A MEDIÇÃO DA PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE

JOSÉ ALBERTO DA COSTA MACHADO

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUESITOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Ana Regina Cavalcanti da Rocha, D.SC
(Orientadora)

Sueli Bandeira Teixeira Mendes, PhD

Julio Cesar S. Leite, PhD

Rio de Janeiro, RJ - Brasil

Abril de 1990.

MACHADO, JOSÉ ALBERTO DA COSTA

A Medição da Produtividade no Desenvolvimento de Software (Rio de Janeiro) 1990

XI,250 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1990)

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Produtividade I.COPPE/UFRJ II.Título(Série)

Dedico este trabalho à minha esposa Maria das Dores e às minhas filhas Janara, Juliana e Joice.

AGRADECIMENTOS

Ao agradecer aqueles que tornaram possível a realização deste trabalho, desejo destacar:

A Prof. ANA REGINA CAVALCANTI DA ROCHA, não só pela sua orientação acadêmica, mas também pela confiança que depositou na minha admissão como mestrando.

A Prof. SUELI BANDEIRA T. MENDES, pela confiança que me ensejou ao apoiar minha admissão no mestrado.

Os srs. JOEL CRUZ e JOSE EDUARDO BELARMINDO ALCOFORADO que me ensejaram apoio significativo para a realização do mestrado.

A Empresa de Águas Santa Claúdia, na pessoa do seu Diretor Geral FERNANDO MATOS DE SOUZA FILHO pela ajuda que me prestou.

A LEA FERREIRA DE JESUS e IOLANE DA COSTA MACHADO pela tranquilidade que o auxílio de ambas me propiciou.

A UNISYS ELETRONICA LTDA que me possibilitou acesso a material bibliográfico.

A RONALDO BRANDÃO, ANTONIO BRAGA e JOSE ROBERTO que, como funcionários da UNISYS ou em carater pessoal, me ensejaram material de pesquisa.

Os meus pais pela confiança e estímulo que me ensejaram.

O meu amigo RAIMUNDO GUEDES MOURA que, ao longo das lides profissionais comuns, me propiciou reflexões aqui incluidas.

A LEYLA MARIA SILVA DA CUNHA pelo grande auxílio que me prestou no preparo da versão inicial deste trabalho.

A SUELLY NUNES MENDONÇA pelo auxílio que me prestou no preparo da parte final do trabalho.

Desejo registrar, com ênfase possível, minha gratidão especial a JOSÉ LAURINDO CAMPOS DOS SANTOS. Sem ele, que também é M.Sc em Sistemas e Computação pela Universidade Federal da Paraíba, teria sido mais difícil a realização deste trabalho. Seu apoio na edição e no ajuste final da tese, bem como, seu incentivo initerupto, ultrapassaram, de muito, o dever da amizade.

Aos meus alunos ENIO BARBOSA, LISA MARA LINS E LUCIO MARIO DIAS, do Curso de Graduação em Processamento de Dados - Universidade do Amazonas, pelo trabalho de implementação que fizeram.

Sobretudo desejo registrar minha gratidão a Misericórdia Divina que, através de mil modos, sempre esteve presente ao meu lado durante este trabalho.

Resumo da tese apresentada a COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

" A MEDIÇÃO DA PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE "

JOSÉ ALBERTO DA COSTA MACHADO

Abril de 1990

Orientador: ANA REGINA CAVALCANTI DA ROCHA

Programa: ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Esta tese objetiva apresentar formas consistentes de medir a produtividade na produção de software. Após analisar o significado que a produtividade assume no contexto da área, são discutidas as diversas expressões com as quais a produtividade é medida. São examinadas as métricas mais comuns usadas na medição de software bem como os problemas gerais de todas e as particulares de cada uma. Com base nas características da medida ideal, propostas no trabalho, é escolhida a metodologia Pontos por Função, que é então detalhada e examinada em todos os seus aspectos. Como parte dessa metodologia são apresentadas formas consistentes de calcular tamanho de software, esforço do trabalho e a determinação da produtividade.Tambem são apresentadas alternativas para estimar tempo de desenvolvimento. São analisados os fatores que influenciam a produtividade e, por fim, é proposta a estrutura geral de um ambiente de gerência na produção de software.

Abstract of thesis presentes to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

" THE MEASUREMENT OF PRODUCTIVITY ON SOFTWARE DEVELOPMENT "

JOSE ALBERTO DA COSTA MACHADO

April/1990

Thesis Supervisor: ANA REGINA CAVALCANTI DA ROCHA

Department: COMPUTER SCIENCE AND SYSTEM ENGINEERING

The aim of this thesis is to present consistent ways of measuring productivity on software development. After analysing the real meaning of the productivity, different forms of measurement, are discussed. The most common metrics used on software measurement are examined, as well as their specific and general problems. Considering the characteristics of the ideal metric, proposed in this thesis, the Function Point Methodology was chosen and examined in details. Consistent ways to determine the software size, work-effort and the measurement of the productivity results are presented, as well as a method to support the process of the estimating on software development. Factors affecting productivity are defined and a system to support the management process on software development is proposed.

A MEDIÇÃO DA PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE

INDICE PAG

CAPÍTULO I

INTRODUÇÃO

No sentido econômico a produtividade é expressada através da relação existente na fórmula abaixo:

$$\text{Produtividade} = \frac{\text{Produto do Trabalho}}{\text{Esforço do Trabalho}}$$

Em ambientes de produção de software, a fórmula simples acima é um problema complicado e não resolvido.

A dificuldade não está somente na forma de se expressar o produto do trabalho, ou seja, o tamanho do software. Ela se verifica também na indefinição sobre o que considerar para medição do Esforço do Trabalho.

Como é que se mede o tamanho de um software de forma que ele expresse a real grandeza do software e não seja condicionado por fatores externos tais como experiência da equipe, linguagem usada, etc ?

Qual o escopo a ser considerado para a medição do esforço empregado ? Seriam todas as atividades ligadas ao ciclo de desenvolvimento ou somente à fase de codificação ?

Que tipos de atividades devem ser consideradas uma vez que existem atividades como treinamento do usuário, documentação, etc, que não são exclusivas de nenhuma fase específica mas perpassam o projeto como um todo ? Há outras que não pertencem a nenhum projeto especificamente, como cursos, reuniões, atividades

administrativas, etc mas que são importantes no contexto do desenvolvimento como um todo. Como considerar, por exemplo, o tempo improdutivo das pessoas como fins de semana, férias, etc ?

Como devem ser considerados os projetos que foram cancelados ? E os códigos incluídos no software com objetivo de suporte ou de apoio à qualidade ? Esses códigos, que não fazem parte do projeto em si, entram na medição do tamanho ?

O que deve ser entendido por manutenção e como medir seu tamanho e o esforço nela empregados ?

Como medir a produtividade das pessoas, de cada projeto e da empresa como um todo ?

Estas e outras questões são problemas pendentes de solução e, de forma geral, estão presentes na maioria das técnicas utilizadas atualmente.

Entre as técnicas mais utilizadas atualmente está o uso das Linhas de Código Fonte como métrica para expressar o tamanho do software. Entretanto, essa técnica tem uma série de problemas, entre os quais destacamos:

- origem do seu uso;
- efeitos paradoxais;
- ausência de regras padrões na sua contagem;
- inadequada para linguagens de alto nível;
- ambiguidade.

Uma outra técnica muito utilizada é a conhecida por métricas de Halstead. Como problemas dessa técnica, podemos apontar:

- deduções baseadas em resultados esperados;
- presunção científica indevida;
- repetição dos problemas de LCFs;
- inadequada para previsão;
- não aborda manutenção.

A Complexidade Ciclomática é outra técnica utilizada para medir tamanho de software, que também tem seus problemas, entre os quais, destacamos:

- inadequado potencial de expressão;
- abrangência restrita;
- imprópria para estimativas;
- ofusca origem da complexidade;
- dependente da experiência pessoal;
- dependente da tecnologia.

Uma técnica nova proposta em 1979 por Allan Albrecht, é a chamada Pontos por Função que, apesar de alguns problemas, tem apresentado ampla aceitação e promissores resultados.

Após exame detalhado de todas essas técnicas este trabalho propõe um conjunto de características que deveriam fazer parte de uma medida ideal. São elas: generalidade, objetividade, facilidade, significabilidade, mutabilidade, expressividade, isolabilidade, previsibilidade, agregabilidade, computabilidade, suficiência e cientificidade.

A metodologia Pontos por Função, destinada a medir o tamanho de software por atender à maioria dessas características, é então detalhada, aprofundada conceitualmente e expandida.

Em seguida, um método sistematizado para medição do Esforço do Trabalho é detalhado e abangentemente proposto.

Com as duas variáveis, Produto do Trabalho e Esforço do Trabalho, completa e objetivamente definidas, é então apresentado uma forma de medir e avaliar a produtividade tanto para desenvolvimento quanto para manutenção, bem como, a nível individual, de projeto e da organização como um todo.

Com base nos resultados ensejados pelas definições anteriores são propostas alternativas para, de forma simples e consistente, estimar o esforço no desenvolvimento e manutenção de software.

Apresenta também, em análise e justificativa individual, uma ampla lista de fatores considerados como influenciadores da produtividade.

Reunindo todos os elementos tratados ao longo do seu texto, o trabalho propõe a estrutura geral de um possível ambiente de gerência na produção de software.

CAPÍTULO II

O SIGNIFICADO DA PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE

Neste capítulo, após considerações sobre a dimensão que a produtividade assume no ambiente de produção de software, será analisado o contexto no qual ela se expressa, sobretudo considerando as diversas formas sob as quais ela tem sua medida expressada. Assim, serão examinados a expressão econômica, custo por defeito, percentagem por fase e produtividade pela efetividade.

II.1 - A DIMENSÃO DA PRODUTIVIDADE

A literatura sobre computação tem-na classificado como o maior fenômeno industrial da história humana especialmente pela sua penetração e associação à maioria dos ramos industriais, como à indústria automobilística, à indústria aeronáutica, às telecomunicações e todas as novas tecnologias que estão moldando a vida humana.

Prevê-se que até o final do século o computador seja um eletrodoméstico de uso comum nas residências e que sua programação seja uma das matérias ensinadas nos primeiros anos escolares juntamente com leitura, escrita e aritmética básica.

Esse avanço, entretanto, não tem se transformado em benefícios concretos para a maioria da humanidade. Ao

contrário, os custos e recursos que essa indústria tem consumido são tão significativos que ofuscam os ganhos e melhorias proporcionados.Não é incomum que os custos com pessoal de processamento de dados (PD), na maioria das empresas, ultrapasse hoje 3% do total da folha de pagamento e que as despesas com computação em geral ultrapasse 5% de todas as vendas, conforme JONES(9).

Tentativas de automatizar atividades críticas nem sempre tem sido bem sucedidas, seja pelo alto custo e demora no desenvolvimento, erro em demasia, baixo nível de qualidade, sistemas inamigáveis, alto custo de manutenção, etc. Isso tem levado a um consenso entre as gerências de negócios em geral que a produçao de software é a menos profissional e a mais problemática das altas tecnologias emergentes. Normalmente ela é vista como a mais custosa, errônea, lenta, não medível e imprevisível quanto a duração, recursos e confiabilidade.

Some-se a isso a importância vital que a computação assumiu ante as operações de empresas e governos; ante o projeto e desenvolvimento de produtos de alta-tecnologia; ante o crescente uso de software embutido em produtos críticos; ante as atividades e programas militares de pequena e larga escala; enfim, ante uma gama ampla de frentes com impactos ostensivos na vida humana e então teremos justificado o interesse que ultimamente tem levado a comunidade da área a estudar formas de produzir produtos de software que minimizem os problemas antes considerados.

Entre as diversas áreas de pesquisa, situa-se a produtividade, interessada em examinar todos os fatores que contribuem para a produção de software em menos tempo e com menos custos.

II.2 - O CONTEXTO DA PRODUTIVIDADE

II.2.1 - A EXPRESSÃO ECONOMICA

Apesar da ambiguidade no uso do termo, consideraremos aqui que " produtividade ", no sentido econômico, representa a relação entre o que foi produzido e o esforço feito para produzi-lo. Esse esforço pode significar tempo dispendido ou custo. No caso de custo, a relação se inverte, como na fórmula baixo:

$$\text{PRODUTIVIDADE} = \frac{\text{PRODUTO}}{\text{ESFORÇO}} \quad (a) \text{ ou } \frac{\text{CUSTO}}{\text{PRODUTO}} \quad (b)$$

A medida que a relação(a) aumenta ou a relação(b) diminui diz-se que há um aumento de produtividade e, dependendo em qual das duas haja a diminuição, podemos ter as seguintes hipóteses:

a) Produzir em menos tempo com menos recursos (relação aumenta em (a) e diminui em (b)).
b) Produzir em menos tempo com os mesmos recursos (relação aumenta em (a) e permanece igual em (b)).
c) Produzir em menos tempo com mais recursos (relação aumenta em (a) e aumenta em (b)).

d) Produzir no mesmo tempo com menos recursos (relação permanece igual em (a) e diminui em (b)).

e) Produzir em mais tempo e com menos recursos (relação diminui em (a) e diminui em (b)).

Nas hipóteses "c" e "d" poderá haver ou não produtividade, enquanto nas demais sempre haverá produtividade real.

Para o caso de software, entretanto, a fórmula simples considerada anteriormente é um problema complicado e ainda não resolvido.Com referência a "esforço" e "custo" as tentativas de solucionar ja são significativas na literatura. Entre as contribuições consistentes situa-se LEITE(56) não só pela compilação das possíveis alternativas, mas tambem pela proposta apresentada que baseia-se na visão clássica da contabilidade de custos. Outra fonte clássica no trato dessas duas questões é BOEHM(8) com as diversas variações do seu modelo COCOMO(COnstructive COst MOdel). Entretanto, com referência a "produto", não acontece o mesmo.

O que é um produto de software ? Como se mede e de que forma essa medida é expressada ? O que compõe um software para efeito de medir seu tamanho ? Quais as variáveis e fatores que influenciam a expressão de sua grandeza ? Todas essas questões ainda não tem respostas universalmente aceitas, entretanto, do que já existe, retiraremos o apoio para tratar delas nos capitulos seguintes.

Conhecer a exata grandeza de um software não é

somente importante para medir produtividade na sua criação, mas, na mesma dimensão de importância, servirá também para prever custo e tempo de desenvolvimento com um grau de acurácia que se situe no limite de mais ou menos 20% de erro em pelo menos 70% das vezes conforme BOEHM(8) ou, mais otimizadamente, em mais ou menos 15% em pelo menos 90% das vezes conforme JONES(9) afirma já ser possivel.

Com isso, e considerando o conhecimento já existente sobre os fatores que influenciam a produção de software, estaremos instrumentalizados para resolver as cinco grandes questões tidas por JONES(9) como responsabilidade dos que trabalham na produção de software. São elas:

- diminuir o tempo de desenvolvimento de novos softwares ;
- reduzir substancialmente os custos de desenvolvimento ;
- minimizar a taxa do aumento de pessoal envolvido com computação nas instituições ;
- tornar previsíveis e controláveis o tempo, o custo, e a produtividade no desenvolvimento de software, e,
- aumentar o nível de qualidade e confiabilidade dos softwares produzidos.

Neste trabalho trataremos produtividade no seu sentido econômico, tal como considerado nesta seção.

II.2.2 - OUTRAS EXPRESSÕES DA PRODUTIVIDADE

As expressões relacionadas nesta seção também são consideradas na literatura como indicadores de produtividade, entretanto, como elas se referem a visões especificas e com

objetivos bem particularizados não receberão destaque no escopo deste trabalho, embora, quando necessário, venham a ser referidas.

Além das razões acima, considera-se que elas tem campo próprio de estudo (qualidade, confiabilidade, etc) ou então que seus significados são plenamente cobertos pelo sentido econômico da produtividade. São elas :

a) Custo por defeito.

Esta medida, associada à qualidade do software, expressa os custos incorridos para remoção de um defeito. Normalmente, é utilizada para comparar o custo por defeito ocorrido no desenvolvimento, com o custo de defeito ocorrido na manutenção ou, para comparar este último em softwares diferentes. Apesar de ser comum a idéia de que a remoção de defeitos durante a manutenção custa até 100 vezes mais do que durante o desenvolvimento, essa ocorrência, " tal como usada, é incorreta e não tem base em fatos " conforme observa JONES(9).

O que ocorre é que, à medida que o numero de erros declina, o custo de cada erro aumenta por causa dos custos fixos associados à manutenção. Isso conduz a uma conclusão paradoxal, pois, quanto menos erros apresentar o software na fase de manutenção maiores serão os custos para removê-los, o que não é verdade.

Quando compara-se o custo por defeito de um software de alta qualidade com um de baixa qualidade chega-se a falsa

conclusão de que este último é mais produtivo. Em verdade o que acontece é que, seus custos com atividades relativas a remoção de erros (revisões, teste, etc), apesar de muito maiores no total, se tornam menores quando considerados unitariamente, por causa da maior quantidade de erros encontrada.

b) Percentagem por fase.

É comum o uso de percentagem para definir o esforço a ser dispendido em cada fase do desenvolvimento. Costuma-se considerar a distribuição abaixo (figura 1) para comparar a produtividade obtida conforme a linguagem utilizada:

LINGUAGEM	PROJETO	CODIFICAÇAO	INTEGRAÇÃO/TESTE
BAIXO NIVEL	20%	30%	50%
ALTO NIVEL	40%	35%	25%
MUITO-ALTO NIVEL	5%	90%	5%

Figura 1 : A participação por fase relacionada ao nivel da linguagem
Fonte : JONES(9)

Entretanto, essa comparação pode ser enganosa quando software em linguagens diferentes são examinados. Considerando a tabela abaixo apresentada por JONES(9), na qual foram comparados dois programas com as mesmas funções mas em linguagens diferentes, temos os seguintes paradoxos:

ATIVIDADE	30.000 LCF/ASSEMBLER		10.000 LCF/COBOL	
	ESFORÇO (meses)	PERCENTAGEM	ESFORÇO (meses)	PERCENTAGEM
1-ESPECIFICAÇÃO	5	3,33	5	5,00
2-PROJETO	10	6,66	10	10,00
3-CODIFICAÇÃO	50	33,33	20	20,00
4-DOCUMENTAÇÃO	30	20,00	30	30,00
5-INTEGRAÇÃO	15	10,00	10	10,00
6-TESTE	30	20,00	20	20,00
7-GERÊNCIA	10	6,66	5	5,00
TOTAL	150	100,00%	100	100,00%

Figura 2 : O paradoxo da percentagem como indicador da produtividade

Legenda : LCF - linhas código fonte
Fonte : JONES(9)

- as atividades 1,2 e 4 consumiram o mesmo tempo mas apresentaram percentagens diferentes.

