SOCIEDADE EDUCACIONAL DE SANTA CATARINA - SOCIESC

INSTITUTO SUPERIOR TUPY - IST

CARLOS SCHUSTER

MÉTRICAS DE SOFTWARE COM ÊNFASE EM ANÁLISE DE PONTO DE FUNÇÃO APRESENTANDO ESTUDO DE CASO NA EMPRESA DATASUL S.A.

Joinville

2006

CARLOS SCHUSTER

MÉTRICAS DE SOFTWARE COM ÊNFASE EM ANÁLISE DE PONTO DE FUNÇÃO APRESENTANDO ESTUDO DE CASO NA EMPRESA DATASUL S.A.

O presente trabalho de conclusão de Curso apresentado ao Instituto Superior Tupy como requisito parcial para a obtenção de grau de Bacharel em Sistemas de Informação, sob orientação do Prof. MSc. Luiz Camargo.

Joinville

2006

CARLOS SCHUSTER

MÉTRICAS DE SOFTWARE COM ÊNFASE EM APF

Este trabalho foi julgado e aprovado em sua forma final para a obtenção do grau de Bacharel em Sistemas de Informação do Instituto Superior Tupy.

Joinville, _____ de _____de 2006.

_____
Prof. MSc. Luiz Carlos Camargo – IST

_____
Prof. MSc. Jefferson Luiz Sayão – IST

_____
MSc. Ana Cristina Gozdziejewski – Datasul S.A.

Dedico este trabalho a meus pais, irmãos e amigos
que sempre estiveram ao meu lado
em todas as ocasiões
da minha vida.

AGRADECIMENTOS

A Deus por ter renovado sempre minhas forças durante todos os momentos desta minha caminhada.

Aos meus pais, João e Lucila, que sempre me incentivaram aos estudos e nunca deixaram que me abatesse diante das dificuldades encontradas no caminho.

Aos meus dois irmãos, Luciana e Gerson, que de forma indireta contribuíram apoiando-me nos momentos difíceis.

Ao Instituto Superior Tupy por ter concedido toda a sua estrutura para a realização deste trabalho.

A empresa Datasul por ter gentilmente me permitido abordar neste trabalho o processo de implantação da técnica de Análise de Ponto de Função.

Ao professor Sandro pela sua dedicação no auxílio com as regras de metodologia científica.

Ao meu orientador Luiz Camargo que através do seu conhecimento soube direcionar-me para o melhor caminho nos momentos de dificuldade e de dúvida sobre o que fazer e como fazer da melhor forma possível.

A minha ex-professora e também co-orientadora deste trabalho, Alessandra Anacleto, que mesmo estando distante fisicamente, se fez presente em momentos importantes que ajudaram em diversas definições deste trabalho.

De forma geral aos amigos que sempre estiveram ao meu lado e souberam das minhas conquistas e também das dificuldades que foram encontradas durante cada etapa desta empreitada.

Não se pode controlar o que não se pode medir.

Tom DeMARCO

RESUMO

Este trabalho tem como objetivo apresentar a métrica de Análise de Ponto de Função (APF) e sua forma de utilização e aplicação. Essa métrica também é bastante conhecida internacionalmente com o nome de *Function Point Analysis* (FPA). Atualmente a métrica de software APF está em constante processo de expansão. Sua utilização e estudo têm se difundido no mundo inteiro, e o Brasil é o um dos grandes responsáveis por este crescimento. A principal motivação para a realização deste trabalho é o de desmistificar a métrica de APF, expondo seus principais conceitos descritos até a versão 4.2 do Manual de Práticas de Contagem, que também é conhecido como CPM (*Counting Practices Manual*). Por fim, será apresentado um estudo de caso descrevendo os passos realizados pela empresa Datasul S.A. para a implantação da métrica de APF até o ano de 2006, apresentando os principais resultados obtidos com a mesma e também as perspectivas e passos futuros para a continua utilização da métrica como padrão para medir e estimar seus produtos e projetos.

Palavras-Chave: Métricas. Software. Qualidade. APF.

ABSTRACT

The main goal of this work is to present the Function Point Analysis, its application and way of usage which is also internationally known by FPA (Function Point Analysis). Nowadays, the study and the usage of the FPA software metric are in process of expansion all over the world. Brazil is one of the responsible countries for this growth due to the incentives on this issue. The main motivation for the accomplishment of this work is to demystify the APF metric, displaying its main described concepts until version 4.2 of the Counting Practical Manual, that is also known as CPM (Counting Practices Manual). Finally, an example will be presented describing the carried steps by the Datasul S.A. company to implement the metric of FPA until the end of 2006, showing the main returns achieved with that and also the perspectives and further steps for the continue usage of this metric as a default to measure and estimate their products and projects.

Keywords: Metric. Software. Quality. APF.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Face à competitividade globalizada, as empresas têm buscado por novas tecnologias, novos produtos e novos serviços, onde o objetivo principal é alcançar o sucesso em seus negócios.

Inevitavelmente, as organizações vêm criando uma dependência cada vez maior da tecnologia da informação, fazendo com que o seu papel seja fundamental dentro das empresas.

A conseqüência para este crescimento é o aumento do número de produtos de software, onde os custos de desenvolvimento e de manutenção de sistemas e a garantia da qualidade desses produtos são itens que geram divergências, principalmente para a área de gerenciamento de projetos de software.

Através da mensuração, é possível obter informações quanto à qualidade, adequação e o progresso evolutivo de processos, produtos e projetos de software. A métrica não pode garantir que um projeto tenha êxito, entretanto, auxilia principalmente os gerentes de projeto de software e desenvolvedores a utilizar uma abordagem pró-ativa em relação aos projetos que apresentam distorções nas informações obtidas.

A métrica de Análise de Pontos de Função (APF) é uma das ferramentas de mensuração existente no mercado em destaque. Por meio dela é possível dimensionar o tamanho de um software, podendo esta métrica ser aplicada em qualquer fase do projeto, levando em consideração apenas a funcionalidade geral que o produto proporciona aos usuários finais.

Segundo Vazquez (2004, p. 9):

> A análise de pontos de função é atualmente um instrumento utilizado por profissionais da área de sistemas e em empresas de todos os portes e segmentos da economia brasileira. Inicialmente o foco principal de sua aplicação era em estimativas. Atualmente ainda continua sendo uma ferramenta importante em estimativas, mas também tem sido aplicada cada vez com mais sucesso como unidade de medição de contratos de desenvolvimento de software e como ferramenta na gerência de projetos.

Com a exigência de produtos e serviços de informática de alta qualidade e menor custo, a utilização de métricas vem conquistando cada vez mais espaço no mercado de software e desta forma contribuindo com a melhoria do gerenciamento de projetos.

O objetivo geral deste trabalho é apresentar algumas métricas de software existentes no mercado, detalhando a métrica de Análise de Ponto de Função, que atualmente é utilizada pela empresa Datasul S.A., cenário de um estudo de caso apresentado.

Com base no objetivo geral, os objetivos específicos deste trabalho são:

a) Citar algumas métricas de software existentes no mercado;
b) Conceituar a métrica de software de Análise de Ponto de Função e seu histórico;
c) Apresentar estudo de caso real aplicado a Datasul com a métrica de Análise de Ponto de função;
d) Aperfeiçoar a gerência de projetos de software e o relacionamento com os clientes, padronizando a forma de negociação e assim gerando menos ruídos no processo de negociação;
e) Formar um *baseline*[1] para estimativas de futuras alterações no produto;
f) Gerenciar contratos de software, baseados em uma métrica de software.

Baseline[1] – Também conhecido como Linha Base, que representa uma base de dados com informações iniciais. Repositório de dados e informações.

Os assuntos que serão abordados nesse trabalho têm a seguinte estrutura de divisão:

No capitulo 2 é apresentada o estado da arte, onde são mostradas pesquisas de mercado e tendências no uso da métrica de software de APF.

No capítulo 3 é abordado o assunto de processo de desenvolvimento de software e alguns de seus paradigmas.

Já para o capitulo 4 o assunto apresentado é a relação existente entre processo x produto x projeto, apresentando a importância dessa relação.

Para o capítulo 5 o assunto métricas é descrito, apresentando uma breve introdução de métricas para produto existentes atualmente e utilizadas pelo mercado.

No capítulo 6 é descrito a métrica de APF, com um breve histórico e os passos para realização de uma medição.

Para o capítulo 7 o tema é estimativas, onde serão apresentados alguns métodos para a realização de estimativas de produto e projeto, baseadas em APF.

No capítulo 8 será apresentado um estudo de caso que descreve o processo de implantação da métrica de Análise de Ponto de Função na empresa Datasul S.A., bem como sua relevância para essa empresa.

Por fim serão apresentados os resultados obtidos e considerações finais sobre este trabalho desenvolvido.

2 ESTADO DA ARTE

Nas últimas décadas, cada vez mais presencia-se o envolvimento de vários setores da sociedade com a tecnologia da informação Para as empresas que estão em busca de um espaço cada vez maior no cenário mundial, a tecnologia tornou-se algo fundamental no mercado que a cada dia está mais competitivo, buscando as melhores soluções baseadas no custo x benefício para cada ramo de atividade.

Em contra partida, as organizações que trabalham com a tecnologia da informação tem fornecido soluções para empresas de diversos ramos de atividades, mas não despendendo a atenção necessária para pontos importantes, como os custos de desenvolvimento e de manutenção de sistemas, bem como, a garantia da qualidade dos produtos e projetos realizados.

Pressman (1995, p. 48) comenta:

> Software tornou-se o elemento chave da evolução dos sistemas e produtos baseados em computador. No decorrer das últimas quatro décadas, o software evoluiu de uma ferramenta de análise de informações e de resolução de problemas especializada para uma indústria em si mesma. Mas logo a cultura e a história da "programação" criaram um conjunto de problemas que persistem até hoje. O software tornou-se um fator limitante na evolução dos sistemas baseados em computador.

Parte dos problemas que ocorrem nas empresas que fornecem soluções baseadas em tecnologia da informação é em decorrência da velocidade com que as mudanças acontecem nesse setor de mercado, onde muitas vezes essas mudanças acabam não sendo assimiladas adequadamente pelos fornecedores de soluções baseadas em tecnologia da informação.

Junto a isto, nos últimos anos, o mercado globalizado está se tornando cada vez mais exigente e competitivo, fazendo com que as empresas fornecedoras de soluções baseadas em

tecnologia da informação procurem alternativas para melhorar a qualidade de seus processos e seus produtos, gerando assim benefícios aos clientes e usuários finais.

Com esta visão, Pressman (1995, p. 56) comenta que:

> Na maioria dos empreendimentos técnicos, as medições e as métricas ajudam-nos a entender o processo técnico usado para se desenvolver um produto, como também o próprio produto. O processo é medido, num esforço para melhorá-lo. O produto é medido, num esforço para aumentar sua qualidade.

Para que a atividade de desenvolvimento de software seja executada de forma adequada, seria necessário que existisse um grau de entrosamento entre os interessados no produto final, *stakeholders*[1] e equipe de desenvolvimento. Mas muitas vezes este entrosamento não ocorre e consequentemente o resultado final acaba não sendo o esperado, gerando assim futuros retrabalhos e adaptações no projeto após a sua entrega (VAZQUEZ: 2004).

Para encurtar a distância entre os envolvidos no processo de desenvolvimento de software, a engenharia de software tem evoluído na busca de diferentes metodologias que basicamente auxiliam, mas não garantem o sucesso de um projeto ou um produto de software. É muito comum ver projetos de desenvolvimento de software com foco maior em análise, projeto e implementação e muito pouco tempo aplicado para o controle da qualidade.

Com o uso de uma métrica, é possível obter diversos benefícios como a geração de dados estatísticos sobre o desenvolvimento do projeto ou produto de software, como por exemplo, informações sobre produtividade, gerenciamento, estimativa, qualidade, custos entre outros. Para Pressman (1995, p. 60) existem algumas razões pelas quais medimos um software:

> O software é medido por muitas razões: (1) indicar a qualidade do produto; (2) avaliar a produtividade das pessoas que produzem o produto; (3) avaliar os benefícios (em termos de produtividade e qualidade) derivados de novos métodos e ferramentas de software; (4) formar uma linha básica para estimativas; (5) ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional.

Stakeholders[1] *– Pessoas envolvidas em um determinado assunto de interesse comum. Partes interessadas.*

Os benefícios adquiridos através de um processo de medição são inúmeros, mas os principais são o de ter um histórico de dados para os processos e produtos da organização, e assim poder utilizá-los nos futuros projetos como referência.

Lord Kelvin apud Pressman (1995, p. 59) é bem direto ao relatar que:

> Quando se pode medir aquilo sobre o qual se está falando e expressá-lo em números, sabe-se alguma coisa sobre o mesmo; mas quando não se pode medi-lo, quando não se pode expressa-lo em números, o conhecimento que se tem é de um tipo inadequado e insatisfatório; este pode ser o começo do conhecimento, mas dificilmente alguém terá avançado em sua idéias para o estágio de ciência.

Para que as organizações que trabalham com processo de desenvolvimento de software e buscam um meio de alcançar um melhor custo/benefício para seus clientes, elas podem alcançar esses objetivos através da adoção e aplicação de alguma métrica de software, e assim obter o real o tamanho da aplicação e seu real valor. Mas é preciso atenção conforme comenta Sommerville (2003, p. 473):

> Um dos problemas com a coleta de dados quantitativos sobre software e projetos de software está em compreender o que esses dados realmente significam. É fácil interpretar mal os dados e produzir inferências incorretas. As medições devem ser cuidadosamente analisadas para se compreender o que elas de fato significam .

A mensuração vem de encontro com a necessidade das organizações obterem informações mais precisas sob diferentes aspectos de seus produtos de software e auxiliam os gestores, fornecendo informações significativas em relação à qualidade, adequação e progresso evolutivo de processos, produtos e projetos de software.

Segundo pesquisa realizada no ano de 2005 (Figura 1 a seguir) pelo Ministério da Ciência e Tecnologia do governo, atuando na área de Tecnologia da Informação e Comunicação, pode-se perceber que mais da metade das organizações que participaram da

pesquisa utilizavam algum tipo de métrica para auxiliar na medição da qualidade dos processos de software (MCT: 2006):

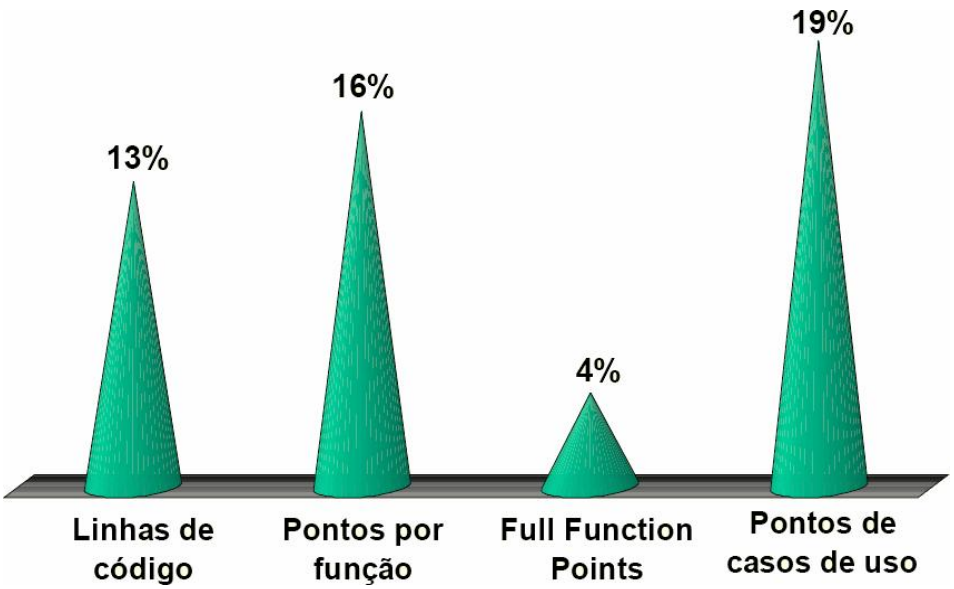


Figura 1 – Medição da Qualidade dos Processos de Software – Métricas Primitivas.

Fonte: Ministério da Ciência e Tecnologia do Brasil, 2006.

Esta amostra composta por 701 organizações abrange empresas desenvolvedores de software pacote (*packaged software*) ou software sob encomenda *(custom software).*

Nesta mesma pesquisa realizada, foi possível identificar o nível de conhecimento e utilização das normas mais utilizadas nos processos de software, como CMM[1], CMMI[2], ISO/IEC 12207[3] e SPICE[4]. Segue o gráfico:

CMM[1] – Do acrônimo em inglês de Capability Maturity Model, é uma metodologia de diagnóstico e avaliação de maturidade do desenvolvimento de softwares em uma organização.
CMMI[2] – Capability Maturity Model Integration. A proposta do CMMi é prover a melhoria dos processos das organizações e a habilidade de gerenciar, desenvolver e manter os seus produtos (software).
ISO/IEC 12207[3] – A ISO/IEC 12207 é a norma ISO/IEC que define processo de desenvolvimento de software.
SPICE[4] – Também conhecida como ISO/IEC 15504, é a norma ISO/IEC que define processo de desenvolvimento de software. Ela é uma evolução da ISO/IEC 12207, mas possui níveis de capacidade para cada processo assim como o CMMI.

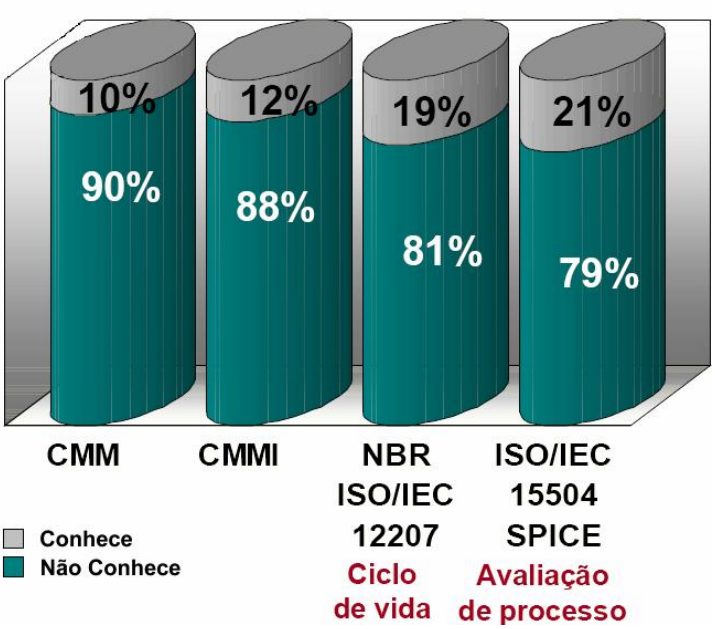


Figura 2 – Conhecimento e utilização de normas e modelos da qualidade em empresas no Brasil.

Fonte: Ministério da Ciência e Tecnologia do Brasil, 2006

Com essas informações coletadas por meio de uma pesquisa realizada pelo governo brasileiro no ano de 2005, percebe-se e comprova-se a pouca atenção dispensada para os processos de qualidade de software e também para os processos de desenvolvimento de software. Vale ressaltar que este cenário provavelmente melhorou nos anos seguintes a pesquisa, porém não existem dados oficiais que comprovem.

3 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Nos últimos anos, a engenharia de software tem evoluído consideravelmente, conquistando assim, um espaço e estudo como sendo uma disciplina fundamental para o desenvolvimento de software.

Nas primeiras décadas do surgimento dos computadores, o principal pensamento era de desenvolver hardwares que diminuíssem o custo dos processamentos e dispositivos de armazenagem de dados. Com o passar do tempo, pode-se ver que não apenas era necessário evoluir os dispositivos físicos, mas também os softwares desenvolvidos, buscando assim uma sinergia entre hardware e software.

Segundo Pressman (1995, p. 4):

> O poder de um computador mainframe da década de 1980 agora está à disposição sobre uma escrivaninha. As assombrosas capacidades de processamento e armazenagem do moderno hardware representam um grande potencial de computação. O software é o mecanismo que nos possibilita aproveitar e dar vazão a este potencial.

Para conseguir uma definição mais precisa do que é software, é necessário examinar as características do mesmo que o torna diferente do que os homens constroem. Para a construção de um hardware, todo o processo criativo elaborado acaba sendo transcritos em uma forma física.

Esta forma física é formada por um conjunto de componentes eletrônicos, circuitos integrados e placas, que se comunicam através de barramentos existentes no hardware (WIKIPÉDIA: 2006).

Já o software é formado da parte lógica, onde possui um conjunto de instruções e dados que são processados pelos circuitos eletrônicos do hardware. É reconhecido também

como programa de computador composto por uma ou mais seqüências de instruções a serem seguidas e/ou executadas, na manipulação, encaminhamento ou modificação de um dado ou informação. (WIKIPÉDIA: 2006).

Em um desenvolvimento de software, o resultado obtido após sua construção é de um sistema lógico, e não físico como na construção de um hardware.

Assim, nas duas atividades (construção de hardware e construção de software) a qualidade é alcançada com a elaboração e desenvolvimento de um bom projeto, mas na fase de manufatura (construção) do hardware, é possível existirem problemas de qualidade que seriam mais facilmente corrigidos caso estivesse realizada a construção de um software.

Para Sommerville (2003, p. 5), software é:

> Muita gente associa o termo software aos programas de computador. Na verdade, essa é uma visão muito restrita. Software não é apenas o programa mas também toda a documentação associada e os dados de configuração necessários para fazer com que esses programas operem corretamente. Um sistema de software, usualmente consiste em uma série de programas separados, arquivos de configuração que são utilizados para configurar esses programas, documentação do sistema, que descreve a estrutura desse sistema, e documentação do usuário, que explica como utilizar o sistema e, no caso dos produtos de software, sites Web para os usuários fazerem o download das informações recentes sobre o produto.

Os softwares de computador estão divididos em duas formas básicas: componentes não executáveis em máquina e componentes executáveis em máquina. Para Pressman (1995, p. 17) a definição para componentes é a seguinte:

> Os componentes de software são construídos usando uma linguagem de programação que tem um vocabulário limitado, uma gramática explicitamente definida e regras de sintaxe e semântica bem formadas. Esses atributos são essenciais para a tradução por máquina. As formas de linguagem em uso são as linguagens de máquina, linguagens de alto nível e a linguagens não-procedimentais.

Para entender a engenharia de software é necessário ter o conhecimento de um conjunto de três elementos fundamentais – métodos, ferramentas e procedimentos – o que

possibilita ao responsável um controle do processo de desenvolvimento e uma base para a construção de software de alta qualidade.

Em engenharia de software, os métodos envolvem um conjunto de tarefas que incluem o planejamento do projeto, estimativa, as análises de requisitos de software e de sistemas, projeto de estruturação dos dados, arquitetura de programa e algoritmo de processamento, codificação, testes e manutenção (PRESSMAN: 1995).

Já as ferramentas de engenharia de software auxiliam de forma automatizada ou semi-automatizada os métodos. Quando as ferramentas utilizadas para automatizar cada um dos métodos, citados no parágrafo anterior, e as mesmas possuem integração, fazendo com que o resultado produzido por uma destas ferramentas possa ser reaproveitado por outra, é estabelecido um sistema de suporte conhecido como CASE (Computer-Aided Software Engineering). O CASE é uma combinação software, hardware e um banco de dados que auxilia na criação de um ambiente de engenharia de software (PRESSMAN: 1995).

Os procedimentos são o elo de ligação entre os métodos e as ferramentas, possibilitando o desenvolvimento de software de computador. Eles definem a seqüência em que os métodos serão aplicados, os produtos que devem ser entregues e os controles que contribuem para assegurar a qualidade, gerando assim referências que ajudam os gerentes de software avaliar o progresso do mesmo (PRESSMAN: 1995).

Existem no mercado, diversos processos de desenvolvimento de software onde cada um possui suas vantagens e desvantagens. Alguns deles têm suas próprias particularidades e também melhor desempenho em determinado tipo de projeto a ser desenvolvido.

