagem dos pontos de função não ajustados das funções que serão alteradas pelo

projeto de manutenção. Este número reflete o tamanho das funções depois das alterações.
CFP é a contagem dos pontos de função das funções adicionadas pela conversão.
VAFA é o fator de ajuste depois do projeto de manutenção.
DEL é a contagem dos pontos de função não ajustados das funções que serão excluídas pelo projeto de manutenção.
VAFB é o fator de ajuste antes do projeto de manutenção.

Aplicação

Existem duas fórmulas para se calcular os pontos de função de uma aplicação:

- a primeira estabelece a contagem inicial dos pontos de função de uma aplicação.
- a segunda estabelece a contagem dos pontos de função de uma aplicação após um projeto de manutenção ter alterado a funcionalidade da aplicação.

i) **Fórmula para estabelecer a contagem inicial**

Inicialmente, o usuário está recebendo novas funcionalidades, e não há mudanças ou exclusões. A fórmula a seguir representa todas as funcionalidades requisitadas pelo usuário em uma aplicação instalada:

$$AFP = ADD * VAF$$

Onde
AFP é a contagem inicial dos pontos de função de uma aplicação.
ADD é a contagem dos pontos de função não ajustados das funções instaladas.
VAF é o valor do fator de ajuste.

ii) **Fórmula após o projeto de manutenção**

Quando um projeto de manutenção é concluído, a contagem dos pontos de função existente deve ser atualizada para refletir as modificações na aplicação. A funcionalidade da aplicação pode ser alterada de várias formas:

- a adição de funcionalidade aumenta o tamanho da aplicação.
- a alteração de funcionalidade aumenta, diminui, ou não tem efeito no tamanho da aplicação.

- a exclusão de funcionalidade diminui o tamanho da aplicação.

- mudanças no valor do fator de ajuste afetam a contagem dos pontos de função ajustados.

$$AFP = [(UFPB + ADD + CHGA) - (CHGB + DEL)] * VAFA$$

Onde

AFP é a contagem dos pontos de função ajustados da aplicação.

UFPB é a contagem dos pontos de função não ajustados da aplicação antes do início da manutenção.

ADD é a contagem dos pontos de função não ajustados das funções que foram instaladas pelo projeto de manutenção.

CHGA é a contagem dos pontos de função não ajustados das funções que foram alteradas pelo projeto de manutenção. Este número reflete o tamanho das funções antes das mudanças.

CHGB é a contagem dos pontos de função não ajustados das funções que foram alteradas pelo projeto de manutenção. Este número reflete o tamanho das funções depois que as mudanças foram efetuadas.

DEL é a contagem dos pontos de função não ajustados das funções que foram excluídas pelo projeto de manutenção.

VAFA é o fator de ajuste da aplicação depois que o projeto de manutenção é concluído.

3.2 Contagem dos Pontos de Função Segundo a NESMA

A NESMA reconhece três tipos de contagem de pontos de função:

- Detalhada
- Estimada
- Indicativa

A contagem detalhada é similar à utilizada pelo IFPUG. Segundo o próprio IFPUG, elas são similares em 95 % [42]. A contagem detalhada é determinada da seguinte forma:

- Identificar todos os tipos de funções (ALIs, AIEs, EEs, SEs, CEs).

- Determinar a complexidade de cada uma das funções.

- Calcular o total de pontos de função não ajustados.

As contagens estimada e indicativa (também referida como "método holandês") foram desenvolvidas pela NESMA para permitir uma contagem simplificada dos pontos de função.

A contagem estimada é realizada como segue:

- Identificar todos os tipos de funções (ALIs, AIEs, EEs, SEs, CEs).
- Determinar a complexidade de cada função de dado (ALI ou AIE) como baixa e a complexidade de cada função transacional (EE, SE, ou CE) como média.
- Calcular o total de pontos de função não ajustados.

A diferença entre a contagem detalhada e a contagem estimada, é que, na contagem estimada, a complexidade das funções é determinada por um valor pré-definido. Após a identificação de todos os tipos de funções (de dados e transacionais), os ALIs e AIEs são sempre classificados como de complexidade baixa, e as EEs, SEs, e CEs são sempre consideradas como de complexidade média.

Já a contagem indicativa é baseada somente no número de funções de dados (ALIs e AIEs), calculando o número de pontos de função não ajustados da seguinte forma:

$$ContagemIndicativa = (35 * NroDeALIs) + (15 * NroDeAIEs)$$

Nessa fórmula, é assumido que existem três entradas externas (para incluir, alterar, e excluir informações do ALI), duas saídas externas, e uma consulta para cada ALI; uma saída externa e uma consulta externa para cada AIE.

Os números 35 e 15 são obtidos considerando-se as funções transacionais como de complexidade média, e as funções de dados como de complexidade baixa. Além disso, são assumidos dois pontos de função a mais para os arquivos lógicos, e um ponto de função a mais para os arquivos de interface externa. Esses pontos de função assumidos são, segundo a NESMA, uma forma de ajustar a contagem. Os números 35 e 15 são obtidos como segue:

$$((3EEs) * 4) + ((2SEs) * 5) + ((1CE) * 4) + ((1ALI) * 7) - 2 = 35$$

$$((1EE) * 4) + ((1SE) * 5) + ((1AIE) * 5) + 1 = 15$$

3.2.1 Exemplo das Contagens Indicativa, Estimada e Detalhada

Esta subseção mostra um exemplo de uma aplicação simples que mantém dados do Consumidor e do Produto, e referencia dados do Fornecedor.

Baseando-se no fato de que quanto mais precisão o usuário quer, mais detalhes nos requisitos ele precisa, o exemplo apresenta os três métodos de contagem em ordem crescente de precisão:

- Contagem Indicativa
- Contagem Estimada
- Contagem Detalhada

Contagem Indicativa

Requisitos do Usuário:

- O usuário quer manter dados de Consumidor e Produto, e referenciar dados de Fornecedor.

Essa especificação é suficiente para se determinar uma contagem indicativa:

- Consumidor e Produto são ALIs.
- Fornecedor é um AIE.

Função de Dados	Tipo da Função	Pontos de Função
Consumidor	ALI	35
Produto	ALI	35
Fornecedor	AIE	15
Tamanho funcional Indicativo		85

Contagem Estimada

Para que uma contagem estimada seja efetuada, são necessárias, além das informações sobre as funções de dados, também as informações sobre as funções transacionais, o que implica num maior detalhamento dos requisitos.

Requisitos do Usuário:

- O usuário quer adicionar, alterar e apagar dados de Consumidor, quer realizar consultas nos dados de Consumidor, e também quer quatro tipos diferentes de relatórios, com cálculos, em Consumidor.
- O usuário quer adicionar, alterar e apagar dados de Produto, quer realizar consultas nos dados de Produto, e também quer um relatório, com cálculos, em Produto.
- O usuário quer realizar consultas em Fornecedor através do código do fornecedor e também quer um relatório com o total de fornecedores.

A partir da especificação mais detalhada apresentada acima, é possível realizar a contagem estimada:

Função de Dado ou Transacional	Tipo de Função	Complexidade	Pontos de Função
Consumidor	ALI	Baixa	7
Produto	ALI	Baixa	7
Fornecedor	AIE	Baixa	5
Adicionar Consumidor	EE	Média	4
Alterar Consumidor	EE	Média	4
Apagar Consumidor	EE	Média	4
Consulta em Consumidor	CE	Média	4
Relatório 1 em Consumidor	SE	Média	5
Relatório 2 em Consumidor	SE	Média	5
Relatório 3 em Consumidor	SE	Média	5
Relatório 4 em Consumidor	SE	Média	5
Adicionar Produto	EE	Média	4
Alterar Produto	EE	Média	4
Apagar Produto	EE	Média	4
Consulta em Produto	CE	Média	4
Relatório em Produto	SE	Média	5
Consulta em Fornecedor	CE	Média	4
Relatório em Fornecedor	SE	Média	5
Tamanho Funcional Estimado			85

Contagem Detalhada

Para que a contagem detalhada seja efetuada, é necessário identificar e classificar cada um dos tipos de função (ALI, AIE, EE, SE, CE) e também determinar a complexidade funcional de cada uma delas individualmente (baixa, média, complexa) baseando-se no números de DETs, RETs, e FTRs.

Devido a isso, os requisitos devem ser analisados em maiores detalhes, visando determinar

quais DETs e FTRs são utilizados pelas funções transacionais e de quais DETs e RETs as funções de dados são compostas.

A análise detalhada poderia resultar na seguinte contagem de pontos de função:

Função de Dados ou Transacional	Tipo de Função	Complexidade	Pontos de Função
Consumidor	ALI	Média	10
Produto	ALI	Baixa	7
Fornecedor	AIE	Baixa	5
Adicionar Consumidor	EE	Alta	6
Alterar Consumidor	EE	Média	4
Apagar Consumidor	EE	Baixa	3
Consulta em Consumidor	CE	Baixa	3
Relatório 1 em Consumidor	SE	Baixa	4
Relatório 2 em Consumidor	SE	Média	5
Relatório 3 em Consumidor	SE	Baixa	4
Relatório 4 em Consumidor	SE	Alta	7
Adicionar Produto	EE	Média	4
Alterar Produto	EE	Baixa	3
Apagar Produto	EE	Baixa	3
Consulta em Produto	CE	Média	4
Relatório em Produto	SE	Média	5
Consulta em Fornecedor	CE	Baixa	3
Relatório em Fornecedor	SE	Média	5
Tamanho Funcional Estimado			85

Neste caso particular, os três métodos apresentam o mesmo tamanho funcional, de 85 pontos de função. Geralmente os resultados não são exatamente os mesmos, mas são bastante próximos.

3.2.2 Estudo de Caso

A NESMA utilizou uma base com mais de cem aplicações implementadas para determinar a eficácia das contagens estimada e indicativa. Após as medições utilizando-se as três formas, foi possível observar uma boa aproximação entre os resultados das contagens. Entretanto, algumas vezes, o desvio da contagem indicativa era muito grande (em torno de 50%) quando comparado com as outras duas, demonstrando que ela deve ser utilizada com muito cuidado.

Várias empresas utilizam a contagem indicativa para obter uma contagem prematura e aproximada do tamanho do sistema. Segundo o NESMA, os números 35 e 15 funcionam de forma surpreendente com sistemas administrativos (MIS - *Management Information Systems*), mas para outros tipos de sistemas é necessário que a empresa utilize seus próprios fatores.

3.3 Considerações Finais

Neste Capítulo foram apresentados os métodos de contagem de pontos de função promovidos pelo IFPUG e pela NESMA. Baseado nos estudos e análises desses métodos elaborou-se um método de simplificação da contagem dos pontos de função para aplicativos web. O método é descrito no capítulo que se segue.

CAPÍTULO

4

Estudo de Caso

Neste capítulo são apresentadas as etapas de desenvolvimento do método simplificado.

4.1 Caracterização da Empresa

O método foi desenvolvido para a empresa *Linkway* [49]. Essa empresa está situada na cidade de São Carlos, interior de São Paulo, e possui também escritórios nas cidades de Araras, Descalvado, Leme, Limeira, Pirassununga, Porto Ferreira, Rio Claro e Santa Rita. A unidade de São Carlos, objeto deste estudo, possui 25 funcionários e é considerada uma pequena empresa, segundo o Ministério da Ciência e Tecnologia [15]. Ela é uma empresa prestadora de serviços que atua como provedor de Internet e também desenvolve aplicativos web. As tecnologias utilizadas para o desenvolvimento dos aplicativos web são PHP, HTML, Java, e banco de dados MySQL.

A *Linkway* participa de estudos realizados pelo ICMC (Instituto de Ciências Matemáticas e de Computação) já há algum tempo [4], [16], [58]. O objetivo da empresa é estabelecer métodos e ferramentas capazes de aumentar a qualidade do seu processo de desenvolvimento de software.

4.2 NESMA x IFPUG para aplicativos web

Inicialmente, a pesquisa tinha como foco a análise da possibilidade de se utilizar os métodos simplificados de contagem dos pontos de função elaborados pela NESMA (contagens indicativa e estimada) para a contagem dos pontos de função de aplicativos desenvolvidos para web. Para isso, uma comparação entre os métodos da NESMA e o método detalhado do IFPUG foi realizada.

Essa comparação foi elaborada através de um estudo de caso reunindo dados de vinte aplicativos web, obtidos junto à equipe de desenvolvimento, e cujas características são comuns à maioria dos softwares desenvolvidos pela empresa. Algumas dessas características são: criação de enquetes para o usuário, gerenciamento de usuários e notícias, geração de relatórios, gerenciamento de produtos, elaboração de carrinho de compras, formas de pagamento. O estudo de caso foi dividido em três etapas. As estapas podem ser visualizadas na figura 4.1:

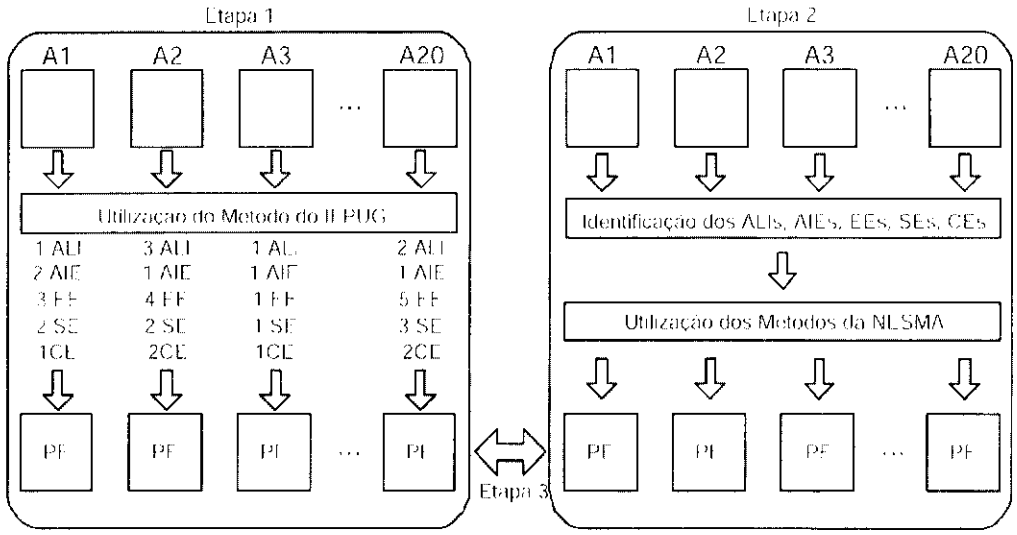


Figura 4.1: Etapas do Estudo de Caso

Etapa 1) Contagem dos pontos de função de cada um dos aplicativos utilizando o método detalhado elaborado pelo IFPUG: Na primeira etapa foram contados os pontos de função de cada uma das vinte aplicações utilizando-se o método detalhado elaborado pelo IFPUG. A contagem foi determinada com base na análise de requisitos dos aplicativos, nos aplicativos em si (todos os aplicativos analisados já haviam sido desenvolvidos) e também com o auxílio de um dos profissionais da empresa. O principal objetivo do auxílio desse profissional foi esclarecer as dúvidas relacionadas com a análise de requisitos de cada um dos

aplicativos e mostrar qual é a visão do usuário nos casos em que havia dúvidas. Além disso, as dúvidas ocorridas em relação ao processo de identificação dos pontos de função foram enviadas para o fórum do grupo BFPUG [3], e prontamente respondidas.