- as atividades 5 e 6 apresentam, entre os dois exemplos, uma diferença de 15 meses apesar de possuirem a mesma participação percentual.

Além desses contrastes, devidos a linguagem, existem outros decorrentes do uso ou não de ferramentas automatizadas, a novidade do assunto que está sendo automatizado, experiência da equipe, etc. Todos esses são fatores que distorcem o uso de

percentagem como medida de produtividade.

c) Produtividade pela efetividade.

Desenvolve-se software para aumentar a velocidade dos processos manuais, para prover acesso rápido a um grande número de informações e para desempenhar funções que não possam ser feitas manualmente.

Medir a produtividade no atendimento a esses objetivos significa examinar se se está fazendo tão bem quanto outros, se se está fazendo melhor à medida que o tempo passa e se se está fazendo tão bem quanto é possível fazer.

Muitos estudos, incluindo JONES(9), chamam isso de produtividade econômica. Neste trabalho, esse aspecto não é considerado e a dimensão econômica, descrita anteriormente neste capítulo, está associada ao processo de desenvolvimento do software e não à efetividade do seu uso.

Do ponto de vista de como é aqui considerada, pode haver um excelente nível de produtividade durante a criação do software e o seu uso não ter a mínima efetividade para a organização.

Ambos são apresentados na literatura como conceitos econômicos da produtividade, sendo que um trata da produtividade no processo de desenvolvimento e o outro da produtividade decorrente da utilização do software.

Neste trabalho, o conceito considerado é o da produtividade no desenvolvimento, isto é, a relação existente entre o Produto do Trabalho e o Esforço feito para produzi-lo.

CAPÍTULO III

AS MEDIDAS ATUAIS E SEUS PROBLEMAS

Neste capítulo, serão analisadas as medidas que, segundo CURTIS(18), são as pesquisadas experimentalmente. Após análise dos problemas comuns, será discutida cada uma especificamente.

Assim, sob o título de problemas comuns, serão examinados no ítem III.1 a extensão do escopo, as atividades consideradas projetos cancelados, ausência de medidas específicas, código de suporte, código de apoio à qualidade, desvios conceituais e a indistinção existente entre os vários conceitos ligados às atividades de manutenção.

No ítem III.2 será analisada a métrica mais comumente utilizada que é LCF - Linhas de Código Fonte. Depois de analisar seu histórico e os efeitos paradoxais que seu uso causa, será discutida a grave questão da ausência de regras de contagem para essa métrica.

No ítem III.3 será analisado o sistema métrico proposto por Halstead, considerando suas características gerais e os diversos problemas que estão relacionados a esse tipo de medida.

No ítem III.4 será analisada a complexidade como medida, incluindo uma definição geral e os problemas que lhe são inerentes.

No item III.5 será analisada a metodologia Pontos por Função, onde serão discutidas sua natureza e origem, será apresentado um resumo de sua proposta inicial, e por fim, serão examinadas as restrições e aceitação que essa metodologia tem recebido.

No item III.6 deste capítulo será apresentado um conjunto de características que, segundo a ótica deste trabalho, deveriam fazer parte de uma medida ideal.

III.1 - PROBLEMAS COMUNS A TODAS AS MEDIDAS

Seja qual for a medida a ser utilizada, uma série de problemas surge como sendo comuns a todas elas. Nesta seção serão apresentados os principais problemas e abordada a natureza de cada um.

- A extensão do escopo

Qual o escopo que deve ser considerado para medir o custo e o tempo gasto na produção de um software? Devem-se considerar todas as atividades que constituem o ciclo de desenvolvimento ou somente a fase de codificação? Em que ponto, exatamente, começam a ser considerados os custos de um projeto e qual é o momento em que cessa sua ocorrência?

- Atividades consideradas

Além das atividades inerentes a cada fase do escopo considerado existem outras que não são comuns a nenhuma

fase específica mas perpassam o projeto como um todo, como documentação, treinamento do usuário, revisão de qualidade, etc. Há outras, ainda, que não são específicas de nenhum projeto mas que influenciam a cada um em determinados aspectos, como por exemplo, participação em cursos, reuniões, atividades administrativas, etc. Como considerar essas atividades? E o tempo improdutivo das pessoas, como por exemplo fins de semana, férias e outros?

- Projetos cancelados

Como considerar as atividades executadas em projetos que foram cancelados? Devem ser consideradas atividades produtivas? De que maneira elas serão relacionadas à produtividade?

- Ausência de medidas específicas

Existem muitas atividades ligadas ao desenvolvimento que não tem nenhuma medida para serem expressadas, inviabilizando assim qualquer acompanhamento na produtividade das pessoas que se ocupam das mesmas. Por exemplo, como medir as atividades de especificação, gerência ou educação ?

- Código de suporte

Em um projeto, muitas vezes é necessária a criação de código que não é relativo ao projeto em si mas que é indispensável ao seu funcionamento, ou que serve para auxiliar

no seu desenvolvimento. Um exemplo disso são os classificadores, os programas auditores, os otimizadores de desempenho, as ferramentas de apoio ao desenvolvimento. Esses códigos não aumentam as funções entregues ao usuário mas consomem tempo/custo para sua criação. São esses produtos softwares partes do produto final? Seus custos devem ser considerados para medir a produtividade?

- Código de apoio à qualidade

Existem funções que são adicionadas ao software tão só com o objetivo de garantir a qualidade. Algumas tem existência temporária e servem para gerar testes, monitorar execuções dirigidas, etc. Outras tem existência permanente e servem para acompanhar ocorrências, fazer estatísticas relevantes ou detectar falhas potenciais. Como considerar essas funções na hora de medir a produtividade? Seus custos se aditam ao do principal? Seu tamanho deve ser adicionado ao tamanho do software?

- Desvios conceituais

Tem havido desencontro entre o conceito econômico de produtividade e o conceito comum (terminar tão rápido quanto possível). Por essa razão, os estudos sobre produtividade, tem se restringido principalmente ao desenvolvimento de código ao invés de considerar o produto que está sendo entregue em seu todo e sob o ponto de vista do valor efetivo que tem para o usuário, ou seja, das funções que vai executar.

- Indistinção nas atividades de manutenção

Dois tipos de atividades são consideradas sob o título genérico de manutenção: aquelas destinadas a corrigir erros e aquelas destinadas a adicionar novas funções ou modificar funções já existentes. Entretanto, essas atividades são de natureza absolutamente distintas e devem possuir métricas diferentes para sua mensuração.

No caso das atividades destinadas à correção de problemas temos ainda que considerar a natureza do software. Se for por exemplo em software de uso interno, não destinado a comercialização, esse tipo de manutenção é normalmente feito pelo próprio pessoal de desenvolvimento à medida que as necessidades surgem e, como afirma JONES(9), dificilmente ultrapassa 10% dos custos de desenvolvimento.

Entretanto, quando o software é destinado a comercialização muda a natureza do problema e outros fatores precisam ser considerados. Para software que tem suporte na sua instalação, feita por pessoal especializado, e onde esses custos são considerados a título de manutenção, dependendo do número de localidades onde serão instalados, os custos podem ser maiores que todos os custos de desenvolvimento somados.

Após a instalação do software, outras questões surgem e apresentam características diferentes. Como será feita a correção de problemas que surgirem? Através de técnicos de campo que visitarão cada instalação com problema ou através de

uma central de manutenção que solucione as ocorrências por carta ou telefone ? No primeiro caso, e dependendo do número de instalações, os custos podem chegar a ser muito maiores do que os de desenvolvimento. No segundo caso, não serão mais que 25% dos custos de desenvolvimento.

No estudo de JONES(9) é apresentada a tabela abaixo que mostra quão diferentes são os custos em um ano de manutenção, dependendo da forma como a manutenção é feita.

ATIVIDADES	ESFORÇO EM HOMENS - MÊS			
	ÚNICO USUÁRIO	DEZ USUÁRIOS	CEM USUÁRIOS	MIL USUÁRIOS
DESENVOLVIMENTO	600	600	600	600
SUPORTE DE INSTALAÇÃO	1	10	50	250
CENTRAL DE MANUTENÇÃO	2	5	10	50
TÉCNICOS DE CAMPO	2	20	75	700
TOTAL	605	635	735	1600

Figura 3 : Esforço de manutenção como função do número de usuários de um software.

Fonte : JONES(9)

Essas questões não têm sido devidamente tratadas na literatura e por isso, afirmações do tipo " custo de manutenção é igual a 70% do custo total do ciclo de vida " soam como inexatas e sem base na realidade.

Um outro problema relacionado à manutenção é que, a adição de novas funções tem sido tratada indistintamente

de desenvolvimento novo. Com exceção de poucos estudos, incluindo BELADY e LEHMAN(6), a maioria dos estudos sobre produtividade não estabelece essa diferença, o que certamente existe de forma significativa, pois adicionar funções a produtos de softwares já existentes, custa sempre mais que desenvolver as mesmas funções como software novo. Pela natureza e pelas características distintas que essas atividades assumem, elas não podem ser comparadas para efeito de estimativas ou produtividade.

Das principais atividades ligadas ao desenvolvimento é certo afirmar que, quando comparamos a adição de funções com desenvolvimento novo, especificação e projeto são afetadas moderadamente, enquanto codificação, integração, teste e conserto de erros são severamente afetadas para pior. Documentação é afetada normalmente, sendo que a qualidade e produtividade diminuem, enquanto o tempo e esforço aumentam em torno de 34% e 36% respectivamente, segundo JONES(9).

III.2 - LINHAS DE CÓDIGO FONTE (LCF)

III.2.1 - HISTÓRICO

Linhas de código fonte (LCF) é, atualmente, a métrica mais usada para medir o tamanho de software. Seu uso generalizado é devido ao início da computação quando cada linha codificada em Assembler representava realmente uma instrução executável em código objeto. Havia portanto uma relação direta entre as LCFs que o software possuia e a grandeza que ele assumia

quando visto como objeto executável.

O primeiro problema surgiu quando notou-se que as instruções-fonte ocupavam apenas uma pequena parte das 80 colunas do cartão no qual elas eram introduzidas para a conversão em código-objeto. Tentou-se, então, aproveitar o resto do cartão com tantas instruções-fonte quantas fossem possíveis separando-as por delimitadores que indicavam ao compilador tratar-se de outra instrução. Com isso, adveio a primeira questão: o que considerar, as linhas físicas do software ou as instruções lógicas?

O segundo problema surgiu, quando notou-se que instruções repetitivas para executar determinadas funções, podiam ser codificadas previamente sob um determinado título e, quando necessário, ao invés de codificar individualmente cada instrução, usar-se-ia essa macro-instrução. Foi o advento do Macro-Assembler. Agora, ao primeiro problema, aditou-se esse outro: como considerar produtos de software que executando as mesmas funções tem seus programas utilizando macros em um, e em outro utilizando codificação livre?

Seguindo o raciocínio motivador das macro-instruções surgiram as linguagens de alto nível, cuja principal diferença para o Assembler, era o poder de suas instruções, no sentido de que cada uma de suas instruções-fonte podiam gerar muitas instruções executáveis.

Por causa do direcionamento para àreas

específicas, essas linguagens foram diferenciando-se umas das outras, tanto em forma como em potencial, tornando-se agora inviável a comparação entre softwares escritos em linguagens distintas. As LCFs, produzidas em linguagens distintas, não possuem mais equivalência conceitual e, por isso, perderam confiabilidade quando usadas para comparar grandeza de softwares.

Esse deterioramento das LCFs como métrica para software tende a acentuar-se á medida que surgem novas linguagens, cada dia mais potentes e mais distanciadas da idéia original que introduziu LCF, como unidade de medida. Tão amplo tem sido esse alargamento que as linguagens não são mais analisadas isoladamente mas agrupadas por categorias que, segundo JONES(5), chegam a atingir até 10 categorias diferentes, nas quais o conceito de LCF se apresenta, às vezes, completamente ambíguo, outras vezes, simplesmente inaplicável.

Além do acima considerado, descreveremos a seguir outros problemas que comprometem ainda mais a métrica LCF como determinante da grandeza de um software.

III.2.2 - EFEITO PARADOXAL

Desde o início das atividades de programação se ouve que, aumentar produtividade significa aumentar a habilidade em escrever linhas de código. Também é tido como certo que, quanto mais alto é o nivel da linguagem, maior é o aumento da produtividade. Entretanto, quando é tomada a primeira assertiva para medir a produtividade sugerida na segunda assertiva chega-se

a uma conclusão paradoxal, levando a deduções em desacordo com a realidade e ofuscando resultados significativos. Na tabela abaixo apresentada por JONES(9), que compara um software com as mesmas funções escrito em linguagens diferentes, destacam-se os seguintes efeitos desse paradoxo:

- enquanto custo total (fator 4) diminui, sugerindo o aumento da produtividade, LCF por homem/mês (fator 5) diminui sugerindo decréscimo de produtividade ;
- da mesma forma o custo unitário de LCF (fator 6) aumenta, sugerindo um decréscimo de produtividade.

FATORES	SOFTWARE-1	SOFTWARE-2	SOFTWARE-3
1-LINGUAGEM	ASSEMBLER	PL/I	APL
2-TAMANHO EM LFC	100.000	25.000	10.000
3-ESFORÇO (HOMENS/MÊS) DESENV.	200	100	80
4-CUSTO TOTAL DESENV.	$1.000.000	250.000	400.000
5-LCF POR HOMEM/MÊS	500	250	125
6-CUSTO POR LCF	$10	$20	$40

Figura 4 : O paradoxo da LCF como indicador de produtividade

Fonte : JONES (9)

Esse mesmo efeito também ofusca a medição da qualidade quando a unidade de medida é LCF. Na tabela abaixo, mostrando os defeitos esperados nas atividades de desenvolvimento

dos softwares comparados anteriormente, destaca-se o seguinte efeito:

- embora o total de defeitos tenha sido reduzido pela metade, os defeitos p/1.000 LCF aumentaram, sugerindo um decréscimo de qualidade.

FATORES	SOFTWARE-1	SOFTWARE-2	SOFTWARE-3
1-LINGUAGEM	ASSEMBLER	PL/I	APL
2-TAMANHO EM LCF	100.000	25.000	10.000
3-DEFEITOS-ESPECIFICAÇÃO	500	500	500
4-DEFEITOS-PROJETO	1.500	1.500	1.500
5-DEFEITOS-CODIFICAÇÃO	4.000	1.000	500
6-DEFEITOS-DOCUMENTAÇÃO	1.000	1.000	1.000
TOTAL DEFEITOS	7.000	4.000	3.500
7-DEFEITOS P/1.000 LCF	70	160	350

Figura 5 : O paradoxo da LCF como indicador de qualidade

Fonte : JONES(9)

Evitar esses paradoxos, usando o código objeto em vez de LCF, não soluciona a impropriedade dessa medida pois, nesse caso, seriam penalizados os esforços dos compiladores no sentido de gerar códigos mais compactos e mais otimizados.

Dado que produtividade tem sido considerada como aumento de LCF por unidade de tempo, e que o uso de linguagens de alto nível aumenta a produtividade, tem sido assumido erroneamente, que linguagens de alto nível aumentam a produção

de LCF causando com isso, erros de estimativas, ofuscando ganhos de produtividade obtidos com o uso de linguagem de alto nivel e gerando decepção no sentimento e nas expectativas dos que produzem software sem ter em conta esses paradoxos.

III.2.3 - AUSÊNCIA DE PADRONIZAÇÃO NA CONTAGEM

Além dessas considerações que efetivamente geram efeitos paradoxais, existem outras inconsistências no uso de LCF que tem causado distorções intoleráveis no exame da produtividade. Essas inconsistências originam-se na ausência de definições e regras sobre como considerar as LCF. Há muitas formas de considerar as LCFs de um software. São elas:

a) contar somente linhas executáveis;
b) contar linhas executáveis mais definição de dados;
c) contar linhas executáveis,definição de dados e comentários;
d) contar linhas executáveis, definição de dados, comentários e JCLs (Jobs Control Languages);
e) contar como linha cada linha que fisicamente o software possui;
f) contar como linha cada conjunto de caracteres que tenham valor lógico, independente de existirem vários em uma só linha ou um só ocupando várias linhas;
g) em adição de funções a software existente contar somente as linhas adicionadas;
h) em adição de funções a software existente contar as linhas adicionadas e as modificadas;

i) incluir na contagem os códigos reutilizados, tais como, rotinas padrões, pacotes pré-definidos, bibliotecas, etc.

Dependendo da forma utilizada os resultados sobre produtividade poderão assumir valores completamente diferentes e induzir a conclusões ilógicas e sem nenhum sentido, especialmente quando são comparados projetos e organizações diferentes.

Essa distinção no uso não é incomum e está amplamente registrada na literatura corrente. Considerando duas das mais representativas e autorizadas fontes no estudo do assunto temos:

- BOEHM, em (8) utiliza as formas "d" sem comentários, "e" e "h".
- CAPERS JONES, em (9) utiliza as formas "b", "f", "h" e "i".

Ainda outro problema surge, quando LCF é utilizada como medida. Trata-se das instruções-fonte descontinuadas em função de otimização feita para aumentar a velocidade, diminuir a ocupação de memória, etc. Nessa situação, embora o software permaneça com as mesmas funções e aumente o desempenho, a indicação de produtividade irá diminuir.

Mesmo os que acham viável o uso de LCF, como medida, concordam que:

- ela não pode ser usada sem uma clara definição de regras para sua contagem;

- para compensar os paradoxos inerentes é necessário alguma formulação matemática relativamente complexa;
- mesmo com precauções possíveis não serão apresentados resultados exatos e livres de inconsistências.
- para linguagens acima da terceira geração não é possível seu uso,conforme tratado em III.2.1.

Apesar disso, há que se reconhecer, que LCF é a métrica mais utilizada e mais referenciada na literatura.

III.3 - MÉTRICAS DE HALSTEAD

III.3.1 - CARACTERÍSTICAS GERAIS

Em 1972, Maurice Halstead da Universidade de Pardue, iniciou estudo sobre algoritmos que visava testar empiricamente a hipótese de que os operadores (expressões verbais) e operandos (nomes ou expressões de dado) em um programa, deviam relacionar-se com a quantidade de erros no algoritmo.