Dentre os modelos existentes no mercado, o ciclo de vida clássico, também conhecido como modelo cascata, é o paradigma mais antigo e consequentemente um dos mais utilizados.

Este modelo também é utilizado como base para diversos outros modelos de desenvolvimento.

Vale ressaltar o que Pressman (1995, p. 34) comenta:

> O ciclo de vida clássico é o paradigma mais antigo e o mais amplamente usado da engenharia de software. Porém, no decorrer da última década, as criticas ao paradigma fizeram com que até mesmo seus ativos defensores questionassem sua aplicabilidade em todas as situações.

Apresenta-se a seguir quatro dos principais processos de desenvolvimento de software: o ciclo de vida clássico, modelo de prototipação, modelo em espiral e modelo extreme programing (XP).

3.1 O CICLO DE VIDA CLÁSSICO

O ciclo de vida clássico foi modelado em função do ciclo de engenharia convencional, e possui as seguintes atividades conforme figura abaixo:

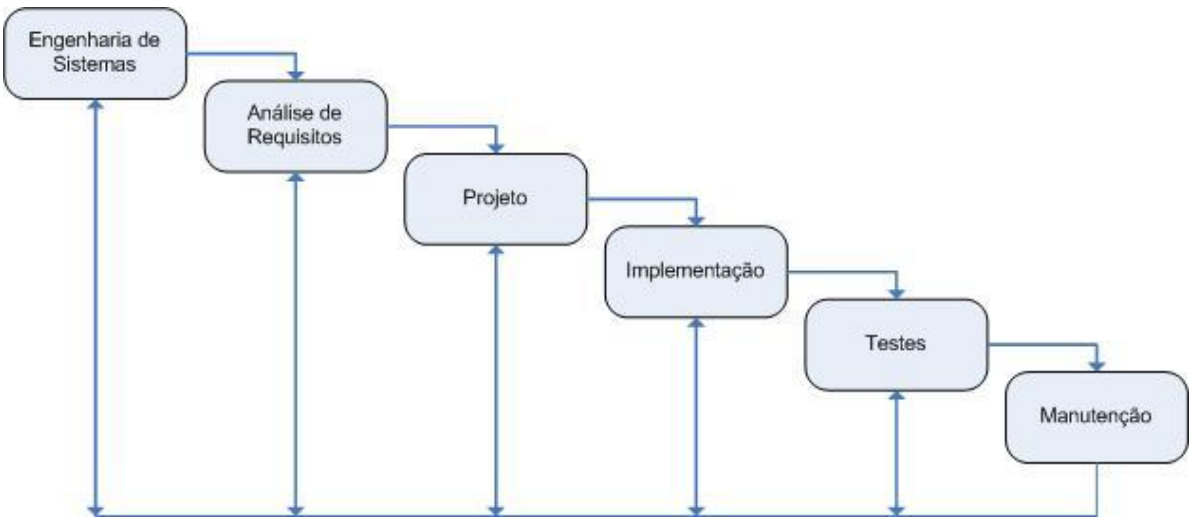


Figura 3 – O ciclo de vida clássico

Fonte: Victor Medeiros, 2006.

Engenharia de Sistemas: Nesta etapa o trabalho inicia-se com o estabelecimento dos requisitos para todos os elementos do sistema. Essa visão é bastante importante, em situações quando o software deve interagir com outros elementos como hardware, pessoas e banco de dados (PRESSMAN: 1995).

Análise de Requisitos: Nesta fase é realizada a coleta dos requisitos de forma concentrada especificamente no software. Para um melhor entendimento, esta fase pode ser dividida em cinco etapas de esforço distinto que são realizadas na seguinte seqüência: Reconhecimento do problema, Avaliação da Síntese, Modelagem, Especificação e Revisão. Os requisitos são documentados e revistos com o cliente (PRESSMAN: 1995).

Projeto: Nesta etapa o projeto também pode ser dividido em quatro menores passos que se concentram em atributos distintos do programa: a estrutura de dados, arquitetura de software, detalhes procedimentais e caracterização de interface. Também nesta etapa, o projeto é documentado e torna-se parte da configuração do software (PRESSMAN: 1995).

Implementação: Acontece a transcrição de todas as etapas anteriores em uma linguagem que possa ser interpretada pela máquina. É realizada a codificação das especificações atendendo os requisitos levantados pelo analista. Caso a etapa anterior for executada de forma detalhada, a codificação pode ser executada de forma mecânica (PRESSMAN: 1995).

Testes: Nesta etapa ocorrem os testes das codificações realizadas na etapa anterior. Este processo de testes tem por objetivo principal averiguar os aspectos lógicos internos do software, garantindo que todos os comandos e instruções tenham sido testados. Nesta fase também são avaliados os aspectos funcionais externos, garantindo que uma entrada definida produza os resultados exigidos pelo usuário/cliente (PRESSMAN: 1995).

Manutenção: Sempre ao final dos testes de funcionalidade do software o mesmo sofrerá mudanças após ter sido entregue ao usuário/cliente. Estas mudanças podem ser provenientes de muitos fatores, como mudança de regra de negócio interna da empresa, ou mudança de algum dispositivo (hardware) utilizado pelo sistema, entre outros. Para que estas ocorrências de mudança possam ser tratadas, a manutenção reaplica todas as etapas do ciclo de vida de um programa existente (PRESSMAN: 1995).

Assim como citado por Pressman, o ciclo de vida clássico, descrito acima não é perfeito e possui falhas. Veja o que Sommerville (2003, p. 38) comenta:

> O problema com o modelo em cascata é sua inflexível divisão do projeto nesses estágios distintos. Os acordos devem ser feitos em um estágio inicial do processo, e isso significa que é difícil responder aos requisitos do cliente, que sempre se modificam. Portanto, o modelo em cascata deve ser utilizado somente quando os requisitos forem bem compreendidos. Contudo, ele reflete a prática da engenharia. Consequentemente, os processos de software com base nesta abordagem ainda são utilizados no desenvolvimento de software, em particular quando fazem parte de um projeto maior de engenharia de sistemas.

Entretanto, o paradigma ciclo de vida clássico possui um lugar importante e bem definido na aplicação da engenharia de software. Embora este modelo tenha algumas particularidades questionáveis, ele é significativamente melhor do que uma simples abordagem casual ao desenvolvimento de software (PRESSMAN: 1995).

3.2 PROTOTIPAÇÃO

A prototipação é um processo de desenvolvimento de software que objetiva a criação um protótipo ou modelo de software que será posteriormente implantado. Existem três formas que este protótipo pode ser exteriorizado:

a) Protótipo em papel ou baseado em PC que retrata a interação homem-máquina, capacitando o usuário a entender as interações;
b) Protótipo que implementa algum subconjunto de uma função do software desejado pelo usuário, para assim visualizar parte da funcionalidade;
c) Programa já existente que executa toda ou parte da função desejada pelo usuário, que posteriormente serão melhoradas no futuro desenvolvimento (PRESSMAN: 1995).

Esta forma de exteriorização ainda pode ser classificada da seguinte forma:

a) Prototipação evolucionária – Fornecer um sistema funcional ao usuário final, iniciando-se pelos requisitos mais compreendidos e com maior prioridade. Os requisitos de menor prioridade serão desenvolvidos apenas quando solicitado pelo usuário;

b) Prototipação descartável – Seu objetivo é validar ou derivar os requisitos do sistema, iniciando o desenvolvimento pelos itens que não são bem compreendidos. Os requisitos de simples identificação podem nunca ser prototipados (SOMMERVILLE: 2003).

Com a primeira etapa concluída, o cliente / usuário pode avaliar de forma parcial o resultado final obtido com o desenvolvimento de um protótipo, percebendo o que o software irá lhe fornecer.

A figura a seguir apresenta os passos para uma prototipação evolutiva, onde esses passos podem ser repetidos indefinidas vezes até que haja o aceite do cliente.

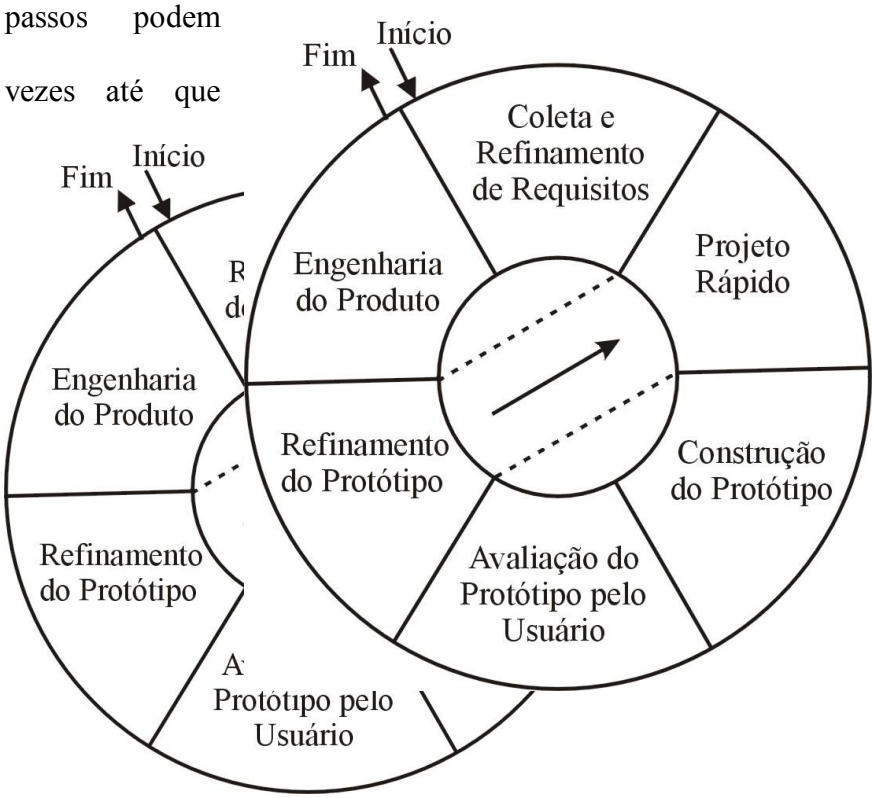


Figura 4 – A seqüência de eventos do modelo de prototipação evolutiva.

Fonte: Júnia Coutinho Anacleto Silva, 2006.

Assim como no modelo do ciclo de vida clássico, o modelo de prototipação também pode ser problemático devido algumas situações:

a) O cliente acredita e entende que o protótipo desenvolvido pode ser utilizado imediatamente, sem levar em consideração a qualidade global do software e manutenibilidade a longo prazo. Quando é dito ao cliente que o protótipo precisa ser reconstruído, levando as considerações as premissas da qualidade, o mesmo não compreende e pressiona para tornar o mesmo em um produto de trabalho;

b) Para o desenvolvimento de um protótipo, o desenvolvedor pode utilizar um sistema operacional ou linguagem de programação imprópria, apenas por que esta disponível no momento, pois o objetivo é operacionalizar o protótipo. Depois de algum tempo, o desenvolvedor pode esquecer dos reais motivos que a linguagem de programação é inadequada e acabar tornando o protótipo parte do sistema final (PRESSMAN: 1995).

Em muitas situações, o modelo de prototipação evolucionário é mais eficaz que o modelo de ciclo de vida clássico, pois acabam por produzir sistemas que atendem as principais necessidades dos clientes de forma rápida. Seu desenvolvimento pode ser feito de forma evolutiva. Com a compreensão do que já esta desenvolvido, é possível entender melhor os problemas futuros a serem resolvidos durante a construção do sistema (SOMMERVILLE: 2003).

3.3 MODELO ESPIRAL

O modelo de desenvolvimento de software em espiral foi desenvolvido para agrupar as melhores características do modelo de prototipação e também do modelo cascata.

Neste modelo também foi adicionado um novo elemento que é a análise dos riscos durante o processo de desenvolvimento de software.

A figura a seguir ilustra a seqüência de atividades que compõem o modelo em espiral.

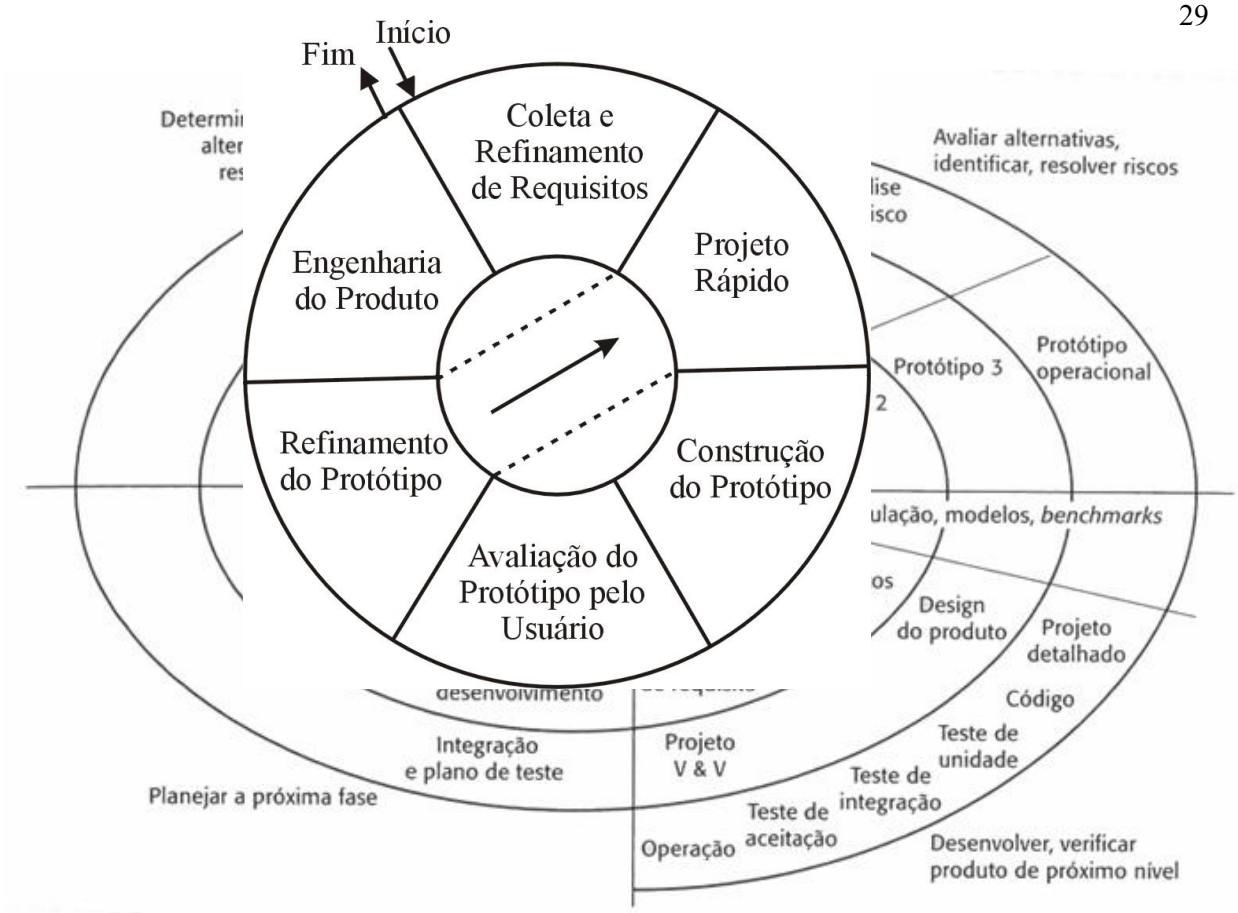


Figura 5 – Modelo em Espiral.

Fonte: Sommerville, 2003, p. 47.

A principal distinção deste modelo em relação aos demais processos de desenvolvimento de software é a consideração dos riscos na sua execução (SOMMERVILLE: 2003).

O modelo em espiral para a engenharia de software é um paradigma que possuí uma abordagem mais realística para o desenvolvimento de software em larga escala (PRESSMAN: 1995).

3.4 MODELO EXTREME PROGRAMING

Extreme Programing, também conhecido como XP trata de uma metodologia de desenvolvimento que possui foco principal em agilidade de equipes e qualidade de projetos. Utiliza como apoio os valores como simplicidade, comunicação, feedback. É uma metodologia baseada em comportamentos e atitudes (MEDEIROS: 2006).

A seguir apenas algumas sugestões de boas práticas utilizadas na metodologia XP:

a) O cliente sempre disponível para colaborar em momentos de dúvidas;
b) Uso de metáforas no projeto visando facilitar a comunicação da equipe;
c) Pequenas versões. O cliente recebe pequenas releases do sistema;
d) Testes de aceitação. São definidos no início do projeto que definem os critérios de aceitação do software;
e) Simplicidade de projeto. O código está claro e simples a qualquer momento do projeto, seguindo os padrões estabelecidos pela equipe de desenvolvimento (MEDEIROS: 2006).

Como comentado anteriormente estas são algumas práticas que podem ser adotadas no processo de desenvolvimento XP e ainda somadas com outras técnicas existentes no mercado, gerando assim um modelo próprio de desenvolvimento.

O modelo de desenvolvimento não é uma metodologia estática, é dinâmico, possibilitando que cada um modele sua forma de trabalhar com XP. O que a metodologia sugere são algumas práticas existentes em seu modelo padrão e deixa a escolha do usuário que técnicas ele poderá escolher, podendo fazer um "mix" de práticas XP com práticas pessoais adquiridas com experiências anteriores em projetos ou ainda práticas vindas de modelos como RUP[1], CMM e PMI[2] para a melhoria do processo de software (MEDEIROS: 2006).

RUP[1] – O RUP, abreviação de Rational Unified Process (ou Processo Unificado da Rational), é um processo proprietário de Engenharia de software criado pela Rational Software Corporation, adquirida pela IBM tornando-se uma brand na área de Software, fornecendo técnicas a serem seguidas pelos membros da equipe de desenvolvimento de software com o objetivo de aumentar a sua produtividade.
PMI[2] - Project Management Institute (PMI) – Instituto de Gerenciamento de Projeto estabelecido em 1969 e situado nos arredores da Filadélfia, Pensilvânia e Estados Unidos.

4 PROCESSO X PRODUTO X PROJETO

Um dos principais objetivos da engenharia de software é, sem dúvida, melhorar a qualidade do software. Mas a qualidade de produtos de software está fortemente relacionada à qualidade do processo de software (FUGGETTA: 2000).

Para muitos engenheiros de software, a qualidade do processo de desenvolvimento de software é tão importante quanto à qualidade do produto final. Com esta visão, houve uma preocupação elevada com a modelagem e melhorias no processo de software no decorrer dos anos.

Existem no mercado diversas normas e modelos, que se aplicadas ao processo de software, provavelmente trarão grandes benefícios e consequentemente o produto de software terá mais qualidade. Seguem algumas delas:

a) Norma ISO 9000[1];
b) Norma ISO/IEC 12207;
c) Modelo CMM (Capability Maturity Model);
d) Modelo SPICE (Software Process Improvement and Capability dEtermination);
e) Modelo PMBOK[2] (ROCHA: 2001).

Segundo Rocha (2001, p. 2) a preocupação com o processo de software é algo importante por alguns motivos:

> A preocupação com o processo de software está relacionada a necessidade de entender, avaliar, controlar, aprender, comunicar, melhorar, predizer e certificar o nosso trabalho como engenheiros de software. Para isso é preciso documentar, definir, medir, analisar, avaliar, comparar e alterar os processos.

ISO 9000[1] – A série ISO 9000 é um conjunto de normas que formam um modelo de gestão da qualidade para organizações que podem, se desejarem, certificar seus sistemas de gestão através de organismos de certificação (tais como o BVQI, A.B.S, Loyds, ou o DNV, entre outros).
PMBOK[2] – Project Management Body of Knowledge (PMBOK) é um padrão de Gerência de Projetos desenvolvido pelo Project Management Institute (PMI).

Mas um único modelo não serve para qualquer tipo de empresa ou projeto a ser implementado. Existem muitas variáveis que influenciam, como: a cultura da organização; os objetivos do projeto; os recursos disponíveis; a experiência da equipe envolvida entre outros fatores (ROCHA: 2001).

Para que se possa compreender melhor um processo de desenvolvimento de software é necessário investigar todas as atividades que estão ligadas ao processo bem como suas relações. Nos itens a seguir serão abordados de forma breve os conceitos referentes a processo, produto e projeto, finalizando com algumas considerações sobre essa relação.

4.1 PROCESSO

A engenharia de software é uma disciplina que pode ser vista de forma objetiva e simples, onde utilizando os princípios básicos da engenharia deve-se desenvolver um produto de software de maneira sistemática e com baixo custo, resultando em um produto final confiável e eficiente (ROCHA: 2001).

Para o processo de desenvolvimento, um requisito básico de um processo de qualidade é que o mesmo deva ser sistemático e que sua repetição seja possível, independente de quem o execute (ROCHA: 2001).

Como citado no capitulo anterior, existem diversos modelos de processos de desenvolvimento de software, cada um com suas características e particularidades. Independente do paradigma escolhido, da área de atuação da aplicação, do tamanho do projeto e complexidade, o processo de desenvolvimento de software possuí três fases genéricas:

a) Definição. Tem por objetivo levantar as informações que serão processadas, identificar quais as funções e o desempenho esperado delas, definir interfaces, restrições do projeto e validações necessárias.

b) Desenvolvimento. Nesta etapa devem ser definidas como serão as estruturas de dados, a arquitetura do software. Deve ser escolhida a linguagem de programação, como as definições serão codificadas e também como serão realizados os testes.

c) Manutenção. Procura focar seu objetivo nas mudanças, sejam correções provenientes de erros ou adaptações necessárias decorrentes de mudanças no ambiente ou através de novas solicitações do usuário final (ROCHA: 2001).

Segundo Sommerville (2003, p. 7) existem diferentes processos:

> Diferentes processos de software organizam essas atividades de maneiras diversas e são descritos em diferentes níveis de detalhes. Os prazos das atividades variam, do mesmo modo que os resultados de cada atividade. Diferentes organizações podem utilizar processos diferentes para produzir o mesmo tipo de produto. No entanto, alguns processos são mais adequados do que outros para alguns tipos de aplicação. Se um processo inadequado for utilizado, isso provavelmente reduzirá a qualidade ou a utilidade do produto de software a ser desenvolvido.

É importante ressaltar que para se alcançar a qualidade e produtividade de um processo de software, é necessário levar em consideração além dos aspectos essencialmente técnicos, os aspectos como características da organização, do pessoal técnico envolvido e do domínio de aplicação. Caso isto não ocorra, o produto também pode ser afetado (ROCHA: 2001).

4.2 PRODUTO

A qualidade de um produto de software é resultante das atividades realizadas no processo de desenvolvimento do mesmo. Para avaliar a qualidade de um produto de software é necessário verificar através de técnicas e atividades operacionais, o quanto os requisitos definidos são atendidos. Estes requisitos podem ser de natureza quantitativa ou qualitativa que tem por objetivo definir as características de um software.

Na avaliação do produto, segundo norma ISO/IEC 9126-1[1], são consideradas algumas características que acabam por definir e avaliar a qualidade do produto de software caso as mesmas sejam atendidas. A seguir as características da norma ISO/IEC 9126-1:

a) Funcionalidade. O software precisa satisfazer as necessidades explicitas e implícitas do usuário
b) Confiabilidade. Garantir com que o software funcione por um determinado período atendendo o que foi pré-estabelecido.
c) Usabilidade. O usuário precisa interagir de forma fácil com o software
d) Eficiência. É necessário verificar se o software não está desperdiçando recursos do sistema como um todo.
e) Manutenibilidade. Para futuras implementações e manutenções o software precisa ser fácil e simples de se alterar.
f) Portabilidade. Poder utilizar o software em diversas plataformas de forma bastante flexível (PRESSMAN: 1995).

Para produto de software, é importante dizer que é fundamental que sua qualidade seja independente de quem o desenvolveu (ROCHA: 2001).

ISO/IEC 9126-1[1] – fornece características e sub-características de qualidade, sendo uma norma essencialmente de definições.

4.3 PROJETO

Para que se consiga alcançar os objetivos de qualidade, é necessário que exista um controle que organiza e determina todo o andamento do processo de desenvolvimento até o momento da conclusão do produto final. Este controle é realizado pelo projeto, que gerencia e controla o desenvolvimento de software.