A análise e contagem dos pontos de função de cada uma das aplicações segundo o método detalhado elaborado pelo IFPUG mostrou-se de difícil execução. A maior dificuldade está no tempo perdido com a identificação dos DETs e RETs, o que numa pequena empresa, acaba se tornando decisivo para a rejeição da utilização do método. Os resultados da contagem são mostrados na tabela 4.1:

Aplicação 1 - 350 pf	Aplicação 11 - 168 pf
Aplicação 2 - 157 pf	Aplicação 12 - 265 pf
Aplicação 3 - 188 pf	Aplicação 13 - 111 pf
Aplicação 4 - 283 pf	Aplicação 14 - 248 pf
Aplicação 5 - 282 pf	Aplicação 15 - 131 pf
Aplicação 6 - 69 pf	Aplicação 16 - 274 pf
Aplicação 7 - 192 pf	Aplicação 17 - 240 pf
Aplicação 8 - 101 pf	Aplicação 18 - 251 pf
Aplicação 9 - 89 pf	Aplicação 19 - 206 pf
Aplicação 10 - 238 pf	Aplicação 20 - 163 pf

Tabela 4.1: Contagem dos PF Segundo o IFPUG

Etapa 2) Contagem dos pontos de função de cada um dos aplicativos utilizando os métodos simplificados (contagens indicativa e estimada) elaborados pela NESMA: Foram calculados os pontos de função de cada um dos aplicativos utilizando-se os princípios da contagem indicativa desenvolvida pela Nesma, ou seja, através da fórmula:

$$ContagemIndicativa = (35 * NroDeALIs) + (15 * NroDeAIEs)$$

O resultado dessa contagem pode ser visualizado na tabela 4.2:

Após isso, os pontos de função das aplicações foram calculados novamente, agora utilizando os princípios da contagem estimada desenvolvida pela Nesma, ou seja, contando todas as funções de dados como de complexidade funcional baixa, e todas as funções

Aplicação 1 - 470 pf	Aplicação 11 - 280 pf
Aplicação 2 - 245 pf	Aplicação 12 - 385 pf
Aplicação 3 - 280 pf	Aplicação 13 - 175 pf
Aplicação 4 - 385 pf	Aplicação 14 - 330 pf
Aplicação 5 - 435 pf	Aplicação 15 - 210 pf
Aplicação 6 - 105 pf	Aplicação 16 - 385 pf
Aplicação 7 - 245 pf	Aplicação 17 - 350 pf
Aplicação 8 - 175 pf	Aplicação 18 - 330 pf
Aplicação 9 - 140 pf	Aplicação 19 - 245 pf
Aplicação 10 - 365 pf	Aplicação 20 - 260 pf

Tabela 4.2: Contagem Indicativa dos PF

transacionais como de complexidade funcional média. O resultado é apresentado na tabela 4.3:

Aplicação 1 - 396 pf	Aplicação 11 - 202 pf
Aplicação 2 - 188 pf	Aplicação 12 - 316 pf
Aplicação 3 - 224 pf	Aplicação 13 - 130 pf
Aplicação 4 - 350 pf	Aplicação 14 - 277 pf
Aplicação 5 - 313 pf	Aplicação 15 - 160 pf
Aplicação 6 - 84 pf	Aplicação 16 - 330 pf
Aplicação 7 - 228 pf	Aplicação 17 - 289 pf
Aplicação 8 - 116 pf	Aplicação 18 - 283 pf
Aplicação 9 - 103 pf	Aplicação 19 - 229 pf
Aplicação 10 - 281 pf	Aplicação 20 - 199 pf

Tabela 4.3: Contagem Estimada dos PF

Etapa 3) Comparação entre as etapas 1 e 2: As contagens dos pontos de função utilizando o método detalhado do IFPUG e os métodos simplificados da NESMA (contagens indicativa e estimada) são apresentadas na figura 4.2.

Após a realização das contagens, houve uma comparação entre os resultados encontrados, mostrada na tabela 4.4. Nessa comparação, é possível observar que a contagem indicativa apresentou resultados muito ruins, com um erro variando entre no mínimo 19% (aplicação

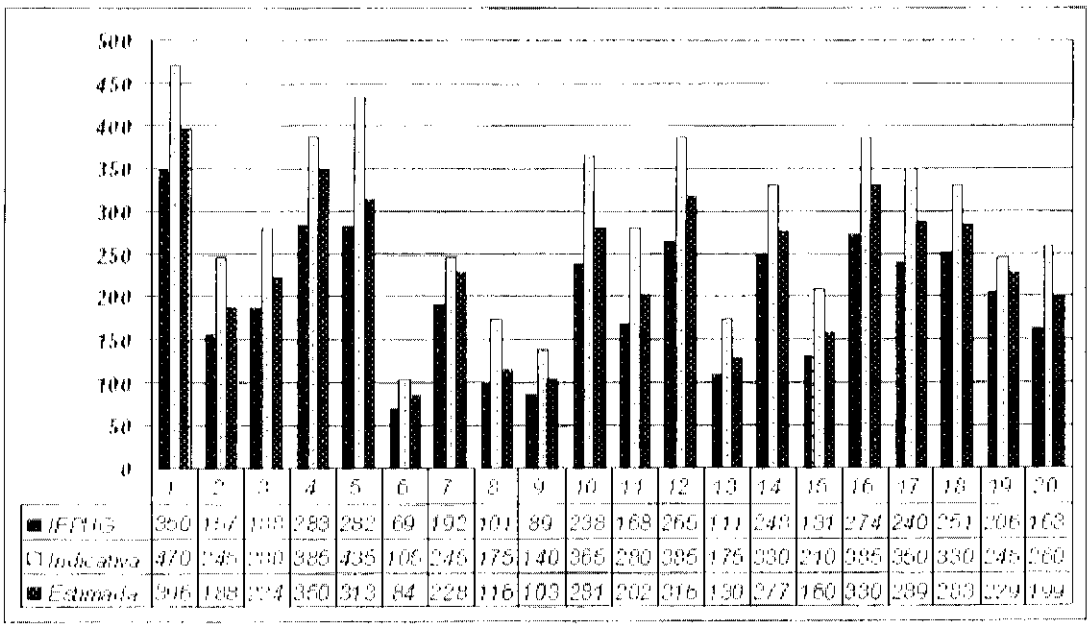


Figura 4.2: Comparação Entre o Método do IFPUG e os Métodos da NESMA

19) e no máximo 73% (aplicação 08). Em média, as contagens apresentaram um erro de 48%. Já a contagem estimada apresentou erros entre no mínimo 11% (aplicações 5 e 19) e no máximo 26% (aplicação 4). Em média os erros foram de 18%. Tanto para a contagem indicativa quanto para a contagem estimada, o número de pontos de função encontrados foi superior aos do método detalhado do IFPUG.

Esses resultados demonstraram que os métodos simplificados elaborados pela NESMA não podem ser aplicados para estimar o tamanho dos aplicativos web desenvolvidos na empresa em questão. Isso é observado pelo fato das contagens preconizadas pela NESMA apresentarem resultados muito diferentes dos apresentados pelo método detalhado do IFPUG.

	IFPUG	NESMA Indicativa	NESMA Estimada
1	350	34% (470)	13%(396)
2	157	56% (245)	20% (188)
3	188	49% (280)	19% (224)
4	283	39% (385)	26% (350)
5	282	54% (435)	11% (313)
6	69	52% (105)	22% (84)
7	192	28% (245)	19% (228)
8	101	73% (175)	15% (116)
9	89	57% (140)	16% (103)
10	238	53% (365)	18% (281)
11	168	67% (280)	20% (202)
12	265	45% (385)	19% (316)
13	111	58% (175)	17% (130)
14	248	33% (330)	12% (277)
15	131	60% (210)	22% (160)
16	274	41% (385)	20% (330)
17	240	46% (350)	20% (289)
18	251	31% (330)	13% (283)
19	206	19% (245)	11% (229)
20	163	60% (260)	22% (199)

Tabela 4.4: Comparação Entre os Métodos

4.3 Elaboração do Método Simplificado

Durante a realização do estudo comparativo, a análise das planilhas contendo a contagem dos pontos de função dos aplicativos mostrou que havia um predomínio de funções com complexidade funcional baixa. Isso motivou a tentativa de elaborar um método simplificado baseado em valores fixos para as complexidades dos ALIs, AIEs, EEs, SEs e CEs.

O método simplificado foi construído através de dois passos. No primeiro passo analisou-se como deveriam ser fixadas as complexidades das funções de dados (ALIs e AIEs). A análise dessas funções, identificadas durante a contagem dos pontos de função dos aplicativos segundo

o método detalhado do IFPUG, mostrou que elas apresentavam, em sua grande maioria, complexidade funcional baixa. Portanto, para a elaboração do método simplificado, a complexidade funcional dos ALIs e AIEs foi fixada como baixa.

O segundo passo foi analisar como deveriam ser fixadas as complexidades funcionais das funções transacionais (EEs, SEs, CEs). A análise mostrou que havia um predomínio da complexidade funcional baixa, mas que havia também várias funções com complexidade funcional média. Dessa forma, a solução encontrada foi determinar o número de pontos de função para cada uma das possíveis combinações das complexidades funcionais e analisar qual delas apresentava o melhor resultado.

As combinações foram criadas identificando cada entrada externa pela letra ***E***, cada saída externa pela letra ***S***, e cada consulta externa pela letra ***C***. Além disso, as complexidades funcionais foram identificadas pelas letras ***b*** para baixa, ***m*** para média, e ***a*** para alta. Foram excluídas dessa análise as combinações que possuíam mais de uma função como de complexidade funcional alta, e aquelas que não possuíam ao menos uma função com complexidade funcional baixa, pois essas combinações apresentavam resultados muito distantes dos encontrados pelo método detalhado.

Na tabela 4.5 são apresentados os pontos de função encontrados para cada uma das combinações, analisando os vinte aplicativos.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
IFPUG	350	157	188	283	282	69	192	101	89	238	168	265	111	248	131	274	240	251	206	163
E=b;S=b;C=b	324	154	183	283	258	69	185	97	85	230	166	257	107	227	131	267	235	232	186	163
E=b;S=m;C=b	336	157	187	288	262	72	192	102	88	232	168	260	110	236	133	268	238	243	194	164
E=b;S=a;C=b	360	163	195	298	270	78	206	112	94	236	172	266	116	254	137	270	244	265	210	166
E=b;S=a;C=m	391	179	215	332	298	84	226	118	102	259	188	294	125	274	153	305	272	284	227	186
E=b;S=b;C=m	355	170	203	317	286	75	205	103	93	253	182	285	116	247	147	302	263	251	203	183
E=b;S=m;C=m	367	173	207	322	290	78	212	108	96	255	184	288	119	256	149	303	266	262	211	184
E=a;S=b;C=m	439	215	254	401	355	93	235	127	114	331	236	369	149	310	180	383	332	314	257	228
E=a;S=b;C=b	408	199	234	367	327	87	221	121	106	308	220	341	140	290	164	348	304	295	240	208
E=a;S=m;C=b	420	202	238	372	359	90	228	126	109	310	222	344	143	299	166	349	307	306	248	209
E=m;S=m;C=b	366	172	204	316	313	78	208	110	95	258	186	288	121	257	144	295	261	264	212	179
E=m;S=a;C=b	388	178	212	326	321	84	222	120	101	262	190	294	127	275	148	297	267	286	228	181
E=b;S=m;C=a	429	205	247	390	346	90	252	120	112	301	216	344	137	296	181	373	322	300	245	224
E=b;S=b;C=a	417	202	243	385	342	87	245	115	109	299	214	341	134	287	179	372	319	289	237	223
E=m;S=b;C=a	445	217	260	413	365	93	261	123	116	325	232	369	145	308	190	399	342	310	255	238
E=m;S=b;C=b	352	169	200	311	281	75	201	105	92	256	184	285	118	248	142	294	258	253	204	178
E=m;S=b;C=m	383	185	220	345	309	81	221	111	100	279	200	313	127	268	158	329	286	272	221	198

Tabela 4.5: Análise das Combinações das Complexidades Funcionais

Análises estatísticas foram realizadas com os dados da tabela 4.5 no intuito de determinar qual das combinações apresentava os resultados mais próximos daqueles encontrados pelo método detalhado do IFPUG. O software estatístico MiniTab versão 14 [52] foi utilizado para apoiar a execução das análises estatísticas.

O primeiro teste realizado foi o de análise de variância (ANOVA - *Analysis Of VAriance*). Com esse teste, foi verificado se havia diferença entre empregar uma ou outra combinação para estimar os pontos de função de forma simplificada. Caso não houvesse diferença, o resultado encontrado seria sempre o mesmo, independente da combinação utilizada.

Assim, o teste ANOVA verifica se há diferenças entre um conjunto de amostras fornecidas. A resposta é fornecida pelo resultado do *p-valor*. O *p-valor* indica a probabilidade de a hipótese *'Há diferença entre as combinações'* ser rejeitada. Quanto menor o *p-valor*, menor a probabilidade de que a hipótese seja rejeitada. É importante ressaltar, porém, que o teste ANOVA não identifica qual das amostras produz o melhor ou o pior resultado quando comparada com as outras amostras, ele apenas indica se há diferenças entre as amostras.

Para a realização do teste de análise de variância, foram utilizados os erros obtidos pelas combinações em relação ao método detalhado. Como exemplo, na tabela 4.6, são apresentados os erros encontrados quando a combinação (E=b;S=b;C=b) foi utilizada.

Aplicação 1 - 23 pf	Aplicação 11 - 2 pf
Aplicação 2 - 3 pf	Aplicação 12 - 8 pf
Aplicação 3 - 5 pf	Aplicação 13 - 4 pf
Aplicação 4 - 0 pf	Aplicação 14 - 21 pf
Aplicação 5 - 24 pf	Aplicação 15 - 0 pf
Aplicação 6 - 0 pf	Aplicação 16 - 7 pf
Aplicação 7 - 7 pf	Aplicação 17 - 5 pf
Aplicação 8 - 4 pf	Aplicação 18 - 19 pf
Aplicação 9 - 4 pf	Aplicação 19 - 20 pf
Aplicação 10 - 8 pf	Aplicação 20 - 0 pf

Tabela 4.6: Erros Gerados pela Combinação E=b;S=b;C=b

Os erros foram transformados em valores percentuais. Para isso, cada contagem determinada pelas combinações para cada um dos aplicativos foi subtraída do número encontrado pelo método detalhado. O resultado foi dividido pelo valor do método detalhado. A operação foi

realizada em módulo para garantir que se trabalhasse com números positivos. Ex: para a aplicação 1, a combinação (E=b;S−b;C−b) identificou 324 pontos de função e o método detalhado identificou 350 pontos de função. Portanto:

$$|(324\text{-}350)/350| = |(-26)/350| = 0{,}074285$$

O procedimento para encontrar os erros das outras combinações foi idêntico.

Uma vez determinados os dados de entrada, o teste ANOVA foi realizado. O resultado do teste é mostrado na tabela 4.7.

	Graus de Liberdade	Soma dos Quadrados	Quadrado Médio	F	p-valor
fator	15	3,71357	0,24757	104,70	0,000
Erro	304	0,71884	0,00236		
Total	319	4,43241			

Tabela 4.7: One-way ANOVA

O resultado do teste ANOVA indicou que havia diferença significativa entre as combinações (p-valor é próximo de zero). Na figura 4.3 é apresentado um gráfico contendo os resultados da análise de variância.

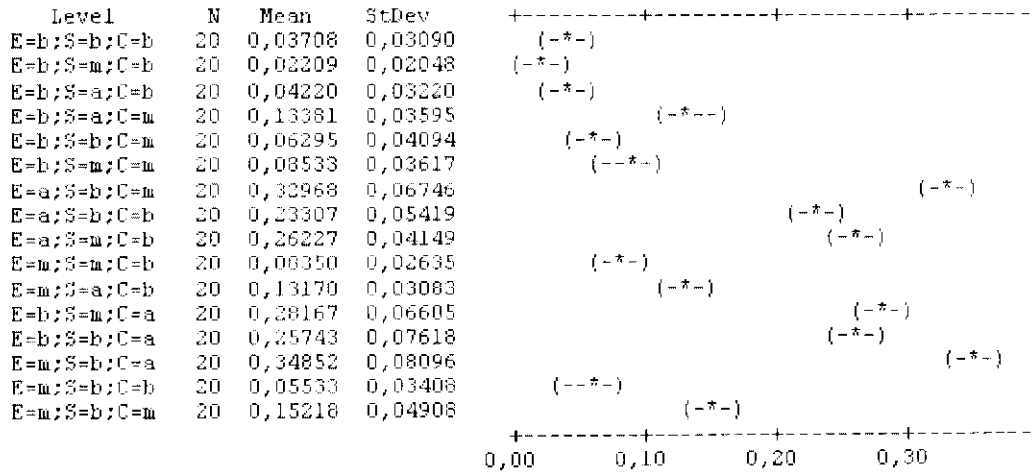


Figura 4.3: Gráfico One-way ANOVA

Na figura 4.3, *Level* são as combinações, *N* o número de aplicações, *Mean* o erro médio e *StDev* o desvio padrão. É possível observar na figura 4.3 que as combinações não estão em grupos iguais, o que é representado pelo fato de elas terem ficado em intervalos diferentes. Essa é a forma gráfica de se observar que as combinações apresentam resultados distintos.

Portanto, o teste de análise de variância identificou diferenças entre as combinações. Logo, é possível classificá-las de modo a determinar qual delas apresenta a contagem de pontos de função mais próxima da obtida pelo método detalhado.

O teste do método de múltiplas comparações com o melhor, de Hsu (Hsu's MCB - *Multiple Comparisons with the Best*) [5], [6], é capaz de determinar qual das combinações é a melhor. Como o objetivo é encontrar aquela que apresenta o menor erro, deve-se utilizar o método com a opção *'Smallest is best'* (o método que apresenta o menor erro é o melhor) no MiniTab.

Com essa configuração, o método Hsu's MCB determina a melhor combinação procurando qual delas apresenta um intervalo de confiança que contenha o número zero. Na figura 4.4 os resultados do teste Hsu's MCB são apresentados.