Dado o sucesso desses estudos, a pesquisa continuou e em 1977 foi publicada um livro (4), que apresentava um sistema de métricas para medição de diversos aspectos de um programa de computador.

Esse sistema consistia na contagem de quatro variáveis básicas das quais eram derivadas uma série de outras medidas com cujo resultado o programa podia ser totalmente analisado.

Essas variáveis básicas, cujas regras de contagem

estão em FITSOS(11), são:

n1 = número de operadores distintos presentes no programa. Por operadores entende-se as expressões que sugerem ação, tais como verbos, comandos, códigos de operação, delimitadores, símbolos aritméticos, pontuação, etc.

n2 = número de operandos distintos presentes no programa. Por operandos entende-se as expressões contendo valores a serem mudados ou que são usados como referência para mudanças. Essas expressões são as constantes e variáveis do programa, entre as quais, nome de campo/variável, literais, argumentos de sub-rotinas/procedures, etc.

N1 = número total de vezes que todos os operadores foram usados no programa. Esse número é a simples somatória das vezes que cada operador foi usado.

N2 = número total de vezes que todos os operandos foram usados no programa. Da mesma forma como para N1, N2 é obtido pela simples somatória das vezes que cada operando foi usado.

Da manipulação dessas variáveis, são extraidas as medidas abaixo:

Vocabulário(n) = soma de todos os operadores e operandos distintos. Trata-se do repertório de elementos manipulado pelo programa.

É calculado pela fórmula:

$$n = n1 + n2 \qquad (a)$$

Extensão(N) = soma de todas as vezes que todos os operadores e operandos foram usados no programa.

É calculada pela fórmula:

$$N = N1 + N2 \qquad (b)$$

As pesquisas de Halstead sugeriram que havia uma relação entre extensão(N) e vocabulário(n) podendo pois, aquela, ser estimada a partir deste. Portanto,

$$\text{Extensão Estimada}(\hat{N}) = n1 \log_2 n1 + n2 \log_2 n2 \qquad (c)$$

Com o objetivo de apresentar medidas que fossem independentes da linguagem Halstead apresentou Volume(V). Com essa dimensão, era apresentada a extensão mínima que programa poderia assumir, nos termos abaixo:

$$V = N \log_2 n \qquad (d)$$

Tentando apresentar o mínimo que esse volume poderia assumir, Halstead determinou Volume Potencial (V^*) nos termos abaixo:

$$\text{Volume Potencial } V^* = (2 + n2^*) \log_2 (2 + n2^*) \qquad (e)$$

sendo $n2^*$ a quantidade de operandos estimada.

Outras medidas são:

$$\text{Nível de Programa } L = \frac{V^*}{V} \qquad (f)$$

$$\text{Dificuldade } (D) = \frac{n1}{2} \frac{N2}{n2} \qquad (g)$$

$$\text{Nível da Linguagem (lambda)} = LV^* \qquad (h)$$

$$\text{Esforço } E = V / L \qquad (i)$$

Além das medidas cujas fórmulas foram anteriormente apresentadas, uma série de outras conclusões são extraidas da análise conjunta desses resultados. CHRISTENSEN, FITSOS e SMITH (10) e ALBRECHT e GAFFNEY(12) apresentam boas discussões sobre o assunto.

III.3.2 - PROBLEMAS RELACIONADOS

Não obstante o mérito de ter sido uma das primeiras tentativas sistematizadas de evitar as inconsistências de LCFs, as métricas de Halstead possuem uma série de problemas que ofuscam seus aparentes méritos. São eles:

- Deduções Baseadas em Resultados Esperados

Quando Halstead mostrou que sua fórmula para Extensão Estimada (c) guardava clara relação com a Extensão

Real (b), então uma série de experiências foram feitas por fontes independentes, entre elas BOHRER(13), ELSHOFF(14), GORDON(15) e HALSTEAD(4) todas confirmando a suposição original. O que não foi considerado, entretanto, é que, a fórmula (c) apresentada como lei natural, não passava de um resultado que, estatisticamente era esperado, exclusivamente por causa da probabilidade incidente. JONES(9) aborda as razões e as origens dessa inconsistência.

- Presunção Científica Indevida

Apesar de ter sido apresentada como " software science " e de afirmar no prefácio que essa " monografia está firmemente baseada em métodos e princípios da ciência experimental clássica ", o que se vê ao longo de seu conteúdo é que:

- baseia-se em variáveis (operadores e operandos) cuja definição é ambígua a ponto de precisar regras, além de abranger uma rica e diversificada gama de construções que não simplesmente operadores e operandos;

- muitas de suas asserções são apenas intuitivas e não baseadas em observações científicas. Aliás, as vezes que as palavras " intuição ", " intuitivamente ", " claramente ", "dedução intuitiva " e similares aparecem no texto, demonstra seu distanciamento da ciência clássica.

- medidas do tipo dificuldade, inteligência, etc são muito

subjetivas e, seus resultados, embora sugiram conceitos interessantes, não foram deduzidos de dados reais. Na realidade, os promissores resultados iniciais, que confirmavam experimentalmente as hipóteses de Halstead, deram uma certa autoridade para essas métricas e ofuscaram a discussão dos resultados restantes.

- a fórmula básica (c) apresenta uma limitação no primeiro termo da direita. Posto que n1 são os operadores, e que estes são as ações/comandos que a linguagem dispõe, então, esse termo n1 é limitado pelo vocabulário da linguagem. Significa dizer que n1 tem um valor máximo em cada linguagem. Como consequência, essa fórmula deveria falhar para programas grandes e muito grandes. Foi exatamente isso que observaram SMITH(17) e FITSOS(16).

- para as linguagens de 4a e 5a gerações e para as chamadas linguagens naturais essas métricas não são válidas.

- Repetição dos Problemas de LCE

Porque baseiam-se na análise do código fonte, as métricas de Halstead, acabam reproduzindo os problemas apontados para LCF e, embora seja afirmado que ela independe da linguagem, isso não se verifica para todas as métricas. Assim, a extensão em uma linguagem vai ser diferente em outra, embora volume deva conservar-se o mesmo.

Um outro problema, é que essas métricas limitam-se a atividade de codificação deixando sem considerar todas as

outras atividades do desenvolvimento.

- Inadequado para Previsão

Além dos resultados não terem o mínimo significado para o usuário, essas métricas só podem ser estimadas com base em uma fase já bem adiantada do desenvolvimento, inviabilizando assim qualquer estimativa prévia na definição de prazos e custos. Além disso elas referem-se a programa exclusivamente e não a sistema ou software.

- Não Trata de Manutenção

As métricas de HALSTEAD não possuem um mínimo de abordagem para as atividades relativas a manutenção. Nem para as adições a software existente nem para modificações e consertos. Se for considerado o fato de que grande parte das atividades em PD, hoje, é dedicada a esse mister, então essa é uma grande falha do trabalho de Halstead.

- Outras Considerações

O trabalho de Halstead tem vários méritos:

- mostrou viabilidade para alternativas diversas das LCFs;
- ensejou um largo trabalho de pesquisa e experimentação;
- demonstrou a necessidade de que as novas linguagens de alto nível deviam dar atenção em compactar definições de dados (operandos), uma vez que estas assumiam proporções grandes

quando comparadas aos componentes de comando/procedimentos (operadores).

CURTIS(18) considera que não há evidências claras quanto a pretensa superioridade das métricas de Halstead sobre LCF e JONES(9) afirma que tal sistema de medidas se tornou um beco tecnológico sem saída, algo parecido com as concepções pré-Newtonianas da gravidade.

III.4 - A COMPLEXIDADE COMO MEDIDA

III.4.1 - DEFININDO COMPLEXIDADE

Há uma corrente de pesquisa que tem se preocupado em determinar a complexidade do software para expressar sua grandeza. Além da questão tamanho, a complexidade também ofereceria base para a medição da qualidade, confiabilidade, previsão de erros, etc.

THOMAS McCABE(19) apresentou uma proposta para medição da complexidade que ficaria conhecida como Complexidade de McCABE ou Complexidade Ciclomática. Nela, o programa seria representado por um grafo que reproduzisse o fluxo de controle incluso no programa. A figura propicia um entendimento melhor da idéia exposta por McCabe.

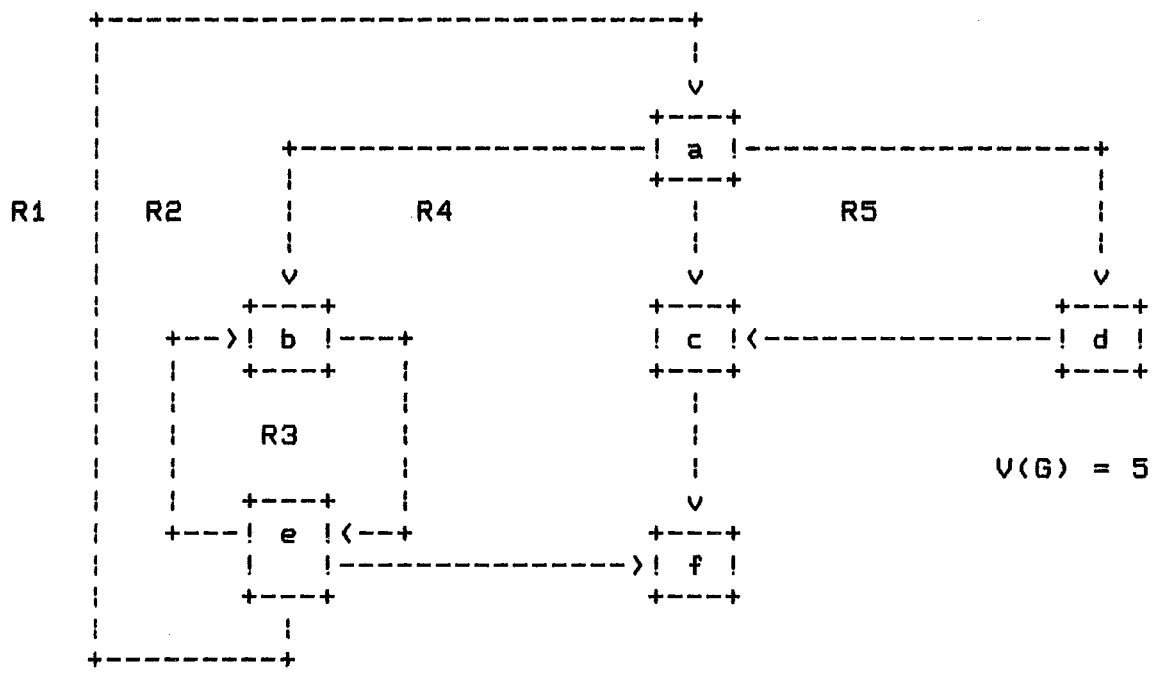


Figura 6 : Um exemplo de grafo da complexidade ciclomática

Fonte : PRESSMAN(20)

Na figura:

- cada letra no círculo representa uma atividade de processamento do programa;

- cada seta ligando um círculo ao outro representa as possíveis ramificações existentes;

- região R1, R2...Rn seria cada área cercada do grafo. O total de regiões seria a soma de todas as áreas cercadas do grafo mais a área externa. A figura considerada tem 5 regiões (R1, R2, R3, R4 e R5), então sua complexidade ciclomática é V(G) = 5 .

McCabe notou uma relação entre o crescimento dessa complexidade ciclomática e a real dificuldade quando, na

prática, era considerado o software ou partes dele. Estabeleceu então um modelo de referência para a comparação da complexidade:

- módulos ou programas com complexidade até cinco são considerados simples;

- módulos ou programas com complexidade de 5 a 10 são considederados bem estruturados e estáveis. Para módulo, 10 seria o limite máximo, a partir do qual, é considerado extremamente difícil seu teste/verificação;

- programas com complexidade entre 20 e 50 são considerados, em princípio, complexos, além de serem propensos à subjetividade;

- programas com complexidade maior que 50 são, normalmente problemáticos e propensos a erros.

Estudos e experiências posteriores sugeriram ser correta a conclusão de McCabe, ou seja: a medida que as ramificações de um programa ou módulo aumentam, a complexidade também aumenta e, por consequência, os erros e falhas também crescem.

Segundo CURTIS(18), essa medida tem se revelado uma das mais bem sucedidas formas de medir um software, entretanto, sua expressão e significado tem se limitado à qualidade, previsão de erros, testabilidade, confiabilidade e similares.

III.4.2 - OS PROBLEMAS DA COMPLEXIDADE

Além dos problemas comuns relacionados a todas as formas de medir e apesar da aceitação e sucesso obtidos, a complexidade ciclomática apresenta alguns problemas sérios ainda não resolvidos. Nesta seção eles serão apresentados:

Potencial de Expressão - Embora adequado para avaliar a qualidade e similares ela não possue expressão adequada para refletir o tamanho do software e, tão pouco tem qualquer significado para o usuário.

Abrangência Restrita - Criada para avaliar módulos e programas essa medida não tem mecanismos de agregação para medição do software inteiro, dai sua abrangência ser restrita, no máximo, ao nível de programa.

Inadequada para Estimativas - Porque baseia-se no programa já pronto, ou no mínimo, em pseudo código já definido, essa medida não oferece base para previsão de custo e tempo de desenvolvimento, uma vez que ela só pode ser obtida em fases adiantadas desse ciclo. Entretanto, para estimar problemas na fase de manutenção, ela se revelou excelente.

Ofusca a Origem da Complexidade - O resultado encontrado não é capaz de indicar se a complexidade é do problema em si ou da solução implementada para resolvê-lo. Não é incomum serem encontrados problemas simples com soluções complicadas, seja pela inexperiência com a linguagem, seja pela inexperiência em solucionar problemas ou seja pelo estilo pessoal de quem propõe a

solução.

Dependência da Experiência Pessoal - O sentido comum que normalmente se da a complexidade, depende muito da experiência de cada um. O que para um noviço é complexo, para um especialista pode não passar de banal. A interpretação, portanto, não tem um significado comum, apesar do modelo referencial proposto por McCabe.

Dependente da Tecnologia - À medida que as linguagens de programação se tornam mais potentes, que os ambientes de banco de dados se tornam mais comuns, que ferramentas de auxílio a codificação se tornam usuais, a complexidade de um software tende a diminuir, pois que boa parte da complexidade inerente a um software antigo pode ser minimizada com esses recursos. Por outro lado, um software implementado sob a ótica dos sistemas especialistas, teria que possuir outra forma de apreciação da sua complexidade, pois que, sua implementação é, basicamente rede de decisões.

III.5 - PONTOS POR FUNÇÃO SEGUNDO ALBRECHT

III.5.1 - SUA NATUREZA E ORIGEM

A tentativa de isolar as funções críticas de um software para tomá-las como medida não é nova. Segundo RUDOLPH (23), desde o início dos anos 70 a IBM, através dos seus Serviços de Processamento de Dados e do grupo de usuários GUIDE, iniciou estudos nesse sentido e de alguma forma passou a utilizar esses

resultados.

Segundo CROSSMAN(21) tentativas concretas e sistemamatizadas foram feitas pelo Chase Manhattan Bank e Standard Bank of South Africa também na década de 70 e, a essa época, possivelmente, muitos membros do GUIDE-IBM já trilhavam o mesmo caminho.

Mais recentemente, JONES(9), abordando considerações de R. Kendall de que com no máximo 50 funções genéricas é possível construir qualquer software e que essas funções poderiam estar embutidas no hardware, afirma que, em termos de medida, o mais importante em um software, são as funções que ele contém e não as LCF pelas quais se expressa.

Nessa linha de raciocínio, ALBRECHT(1), apresentou uma metodologia consistente que tomava por base as funções que o software executava. Essas funções foram determinadas a partir do exame de mais de 400 programas no sentido de isolar os fatores mais críticos da construção de um software. Com o nome de Pontos por Função, essa metodologia se tornou uma das mais utilizadas formas de medir.

Como o objetivo de Albrecht era medir os ganhos de produtividade, sua metodologia ignorava fatores alheios ao software em si. Esses fatores, como linguagem utilizada, experiência da equipe, ferramentas automatizadas, métodos estruturados, etc passaram a ser considerados como elementos que influiam ou não na produtividade, porém não eram parte do

software.

Sua tentativa era determinar o tamanho do software baseado nas funções que ele executava e que eram tornadas disponíveis ao usuário.

III.5.2 - RESUMO DA METODOLOGIA DE ALBRECHT

Em síntese a metodologia de Albrecht compreende as seguintes fases:

a) Tomando a aplicação como um todo, seus elementos principais são classificados em:

- entradas externas - tipos de entradas diferentes provindas do ambiente externo da aplicação;
- saidas externas - tipos de saidas diferentes produzidas pela aplicação;
- consultas externas - tipos de consultas diferentes provindas do ambiente externo da aplicação;
- arquivos-mestre - tipos de arquivos (visão lógica do usuário) diferentes existentes na aplicação;
- interfaces externas - tipos diferentes de interfaces com tais aplicações;

b) Cada um desses elementos é reunido em função da

complexidade e o total de cada um deles é multiplicado por um fator-peso nos termos abaixo:

- entradas externas X 4;
- saidas externas X 5;
- consultas externas X 4;
- arquivos-mestre X 10;
- interfaces externas X 7.

c) A soma resultante da fase anterior é então, ajustada em função do nível de influência (0 a 5) dos fatores abaixo:

- comunicação de dados;
- processamento distribuido;
- objetivos de desempenho;
- restrição de configuração;
- volume de transação;
- entrada de dados "on-line";
- transações interativas "on-line";
- complexidade do processamento;
- projetado para reutilização;
- facilidade de conversão e instalação;
- facilidade de operação;
- instalações em múltiplos locais;
- fácil de modificar e usar.

d) O resultado é então o total de pontos por função da aplicação.

Para atividades de manutenção um outro critério

era seguido e será apresentado quando da descrição detalhada da metodologia.

III.5.3 - ACEITAÇÃO E RESTRIÇÕES

BOEHM(8) considera que, para instalações com certas particularidades, o trabalho de Albrecht é o melhor enfoque surgido até então.

RUDOLPH(24) afirma que ela não cobre todos os tipos de software com igual resultado, e que, os pesos e fatores de ajustes são puramente empíricos e passíveis de discussão, porém é um forte e efetivo instrumento para se medir produtividade, quando realmente há interesse nisso. Nesse caso, com poucos ajustes na versão padrão para o caso específico de cada organização, a metodologia apresentada por Albrecht ainda é, segundo ele, o melhor caminho.

VACCA(25) afirma que não há notícias de experiências com a metodologia para projetos de manutenção, embora ressalte que " Ponto por Função é uma das poucas maneiras de medir que permitem um salto de tecnologia atravessando diversos ambientes de computação diferentes e várias linguagens de 2a, 3a e 4a gerações ".