Segundo Prado (2004, p. 15), o conceito de projetos é:

> Projetos são esforços temporários que ocorrem em todas as organizações. Executar projetos é uma característica de sobrevivência da empresa moderna e saber gerenciar projetos é uma necessidade marcante dos executivos. Tom Peters, renomado consultor norte-americano, em sua recomendação de carreira para executivos diz: "se você não despende 70% do seu tempo com projetos você vive no passado".

Quando se fala em projetos, é necessário falar também sobre gerenciamento de projetos. Gerenciar um projeto significa planejar sua execução antes de começar a executá-lo e no decorrer da sua execução acompanhar e controlar seu desenvolvimento (PRADO: 2004).

Para Sommerville (2003, p. 60), os gerentes de projeto são os responsáveis por esta atividade de acompanhar e controlar, conforme ele comenta:

> A necessidade de gerenciamento é uma importante distinção entre o desenvolvimento profissional de software e a programação em nível amador. Necessitamos do gerenciamento de projetos de software porque a engenharia de software profissional está sempre sujeita a restrições de orçamento e de prazo. Essas restrições são estabelecidas pela organização que desenvolve o software. O trabalho do gerente de projeto de software é garantir que o projeto de software cumpra essas restrições e entregar um produto de software que contribua para as metas da empresa.

Como um exemplo de ferramenta para o auxílio deste gerenciamento, pode-se citar o livro *A Guide to the Project Management Body of Knowledge* ou *Guia para o Universo do Conhecimento de Gerenciamento de Projetos,* mais conhecido como *PMBOK Guide*. Este

guia não é uma metodologia, mas sim uma padronização de processos, áreas de conhecimento, técnicas e regras. Em 1999 ele foi reconhecido como um padrão de gerenciamento de projetos pelo ANSI (American National Standards Institute) (PRADO: 2004).

4.4 CONSIDERAÇÕES

Existem atualmente diversas métricas para medição de desempenho para os itens projeto, produto e processo. Os resultados obtidos através da aplicação das mesmas acabam respondendo de uma forma analítica e interpretativa se determinado projeto obteve sucesso ou não. Vale lembrar que todo projeto sempre será diferente do outro, causando assim a necessidade de levar em consideração a interpretação dos resultados e as variáveis que são pertinentes a cada projeto (ROCHA: 2001).

É importante destacar que o principal objetivo do gerenciamento de projetos é poder garantir que o produto resultante do projeto, seja ele um bem ou serviço, seja obtido de acordo com o que foi planejado, no que diz respeito a escopo, prazo, custo e qualidade. O produto, que pode ser um bem ou serviço, é o que se espera no encerramento de um projeto (PRADO: 2004).

5 MÉTRICAS DE SOFTWARE

Métrica de software é uma medida de alguma propriedade do software ou da sua especificação. É bastante utilizada para realizar orçamentos, medir o desempenho dos desenvolvedores entre outros indicadores (WIKIPÉDIA: 2006).

Um dos maiores problemas enfrentados pelos gerentes de projetos de software estão ligados de forma direta com a estimativa dos prazos de entrega e custos dos projetos. Somam-se a isso, as constantes mudanças no mercado e o surgimento de novas tecnologias para o desenvolvimento de software. Por maior que seja a experiência do profissional que está atuando como gerente de projetos, ele acabará tendo dificuldades no processo de estimativas de prazos e custos, pois cada projeto executado é diferente e possui suas variáveis e particularidades.

A realidade que se percebe no mercado atual é de um setor muito competitivo e extremamente dinâmico, em que as empresas desenvolvedoras de software precisam adaptar-se as mudanças rapidamente para continuarem competitivas no mercado.

Nesse cenário, é necessário que as empresas busquem constantemente uma melhoria na qualidade do processo de desenvolvimento de software por meio da adoção de padrões, normas e ferramentas, fazendo com que as empresas que tenham acordado prazos de entrega e custos estabelecidos com seus clientes possam cumpri-los. A busca pela redução nos prazos de entrega e dos custos, mantendo a qualidade do software, são fatores importantes para conquistar vantagens consideráveis sobre os concorrentes.

Para DeMarco (1989, p. 51) o conceito de métrica é o seguinte:

> Métrica é o número que você vincula a uma idéia. Mais precisamente, é uma indicação mensurável de algum aspecto quantitativo do sistema. Para o projeto de um software comum, os aspectos quantitativos onde mais precisamos usar a métrica são: escopo, tamanho, custo, risco e tempo empregado.

Com o auxílio de métricas de software é possível obter informações para serem analisados e utilizados como apoio pelos gerentes de projetos de software.

Os motivos que levam uma organização, equipe ou gerente de projetos a utilizar alguma métrica de software estão baseados na necessidade de compreender, avaliar, estimar e melhorar seus projetos de forma geral.

A mensuração é algo muito importante em qualquer área de engenharia, inclusive na engenharia de software, onde um dos objetivos a serem alcançados com a mensuração é uma melhoria nos níveis de qualidade do projeto como um todo. O termo métricas de software engloba um conjunto de mensurações que podem servir tanto para processo, como produto e também projeto.

Porém, para uma implantação eficaz de um processo de medição, é necessária a motivação de todos os profissionais que atuam no processo de desenvolvimento de sistemas (HAZAN: 1999).

É importante ressaltar que no processo de medição as pessoas não são avaliadas, mas sim o processo e para isto é necessário que as pessoas envolvidas devem estar cientes e confiantes desta visão, caso contrário os dados obtidos através da medição, não indicarão a realidade.

Para a implantação de alguma métrica é necessário verificar sua importância e sua utilidade dentro do contexto do projeto ou da organização. Além da definição e aplicação de uma métrica, é necessário determinar as regras de utilização da mesma e a interpretação dos resultados obtidos através de sua aplicação.

Os principais fatores que devem ser considerados na escolha das métricas são:

a) Detalhamento dos objetivos que precisam ser atingidos com a utilização da métrica;
b) As métricas devem prover resultados consistentes;
c) Devem ser de fácil entendimento;
d) Precisam ser objetivas, minimizando a influência de julgamentos pessoais;
e) As métricas devem ser efetivas no custo, ou seja, o resultado obtido deve compensar e exceder o esforço de utilização da métrica;
f) Precisam fornecer informações que evidenciem a situação atual e sejam relevantes para a tomada de decisões;
g) As métricas devem ser apropriadas para cada etapa do ciclo de vida de um software;
h) Devem servir para a realização de estimativas;
i) Devem possibilitar a obtenção de séries históricas (SIMÕES: 1999).

É importante ressaltar que apesar dos grandes benefícios que podem ser obtidos com a utilização de práticas de mensuração no gerenciamento dos projetos e processos de software, o fator humano é um aspecto muito sensível e que deve ser levado em consideração, pois caso seja desconsiderado, pode conduzir ao insucesso as tentativas de mensuração.

5.1 MÉTODOS E TÉCNICAS PARA MENSURAÇÃO DE SOFTWARE

As empresas que trabalham com a engenharia de software estão, durante anos, buscando técnicas e métodos quantitativos aceitáveis e consistentes para medir a eficiência e

eficácia dos processos e produtos e para gerenciar também os sistemas por elas adquiridos, desenvolvidos, melhorados ou mantidos.

Existem no mercado diversos tipos de métodos e técnicas que são utilizados para mensurar o tamanho de um software. Segue abaixo uma lista com alguns métodos e técnicas conhecidos e uma breve explicação sobre os mesmos:

Contagem de Linhas de Código Fonte: Esta técnica é a mais antiga forma existente para medir um fator que é considerado crítico em um programa de mensuração: o tamanho de um sistema. A técnica tem como vantagem a simplicidade, porém está totalmente ligada à linguagem de programação utilizada e com a forma de implementação, tendo praticamente nenhum significado para o usuário final (SCHROEDER: 2003).

Além disso, ela só pode ser aplicada após a codificação dos programas de um produto, tornando-se inviável para estimativas de projetos (SCHROEDER: 2003).

COCOMO: O método de COCOMO (Constructive Cost Model) tem sua teoria baseada no número de instruções fonte (número de linhas de código) e fundamentada em fórmulas matemáticas, podendo ser aplicado nas diversas fases do ciclo de desenvolvimento. O COCOMO está dividido em três modelos, conforme abaixo:

a) Cocomo Básico: Modelo que pode ser aplicado a grande maioria de projetos de software de pequeno e médio porte. As estimativas fornecidas por este método são limitadas, pois desconsidera alguns atributos importantes pertinentes ao projeto como, por exemplo, as restrições de hardware e a qualificação e experiência das pessoas envolvidas no projeto.
b) Cocomo Intermediário: Este modelo é uma versão melhorada do modelo básico, abrangendo as limitações contidas no outro modelo e também suas estimativas são mais precisas.

c) Cocomo Detalhado: É uma técnica que permite realizar estimativas em nível de módulo, subsistema e sistema, inclusive mostrando os atributos de custo em cada fase do projeto (SCHROEDER: 2003).

Em contrapartida, o método apresenta algumas desvantagens, pois depende da tecnologia aplicada, depende de experiências anteriores e não produz indicadores (SCHROEDER: 2003).

Modelo GQM (Goal/Question/Metrics): O requisito básico dessa abordagem é que a mensuração deve ser orientada a metas, ou seja, toda coleta de dados deve estar baseada em um fundamento lógico muito bem definido e documentado.

Os seguintes princípios devem ser considerados e executados para se obter sucesso e vantagens de um programa de mensuração baseado em GQM:

a) A execução da tarefa de análise precisa ser especificada precisamente e explicitamente através de uma meta de mensuração;
b) As medidas precisam ser derivadas de uma forma top-down baseada em metas e perguntas. Uma estrutura de metas e perguntas não pode ser adaptada de forma retroativa a partir de um conjunto de medidas existente;
c) Cada medida precisa ter um fundamento lógico relacionado e devidamente documentado. O fundamento lógico é usado para justificar a coleta de dados e guiar a análise e interpretação posterior;
d) Os dados coletados precisam ser interpretados de uma forma bottom-up no contexto das metas GQM e das perguntas;
e) As pessoas que vão utilizar os resultados do programa de mensuração precisam estar envolvidas profundamente na definição e interpretação do programa (SCHROEDER: 2003).

Este modelo possui como uma das suas principais vantagens a identificação das métricas consideradas relevantes e úteis dentro de uma empresa, uma vez que suporta a

análise e interpretação dos dados coletados, permitindo assim uma melhor compreensão de seus produtos e processos internos (SCHROEDER: 2003).

Como desvantagens deste modelo podemos citar que modelo apóia seu desenvolvimento de software baseando-se no paradigma do Ciclo de Vida Clássico e também que o gerenciamento da equipe de desenvolvimento trabalha sem nenhum tempo ocioso, o que pode ser perigoso para o projeto como um todo (MELLER: 2002).

Modelo de Estimativa de Putnam: O modelo de estimativa Putnam é um modelo dinâmico e que permite equacionar matematicamente a distribuição de esforço necessário para o desenvolvimento de um projeto. O modelo possui múltiplas variáveis e pressupõe uma distribuição de esforço específica para cada fase do ciclo de vida de um software (SCHROEDER: 2003).

Assim como o COCOMO, o modelo de estimativa de Putnam se baseia em estimativas do número de linhas de código. O modelo foi elaborado no inicio a partir de informações de distribuição de esforço proveniente de grandes projetos, porém, a sua aplicação em projetos de menor porte é totalmente viável (SCHROEDER: 2003).

Technical Use Case Point: A TUCP (Pontos de Caso de Uso Técnicos) utiliza na sua essência a geração de estimativas através de casos de uso. Com a utilização desta técnica é possível chegar a um maior detalhamento das estimativas nas principais etapas existentes em um ciclo de vida de software, possibilitando assim um acompanhamento mais preciso do projeto (PADUELLI: 2006).

O nível de precisão na utilização TUCP é dependente direto da existência da organização possuir um documento como modelo onde sejam especificados os casos de usos e sua utilização, como também é necessário o entendimento claro do que seja o conceito de transação em uma aplicação avaliada por esta técnica (PADUELLI: 2006).

Análise de Ponto de Função: A APF mede apenas o tamanho funcional do software. Como comparativo, pode-se dizer que da mesma forma, onde somente os metros quadrados de uma construção não são suficientes para um construtor administrar uma obra. Apenas

Pontos de Função são insuficientes para um desenvolvedor de software administrar um projeto de software (VAZQUEZ: 2004).

O tamanho funcional só é realmente relevante quando utilizado em conjunto com outras medidas a fim de produzir outras métricas normalizadas que possuam sentido ao desenvolvedor ou gerente de projeto. Como exemplo pode-se citar o esforço, prazo e custo como sendo informações normalmente selecionadas como objeto da medição (VAZQUEZ: 2004).

No capítulo seguinte, será apresentada a métrica de software de Análise de Ponto de Função, utilizando como base a versão 4.2 do Manual de Práticas de Contagem do *IFPUG - International Function Point Users Group.*

6 APF – ANÁLISE DE PONTOS DE FUNÇÃO

A análise de pontos de função é uma métrica de software utilizada por profissionais da área de sistemas e também por empresas de todos os portes e segmentos da economia. Seu uso inicial era com foco em estimativas para projetos, mas com o passar dos anos ganhou espaço para ser aplicada também com sucesso em unidade de medição de contratos de desenvolvimento de software e como ferramenta de gerenciamento de projetos (VAZQUEZ: 2004).

É uma medida de tamanho, de claro significado ao nível do negócio. A APF quantifica as funções contidas no software em termos significativos para os respectivos usuários do mesmo.

A métrica relaciona-se diretamente aos requisitos do negócio que o software pretende tratar, onde dessa forma, a APF é imediatamente aplicável a um amplo espectro de ambientes e ao longo da vida de um projeto de desenvolvimento, desde a definição inicial dos requisitos até a fase de plena utilização operacional.

Também podem ser derivadas outras medidas úteis ao negócio, tais como a produtividade do processo de desenvolvimento e o custo unitário de suporte ao software.

A norma ISO/IEC 20926 reconhece a APF como uma métrica funcional sem a utilização das CGS (características gerais do sistema) e consequentemente do valor de VAF (valor do fator de ajuste) que serão explicados no item 6.2.5 deste trabalho.

Em resumo, a técnica de pontos de função fornece uma medida objetiva e comparável que auxilia a avaliação, planejamento, gerência e controle no desenvolvimento de software.

6.1 A HISTÓRIA DA APF

A técnica de Análise de Ponto de Função surgiu no início da década de 70, na empresa *IBM*[1] como sendo uma alternativa as métricas já existentes que são baseadas em linhas de código conhecidas como LOC (BFPUG: 2006).

Nesta época, Allan Albrecht foi encarregado de medir a produtividade nos projetos de desenvolvimento realizados pela empresa *IBM*. O maior problema que encontrado por Allan era a diversidade de linguagens utilizadas nos desenvolvimentos, o que tornava inviável uma análise em conjunto de produtividade dos projetos (BFPUG: 2006).

Com esta situação, surgiu a necessidade de uma métrica de software que fosse independente da linguagem de programação utilizada, tornando assim a medição de produtividade viável (BFPUG: 2006).

Após a criação da técnica no início dos anos 70, a mesma passou a ser aperfeiçoada e foi cada vez mais ganhando adeptos, o que acabou causando a criação em 1986 do *IFPUG – International Function Point Users Group* (BFPUG: 2006).

Com a criação do grupo, a técnica passou a ter maior aceitação no mercado o que fez com os participantes deste grupo lançassem em 1990 a primeira versão do Manual de Práticas de Contagem ou CPM (*Counting Practices Manual*) com o objetivo de padronizar a técnica de contagem. A partir deste momento este é o padrão reconhecido pelo mercado para contagem de Pontos de Função (BFPUG: 2006).

No Brasil, a utilização da técnica iniciou-se no começo da década de 90 com o apoio da empresa *Unisys*[2]. De 1991 até 1994 foram realizados seis ENUPF – Encontro Nacional de Usuários de Ponto de Função (BFPUG: 2006).

IBM[1] – International Business Machines - é uma empresa americana de informática.
Unisys[2] – É uma grande empresa produtora de computadores. Resultou da união em 1986 das empresas Sperry Rand Corporation e Burroughs Corporation (empresas norte-americanas).

A técnica APF teve sua abrangência e visibilidade aumentada a partir do momento em que grandes contratos públicos passaram a ser baseados em pontos de função.

No ano de 1998 foi criado o *BFPUG – Brazilian Function Point Users Group*, onde até hoje é organiza e regulamenta a técnica de APF no país, através das definições vindas do *IFPUG*.

Atualmente existem centenas de associados entre estudantes, desenvolvedores, consultores e gerentes que participam de discussões sobre o assunto (BFPUG: 2006).

6.2 O PROCESSO DE CONTAGEM DE PONTOS DE FUNÇÃO

Para a realização de uma contagem de pontos de função é necessário que se tenha uma visão geral dos passos necessários para que a mesma possa ser realizada.

De uma forma resumida os passos necessários para uma contagem podem ser vistos na figura a seguir:

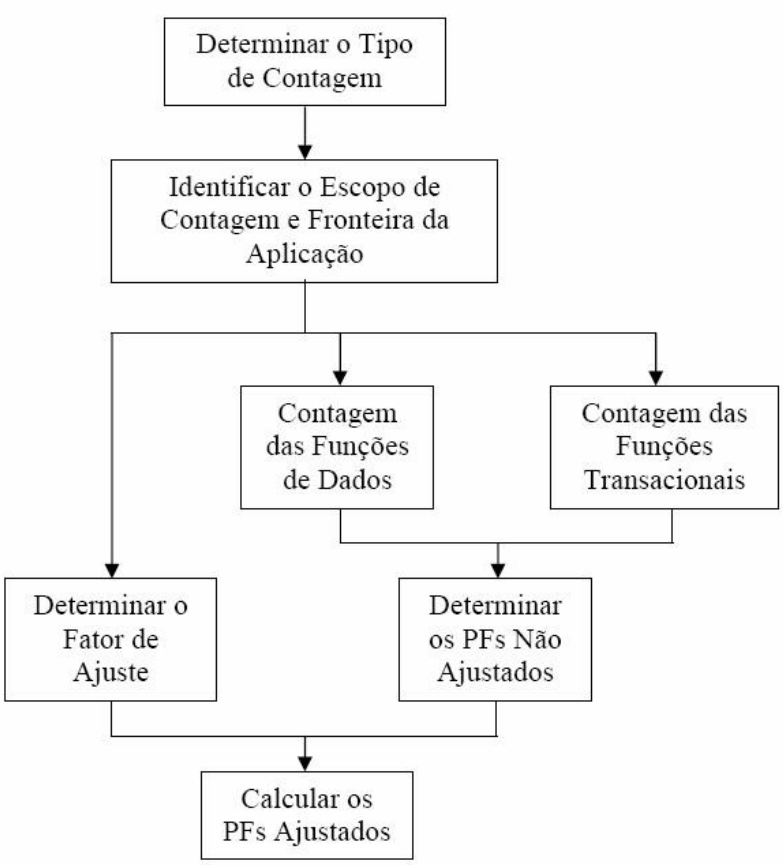


Figura 6 – Processo de Contagem APF.

Fonte: Claudia Hazan, 2006.

A técnica APF pode ser aplicada em projetos que estão em sua fase de elaboração e definição, bem como também em projetos já concluídos e que precisam ter seu tamanho funcional definidos para a geração de mais informações sobre produtividade.

Para que seja realizada uma contagem é necessário realizar uma abstração do que é possível ser contado em uma aplicação, conforme as regras definidas pela técnica APF conforme figura seguinte:

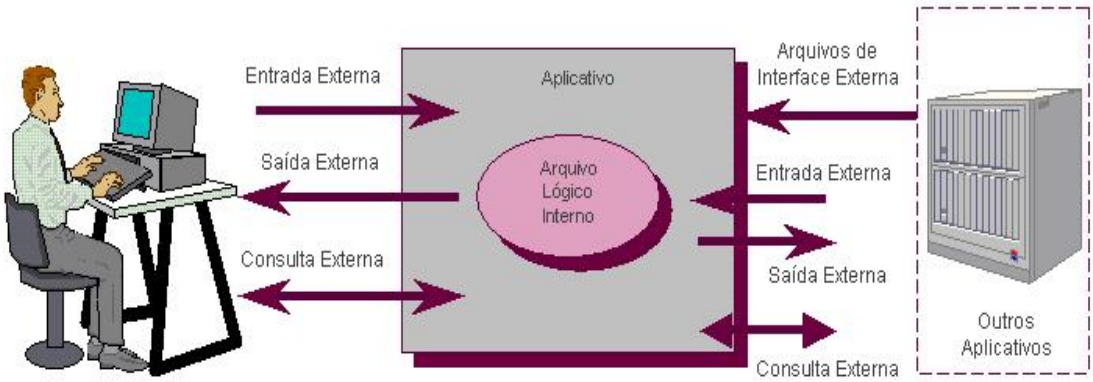


Figura 7 – Formas de interação com o sistema a ser contado.

Fonte: Guia de contagem APF Datasul, 2005, p. 1.

Em uma contagem APF o seu fim não termina em si mesmo. Sempre existe uma motivação maior que seja necessária à realização da contagem para que se possa alcançar este objetivo maior. Quanto mais preciso for este objetivo final, mais simples será achar a resposta para um problema de negócio (VAZQUEZ: 2004).

Antes de iniciar-se qualquer atividade de contagem é necessário definir algumas premissas como:

a) Determinar o tipo de contagem: projeto de desenvolvimento, projeto de melhoria ou aplicação;
b) Determinar o escopo da contagem: abrangerá uma ou mais aplicações ou apenas parte de uma aplicação;
c) Determinar o nível de detalhamento da contagem. Caso seja necessária uma auditoria posterior, é importante que o processo de contagem esteja bem documentado (HAZAN: 2006).

6.2.1 Determinar o tipo de contagem

Para iniciar-se um processo de contagem é necessário definir o tipo de contagem que será utilizada para medir o software. A contagem de pontos de função pode estar associada tanto a projetos quanto a aplicações. Existem três tipos de contagens:

a) Contagem de um projeto de desenvolvimento;

b) Contagem de um projeto de melhoria;

c) Contagem de uma aplicação (baseline).

Em um projeto de desenvolvimento a contagem de Pontos de função determina as funcionalidades fornecidas aos usuários finais do software na sua primeira instalação, envolvendo também possíveis funções de conversão ou migração de dados que sejam necessárias.

Mas é preciso ter atenção neste tipo de contagem como comenta Vazquez (2004, p. 56):

> É importante destacar que qualquer contagem realizada antes do término de um projeto é na verdade uma estimativa da funcionalidade que será entregue ao seu final. Conforme os requisitos vão ficando mais claros durante o projeto, e as funções vão sendo desenvolvidas, é bastante natural identificar funcionalidades que não haviam sido especificadas inicialmente. Esse fenômeno é chamado de scope creep.

Já em uma contagem de um projeto de melhoria, o número de pontos de função será medido pelas funções adicionadas, modificadas ou excluídas do sistema em questão pelo projeto. Ao final do projeto de melhoria, o número de pontos de função contados deve ser atualizado na contagem original do software para que as alterações efetuadas reflitam o novo número final de pontos de função do software.

E por último, em uma contagem de aplicação (baseline), a mesma mede a funcionalidade fornecida aos usuários por uma aplicação já instalada. É também conhecida como pontos de função instalados. Este tipo de contagem é iniciado logo após o final da

contagem do projeto de desenvolvimento, sendo atualizado ao final de todo o projeto de melhoria que cause alguma alteração das funcionalidades da aplicação (VAZQUEZ: 2004).

6.2.2 Identificar o escopo da contagem e fronteira da aplicação

Como segunda tarefa para realização de uma contagem, deve ser definido o escopo do que se pretende contar. Isto pode ser entendido como todas as funcionalidades disponíveis em um sistema, ou somente partes dele como, por exemplo, as funcionalidades mais utilizadas pelos usuários ou ainda algumas funcionalidades específicas determinadas como transações de manutenção, relatórios, cadastros, etc.