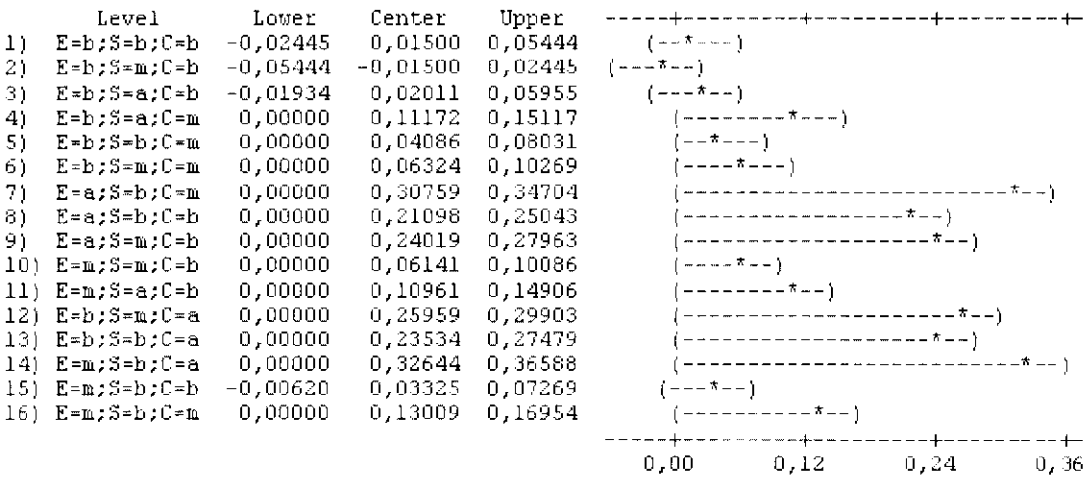


Figura 4.4: Gráfico Hsu's MCB

As combinações (E=b;S=b;C=b), (E=b;S=m;C=b), (E=b;S=a;C=b) e (E=m;S=b;C=b), indicadas pelos números 1), 2), 3) e 15), respectivamente, são as melhores, pois possuem um intervalo de confiança incluindo o valor zero. As outras combinações apresentam valores estritamente positivos. Portanto, estatisticamente, as combinações (E=b;S=b;C=b), (E=b;S=m;C=b), (E=b;S=a;C=b) e (E=m;S=b;C=b) são as melhores e podem ser utilizadas como uma forma simplificada do método detalhado elaborado pelo IFPUG.

Após a identificação das melhores combinações, dentro do conjunto analisado, elaborou-se o ajuste para cada uma delas através da análise de regressão. O ajuste permite que futuras contagens apresentem um resultado próximo do ideal, que seria obtido caso a contagem detalhada do IFPUG fosse realizada. Nas figuras 4.5, 4.6, 4.7 e 4.8 são apresentados os gráficos de dispersão para as combinações (E=b;S=b;C=b), (E=b;S=m;C=b), (E=b;S=a;C=b) e (E=m;S=b;C=b).

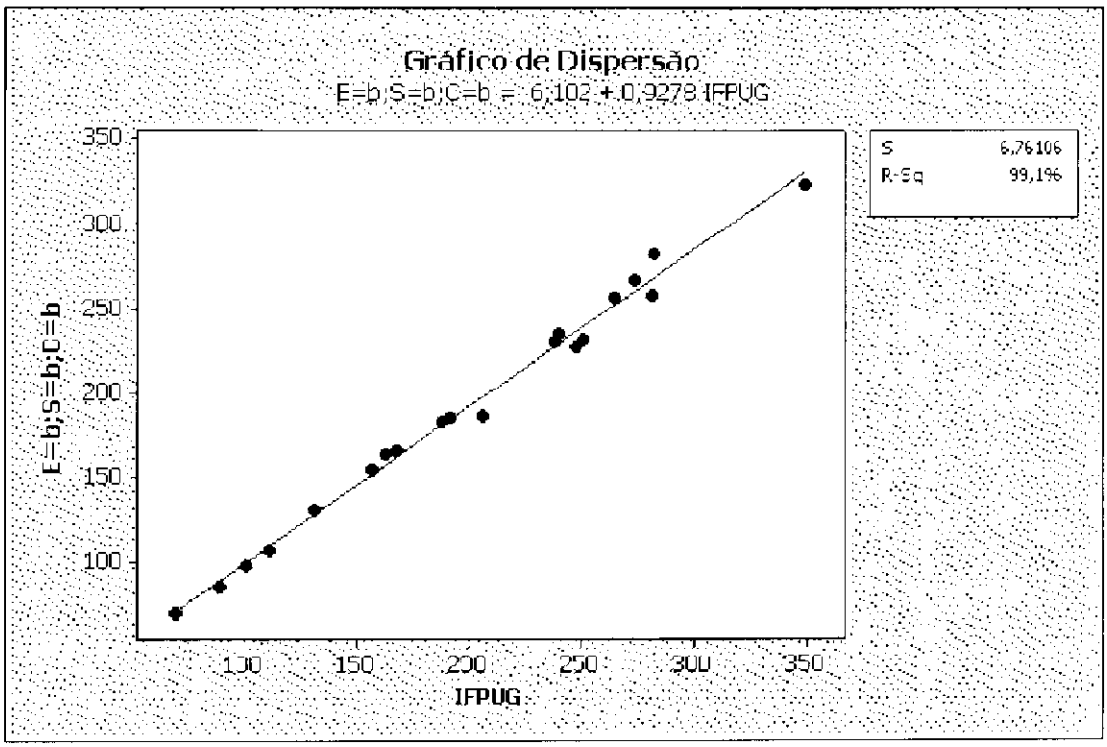


Figura 4.5: Gráfico de Dispersão (E=b;S=b;C=b)

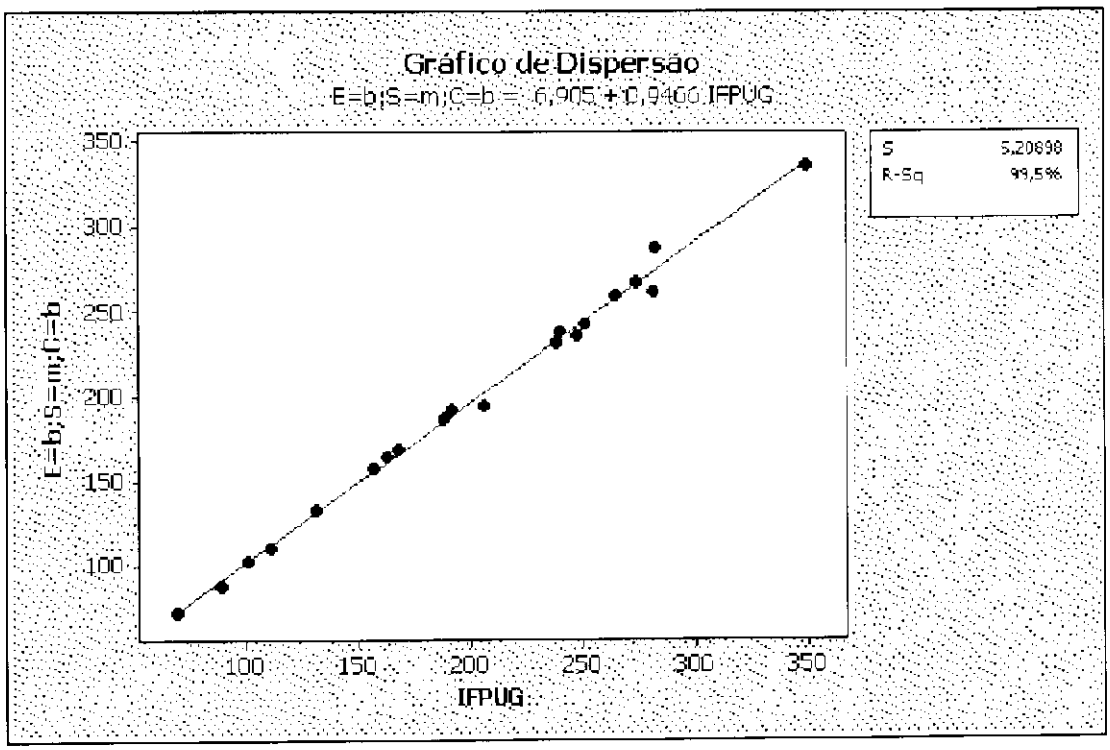


Figura 4.6: Gráfico de Dispersão (E=b;S=m;C=b)

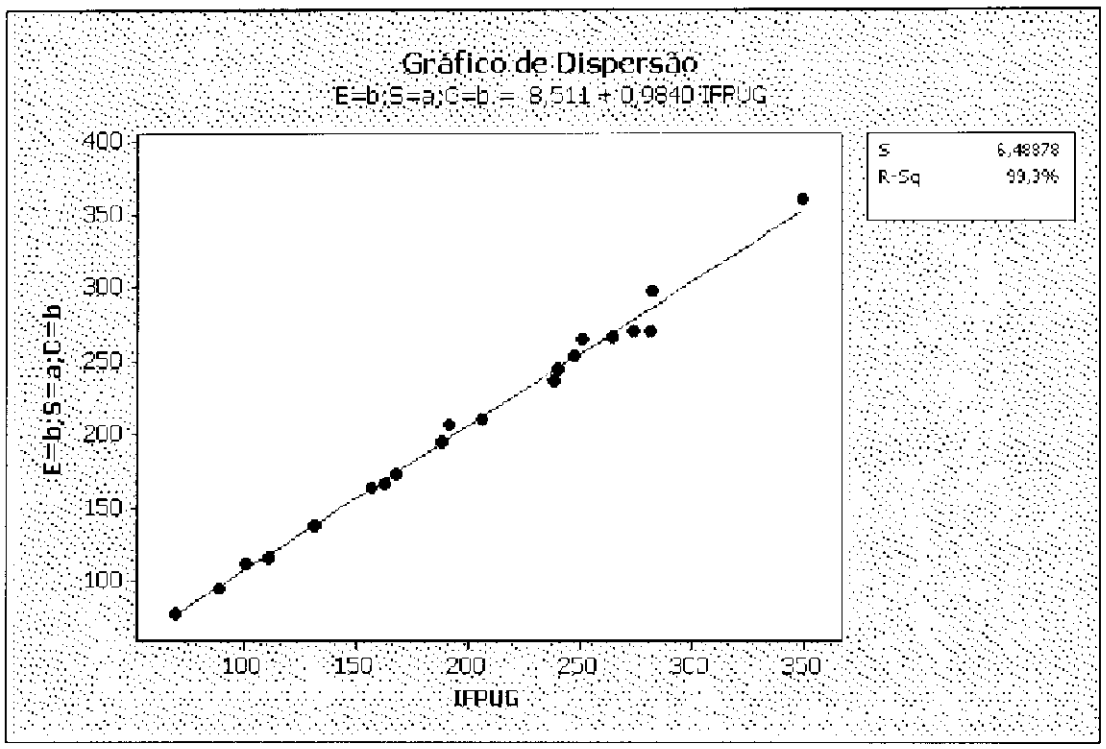


Figura 4.7: Gráfico de Dispersão (E=b;S=a;C=b)

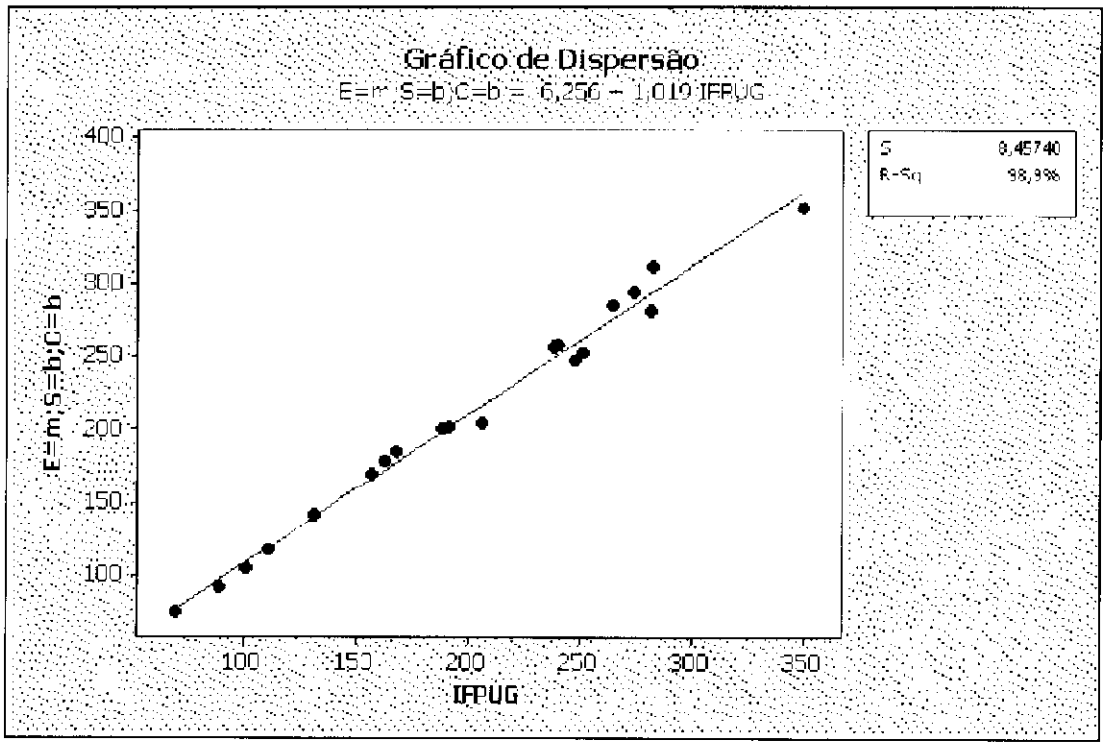


Figura 4.8: Gráfico de Dispersão (E=m;S=b;C=b)

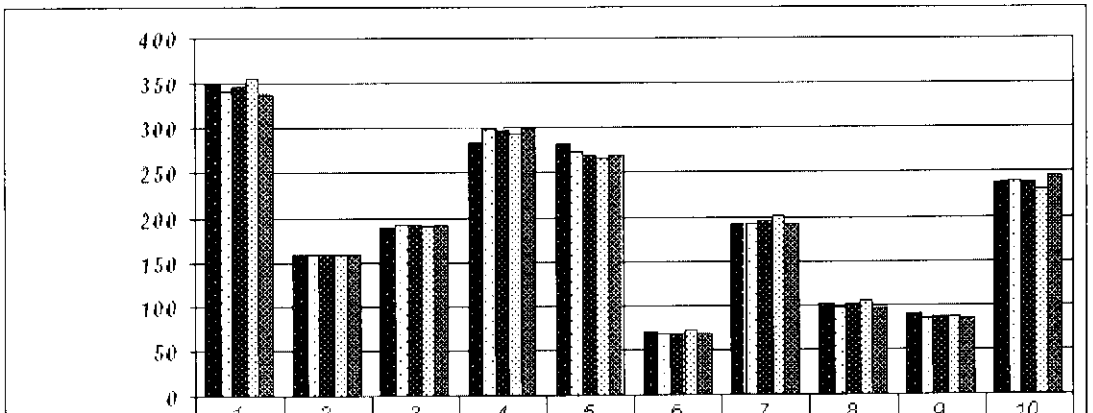
Pode-se observar nas figuras 4.5, 4.6, 4.7 e 4.8, que as combinações apresentam um ajuste muito bom em relação à reta (R-Sq alto). A seguir, na tabela 4.8, é exibido o ajuste que deve ser utilizado para cada uma das combinações.

Combinação	Ajuste
1) E=b;S=b;C=b	PF = (valor - 6,102)/0,9278
2) E=b;S=m;C=b	PF = (valor - 6,905)/0,9466
3) E=b;S=a;C=b	PF = (valor - 8,511)/0,9840
15) E=m;S=b;C=b	PF = (valor - 6,256)/1,019

Tabela 4.8: Ajuste para as Combinações

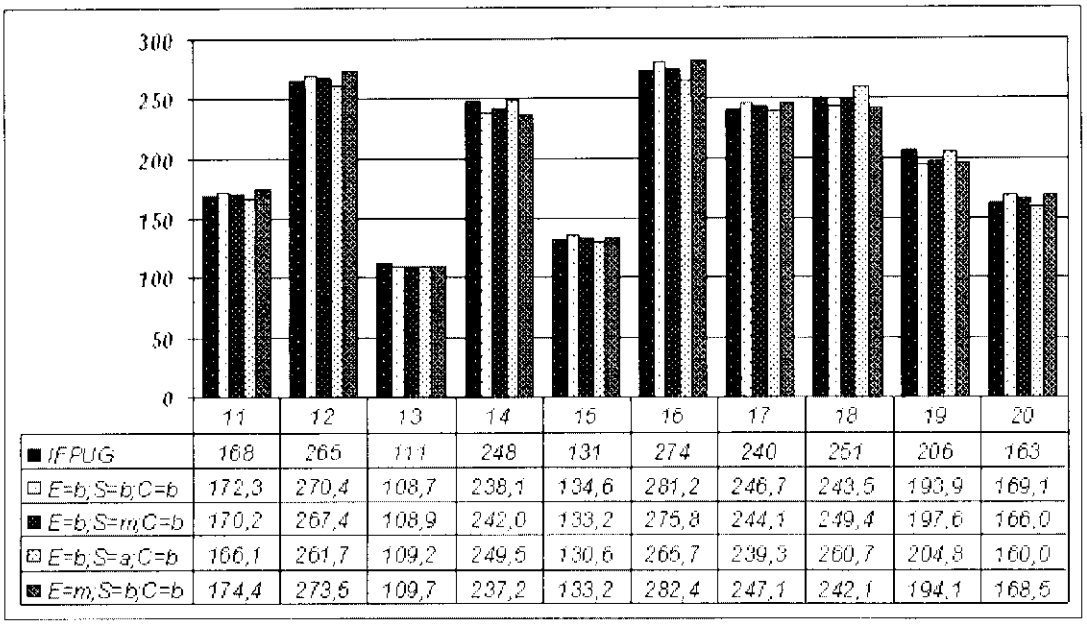
Assim, contados os pontos de função utilizando uma das combinações apresentadas na tabela 4.8, deve-se subtrair uma constante do número encontrado (valor) e, em seguida, dividir o resultado por outra constante.