JONES(9) considera que, embora seja substancialmente incompleto quanto aos aspectos estruturais do software, que tenha suas próprias fontes de ambiguidade quanto às regras de contagem e ajuste, e que não seja ainda

largamente usado para projetos militares, de tempo-real e outros com alta complexidade, a metodologia Pontos por Função, é potencialmente de valor por diversas razões, das quais, a mais importante, é a sua utilidade para fazer estimativas.

RUDOLPH(23) pondera que o ponto fraco dessa medida é a sua relativa subjetividade que deixa espaço para erros ou tendências subjetivas na maneira de apurar seus resultados. Rudolph relata que em experiência realizada pelo grupo de produtividade do GUIDE-IBM, na qual 20 pessoas avaliaram independente e isoladamente um mesmo software, foram observadas diferenças para mais e para menos de até 30% da média geral, embora isso fosse devido a ausência de documentação adequada do software o que motivou interpretações divergentes. Segundo RUDOLPH(23), a média de variação esperada para produtos de software bem documentados deve situar-se no limite de mais ou menos 10% e, para produtos software com pouca documentação, o limite seria de mais ou menos 30%.

Em verdade, Pontos por Função, com pouca ou nenhuma restrição, tem sido largamente utilizada, testada, pesquisada e divulgada.

ALBRECHT(27) informa que entre 1979 e 1985 ele participou, fora da IBM, em mais de 25 grandes conferências sobre a metodologia, envolvendo mais de 400 grandes empresas em todo o mundo. No " GUIDE AND SHARE " da IBM existem projetos para incrementar cada vez mais o seu uso e aprofundar seus conceitos. Pelo menos dois grupos de usuários foram formados para trocar

dados a respeito. Informa ainda Albrecht que, na IBM, essa medida tomou tal dimensão que está se transformando no mais importante instrumento para avaliação de produtividade em desenvolvimento e manutenção de software. Em 1984, por exemplo, foram liberados 241.000 pontos por função para uso interno na IBM como resultado de desenvolvimento e, feita manutenção a 1.115.000 pontos por função já instalados. No primeiro caso houve um esforço de 11.200 meses e no segundo de 6.200.

KUNKLER(26), na fase II do seu estudo sobre produtividade, analisando 87 projetos desenvolvidos entre 1974 e 1983, concluiu que " o uso de pontos por função está se tornando comum tanto para avaliação de produtividade como em ferramentas para realizar estimativas ".

Em (2) há informações de que essa metodologia é usada nos EUA, Canadá, Reino Unido, Austrália e Nova Zelandia por grandes empresas como IBM, Burroughs, Bank of New Scotia, Bank of America, Bell Canada, ITT, John Hancock, Connecticut General Life, London Life e General Motors.

DRUMMOND(28) utilizou Pontos por Função para avaliar produtividade em 31 projetos e considera que ela confirma a credibilidade que lhe tem sido atribuida e que essa técnica representa um significativo avanço nas estratégias de medidas.

Onde LCF é inadequada, JONES(9) utiliza Pontos por Função para mostrar variações de produtividade afirmando, que " claramente, pontos por função é um indicador comparativo de produtividade muito melhor que LCF, quando em ambiente de

múltiplas linguagens ".

SIMPSON e RUDOLPH(30), RUDOLPH(29)(23)(24), CROSSMAN(21), KUNKLER(26), BEHRENS(31), ALBRECHT e GAFFNEY(12), LAMBERT(32) e outros usaram em suas pesquisas e trabalhos a metodologia Pontos por Função como sendo a que mais minimiza os problemas discutidos neste capítulo e a que melhor expressa, em termos conceituais, a grandeza de um software.

III.6 - CARACTERÍSTICAS GERAIS DA MEDIDA IDEAL

Na busca de uma forma ideal para medir software existem pois, várias correntes que exploram campos diferentes.

As que foram tratadas aqui, JONES(9) as resume da seguinte forma:

- aqueles que procuram minimizar os problemas com LCF a fim de torná-la usável de forma padrão, sem ambiguidades e sem paradoxos;

- aqueles que procuram explorar aspectos funcionais do código utilizando operadores, operandos e outras construções para expressar a grandeza do software;

- aqueles que buscam no exame da complexidade estrutural uma forma de dar tamanho ao software;

- aqueles que se encaminham pela exploração das funções que o software executa como forma de expressão da grandeza.

Todas tem problemas e potenciais , sendo que a que

mais minimiza esses problemas é a de Pontos por Função que, por isso mesmo, tem sido largamente aceita e utilizada.

Várias tentativas de resolver a subjetividade de alguns aspectos do trabalho de Albrecht tem sido feitas. Uma das mais expressivas é a que tenta uma união entre o trabalho de Albrecht e o de McCabe, ou seja, pontos por função adicionado da complexidade ciclomática. JONES(9) relata que um dos expoentes nesse caminho é C.Symons com o trabalho, em forma de livro, " Extend Function Points with Entity Type Complexity Rules " editado por Nolam, Norton and Company em Londres - 1984.

De qualquer maneira, do que foi visto até agora, é possível enumerar aquelas que seriam as características principais de uma medida ideal. São elas:

- Generalidade - Deve possibilitar a medição de qualquer tipo de software, ou seja, software básico, tempo-real, comerciais, etc;
- Objetividade - A forma de utilização não deve permitir dúvidas, de modo a possibilitar que pessoas diferentes cheguem a um resultado com não mais de 10% de diferença;
- Facilidade - Seu uso deve ser de tal forma fácil e rápido que impeça bloqueios na sua utilização;
- Significabilidade - A unidade de sua expressão deve ser compreensível e ter significado objetivo ao usuário;
- Imutabilidade - Os resultados devem ser os mesmos em qualquer

tipo de ambiente, com qualquer tecnologia e com a utilização de qualquer linguagem;

.Expressividade - Deve expressar todos os fatores que realmente influenciam no tamanho do software de modo significativo.

.Isolabilidade - Não deve permitir que a expressão do tamanho seja influenciada por fatores que não digam respeito ao software em si, como por exemplo, experiência da equipe, ambiente físico, uso de banco de dados, etc;

.Previsibilidade - Deve possibilitar seu uso como fator de previsão, antecipando, o mais cedo possível, o tamanho do software. Isto significa que a medição deve ser possível em estágios iniciais do ciclo de vida, com margem de erro não superior a 15% em pelo menos 90% das vezes;

.Agregabilidade - Deve possibilitar a medição tanto de programas isoladamente, quanto de sistemas como um todo e, neste caso, com mecanismos de agregação. Também deve ser possível generalizar para a organização;

.Automatização - Deve permitir que sua contagem seja feita automaticamente;

.Suficiência - Deve possuir abordagem clara para atividades de desenvolvimento novo, adição de funções a software existente e atividades de manutenção;

.Cientificidade - Deve apoiar-se em argumentos cientificamente

comprováveis, e não basear-se em simples intuição, opinião ou coisa semelhante.

CAPÍTULO IV

MÉTODO PARA MEDIÇÃO DA PRODUTIVIDADE BASEADO NA METODOLOGIA PONTOS POR FUNÇÃO

Com base na literatura já apresentada e principalmente, considerando (1), (2) e (3), a metodologia Pontos por Função será descrita em detalhes neste capítulo.

Sua apresentação, não obstante a base bibliográfica na qual se apoia, obedecerá aqui, a uma ótica diferente e será acrescida de várias contribuições pessoais.

Na seção IV.1 são repassados os objetivos da metodologia, as razões e finalidades do seu uso e definidos todos os principais conceitos associados a ela.

Na seção IV.2 são examinadas considerações importantes ao entendimento do cálculo e, em seguida, é feito um detalhamento na forma de calcular o Processamento Padrão Associado ou Pontos por Função Brutos, o Processamento Geral Associado e os Pontos por Função Líquidos. Também são examinadas as fórmulas para cálculo do Produto do Trabalho de Desenvolvimento e de Suporte ou Manutenção.

Na seção IV.3 é examinada a determinação do Esforço do Trabalho em duas partes. Uma que trata do registro e outra que trata do cálculo do Esforço do Trabalho.

A última parte deste capítulo (seção IV.4), trata da parte final relativa à determinação da produtividade.

IV.1 - OBJETIVOS, UTILIZAÇÃO E CONCEITOS

IV.1.1 - OBJETIVOS DO METODO

Pontos por Função é um método que tem por objetivo medir o tamanho de um software baseando-se nas funções que ele executa por pedido e de acordo com a visão externa do seu usuário.

Essa medida reflete também a expressão do trabalho feito no desenvolvimento de um software totalmente novo, no desenvolvimento de uma expansão ou nas atividades de manutenção a um software existente. Mais precisamente, ela expressa ou o tamanho de um software existente, ou o produto do trabalho de desenvolvimento (incluindo expansão), ou o produto do trabalho de manutenção ou suporte.

Entretanto, o tamanho somente não tem expressão significativa a menos que seja visto em relação a outra medida ou usado para um objetivo específico. Como a metodologia foi criada tendo em vista a avaliação da produtividade, é correto estabelecer-se que, ao objetivo de calcular o tamanho de software, adita-se o de calcular e avaliar a produtividade na produção manutenção de software.

IV.1.2 - RAZÕES E FINALIDADES DA UTILIZAÇÃO

Pontos por Função é utilizada por atender a maioria das características relacionadas no ítem III.6. Entre as

que lá foram descritas ela atende:

Facilidade - É simples para utilizar e fácil para aprender, requerendo apenas três dias para seu aprendizado e algumas horas para aplicação.

Significabilidade - Tem significado concreto para o usuário podendo ser por ele analisada, discutida e validada sem entretanto lhe exigir conhecimento técnico de computação.

Imutabilidade - É baseada na visão externa do usuário e, por isso, é independente da tecnologia a ser utilizada.

Expressividade - Embora com algumas ausência de aprofundamento ela inclui todos os fatores que influenciam o tamanho do software.

Isolabilidade - Não é influenciada por atributos externos como experiência da equipe, ambiente físico, ferramentas, experiência com a aplicação, etc.

Previsibilidade - Pode ser utilizada no início do ciclo de desenvolvimento, tão logo as especificações estejam concluidas. Dessa forma pode servir para previsão de tempo, custo, planos, orçamentos, etc.

Agregabilidade - Pode ser usada para medir produtividade em programas, projetos e no orgão de computação como um todo.

Suficiência - Considera e distingue o desenvolvimento novo, a expansão e a manutenção.

Cientificidade - Embora tenha sido estabelecida de forma empírica, seus resultados já foram exaustivamentes testados e comprovados.

Pontos por Função ainda não atende, de forma satisfatória, as seguintes características:

.Generalidade- o uso do método tem sido centrado para software com características de "processamento de dados", isto é, softwares comerciais, sistemas de informação, etc.Não há ainda evidências de sua utilidade para outros tipos de software.

.Automatização- as funções e características do software ainda precisam ser deduzidas de forma manual, através de métodos não formais;

.Objetividade - não há consenso generalizado sobre o que considerar na contagem das funções. Este trabalho, por seu aspecto detalhista, diminui essas ambiguidades.

Além do objetivo principal, que é medir o tamanho de um software, Pontos por Função propicia:

- determinar a produtividade no desenvolvimento e manutenção de software;
- determinar tendências de produtividade e identificar os fatores responsáveis;
- avaliar o impacto na produtividade de medidas tomadas no

âmbito da produção ou manutenção de software;

- justificar medidas ou providências que sejam requeridas para aumento da produtividade;
- avaliar conveniências entre expandir ou desenvolver tudo de novo, entre manutenção ou substituição e entre compra ou desenvolvimento interno;
- produzir dados consistentes para previsões, planos, orçamentos e procedimentos de gerência em geral;
- estabelecer, com consistência, os custos de desenvolvimento e manutenção de software.

IV.1.3 - CONCEITOS ASSOCIADOS

Para o entendimento correto dos termos que serão utilizados é importante defini-los. Essa definição, entretanto, não tem caráter abrangente e visa, tão somente, o contexto deste trabalho. São eles:

Software - Conjunto de um ou mais programas executáveis em computador, de procedimentos e de documentação que, relacionados entre si, tornam concreta a expressão de seus objetivos. Um software, dependendo do seu tamanho, pode ser dividido para facilitar seu desenvolvimento e uso, dando origem a outros softwares de menor tamanho. Neste trabalho a nomeclatura adotada para expressar essa divisão é a seguinte:

- Projeto ou Sistema - pode se constituir de uma ou mais aplicações ou sub-sistemas;
- Aplicação ou Sub-sistema - pode se constituir de uma ou mais rotina ou módulos;

- Rotina ou Módulo - pode se constituir de um ou mais programas.

Programa - Menor unidade mensurável de um software. Um software pode possuir um só programa.

Desenvolvimento Novo - Atividades de desenvolvimento que visam produzir um software totalmente novo, ou seja, software que não possua ainda nenhum dos seus componentes desenvolvidos.

Expansão - Diferente do desenvolvimento novo, a expansão pressupõe o desenvolvimento de software para ser aditado a um já existente. Isso significa que partes do software em desenvolvimento já se encontram desenvolvidas. Este termo também poderá ser usado para significar adição de funções novas.

Suporte ou Manutenção - Atividades de desenvolvimento que, sem cobrir o ciclo de desenvolvimento padrão, tem por objeto o conserto de erros, otimização eventual, adaptação a legislação, adequação localizadas, manutenção em geral para que o software continue funcionando conforme o especificado. O trabalho de suporte pode também produzir funções novas, desde que localizadas e sem se caracterizar em expansão conforme já definido.

Produto do Trabalho - É a quantidade de pontos por função gerados nas atividades decorrentes da produção ou modificação de software.

Produto do Trabalho de Desenvolvimento - É a soma total de todos os pontos por função adicionados, modificados ou excluidos por um projeto de desenvolvimento ou expansão.

Neste conceito estão incluidas as atividades decorrentes de um " desenvolvimento novo " quanto as de uma " expansão " tal como anteriormente definidas.

Produto do Trabalho de Suporte ou Manutenção - São os pontos por função que o software tinha originalmente ajustados pelas mudanças decorrentes de expansões ou melhoramentos subsequentes. Essas mudanças são: (1) as mudanças para mais ou menos na complexidade dos pontos por função originais; (2) mais os pontos por função acrescentados, e, (3) menos os pontos por função removidos. Em verdade o produto do trabalho de suporte, indica o tamanho do software após uma modificação ou expansão.

Esforço do Trabalho - É a quantidade total de tempo/homens gastos para produzir um determinado trabalho de desenvolvimento, expansão ou suporte em software. Esse tempo pode ser expresso em diversas unidades como hora/homem, mês/homem, ou ano/homem.

Esforço Bruto do Trabalho - É o esforço no qual, além do tempo gasto diretamente no trabalho, inclui-se também, os tempos improdutivos como férias, feriados, educação geral e outras ausências pessoais. Considerando a semana de 40 horas e um ano de 52 semanas temos as seguintes unidades:

- Ano Bruto de Trabalho : cerca de 2.080 horas (52 semanas X 40 horas) de trabalho de uma pessoa.
- Mes Bruto de Trabalho : cerca de 173 horas (52 semanas/ 12 meses X 40 horas) de trabalho de uma pessoa.
- Hora Bruta de Trabalho : uma hora de trabalho de uma pessoa.

Esforço Líquido do Trabalho - É o esforço no qual, o tempo registrado, foi gasto exclusiva e diretamente no trabalho. Embora podendo variar de uma organização para outra, tem sido aceito como padrão, que o Esforço Líquido representa 75% do Esforço Bruto do Trabalho. Assim:

- Ano Líquido de Trabalho : 2.080 X 75% = 1.565 horas líquidas de trabalho de uma pessoa.

- Mes Líquido de Trabalho : 173 X 75% = 130 horas líquidas de trabalho de uma pessoa.

- Hora Líquida de Trabalho : Cerca de 75% da hora bruta de trabalho de um indivíduo.

A fim de evitar variações sazonais, o tempo gasto no projeto deve ser registrado de forma líquida, tal como aqui descrito. Para apuração do Esforço Bruto basta usar o fator padrão para cada organização que, neste trabalho, foi fixado em 75%.

Produtividade - Tal como definido no capítulo II, é a relação entre o Produto do Trabalho e o Esforço do Trabalho. Quando se quer a ênfase em custo ela significa tambem a relação entre o Custo do Trabalho e o Produto do Trabalho, sendo que neste caso ela é chamada de " custo unitário ". Ambas são expressadas nas fórmulas abaixo:

a) Produtividade no Desenvolvimento = Produto do Trabalho de Desenvolvimento / Esforço Bruto do Trabalho

OBS.: a1) Desde que se está interessado na demora real então, o esforço utilizado deve ser realmente o Esforço Bruto, sendo que este, para o caso de desenvolvimento, deve ser apresentado em meses brutos de trabalho;

a2) O resultado indica a quantidade de pontos por função produzidos por homem/mês-bruto de trabalho. A tendência deverá ser aumentar.

$$\text{b) Custo do Desenvolvimento} = \frac{\text{Custo do Trabalho}}{\text{Produto do Trabalho de Desenvolvimento}}$$

OBS.: b1) Como Custo do Trabalho, deve ser utilizado o total de Horas Brutas de Trabalho. Caso haja valor associado a cada hora bruta então, o resultado pode ser apresentado também em termos monetários;

b2) O resultado indica quanto custa (em termos monetários) ou quantas horas-brutas/homem são necessários para produzir um ponto por função. A tendência deverá ser diminuir.

$$\text{c) Produtividade no Suporte ou Manutenção} = \frac{\text{Produto do Trabalho de Suporte}}{\text{Esforço Bruto do Trabalho}}$$

OBS.: c1) Valem, aqui, as mesmas observações anotadas para Produtividade no Desenvolvimento, sendo que, para

Esforço Bruto, devem ser usadas as Horas Brutas de Trabalho.

Tanto o Produto do Trabalho de Suporte quanto o Esforço Bruto do Trabalho devem ser os ocorridos, para cada projeto, ao longo de um ano.

$$\text{d) Custo Unitário do Suporte ou Manutenção} = \frac{\text{Custo do Trabalho}}{\text{Produto do Trabalho de Suporte}}$$

OBS.: c1) Valem, aqui, as mesmas observações anotadas para Custo Unitário do Desenvolvimento, sendo que, tanto o Custo do Trabalho quanto o Produto do Trabalho devem ser os ocorridos, para cada projeto, ao longo de um ano.