Após definido o escopo do que será contado, deve-se definir uma fronteira da aplicação em questão, para que se iniciem os trabalhos de contagem. Segundo Vazquez (2004, p. 58), ele conceitua que:

> A fronteira da aplicação é a interface conceitual que delimita o software que será medido e o mundo exterior (seus usuários). É importante ressaltar que o termo usuário deve ser entendido como qualquer pessoa ou coisa (por exemplo, outra aplicação) que interaja com a aplicação, recebendo ou enviando dados. Quando há mais de uma aplicação incluída no escopo de uma contagem, várias fronteiras devem ser identificadas.

Para determinar a fronteira de uma aplicação é necessário seguir algumas regras:

a) A determinação fronteira deve ser feita com base no ponto de vista do usuário. O foco deve estar no que ele compreende e sabe descrever, conforme sua visão;
b) A separação das funções deve ser estabelecida conforme os processos de negócio, não considerando as características tecnológicas;

c) Em projetos de melhoria, a fronteira estabelecida no início do projeto, deve ser a mesma para a aplicação que está sendo modificada (PRACTICES MANUAL: 2006).

A identificação da fronteira da contagem é um passo essencial para a medição funcional de uma aplicação. A contagem deve ser baseada em uma perspectiva de negócio, não levando em consideração os aspectos técnicos. O não posicionamento correto de uma fronteira pode alterar a perspectiva da uma contagem de uma visão lógica para uma visão física, acarretando erros como duplicidade de funções e arquivos contados em várias aplicações (VAZQUEZ: 2004).

6.2.3 Contagem das funções do tipo dado

Para a contagem das funções do tipo dado, é necessário que sejam compreendidos alguns conceitos. Para Vazquez (2004, p. 75):

> As funções do tipo dado representam a funcionalidade fornecida ao usuário para atender a suas necessidades de dados internos e externos à aplicação. São classificadas em Arquivos Lógicos Internos (ALI) e Arquivos de Interface Externa (AIE).

O nome *arquivo* utilizado para esta definição, não significa que seja um arquivo do sistema operacional. Este termo refere-se a um grupo de dados organizados de forma lógica e que são reconhecidos pelo usuário.

Através da visão da APF, um arquivo pode estar mapeado em um ou mais arquivos físicos do sistema operacional ou em tabelas do banco de dados. A forma com que a aplicação implementa ou resolve essas funcionalidades, seja de forma agrupada ou distribuída, não é relevante para a determinação das funções do tipo dado (VAZQUEZ: 2004).

6.2.3.1 Identificação de um arquivo lógico interno (ALI)

Um ALI é um grupo de dados ou de informações de controle que estão logicamente relacionados, mantidos dentro da fronteira da aplicação e reconhecidos pelo usuário.

Para definir um ALI, pode-se afirmar que sua principal intenção é armazenar dados mantidos por um ou mais processos elementares que aplicação que está sendo contada (PRACTICES MANUAL: 2006).

Para a compreensão melhor do que é um ALI, segue abaixo alguns exemplos:

a) Tabelas que registram dados mantidos pela aplicação;
b) Arquivos de configuração mantidos pela aplicação;
c) Arquivos de mensagem de erros, desde que mantidos pela aplicação (VAZQUEZ: 2004).

E também exemplos que não atendem os requisitos de um ALI:

a) Arquivos temporários, de trabalho ou classificação;
b) Arquivos de backup;
c) Arquivos de índices;
d) Arquivos introduzidos exclusivamente em função da tecnologia utilizada (VAZQUEZ: 2004).

Resumidamente um ALI é mantido dentro da fronteira da aplicação que está sendo contada.

6.2.3.2 Identificação de um arquivo de interface externa (AIE)

Para a definição de um AIE, pode-se considerar que eles também são um grupo de dados ou de informações de controle que estão logicamente relacionados, e que são referenciados (lidos) pela aplicação e que são reconhecidos pelo usuário. Vazquez (2004, p. 77) explica que: “A principal intenção de um AIE é armazenar dados referenciados por meio de um ou mais processos elementares dentro da aplicação sendo contada. Isto é, o AIE deve obrigatoriamente ser um ALI de outra aplicação”.

Para a compreensão melhor do que é um AIE, seguem alguns exemplos:

a) Dados de referência externos utilizados pela aplicação;
b) Arquivos de Ajuda, desde que mantidos por outra aplicação;
c) Arquivos de mensagens de erro, desde que mantidos por outra aplicação (VAZQUEZ: 2004).

E também exemplos que não atendem os requisitos de um AIE:

a) Dados mantidos pela aplicação e utilizados por outra aplicação;
b) Dados formatados e processados para o uso de outras aplicações;
c) Arquivos de movimento recebidos de outra aplicação para manter um ALI, no entanto, estes podem ser considerados como funções do tipo transação (VAZQUEZ: 2004).

Resumidamente um AIE é mantido fora da fronteira da aplicação que está sendo contada, ou seja, é mantido por outra aplicação onde ele é um ALI.

6.2.3.3 Determinação da complexidade

Com a definição dos ALI e AIE de uma determinada aplicação, conforme escopo e fronteira pré-estabelecidos é necessário determinar o nível de complexidade dos mesmos.

Para a determinação da complexidade funcional (baixa, média ou alta) devem ser levados em consideração os seguintes itens abaixo:

a) Número de Tipos de Dados (TD);

b) Número de Tipos de Registros (TR) (HAZAN: 2006).

Com as informações de quantidades de TD e TR determinadas, é possível classificar sua complexidade conforme tabela abaixo:

Tabela 1 – Tabela de complexidade funcional dos ALI e AIE

Tipos de Registros	Tipos de Dados			
		< 20	20 - 50	> 50
	1	Baixa	Baixa	*Média*
	2 - 5	Baixa	*Média*	Alta
	> 5	*Média*	Alta	Alta

Fonte: Carlos Eduardo Vazquez, 2004, p. 81

Para que seja feita a contagem dos tipos de dados de forma correta, é preciso saber que um TD é um campo único, reconhecido pelo usuário e não repetido. Com base neste conceito existem algumas regras que precisam ser seguidas conforme abaixo:

a) Na execução de um processo elementar, deve ser contado um tipo de dado para cada campo único reconhecido pelo usuário e não repetido, recuperado ou mantido por um ALI ou AIE;

b) Quando existem duas aplicações que mantém ou referenciam o mesmo ALI ou AIE, devem ser contados apenas os campos usados por cada aplicação;

c) Para cada campo solicitado pelo usuário para estabelecer um relacionamento com outro arquivo, deve ser contado um tipo de dado (PRACTICES MANUAL: 2006).

Para a contagem de um tipo de registro (TR) é necessário entender que um tipo de registro é um subgrupo de tipos de dados, que é reconhecido pelo usuário, que está inserido em um ALI ou um AIE.

Existem dois tipos de subgrupos de tipos de dados. São eles:

a) Opcionais: o usuário tem a opção de não informar no processo elementar que adiciona dados ao arquivo;
b) Obrigatórios: os mesmos são sempre utilizados pelo processo elementar que adiciona dados ao arquivo (PRACTICES MANUAL: 2006).

Para a identificação dos tipos de registros devem ser seguidas duas regras:

a) Deve ser contado um tipo de registro para cada subgrupo, sendo ele obrigatório ou opcional, e ainda que seja de um ALI ou AIE.
b) Se não existir nenhum subgrupo para ser contado, conte o próprio ALI ou AIE como sendo um tipo de registro (PRACTICES MANUAL: 2006).

Como exemplo, pode-se citar um ALI que contenha 47 tipos de dados e um tipo de registro ele terá complexidade baixa e um AIE que contenha 52 tipos de dados e um tipo de registro terá complexidade média.

6.2.3.4 Determinação da contribuição

Ao final de da identificação dos tipos de dados (TD) e os tipos de registros (TR) de um ALI ou de um AIE, é possível obter o seu grau de complexidade e assim chegar-se ao valor de contribuição de um arquivo. Segue a tabela de contribuição.

Tabela 2 – Contribuição dos pontos de função não-ajustados das funções do tipo dado

Tipo de Função	Baixa	Média	Alta
Arquivo Lógico Interno	7 PF	10 PF	15 PF
Arquivo de Interface Externa	5 PF	7 PF	10 PF

Fonte: Carlos Eduardo Vazquez, 2004, p. 85

Como exemplo, pode-se citar que um ALI que possui complexidade alta, o mesmo contribui com 15 pontos de função não-ajustados, enquanto um AIE que tem complexidade alta contribui com 10 pontos de função não-ajustados.

6.2.4 Contagem das funções transacionais

Não menos importantes, as funções transacionais precisam ser identificadas para que possam ser relacionadas aos ALI e AIE conforme explicado anteriormente na seção 6.2.3.

Para melhor entender o conceito de funções transacionais, Vazquez (2004, p. 89) diz que:

> As funções do tipo transação representam à funcionalidade fornecida ao usuário para atender as suas necessidades de processamento de dados pela aplicação. São classificadas em Entradas Externas, Saídas Externas ou Consultas externas.

Uma transação que ocorre em um sistema, segundo a APF, sempre será uma entrada externa (EE), saída externa (SE) ou ainda uma consulta externa (CE). Esta transação chamada de processo elementar (PE) que é a menor unidade com significado para o usuário final e deve ser completo na ação que está acontecendo deixando a aplicação em estado consistente ao final da sua execução. Veja o que Vazquez (2004, p. 92) comenta sobre processo elementar:

> Esta definição é a mais importante para a contagem das funções do tipo transação. Se por um lado ela evita que vários processos elementares sejam contados como um único, por outro lado evita que subprocessos, componente de processos elementares também sejam contados .

Para que se tenha certeza de que um processo elementar é único, pelo menos uma das três proposições a baixo deve ser obrigatoriamente verdadeira, caso contrário, deve-se contar apenas um PE.

a) A lógica de processamento é diferente da executada pelo novo processo elementar. Por exemplo, relatórios onde a diferença seja apenas a ordenação dos dados representam apenas um único processo elementar;
b) O conjunto de tipos de dados identificado é diferente do identificado para outros processos elementares da mesma aplicação.
c) Os ALI e AIE são diferentes dos arquivos referenciados por outros processos elementares da mesma aplicação (PRACTICES MANUAL: 2006).

6.2.4.1 Identificação de uma entrada externa (EE)

Uma Entrada Externa (EE) pode ser reconhecida como sendo um processo elementar, que processa dados ou informações de controle e que são recebidos de fora da fronteira da

aplicação. Sua principal função é manter um ou mais arquivos lógicos internos (ALI) e/ou modificar o comportamento do sistema (PRACTICES MANUAL: 2006).

Para a um melhor entendimento do que é uma EE, segue abaixo alguns exemplos:

a) Tela que permite ao usuário incluir, alterar ou excluir registros em arquivos. Neste caso seriam três entradas externas;
b) Atualizações em lotes de bases cadastrais (VAZQUEZ: 2004).

E também exemplos que não atendem os requisitos de um EE:

a) Telas de filtro de relatórios e também de consultas;
b) Telas de login;
e) Menus (VAZQUEZ: 2004).

6.2.4.2 Identificação de uma saída externa (SE)

Uma saída externa (SE) pode ser vista pelo usuário como algo que envia dados para fora da fronteira da aplicação. Sua intenção principal é apresentar dados para o usuário através de alguma lógica de processamento. Este tipo de função deve apresentar cálculos, criar dados derivados ou ainda alterar o comportamento do sistema (HAZAN: 2006).

Para a uma melhor compreensão do que é uma SE, segue abaixo alguns exemplos:

a) Consultas que apresentem dados derivados ou calculados;
b) Arquivo de movimento gerado para outra aplicação;
c) Informações em formato gráfico;
d) Relatórios que contenham totalização de dados (VAZQUEZ: 2004).

E também exemplos que não atendem os requisitos de um SE:

a) Telas de Ajuda;
b) Consultas e relatórios que não contenham cálculo ou totalizadores, não atualizam arquivos ou alterem o comportamento do sistema (VAZQUEZ: 2004).

6.2.4.3 Identificação de uma consulta externa (CE)

Uma consulta externa (CE) é um processo elementar que envia dados para fora da fronteira de aplicação. Ela apresenta informações ao usuário através de uma recuperação de dados ou informações de controle, sem efetuar uma nenhuma alteração no estado do sistema (PRACTICES MANUAL: 2006).

Para um melhor entendimento do que é uma CE, segue abaixo alguns exemplos:

a) Telas de Ajuda;
b) Telas de logon;
c) Menu gerado dinamicamente usando como base uma configuração da aplicação (VAZQUEZ: 2004).

E também exemplos que não atendem os requisitos de um CE:

a) Relatórios ou consultas que não contenham cálculos ou que gerem dados derivados;
b) Menus estáticos, colocados de forma fixa (VAZQUEZ: 2004).

6.2.4.4 Determinação da complexidade

Com o final do reconhecimento dos processos elementares (PE) de uma aplicação, como sendo do tipo entrada externa (EE), saída externa (SE) e consulta externa (CE), seguindo escopo e fronteira pré-estabelecidos é necessário determinar o nível de complexidade dos mesmos.

Para a determinação da complexidade funcional (baixa, média ou alta) devem ser levados em consideração os seguintes itens abaixo:

c) Número de Arquivos Referenciados (AR);

d) Número de Tipos de Dado (TD) (HAZAN: 2006).

Para a definição dos AR pertencentes a um PE, não é necessário, neste caso, a definição da sua complexidade e contribuição. Basta verificar o número total de arquivos que foram referenciados.

Com as informações de quantidades de AR e TD determinadas, é possível classificar a complexidade de uma EE conforme a tabela abaixo:

Tabela 3 – Tabela de complexidade funcional para entrada externa

Arquivos Referenciados	Tipos de Dados			
	< 5	5 - 15	> 15
< 2	Baixa	Baixa	*Média*
2	Baixa	*Média*	Alta
> 2	*Média*	Alta	Alta

Fonte: Carlos Eduardo Vazquez, 2004, p. 97

Como exemplo, pode-se citar uma EE que contenha 18 tipos de dados e dois arquivos referenciados ele terá complexidade alta e uma outra EE que contenha 18 tipos de dados e um arquivo referenciado terá complexidade média (VAZQUEZ: 2004).

Para a determinação de complexidade de uma SE e uma CE, deve ser utilizada a seguinte tabela abaixo:

Tabela 4 – Tabela de complexidade funcional para saídas externas e consultas externas

Arquivos Referenciados	Tipos de Dados			
	< 6	6 - 19	> 19
< 2	Baixa	Baixa	*Média*
2 - 3	Baixa	*Média*	Alta
> 3	*Média*	Alta	Alta

Fonte: Carlos Eduardo Vazquez, 2004, p. 97

Como exemplo, pode-se citar uma SE que contenha 16 tipos de dados e dois arquivos referenciados ele terá complexidade média e uma outra SE que contenha 16 tipos de dados e um arquivo referenciado terá complexidade baixa. Para uma CE com 5 tipos de dados e três arquivos referenciados, sua complexidade será baixa (VAZQUEZ: 2004).

6.2.4.5 Determinação da contribuição

Ao final da identificação dos tipos de dados (TD) e arquivos referenciados (AR) de um processo elementar (PE), é possível obter-se o seu grau de complexidade e assim chegar-se ao valor de contribuição.

Segue abaixo a tabela de contribuição para entradas externas, saída externas e consultas externas.

Tabela 5 – Contribuição dos pontos de função não-ajustados das funções do tipo transação

Tipo de Função	Baixa	Média	Alta
Entrada Externa	3 PF	4 PF	6 PF
Saída Externa	4 PF	5 PF	7 PF
Consulta Externa	3 PF	4 PF	6 PF

Fonte: Carlos Eduardo Vazquez, 2004, p. 99

Como exemplo, pode-se citar que um EE que possui complexidade alta, o mesmo contribui com 6 pontos de função não-ajustados, enquanto um SE que tem complexidade alta contribui com 7 pontos de função não-ajustados.

6.2.5 Determinar o fator de ajuste

Para determinar o valor do fator de ajuste – VAF (Value Adjustment Factor) são utilizadas 14 características gerais de sistema, conhecidas como CGS. Estas características fazem referencia as funções que afetam a aplicação de maneira geral, independente de serem funções do tipo dado ou funções do tipo transação (HAZAN: 2006).

Abaixo segue a tabela com as CGS:

Tabela 6 – Tabela de características gerais do sistema

1. Comunicação de Dados	8. Atualizações On-Line
2. Processamento Distribuído	9. Processamento Complexo
3. Performance	10. Reutilização
4. Configuração Altamente Atualizada	11. Facilidade de Instalação
5. Taxa de Transações	12. Facilidade de Operação
6. Entrada de Dados On-Line	13. Múltiplos Locais
7. Eficiência do Usuário Final	14. Modificações Facilitadas

Fonte: Carlos Eduardo Vazquez, 2004, p. 105

Para cada uma destas CGS, deve ser definido um grau de importância e influência sobre a aplicação que pode variar entre zero e cinco conforme tabela abaixo:

Tabela 7 – Tabela de nível de influência de cada CGS

0	Nenhuma influência
1	Influência mínima
2	Influência moderada
3	Influência média
4	Influência significativa
5	Grande influência

Fonte: Carlos Eduardo Vazquez, 2004, p. 106

Para efetuar o cálculo do fator de ajuste deve ser utilizada a seguinte fórmula abaixo:

$$VAF = (TDI * 0,01) + 0,65$$

Onde VAF é o valor do fator de ajuste, TDI (Total Degree of Influence) Nível Total de Influência é o somatório do nível de influência de todas as 14 características gerais do sistema (HAZAN: 2006).

Para melhor entendimento, segue um exemplo demonstrando a forma de cálculo do VAF. Em sistema qualquer, foram identificados os pesos referentes a cada CGS conforme tabela abaixo:

Tabela 8 – Tabela de peso das características gerais do sistema de um sistema qualquer

CGS	Peso	CGS	Peso
1. Comunicação de Dados	4	8. Atualizações On-Line	4
2. Processamento Distribuído	3	9. Processamento Complexo	2
3. Performance	5	10. Reutilização	0
4. Configuração Altamente Atualizada	3	11. Facilidade de Instalação	0
5. Taxa de Transações	2	12. Facilidade de Operação	3
6. Entrada de Dados On-Line	5	13. Múltiplos Locais	2
7. Eficiência do Usuário Final	5	14. Modificações Facilitadas	3

Fonte: o Autor, 2006

Com os pesos definidos para cada CGS, teremos um total de 41 que é representado pelo TDI. Aplicando a fórmula do fator de ajuste teremos:

$$VAF = (41 * 0,01) + 0,65$$

$$VAF = 1,06$$

Apesar de a técnica APF contemplar a existência do VAF, o mesmo tornou-se opcional no final do ano de 2002, pois várias das CGS contemplam requisitos não-funcionais.

Existem diversas discussões sobre o VAF, onde o mesmo pode não estar representando de forma real os valores ali encontrados devido as CGS não estarem atualizadas e revisadas.

Sobre este assunto, Vazquez (2004, p. 107) comenta que:

> Este é um ponto tão criticado da técnica que há um grupo de trabalho do comitê de práticas de contagem do IFPUG dedicado ao assunto. Duas razões para a criação desse grupo são: a grande variação na interpretação da CGS pelos membros do IFPUG e a constatação de que algumas CGSs estão desatualizadas. O produto esperado para este grupo de trabalho é uma das seguintes possíveis recomendações: deixar tudo do jeito que está, fornecer um plano de atualização das CGSs ou eliminá-las por completo da técnica.

Abaixo segue uma lista de algumas críticas levantadas através de pesquisas realizadas referente ao VAF e as CGSs, tanto do ponto de vista teórico como o prático:

a) Existem requisitos muito importantes que não estão contemplados em nenhuma das 14 CGSs;
b) Algumas CGSs são de natureza muito próxima ou inter-relacionadas. Por exemplo, Performance e Volume de Transações;
c) A variação é de mais ou menos 35% para representar as funcionalidades gerais da aplicação;
d) O uso do fator de ajuste não adiciona nenhum benefício nos pontos de função não-ajustados para se fazer uma estimativa de esforço (VAZQUEZ: 2004).

6.2.6 Calcular os pontos de função ajustados

Este é o último passo a ser executado para finalizar o processo de contagem de um projeto, seja ele de desenvolvimento, melhoria ou aplicação.

As fórmulas que serão apresentadas nos próximos itens são exatamente iguais como no manual de contagem produzido pelo IFPUG. Nenhuma variável teve seu nome alterado ou traduzido, visto que não existe uma tradução oficial para o português.

6.2.6.1 Projeto de desenvolvimento

Para se efetuar o cálculo de um projeto de desenvolvimento são necessárias algumas informações, conforme abaixo:

a) Funções utilizadas após a instalação do software;
b) Funções que estão disponíveis no momento da instalação da aplicação para conversão de dados. Após a instalação, estas informações são descartadas;
c) O valor do fator de ajuste da aplicação, conforme as 14 características gerais do sistema (VAZQUEZ: 2004).

De posse destas informações, segue a fórmula utilizada para calcular os pontos de função ajustados:

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

Em que:

a) DFP: Número de pontos de função do projeto de desenvolvimento;
b) UFP: Número de pontos de função não-ajustados das funções disponíveis após a instalação;
c) CFP: Número de pontos de função não-ajustados das funções de conversão;
d) VAF: Valor do fator de ajuste (PRACTICES MANUAL: 2006).

6.2.6.2 Projeto de Melhoria

Assim como em um projeto de desenvolvimento, em um projeto de melhoria são necessárias algumas informações para que se possa calcular o valor de pontos de função ajustados conforme abaixo:

a) Funções que foram adicionadas, alteradas ou excluídas do projeto de melhoria;
b) Funcionalidades de conversão necessárias para a instalação do projeto de melhoria;
c) O valor do fator de ajuste da aplicação, conforme as 14 características gerais do sistema (VAZQUEZ: 2004).

Para identificar o número de pontos de função um projeto de melhoria não é necessário saber o tamanho da aplicação para que seja definido o tamanho do projeto de melhoria. Em outra situação é que as funções excluídas no projeto irão diminuir o tamanho da aplicação, mas irão aumentar o tamanho do projeto de melhoria (VAZQUEZ: 2004).

Com as informações coletadas, é possível então chegar ao valor de pontos de função ajustados através da seguinte fórmula:

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

Em que:

a) EFP: Número de pontos de função do projeto de melhoria;

b) ADD: Número de pontos de função não-ajustados das funções incluídas pelo projeto;

c) CHGA: Número de pontos de função não ajustados das funções modificadas. Este reflete também para as funções afetadas após as modificações;

d) CFP: Número de pontos de função não-ajustados adicionados pela conversão;

e) VAFA: Valor do fator de ajuste depois de aplicado o projeto de melhoria;

f) DEL: Número de pontos de função não-ajustados das funções excluídas pelo projeto de melhoria;

g) VAFB: Valor do fator de ajuste da aplicação antes do projeto de melhoria (PRACTICES MANUAL: 2006).

6.2.6.3 Projeto de Aplicação

Para calcular o valor de pontos de função ajustados de um projeto de aplicação, são utilizadas duas fórmulas. A primeira é utilizada para a contagem inicial antes de um projeto de melhoria. Segue abaixo a primeira fórmula:

$$AFP = ADD * VAF$$

Em que:

a) AFP: Número de pontos de função ajustados da aplicação;

b) ADD: Pontos de função não-ajustados das funções instaladas;

c) VAF: Valor do fator de ajuste da aplicação (PRACTICES MANUAL: 2006).

Para a aplicação desta fórmula, Vazquez (2004, p. 124) comenta que:

> Esta fórmula representa todas as funcionalidades requisitadas pelo usuário de uma aplicação instalada. Essa aplicação pode ser um pacote de software, um software recentemente desenvolvido e instalado ou um software instalado há algum tempo e que já sofreu diversas manutenções. Funções de conversão de dados não fazem parte da aplicação. Observe que pela fórmula do projeto de desenvolvimento, se não houve conversão de dados, o tamanho do projeto de desenvolvimento será igual ao da aplicação.