As quatro combinações, depois de utilizado o ajuste, apresentaram, em média, erros de 2,8% (E=b;S=b;C=b), 1,8% (E=b;S=m;C=b), 2,2% (E=b;S=a;C=b) e 3,2% (E=m;S=b;C=b). Nas figuras 4.9 e 4.10, gráficos mostram os pontos de função obtidos por essas combinações em comparação com a contagem detalhada. Na figura 4.9 são exibidas as contagens dos dez primeiros aplicativos e na figura 4.10 as contagens dos outros dez.



	1	2	3	4	5	6	7	8	9	10
IFPUG	350	157	188	283	282	69	192	101	89	238
E=b;S=b;C=b	342,6	159,4	190,7	298,4	271,5	67,8	192,8	98,0	86,0	241,3
E=b;S=m;C=b	347,7	158,6	190,3	297,0	269,5	68,8	195,5	100,5	86,7	237,8
E=b;S=a;C=b	357,2	157,0	189,5	294,2	265,7	70,6	200,7	105,2	86,9	231,2
E=m;S=b;C=b	339,3	159,7	190,1	299,1	269,6	67,5	191,1	96,9	84,1	245,1

Figura 4.9: Total de Pontos de Função (aplicações 1 a 10)



	11	12	13	14	15	16	17	18	19	20
IFPUG	168	265	111	248	131	274	240	251	206	163
E=b,S=b,C=b	172,3	270,4	108,7	238,1	134,6	281,2	246,7	243,5	193,9	169,1
E=b,S=m,C=b	170,2	267,4	108,9	242,0	133,2	275,8	244,1	249,4	197,6	166,0
E=b,S=a,C=b	166,1	261,7	109,2	249,5	130,6	266,7	239,3	260,7	204,8	160,0
E=m,S=b,C=b	174,4	273,6	109,7	237,2	133,2	282,4	247,1	242,1	194,1	168,5

Figura 4.10: Total de Pontos de Função (aplicações 11 a 20)

Os resultados observados nos gráficos (figuras 4.9 e 4.10) comprovam a possibilidade de utilização dessas combinações para estabelecer a contagem dos pontos de função de um aplicativo de forma simplificada.

4.4 Refinamento do Método Simplificado

Após as análises que determinaram os quatro possíveis métodos de simplificação, os aplicativos foram analisados novamente na tentativa de identificar alguma característica que colaborasse para refinar a análise anterior. Assim, comparando os requisitos dos aplicativos, foi possível dividi-los em dois grupos, de acordo com as funcionalidades que normalmente são implementadas quando cada aplicativo é desenvolvido.

O grupo 1, formado pelas aplicações 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17 e 20 (tabela 4.5), possui funcionalidades como a criação de enquetes para o usuário, gerenciamento de usuários e notícias, e geração de relatórios. Já o grupo 2, formado pelas aplicações 1, 5, 14, 18, e 19 (tabela 4.5), possui, além das funcionalidades presentes nas aplicações do grupo 1, gerenciamento de produtos, carrinho de compras, formas de pagamento. A maneira como cada aplicativo de um determinado grupo é construído é bastante semelhante. Isso induziu a verificação do comportamento de cada um dos grupos em separado.

Os passos realizados para a análise do comportamento dos grupos em separado são idênticos

aos utilizados para a análise de todos os aplicativos, exibida anteriormente. Portanto, primeiro o teste ANOVA é realizado, e caso as combinações sejam diferentes, o teste Hsu's MCB é realizado. Em seguida, os ajustes são encontrados através do teste de regressão.

Para os aplicativos do grupo 1, o resultado do teste ANOVA é apresentado na tabela 4.9.

	Graus de Liberdade	Soma dos Quadrados	Quadrado Médio	F	P-valor
fator	15	3,31749	0,22117	138,50	0,000
Erro	224	0,35770	0,00160		
Total	239	3,67519			

Tabela 4.9: One-way ANOVA (Grupo 1)

Na figura 4.11, o resultado é exibido graficamente.

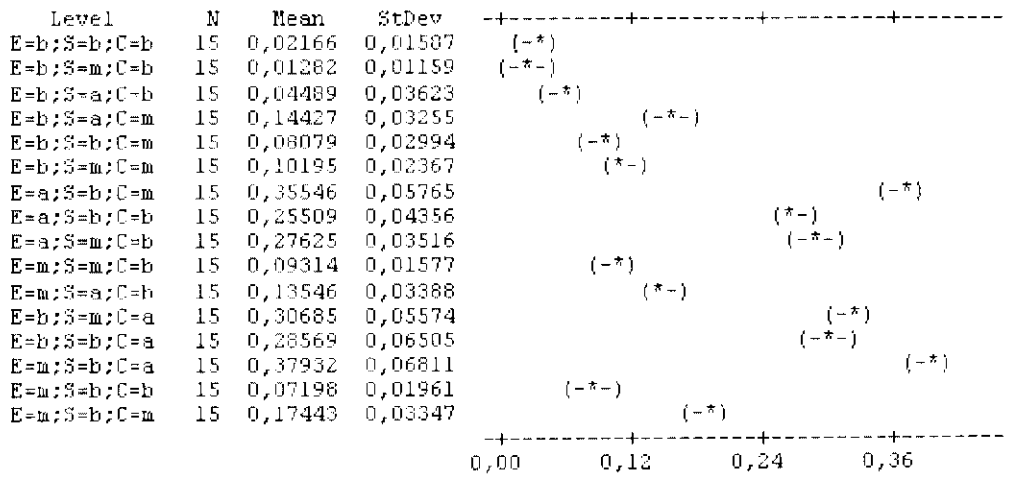


Figura 4.11: Gráfico One-way ANOVA (Grupo 1)

Segundo o resultado do teste ANOVA (tabela 4.9), as combinações são diferentes (p-valor próximo de zero), permitindo a realização do teste Hsu's MCB. Os resultados desse teste são apresentados na figura 4.12.

O teste Hsu's MCB indicou como as combinações (E=b;S=b;C=b), (E=b;S=m;C=b) e (E=b; S=a;C=b), respectivamente números 1), 2) e 3) como as melhores. A obtenção dos ajustes é mostrada nas figuras 4.13, 4.14 e 4.15.

```
      Level          Lower    Center    Upper   ----+---------+---------+---------+----
1)  E=b;S=b;C=b  -0,02886  0,00883  0,04653     (--*--)
2)  E=b;S=m;C=b  -0,04653 -0,00883  0,02886  (--*--)
3)  E=b;S=a;C=b  -0,00562  0,03207  0,06976       (--*--)
4)  E=b;S=a;C=m   0,00000  0,13145  0,16915       (----------*--)
5)  E=b;S=b;C=m   0,00000  0,06797  0,10566       (-----*--)
6)  E=b;S=m;C=m   0,00000  0,08913  0,12682       (------*---)
7)  E=a;S=b;C=m   0,00000  0,34263  0,38033       (----------------------------*--)
8)  E=a;S=b;C=b   0,00000  0,24327  0,27996       (-------------------*--)
9)  E=a;S=m;C=b   0,00000  0,26343  0,30112       (---------------------*--)
10) E=m;S=m;C=b   0,00000  0,08032  0,11801       (-----  *--)
11) E=m;S=a;C=b   0,00000  0,12264  0,16034       (---------*--)
12) E=b;S=m;C=a   0,00000  0,29403  0,33172       (-----------------------*--)
13) E=b;S=b;C=a   0,00000  0,27287  0,31056       (----------------------*--)
14) E=m;S=b;C=a   0,00000  0,36650  0,40420       (-------------------------------*--)
15) E=m;S=b;C=b   0,00000  0,05916  0,09685       (----*--)
16) E=m;S=b;C=m   0,00000  0,16161  0,19930       (-----------*---)
                                                ----+---------+---------+---------+----
                                                  0,00      0,12      0,24      0,36
```

Figura 4.12: Gráfico Hsu's MCB (Grupo 1)

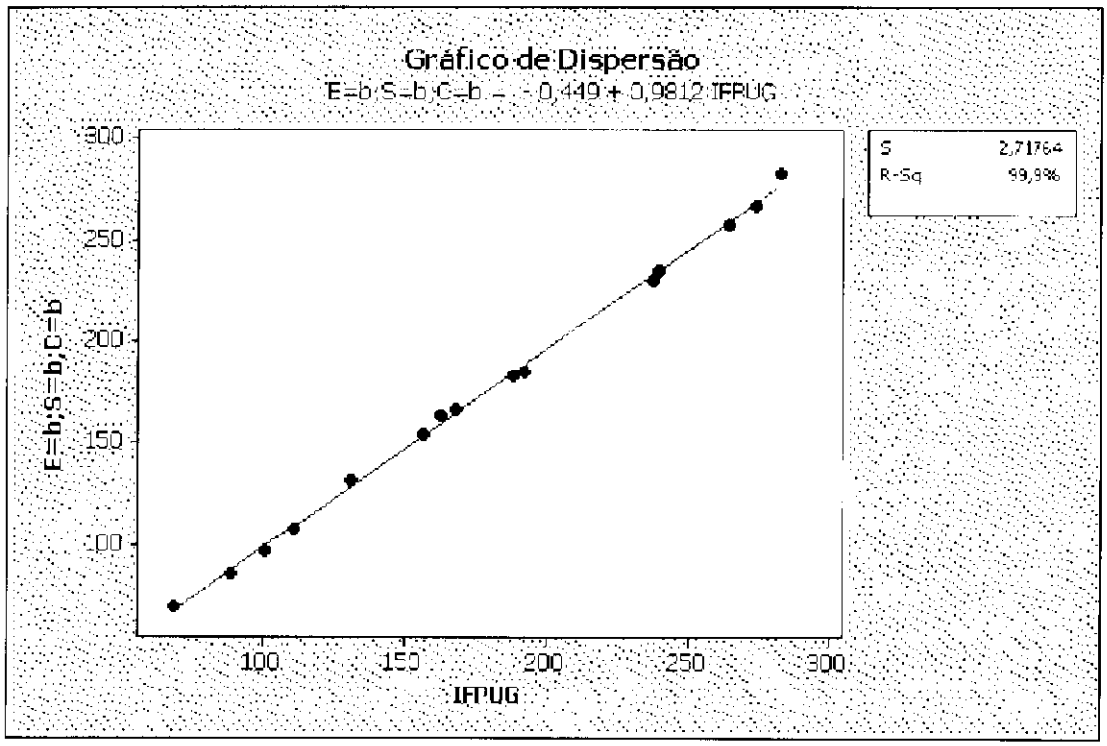


Figura 4.13: Gráfico de Dispersão (E=b;S=b;C=b) para o Grupo 1

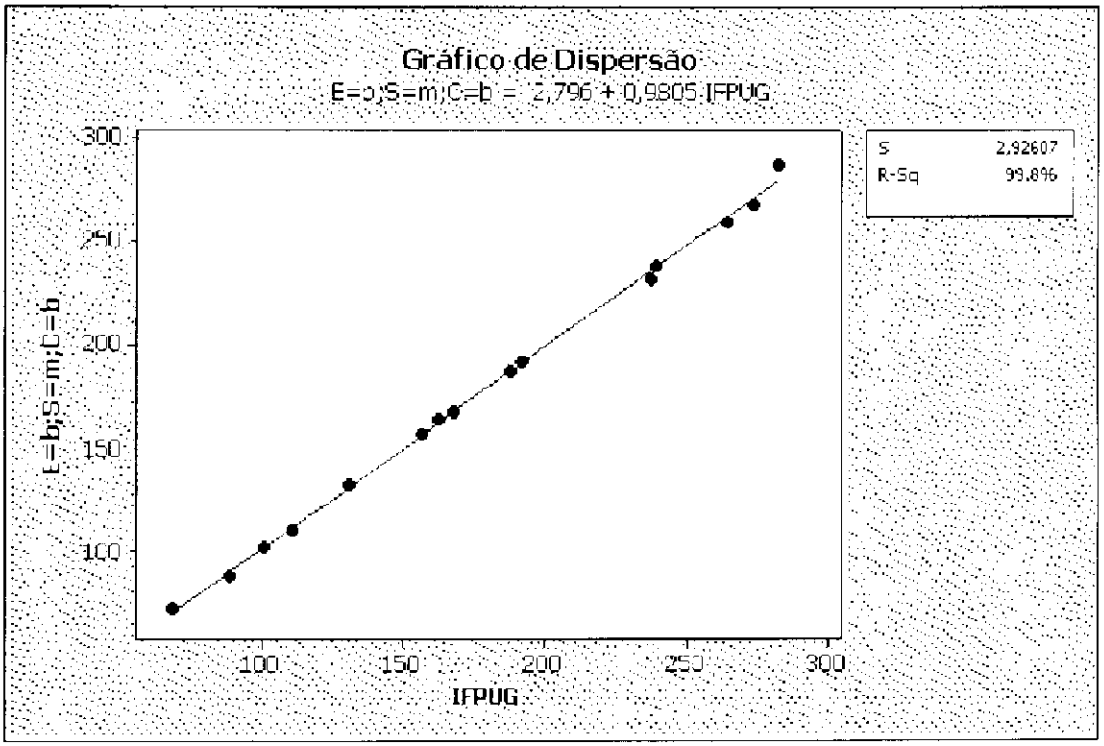


Figura 4.14: Gráfico de Dispersão (E=b;S=m;C=b) para o Grupo 1

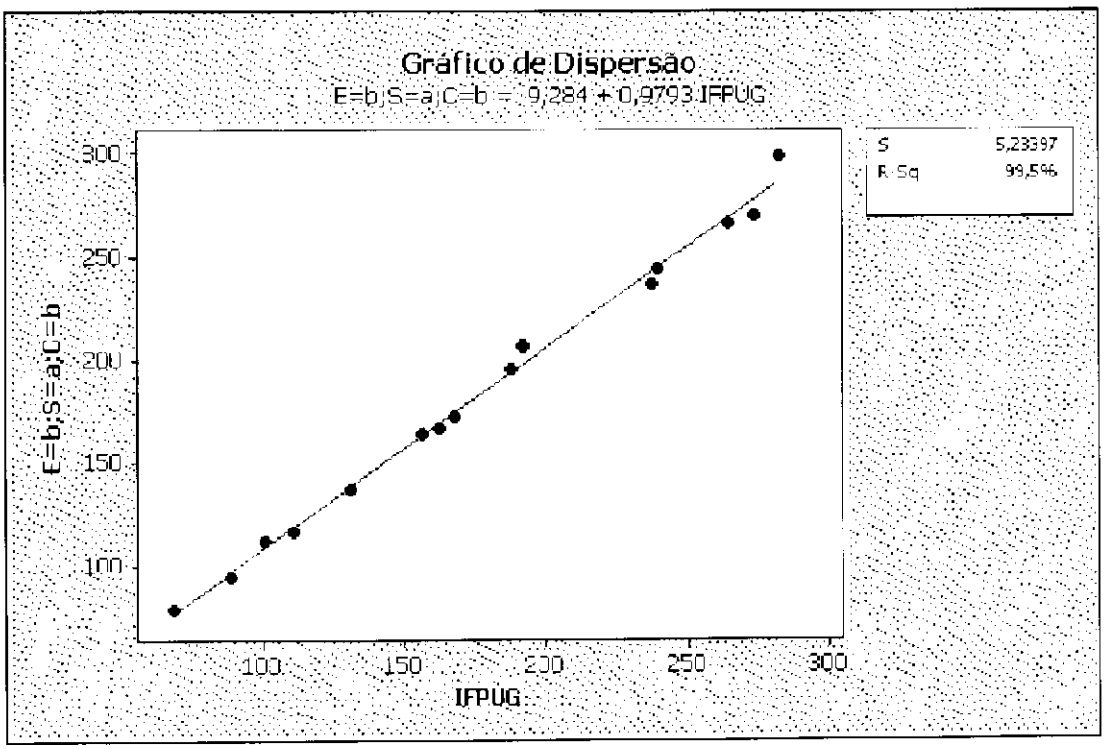


Figura 4.15: Gráfico de Dispersão (E=b;S=a;C=b) para o Grupo 1

Na tabela 4.10 é apresentado o ajuste que deve ser utilizado para cada uma das combinações.

Combinação	Ajuste
1) E=b;S=b;C=b	PF = (valor + 0,449)/0,9812
2) E=b;S=m;C=b	PF = (valor - 2,796)/0,9805
3) E=b;S=a;C=b	PF = (valor - 9,284)/0,9793

Tabela 4.10: Ajuste para as Combinações do Grupo 1

Os pontos de função determinados por essas combinações são exibidos na figura 4.16. As combinações apresentaram, em média, erros de 1,4% (E=b;S=b;C=b), 1,1% (E=b;S=m;C=b) e 2% (E=b;S=a;C=b).