EXEMPLOS:

Para que um resultado de produtividade seja completo e sem ambiguidades ele deve ter uma das seguintes estruturas:

" A produtividade no desenvolvimento é de XX pontos por função por mês-bruto/homem "

ou

" A produtividade na manutencao/suporte é de XX pontos por função por hora-bruta/homem no período de um ano "

onde XX é a quantidade de pontos por função obtida nas relações "a" e "c".

Para o caso de produtividade-custo as estruturas devem ser as abaixo:

" Custa YY horas-brutas/homem para desenvolver um ponto por

função "

ou

" Custa YY horas-brutas/homem para manter um ponto por função, no período de um ano "

onde YY é a quantidade de horas oriundas das relações "b" e "d".

O caso de produtividade-custo pode também ser apresentado em valores monetários, tal como já referido anteriormente.

Fatores do Tamanho - São as variáveis, intrínsecas ao software em si, que influenciam de modo significativo e constante o seu tamanho. Essas variáveis compõem a grandeza do software em si, independente dos atributos externos que condicionam sua produção. Os valores dessas variáveis não mudam quando a forma ou tecnologia associada mudam, daí o software ter um só tamanho em qualquer ambiente. Neste trabalho os fatores do tamanho são valorizados através de pontos dependendo da função associada. Fatores do tamanho são aqueles tratados pela característica " expressividade " descrita anteriormente, na seção III.6.

Fatores de Produtividade - São as variáveis externas que condicionam a producão de software. Chamados por Albrecht(1) de " atributos de produtividade " eles se caracterizam por assumirem expressões diferentes dependendo da tecnologia, do ambiente ou da forma sob a qual o software é produzido. Entre muitos existentes incluem-se, a linguagem utilizada, a experiência da equipe, ambiente "on-line", uso de padrões,

ambiente físico, uso de ferramentas automatizadas, uso de métodos estruturados, etc. Como variam com o ambiente eles não devem influir na determinação do tamanho do software, daí que, um método de cálculo deve possuir a característica " isolabilidade " tal como descrita anteriormente na seção III.6.

Esforço Total - Trata-se da quantidade total de tempo gasta por todos os envolvidos no projeto. Representa o quanto custa de esforço para produzir um software. Nesse esforço estão inclusos todos os tempos empregados no projeto, não importando a fase, a atividade ou a pessoa.

Tempo Total - Representa o tempo no qual o projeto será desenvolvido. Trata-se do período que vai decorrer entre o início e o fim do projeto.

IV.2 - DETERMINAÇÃO DOS PONTOS POR FUNÇÃO

IV.2.1 - CONSIDERAÇÕES IMPORTANTES

Os pontos por função de um software são calculados com base nas especificações e documentação relacionada e decorrem da atribuição de pontos a cada um dos tipos de funções que estão intrinsecamente, associadas a qualquer software. Essas funções podem ser agrupadas em dois grupos distintos:

Funções do Processamento Padrão Associado - São as funções associadas com os principais tipos de dados ou controles que entram e saem de uma aplicação, com os arquivos lógicos permanentes, com os arquivos que servem de interface e com as consultas externas propiciadas. Esses tipos de funções cobrem

a totalidade das manifestações externas de um software e, a cada um, é atribuida uma quantidade de pontos dependendo da complexidade associada. A soma desses pontos expressa o Processamento Padrão Associado ao software ou os Pontos por Função Brutos.

Funções do Processamento Geral Associado - São as funções associadas às características gerais da aplicação que afetam o processamento padrão entregue ao usuário. Essas características, face ao grau de influência de cada uma, são transformadas em um fator de ajuste variando entre +/- 35%, que é então aplicado ao processamento padrão. O resultado dessa operação, expresso em termos de pontos por função, é o tamanho do software ou Produto do Trabalho nos termos da definição já feita anteriormente.

A fórmula geral do tamanho da aplicação é, pois:

Pontos por Função = Pontos por Função Brutos X Fator de Ajuste

Outra consideração importante é com referência aos limites externos da aplicação. É fundamental que esses limites estejam claramente estabelecidos a fim de que a caracterização dos diversos tipos de dados e controles que entram, saem, interfaceiam, permanecem ou são consultados possa ser feita sem dificuldades.

Há projetos grandes que são divididos, gerenciados e desenvolvidos em aplicações separadas. Neste caso, cada aplicação individualmente deve ser considerada como um

software diferente e, portanto, a contagem de pontos deve ser feita considerando todas as entradas, saídas, interfaces e consultas que atravessam seus limites, ainda que se destinem às outras aplicações do projeto.

Há situações em que a divisão é feita apenas para facilitar o desenvolvimento mas, seu gerenciamento é feito como um único projeto o qual é tido e considerado como um software único. Nesse caso estão inclusas as divisões em programas, rotinas, módulos etc, que precisam ser controladas e avaliadas isoladamente, mas que fazem parte de um mesmo todo. Para essa situação cada programa, rotina, módulo etc, deve ter sua contagem como se fosse um software único, contando todas as entradas, saídas, interfaces e consultas que atravessam seus limites. Entretanto, a soma dos resultados dessas sub-partes não deve ser considerada como os pontos por função da aplicação inteira. Uma contagem separada deve ser feita para a aplicação como um todo e, nesse caso, desconsiderando as entradas, saídas, interfaces e consultas que passaram a existir somente por causa da divisão estabelecida.

Há outras situações, ainda, em que a aplicação é dividida apenas para efeito de contagem dos pontos por função. Os limites internos são arbitrários e as entradas, saídas, interfaces e consultas que os atravessam não devem ser contados. Nesse caso, a soma dos resultados das sub-partes representa o total de pontos por função da aplicação.

IV.2.2 - CÁLCULO DO PROCESSAMENTO PADRÃO ASSOCIADO

O primeiro passo desse cálculo e identificar os principais tipos de dados ou controles presentes na aplicação. Esses tipos são:

- Entradas Externas (EE)
- Saidas Externas (SE)
- Arquivos Lógicos Internos (AI)
- Arquivos de Interfaces Externas (IE)
- Consultas Externas (CE)

O segundo passo é classificá-los segundo o grau de complexidade (baixo, médio ou alto) associado a cada um desses tipos.

O terceiro passo é multiplicar a quantidade de cada uma das 15 classificações possíveis por um peso associado a cada uma delas.

O quarto passo é somar todos os produtos obtidos no passo anterior. O resultado, que representa o Processamento Padrão Associado, são os Pontos por Função Brutos, como abaixo:

$$
\begin{aligned}
PFB = {} & (QT - EEcs \times 3) + (QT - EEcm \times 4) + (QT - EEcc \times 6) + \\
& (QT - SEcs \times 4) + (QT - SEcm \times 5) + (QT - SEcc \times 7) + \\
& (QT - AIcs \times 7) + (QT - AIcm \times 10) + (QT - AIcc \times 15) + \\
& (QT - IEcs \times 5) + (QT - IEcm \times 7) + (QT - IEcc \times 10) + \\
& (QT - CEcs \times 3) + (QT - CEcm \times 4) + (QT - CEcc \times 6)
\end{aligned}
$$

onde

PFB = Pontos por Função Brutos

QT-EE = Quantidade de Entradas Externas

QT-SE = Quantidade de Saídas Externas

QT-AI = Quantidade de Arquivos Lógicos Internos

QT-IE = Quantidade de Interfaces Externas

QT-CE = Quantidade de Consultas Externas

cs = Complexidade Simples

cm = Complexidade Média

cc = Complexidade Alta (Complexa)

IV.2.2.1 - DEFINIÇÃO DE ENTRADA EXTERNA (EE)

Entradas Externas são todos os diferentes tipos de entrada de dados e as diferentes inclusões/modificações a arquivos lógicos internos que, provindas diretamente do usuário ou do ambiente externo da aplicação, guardam uma ou várias das seguintes características:

a)possuem formatos diferentes;

b)possuem formatos iguais mas diferentes lógicas de processamento

c)não são dados de consulta;

d)não representam exigência da tecnologia utilizada;

e)provem do ambiente externo da aplicação na forma de arquivo de transações as quais, não foram ainda contadas como entradas, ou então, foram geradas como saída externa de outra aplicação e, além disso, exigem alguma lógica especifica de processamento. Se as transações possuirem formatos diferentes ou, sendo iguais, exigirem lógica específica de processamento, então devem ser contadas como entradas diferentes;

f)são dados ou transações fornecidas por outra aplicação e recebidos automaticamente, ou seja, sem armazenamento intermediário. Se esses dados ou transações possuirem formatos diferentes ou, sendo iguais, exigirem lógica específica de processamento devem ser contados como entradas diferentes;

g)não possuem características de interfaces externas originadas em outra aplicação.

As entradas externas devem ser classificadas em um dos três níveis de complexidade abaixo:

.Simples - Além de poucos ítens ou campos de dados a entrada está relacionada a poucos arquivos lógicos internos. Considerações devidas ao fator humano não afetam a estrutura da entrada.

.Médio - A entrada não está claramente caracterizada nem como simples nem como complexa.

.Complexo - Além de muitos ítens ou campos de dados a entrada está relacionada a muitos arquivos lógicos internos e os fatores humanos foram fortemente considerados no projeto da entrada.

QUANTIDADE DE ARQUIVOS REFERENCIADOS	QUANTIDADE DE CAMPOS		
	1 A 4	5 A 15	16 OU MAIS
0 - 1	SIMPLES	SIMPLES	MÉDIO
2	SIMPLES	MÉDIO	COMPLEXO
3 OU MAIS	MÉDIO	COMPLEXO	COMPLEXO

OBS.: Entradas externas classificadas em um desses níveis podem, ainda, ser ajustadas para mais ou menos um nível segundo um dos seguintes fatores: movimento automático de cursor, conversão de dados, desempenho da aplicação ou outros fatores humanos.

Possíveis Entradas Externas

- Tela
- Cartão
- Documento com reconhecimento magnético (MICR)
- Documento com reconhecimento ótico (OCR)
- Sensor digital
- Sensor analógico
- "Light pen"
- Controle da aplicação do usuário
- Tecla de função
- Listra imantada magneticamente
- Chave de comutação
- Transação em diskette
- Transação em fita magnética
- Transação em fita de papel
- Transação automática

Sugestões Práticas de Contagem

Seguem abaixo sugestões de contagem, extraidas das práticas correntes entre aqueles que usam a metodologia:

Caracterização	Contagem	Observações
.Arquivos de cartões.....	1EE/TIPO	
.Transações em diskette, fita magnética ou fita de papel..	1EE/TIPO	
.Tela de entrada de dados....	1EE	
.Tela de totais de lote.....	1EE	
.Diversas telas diferentes lidas e juntadas para processamento como transação única	1EE	
.Tela de função.....	1EE	
.Tela com múltiplas funções diferentes.....	1EE/FUNÇÃO	
.Dado ou transação fornecido por outra aplicação e recebido automaticamente.....	1EE	
.Entrada p/ atualização gerada por uma consulta.....	1EE,1CE	
.Entrada por "backup" com mesma lógica de processamento da entrada original.....	0EE	NÃO CONSIDERAR
.Tecla de função duplicada de uma tela já contada como Entrada.....	0EE	NÃO CONSIDERAR
."Light Pen" duplicado de uma tela ja contada como Entrada	0EE	NÃO CONSIDERAR
.Duas ou mais telas de entrada com formato e lógica de processamento iguais.....	1EE	

Caracterização	Contagem	Observações
.Duas ou mais telas com mesmo formato mas lógica de processamento diferentes.....	1EE/TELA	
.Duas ou mais telas com formato diferentes mas lógica de processamento iguais	1EE/TELA	
.Tela de menu.....	1EE	
.Tela de menu com capacidade de salva.....	2EE	
.Tela de entrada ADF target..	1EE	AMB. IBM
.Tela utilizada como entrada e saída sem ser consulta....	1TE,1SE	
.Tabela ou arquivo mantido pelo usuário.....	3EE,1AI,1CE,1SE	POTENCIALMENTE
.Formulário de entrada (MICR/OCR).....	1EE	
.Entrada para controle da aplicação.....	1EE	
.Entrada de controle para emissão relatório.....	1EE	
.Componente regular / table LINC.....	3EE,1AI,1CE	AMB. UNISYS
.Evento LINC.....	1EE	AMB. UNISYS

Observações Complementares

. se houver conversão dos dados do usuário em novos formatos

dentro da aplicação, esses arquivos devem também ser considerados como Entradas Externas;

- parâmetros externos fornecidos para programa de classificação ou outro de caracter utilitário não devem ser considerados como de Entradas Externas;
- normalmente, cada Arquivo Lógico Interno tem necessidade de três Entradas Externas (inclusão, alteração e exclusão). Elas porém, devem ser identificadas antes de serem consideradas na contagem. Se não for possível identificá-las, devem ser consideradas e contadas como sendo duas.

IV.2.2.2 - DEFINIÇÃO DE SAÍDA EXTERNA (SE)

Saídas Externas são todos os diferentes tipos de saídas de dados ou de controle que, gerados pelo software e destinados diretamente ao usuário ou a outra aplicação, guardam uma ou várias das seguintes características:

a)possuem formatos diferentes;

b)possuem formatos iguais mas exigem lógicas de processamento diferentes;

c)não são dados de consulta;

d)não representam exigência da tecnologia utilizada;

e)destinam-se ao ambiente externo da aplicação na forma de arquivo de transações, isto e, serão armazenadas em um meio móvel (fita, diskette, etc) para posterior tratamento como Entrada Externa de outra aplicação. Se essas transações possuirem formatos diferentes ou,

sendo iguais, exigirem lógica específica, então devem ser contadas como saídas diferentes;

f)são dados, mensagens ou transações fornecidas automaticamente para outra aplicação, ou seja, sem armazenamento intermediário. Se esses elementos possuirem formatos diferentes ou, sendo iguais, exigirem lógica específica, então devem ser contados como saídas diferentes;

g)não possuem características de Interface Externa a ser utilizada por outra aplicação;

h)em geral são relatórios e mensagens de controle que, gerados pela aplicação, destinam-se ao ambiente externo, mesmo quando gerados como "back up" ou tenham apresentação através de telas;

As saídas externas devem ser classificadas em um dos níveis de complexidade abaixo:

.Simples - Além de conter poucos ítens ou campos de dados, a saída referencia poucos Arquivos Lógicos Internos. No caso de relatório possui no máximo 2 colunas e as transformações de dados são simples. Fatores humanos não foram considerados no projeto da saída.

.Médio - A saída não está claramente considerada nem como simples nem como complexa. No caso de relatório possui várias colunas com sub-totais e muita transformação de dados.

.Complexo - Além de muitos ítens ou campos de dados, a saída

referência muitos arquivos lógicos internos. Fatores humanos foram fortemente considerados no projeto da saída. No caso de relatório possui multiplas e intricadas transformações de dados e considerações de desempenho são significativas.

QUANTIDADE DE ARQUIVOS REFERENCIADOS	QUANTIDADES DE CAMPOS		
	1 A 5	6 A 19	20 OU MAIS
0 - 1	SIMPLES	SIMPLES	MÉDIO
2 A 3	SIMPLES	MÉDIO	COMPLEXO
4 OU MAIS	MÉDIO	COMPLEXO	COMPLEXO

OBS.: Saídas Externas classificadas em um desses níveis podem ainda ser ajustadas para mais ou menos um nível segundo um dos seguintes fatores: complexidade do "lay out", importância do fator humano, quantidade de sub-totais, nível de transformação de dados e relevância do desempenho da aplicação.

Possíveis Saídas Externas

- Relatório em batch
- Relatório em terminal-impressor
- Relatório em tela de terminal
- Transação em fita magnética
- Transação em fita de papel
- Transação em diskette
- Saída em cartão
- Saída em listras magnética

- Transação automática
- Linha digital
- Atuador digital
- Atuador analógico
- Controle da aplicação do usuário
- Fatura
- Etiqueta de material
- Mensagem em tela

Sugestões Práticas de Contagem

Seguem abaixo sugestões de contagem, extraidas das práticas correntes entre aqueles que usam a metodologia:

Caracterização	Contagem	Observações
.Tela de saída de dados.....	1SE	
.Dado ou transação fornecido automaticamente para outra aplicação	1SE	
.Transações geradas em diskette, fita magnética ou fita de papel.....	1SE/TIPO	
.Saída em cartões.....	1SE/TIPO	
.Formato comum de mensagem do operador para uso em múltiplas mensagens similares.....	1SE	
.Formato comum usado em múltiplas mensagens de erro ou de confirmação associadas a 1EE.....	1SE	

Caracterização	Contagem	Observações
.Mensagem individual de erro ou confirmação enviados em formato padrão.....	0SE	NÃO CONSIDERAR
.Relatório impresso ou executado em batch.....	1SE	
.Relatório de erros.....	1SE	
.Relatório em terminal impressor.....	1SE	
.Totais de controle.....	1SE	
.Lista de checagem, de auditoria ou de traces.....	1SE	
.Tabela ou arquivo mantido pelo usuário.....	3EE,1AI,1CE,1SE	POTENCIALMENTE
.Tela de menu.....	1SE	
.Tela de inicialização e/ou finalização.....	1SE	
.Relatório com página-resumo.	2SE	
.Dump.....	0SE	NÃO CONSIDERAR
.Arquivo relatório acessado pelo usuário.....	1SE	
.Relatório com diversas ordenações e mesmo formato..	1SE	
.Varios relatórios com mesmo formato mas lógicas de processamento diferentes....	1SE/RELATÓRIO	

Caracterização	Contagem	Observações
.Vários relatórios com formatos diferentes mas lógicas de processamento iguais.....	1SE/RELATÓRIO	

Observações Complementares

- não devem ser considerados como Saída Externa os arquivos criados por razões técnicas tais como: arquivos de pontos de teste, arquivos intermediários de trabalho, arquivos em fita convertidos para acesso mais rápido, backup de arquivos, etc. Entretanto, se a existência decorre das necessidades do usuário, então devem ser considerados;

- arquivos de relatórios criados em um programa e impressos em outro devem ser contados uma só vez, a menos que seja preciso uma grande integração para processar o arquivo. Neste caso ele deve ser contado tanto como Interface Externa quanto como Saída Externa;

- relatórios gerados em backup e depois descarregado por outro programa devem ser contados uma só vez;

- os arquivos contendo relatórios formatados que são acessados, via on-line, pelo usuário através de utilitário presente no ambiente devem ser contados apenas uma vez. Entretanto, se o acesso on-line é uma função requerida pelo usuário e provida pelo projeto, então devem ser contados tanto como Interface Externa quanto como Saída Externa;

IV.2.2.3 - DEFINIÇÃO DE ARQUIVO LÓGICO INTERNO(AI)

Arquivo Lógico Interno é cada grupo lógico de dados ou de informações de controle que, gerado, usado e mantido pela aplicação, fica acessível ao usuário através de entradas, saídas e consultas externas definidas na aplicação. Caracterizam-se por uma ou várias das características abaixo:

a)representam a visão externa do usuário tal como definido nas especificações;

b)não possuem finalidades exclusivamente internas ao projeto, tais como, arquivos de trabalho, sorts, arquivos temporários ou intermediários, backup, etc;

c)em banco de dados, caracterizam-se por representar a visão lógica do usuário e não a sua implementacão física;

d)não representam exigência da tecnologia utilizada;

e)não tem fins exclusivos de interfacear com outras aplicações.