A segunda fórmula utilizada para projetos de aplicação é utilizada depois de concluído um projeto de melhoria. Abaixo segue a fórmula:

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

Em que:

a) AFP: Número de pontos de função ajustados da aplicação;

b) UFPB: Número de pontos de função não-ajustados da aplicação antes da execução do projeto de melhoria;

c) ADD: Número de pontos de função não-ajustados das funções incluídas no projeto de melhoria;

d) CHGA: Número de pontos de função não-ajustados das funções alteradas pelo projeto de melhoria após o seu término;
e) CHGB: Número de pontos de função não-ajustados das funções alteradas pelo projeto antes do seu término;
f) DEL: Número de pontos de função não-ajustados das funções excluídas pelo projeto de melhoria;
g) VAFA: Valor do fator de ajuste após a conclusão do projeto de melhoria (PRACTICES MANUAL: 2006).

Para esta fórmula aplicada após o projeto de melhoria, Vazquez (2004, p. 125) comenta que:

> Quando um projeto de melhoria é concluído, o número de pontos de função da aplicação deve ser atualizado para refletir as modificações na aplicação. Deve-se destacar novamente que mesmo que o projeto de melhoria tenha funções de conversão de dados, elas não fazem parte da aplicação.

Para melhor compreensão, abaixo seguem os itens que definem quando uma funcionalidade da aplicação pode ser alterada:

a) Quando uma nova funcionalidade é adicionada e assim aumentando o tamanho da aplicação;
b) Quando uma funcionalidade já existente é excluída e assim diminuindo o tamanho da aplicação;
c) Uma funcionalidade que foi modificada pode aumentar, diminuir ou não afetar o tamanho da aplicação;
d) Mudanças que podem ocorrer no valor do fator de ajuste que pode também aumentar, diminuir ou não afetar o tamanho da aplicação (VAZQUEZ: 2004).

É importante lembrar que após a realização de um projeto de melhoria, é possível que a aplicação continue com o seu mesmo tamanho funcional em pontos de função.

7 ESTIMATIVAS EM APF

Assim como em outras métricas de software, a APF também é utilizada para a realização de estimativas, onde através dos valores estimados é possível obter indicadores estimados de custo, prazo entre outros.

Em uma organização o surgimento da necessidade de um desenvolvimento, alguma manutenção, ou até mesmo uma aquisição de uma aplicação para uma posterior customização devem ser levados em consideração alguns questionamentos importantes:

a) Os requisitos levantados, realmente traduzem a realidade as necessidades solicitadas pelos usuários?;
b) A equipe de desenvolvimento esta preparada e possui conhecimento do negócio a ser executado?;
c) Existe conhecimento por todos os integrantes da equipe da tecnologia a ser utilizada, com documentação de apoio para o aprendizado da mesma? (VAZQUEZ: 2004).

Com estes questionamentos Vazquez (2004, p. 144) ainda comenta que:

> Devido a estas e outras particularidades inerente a cada projeto, determinar com exatidão seu custo final e data de conclusão é uma tarefa que só pode ser realizada quando ele já está finalizado. Antes disso, tudo que pode ser feito, até os cálculos mais rigorosos, são estimativas. Seja qual for o método empregado, o que muda é a precisão e a acurácia dos resultados obtidos quando comparado aos dados reais.

Para consulta, encontra-se no ANEXO A deste trabalho, informações sobre a algumas modalidades de contratação de projetos atualmente utilizadas pelo mercado, com suas principais vantagens e desvantagens.

A seguir serão abordados métodos utilizados para a realização de estimativas para tamanho de produto de software, bem como estimativa de esforço para a realização de um projeto.

7.1 ESTIMATIVA DO PRODUTO A SER GERADO

Para se ter estimativas efetivas de um produto de software, pode-se utilizar uma técnica simples e rápida que é classificada como um método direto, chamado de analogia simples. Ela consiste em fazer comparativos do projeto atual com projetos similares feitos no passado, através de uma base histórica de projetos (VAZQUEZ: 2004).

Para existir eficácia neste método, é necessário que o tamanho do projeto da base histórica esteja correto e que o projeto atual seja similar em seu conteúdo ao da base histórica de projetos.

Sobre as técnicas mais utilizadas, Vazquez (2004, p. 147) afirma que: "... as técnicas mais utilizadas para a estimativa de tamanho são encontradas na categoria dos métodos derivado".

Existem diversas técnicas que são classificadas como métodos derivados. Segue abaixo três modelos de métodos derivados que são baseados em ponto de função:

Contagem Dedutiva: Esta técnica considera de um único componente para a aplicação da mesma, geralmente um ALI, e a partir dele, deriva-se o resto da contagem a partir de bases estatísticas. De forma resumida, esta técnica considera 35 pontos de função para cada ALI e 15 pontos de função para cada AIE, como sendo uma média definida por este tipo de contagem.

Complexidade Média: Para este tipo de estimava existem duas abordagens diferentes. Para a realização de ambas, é necessário a identificação de todas as funcionalidades do projeto. A primeira abordagem é baseada na versão cinco do ISBSG Benchmark (International Software Benchmarking Standards Group) onde é apresentada uma tabela com a complexidade média para os projetos de desenvolvimento conforme tabela a seguir:

Tabela 9 – Tabela de média de PF não-ajustados, por tipo de função, do ISBSG comparando com a tabela de complexidade do IFPUG

Tipo de Função	Média de PF	IFPUG (baixa)	IFPUG (média)	IFPUG (alta)
ALI	7,4	7	10	15
AIE	5,5	5	7	10
EE	4,3	3	4	6
SE	5,4	4	5	7
CE	3,8	3	4	6

Fonte: Carlos Eduardo Vazquez, 2004, p. 147

Seu cálculo seria realizado através da seguinte fórmula:

e = #EE x 4,3 + #SE x 5,4 + #CE x 3,8 + #ALI x 7,4 + #AIE x 5,4

Em que:

a) e – Estimativa;

b) #EE – Número total de entradas externas;

c) #SE – Número total de saídas externas;

d) #CE – Número total de consultas externas;

e) #ALI – Número total de arquivos lógicos internos;

f) #AIE – Número total de arquivos de interface externa.

A NESMA (Nederlandse Software Metrieken Associatie) simplificou esta abordagem utilizada a tabela de complexidade do IFPUG. Defini-se que cada ALI e AIE tem complexidade baixa, o que resulta em 7 PF para um ALI e 5 PF para um AIE. Já para as EE,

SE e CE, é utilizada a complexidade média, o que gera 4 PF para uma EE, 5 PF para uma SE e 4 PF para uma CE.

Seu cálculo seria realizado através da seguinte fórmula:

$$e = \#EE \times 4 + \#SE \times 5 + \#CE \times 4 + \#ALI \times 7 + \#AIE \times 5$$

Em que:

a) e – Estimativa;

b) #EE – Número total de entradas externas;

c) #SE – Número total de saídas externas;

d) #CE – Número total de consultas externas;

e) #ALI – Número total de arquivos lógicos internos;

f) #AIE – Número total de arquivos de interface externa.

Backfiring: Este método tem por finalidade derivar o número de pontos de função com base no seu número de linhas de código (LOC). Consiste em contar as linhas de código de uma aplicação e utilizar um fator de conversão constante que depende da linguagem de programação utilizada.

Vazquez (2004, p. 148) descreve a respeito dos dois primeiros métodos apresentados que:

> Tanto a contagem dedutiva quanto a técnica de complexidade média são baseadas em pontos de função não-ajustados para a obtenção das estimativas, uma vez que o fator de ajuste pode ser determinado sem muita dificuldade mesmo em estágios iniciais do projeto.

Sobre o método Backfiring, Vazquez (2004, p. 148) ressalta que:

> Entretanto, não raro, essa técnica apresenta erros consideráveis quando confrontada com uma contagem manual dos pontos de função de uma aplicação. Isso porque assume uma relação linear entre tamanho funcional e tamanho físico, que são grandezas distintas.

O que ainda pode ocorrer no método Backfiring é uma variação de até 100% nos fatores de conversão para cada linguagem, dependendo da interpretação de cada autor que os define.

7.2 ESTIMATIVA DO ESFORÇO PARA EXECUÇÃO DO PROJETO

Para identificar o esforço que será empregado a um determinado projeto, é necessário que se tenha uma previsão ou estimativa do que será desenvolvido. Geralmente estima-se o esforço total de um projeto pelo somatório de todas as estimativas de esforço de cada atividade do projeto. Como base para este processo, o grande aliado é a experiência individual de cada um dos responsáveis por elaborar as estimativas (VAZQUEZ: 2004).

Também existem no mercado alguns métodos diretos para estimar o esforço aplicado em um projeto. Segue abaixo dois deles:

Delphi: É uma técnica de grupo interativa, onde um participante pode melhorar as estimativas de seus colegas a partir da colaboração com os demais pontos de vista. São dois ciclos de estimativas anônimas por cada especialista do grupo, até que os valores alcancem uma faixa aceitável. O resultado é uma estimativa aprovada pelo consenso do grupo, o que geralmente é algo melhor do que a avaliação individual de cada um (VAZQUEZ: 2004).

Tree-Point: Esta técnica é utilizada para melhorar as estimativas diretas, quando acabam ocorrendo mais valores fornecidos pelos responsáveis pela execução das estimativas.

Neste caso é utilizada uma fórmula para definir a estimativa. São utilizados os valores máximos e mínimos e os mais comuns e aplicados conforme abaixo:

$$e = Min + 4 * Comum + Max / 6$$

Com o seguinte desvio padrão:

$$\sigma = Max - Min / 6$$

Em que:

a) e – Estimativas;
b) Min – Valor mínimo encontrado entre todos os valores fornecidos pelos responsáveis;
c) Max – Valor máximo encontrado entre todos os valores fornecidos pelos responsáveis;
d) σ – Desvio padrão.

Mas apesar das maneiras apresentadas para chegar-se as estimativas de esforço mais próximas da realidade, Vazquez (2004, p. 153) diz que:

> A melhor forma de chegar à estimativa de esforço a partir da estimativa de tamanho de projeto é utilizar dados internos à própria organização, como o número de horas apropriadas em projetos passados, conjuntamente com a quantidade de pontos de função dimensionados nas diversas fases dos respectivos projetos.

Ainda assim, são poucas as organizações que possuem uma base de dados histórica de suas aplicações. Menor ainda são as empresas que mantêm informações sobre dimensionamento das diversas fases de cada projeto desenvolvido. Sem estas informações o processo de geração de estimativas torna-se mais difícil (VAZQUEZ: 2004).

8 APLICAÇÃO DA TÉCNICA APF NA DATASUL

Neste capítulo será apresentada uma experiência prática do processo de implantação da métrica de APF na empresa Datasul, na qual serão apresentados os passos realizados para que possibilitasse a implantação da métrica de APF, bem como as dificuldades encontradas durante esse processo de implantação.

O que motivou a apresentação desta métrica de software, bem como todo seu processo de implantação vem ao encontro em desmistificar e conceituar de forma prática o que AFP, e sua aplicação em algumas empresas, mas ainda muito pouco difundida a nível Brasil.

Será apresentada uma breve introdução sobre a história da Datasul e como a mesma está estruturada com sua forma inovadora de divisão interna, conhecida como modelo de franquias, adotado em 1999. A seguir será apresentado o processo de implantação da métrica de APF na Datasul, bem como os passos futuros e perspectivas de crescimento e evolução da empresa através da utilização da métrica de APF.

8.1 A EMPRESA DATASUL

O caminho percorrido pela Datasul confunde-se com a história da tecnologia da informação. Ela foi fundada no ano de 1978, por Miguel Abuhab, para assessorar empresas do setor industrial na implantação de centros de processamentos de dados, a companhia se estabeleceu como pioneira no fornecimento de softwares de automatização de sistemas empresariais no país (DATASUL: 2006).

Os softwares que são desenvolvidos pela empresa Datasul destinam-se a automatizar e gerenciar processos críticos de todos os seus clientes nas mais diversas áreas de atuação, tais como: finanças, recursos humanos, logística, manufatura, dentre outros (DATASUL: 2006).

Com isto, a Datasul iniciou o século XXI com um lugar de destaque entre os maiores fornecedores mundiais de soluções para gestão empresarial, com o Datasul EMS (Enterprise Management System) e soluções integradas de BI (Business Intelligence), CRM (Customer Relationship Management), ECM (Enterprise Content Management) e B2B (Business to Business), serviços de educação corporativa, entre outras soluções (DATASUL: 2006).

Ao longo de sua história a Datasul vem se destacando como a empresa brasileira líder no fornecimento de softwares empresariais e conta hoje com aproximadamente 2,2 mil profissionais em seu network em todo o Brasil e na América Latina.

Em 1999, a Datasul apostou em uma estratégia para manter o seu crescimento e implantou um modelo inovador de franquias para o desenvolvimento e venda de seus produtos, onde transformou centenas de colaboradores da companhia na época em novos empreendedores (DATASUL: 2006).

Este novo modelo foi composto de dois tipos de franquias, que foram divididas internamente conforme as áreas internas existentes na época, de acordo com o foco de negócio de cada uma. Estas franquias são conhecidas como:

a) Franquias de Desenvolvimento – Também conhecida como FDES, onde nessas franquias são desenvolvidos os diversos produtos, ou partes deles e as integrações a outros produtos;
b) Franquias de Distribuição – Também conhecidas como FDIS, onde nessas franquias os produtos desenvolvidos nas FDES são distribuídos (vendidos) para todo o Brasil e mercado internacional.

Para melhor entendimento, a figura a seguir representa a estrutura da empresa Datasul:

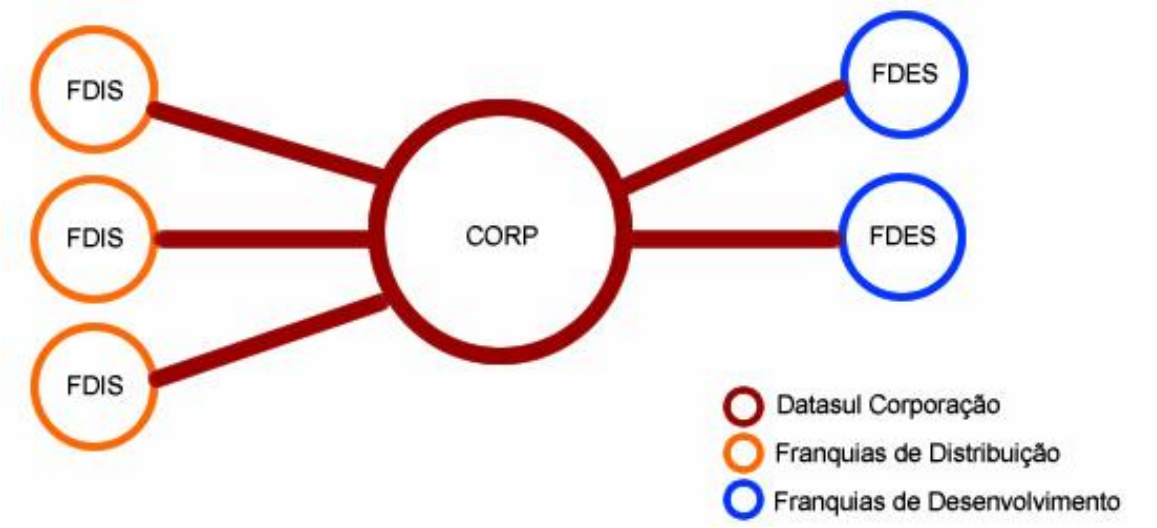


Figura 8 – Estrutura organizacional da empresa Datasul S.A.

Fonte: O Autor, 2006.

Como exemplo, a seguir algumas das franquias de desenvolvimento e distribuição existentes, além da própria matriz, conhecida como Datasul Corporação:

a) Matriz – Datasul Corporação (Datasul Corp);

b) FDES – Datasul Tecnologia;

c) FDES – Datasul Logística;

d) FDES – Datasul Finanças;

e) FDES – Datasul Manufatura;

f) FDIS – Datasul Santa Catarina;

g) FDIS – Datasul São Paulo;

h) FDIS – Datasul Morumbi;

i) FDIS – Datasul Rio de Janeiro.

Atualmente existem aproximadamente 10 franquias de desenvolvimento (FDES) e mais de 30 franquias de distribuição (FDIS) espalhadas por todo o Brasil. Este último número vem crescendo constantemente, alcançando novos lugares onde os produtos Datasul ainda não estão presentes a nível nacional e internacional.

Em 2006, a empresa decidiu abrir o capital e negociar suas ações no Novo Mercado da Bovespa. Seu objetivo foi de democratizar seus negócios e aproveitar novas oportunidades de crescimento que são oferecidas pelo seu mercado de atuação (DATASUL: 2006).

Atualmente a empresa está sediada em Joinville – SC, onde é pioneira no desenvolvimento e comercialização de soluções integradas de softwares de gestão empresarial, com 28 anos de presença no mercado (DATASUL: 2006).

8.2 O INÍCIO DO PROCESSO DE IMPLANTAÇÃO

No ano de 2004, teve início na Datasul o processo de implantação da métrica de APF. Durante este período até os dias atuais, o responsável pela divulgação e manutenção do projeto de implantação da métrica é do departamento da qualidade de software da Datasul Corporação.

O motivo para implantação da técnica de APF na Datasul, foi o de adequar os processos de desenvolvimento e manutenção de seus produtos ao modelo de qualidade CMM/CMMI[1]. Outro fator importante para investir na em uma métrica de software foi à necessidade de comparação com o mercado internacional.

A primeira medida tomada foi um avaliação sobre o que o mercado nacional e internacional estava utilizando como métrica, bem como qual dela traria melhores resultados para o tipo de produto desenvolvido pela Datasul. Dentre as técnicas avaliadas (LOC, APF e outras de menor expressão a nível de mercado), foi escolhida a técnica de APF para ser apresentada ao nível gerencial da empresa.

CMM/CMMI – O CMMI (Capability Maturity Model Integration) é uma evolução do CMM e procura estabelecer um único modelo para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas.

Com o apoio da alta direção, a técnica de APF foi adotada como sendo oficial para a geração de estimativas de projetos de desenvolvimento, bem como para a contagem das aplicações já existentes da Datasul.

Após a definição da métrica a ser utilizada, iniciou-se o processo de capacitação das equipes que iriam disseminar o conhecimento da métrica em todas as FDES. Neste momento a Datasul Corporação contratou uma empresa de consultoria em APF, para auxiliar no processo inicial da implantação da métrica de software.

Paralelo a isto, a Datasul Corporação também contratou horas mensais de um recurso da franquia Datasul Tecnologia para que iniciasse o processo de estudo da técnica de APF e também o desenvolvimento de um guia de contagem interno exclusivo para a Datasul, a fim de desmistificar as principais dúvidas encontradas no processo de contagem e definir pontos polêmicos existentes na técnica de APF.

Este recurso contratado inicialmente teria a responsabilidade de auxiliar a todas as franquias prestando suporte no processo de contagem de seus produtos.

Este processo inicial durou cerca de seis meses, onde a consultoria externa participou fornecendo suporte e efetuando as primeiras auditorias nas contagens realizadas pelas FDES, acompanhado pela franquia da Datasul Tecnologia que estaria assumindo esta tarefa posteriormente. Vale ressaltar, que estas contagens iniciais foram não foram automatizadas.

Durante todo o processo de contagem realizado na Datasul as principais dificuldades encontradas foram à falta de conhecimento necessário da métrica de APF por parte das franquias que realizavam as contagens dos aplicativos sob sua responsabilidade.

O fato de que as contagens estarem sendo realizadas por analistas e desenvolvedores, fez também com que a visão do usuário (que é a visão correta para a realização da análise do aplicativo), fosse um pouco distorcida devido o conhecimento mais aprofundado do software. Todos estes problemas encontrados foram resolvidos com orientações pontuais e

esclarecimentos sobre a métrica de APF realizados pelo responsável pelo suporte, no caso a franquia da Datasul Tecnologia, que foi a empresa contratada para a prestação de suporte a todas as franquias da Datasul Corporação.

Em dezembro de 2004, foi realizado um acordo com os gerentes de produtos da Datasul Corporação, onde cada um deveria ter um projeto estimado em horas e pontos de função. Por não existir uma base história, esse processo foi necessário para ajudar na definição de um valor por ponto de função.

A partir deste momento, demais projetos que passaram a ser desenvolvidos pelas franquias tiveram seu esforço medido em horas e também em pontos de função. Esta atividade teve duração de três meses, onde foi definido um valor de ponto de função para a remuneração das franquias, tornando-as todas semelhantes no quesito de contratação de novos projetos solicitados pela Datasul Corporação. É importante dizer que 85% das contratações são baseadas em projetos evolutivos, efetuando melhorias em aplicativos já existentes.

Para entender melhor, segue abaixo a tabela de evolução das contratações em APF pela Datasul Corporação:

Tabela 10 – Tabela de contratações realizadas em APF

Meses / Ano	Contratações realizadas em APF (%)
Abril / 2005	20
Maio / 2005	40
Junho / 2005	60
Julho / 2005	80
Agosto / 2005	100

Fonte: Datasul, 2006

Com o conhecimento do tamanho do seu produto já instalado e em funcionamento, a Datasul buscou informações históricas do tempo e do custo empregado nos projetos de desenvolvimento e também o valor inicial dos indicadores de taxa de entrega de homem hora

por ponto de função (HH/PF) e da produtividade de ponto de função por homem hora mês (PF/HHM).

De posse dessas informações, durante seis meses, estes indicadores foram avaliados em todos os novos projetos contratados através de ponto de função, onde foi possível constatar uma melhoria de aproximadamente 20% na taxa de entrega e produtividade dos projetos realizados com base em ponto de função.

Durante este processo de acompanhamento dos indicadores, foi desenvolvido internamente um sistema que armazena as contagens realizadas pelas franquias dos aplicativos e também dos projetos de desenvolvimento e melhoria do produto. Neste momento definiu-se que a franquia Datasul Tecnologia seria a responsável por fornecer suporte a todas as demais franquias sobre dúvidas de contagem, bem como estaria realizando auditorias e melhorias neste novo sistema em desenvolvimento.

A partir do ano de 2005 a Datasul assumiu como unidade padrão a APF para todas as contratações dos projetos de desenvolvimento das suas FDES e a forma de remuneração das equipes de suporte e manutenção, baseando-se no total de pontos de função de cada franquia. Estes valores de ponto de função por franquias são atualizados trimestralmente, onde neles são inseridos os novos desenvolvimentos realizados no trimestre.

Por se tratar de uma técnica subjetiva e passível de interpretação, durante todo o processo de contagem realizado pelas franquias, foram identificadas situações onde foi preciso um estudo mais aprofundado de determinadas funções para uma contagem mais adequada. Devido a isto, nos meses de abril até setembro de 2006 foram realizadas diversas auditorias nas franquias, baseadas em um calendário definido pela Datasul Corporação e também seguindo os critérios de maior módulo e módulo mais vendido da franquia.

Estas auditorias tiveram o objetivo de esclarecer as principais dúvidas encontradas pelo responsável da franquia, bem como identificar erros e vícios de contagem, que acabam distorcendo o resultado final da contagem.

Com o intuito de enriquecer este trabalho, é possível consultar parte do Guia de Contagem de APF desenvolvido pela própria Datasul nos anexos (ANEXO B).

8.3 SITUAÇÃO ATUAL DO PROJETO

No início do ano de 2006, as franquias Datasul Finanças, Datasul Logística, Datasul Manufatura, Datasul Tecnologia e Datasul HCM estavam com seus produtos contados segundo a métrica de APF. As franquias Datasul BI e Datasul CRM inciaram a contagem de seus produtos a partir do segundo trimestre do ano. Para estas duas novas franquias ainda não possuem as contagens disponíveis através do sistema que gerencia os pontos de função das franquias.

O sistema está sendo utilizado de forma que os projetos a serem desenvolvidos são criados e armazenados em um sistema chamado Sistema de Encomendas, que utiliza as funções já existentes no sistema de APF.