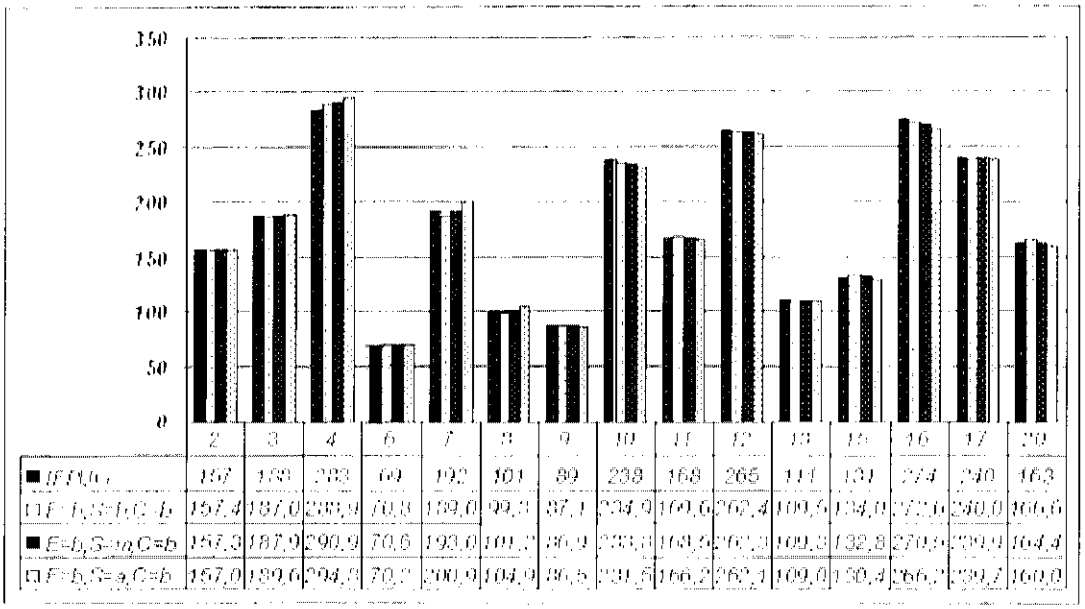


Figura 4.16: Pontos de Função - Melhores Combinações para o Grupo 1

Para os aplicativos do grupo 2, o teste ANOVA determinou os resultados exibidos na tabela 4.11.

	Graus de Liberdade	Soma dos Quadrados	Quadrado Médio	F	P-valor
fator	15	0,550936	0,036729	103,50	0,000
Erro	64	0,022712	0,000355		
Total	79	0,573648			

Tabela 4.11: One-way ANOVA (Grupo 2)

Na figura 4.17, o resultado é apresentado graficamente.

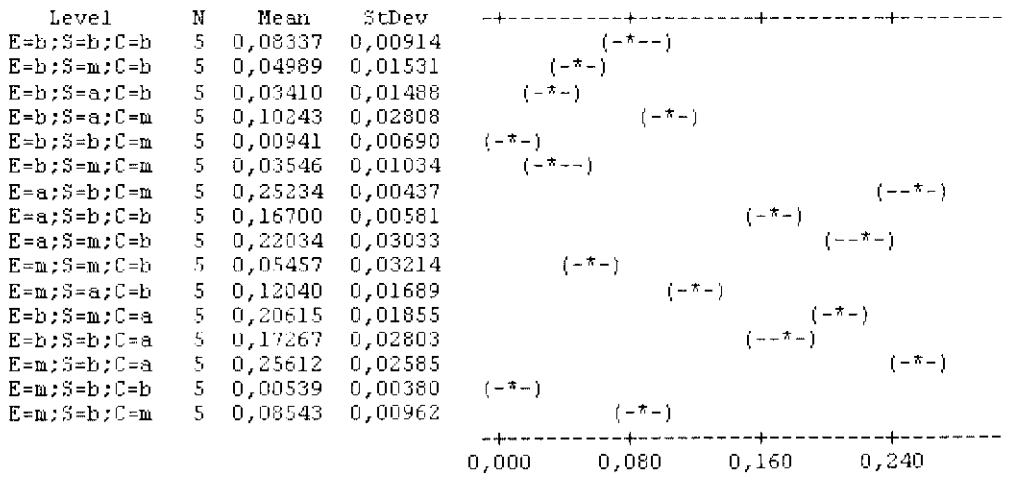


Figura 4.17: Gráfico One-way ANOVA (Grupo 2)

Como há diferença entre as combinações (p-valor próximo de zero), o teste Hsu's MCB foi realizado para encontrar as melhores combinações para os aplicativos do grupo 2. Os resultados desse teste são mostrados na figura 4.18.

O teste Hsu´s MCB indicou as combinações (E=b;S=a;C=b), (E=b;S=b;C=m), (E=b;S=m;C=m) e (E=m;S=b;C=b), respectivamente números 3), 5), 6) e 15), como as melhores. A obtenção dos ajustes é mostrada nas figuras 4.19, 4.20, 4.21 e 4.22.

	Level	Lower	Center	Upper
1)	E=b;S=b;C=b	0,00000	0,07798	0,10930
2)	E=b;S=m;C=b	0,00000	0,04450	0,07581
3)	E=b;S=a;C=b	-0,00260	0,02872	0,06003
4)	E=b;S=a;C=m	0,00000	0,09704	0,12835
5)	E=b;S=b;C=m	-0,02729	0,00403	0,03534
6)	E=b;S=m;C=m	-0,00124	0,03007	0,06138
7)	E=a;S=b;C=m	0,00000	0,24696	0,27827
8)	E=a;S=b;C=b	0,00000	0,16161	0,19292
9)	E=a;S=m;C=b	0,00000	0,21495	0,24627
10)	E=m;S=m;C=b	0,00000	0,04918	0,08050
11)	E=m;S=a;C=b	0,00000	0,11501	0,14632
12)	E=b;S=m;C=a	0,00000	0,20076	0,23208
13)	E=b;S=b;C=a	0,00000	0,16728	0,19859
14)	E=m;S=b;C=a	0,00000	0,25074	0,28205
15)	E=m;S=b;C=b	-0,03534	-0,00403	0,02729
16)	E=m;S=b;C=m	0,00000	0,08004	0,11136

0,000 0,080 0,160 0,240

Figura 4.18: Gráfico Hsu's MCB (Grupo 2)

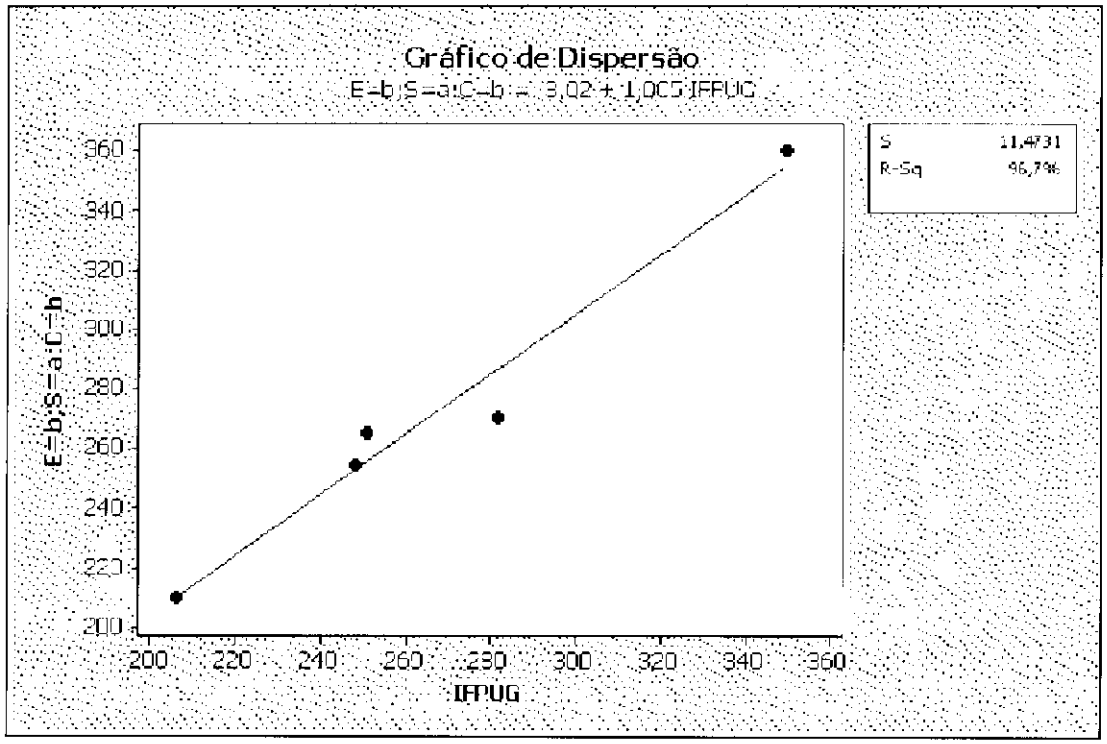


Figura 4.19: Gráfico de Dispersão (E=b;S=a;C=b) para o Grupo 2

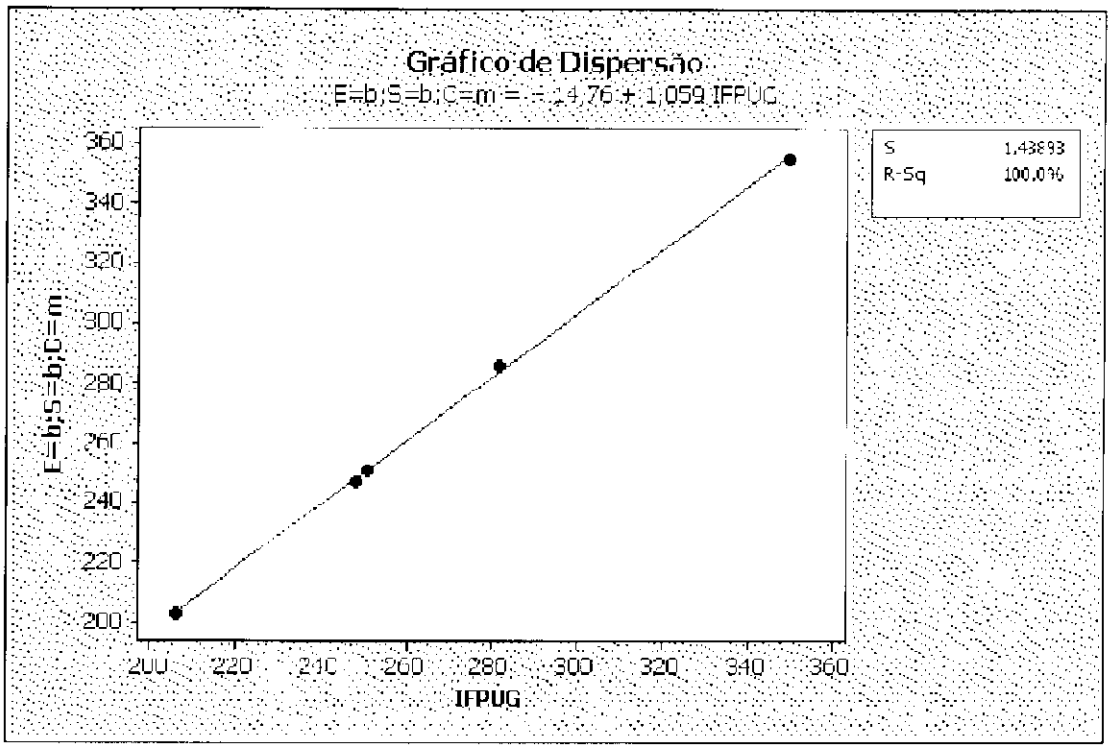


Figura 4.20: Gráfico de Dispersão (E=b;S=b;C=m) para o Grupo 2

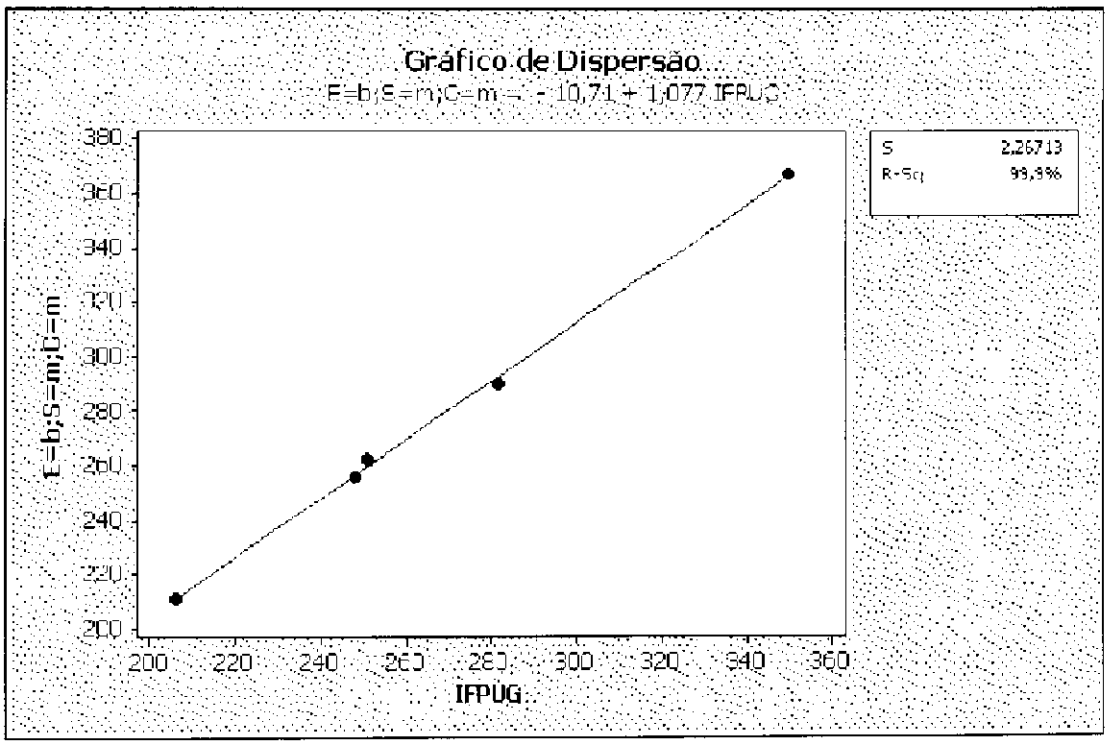


Figura 4.21: Gráfico de Dispersão (E=b;S=m;C=m) para o Grupo 2

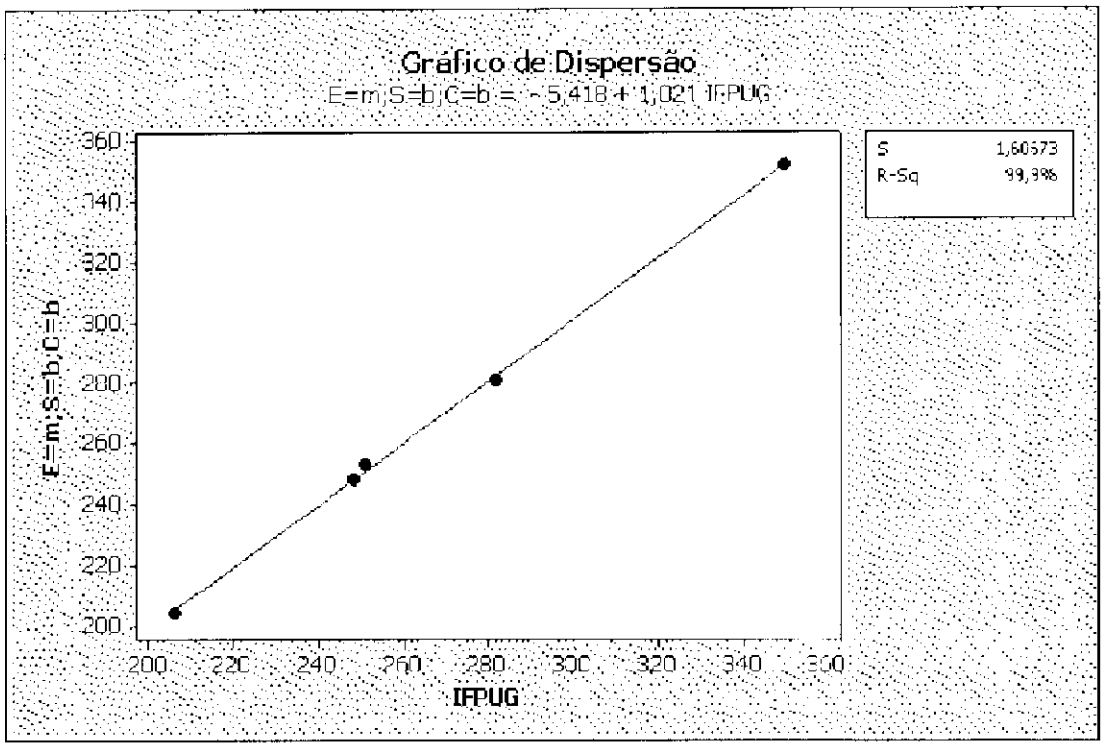


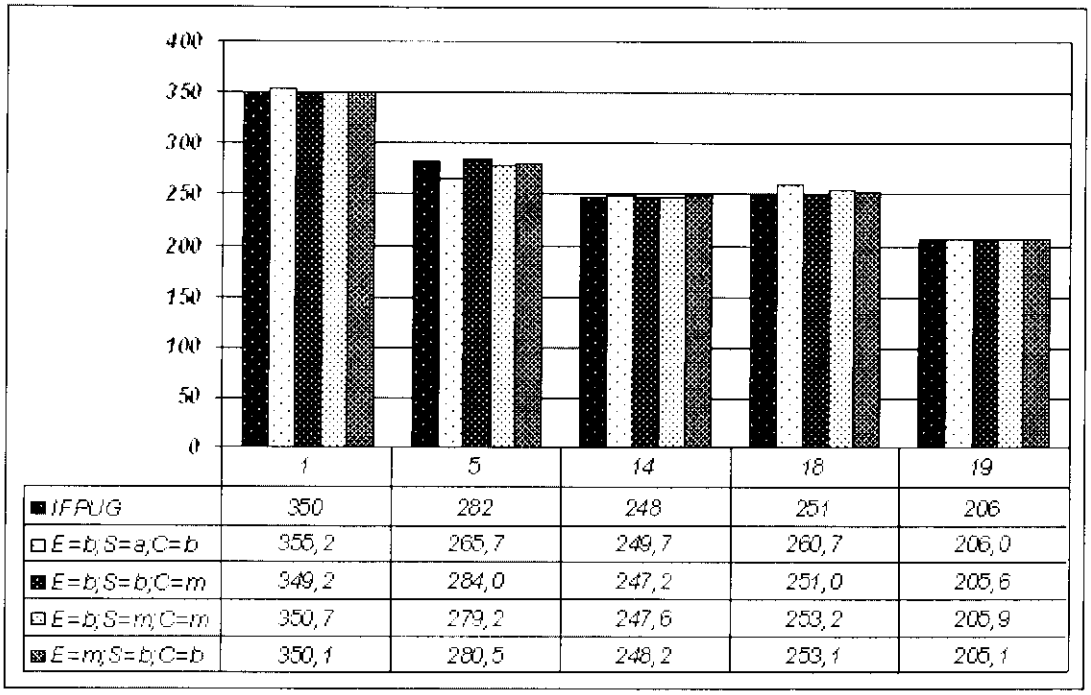
Figura 4.22: Gráfico de Dispersão (E=m;S=b;C=b) para o Grupo 2

Na tabela 4.12 é apresentado o ajuste que deve ser utilizado para cada uma das combinações.