Os Arquivos Lógicos Internos devem ser classificados em um dos níveis de complexidade abaixo:

.Simples - Além de poucos tipos de registros e de poucos ítens ou campos de dados, o arquivo não requer considerações sobre desepenho e recuperação;

.Médio - O Arquivo Lógico Interno não é claramente, nem simples nem complexo.

.Complexo - Além de muitos tipos de registros e de muitos ítens ou campos de dados, o arquivo requer importantes considerações relativas a desempenho e recuperação.

QUANTIDADE DE TIPOS DE REGISTROS LÓGICOS	QUANTIDADE DE CAMPOS		
	1 A 19	20 A 50	51 OU MAIS
1	SIMPLES	SIMPLES	MÉDIO
2 - 5	SIMPLES	MÉDIO	COMPLEXO
6 OU MAIS	MÉDIO	COMPLEXO	COMPLEXO

OBS.:Arquivos Lógicos Internos classificados em um desses níveis podem ainda ser ajustados para mais ou menos um nível, segundo um dos seguintes fatores: exigência quanto ao desempenho, importância dos critérios de pesquisa e considerações referentes à recuperação e "back up ".

Possíveis Arquivos Lógicos Internos

- Cadastros ou arquivos lógicos internos
- Banco de dados
- Tabela de usuário
- Arquivo ou tabela de mensagens
- Arquivo de controle para processamento em batch por etapa
- Arquivo de consulta do usuário
- Componente LINC (ambiente UNISYS)
- Evento LINC (ambiente UNISYS)

Sugestões Práticas de Contagem

Seguem abaixo sugestões de contagem, extraidas das práticas correntes entre aqueles que usam a metodologia:

Caracterização	Contagem	Observações
.Entidade lógica de dados do ponto de vista do usuário...	1AI	AMBIENTE BD
.Arquivo lógico interno gerado e/ou mantido pela aplicação.....	1AI	
.Arquivos acessíveis ao usuário através de palavras chaves ou parâmetros.....	1AI	
.Tabela ou arquivo mantido pelo usuário.....	3EE,1AI,1CE,1SE	POTENCIALMENTE
.Arquivo de controle para processamento batch por etapa.....	1AI	
.Cada tipo de percurso (caminho) hierárquico em BD, derivado da necessidade do usuário (inclusive caminho formado por indice secundário ou relação lógica).....	1AI	AMBIENTE BD
.Trajetórias ou caminhos, hierárquicos ou não, não solicitadas pelo usuário....	0AI	
.Arquivos de trabalho		

Caracterização	Contagem	Observações
intermediário ou de sort....	0AI	
.Cada evento de arquivo de eventos-LINC.....	1AI	AMB. UNISYS
.Primeiro perfil de um evento-LINC.....	0AI	AMB. UNISYS
.Cada outro perfil (não primeiro) de evento-LINC.....	1AI	AMB. UNISYS
.Cada componente-LINC.....	1AI	AMB. UNISYS
.Cada perfil de um componente-LINC.....	1AI	AMB. UNISYS
.Arquivo concatenado, de extensão ou complemento, sem acesso particular ou cujo arquivo principal já tenha sido do contado.....	0AI	

Observações Complementares

- arquivos do tipo sistema não devem ser considerados para contagem. Entre eles destacamos biblioteca de programa, biblioteca de procedimentos, manuais e documentação, etc. Entretanto, se o usuário solicitou, por exemplo, documentação e manuais on-line, então as funções daí decorrentes devem ser consideradas;
- arquivos que possuirem registros diferentes, os quais requerem lógicas distintas de processamento devem ser contados um para cada tipo de registro. Considera-se, portanto, a visão lógica que o usuário tem e não como está implementado fisi-

camente o arquivo. Por exemplo, um arquivo-tabelas que contenha 10 tipos diferentes de registro-tabela deverá ser considerado como 10 Arquivos Lógicos Internos;

- quando um software possui, por necessidade do usuário, um arquivo diário e um arquivo mensal com a consolidação de todos os diários, então, os dois arquivos devem ser considerados como arquivos diferentes, independente de terem ou não a mesma estrutura;
- normalmente, cada Arquivo Lógico Interno tem pelo menos uma Saída Externa ou uma Consulta Externa. Deve ser assumida uma Saída Externa quando não houver indicação alguma.

IV.2.2.4 - DEFINIÇÃO DE INTERFACE EXTERNA (IE)

São todos os arquivos ou dados de controle que são passados ou recebidos pela aplicação e/ou que são compartilhados entre a aplicação em questão e outras aplicações. Caracterizam-se por uma ou várias das caracteristicas abaixo:

a) não são arquivos de transações, isto é , não serão ou não virão armazenados em meio móvel (fita, diskette) ainda dependente de tratamento como entrada externa;

b) não representam exigência da tecnologia utilizada;

c) possuem formatos diferentes;

d) possuem formatos iguais mas diferente lógicas de processamento.

e) não possuem características de Entrada Externa ou Saida Externa;

f) são arquivos de registros (não transações) ou dados de controle que são gerados com o fim claro de ser

enviado a outra aplicação, ou que é pressuposto estar sendo recebido de outra aplicação. Se esses registros ou dados de controle possuirem formatos diferentes ou, sendo iguais, exigirem lógica específica de processamento devem ser contados como interfaces diferentes;

g) são arquivos que são gerados, usados e/ou mantidos pela aplicação mas que também são compartilhados ou usados por outras aplicações. Nesse caso inclusive, o arquivo deve ser considerado tanto Arquivo Lógico Interno como também Arquivo de Interface Externa;

h) são interfaces inteligíveis pela máquina sem caracteristicas de transações ou, no caso de transações, exigem conversão de dados.

Os Arquivos de Interface Externa devem ainda ser classificados segundo a complexidade com a qual se apresentam. As definições para essa classificação são as mesmas usadas para Arquivo Lógico Interno.

A única diferença é que, Arquivos de Interface Externa classificados em um daqueles níveis de complexidade, podem ainda ser ajustados para mais ou menos um nível, segundo um dos seguintes fatores: exigência quanto a performance, importância dos critérios de pesquisa, considerações referente a recuperação/backup, grau de conversão de dados e quantidade de aplicações que usarão a interface.

Possíveis Arquivos de Interface Externa

. Arquivo Lógico Interno de outra aplicação que possa ser

acessado

. Banco de dados compartilhado

. Lista de parâmetros compartilhada

. Arquivo Lógico Interno da aplicação que possa ser acessado por outra

Sugestões Práticas de Contagem

Seguem abaixo sugestões práticas de contagem extraidas das práticas correntes entre aqueles que usam a metodologia:

Caracterização	Contagem	Observações
.Arquivo de registro de outra aplicação.....	1IE	POR TIPO REGISTRO
.Arquivo de registros para outra aplicação mesmo sendo AI nesta aplicação.....	1IE	POR TIPO REGISTRO
.Arquivo de registros para várias outras aplicações mesmo sendo AI nesta (distribuição múltipla pode aumentar complexidade).....	1IE	POR TIPO REGISTRO
.Banco de dados compartilhado para e/ou de outras aplicações.....	1IE	
.Arquivo de dados compartilhado com outras aplicações com ou sem conversão de dados.....	1EE,1IE	
.Lista de parâmetros passado de/para outras aplicações	1IE	POR TIPO PARÂMETRO

Observações Complementares

. Arquivos criados em um processo e transferidos para outro devem ser contados como interfaces externas se isso foi solicitado pelo usuário. Caso não tenha sido, mas decorre da estrutura do software, então não deve ser contado, independente de ter sido ou não removido.

IV.2.2.5 - DEFINIÇÃO DE CONSULTA EXTERNA (CE)

Consultas Externas são todas as combinações de entrada/saida, nas quais, a ocorrência de uma entrada causa e gera a imediata ocorrência de uma saida. Caracterizam-se por uma ou várias das características abaixo:

a)possuem formatos diferentes;

b)possuem formatos iguais mas diferentes lógicas de processamento;

c)não representam exigência da tecnologia utilizada;

d)são consultas originadas tanto no usuário quanto em outras aplicações;

f)a parte de entrada da consulta serve somente para direcionar a pesquisa sem ocorrência de atualização em Arquivos Lógicos Internos;

f)não guarda característica de software de consulta, ou seja, é apenas a busca direta de dados específicos, geralmente com a utilização de uma única chave. Um software de consulta (QBE, SQL, DMS/INQUIRY), ao contrário, permite qua uma estrutura organizada com entradas, saídas e consultas realize várias consultas com a utilização de muitas chaves e operações. Essas

entradas, saidas e consultas externas, organizadas sob a ótica de um software de consulta, devem ser consideradas quando da medição desse software.

As Consultas Externas devem ser classificadas em um dos três níveis de complexidade, como segue:

. Classificar a parte de entrada da Consulta Externa usando as mesmas definições utilizadas para classificar Entradas Externas.
. Classificar a parte de saida da Consulta Externa usando as mesmas definições utilizadas para classificar as Saidas Externas.
. O nível de complexidade da Consulta Externa será o maior entre a parte de entrada e parte de saida. Significa dizer que a Consulta Externa possuirá um nível de complexidade igual ao da parte mais complexa.

Possíveis Consultas Externas

. Consulta do usuário sem atualização de arquivo
. Mensagem e tela de ajuda (HELP)
. Menu ADF (IBM)
. Tela ADF de seleção (IBM)
. Tela ADF de regras (IBM)
. Tela de menu (seleção)
. Consulta componente-LINC (REC/INQ)

Sugestões Práticas de Contagem

Seguem abaixo sugestões práticas de contagem extraidas das práticas correntes entre aquêles que usam a metodologia:

Caracterização	Contagem	Observações
.Consulta com entrada e saída on-line sem atualizar arquivos.....	1CE	
.Consulta seguida entrada p/ atualização.....	1CE,1EE	
.Tela de ajuda (HELP).....	1CE	
.Tela de menu de entrada e saída.....	1CE	
.INQ ou REC a componente-LINC	1CE	AMB. UNISYS
.Menu ADF (entrada/saída)....	1CE	AMB. IBM
.Tela ADF de seleção (entrada/ saída).....	1CE	AMB. IBM
.Tela ADF de regras (entrada/ saída).....	1CE	AMB. IBM

OBS.:Um software ou linguagem de consulta deve ser decomposto em sua estrutura hierárquica de EE(s), SE(s) e CE(s) utilizando-se as definições existentes e as práticas correntes.

Observações Complementares

- normalmente, cada Arquivo Lógico Interno, é potencialmente candidato a possuir uma Consulta Externa. Entretanto, ela só deve ser considerada se for identificada na documentação do software;
- na manutenção (alteração) "on-line" de arquivo é comum que antes se chame o dado ou registro na tela para em seguida altera-lo. Nesse caso, a chamada para tela deve ser

considerada como uma Consulta Externa.

IV.2.2.6 - INDICAÇÕES GERAIS ADICIONAIS

Além das definições, conceitos e práticas correntes já estabelecidas anteriormente, serão apresentadas nesta seção, indicações adicionais que auxiliam o uso objetivo da metodologia. São elas:

a)não importa se o código que implementa uma determinada função foi desenvolvido para o software, se foi reutilizado de outra aplicação ou se foi comprado como pacote. Desde que ele esteja atendendo às necessidades e requisições do usuário, tal como definido nas especificações, então ele deve ter seus pontos por função contados como pertencentes a aplicação;

b)não devem ser considerados como pertencentes a aplicação (portanto não devem ser contadas) as funções providas por softwares básicos já presentes no ambiente. Entre esses softwares destacamos: compiladores de tela, sistemas gerenciadores de banco de dados, softwares de consulta, otimizadores de desempenho, controladores de seguranca, registradores estatísticos, monitores de execução, etc. Entretanto, se as funções executadas por um desses softwares, foram claramente solicitadas pelo usuário ou se foram desenvolvidas para atender a pedido do usuário, então devem também ser

contadas como pontos por função da aplicação;

c)o software deve ser considerado sob o ponto de vista do que ele efetivamente faz, tal como solicitado pelo usuário. Não devem ser consideradas suas capacidades potenciais ou possibilidades disponíveis, ainda que estas sejam providas automaticamente. Entretanto, se essas características foram solicitadas, então devem ser consideradas;

d)ao medir um software, devem ser considerados todos os seus usuários, uma vez que, a cada tipo de usuário devem ser providas funções específicas, tais como:

- funções do usuário final (entrada de dados, consulta, etc).
- funções do usuário convertedor e instalador da aplicação (auditoria de arquivos, lista de discrepância entre arquivos, etc).
- funções do usuário da operação (recuperação, totais de controle, etc).

Desde que essas funções devem ser providas pelo projeto por necessidade do usuário, então devem ser consideradas na medição do software.

É importante notar que algumas funções de conversão e instalação nao precisarão mais ser mantidas após a implantação. Neste caso elas nao devem ser consideradas na medição do produto do tabalho de suporte;

e)arquivos, de qualquer caracterização, que forem classificados em uma ou mais sequências diferentes,

devem ser considerados para contagem uma só vez;

f)sempre que possível é conveniente utilizar dois indivíduos diferentes para medir a aplicação, sendo que um deles deve estar familiarizado, com a aplicação;

g)se houver dúvida quanto a classificação da complexidade associada considere a média, mas não deixe de identificar a função, pois a identificação é mais significativa, tem mais expressão do que a classificação;

h)no estágio inicial do desenvolvimento é normal que ocorra a omissão de funções, sendo essa a principal causa de erros. Portanto, não deve ser pressuposto que a documentação exibida está completa, ao contrário, é necessário procurar identificar mais funções.

Porém, só devem ser consideradas aquelas solicitadas pelo usuário ou pelas quais ele está disposto a assumir os custos decorrentes;

i)se funções fundamentais a confiabilidade do software (cópia ou recuperação de arquivo, reinício, etc) não foram clara e conscientemente recusadas pelo usuário, elas devem ser aditadas ao software e, portanto, consideradas na medição;

j)uma estratégia corrente para medição de softwares já implementados é identificar as diversas funções na seguinte ordem:

- Arquivos Lógicos Internos

- Interfaces Externas
- Entradas Externas
- Saídas Externas
- Consultas Externas;

k)em software planejado, no qual as especificações foram feitas mediante alguma metodologia sistematizada, uma boa estratégia seria:

- identificar todas as Saidas e Consultas Externas que o software deve fornecer. Identificar também as entradas que requisitam as saídas identificadas;
- identificar todos Arquivos Lógicos Internos e as Interfaces Externas necessárias para as funções identificadas na etapa anterior;
- identificar as Entradas Externas que são necessárias para manutenção dos arquivos identificados na etapa anterior;
- assegurar que existe pelo menos uma Saída ou uma Consulta Externa para cada arquivo identificado na segunda etapa.

IV.2.3 - CÁLCULO DO PROCESSAMENTO GERAL ASSOCIADO

IV.2.3.1 - CONSIDERAÇÕES PRELIMINARES

Uma vez calculado o Processamento Padrão ou Pontos por Função Brutos é necessário calcular o nível do Processamento Geral Associado ou Fator de Ajuste. Esse Fator de Ajuste vai

refletir o nível de influência que as características gerais da aplicação exercem sobre o processamento padrão. Esse fator produz um ajuste entre +/- 35% dependendo do grau de influência de cada uma das características. Sua fórmula geral é:

```
+-----------------------------------------+
!                                         !
!  FATOR DE AJUSTE = 0,65 +(0,01 X SGI)   !
!                                         !
+-----------------------------------------+
```

onde

SGI é a soma dos graus de influência de todas as características gerais da aplicação.

Assim, para concluir o cálculo do tamanho do software são necessários ainda os seguintes passos:

- estimar para cada uma das 14 características gerais um grau de influência que, à vista do usuário, elas exercem sobre o processamento padrão. Os possíveis graus de influências são:

  . Característica ausente ou sem influência = 0
  . Influência insignificante = 1
  . Influência moderada = 2
  . Influência média = 3
  . Influência significante = 4
  . Forte e generalizada influência = 5

  O nível ou grau de influência dependerá também do quanto, em termos percentuais, a aplicação necessita daquela característica ou é por ela afetada. Como abaixo:

0 = 0%

1 = 1 - 20%

2 = 21 - 40%

3 = 41 - 60%

4 = 61 - 80%

5 = 81 - 100%

As características gerais referidas são:

. Comunicação de Dados

. Funções Distribuidas

. Objetivos de desempenho

. Carga da Configuração

. Volume de Transações

. Entrada de Dados on-line

. Eficiência Usuário Final

. Atualizações on-line

. Processamento Complexo

. Reutilizabilidade

. Facilidade de Instalação

. Facilidade de Operação

. Multiplicidade de Locais

. Facilidade de Mudança

- Os 14 graus estimados devem ser somados, e o total usado para substituir o SGI da fórmula acima. A aplicação na fórmula acima resultará em um fator de ajuste entre 0,65 e 1,35.

OBS.:Se, após o exame das características gerais, houver dúvida quanto ao nível de influência, o fator deve ser assumido como:

1,00 para aplicações on-line

0,80 para aplicações batch.

- O Fator de Ajuste deve então ser multiplicado pelo Processamento Padrão Associado ou Pontos por Função Brutos e o resultado será o tamanho do software em termos de pontos por função.

IV.2.3.2 - DEFINIÇÃO DAS CARACTERÍSTICAS GERAIS

Uma característica só estará qualificada para entrar no cálculo do nível de influência quando afeta o projeto, a implementação e o suporte/manutenção de uma aplicação. Ela deve ter sido especificada direta ou indiretamente na documentação relacionada. Caso contrário, mesmo que se torne útil posteriormente, ela deve ter nível de influência zero.