Durante o mês de novembro de 2006, foram realizadas duas capacitações em São Paulo sobre a APF com o objetivo efetuar melhorias no Guia de contagem da Datasul, explorando pontos em que o mesmo ainda não deixa explicito situações divergentes da técnica original, manter o conhecimento atualizado conforme o mercado e também realizar uma preparação para conquistar a certificação CFPS (*Certified Function Point Specialist*[4]).

8.4 PERSPECTIVAS FUTURAS

A Datasul tem por objetivo transformar a métrica de APF em um padrão para remuneração de todas as FDES, no que tange novos desenvolvimentos, bem como o suporte e a manutenção para os aplicativos já existentes, baseando-se no total de pontos de função de cada franquia.

Possuir um profissional certificado em APF, para que haja uma maior credibilidade e confiança nos processos de auditoria e também no suporte para as franquias realizarem corretamente suas contagens.

Continuar a manutenção evolutiva do sistema de APF, com o intuito de extrair informações históricas importantes para análise e definição de novas estratégias para manutenção de contratos e da remuneração das franquias.

Certified Function Point Specialist[1] – Pessoa que conquista a certificação de Especialista em Análise de Ponto de Função segundo o IFPUG.

Ser reconhecida em nível nacional e internacional como empresa que possui uma métrica de software, e assim conquistar novos espaços no mercado com este diferencial.

8.5 PRÁTICAS DE CONTAGEM REALIZADAS NA DATASUL

Para compreender melhor a métrica de APF será apresentando dois programas pertencentes aos produtos Datasul de forma detalhada, para visualizar como foi realizado o procedimento de contagem nas franquias. Para mais exemplos, poderá ser consultado o Anexo B deste trabalho que contém parte do Guia de contagem da Datasul S.A..

Tela de Manutenção de Módulos: Esta tela faz parte do Aplicativo de Integração e Tecnologia, que é mantido pela franquia Datasul Tecnologia. Sua função principal é realizar o cadastro das rotinas (menus principais) para todos os módulos de todas as franquias de desenvolvimento da Datasul.

Figura 9 – Manutenção de Rotinas.

Fonte: Sistema Datasul, 2006.

Para a APF, esta tela representa uma única função que é chamada neste caso de "Manutenção Rotinas" onde a mesma possuí alguns processos elementares segundo a visão da APF.

Após a identificação dos seus processos elementares, que neste caso são o de incluir, alterar, excluir e consultar, é necessário identificar o tipo de cada um, onde neste caso teríamos três entradas externas (EE) e uma consulta externa (CE). A seguir devem ser contados seus respectivos TDs e TRs para que através das tabelas padrões da APF, posso ser verificado sua complexidade e consequentemente sua contribuição em pontos de função não-ajustados.

Tabela 11 – Contagem da Função de Manutenção de Rotinas

Nr.	Função	Processo Elementar	Tipo	TD	TR	Complexidade	Total PF
1	Manutenção Rotinas	Incluir	EE	8	1	Baixa	3
2	Manutenção Rotinas	Alterar	EE	8	1	Baixa	3
3	Manutenção Rotinas	Excluir	EE	2	1	Baixa	3
4	Manutenção Rotinas	Consultar	CE	8	1	Baixa	3

Fonte: O Autor, 2006.

Com esta classificação efetuada, é possível identificar que esta função contribui com 12 pontos de função não-ajustados para o aplicativo onde esta função esta inserida. Este resultado não considera o valor do VAF, visto que para isto seria necessário avaliar a aplicação como um todo.

Utilizando um valor hipotético de VAF (sem avaliar as 14 CGS) para este exemplo teríamos o seguinte resultado para o total de pontos de função ajustados para uma contagem do tipo aplicação:

$$AFP = ADD * VAF$$

$$AFP = 12 * 1,10$$

$$AFP = 13,2$$

Considerando a aplicação do valor de VAF, é possível identificar que esta função contribui com 13,2 pontos de função ajustados para o aplicativo onde esta função esta inserida.

Tela de Baixa Bem Patrimonial: Esta tela faz parte do Aplicativo de Controladoria e Finanças, que é mantido pela franquia Datasul Finanças. Sua função principal é realizar o a

baixa dos Bens Patrimoniais. Segue uma visão da tela inteira com todos os campos visíveis para o usuário.

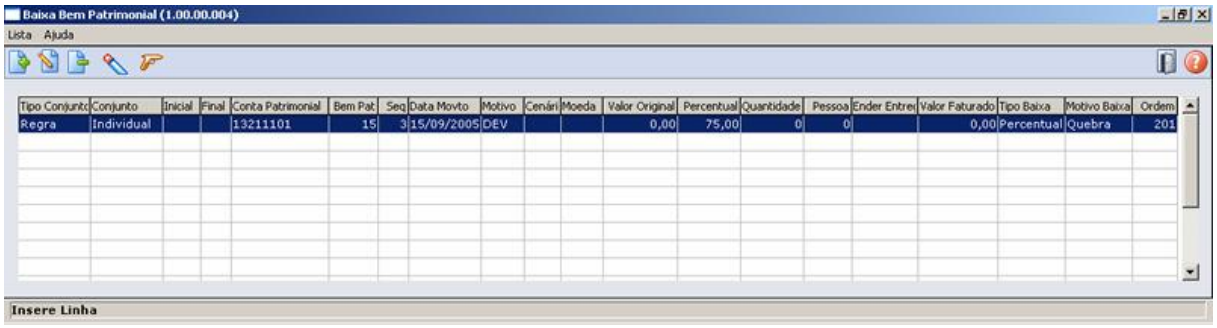


Figura 10 – Baixa de Bem Patrimonial completa.

Fonte: Sistema Datasul, 2006.

Para uma melhor visualização, segue abaixo apenas parte da tela onde serão descritos os passos para a contagem da mesma.

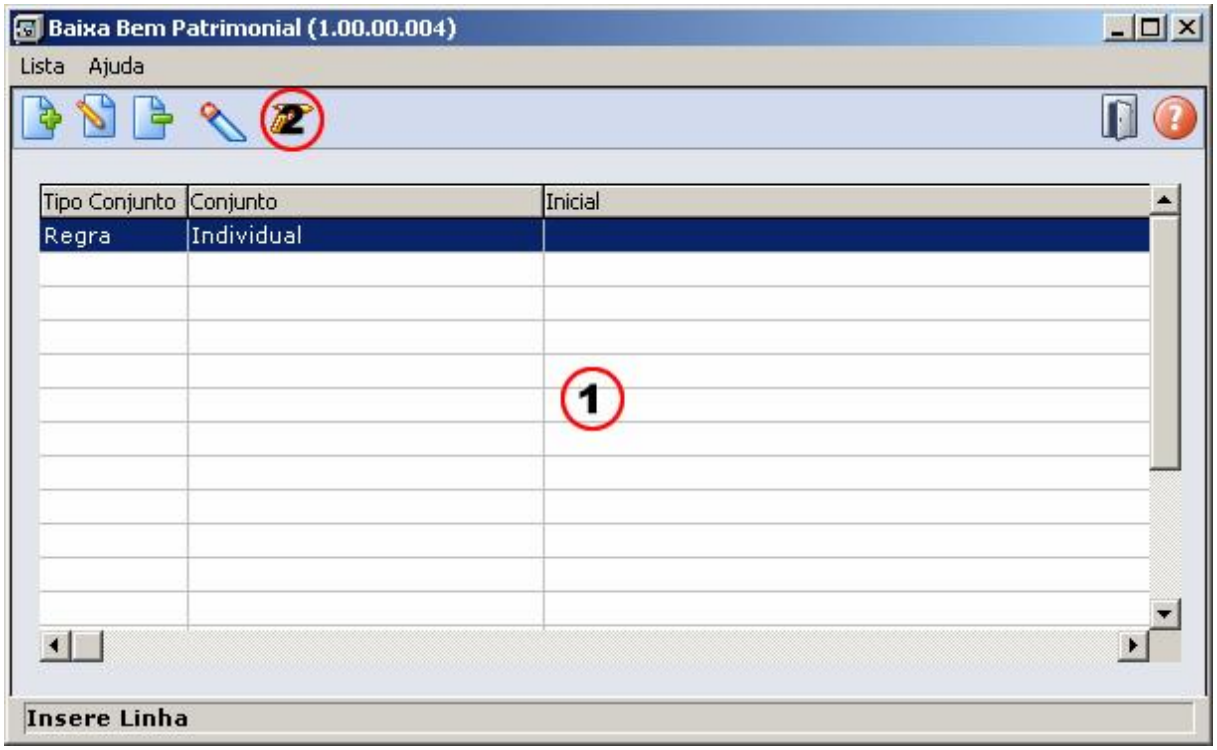


Figura 11 – Baixa de Bem Patrimonial.

Fonte: Sistema Datasul, 2006.

Para a APF, esta tela representa uma única função que é chamada neste caso de “Baixa Bem Patrimonial”. Ao contrário do outro programa apresentado anteriormente, esta tela

possuí alguns processos elementares ocultos, que fazem parte da regra de negócio envolvida pela mesma.

Os processos elementares identificados para esta tela são:

a) Tela Base Parâmetros;

b) Executa Baixa;

c) Calcula Depreciação;

d) Calcula Amortização;

e) Calcula Correção Monetária;

f) Atualiza Saldo Depreciação;

g) Atualiza Saldo Amortização;

h) Atualiza Saldo Correção Monetária;

i) Gera Apropriação Depreciação;

j) Gera Apropriação Amortização;

k) Gera Apropriação Correção Monetária;

l) Calcula Depreciação Incorporações;

m) Calcula Amortização Incorporações;

n) Calcula Correção Monetária Incorporações;

o) Atualiza Saldo Depreciação Incorporações;

p) Atualiza Saldo Amortização Incorporações;

q) Atualiza Saldo Correção Monetária Incorporações;

r) Gera Apropriação Depreciação Incorporações;

s) Gera Apropriação Amortização Incorporações;

t) Gera Apropriação Correção Monetária Incorporações.

É necessário identificar também o tipo de cada processo elementar e o número de TDs e TRs para que através das tabelas padrões da APF, posso ser verificado sua complexidade e

consequentemente sua contribuição em pontos de função não-ajustados. Segue abaixo a tabela resultante da contagem desta tela:

Tabela 13 – Contagem da Função de Baixa Bem Patrimonial

Nr.	Função	Processo Elementar	Tipo	TD	TR	Complexidade	Total PF
1	Baixa Bem Patrimonial	Tela Base Parâmetros	CE	22	1	Média	4
2	Baixa Bem Patrimonial	Executa Baixa	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Calcula Depreciação	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Calcula Amortização	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Calcula Correção Monetária	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Atualiza Saldo Depreciação	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Atualiza Saldo Amortização	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Atualiza Saldo Correção Monetária	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Gera Apropriação Depreciação	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Gera Apropriação Amortização	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Gera Apropriação Correção Monetária	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Calcula Depreciação Incorporações	EE	7	4	Alta	6
2	Baixa Bem Patrimonial	Calcula Amortização Incorporações	EE	7	4	Alta	6

Fonte: O Autor, 2006

Com esta classificação efetuada, é possível identificar que esta função contribui com 76 pontos de função não-ajustados para o aplicativo onde esta função esta inserida. Este

resultado não considera o valor do VAF, visto que para isto seria necessário avaliar a aplicação como um todo.

Utilizando um valor hipotético de VAF (sem avaliar as 14 CGS) para este exemplo teríamos o seguinte resultado para o total de pontos de função ajustados para uma contagem do tipo aplicação:

$$AFP = ADD * VAF$$

$$AFP = 76 * 0,90$$

$$AFP = 68,4$$

Considerando a aplicação do valor de VAF, é possível identificar que esta função contribui com 68,4 pontos de função ajustados para o aplicativo onde esta função esta inserida.

9 CONCLUSÃO

Uma das causas para a não utilização de uma métrica de software em uma organização é o seu tempo de implementação. A sua implantação não deve ter por objetivo resolver problemas imediatos, mas sim tornar-se um instrumento chave para um gerenciamento cada vez mais eficaz dentro das organizações. O seu sucesso no processo de implantação esta diretamente ligada ao envolvimento e comprometimento dos responsáveis pela sua disseminação dentro das empresas.

A métrica de APF possui adeptos pelo mundo inteiro, onde os responsáveis pela sua evolução são os próprios usuários. É um padrão reconhecido pela ISO/IEC 20926 para determinação de tamanho funcional sem a aplicação das CGS e consequentemente o VAF. Além disto, a métrica de APF possui um processo de certificação que estimula uma uniformidade de conhecimento e métodos de aplicação da em qualquer parte do mundo.

Pode-se perceber que apesar de a métrica APF possuir diversos conceitos e regras, a mesma é de fácil aprendizagem e requer certo conhecimento sobre a aplicação a ser contada. Um de seus principais benefícios é o de simplificar e padronizar a comunicação em diversos níveis da hierarquia de uma organização.

Com o estudo de caso apresentado, foi possível verificar que a métrica de APF tem sido amplamente utilizada pela empresa Datasul S.A. para a realização de estimativas e como ferramenta de auxilio para o gerenciamento de projetos de software.

Para a Datasul S.A. a métrica contribui para minimizar as diferenças existentes no processo de contratação das franquias de desenvolvimento, tornando a comunicação e negociação uniforme e padronizada entre os envolvidos no processo.

Por tratar-se de um processo de implantação desconhecido para a Datasul S.A., alguns cronogramas internos para a implementação da métrica foram atrasados devido o volume elevado de informações envolvido em todas as franquias, mas não afetando de forma a comprometer o processo de implantação da métrica de software na empresa.

Na seqüência da implantação da APF, os contratos de software passaram a ser estimados e realizados baseando-se em pontos de função, o que ajudou bastante a minimizar os ruídos gerados entre contratados e contratantes.

Após a implantação dos contratos baseados em pontos de função, foi realizado um estudo pela própria Datasul S.A., onde se comprovou que houve uma redução nos custos com contratações em torno de 20%. Este estudo foi realizado nos seis primeiros meses após o início das contratações baseadas em pontos de função.

Outro benefício conquistado foi a construção de um *baseline* com o total de pontos de função de cada produto da Datasul S.A. separado por franquia, podendo o mesmo ser utilizado por todos os gerentes de projetos, analistas e desenvolvedores que realizam contagens e estimativas. Este *baseline* também pode ser utilizado para a obtenção de outras informações relacionando os resultados da APF com outros indicadores como tempo, número de recursos, etc.

A métrica de APF possui grande potencial para torna-se uma referência para medição de software no mercado mundial. Quem a utiliza pode comprovar os resultados obtidos com a sua utilização, assim como os resultados positivos do estudo de caso apresentado neste trabalho.

REFERENCIAS

_____. **NBR ISO/IEC 9126-1: engenharia de software – qualidade de produto. Parte 1: modelo de qualidade**. Rio de Janeiro, jun. 2003.

BFPUG. **Brazilian Function Point Users Group**. Disponível em: <http://www.bfpug.com.br > Acesso em 28 out. 2006.

CAMPOS, F. M. **Métricas de software como ferramenta de apoio ao gerenciamento de projetos**. Disponível em: <http://www.apinfo.com/artigo64.htm> Acesso em: 9 set. 2006.

DATASUL S/A. Disponível em: <http://www.datasul.com.br> Acesso em: 28 out. 2006.

DEMARCO, Tom. **Controle de Projetos de Software**. Rio de Janeiro: Campus, 1989.

FUGGETTA, P.A. **Software process: a roadmap. In: The Future of software Engineering**, v 17, 6 ed., 1991.

HAZAN, C. **Análise de pontos de Função**. Disponível em: < http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/APF.pdf> Acesso em: 30 set. 2006.

HAZAN, C. **Metodologia para o Uso de Indicadores na Gerência de Projetos de Desenvolvimento de Software**. Tese de Mestrado, IME, Maio 1999.

IFPUG. **International Function Point Users Group**. Disponível em: <http://www.ifpug.com > Acesso em: 28 out. 2006.

ISBSG B. **International Software Benchmarking Standards Group**. Disponível em: < http://www.isbsg.org.au> Acesso em: 28 out. 2006.

MCT. **Ministério da Ciência e Tecnologia**. Disponível em: < http://www.mct.gov.br/index.php/content/view/3253.html> Acesso em: 22 jul. 2006.

MEDEIROS, M. P. **Extreme Programming – Conceitos e Práticas**. Disponível em: < http://manoelpimentel.blogspot.com/2006/04/extreme-programming-conceitos-e.html> Acesso em: 26 ago. 2006.

MEDEIROS, M. P. **Implementando Pair Programming em sua equipe**. Disponível em: < http://www.devmedia.com.br/articles/visualizacomponente2.asp?comp=1694> Acesso em: 26 ago. 2006.

MEDEIROS, V. **Paradigmas da Engenharia de Software**. Disponível em: < http://www.cin.ufpe.br/~vwcm/es/ > Acesso em: 5 ago. 2006.

MELLER, M. C. **Modelos para estimar custos de software: Estudo comparativo com softwares de pequeno porte**. Florianópolis, 2002.

NESMA. **Nederlandse Software Metrieken Associatie**. Disponível em: < http://www.nesma.nl/> Acesso em: 4 nov. 2006.

PADUELLI, M. M ; SANCHES, R. **Problemas em manutenção de software: caracterização e evolução**. São Paulo, 2006.

Practices Manual. Function Point Couting. Disponível em: <http://www.ifpug.org/> Acesso em: 14 out. 2006.

PRADO, D. S. **Planejamento e Controle de Projetos (série Gerência de Projetos – volume 2)**. 5 ed. Nova Lima: INDG Tecnologia e Serviços Ltda., 2004.

PRESSMAN, R. S. **Engenharia de Software.** 3 ed. São Paulo: Pearson – Makron Books, 1995.

ROCHA, A. R. C. (Org.) ; WEBER, K. C. (Org.) ; MALDONADO, J. C. (Org.) . **Qualidade de Software: Teoria e Prática**. 1. ed. São Paulo: Prentice Hall, 2001.

SCHROEDER, R. M. **Aplicação da técnica Fuction Point Analysis para mensuração de software e melhoria da qualidade**. Florianópolis, 2003.

SILVA, J. C. A. **Tópicos em Engenharia de Software**. Disponível em: < http://www.dc.ufscar.br/~junia/index-topicos.htm> Acesso em: 26 ago. 2006.

SIMÕES, C. A. **Sistemática de Métricas, Qualidade e Produtividade**. Disponível em: < http://www.bfpug.com.br/Artigos/sistematica_metricas_simoes.htm> Acesso em: 30 set. 2006.

SOMMERVILLE, I. **Engenharia de Software**. 6 ed. São Paulo: Pearson – Addison Wesley, 2003.

VAZQUEZ, C. E. ; SIMÕES, G. S. ; ALBERT, R. M. **Análise de Pontos de Função**. 2 ed. São Paulo: Érica, 2004.

XiSPÊ. **Referência Nacional em Extreme Programming**. Disponível em: < http://www.xispe.com.br/> Acesso em: 26 ago. 2006

WIKIPÉDIA. **Wikipédia, a enciclopédia livre**. Disponível em: < http://pt.wikipedia.org/wiki/Hardware> Acesso em: 4 nov. 2006

ANEXOS

ANEXO A

MODALIDADES DE CONTRATAÇÃO

Boa parte dos profissionais envolvidos em processos de contratação de serviços ou que estão sendo contratados para a prestação de serviços, já tiveram a experiência de que o relacionamento entre as partes envolvidas foi desgastado devido o projeto em questão na negociação ter excedido seus custos iniciais, escopo ou prazo (VAZQUEZ: 2004).

Existem atualmente diversas formas de contratação utilizadas para um desenvolvimento de software. Abaixo serão apresentadas quatro delas:

Modalidade de Contratação Homem/Hora: É um modelo que apresenta uma grande simplicidade e flexibilidade tanto para empresa contratada como para a contratante. Uma vez que a relação comercial for estabelecida, não há necessidade de novas negociações caso ocorram mudanças no que foi definido inicialmente, seguindo a regra que todo aumento de escopo pode acarretar em um aumento de horas e consequentemente um aumento de custo. Vale ressaltar que esta flexibilidade tem seu preço (VAZQUEZ: 2004).

Segue algumas vantagens e desvantagens deste modelo de contratação:

Quadro – Vantagens e desvantagens da contratação de horas para o desenvolvimento de software

Vantagens	Desvantagens
Visibilidade sobre o trabalho: Geralmente o desenvolvimento do projeto é feito internamente, facilitando a supervisão. Contudo, esse trabalho acaba por deslocar profissionais internos à organização de atividades fim de seu negócio.	**Controle sobre a produtividade:** O contratante é responsável pelo acompanhamento da produtividade do serviço.
Simplicidade: Pouca tramitação burocrática e esforço na negociação da alocação de recursos no caso de mudanças de requisitos ou outros aspectos conjunturais.	**Problemas trabalhistas:** A questão de haver ou não a caracterização do vínculo empregatício pode ser levantada em eventuais litígios.
	Ausência de Garantia: Dificuldade de identificar a responsabilidade e, como conseqüência, exigir a garantia do serviço quando ele é realizado por equipes mistas de várias empresas.

Fonte: Carlos Eduardo Vazquez, 2004, p. 170.

Modalidade de Contratação Preço Global Fixo (ou Projeto Fechado): Nesta modalidade, sua principal característica é existir um inicio e um fim bem definidos. Quanto melhor forem definidos os requisitos, menos atritos existirão entre contrata e contratante. Para projetos que tenham longa duração é necessário que existam pontos de controle no decorrer do projeto para que a empresa contratante possa acompanhar o andamento do projeto como um todo (VAZQUEZ: 2004).

Para Vazquez (2004, p. 171), esta modalidade pode ter um agravante, conforme ele comenta:

> Um fator que complica a utilização desta abordagem é assumir que os requisitos não mudam após o início do projeto. Assim como o mundo é dinâmico, os requisitos também o são. Quanto maior a duração do projeto, mais provável que haja alteração nos requisitos. E é difícil estimar quanto essas alterações afetam o orçamento proposto originalmente pelo fornecedor. Porém, é quase certo que seja necessária uma nova negociação. Se este for o caso, dificilmente o cliente irá obter as mesmas condições originais, pois dependendo do momento em que o projeto esteja, não haverá concorrência e tampouco uma unidade para comparar o preço originalmente cobrado com o preço cobrado pelas novas características solicitadas.

Segue algumas vantagens e desvantagens deste modelo de contratação:

Quadro – Vantagens e desvantagens da contratação do desenvolvimento de software a preço global fixo

Vantagens	Desvantagens
Transferência de riscos: Variações na produtividade e no escopo durante a execução do projeto são de responsabilidade do fornecedor.	**Preço contingenciado:** Para absorver os riscos de requisitos mal definidos ou de um crescimento de escopo, o fornecedor acrescenta margens de segurança à proposta comercial.
Previsibilidade de custos: Se os requisitos foram definidos de forma adequada e as mudanças forem pequenas, o orçamento será cumprido.	**Desgaste com o fornecedor:** Se os requisitos forem nebulosos ou o fornecedor subestimar o custo, haverá um desgaste em renegociações contratuais.
Garantia: Após a entrega do produto, a responsabilidade de garantia sobre eventuais problemas de conformidade à especificação é bem definida e pode ser cobrada, inclusive, se necessário, com o amparo legal. Tal fato faz com que seja interesse mútuo a qualidade do produto.	**Dependência:** Se não houver a transferência adequada de conhecimento do projeto, o cliente pode ficar "refém" do fornecedor.

Fonte: Carlos Eduardo Vazquez, 2004, p. 172

Modalidade de Contratação Preço Unitário: Para este tipo de modalidade, é negociada entre as partes uma remuneração por elementos do projeto. Esses elementos poderão ser

tabelas, telas, relatórios, linhas de código ou ponto de função. Este modelo procura equilibrar as deficiências existentes entre os dois modelos anteriores apresentados (VAZQUEZ: 2004).

Para esta modalidade a utilização de pontos de função é bastante adequada assim como comenta Vazquez (2004, p. 173):

> A análise de pontos de função é um método padrão de medição funciona. Centenas de empresas e profissionais de todo o mundo participam das ações do IFPUG, que é a organização responsável pela manutenção e evolução da técnica. Essas ações procuram garantir a consistência e a uniformidade da aplicação do método. Experiências de governos e empresas de todo o mundo relatam casos de sucesso em sua aplicação.