Combinação	Ajuste
3) E=b;S=a;C=b	PF = (valor - 3,02)/1,005
5) E=b;S=b;C=m	PF = (valor + 14,76)/1,059
6) E=b;S=m;C=m	PF = (valor + 10,71)/1,077
15) E=m;S=b;C=b	PF = (valor + 5,418)/1,021

Tabela 4.12: Ajuste para as Combinações do Grupo 2

Os pontos de função determinados por essas combinações são exibidos na figura 4.23. As combinações apresentaram, em média, erros de 2,4% (E=b;S=a;C=b), 0,3% (E=b;S=b;C=m), 0,5% (E=b;S=m;C=m) e 0,4% (E=m;S=b;C=b).



	1	5	14	18	19
IFPUG	350	282	248	251	206
E=b;S=a;C=b	355,2	265,7	249,7	260,7	206,0
E=b;S=b;C=m	349,2	284,0	247,2	251,0	205,6
E=b;S=m;C=m	350,7	279,2	247,6	253,2	205,9
E=m;S=b;C=b	350,1	280,5	248,2	253,1	205,1

Figura 4.23: Pontos de Função - Melhores Combinações para o Grupo 2

As análises dos aplicativos dos dois grupos resultaram na possibilidade de utilização de três combinações para a contagem dos pontos de função dos aplicativos do grupo 1 e quatro combinações para os do grupo 2. A divisão dos aplicativos em dois grupos permitiu a obtenção de um melhor ajuste para cada um deles. O resultado das análises é exibido na tabela 4.13.

Grupo 1		Grupo 2	
Combinação	Ajuste	Combinação	Ajuste
1) E=b;S=b;C=b	PF = (valor − 0,449)/0,9812	3) E=b;S=a;C=b	PF = (valor - 3,02)/1,005
2) E=b;S=m;C=b	PF = (valor - 2,796)/0,9805	5) E=b;S=b;C=m	PF = (valor + 14,76)/1,059
3) E=b;S=a;C=b	PF = (valor - 9,284)/0,9793	6) E=b;S=m;C=m	PF = (valor + 10,71)/1,077
		15) E=m;S=b;C=b	PF = (valor + 5,418)/1,021

Tabela 4.13: Quadro Geral das Combinações e Ajustes

Estatisticamente, não há como afirmar que qualquer uma das três combinações escolhidas como melhor solução para as aplicações do grupo 1 é melhor que as outras duas. Para as quatro combinações do grupo 2 o raciocínio é o mesmo. Apesar disso, a utilização do método simplificado requer a escolha de uma única combinação para determinar os pontos de função de futuras aplicações que sejam incluídas no grupo 1 e uma outra combinação para a contagem das aplicações inseridas no grupo 2.

Dessa forma, foi escolhida de forma arbitrária a combinação 2) (E=b;S=m;C=b) para determinar os pontos de função dos aplicativos do grupo 1 e a combinação 5) (E=b;S=b;C=m) para determinar os pontos de função dos aplicativos do grupo 2.

Portanto, uma vez identificados os ALIs, AIEs, EEs, SES e CEs de um aplicativo pertencente a um dos dois grupos, utiliza-se uma das combinações apresentadas na tabela 6.1 para contar os pontos de função de forma simplificada.

Tipo de Função	Arquivo Lógico Interno	Arquivo de Interface Externa	Entrada Externa	Saída Externa	Consulta Externa
Complexidade Grupo 1	baixa	baixa	baixa	média	baixa
Ajuste Grupo 1	PF = (valor - 2,796)/0,9805				
Complexidade Grupo 2	baixa	baixa	baixa	baixa	média
Ajuste Grupo 2	PF = (valor - 14,76)/1,059				

Tabela 4.14: Método Simplificado

Na figura 4.24 é apresentada uma comparação entre o resultado da contagem utilizando essas combinações e o método detalhado do IFPUG.

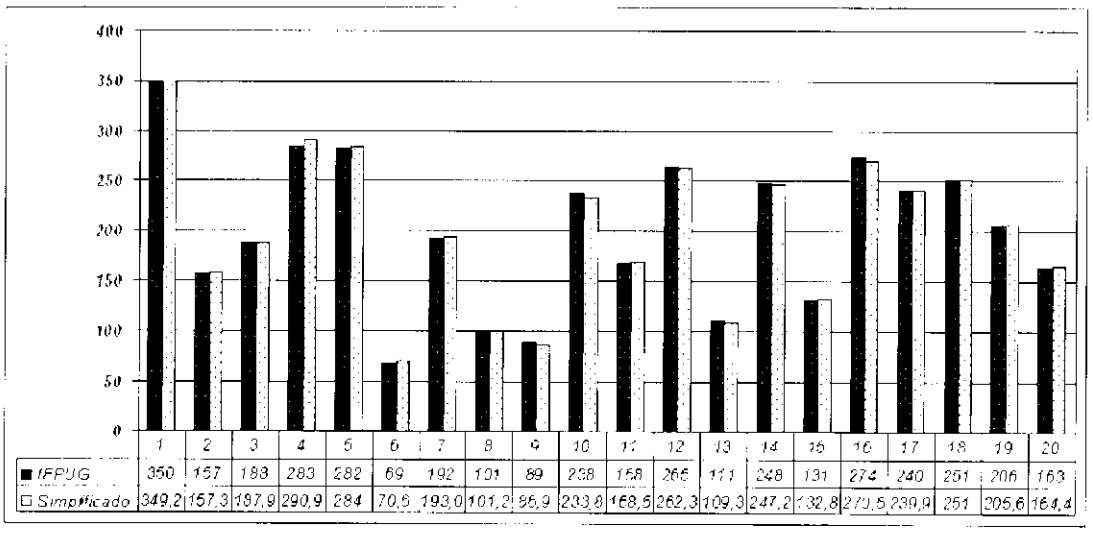


Figura 4.24: Método do IFPUG x Método Simplificado

4.5 Considerações Finais

Neste capítulo foi apresentado o estudo de caso no qual o método simplificado foi obtido. No próximo capítulo os requisitos de uma ferramenta para apoiar o processo de contagem são descritos.

CAPÍTULO

5

Apoio Automatizado

Neste capítulo são apresentados os requisitos de uma ferramenta para determinar a produtividade dos funcionários no desenvolvimento dos aplicativos e a contagem dos pontos de função.

5.1 Descrição da Ferramenta

A ferramenta tem como finalidades determinar a produtividade dos funcionários na construção dos aplicativos e contar os pontos de função utilizando o método simplificado. A partir da produção de um histórico de produtividade, a contagem dos pontos de função, aliada a esse histórico, será capaz de permitir uma estimativa da previsão do tempo de desenvolvimento necessário para cada um dos novos aplicativos.

A ferramenta é constituída por dois módulos. O módulo 1 implementa o gerenciamento das atividades e projetos, obtendo e armazenando informações sobre cada um deles. Esse módulo deve também garantir o controle das horas utilizadas pelos funcionários no desenvolvimento de cada aplicativo, funcionando como um sistema de contabilização de tempo (*time tracking*).

O módulo 1 possuirá as seguintes funcionalidades:

- gerenciamento do acesso dos usuários

- criação de projetos e atividades

- gerenciamento de atividades

- armazenamento de horas utilizadas para o desenvolvimento de cada atividade do projeto

- apresentação da produtividade dos funcionários

- envio de dados para o segundo módulo

Já o módulo 2 é constituído por um software instalado em dispositivos móveis. Esse módulo é responsável por determinar a contagem dos pontos de função através da utilização do método simplificado, apresentado neste trabalho, e estimar o tempo necessário para o desenvolvimento de um aplicativo baseando-se no histórico de produtividade determinado pelo módulo 1. A idéia é permitir a possibilidade de se elaborar uma estimativa do tempo necessário à construção do aplicativo durante as reuniões com o cliente.

Sempre que um projeto é encerrado, o módulo 1 atualiza o histórico de produtividade da equipe e envia essas informações para o módulo 2. Com essas informações, identificados os requisitos de um novo aplicativo, será possível apontar com maior precisão o tempo estimado para o desenvolvimento de tal aplicativo. Uma visão geral da estrutura da ferramenta é apresentada na figura 5.1.

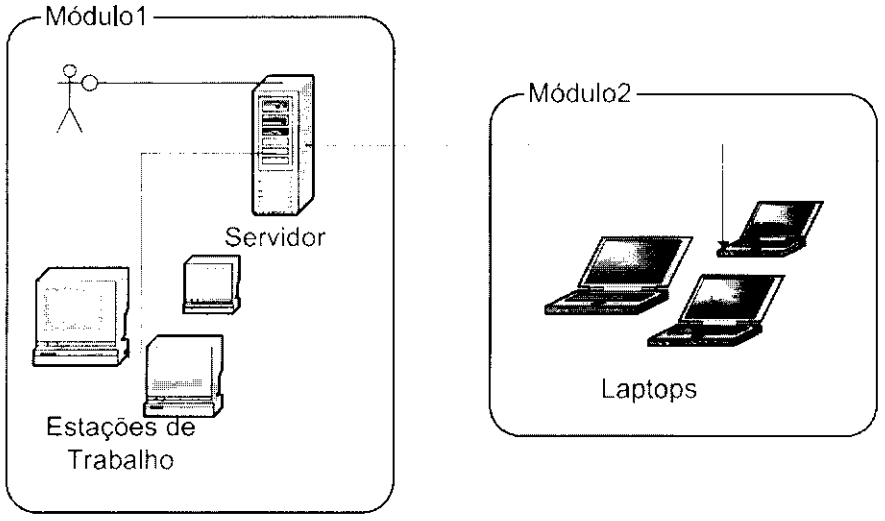


Figura 5.1: Modelo da Ferramenta

Nas próximas seções, são apresentados os casos de uso, os diagramas de seqüência e o diagrama de classes elaborados para descrever a ferramenta.

5.2 Casos de Uso

Os casos de uso descrevem a seqüência de eventos de um ator que usa um sistema para completar um processo. O cabeçalho é o de utilização típica para um caso de uso expandido, sendo descrito através da seguinte estrutura: apresenta-se o caso de uso, o ator, a finalidade do caso de uso e a visão geral do caso de uso. A seqüência típica de eventos possui o rótulo (a) para a ação do ator e o rótulo (r) para a resposta do sistema. Os seguintes casos de uso foram identificados:

1- Iniciar uso (log in)

Caso de Uso: Iniciar Uso

Atores: Programador, Gerente de Projeto, Vendedor

Finalidade: Permitir o acesso ao sistema.

Visão Geral: O ator digita o nome de usuário e a senha, e o sistema verifica se ele possui permissão para acessar o sistema. Ao término da verificação, caso a resposta seja positiva, o acesso ao sistema é liberado.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa quando o ator informa ao sistema o nome de usuário e a senha de acesso.

2-(r) Determina se o nome de usuário e a senha são válidos.

3-(r) Registra o horário de entrada do ator.

4-(a) Inicia a utilização do sistema.

Seqüências Alternativas:

Linha 2: Nome de usuário ou senha inválidos. Reiniciar o processo de identificação.

2- Encerrar uso

Caso de Uso: Encerrar Uso

Atores: Programador, Gerente de Projeto, Vendedor

Finalidade: Encerrar o acesso ao sistema.

Visão Geral: O ator solicita o encerramento de seu acesso ao sistema, que por sua vez, armazena o horário de término do acesso.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa com a solicitação de encerramento do acesso.

2-(r) Armazena o horário de término de acesso.

3-(r) Armazena o horário de paralisação de uma determinada atividade, caso o ator seja o

programador.

4-(r) Encerra o acesso do ator ao sistema.

Seqüências Alternativas:

Linha 3: Caso nenhuma atividade esteja em andamento, nenhuma informação é armazenada.

3- Iniciar Atividade (do Projeto)

Caso de Uso: Iniciar Atividade

Ator: Programador

Finalidade: Acompanhar o início ou a continuidade de uma atividade.

Visão Geral: O programador seleciona uma atividade já em andamento ou uma nova atividade. O sistema exige uma breve descrição da atividade caso ela tenha sido escolhida pela primeira vez. O sistema armazena o horário de início da execução da atividade.

Seqüência Típica de Eventos:

1-(r) Este caso de uso começa com a apresentação das atividades (agendadas ou em andamento) e dos projetos para o programador.

2-(a) Seleciona projeto e atividade.

3-(r) Armazena o horário de (re)início da execução da atividade.

4-(r) Exige uma breve descrição da atividade que será efetuada.

5-(a) Fornece os dados da descrição da atividade.

6-(r) Armazena os dados de descrição da atividade.

7-(r) Atualiza o *status* da atividade (agendada, andamento, paralisada, finalizada) para andamento.

8-(a) (Re)Inicia a execução da atividade.

Seqüências Alternativas:

Linha 4: A atividade foi iniciada anteriormente. A descrição não é necessária.

Linha 5: Nenhum dado é fornecido caso a atividade já tenha sido iniciada anteriormente.

Linha 6: Caso a atividade já tenha sido iniciada anteriormente, os dados da descrição já estão armazenados, e portanto, não serão armazenados novamente.

4- Paralisar Atividade (do Projeto)

Caso de Uso: Paralisar Atividade

Ator: Programador

Finalidade: Permitir a paralisação de uma atividade.

Visão Geral: O programador determina a paralisação de uma atividade. O sistema exige

uma justificativa e armazena o horário de paralisação da atividade.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa com a solicitação de paralisação de uma atividade.

2-(r) Exige uma justificativa.

3-(a) Fornece a justificativa.

4-(r) Atualiza o *status* da atividade (agendada, andamento, paralisada, finalizada) para paralisada.

5-(r) Armazena o horário de paralisação.

5- Encerrar Atividade (do projeto)

Caso de Uso: Encerrar Atividade

Ator: Programador

Finalidade: Gerenciar o encerramento de cada atividade.

Visão Geral: O programador determina o encerramento de uma atividade. O sistema armazena o horário de encerramento e envia as informações para o módulo 2.

Seqüência de Eventos:

1-(a) Este caso de uso começa com a solicitação de encerramento da atividade.

2-(r) Armazena o horário de encerramento da atividade.

3-(r) Atualiza o *status* da atividade (agendada, andamento, paralisada, finalizada) para finalizada.

6- Criar Atividade

Caso de Uso: Criar atividade

Ator: Gerente de Projeto

Finalidade: Criar as atividades do projeto.

Visão Geral: As atividades (ex: manutenção, teste) que serão utilizadas pelos programadores para o desenvolvimento dos projetos são criadas.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa com a solicitação da criação de uma nova atividade.

2-(r) Exige o nome da atividade e o objetivo da atividade.

3-(a) Insere as informações necessárias à criação da atividade.

4-(r) Armazena a atividade.

Seqüências Alternativas:

Linha 3: Caso o nome informado já exista, será solicitado que um nome diferente seja inserido.

7- Criar Projeto

Caso de Uso: Criar Projeto

Ator: Gerente de Projeto

Finalidade: O gerente cria um projeto e aloca as atividades para os programadores.

Visão Geral: O gerente de projeto cria um novo projeto, determina as atividades que serão realizadas durante seu desenvolvimento e as aloca para os programadores.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa com a solicitação de criação de um novo projeto.