As 14 características gerais são:

Comunicação de Dados

Os dados e informaçõe de controle são recebidos e/ou enviados mediante o uso de recursos de comunicação. Terminais locais, conectados diretamente a unidade de controle, costumam utilizar esses recursos. Os possíveis níveis de influência são:

0 = aplicação exclusivamente em batch;

1-2 = somente impressão e/ou entrada de dados remota;

3-5 = teleprocessamento interativo (TP);

3 = teleprocessamento front-end para processo batch;

5 = aplicação dominantemente TP interativo.

Funções Distribuidas

O processamento dos dados ou das funções são feitos de forma distribuida em um dos possíveis níveis abaixo:

0 = aplicação não enseja transferência de dados ou processamento de funções entre componentes do sistema;

1 = aplicação prepara dados para processamento pelo usuário em outro componente do sistema;

2-4 = dados são preparados para transferência, são transferidos, e processado noutro componente do sistema;

5 = processamento de funções feito dinamicamente no mais apropriado componente do sistema.

Objetivos de Desempenho

Os objetivos de performance, nas respostas e no processamento, que foram estabelecidos e aprovados pelo usuário, influenciaram o projeto, o desenvolvimento, a instalação e o suporte/manutenção em um dos possíveis níveis abaixo:

0-3 = aplicação sem requisitos especiais de desempenho. Considerações padrões sobre desempenho;

4 = desempenho estabelecida pelo usuário é rigorosa

o suficiente para requerer análise na fase de projeto;

5 = desempenho estabelecida é rigorosa o suficiente para, em adição, requerer uso de ferramentas de análise de performance no projeto, desenvolvimento, e/ou instalação.

Carga da Configuração

A configuração a ser usada operacionalmente está sobrecarregada ou é intensamente usada, requerendo, por isso, considerações especiais no projeto. Mais precisamente, o usuário deseja operar sua aplicação em equipamento que está ou ficará sobrecarregado. Os possíveis graus são:

0-3 = operação típica em máquina sem restrição de uso devido a carga;

4 = restrições de operação requerem especial constrangimento sobre a aplicação no processador central;

5 = além de haver constrangimento no processador central também existe nos componentes distribuidos do sistema.

Volume de Transações

O volume de transações é alto e influenciou o projeto, o desenvolvimento, a instalação, e o suporte/manutenção. Os possíveis níveis são:

0-3 = volume padrão sem necessitar de considerações

sobre análise de desempenho;

4 = o volume de transações ou de serviços é alto o suficiente para requerer análise de desempenho na fase projeto;

5 = em adição ao requerido no grau anterior, há também, necessidade de utilizar ferramentas de análise de desempenho no projeto, desenvolvimento, e/ou instalação.

Entrada de Dados On-Line

A entrada de dados e de funções de controle, são providas de forma on-line, em um dos possíveis níveis abaixo:

0-2 = nenhum ou no máximo 15% das transações são entradas interativas;

3-4 = entre 15 e 30% das transações são entradas interativas;

5 = entre 30 e 50% das transações são entradas interativas.

Eficiência Usuário Final

As funções on-line providas pela aplicação enfatizam a eficiência do usuário final em um dos possíveis níveis abaixo:

0-3 = nada foi estabelecido referente a eficiência do usuário final

4 = requisitos de eficiência estabelecidos são

fortes o suficiente para requerer inclusão de fatores humanos no projeto;

5 = requisitos de eficiência estabelecidos são fortes o suficiente para requerer o uso de ferramentas especiais tal como " prototipação ".

Atualização On-Line

A aplicação provê atualização on-line nos arquivos lógicos internos, em um dos possíveis níveis abaixo:

0 = nenhuma;

1-2 = além de atualização apenas em arquivos de controle, o volume atualizado é baixo e fácil de recuperar;

3 = atualização on-line nos principais arquivos lógicos;

4 = em adição ao nível anterior, a proteção contra perda de dados e essencial;

5 = em adição aos dois níveis anteriores, os altos volumes forçam a análize de custo nas considerações sobre recuperação.

Processamento Complexo

A aplicação caracteriza-se pela existência de uma ou várias das características abaixo:

- controle sensitivo e/ou processamento de segurança
- processamento matemático e lógico extensivos
- muito processamento de exceção, muitas transações

incompletas, e muito reprocessamento de transações, em um dos possíveis níveis abaixo:

0 = nenhuma das características acima
1-3 = qualquer uma das acima;
4 = qualquer duas das acimas;
5 = todas as características acima.

Reutilizabilidade

A aplicação e o código que a expressa, foi especificamente projetado, desenvolvido e mantido para ser reutilizado em outras aplicações. Os possíveis níveis são:

0 = no máximo 10% da aplicação destina-se a reutilização;
1 = entre 11 e 20% da aplicação destinam-se a reutilização;
2 = entre 21 e 30% destinam-se a reutilização;
3 = entre 31 e 40% destinam-se a reutilização;
4 = entre 41 e 50% destinam-se a reutilização;
5 = acima de 50% da aplicação destina-se a reutilização.

Facilidade de Instalação

A facilidade de instalação e conversão são características da aplicação. Planos de conversão e instalação e/ou ferramentas de conversão foram providas e testadas durante a fase de teste do sistema. Os possíveis níveis são:

0-1 = nenhuma consideração especial foi estabelecida pelo usuario referente a instalação e conversão

2-3 = além de requisitos para conversão e instalação terem sido estabelecidos pelo usuário, manuais e guias de conversão e instalação foram providos e testados;

4-5 = em adição ao anterior, ferramentas de conversão e instalação foram providas e testadas.

Facilidade de Operação

Procedimentos efetivos de inicialização, backup e recuperação foram providos e testados durante a fase de teste do sistema. A aplicação minimiza a necessidade de atividades manuais na operação, tais como montagem de fitas, manuseio de papel e intervenções manuais diretas localizadas. Os possíveis níveis são:

0 = nenhuma consideração especial sobre operação foi estabelecida pelo usuário;

1-2 = processos efetivos de inicialização, backup e recuperação foram requeridos, providos e testados;

3-4 = em adição, a aplicação minimiza a necessidade de atividades manuais, tais como, montagem de fita e manuseio de papel;

5 = a aplicação foi projetada para ter operação auto-atendida.

Multiplicidade de Locais

A aplicação foi especificamente projetada, desenvolvida e recebeu suporte para ser instalada em múltiplos locais, dentro de diversas organizações. Os possíveis níveis são:

0 = nenhum requisito sobre a necessidade de mais de um local usuário;

1-3 = necessidade de multiplos locais foram considerados no projeto;

4-5 = documentação e planos de suporte foram providos e testados para suportar a aplicação em multiplos locais.

Facilidade de Mudança

A aplicação foi especificamente projetada, desenvolvida e recebeu suporte para facilitar mudanças, tais como disponibilidade de consulta flexível e o agrupamento, em tabelas mantidas pelo usuário, de dados sujeitos a constantes mudanças. Os possíveis níveis são:

0 = nenhum requisito especial foi projetado para minimizar ou facilitar mudanças;

1-3 = capacidade de consulta flexível foi provida;

4-5 = em adição, dados de controle ou com mudanças constantes, foram guardados em tabelas que são mantidas pelo usuário com processos interativos on-line.

IV.2.4 - CÁLCULO DO PRODUTO DO TRABALHO

Do que foi visto anteriormente e possível destacar a existência de três tipos diferentes de produto de trabalho:

- Software Existente = Produto do trabalho já feito, representando, em pontos por função, o tamanho que a aplicação tem.

- Software em Desenvolvimento = Produto do trabalho do desenvolvimento representando, em pontos por função, o tamanho do trabalho necessário para concluir o desenvolvimento.

- Software em Manutenção = Produto do trabalho de suporte /manutencão representando, em pontos por função, o tamanho do trabalho necessário para realizar a manutenção/suporte a um software, durante o período de um ano.

Para cada uma das situações acima existe uma fórmula diferente, sendo que, nos ítens seguintes cada uma será discutida particularmente.

IV.2.4.1 - PRODUTO DO TRABALHO - SOFTWARE EXISTENTE

O cálculo do tamanho ou Produto do Trabalho de um software já existente é, essencialmente, aquilo que foi discutido nos ítens IV.2.2 e IV.2.3. Aqui revisaremos, resumidamente, os passos necessários para esse cálculo.

A fórmula geral é:

```
+--------------------+
!                    !
!  PFE = PFB X FAE   !
!                    !
+--------------------+
```

onde

PFE = Pontos por Função Existentes

PFB = Pontos por Função Brutos

FAE = Fator de Ajuste Existente

Considerando o formulário anexo-2, que tem preenchimento auto explicativo, são necessários os seguintes passos:

- usando o formulário anexo-1, identificar e classificar todas as funções de cada tipo (entrada externa, saída externa, etc) obtendo a quantidade de cada um;
- transpor a quantidade de cada tipo para o formulário anexo-2 e multiplicar pelo fator-peso associado à complexidade de cada tipo;
- somar os produtos da multiplicação anterior, obtendo assim os Pontos por Função Brutos.
- ainda considerando o formulário anexo-2, estimar o grau de influência da cada uma das características gerais do software somá-los e calcular o Fator de Ajuste Existente,

conforme a fórmula:

```
+----------------------------+
!                            !
! FAE = 0,65 + (0,01 x SGI) !
!                            !
+----------------------------+
```

onde SGI = Soma dos Graus de Influência;

- por fim, na última parte do formulário, multiplicar os Pontos por Função Brutos pelo Fator de Ajuste Existente. O resultado é o tamanho da aplicação expresso em quantidade de pontos por função.

O formulário que consta do anexo-1 deve ser preenchido conforme as instruções abaixo:

- Tipo de Função = preencher com um dos cinco tipos de função consideradas pela metodologia (vide ítem IV.2.2);

- Software = este campo deve ser preenchido com o código do software. Para melhor esclarecimento sobre as razões que fazem o software ter esta estrutura de código, consulte os ítens IV.1.3 e IV.2.1. Seguem-se as considerações para o seu preenchimento:

  . projeto ou sistema = trata-se do código do projeto ou sistema que está sendo considerado;

. aplicação ou sub-sistema = se o projeto foi dividido em várias aplicações ou sub-sistemas, que por sua vez se constituem em softwares independentes, desenvolvidos, gerenciados e mantidos independentes, então aqui deve ser colocado o código da aplicação ou sub-sistema respectivo. Se,ao contrário, o projeto não teve essa divisão, então este campo deve ser preenchido com "9999";

. rotina ou módulo = se a aplicação foi dividida em rotinas ou módulos, gerenciados e mantidos de forma independente, então aqui deve ser colocado código da rotina ou módulo respectivo. Se não houve essa divisão, então este campo deve ser preenchido com "9999";

. programa = preencher com o código do programa. Quando se tratar de software que não seja programa, preencher com "9999";

. natureza = trata-se da natureza do trabalho que caracteriza o software.

DN - desenvolvimento novo

EX - expansão

MN - manutenção.

. nome = nome do software.

- Código da Função = trata-se de uma função especifica. Aqui deve ser colocado o código que a ela foi atribuido.
- Descrição = descrição da função.
- Complex Inicial = trata-se da complexidade inicial (simples, média ou complexa) que a função possui. Por "inicial" entende-se o seguinte:

. <u>funções de software em desenvolvimento-novo</u>

trata-se da complexidade que a função tem. Essa complexidade pode ser ajustada ao longo do desenvolvimento, de forma que,ao concluir o desenvolvimento, reflita aquilo que a função possui em definitivo.

. <u>funções de software em desenvolvimento-expansão</u>

trata-se da complexidade que a função possuia no momento de começar a expansão.

. funções de software em manutenção

trata-se da complexidade que a função possuia no momento em que o software passou a ficar em manutenção. O "momento" acima referido pode ser considerado como um dos abaixo:

- na finalização de um desenvolvimento novo;
- na finalização de uma expansão;
- na finalização de um período de manutenção.

OBS: a)para funções incluídas ao longo de uma expansão ou de um período de manutenção, a complexidade inicial é aquela com a qual foi incluída.

b)para desenvolvimento novo ou para contagem de softwares já existentes, este formulário só é usado até aqui.

- Forma
Inclusão = representa a maneira como a função participa do software:

. iniciada - a função já existia quando o software foi colocado em manutenção ou quando inciou uma expansão.

. incluida - a função foi incluida durante

um período de manutenção ou durante uma expansão.

- Situação Atual = representa a situação em que a função se encontra ao final de um período de manutenção ou ao final de uma expansão. Pode ser:
  . ativa - a função continua ativa;
  . excluida - a função foi excluída.

- Complex Após = representa a complexidade que a função passou a possuir após uma alteração. Obviamente, se a função for alterada este campo tem que possuir conteúdo.

IV.2.4.2 - PRODUTO DO TRABALHO DE DESENVOLVIMENTO

Com base em tudo que já foi discutido anteriormente, a fórmula geral para o cálculo do Produto do Trabalho de Desenvolvimento é agora apresentada. Ela serve tanto para um desenvolvimento totalmente novo quanto, para uma expansão.

A fórmula geral é:

$$PFD = (PFBinc + PFBalt\text{-}dep)FAEdep + (PFBexc \times FAEant)$$

onde

PFD = Pontos por função decorrentes do trabalho de desenvolvimento;

PFBinc = Pontos por função brutos que foram incluidos no software calculados conforme se espera sejam depois da conclusão do projeto;

PFBalt-dep = Pontos por função brutos que foram alterados no software, calculados conforme se espera sejam depois da conclusão do projeto;

FAEdep = Fator de ajuste existente no software depois da conclusão do projeto.

PFBexc = Pontos por função brutos que foram excluidos do software, calculados conforme eram antes de iniciar o projeto;

FAEant = Fator de ajuste existente no software antes de iniciar o projeto.

Quando o desenvolvimento for totalmente novo, somente os termos PFBinc e FAEdep terão valores significativos, os demais serão todos zeros.

Considerando o formulário do anexo-3, são necessários os seguintes passos:

- usando o formulário do anexo-1, identificar e classificar todas as funções de cada tipo (entrada externa, saída externa, interface externa, etc) que foram ou serão incluidas, alteradas (situação após conclusão) e excluidas;

- transpor a quantidade de cada tipo para o formulário do anexo-3 e multiplicar pelo fator-peso associado à

complexidade de cada tipo;

- somar os produtos da multiplicação anterior obtendo assim, os Pontos por Função Brutos que foram incluidos, excluidos e alterados (situação após conclusão).

- ainda considerando o formulário do anexo-3, estimar o grau de influência de cada uma das características gerais conforme eram antes de iniciar o projeto e conforme serão após conclusão, somá-los e, usando-os na fórmula do fator de ajuste, calcular o Fator de Ajuste Existente antes de iniciar o projeto (FAEant) e após a conclusão do mesmo (FAEdep).

- concluidos os cálculos anteriores resta apenas aplicá-los na fórmula geral para o cálculo do Produto do Trabalho de Desenvolvimento;
- o resultado será o tamanho do trabalho de desenvolvimento expresso em pontos por função.

É importante, no caso de expansão, ter em consideração a diferença existente entre expansão e suporte/manutenção. Para tanto, examine as definições do item IV.1.3 e também os dois últimos paragráfos do ítem IV.2.4.3.

IV.2.4.3 - PRODUTO DO TRABALHO DE SUPORTE/MANUTENÇÃO

Considerando tudo que já foi visto, será definida agora, a forma pela qual se calcula o Produto do Trabalho de Suporte/Manutenção. Essa medida pode ser obtida de duas maneiras:

- Recontando os pontos por função do software como um todo, após as mudanças ocorridas, ou

- Aplicando a fórmula geral para esse fim, que é:

$$PFS = PFant + (PFBinc + PFBalt\text{-}dep)\ FAEdep - (PFBexc + PFBalt\text{-}ant)\ FAEant$$

onde

PFS = pontos por função decorrentes do trabalho de suporte ou manutenção durante o período de um ano;

PFant = tamanho original do software, em termos de pontos por função ajustados ou líquidos, isto é, o tamanho real do software, antes de iniciar o suporte ou manutenção. Esse tamanho é aquele existente no início do período, aqui considerado um ano, no qual se vai medir o suporte;

PFBalt-ant = pontos por função brutos que foram alterados no software, calculados conforme eram antes de iniciar o projeto;

OBS.: Os demais termos da fórmula já foram definidos quando se descreveu, no ítem anterior, o Produto do Trabalho de Desenvolvimento.

No caso de se optar por recalcular o tamanho do software após um ano de manutenção/suporte, então o caminho é o

já definido anteriormente para o cálculo do Produto do Trabalho Software Existente.

Se a opção for a aplicação da fórmula geral então os passos são os seguintes:

- usando o formulário do anexo-1, identificar e classificar todas as funções de cada tipo (entrada externa, saida externa, etc) que, ao longo de um ano, foram incluidas, alteradas (situação antes início/após conclusão) e excluídas através de atividades de suporte;

- transpor a quantidade de cada tipo para o formulário do anexo-4 e multiplicar pelo fator-peso associado à complexidade de cada tipo;

- obter, pela soma dos produtos da multiplicação anterior, os Pontos por Função Brutos que foram incluidos, excluidos e alterados (situação antes início/após conclusão).

- tal como já definido para Produto do Trabalho de Desenvolvimento, calcular o Fator de Ajuste Existente antes de iniciar o projeto (FAEant) e após a conclusão do mesmo (FAEdep);

- após os cálculos anteriores resta aplicar a fórmula e o resultado será o tamanho do Trabalho de Suporte/Manutenção ocorrido, no período de um ano, para aquele software;

É muito importante que seja destacada a diferença entre suporte/manutenção e expansão, tal como ambas foram definidas no ítem IV.1.3. O trabalho da manutenção/suporte é

feito a um software existente podendo consistir de inclusões, alterações e exclusões de funções mas sem se constituir em projeto independente, isto é, sem ser um novo software em si. Isto significa, que sejam quais forem as modificações decorrentes do suporte/manutenção, elas serão incorporadas ao software original como parte integrante e indissociável dele.

Na expansão, ao contrário, o software desenvolvido é um software independente e dissociado do original. Embora partes do software original venham se constituir em base para o desenvolvimento da expansão, essa expansão, em verdade, é um novo software, desenvolvido e tido como tal. Na produção da expansão há obediência ao ciclo de desenvolvimento padrão, o que nao ocorre com a manutenção/suporte.

IV.3 - DETERMINAÇÃO DO ESFORÇO DO TRABALHO

Para a determinação da produtividade, tal como já anteriormente definido neste trabalho, é necessário sejam conhecidos tanto o Produto do Trabalho quanto o Esforço do Trabalho. A determinação do Produto do Trabalho, baseada no método Pontos por Função, foi analisada ao longo do item IV.2.