Mas como se define o preço de um ponto de função? Esta resposta não é simples e precisa de algumas informações para ser respondida. Muito do conhecimento adquirido com os anos de experiência dos profissionais envolvidos, mas isto não da à garantia de assertividade do valor, como comenta Vazquez (2004, p. 178):

> Quando o cliente pergunta para um fornecedor o seu preço por ponto de função, antes de qualquer resposta, o fornecedor deve estar ciente do contexto em que será realizado o serviço. Uma analogia muito comum de ponto de função é com o metro quadrado da construção civil. Ao se perguntar o preço do metro quadrado a um corretor de imóveis, certamente ele não fornecerá um, mas vários preços: de acordo com a região, tipo de acabamento, infra-estrutura adicional do imóvel, etc.

Em resumo, não existe um preço único para um ponto de função. Devem ser avaliadas um conjunto de variáveis pertinentes ao projeto como o cenário, tamanho, nível de detalhamento do projeto, entre outras.

Segue algumas vantagens e desvantagens deste modelo de contratação:

Quadro – Vantagens e desvantagens da contratação do desenvolvimento de software a preço unitário – Pontos de Função

Vantagens	Desvantagens
Melhor distribuição de responsabilidades: Atribuição da responsabilidade pelas variações da produtividade e escopo àqueles que efetivamente as causam. Paga-se pelo que se recebe.	**Necessidade de pessoal qualificado na técnica de APF:** Nessa abordagem há a necessidade de pessoal qualificado na técnica (preferencialmente especialistas certificados) tanto no cliente quanto no fornecedor.
As vantagens apresentadas para os projetos a preço fixo aplicam-se também aos contratados por pontos de função.	**Tamanho mínimo de projeto:** Para projetos pequenos, por exemplo 50 pontos de função, há uma distorção entre o tamanho funcional e o esforço envolvido.

Fonte: Carlos Eduardo Vazquez, 2004, p. 174.

ANEXO B

GUIA DE CONTAGEM DA DATASUL

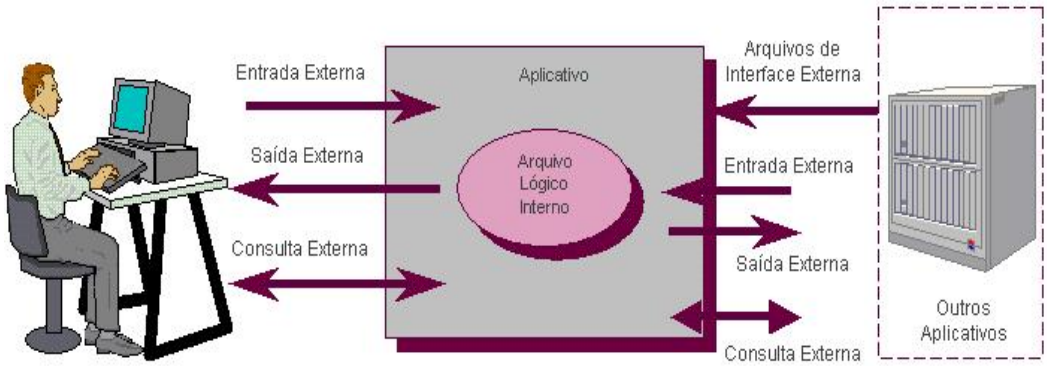
GUIAFPA.01 – Análise de Pontos de Função

AUTOR: CORP1066 – ANA CRISTINA GOZDZIEJEWSKI DATA DE ELABORAÇÃO: 29/06/06

1. OBJETIVO

1.1. ANÁLISE DE PONTOS DE FUNÇÃO

A Análise de Pontos de Função é uma visão do sistema na perspectiva do usuário[1], ou seja, é uma visão em termos de funcionalidades fornecidas aos usuários.



1.2. FRONTEIRA DA APLICAÇÃO

A fronteira de contagem é responsável por delimitar a aplicação que está sendo contada e o mundo externo, separando o que é interno e mantido pela aplicação daquilo que é externo. E é importante ressaltar que o mundo externo é todo aquele que se comunica com a aplicação, seja um usuário final ou uma outra aplicação.

No processo de contagem de pontos de função dos produtos Datasul a fronteira está pré-estabelecida como sendo o aplicativo (Controladoria e Finanças, Integração e Tecnologia, Recursos Humanos etc.).

[1] Usuário pode ser visto como uma pessoa, responsável por especificar os requisitos funcionais, ou qualquer pessoa ou coisa que tenha interação com a aplicação, podendo então ser um outro sistema que se integre com a aplicação.

2. ARQUIVOS LÓGICOS INTERNOS (ALI)

Um arquivo lógico interno é um grupo lógico de dados relacionados, ou informação de controle, identificado e reconhecido pelo usuário e que é mantido dentro da fronteira do aplicativo. A intenção primária de um ALI é manter os dados que sofrem manutenção através de um ou mais processos elementares do aplicativo que está sendo contado.

Geralmente, um ALI equivale a um depósito de dados em um diagrama de fluxo de dados, ou a uma entidade em um modelo entidade-relacionamento. Mas também podem ser arquivos em disco, caso sejam mantidos pelo aplicativo.

2.1. COMO IDENTIFICAR ARQUIVOS LÓGICOS INTERNOS

Para auxiliar na identificação de ALIs utilize os seguintes recursos:

- Para um desenvolvimento novo que tenha um modelo entidade-relacionamento definido, utilize o mesmo para identificar os ALIs;
- Para cada ALI do modelo entidade-relacionamento verifique a quantidade de atributos para determinar a complexidade (baixa, média, alta);
- Para entidades fracas ou sub-entidades (entidades especializadas), geralmente estas não serão consideradas ALIs independentes. Na realidade, estas entidades serão um segundo tipo de registro de um outro ALI (entidade fundamental).
  Mas caso a entidade fraca, na visão do usuário, seja vista como um grupo lógico de dados independente, esta entidade fraca deve ser considerada um ALI independente;
- Para entidades que possuem campos do tipo *RAW*, deve-se considerar estes campos como novos tipos de registros caso representem efetivamente um registro;
- Para contagens onde o modelo entidade-relacionamento não exista, pode-se fazer a contagem baseada em uma listagem que contenha as tabelas e seus atributos.

A identificação de ALIs nos produtos Datasul, e conseqüentemente a contagem de pontos, pode ser automatizada através da utilização de um banco de dados de referência cruzada (*XREF*). Mas, primeiramente, devem-se estabelecer claramente quais são as tabelas pertencentes à fronteira de contagem.

Caso o aplicativo em contagem faça uso de APIs de outros aplicativos, não se devem considerar as tabelas referenciadas como ALIs ou AIEs, pois estas APIs são funcionalidades fornecidas pelo outro aplicativo (por uma outra fronteira de contagem).

2.2. COMO DETERMINAR A COMPLEXIDADE DE ARQUIVOS LÓGICOS INTERNOS

Com a identificação dos ALIs e seus campos concluídos, basta utilizar a tabela abaixo para determinar as suas complexidades:

	Número de Campos		
Número de Registros	< 20	20 – 50	> 50
1	Baixa	Baixa	**Média**
2-5	Baixa	**Média**	Alta
> 5	**Média**	Alta	Alta

2.3. COMO DETERMINAR A CONTRIBUIÇÃO DOS ARQUIVOS LÓGICOS INTERNOS

Com a complexidade determina, o cálculo da contribuição dos ALIs em número de pontos de função é feito baseado na tabela abaixo:

Complexidade		
Baixa	Média	Alta
x 7	x 10	x 15

3. ARQUIVOS DE INTERFACE EXTERNA (AIE)

Um arquivo de interface externa é também um grupo lógico de dados relacionados, ou informação de controle, identificado e reconhecido pelo usuário, mas que é mantida fora da fronteira do aplicativo, porém é referenciado (consultado) dentro da fronteira do aplicativo. A intenção primária de um AIE é manter dados referenciados através de um ou mais processos elementares do aplicativo que está sendo contado, significando que um AIE contado por um aplicativo deve ser obrigatoriamente um ALI em um outro aplicativo.

3.1. COMO IDENTIFICAR ARQUIVOS DE INTERFACE EXTERNA

Todos os recursos utilizados para a identificação de ALIs podem ser utilizados para identificar-se AIEs, desde que se mantenha claras as diferenças entre os arquivos.

3.2. COMO DETERMINAR A COMPLEXIDADE DE ARQUIVOS DE INTERFACE EXTERNA

Com a identificação dos AIE e seus campos concluídos, basta utilizar a tabela abaixo para determinar as suas complexidades:

	Número de Campos		
Número de Registros	< 20	20 – 50	> 50
1	Baixa	Baixa	**Média**
2-5	Baixa	**Média**	Alta
> 5	**Média**	Alta	Alta

3.3. COMO DETERMINAR A CONTRIBUIÇÃO DOS ARQUIVOS DE INTERFACE EXTERNA

Com a complexidade determina, o cálculo da contribuição dos AIEs em número de pontos de função é feito baseado na tabela abaixo:

Complexidade		
Baixa	Média	Alta
x 5	x 7	x 10

4. ENTRADAS EXTERNAS (EE)

Uma entrada externa é um processo elementar que processam dados ou informação de controle proveniente de fora da fronteira do aplicativo. A intenção primária de uma EE é manter um ou mais ALIs ou alterar o comportamento do aplicativo.

Em decorrência disso, um mesmo programa pode apresentar várias EEs. E uma EEs não precisa, obrigatoriamente, ter uma interface gráfica com o usuário ela também pode ser um processo elementar utilizado por outro aplicativo.

4.1. COMO IDENTIFICAR ENTRADAS EXTERNAS

Para auxiliar na identificação de EEs utilize os seguintes recursos:

- Para cada ALI geralmente existirá uma interface gráfica para inclusão, outra para modificação e outra para exclusão;
- Para os produtos Datasul uma parcela significativa dos programas segue estilos pré-definidos (*templates*), permitindo uma rápida identificação das EEs baseando-se apenas no estilo adotado pelo programa;
- Para programa sem interface gráfica também podem ser identificadas EEs, isto geralmente acontece para APIs e DBOs.

9.1.1 *Observações*

Durante o processo de identificação de EEs não se deve considerar a função de execução em batch como uma nova EE em programas baseados nos estilos cálculo, exportação, importação ou relatório, bem como, APIs e DBOs.

Também não devem ser considerados os ALIs ou AIEs referentes a menu, parâmetros globais, segurança e usuário como arquivos referenciados, pois são inerentes aos produtos Datasul. E caso sejam contados, podem superestimar a complexidade das funções do tipo transação rapidamente. A contagem de tais arquivos deve ser restrita ao aplicativo integração e tecnologia.

As APIs e BOs (liberadas) que são utilizadas nos programas de cadastro não devem ser contados os arquivos referenciados (AR) e os tipos de dados (TD) da API ou BO, pois neste caso como a API ou BO já está sendo contada em separado. Deve-se contar apenas uma vez a API ou BO.

Para APIs Visuais deve-se utilizar o mesmo conceito de uma API convencional conta-se apenas uma vez como uma funcionalidade em separada.

4.2. COMO DETERMINAR A COMPLEXIDADE DE ENTRADAS EXTERNAS

Com a identificação das EEs concluída, basta contar o número de campos e controles existentes, e o número de arquivos (ALIs ou AIEs) referenciados, para determinar a complexidade com base na tabela abaixo:

	Número de Campos e Controles		
Número de Arquivos	**< 5**	**5 – 15**	**> 15**
< 2	Baixa	Baixa	**Média**
2	Baixa	**Média**	Alta
> 2	**Média**	Alta	Alta

4.3. COMO DETERMINAR A CONTRIBUIÇÃO DAS ENTRADAS EXTERNAS

Com a complexidade determina, o cálculo da contribuição das EEs em número de pontos de função é feito baseado na tabela abaixo:

Complexidade		
Baixa	Média	Alta
x 3	x 4	x 6

4.4. APLICAÇÃO DE ENTRADAS EXTERNAS

Apresentamos a seguir exemplos de contagem de processos elementares do tipo Entradas Externas (EE) sobre alguns estilos de programas.

Estilos Cadastro Simples e Cadastro Complexo

Um programa baseado em um estilo cadastro simples ou cadastro complexo geralmente possui três EEs: interface de inclusão, de modificação e de exclusão. Caso existam funções adicionais (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis EEs.

A interface de cópia normalmente não deve ser considerada como uma nova EE, pois faz a mesma função de uma inclusão com valores inicial pré-definidos. Mas caso esta realize uma função adicional identificável e reconhecida pelo usuário, então deverá ser considerada uma EE.

Após a identificação das EEs existentes em um determinado programa, deve-se contar o número de campos e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados.

Embora o estilo complexo apresente várias páginas para digitação e apresentação de campos, isto não deve ser visto como novas funções ou EEs. A apresentação em diferentes páginas geralmente é apenas um recurso tecnológico para contornar a deficiência relacionada ao limite da área de tela.

Contagem do Número de Campos e Controles

A contagem do número de campos é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica. Deve-se apenas levar em consideração que campos repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda devem ser contados como dois campos; ex.2.: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*).

Também se deve contar um campo adicional referente às mensagens de erro e sucesso apresentadas ao usuário, ou seja, um único campo contabilizará todas as mensagens apresentadas ao usuário.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex: botões de inclusão, cópia, desfazer digitação, cancelar edição e gravar edição, são contados como apenas um controle*).

Contagem do Número de Arquivos Referenciados (AR)

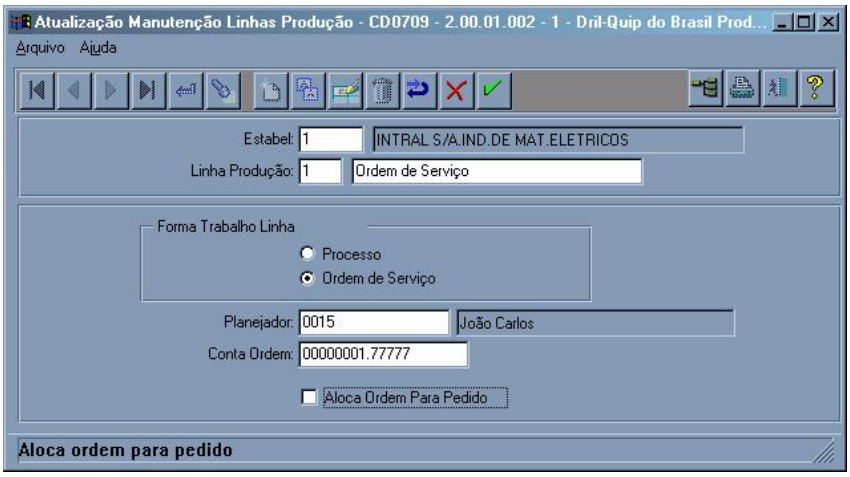
A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs mantidos pela EE após, deve-se então verificar outros ALIs ou AIEs

utilizados (consultados) mas não mantidos pela função (*ex.: interface de inclusão de bancos mantém o ALI banco e referencia o ALI de instrução bancária*).

Geralmente a contagem de ALIs ou AIEs utilizados (consultados) mas não mantidos pela EE se deve a existência de chaves estrangeiras nas interfaces gráficas.

Estudo de Caso: Manutenção de Linhas de Produção

A imagem abaixo apresenta um programa, baseado no estilo cadastro simples, pertencente ao aplicativo de manufatura e responsável pela manutenção de linhas de produção. Neste programa podem-se identificar facilmente três EEs: inclusão, modificação e exclusão.



Analisando a função de inclusão pode-se contar um total de dez campos e um controle, conforme descrito abaixo:

- Campo código do estabelecimento;
- Campo descrição do estabelecimento;
- Campo código da linha de produção;
- Campo descrição da linha de produção;
- Campo forma trabalho linha;
- Campo código do planejador;
- Campo nome do planejador;
- Campo conta-contábil da linha de produção;
- Campo aloca ordem para pedido;
- Campo para representar as mensagens de erro e sucesso;
- Controle para representar os botões de inclusão, cópia, desfazer, cancelar e gravar edição.

Já os arquivos referenciados são cinco, conforme descrito abaixo:

- Tabela linha de produção;
- Tabela estabelecimento;
- Tabela de planejador;
- Tabela de conta-contábil;
- Tabela de ordem de produção.

Entre os arquivos referenciados não há necessidade de classificá-los como ALI ou AIE, pois esta diferença não influencia na determinação da complexidade e

contribuição da EE, porém é obrigatório que a EE mantenha ao menos um ALI ou altere o comportamento da aplicação.

A tabela de ordem de produção, embora não tenha nenhuma informação visível ao usuário na interface gráfica, é utilizada internamente para realizar alguns processos de validação durante a inclusão de uma nova linha de produção, devendo assim ser contabilizada como um arquivo referenciado.

As demais EEs existentes neste programas devem ser analisadas de forma análoga à função de inclusão. E, provavelmente, devem utilizar o mesmo número de campos e controles, e referenciar os mesmos arquivos, conseqüentemente, tendo a mesma complexidade e contribuição em pontos de função.

Estilo Cadastro Pai X Filho

Um programa baseado em um estilo cadastro pai x filho geralmente possui três EEs para o pai: interface de inclusão, de modificação e de exclusão; e três EEs para cada filho: interface de inclusão, de modificação e de exclusão. Caso existam funções adicionais (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis EEs.

Em um programa deste estilo a maioria das funções é executada em telas separadas e disparada diretamente pelo usuário, mas esse comportamento não implica obrigatoriamente que sejam realmente novas EEs. Caso as interfaces de filho dependam de uma ação a ser executada na interface de pai, estas não devem ser tratadas como novas EEs.

Após a identificação das EEs existentes em um determinado programa, deve-se contar o número de campos e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados. Da mesma forma que é feita para um programa baseado nos estilos cadastro simples e cadastro complexo.

Caso o programa apresente várias páginas, representando os diferentes filhos, provavelmente cada interface existente para os filhos será contada como uma nova EE, desde que tais funções sejam realmente processos elementares reconhecidos pelo usuário.

Contagem do Número de Campos e Controles

A contagem do número de campos é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica. Deve-se apenas levar em consideração que campos repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda, devem ser contados como dois campos*).

Também se deve contar um campo adicional referente às mensagens de erro e sucesso apresentadas ao usuário, ou seja, um único campo contabilizará todas as mensagens apresentadas ao usuário.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex.: botões de inclusão, cópia, desfazer digitação, cancelar edição e gravar edição, são contados como apenas um controle*; *ex.2.: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*)

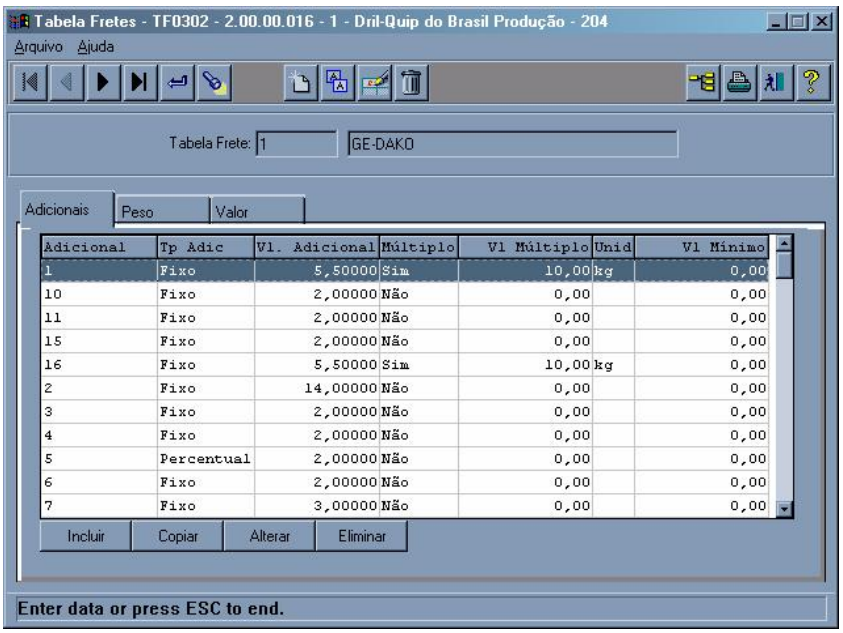
CONTAGEM DO NÚMERO DE ARQUIVOS REFERENCIADOS

A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs mantidos pela EE após, deve-se então verificar outros ALIs ou AIEs utilizados (consultados) mas não mantidos pela função (*ex.: interface de inclusão de bancos mantém o ALI banco e referencia o ALI de instrução bancária*).

Geralmente a contagem de ALIs ou AIEs utilizados (consultados) mas não mantidos pela EE se deve a existência de chaves estrangeiras nas interfaces gráficas.

Estudo de Caso: Manutenção de Tabelas de Frete

A imagem abaixo apresenta um programa, baseado no estilo cadastro pai x filho, pertencente ao aplicativo de distribuição e responsável pela manutenção de tabelas de frete. Neste programa podem-se identificar facilmente três EEs para o pai: inclusão, modificação e exclusão; e nove EEs para os filhos: inclusão, modificação e exclusão.



Analisando a função de exclusão de pai pode-se contar apenas um campo e um controle, conforme descrito abaixo:

- Campo para representar as mensagens de erro e sucesso;
- Controle para representar o botão de exclusão.

Já os arquivos referenciados são quatro, conforme descrito abaixo:

- Tabela tabela de frete;
- Tabela adicionais de tabela de frete;
- Tabela de pesos de tabela de frete;
- Tabela de valores de tabela de frete.

Mas fazendo-se uma análise mais detalhada pode-se avaliar que as tabelas que armazenam os dados dos filhos não representam novos arquivos na visão do usuário. Para o usuário existe um único arquivo, que armazena os dados da tabela de frete como um todo, mas por questões de normalização de modelos entidade-relacionamento este arquivo foi dividido em quatro.

Desta forma, deve-se contar apenas um ALI com quatro tipos de registro, sendo muito provável que este ALI tenha complexidade média ou alta. Mas esta junção somente deve ser feita se na visão do usuário o conjunto de tabelas realmente represente uma informação única.

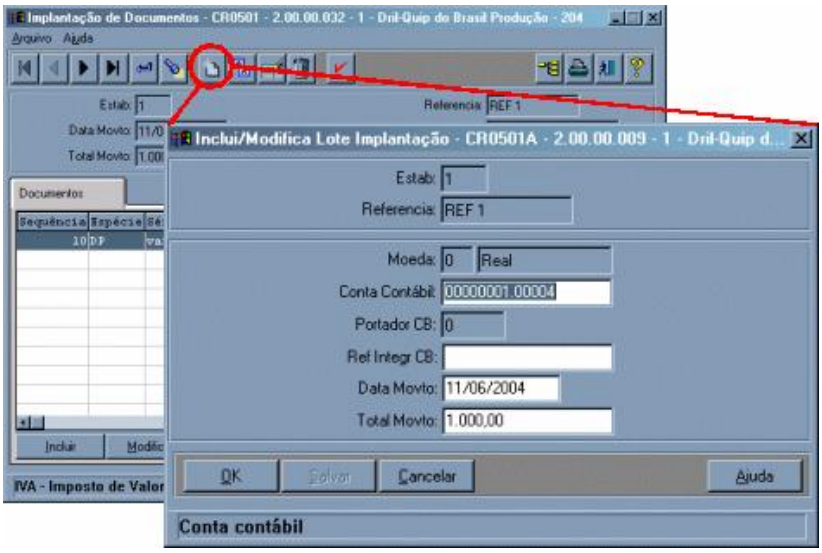
Entre os arquivos referenciados não há necessidade de classificá-los como ALI ou AIE, pois esta diferença não influencia na determinação da complexidade e contribuição da EE, porém é obrigatório que a EE mantenha ao menos um ALI ou altere o comportamento da aplicação.

Existem situações em que os cadastros (funções) são agrupados para existir facilidade para o cliente, mas os mesmos pertencem a franquias diferentes como de Logística e Finanças. Nestes cadastros em que algumas abas são de Logística e outras são de Finanças, por exemplo, devem ser contadas para cada franquia apenas as abas referentes a ela.