2-(r) Exige a descrição do projeto e o conjunto de atividades que serão desenvolvidas.

3-(a) Fornece os dados exigidos pelo sistema.

4-(r) Armazena os dados fornecidos.

5-(a) Define o conjunto de atividades que cada programador irá desenvolver.

6-(r) Distribui as atividades para os respectivos programadores.

8- Atualizar Projeto

Caso de Uso: Atualizar Projeto

Ator: Gerente de Projeto

Finalidade: Alterar as atividades do projeto ou as atividades alocadas para algum dos programadores.

Visão Geral: O gerente de projeto tem a possibilidade de alterar as atividades de um determinado projeto, bem como redistribuir atividades.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa com o pedido de listagem das atividades de um projeto.

2-(r) Fornece as atividades relacionadas com o projeto e os respectivos programadores responsáveis pela execução de cada uma.

3-(a) Determina se alguma atividade ainda não iniciada será excluída ou se novas atividades serão incluídas.

4-(a) Determina se alguma atividade ainda não iniciada será redistribuída.

5-(r) Atualiza as informações sobre o projeto.

9- Acompanhar Projeto

Caso de Uso: Acompanhar Projeto

Ator: Gerente de Projeto

Finalidade: Acompanhar o andamento do(s) projeto(s).

Visão Geral: O gerente acompanha o andamento de cada projeto, a produtividade de cada funcionário, e também o andamento de cada atividade.

Seqüência Típica de Eventos:

1-(a) Este caso de uso começa com a solicitação de informações sobre o andamento dos projetos.

2-(r) A situação de cada atividade é apresentada.

3-(r) A produtividade individual e da equipe são apresentadas.

4-(r) A expectativa de horas para finalizar o projeto é apresentada.

10- Finalizar Projeto

Caso de Uso: Finalizar Projeto

Ator: Gerente de Projeto

Finalidade: Finalizar o projeto

Visão Geral: O gerente finaliza o projeto e as informações são enviadas para o módulo 2.

1-(a) Este caso de uso começa com a solicitação do encerramento do projeto.

2-(r) Envia as informações para o módulo 2.

11- Incluir Usuário

Caso de Uso: Incluir Usuário

Ator: Administrador do Sistema

Finalidade: Gerenciar a inclusão de usuários no sistema.

Visão Geral: O administrador do sistema inclui as informações sobre um novo usuário. Uma senha provisória é gerada pelo sistema e deve ser alterada no primeiro acesso do usuário.

Seqüência Típica de Eventos:

1-(a) Solicita a inclusão de um novo usuário.

2-(r) Exige o nome de usuário.

3-(a) Informa o nome de usuário.

4-(r) Exige o tipo de privilégio.

5-(a) Informa o tipo de privilégio.

6-(r) Gera uma senha para o nome de usuário.

Seqüências Alternativas:

Linha 3: A informação de um nome de usuário já existente implica no reinício do processo.

12- Excluir Usuário

Caso de Uso: Excluir Usuário

Ator: Administrador do Sistema

Finalidade: Gerenciar a exclusão de usuários do sistema.

Visão Geral: O administrador do sistema exclui um usuário do sistema.

Seqüência Típica de Eventos:

1-(a) Solicita a exclusão de um usuário.

2-(r) Verifica se o usuário está acessando o sistema.

3-(r) Apaga as informações sobre o usuário.

Seqüências Alternativas:

Linha 2: Caso o usuário esteja acessando o sistema no momento da tentativa de exclusão, o processo de exclusão é cancelado.

13- Alterar Usuário

Caso de Uso: Alterar Usuário

Ator: Administrador do Sistema

Finalidade: Gerenciar a alteração das informações dos usuários do sistema.

Visão Geral: O administrador do sistema altera informações referentes a um usuário.

Seqüência Típica de Eventos:

1-(a) Solicita a alteração das informações sobre um usuário.

2-(r) Fornece as informações atuais sobre o usuário.

3-(a) Altera as informações.

4-(r) Armazena as alterações.

14- Contar Pontos de Função

Caso de Uso: Contar os Pontos de Função

Ator: Vendedor

Finalidade: Identificar os requisitos e determinar os pontos de função.

Visão Geral: Os requisitos são identificados junto ao usuário e os pontos de função são determinados utilizando-se o método simplificado.

Seqüência Típica de Eventos:

1-(a) Insere as funcionalidades.

2-(r) Disponibiliza os tipos de função nos quais as funcionalidades podem ser classificadas.

3-(a) Determina a classificação das funcionalidades.

4-(r) Determina a complexidade de cada funcionalidade de acordo com o método simplificado.

5-(r) Armazena os dados.

6-(r) Estabelece o total de pontos de função.

15- Estimar Tempo de Desenvolvimento

Caso de Uso: Determinar a estimativa do tempo de desenvolvimento

Ator: Vendedor

Finalidade: Analisar o número de pontos de função do software para determinar uma estimativa do seu tempo de desenvolvimento.

Visão Geral: O histórico da produtividade é utilizado para estimar o tempo necessário para desenvolver um novo software.

Seqüência Típica de Eventos:

1-(a) Fornece o número de pontos de função de uma aplicação.

2-(r) Determina a estimativa do número de horas necessárias para a conclusão do projeto.

5.3 Diagramas de Seqüência

Um diagrama de seqüência é uma figura que mostra os eventos que os atores geram, sua ordem e os eventos entre os sistemas. Os seguintes diagramas de seqüência foram identificados:

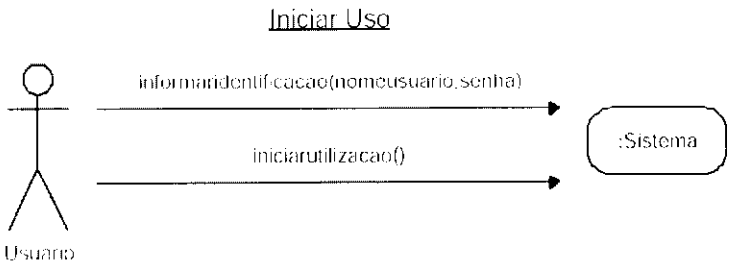


Figura 5.2: Diagrama de Seqüência: Iniciar Uso

Figura 5.3: Diagrama de Seqüência: Encerrar Uso

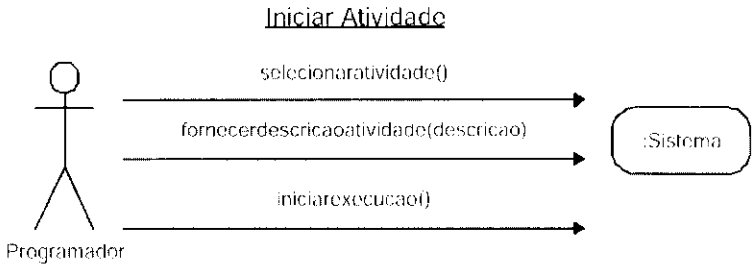


Figura 5.4: Diagrama de Seqüência: Iniciar Atividade

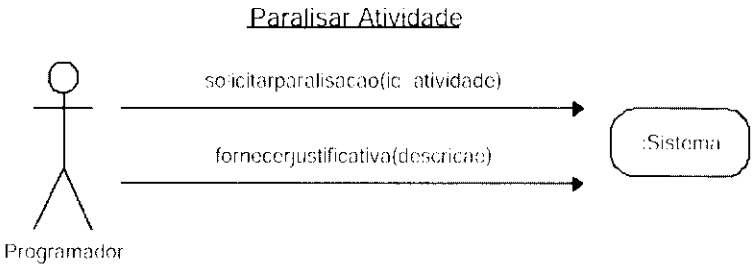


Figura 5.5: Diagrama de Seqüência: Paralisar Atividade

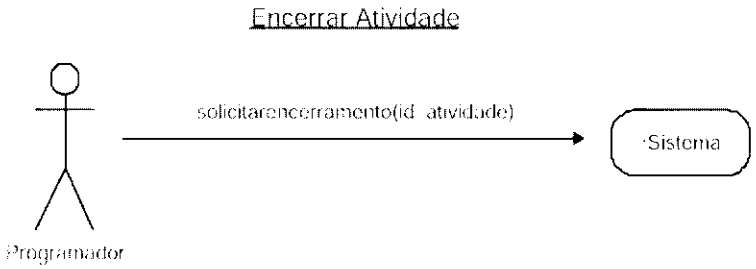


Figura 5.6: Diagrama de Seqüência: Encerrar Atividade

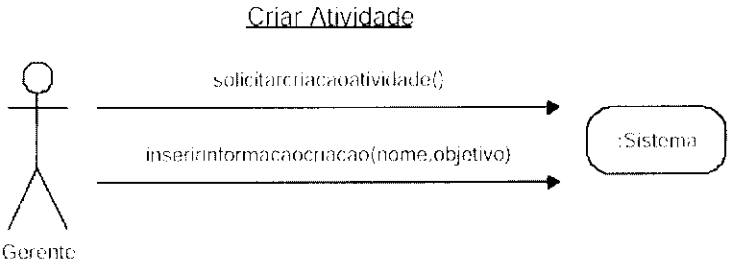


Figura 5.7: Diagrama de Seqüência: Criar Atividade

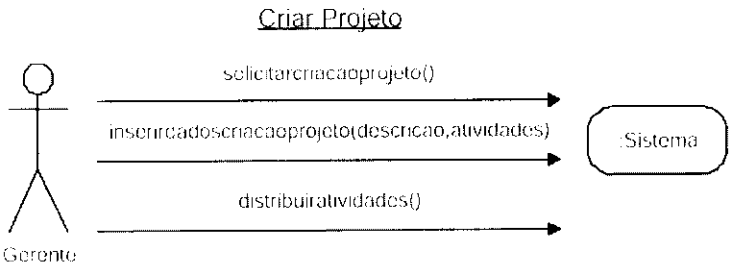


Figura 5.8: Diagrama de Seqüência: Criar Projeto

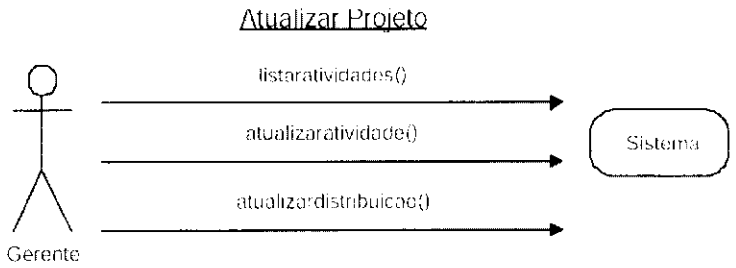


Figura 5.9: Diagrama de Seqüência: Atualizar Projeto

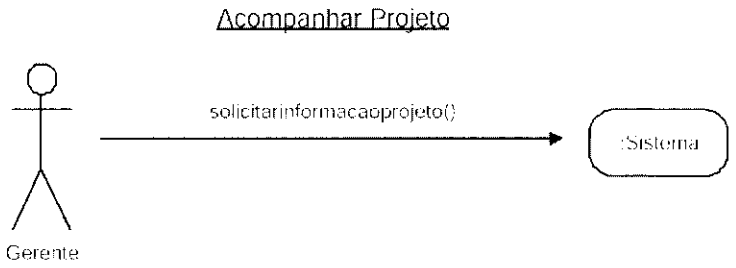


Figura 5.10: Diagrama de Seqüência: Acompanhar Projeto

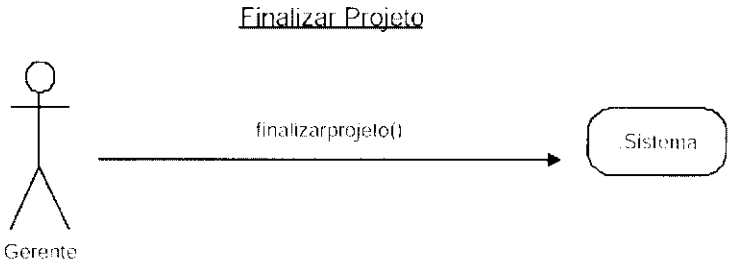


Figura 5.11: Diagrama de Seqüência: Finalizar Projeto

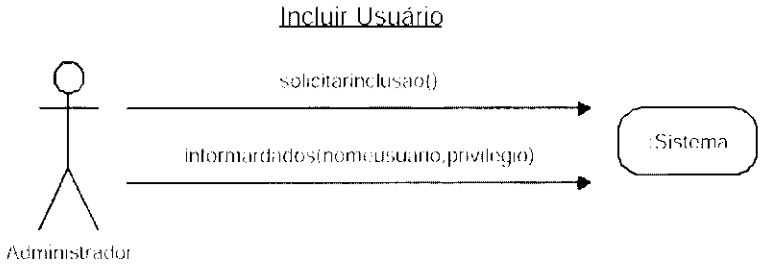


Figura 5.12: Diagrama de Seqüência: Incluir Usuário

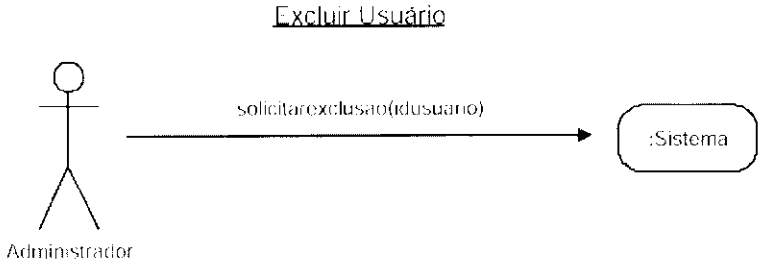


Figura 5.13: Diagrama de Seqüência: Excluir Usuário

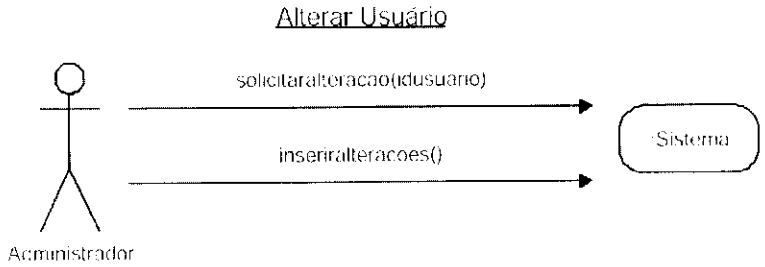


Figura 5.14: Diagrama de Seqüência: Alterar Usuário

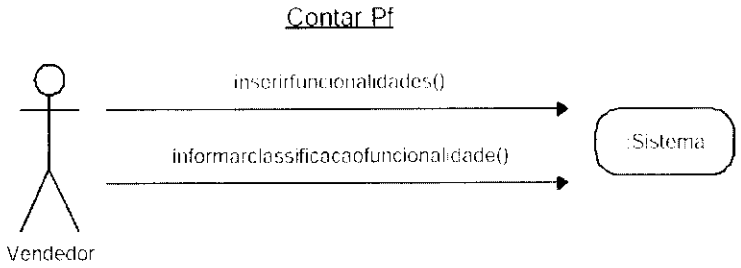


Figura 5.15: Diagrama de Seqüência: Contar Pf

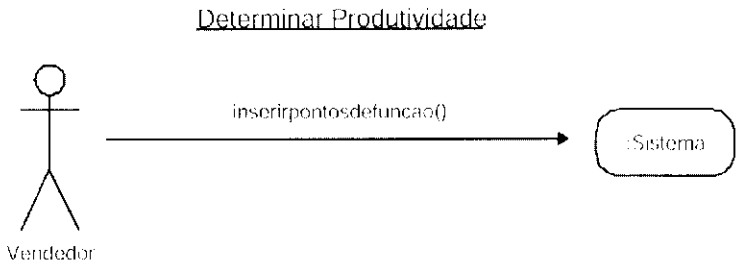


Figura 5.16: Diagrama de Seqüência: Determinar Produtividade

5.4 Diagramas de Classes

O diagrama de classes ilustra as especificações para as classes de software de uma aplicação. A seguir, é apresentado o diagrama de classes:

No diagrama, há uma generalização representada pelas classes Usuário, Programador, Vendedor, e Gerente, na qual a classe Usuário é o elemento geral. Cada usuário pode realizar várias atividades, as quais são registradas pela classe Registro de Atividade. Uma outra função dessa classe é flexibilizar a realização de atividades, permitindo que um programador realize uma atividade não alocada a ele pelo gerente. As atividades que o gerente aloca a um determinado usuário, e que necessariamente devem ser efetuadas por ele, são gerenciadas pela classe Ordem.