As demais EEs existentes neste programas devem ser analisadas de forma análoga às funções a serem analisadas no próximo tópico: **Estilo Cadastro Pai e Cadastro Filho.**

Estilos Cadastro Pai e Cadastro Filho

Um programa baseado em um estilo cadastro pai ou cadastro filho geralmente possui duas EEs: interface de inclusão e de modificação. Embora ao analisar a tela do programa isto não fique claro, pois as EEs de inclusão ou de modificação são disparadas pelo estilo cadastro pai x filho (ver figura abaixo).



Caso existam funções adicionais (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis EEs.

Após a identificação das EEs existentes em um determinado programa, deve-se contar o número de campos e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados.

Nestes dois estilos, os programas podem ou não apresentar diversas páginas para digitação e apresentação de dados, mas isto não deve ser visto como novas funções ou

EEs. A apresentação em diferentes páginas geralmente é apenas um recurso tecnológico para contornar a deficiência relacionada ao limite da área de tela.

Contagem do Número de Campos e Controles

A contagem do número de campos é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica. Deve-se apenas levar em consideração que campos repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda devem ser contados como dois campos; ex.2.: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*).

Também se deve contar um campo adicional referente às mensagens de erro e sucesso apresentadas ao usuário, ou seja, um único campo contabilizará todas as mensagens apresentadas ao usuário.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex: botões de inclusão, cópia, desfazer digitação, cancelar edição e gravar edição, são contados como apenas um controle*).

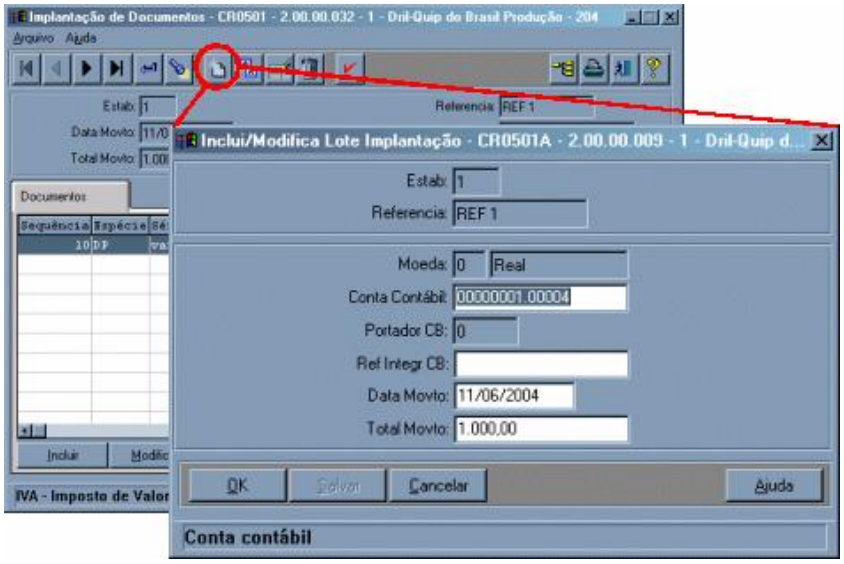
Contagem do Número de Arquivos Referenciados

A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs mantidos pela EE após, deve-se então verificar outros ALIs ou AIEs utilizados (consultados) mas não mantidos pela função (*ex.: interface de inclusão de bancos mantém o ALI banco e referencia o ALI de instrução bancária*).

Geralmente, a contagem de ALIs ou AIEs utilizados (consultados) mas não mantidos pela EE se deve a existência de chaves estrangeiras nas interfaces gráficas.

Estudo de Caso: Implantação de Títulos – Lotes de Registro de Títulos

A imagem abaixo apresenta um programa, baseado no estilo cadastro pai, pertencente ao aplicativo de controladoria e finanças e responsável pela implantação de lotes de registro de títulos. A identificação das EEs não é feita facilmente como nos outros estilos, pois neste caso deve-se primeiramente identificar o programa chamador para então verificar quais as EEs efetivamente realizadas pelo cadastro pai.



Devido ao padrão de desenvolvimento adotado nos produtos Datasul pode-se inferir que um programa baseado em um estilo cadastro pai geralmente terá duas EEs: interface de inclusão e de modificação.

Analisando a função de inclusão de lote de registro de títulos pode-se contar um total de dez campos e um controle, conforme descrito abaixo:

- Campo código do estabelecimento;
- Campo referência do lote de implantação;
- Campo moeda a ser utilizada no registro dos títulos;
- Campo descrição da moeda;
- Campo conta-contábil de crédito (geralmente);
- Campo portador da conta-contábil (quando conta-contábil é uma conta de banco);
- Campo referência para integração com caixa e bancos;
- Campo data de movimentação;
- Campo valor total dos títulos a serem implantados;
- Campo para representar as mensagens de erro e sucesso;
- Controle para representar os botões de gravação (ok e salvar).

Já os arquivos referenciados são sete, conforme descrito abaixo:

- Tabela de Documentos de implantação (lotes de registros de títulos);
- Tabela de parâmetros contas a receber;
- Tabela de estabelecimento;
- Tabela de moeda;
- Tabela de conta-contábil;
- Tabela de conta-corrente;
- Tabela de portador.

Entre os arquivos referenciados não há necessidade de classificá-los como ALI ou AIE, pois esta diferença não influencia na determinação da complexidade e contribuição da EE, porém é obrigatório que a EE mantenha ao menos um ALI ou altere o comportamento da aplicação.

As tabelas de parâmetros de contas a receber e de conta-corrente, embora não tenham nenhuma informação visível ao usuário, na interface gráfica, são utilizadas internamente

para realizar alguns processos de validação durante a inclusão de um novo lote de registro de títulos, devendo assim ser contabilizadas como dois arquivos referenciados.

A segunda EE, referente à modificação, deve ser analisada de forma análoga à função de inclusão. E, provavelmente, devem utilizar o mesmo número de campos e controles, e referenciar os mesmos arquivos, conseqüentemente tendo a mesma complexidade e contribuição em pontos de função.

Estilo Importação

Um programa baseado em um estilo importação geralmente possui uma única EE, que se refere à importação (inclusão). Caso existam funções adicionais (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis EEs.

Em um programa deste estilo não se deve considerar a opção de edição do layout com uma nova EE, pois não manipula nenhum arquivo lógico interno ou altera nenhum comportamento do sistema, embora muitas vezes tenha alta complexidade de implementação.

Após a identificação das EEs existentes em um determinado programa, deve-se contar o número de campos de parametrização e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados.

Quando uma funcionalidade faz a chamada de uma API, muitas vezes é necessário um determinado esforço para preparar informações e/ou tratar o seu retorno.

Nestes casos é correto contar como um processo elementar de integração quando uma funcionalidade chama uma API; mas para chamá-la, tem-se um esforço para preparar informações e/ou tratar o seu retorno.

Para os layouts dos bancos deve-se contar como um processo elementar de layout que terá os TDs e AR que são iguais para a maioria dos bancos e um processo elementar para cada banco considerando apenas os TDs e AR exclusivos daquele banco.

Contagem do Número de Campos de Parametrização e Controles

A contagem do número de campos de parametrização é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica. Deve-se apenas levar em consideração que campos repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda, devem ser contados como dois campos; ex.2.: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*).

Também se devem contar três campos adicionais: um referente às mensagens de erro e sucesso apresentadas ao usuário; outro referente às opções de log do processo de importação; e por último um referente à opção de layout.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex: botões de editar layout, executar, pesquisar arquivo de entrada ou saída, são contados como apenas um controle*).

Contagem do Número de Arquivos Referenciados

A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs mantidos pela EE após, deve-se então verificar outros ALIs ou AIEs utilizados (consultados) mas não mantidos pela função (*ex.: interface de inclusão de bancos mantém o ALI banco e referencia o ALI de instrução bancária*).

Geralmente, a contagem de ALIs ou AIEs utilizados (consultados) mas não mantidos pela EE se deve a existência de chaves estrangeiras nas interfaces gráficas.

Estudo de Caso: Importação de Lotes Comprados de Determinado Item

A imagem abaixo apresenta um programa, baseado no estilo importação, pertencente ao aplicativo de materiais e responsável pela importação de lotes comprados de determinado item. Identificando-se facilmente a EE básica: importação.

Analisando a única função, importação, podem-se contar quatro campos e um controle, conforme descrito abaixo:

- Campo para representar o layout de importação;
- Campo arquivo para importação (entrada);
- Campo para representar as opções de log do processo de importação;
- Campo para representar as mensagens de erro e sucesso;
- Controle para representar os botões de importação e de pesquisa de arquivos (entrada e saída).

Já os arquivos referenciados são cinco, conforme descrito abaixo:

- Tabela lotes comprados;
- Tabela item;
- Tabela estabelecimento;
- Tabela emitente (fornecedor);
- Tabela saldo em estoque.

Entre os arquivos referenciados não há necessidade de classificá-los como ALI ou AIE, pois esta diferença não influencia na determinação da complexidade e contribuição da EE, porém é obrigatório que a EE mantenha ao menos um ALI ou altere o comportamento da aplicação.

As tabelas de item, estabelecimento, emitente (fornecedor) e saldo em estoque, são utilizadas internamente para realizar alguns processos de validação durante a inclusão de um novo lote comprado de determinado item, devendo assim ser contabilizada como quatro arquivos referenciados.

Outros Estilos

Os produtos Datasul fazem uso de outros estilos, além dos mencionados anteriormente, mas o processo de identificação de campos e controles e arquivos relacionados funcionam de forma análoga.

O estilo formação, por exemplo, possui geralmente três EEs: de inclusão, de modificação e de remoção; assemelhando-se em muito com o estilo cadastro pai x filho. Já o estilo parâmetro único é muito parecido com o estilo cadastro pai ou cadastro filho, só que geralmente é contada apenas uma EE, referente à modificação.

Durante o processo de identificação de EEs não se deve considerar que dois programas diferentes mas que apresentam uma mesma funcionalidade ao usuário, com os mesmos campos e arquivos referenciados, sejam duas EEs diferentes. Este comportamento pode ser encontrado em programas que mantém uma mesma tabela, mas um utiliza o estilo cadastro simples e o outro cadastro simples atualiza.

5. CONSULTAS EXTERNAS (CE)

Uma consulta externa é um processo elementar que envia dados ou informação de controle para fora da fronteira do aplicativo. A intenção primária de uma CE é apresentar informações ao usuário através da recuperação de dados e informações de controle de um ou mais ALIs ou AIEs. E o processamento lógico de uma CE não contém nenhuma fórmula matemática ou cálculo, não cria dados derivados e não altera o comportamento do sistema.

Sendo muito comum aos programas que apresentam EEs apresentarem também CEs, pois antes da alteração de dados normalmente é feita a visualização dos mesmos. Em decorrência disso, todos os estilos apresentados anteriormente (exceto o estilo Importação) apresentação uma transação do tipo CE.

5.1. COMO IDENTIFICAR CONSULTAS EXTERNAS

Para auxiliar na identificação de CEs utilize os seguintes recursos:

- Para cada ALIs geralmente existirá uma interface gráfica para consulta, que geralmente também possui funções para manutenção do ALI;
- Para os produtos Datasul uma parcela significativa dos programas segue estilos pré-definidos (*templates*), permitindo uma rápida identificação das CEs baseando-se apenas no estilo adotado pelo programa;

Para programa sem interface gráfica também podem ser identificadas CEs, isto geralmente acontece nas DBOs.

Observações

Durante o processo de identificação de CEs não devem ser considerados os ALIs ou AIEs referentes a menu, parâmetros globais, segurança e usuário como arquivos referenciados, pois são inerentes aos produtos Datasul. E caso sejam contados, podem superestimar a complexidade das funções do tipo transação rapidamente. A contagem de tais arquivos deve ser restrita ao aplicativo integração e tecnologia.

Programas baseados nos estilos apresentados a seguir não precisam ser considerados sempre CEs, pois é comum que programas nestes estilos também sejam classificados como SEs (saídas externas).

5.2. COMO DETERMINAR A COMPLEXIDADE DE CONSULTAS EXTERNAS

Com a identificação das CEs concluída, basta contar o número de campos e controles existentes, e o número de arquivos (ALIs ou AIEs) referenciados, para determinar a complexidade com base na tabela abaixo:

	Número de Campos e Controles		
Número de Arquivos	< 6	6 – 19	> 19
< 2	Baixa	Baixa	**Média**
2 – 3	Baixa	**Média**	Alta
> 3	**Média**	Alta	Alta

5.3. COMO DETERMINAR A CONTRIBUIÇÃO DAS CONSULTAS EXTERNAS

Com a complexidade determinada, o cálculo da contribuição das CEs em número de pontos de função é feito baseado na tabela abaixo:

COMPLEXIDADE		
Baixa	Média	Alta
x 3	x 4	x 6

5.4. APLICAÇÃO CONSULTAS EXTERNAS

Apresentamos a seguir exemplos de contagem de processos elementares do tipo Consulta Externas(CE) sobre alguns estilos de programas.

Estilo Consulta Simples e Consulta Complexa

Um programa baseado em um estilo consulta simples ou consulta complexa geralmente possui duas CEs: interface de consulta (navegação) e de pesquisa. Caso exista função adicional (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis CEs.

A interface de "vá para" não deve ser considerada como uma nova CE, pois é apenas um recurso de navegação adicional. E, geralmente, não são contabilizados novos campos, controles e arquivos, pois estes já são contabilizados pela própria consulta (navegação).

Após a identificação das CEs existentes em um determinado programa, deve-se contar o número de campos e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados.

Embora o estilo complexo apresente várias páginas para digitação e apresentação de campos, isto não deve ser visto como novas funções ou CEs. A apresentação em diferentes páginas geralmente é apenas um recurso tecnológico para contornar a deficiência relacionada ao limite da área de tela.

Observação: Os estilos cadastro simples e cadastro complexo também apresentam CEs, que podem ser identificadas utilizando-se a mesma abordagem adotada para os estilos consulta simples e consulta complexa.

Contagem do Número de Campos e Controles

A contagem do número de campos é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica. Deve-se apenas levar em consideração que campos repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda devem ser contados como dois campos; ex.2.: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*).

Também se deve contar um campo adicional referente às mensagens de erro e sucesso apresentadas ao usuário, ou seja, um único campo contabilizará todas as mensagens apresentadas ao usuário.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex: botões de navegação em ocorrências: primeira, anterior, próxima, última, vá para, são contados como apenas um controle*).

Contagem do Número de Arquivos Referenciados

A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs principais da CE após, deve-se então verificar outros ALIs ou AIEs utilizados (consultados) pela função (*ex.: interface de consulta de bancos recupera o ALI banco e referencia o ALI de instrução bancária*).

Geralmente, a contagem de ALIs ou AIEs adicionais se deve à existência de chaves estrangeiras nas interfaces gráficas.

Estudo de Caso: Consulta de Planejadores

A imagem abaixo apresenta um programa, baseado no estilo consulta simples, pertencente ao aplicativo de manufatura e responsável pela consulta de planejadores. Neste programa podem-se identificar facilmente duas CEs: consulta (navegação) e pesquisa.

Analisando a função de consulta (navegação) pode-se contar um total de quatro campos e um controle, conforme descrito abaixo:

- Campo código do planejador;
- Campo nome do planejador;
- Campo e-mail do planejador;
- Campo para representar as mensagens de erro e sucesso;
- Controle para representar os botões de navegação.

Já o número de arquivos referenciados é apenas um, conforme descrito abaixo:

- Tabela planejador.

E não há necessidade de classificá-lo como ALI ou AIE, pois esta diferença não influencia na determinação da complexidade e contribuição da CE, porém é obrigatório que a CE recupere dados de ao menos um ALI ou AIE.

A outra CE existente neste programa será analisada no tópico **Pesquisa**, que tratará de CEs baseadas neste estilo .

Estilo Consulta Relacionamento

Um programa baseado em um estilo consulta relacionamento geralmente possui duas CEs: interface de consulta (navegação) e de pesquisa. Também é comum a existência de CEs para detalhamento dos registros filhos. Nestes casos, normalmente são apresentados outros programas baseados nos estilos consulta simples, consulta complexa ou consulta relacionamento. Caso existam funções adicionais (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis CEs.

A interface de "vá para" não deve ser considerada como uma nova CE, pois é apenas um recurso de navegação adicional. E, geralmente, não são contabilizados novos campos, controles e arquivos, pois estes já são contabilizados pela própria consulta (navegação).

Após a identificação das CEs existentes em um determinado programa, deve-se contar o número de campos e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados. Da mesma forma que é feita para um programa baseado nos estilos consulta simples e consulta complexa.

E para cada uma das várias páginas, representando os diferentes filhos, devem ser contabilizados os campos e arquivos referenciados como pertencentes a CE do pai. Visto que para o usuário está sendo feita uma única consulta a todas as informações.

***Observação**: Os estilos cadastro pai x filho ou formação também apresentam CEs que podem ser identificadas utilizando-se a mesma abordagem adotada para o estilo consulta relacionamento.*

Contagem do Número de Campos e Controles

A contagem do número de campos é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica, tanto os presentes em *frames* como os presentes em *browsers*. Deve-se apenas levar em consideração que campos

repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda, devem ser contados como dois campos; ex.2..: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*).

Também se deve contar um campo adicional referente às mensagens de erro e sucesso apresentadas ao usuário, ou seja, um único campo contabilizará todas as mensagens apresentadas ao usuário.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex: botões de navegação em ocorrências: primeira, anterior, próxima, última, vá para, são contados como apenas um controle*).

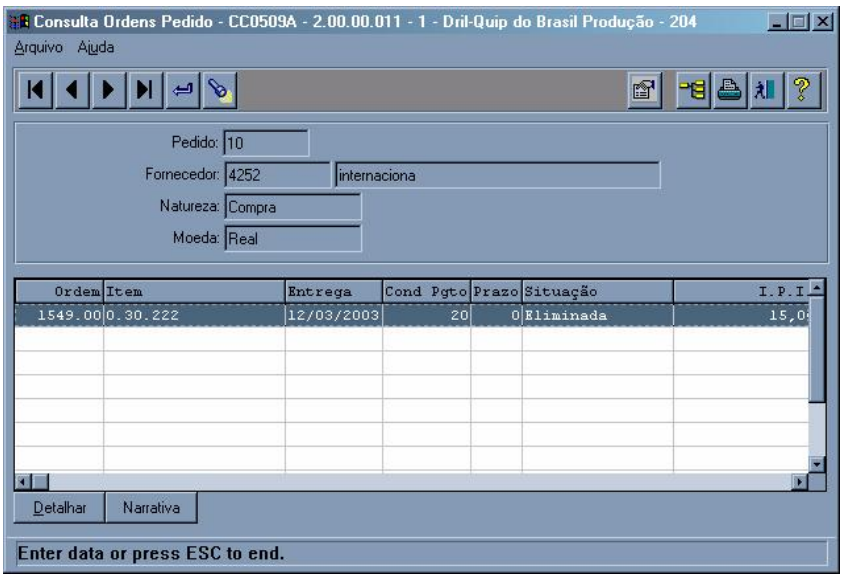
Contagem do Número de Arquivos Referenciados

A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs principais da CE após, deve-se então verificar outros ALIs ou AIEs utilizados (consultados) pela função (*ex.: interface de consulta de bancos recupera o ALI banco e referencia o ALI de instrução bancária*).

Geralmente, a contagem de ALIs ou AIEs adicionais se deve à existência de chaves estrangeiras nas interfaces gráficas.

Estudo de Caso: Consulta de Pedidos de Compra

A imagem abaixo apresenta um programa, baseado no estilo consulta relacionamento, pertencente ao aplicativo de distribuição e responsável pela consulta de ordens de compra associadas a pedidos de compra[2]. E neste programa podem-se identificar facilmente duas CEs: consulta (navegação) e pesquisa.



Analisando a função de consulta (navegação) pode-se contar um total de dezesseis campos e um controle, conforme descrito abaixo:

[2] O programa de consulta de ordens de compra associadas a pedidos de compra foi modificado para não possuir campos calculados, e assim poder ser classificado como uma consulta externa. Caso os campos calculados sejam mantidos, o programa deverá ser classificado como uma saída externa.

- Campo código do pedido;
- Campo código do fornecedor;
- Campo nome do fornecedor;
- Campo natureza do pedido;
- Campo de moeda a ser utilizada nas ordens do pedido;
- Campo (coluna) número da ordem;
- Campo (coluna) código do item;
- Campo (coluna) data de entrega prevista;
- Campo (coluna) condição de pagamento;
- Campo (coluna) prazo de pagamento;
- Campo (coluna) situação;
- Campo (coluna) I.P.I.;
- Campo (coluna) preço do item;
- Campo (coluna) quantidade compra;
- Campo para representar as mensagens de erro e sucesso;
- Controle para representar os botões de navegação.

Já o número de arquivos referenciados são três, conforme descrito abaixo:

- Tabela de pedidos de compra;
- Tabela de ordens de compra;
- Tabela de prazos de entrega de ordem de compra.

Mas, fazendo-se uma análise mais detalhada, pode-se avaliar que a tabela de prazos de entrega de ordem de compra poderia ser vista como um segundo tipo de registro no arquivo de ordens de compra. Para o usuário existe um único arquivo, para armazenar os dados de ordens de compra, mas por questões de normalização de modelos entidade-relacionamento este arquivo foi dividido em dois. E desta forma deve-se contar apenas dois ALI.

Entre os arquivos referenciados não há necessidade de classificá-los como ALI ou AIE, pois esta diferença não influencia na determinação da complexidade e contribuição da CE, porém é obrigatório que a CE recupere dados de ao menos um ALI ou AIE.

A outra CE existente neste programa será analisada no tópico **Pesquisa**, que tratará de CEs baseadas neste estilo.

Estilo Pesquisa

Um programa baseado em um estilo pesquisa geralmente possui uma única CE: interface de consulta (*browser*, filtro e classificação). Caso existam funções adicionais (geralmente disponíveis através de botões extras), estas também devem ser analisadas para identificar possíveis CEs.

As várias páginas de classificação de dados não devem ser consideradas novas CES, pois são apenas recursos tecnológicos para apresentação das opções de classificação, assim como as várias opções de filtro também não devem ser consideradas novas CEs.

Após a identificação das CEs existentes em um determinado programa, deve-se contar o número de campos e controles disponíveis ao usuário, e também o número de arquivos (ALIs e AIEs) referenciados.

As pesquisas (Zoom) que são utilizadas em diversas Funções do aplicativo (reaproveitadas) devem ser contadas apenas uma única vez. A sugestão é que elas

sejam contadas em separado dos programas onde são chamadas, evitando assim que elas sejam contadas de forma duplicada.

Contagem do Número de Campos e Controles

A contagem do número de campos é simples, bastando apenas verificar as variáveis e campos de tabelas apresentados na interface gráfica. Deve-se apenas levar em consideração que campos repetidos devem ser contados como apenas dois campos (*ex.1.: doze campos que representam as cotações mensais de uma moeda, devem ser contados como dois campos; ex.2.: doze meses que são representados em um gráfico associado com um campo de valor, devem ser contados como dois campos apenas, campo mês e campo valor*).

Também se deve contar um campo adicional referente às mensagens de erro e sucesso apresentadas ao usuário, ou seja, um único campo contabilizará todas as mensagens apresentadas ao usuário.

Os controles apresentados devem ser contados como apenas um, mesmo que sejam apresentados ao usuário inúmeros botões (*ex: botões de retorno da pesquisa e ativação de filtro são contados como apenas um controle*).

Contagem do Número de Arquivos Referenciados

A contagem do número de arquivos referenciados inicia-se normalmente pela identificação dos ALIs principais da CE após, deve-se então verificar outros ALIs ou AIEs utilizados (consultados) pela função (*ex.: interface de consulta de bancos recupera o ALI banco e referencia o ALI de instrução bancária*).

Geralmente, a contagem de ALIs ou AIEs adicionais se deve à existência de chaves estrangeiras nas interfaces gráficas.