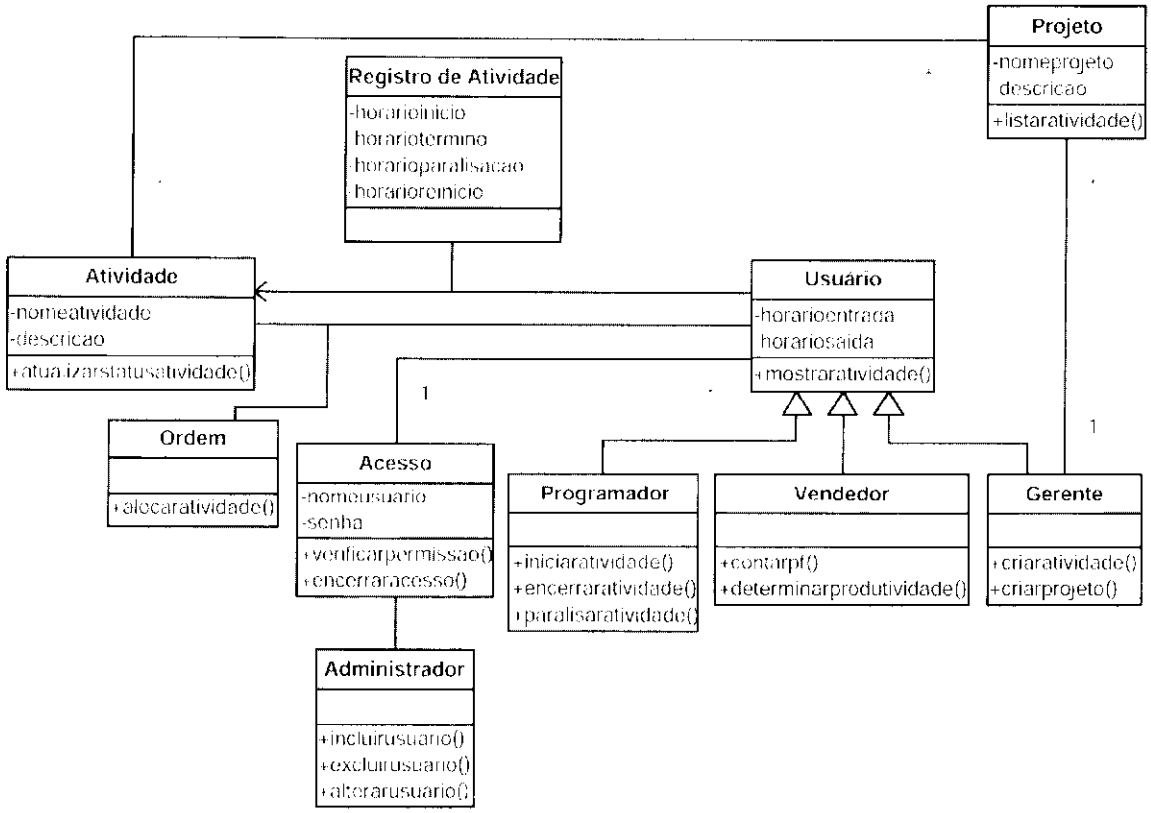


Figura 5.17: Diagrama de Classes

5.5 Considerações Finais

Neste capítulo foram apresentados os requisitos de uma ferramenta para apoiar o processo de contagem dos pontos de função e a estimativa do tempo de desenvolvimento de um aplicativo. A seguir, as conclusões e trabalhos futuros são apresentados.

CAPÍTULO

6

Conclusões e Trabalhos Futuros

6.1 Conclusões

A estimativa de tamanho de software é fundamental para a obtenção dos custos de um projeto de software e, conseqüentemente, uma atividade importante para um bom gerenciamento do projeto de software.

Neste trabalho, foi desenvolvido um método simplificado para determinar os pontos de função de aplicativos web. O método simplificado foi desenvolvido especificamente para as características de desenvolvimento da empresa Linkway [49].

Inicialmente, a pesquisa tinha como foco a análise da possibilidade de utilizar os métodos simplificados de contagem dos pontos de função elaborados pela NESMA (contagens indicativa e estimada) para a contagem dos pontos de função de aplicativos desenvolvidos para web. Para isso, uma comparação entre a contagem utilizando os métodos da NESMA e a contagem utilizando o método detalhado do IFPUG foi realizada.

Essa comparação foi elaborada através de um estudo de caso reunindo dados de vinte aplicativos web, cujas características são comuns à maioria dos softwares desenvolvidos pela empresa.

Os resultados demonstraram que os métodos simplificados elaborados pela NESMA não podem ser aplicados para estimar o tamanho dos aplicativos web desenvolvidos na empresa. Isso é observado pelo fato das contagens preconizadas pela NESMA apresentarem resultados

muito diferentes dos apresentados pelo método detalhado do IFPUG.

Durante a realização do estudo comparativo, a análise das planilhas contendo a contagem dos pontos de função dos aplicativos mostrou que havia um predomínio de funções com complexidade funcional baixa. Isso motivou a tentativa de elaborar um método simplificado baseado em valores fixos para as complexidades dos ALIs, AIEs, EEs, SEs e CEs.

O método simplificado foi construído através de dois passos. No primeiro passo analisou-se como deveriam ser fixadas as complexidades das funções de dados (ALIs e AIEs). O segundo passo foi analisar como deveriam ser fixadas as complexidades funcionais das funções transacionais (EEs, SEs, CEs). A complexidade funcional das funções de dados foi fixada como baixa por elas apresentarem, em sua grande maioria, complexidade funcional baixa. Para as funções transacionais, havia um predomínio da complexidade funcional baixa, mas havia também várias funções com complexidade funcional média. A solução foi determinar o número de pontos de função para cada uma das possíveis combinações das complexidades funcionais e analisar qual delas apresentava o melhor resultado.

Análises estatísticas foram realizadas no intuito de determinar qual das combinações apresentava os resultados mais próximos daqueles encontrados pelo método detalhado do IFPUG. Os resultados das análises apontaram quatro possíveis maneiras de construção do método simplificado.

Os aplicativos foram analisados novamente na tentativa de identificar alguma característica que colaborasse para refinar a análise anterior. Assim, comparando os requisitos dos aplicativos, foi possível dividi-los em dois grupos, de acordo com as funcionalidades que normalmente são implementadas quando cada aplicativo é desenvolvido.

As análises dos aplicativos dos dois grupos resultaram na possibilidade de utilização de três maneiras para a contagem dos pontos de função dos aplicativos do grupo 1 e quatro maneiras para os do grupo 2. Estatisticamente, não há como afirmar que qualquer uma das maneiras seja melhor que as outras, para o grupo em questão.

Dessa forma, foi escolhida de forma arbitrária a forma apresentada na tabela 6.1 para contar os pontos de função de forma simplificada.

Na prática, a grande vantagem da utilização do método simplificado é o fato de identificados os ALIs, AIEs, EEs, SEs, e CEs, a complexidade já está definida automaticamente, facilitando o processo de estimativa do tamanho do software.

Entretanto, deve-se enfatizar que os resultados apresentados valem para a tecnologia, o

Tipo de Função	Arquivo Lógico Interno	Arquivo de Interface Externa	Entrada Externa	Saída Externa	Consulta Externa
Complexidade Grupo 1	baixa	baixa	baixa	média	baixa
Ajuste Grupo 1	PF = (valor - 2,796)/0,9805				
Complexidade Grupo 2	baixa	baixa	baixa	baixa	média
Ajuste Grupo 2	PF = (valor − 14,76)/1,059				

Tabela 6.1: Método Simplificado

domínio de problema e as linguagens utilizadas pela empresa estudada. Generalizar o método para qualquer outra empresa que desenvolve aplicativos web não é a intenção deste trabalho.

6.2 Trabalhos Futuros

A partir dos resultados deste trabalho, as seguintes linhas de pesquisa podem ser sugeridas:

- Apesar da não generalização do método simplificado para qualquer empresa de desenvolvimento de aplicativos web, há a intenção de verificar a validade do método em outras organizações.
- Incluir um número maior de aplicativos para analisar se o comportamento das combinações definidas como as melhores se altera.
- Analisar a necessidade da elaboração de Características Gerais dos Sistemas específicas para o ambiente de desenvolvimento da empresa Linkway.
- Implementar a ferramenta cujos requisitos foram apresentados neste trabalho.

Referências Bibliográficas

[1] Whitmire S. A. *Applying Function Points to Object Oriented Software.* Software Engineering Productivity Handbook, McGraw-Hill, 1993.

[2] Kitchenham B. The problem with function points. *IEEE Software*, Março/Abril 1997; páginas: 30 -31.

[3] BFPUG. Brazilian function point users group. *Disponivel em http://www.bfpug.com.br*, Último acesso em Abril/2004.

[4] Gláucio Brogini. *CQS-AE - Uma abordagem evolucionista para garantia de qualidade de software.* Dissertação de Mestrado, USP - São Carlos, 2003.

[5] Hsu J. C. Constrained two-sided simultaneous confidence intervals for multiple comparisons with the best. *Annals of Statistics*, 1984; vol. 12; páginas: 1136 - 1144.

[6] Hsu J. C. *Multiple Comparisons, Theory and methods.* Chapman e Hall, 1996.

[7] Jones C. *Programming Productivity.* McGraw-Hill, 1986.

[8] Jones C. *Estimating Software Costs.* McGraw-Hill, 1998.

[9] Jones C. Backfiring: Converting lines of code to function points. *IEEE Computer, 29 (11)*, Novembro 1999; páginas: 86 - 87.

[10] Kemerer C. Reliability of function points measurement: a field experiment. *Comm. ACM*, 1993; vol. 36; nro 2; páginas: 85 - 97.

[11] Symons C. Conversion between ifpug 4.0 and mkii function points. *Software Measurement Services*, 1999.

[12] Cosmic-FFP. *Measurement Manual*. Cosmic, 2.1 edition, Maio 2001.

[13] MCT Ministério da Ciência e Tecnologia. Qualidade e produtividade no setor de software. *Disponível em http://www.mct.gov.br/Temas/info/Dsi/Quali2001/ Public2001.htm*, Tabela 40 - Práticas de Engenharia de Software Adotadas no Desenvolvimento e Manutenção de Software, 2001.

[14] MCT Ministério da Ciência e Tecnologia. Qualidade e produtividade no setor de software. *Disponível em http://www.mct.gov.br/Temas/info/Dsi/Quali2001/ Public2001.htm*, Tabela 01 - Atividades das Organizações no Tratamento de Software, 2001.

[15] MCT Ministério da Ciência e Tecnologia. Qualidade e produtividade no setor de software. *Disponível em http://www.mct.gov.br/Temas/info/Dsi/Quali2001/Public2001.htm*, Tabela 06 - Porte das Organizações, Segundo a Força de Trabalho Total e Efetiva, 2001.

[16] Luiz Fernando de Oliveira Silva. *Estudo de alternativas para dar suporte à melhoria de qualidade de processo de software em empresas de pequeno porte com base no SW CMM nível 2*. Dissertação de Mestrado, Ufscar, 2003.

[17] Fetcke T. e Abran A. e Nguyen T. Mapping the oo-jacobson approach into function point analysis. *Proceedings Technology of Object-Oriented Languages and Systems. TOOLS 23*, Julho/Agosto 1997; páginas: 192 - 202.

[18] Caldiera G. e Antoniol G. e Fiutem R. e Lokan C. Definition and experimental evaluation of function points for object-oriented systems. *Proceedings of the Fifth International Software Metrics Symposium*, Novembro 1998; páginas: 167 - 178.

[19] Jeffery J. R. e Barnes M. A. Comparison of function point counting techniques. *IEEE Trans. Software Eng.*, 1993; vol. 19; nro 5; páginas: 529 - 532.

[20] Tavares H. e Carvalho A. e Castro J. Medição de pontos de função a partor da especificação de requisitos. *Anais do WER02 - Workshop em Engenharia de Requisitos, Valencia, Espanha*, Novembro 2002; páginas: 278 - 298.

[21] Jacobson I. e Christerson M. Object-oriented software engineering. a use case driven approach. *Addison-Wesley*, 1992.

[22] Diab H. e Frappier M. e Denis R. Formalizing cosmic-ffp using room. *ACS/IEEE International Conference on Computer Systems and Applications*, Junho 2001; páginas: 312 - 318.

[23] Lai R. e Huang S. A model for estimating the size of a formal communication protocol specification and its implementation. *IEEE Transactions on Software Engineering*, Janeiro 2003; vol. 29; nro 1; páginas: 46 - 62.

[24] Kusumoto S. e Imagawa M. e Inoue K. e Morimoto S. e Matsusita K. e Tsuda M. Function point measurement from java programs. *ICSE 2002. Proceedings of the 24rd International Conference on*, Maio 2002; páginas: 576 - 582.

[25] Kusumoto S. e Inoue K. e Kasimoto T. e Suzuki A. e Yuura K. e Tsuda M. Function point measurement for object-oriented requirements specification. *Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International*, Outubro 2000; páginas: 543 - 548.

[26] Kitchenham B. e Känsälä K. Inter-item correlations among function points. *Proc. 15th International Conference on Software Engineering*, Maio 1993; páginas: 477 - 480.

[27] Agarval R. e Kumar M. e Mallick S. e Bharadwaj R. e Anantwar D. Estimating software projects. *Software Engineering Notes*, Julho 2001; vol. 26; nro 4; páginas: 60 - 67.

[28] Uemura T. e Kusumoto S. e Inoue K. Function point measurement tool for uml design specification. *Proceedings Sixth International Software Metrics Symposium*, Novembro 1999; páginas: 62 - 69.

[29] Desharnais J. M. e Morris P. *Validation process in software engineering: an example with function points*. Software Metrics: Research and Practice in Software Measurement, 1997; páginas: 183 - 191.

[30] Raman A. e Noore A. Software metrics for real-time systems using fuzzy sets. *Proceedings of the 35th Southeastern Symposium on System Theory*, Março 2003; páginas: 74 - 78.

[31] Saito M. e Onari M. e Yuura K. e Kameda T. Visualizing tool for required specifications. *The Hitachi Hyoron*, 1995; vol. 77; nro 12; páginas: 15 - 18.

[32] Fenton N. E. e Pfleeger S. L. *Software Metrics: A Rigorous and Pratical Approach*. PWS, 2 edition, 1997.

[33] Ram J. e Raju S. V. G. K. Object oriented design function points. *Proceedings of the First Asia-Pacific Conference on Quality Software*, Outubro 2000; páginas: 121 - 126.

[34] Hastings T. E. e Sajeev A. S. M. A vector-based approach to software size measurement and effort estimation. *IEEE Transactions on Software Engineering*, Abril 2001; vol. 27; nro 4; páginas: 337 - 350.

[35] Jeffery D. R. e Stathis J. Function point sizing: Structure, validity and applicability. *Journal of Empirical Software Engineering*, 1996; páginas: 11 - 30.

[36] Briand L. C. e Wieczorek I. Software resource estimation. *Encyclopedia of Software Engineering*, 2002; vol. P-Z; nro 2; páginas: 1160 - 1196.

[37] Pöstion E. e Zuba G. Function point analysis as an integrated part of industrial software engineering culture. *Proceedings of ESCOM SCOPE 99 Conference*, Maio 1999.

[38] Bootsma F. How to obtain accurate estimates in a real-time environment using full function points. *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, Março 2000; páginas: 105 - 112.

[39] Teologlou G. Measuring object oriented software with predictive object points. *10th Conference on European Software Control and Metrics*, Maio 1999.

[40] International Function Point Users Group. *Function Point Counting Practices Manual*. IFPUG, 4.1.1 edition, Abril 2000.

[41] International Software Benchmarking Standards Group. *Worldwide Software Development: The Benchmark*. IFPUG, 4 edition, Abril 1997.

[42] IFPUG. International function point users group. *Disponivel em http://www.ifpug.org*, último acesso em Abril de 2004.

[43] Albrecht A. J. Measuring application development productivity. *Proc. IBM Applications Development Symposium*, 1979; páginas - 83 - 92.

[44] Bielak J. Improving size estimates using historical data. *IEEE Software*, Novembro/Dezembro 2000: páginas 27 - 35.

[45] Dolado J. J. A validation of the component-based method for software size estimation. *IEEE Trans. Software Eng.*, Outubro 2000; vol. 26; nro 10; páginas: 1006 - 1021.

[46] Lokan C. J. An empirical analysis of function point adjustment factors. *encontrar a ref*, 2000.

[47] Lokan C. J. An empirical study of the correlations between function point elements. *Proc. of the 6th International Software Metrics Symposium*, Novembro 1999; páginas: 200 - 206.

[48] Rumbaugh J. Object-oriented modelling and design. *Prentice-Hall*, 1991.

[49] Linkway. Linkway. *Disponivel em http://www.linkway.com.br*, Último acesso em Abril/2004.

[50] Ruhe M. Using web objects for estimating software development effort for web applications. *Proceedings of the Ninth International Software Metrics symposium (METRICS'03)*, 2003.

[51] Schooneveldt M. Measuring the size of object oriented systems. *Proceedings of the 2nd Australian Conference on Software Metrics*, 1995.

[52] MiniTab. Minitab versão 14. *Disponivel em http://www.minitab.com/products/minitab/14/default.aspx*, Último acesso em Outubro/2004.

[53] NESMA. Netherlands software metrics association. *Disponivel em http://www.nesma.org/english/index.html*, Último acesso em Abril/2004.

[54] Pressman R. *Engenharia de Software*. Mc Graw Hill, 5 edition, 2001.

[55] Furey S. Why we should use function points. *IEEE Software*, Março/Abril 1997; páginas: 28 - 29.

[56] Humphrey W. S. *Managing the Software Process*. Addison-Wesley, 1989.

[57] Rollo T. Sizing e-commerce. *Proceedings of the ACOSM 2000 - Australian Conference on Software Measurement*, 2000.

[58] Débora Peliciano Diniz Tavares. *Melhoria de Processo de Pequena Empresa: Um Estudo de Caso*. Dissertação de Mestrado, Ufscar, 2002.