A determinação do Esforço do Trabalho, tal como conceituado no item IV.1.3, é básica para aferição da produtividade e deve acontecer pelo registro sistemático de todo esforço investido no desenvolvimento, nas suas diversas fases, conforme LEITE(56).

IV.3.1 - REGISTRO DO ESFORÇO DO TRABALHO

O registro do Esforço do Trabalho é a anotação sistemática de todos os tempos que foram utilizados em um projeto

de desenvolvimento, expansão ou suporte/manutenção.

Considerando-se a diferença conceitual existente entre produtividade individual, produtividade do projeto e produtividade do orgão produtor de software, é necessário que o registro do esforço do trabalho possa refletir essa diferença.

Todas as pessoas que, de alguma forma gastam seu tempo em atividades ligadas, direta e indiretamente, ao desenvolvimento e/ou manutenção/suporte de software devem ter seus tempos registrados. Entre essas pessoas incluem-se, por exemplo, o pessoal de desenvolvimento e suporte/manutenção, usuários, pessoal free-lancer ou contratados externamente, digitadores/conferentes ligados ao desenvolvimento (digitando/conferindo programas, massas de teste, etc), auxiliares administrativos, secretárias, etc. Por outro lado, é conveniente destacar que somente as pessoas realmente ligadas ao desenvolvimento é que devem ter seus tempos anotados.

Assim, o formulário do anexo-5, no qual deve ser diariamente registrado o tempo consumido em cada atividade, deve ter o seguinte preenchimento:

Data = A data do dia no qual se registrou a atividade.

Instalação = O código da instalação a qual pertence o orgão produtor de software. Esse código pode ser definido em tabela à parte em função de necessidades especificas. A razão de se fazer essa anotação é a necessidade de se apurar a

produtividade global do orgão produtor de software naquela instalação.

Projeto ou sistema = Trata-se do código do projeto ou sistema no qual foi aplicado aquele tempo. Tanto pode ser um projeto de desenvolvimento quanto um de suporte/manutenção.

Se a atividade não se referir especificamente à um projeto, mas que, mesmo assim, foi gasta em função de necessidades gerais ligadas ao desenvolvimento ou suporte/manutenção então, o código a anotar e 9999. Isso significa que o tempo registrado não será considerado como esforço de trabalho de nenhum projeto em particular, porém será considerado para calcular a produtividade da instalação como um todo. Haverá um registro a parte de cada projeto, caracterizando-o em todos os seus detalhes.

Aplicação ou Sub-sistema = Muitas vezes, para facilitar o desenvolvimento o projeto é dividido em aplicações ou sub-sistemas com funções específicas. Essas aplicações, conforme tratado no ítem IV.2.1, embora sendo partes de um mesmo projeto, vão se constituir em software independentes e, por isso, precisam ser gerenciados, controlados e

avaliados isoladamente. Assim, se existir tal divisão, esse campo será preenchido com o código da aplicação à qual a atividade registrada está relacionada, entretanto, será preenchido com 9999, se:

- Não existir divisão em aplicações ou subsistemas, isto é, se não for necessário avaliação a nível de aplicação ou subsistema, ou

- Se atividade registrada referir-se ao projeto como um todo e não a uma aplicação ou subsistema específico.

Rotina ou Módulo = Também há, em certas circunstâncias, a divisão aplicação em rotinas ou módulos. Neste caso, este campo será preenchido com o código da rotina ou módulo. Será preenchido com 9999 se não houver tal divisão ou, havendo, a atividade registrada referir-se à aplicação ou sub-sistema como um todo e não à uma rotina ou módulo especifico.

Programa = Preencher este campo com o código do programa qual a atividade registrada está relacionada. Se a atividade registrada não referir-se a um programa específico, então esse campo deverá ser preenchido com 9999, significando com

isso, que o tempo gasto nessa atividade será contado como pertencente à aplicação ou rotina como um todo, embora sem pertencer a nenhum programa específico.

Natureza = Trata-se da natureza do trabalho que pode ser:

DN - desenvolvimento novo;

EX - expansão;

MN - manutenção.

Fase = Anotar neste campo o código da fase do ciclo de vida ao qual esta ligada a atividade, conforme abaixo:

001 - Definição de requisitos

002 - Projeto conceitual

003 - Projeto detalhado

004 - Desenvolvimento - Codificação

005 - Implementação

006 - Pós Implementação

007 - Manutenção

1XX - Se a atividade não está ligada especificamente a nenhuma fase, então deve ser usado este código, substituindo-se o " XX ", pelos dois útimos algarismos do código da fase durante a qual a atividade foi executada.

Tarefa = Preencher com código da tarefa. Tarefa

é a subdivisão das fases.

Atividade = Neste campo deve ser anotado o código da atividade na qual foi consumido o tempo registrado. As atividades a serem consideradas são as abaixo definidas, as quais, representam entre 90 a 100% de todas as atividades possíveis de ocorrer em um ambiente de desenvolvimento e suporte/manutenção de software. É importante destacar, que as atividades relacionadas podem ocorrer em qualquer uma das fases do ciclo de vida. O agrupamento feito tem fim exclusivo de apresentação ou eventuais necessidades estatísticas que vierem a surgir.

Atividades de Projeto e Desenvolvimento:

1.01 - Levantamento de necessidades

1.02 - Análise de dados

1.03 - Desenvolvimento das especificações

1.04 - Desenvolvimento do projeto conceitual

1.05 - Desenvolvimento do projeto detalhado

1.06 - Codificação nova

1.07 - Análise/seleção código reusável

1.08 - Criação/controle código reusável

1.09 - Integração de módulos/unidades

1.10 - Simulação e demonstração

1.11 - Implantação

1.12 - Digitação/conferência de programas

1.13 - Avaliação de impactos de mudança

1.14 - Implementação de mudanças solicitadas

Atividades de Garantia da Qualidade:

2.01 - Revisão das especificações

2.02 - Revisão do projeto conceitual

2.03 - Revisão do projeto detalhado

2.04 - Revisão da codificação

2.05 - Depuração preliminar

2.06 - Análise de erros

2.07 - Conserto de erros

2.08 - Preparação/digitação dados teste

2.09 - Teste de unidade/módulo

2.10 - Teste geral do sistema

2.11 - Controle da qualidade final

2.12 - Auditagem de sistema

2.13 - Análise retrospectiva de manutenção

Atividades de Treinamento/Educação:

3.01 - Treinamento do usuário sobre o projeto

3.02 - Treinamento pessoal desenvolvimento sobre o projeto

3.03 - Treinamento pessoal manutenção sobre o projeto

3.04 - Treinamento em geral pessoal desenvolvimento/manutenção

3.05 - Auto-estudo pessoal desenvolvimento/manutenção

3.06 - Exame/experiência novidades técnicas

Atividades de Documentação:

Documentação Interna

4.01 - Preparação dos planos de desenvolvimento

4.02 - Preparação de orcamentos/estimativas de custo

4.03 - Preparação da proposta de desenvolvimento

4.04 - Documentação do projeto conceitual

4.05 - Documentação do projeto detalhado

Documentação Externa

4.20 - Preparação do manual do usuário

4.21 - Preparação do manual de treinamento

4.22 - Preparação do manual de implantação/conversão

4.23 - Preparação do manual de manutenção

4.24 - Preparação do manual geral do sistema

4.25 - Documentação do código reusavel

4.26 - Tradução de manuais

Atividades de Administração:

5.01 - Planejamento do desenvolvimento

5.02 - Avaliação do desenvolvimento

5.03 - Reuniões sobre projeto

5.04 - Reuniões pessoal tecnico - assuntos gerais

5.05 - Outras reuniões em geral

5.06 - Criação/manutenção de normas padrões de desenvolvimento/manutenção

5.07 - Avaliação/seleção de ferramentas e recursos de desenvolvimento/manutenção

5.08 - Tratativas de aspectos legais ou contratuais

5.09 - Contratação de pessoal de desenvolvimento

5.10 - Atividades de avaliação/controle da produtividade

5.11 - Produção/distribuição de documentos

5.12 - Viagens

5.13 - Outras atividades administrativas

Hora Inicio/Fim = Nesses dois campos devem ser anotadas, em termos de hora/minuto, o momento exato em que iniciou/terminou o tempo dedicado a atividade. Notar que não é necessariamente o término da atividade e sim o término do tempo que,naquela ocasião, foi usado na atividade. Trata-se portanto do esforço líquido do trabalho tal como definido no ítem IV.1.3.

Os dois campos restantes sao auto-explicativos e servem para garantir que o tempo anotado foi corretamente registrado.

IV.3.2 - CÁLCULO DO ESFORÇO DO TRABALHO

No ítem anterior foi descrita uma forma sistemática de registrar o tempo aplicado em cada atividade de desenvolvimento e suporte/manutenção. Aqui será descrita a forma pela qual esse tempo deve ser apurado, a fim de servir para a determinação da produtividade e para compor uma base de informações históricas que viabilizem futuras estimativas.

O formulário do anexo-6 (Resumo do Esforço do Trabalho), deve ser preenchido com a tabulação feita a partir dos dados registrados no formulário do anexo-5 (Registro Diário do Esforço do Trabalho).

Essa tabulação deve ser feita nas seguintes ocasiões:

- Conclusão de Programa - O Esforco do Trabalho apurado servirá para calcular a produtividade naquele programa, possibilitando assim a avaliação da produtividade a nível individual.

- Conclusão de Rotina ou Módulo - Caso o projeto tenha sido dividido em rotinas ou módulos o Esforço do Trabalho apurado servirá para calcular a produtividade naquela rotina ou módulo possibilitando assim, a avaliação da produtividade a nível da equipe responsável por aquela rotina ou módulo.

- Conclusão de Aplicação - Caso o projeto tenha sido dividido em aplicações ou sub-sistemas, o Esforço do Trabalho apurado servirá para calcular a produtividade naquela aplicação, ensejando assim, a avaliação da equipe responsável.

- Conclusão do Projeto - No momento que um projeto de desenvolvimento for dado como concluido, isto é, tenha sido liberado para funcionamento global, ele deve ter seu Esforço do Trabalho apurado, o qual servirá para calcular a

produtividade naquele projeto, possibilitando assim, a avaliação da produtividade a nível de projeto ou da equipe que foi sua responsável.

. Encerramento de Ano - O Esforço do Trabalho de Manutenção/Suporte deve ser calculado ao final de cada ano, a fim de ser apurada a produtividade na manutenção/suporte, permitindo assim, a avaliação da equipe ligada a essa atividade.

Também no final de cada ano deve ser apurado o Esforço do Trabalho total da instalação, a fim de ser calculada e avaliada a produtividade a nível da empresa, ou seja, do orgão produtor de software. É importante destacar que no Esforço do Trabalho total da instalação, deve ser considerado o que segue:

- Não entram as atividades ocorridas durante a fase de " manutenção ".

- Não entram as atividades de um projeto ainda não finalizado.

- Entram todas as atividades de projetos que tenham sido cancelados, não importando o motivo.

É muito importante destacar que, face a existência de atividades que não são específicas de um programa, ou de uma rotina, ou de uma aplicação, ou de um projeto, as somas não sào cumulativas, isto é:

- a soma dos tempos de todos os programas não é igual ao total da rotina ao qual pertencem, isso porque tem atividades que participam da soma da rotina mas não participam de nenhum programa específico;

- a soma dos tempos de todas as rotinas não é igual ao total da aplicação ou sub-sistema, pela mesma razão

- a soma dos tempos de todas as aplicações não é igual ao total do projeto ao qual pertencem. Pode existir atividade ligada ao projeto como um todo mas que não esta ligada a nenhuma aplicação especifica, e

- a soma dos tempos de todos os projetos não é igual ao total da empresa ao qual pertencem, pela mesma razão já citada, isto é, existem atividades que participam da soma da instalação mas não participam de nenhum projeto específico.

Assim sendo, na apuração dos tempos consumidos, é obrigatório que se faça uma tabulação específica para cada coisa, revisando em cada vez, todos os registros feitos. Assim, haverá uma soma para programas, outra soma para rotinas ou módulos, outra soma para aplicação ou sub-sistema, outra soma para projetos ou sistemas e, por fim, outra soma para a instalação.

No formulário do anexo-6 (Resumo do Esforço do Trabalho), existem campos que merecem explicações adicionais para

preenchimento. São eles:

Nível = Por nível deve ser entendido uma das opções abaixo:

1 - Indicará que o resumo se refere à conclusão de um programa.

2 - Indicará que o resumo se refere à conclusão de uma rotina ou módulo.

3 - Indicará que o resumo se refere à finalização de uma aplicação ou sub-sistema, não importando se foi ou não implantado.

4 - Indicará que o resumo se refere à finalização de um projeto, não importando se tenha sido implantado ou cancelado.

5 - Indicará que o resumo se refere a finalização de um ano de manutenção/suporte para aquele projeto.

6 - Indicará que o resumo se refere a finalização de um ano de desenvolvimento naquela instalação. Lembrar que neste resumo entram as atividades de projetos finalizados por cancelamento mas não entram as atividades de projetos ainda não

finalizados ou então que ocorreram durante a fase de manutenção.

Tipo = Este campo indica o tipo de finalização ocorrida, na forma abaixo:

C - Indica que a finalização ocorreu por cancelamento do projeto ou módulo ou programa.

I - Indica que a finalização ocorreu por implantação, ou seja, o projeto/aplicação/rotina/programa foi realmente concluido e implantado.

Obs.: Para resumos anuais (manutenção e empresa) não há necessidade de preenchimento.

Código = Preencher este campo com um dos códigos abaixo dependendo a que se refere o resumo:

- Código do programa;
- Código da rotina ou módulo;
- Código da aplicação ou sub-sistema;
- Código do projeto ou sistema, ou,
- Código da natureza do trabalho

Nome = Preencher com o nome do programa, da rotina da aplicação, do projeto ou da instalação, dependendo a que se referir o resumo.

Instalação = Preecher com o código da instalação

Data Início/Data Conclusão = Deve ser preenchido com a data na qual iniciou/concluiu o programa, rotina, aplicação ou projeto. Não preencher para resumo de manutenção ou de instalação.

Ano-Base = Preencher com o ano ao qual se refere o resumo. Neste caso somente para resumos da instalação ou de manutenção.

A matriz das fases e atividades é auto-explicativa e deve ser preenchida com o resultado da tabulação feita, significando com isso que, em cada posição da matriz será registrado o Esforço Líquido, o Trabalho empregado naquela atividade durante aquela fase.

A transformação das horas líquidas em horas brutas deverá ser feita utilizando o fator de conversão usado na empresa, tal como definido no item IV.1.3.

IV.4 - DETERMINAÇÃO DA PRODUTIVIDADE

Com base no tamanho do Produto do Trabalho, calculado através da metodologia Pontos por Funçao descrita ao longo do ítem IV.2, e no Esforço do Trabalho calculado conforme descrição no ítem IV.3, é possível então apurar a produtividade.

O formulário do anexo-7 (Relatório de Produtividade Resumo Final, será usado para essa apuração. Seu preenchimento, nos campos comuns ao Resumo do Esforço do

Trabalho, tem as mesmas explicações daqueles, sendo os demais auto-explicativos.

O formualário do anexo-10 (Fatores de Produtividade-Estimativa de Influência) deve ser usado para analisar os possíveis fatores que explicam o baixo ou o alto índice de produtividade. Nessa análise deve ser destacado não somente o fator em si, mas também o grau de influência estimado. Essas considerações e análises são muito importantes porque vão servir de base para os processos de estimativas que serão tratados mais adiante.

Ao final de cada ano, os resultados registrados no formulário do anexo-7 (Relatório da Produtividade - Resumo Final), devem ser transportados para os formulários do anexo-8 (Análise Individualizada da Produtividade) e do anexo-9 (Análise Evolutiva Global da Produtividade).

O preenchimento do formulário do anexo-8 deve ser feito como segue:

Ano-Base = Ano ao qual se refere a análise.

Natureza = Indicar se é manutenção, novo desenvolvimento ou expansão.

Nível = Indicar qual o nível de resumo, conforme abaixo:

- Programas
- Rotinas
- Aplicações
- Projetos

Código = Preencher com o código do programa, da rotina, da aplicação ou do projeto conforme seja o nível.

Nome = Idem

Responsável = Anotar o nome de responsável pelo software.

Produtividade = Indicar aqui a quantidade de pontos por função obtida a título de produtividade.

Custo = Indicar aqui a medida de custo obtida.

O preenchimento do formulário do anexo-9 (Análise Evolutiva Global da Produtividade) destina-se a registrar, ano a ano, a evolução dos índices de produtividade da empresa como um todo. Seu preenchimento é auto-explicativo e não demanda maiores comentários.

CAPÍTULO V

ALTERNATIVAS PARA ESTIMAR ESFORÇO DE DESENVOLVIMENTO

Até hoje, a questão do tamanho do software foi sempre um fator impeditivo para o surgimento de fórmulas confiáveis para calcular o esforço e o tempo de desenvolvimento de software.

As fórmulas surgidas na literatura sempre dependeram da métrica utilizada para expressar o tamanho. Daí o surgimento de discrepância, porque as pesquisas que têm produzido essas fórmulas, quase nunca se baseiam em dados conceitualmente uniformes.

Com o advento da métrica Pontos por Função, passará a existir bases de dados criadas com apoio em conceitos homogêneos, viabilizando para o futuro, o surgimento de fórmulas definitivas para esse mister. Enquanto o futuro não chega a realidade presente precisa ser administrada. Na ausência de estimativas totalmente confiáveis, é necessário que existam estimativas consistentemente aceitáveis.

Por essa razão, e embora a metodologia Pontos por Função tenha objetivo centrado no cálculo do tamanho de software e no cálculo e avaliação da produtividade, seus resultados, o enfoque e os recursos que ela enseja, serão utilizados neste trabalho para propor formas alternativas de fazer estimativas no desenvolvimento de software.

Assim, e apesar de não ser esse o objeto principal deste trabalho, serão apresentadas neste capítulo, três alternativas para fazer tais estimativas. Sobre a rápida abordagem feita em (3) será aditado um novo enfoque mais abrangente, especialmente no que diz respeito às diversas opções de apuração desse esforço.

Na seção V.1 será visto um panorama geral dessas alternativas, na V.1.1 será detalhado o cálculo do esforço-histórico, na V.1.2 será detalhado o esforço-atividade e na V.1.3 será detalhado o cálculo do esforço-fórmula.

Embora com característica de total simplicidade, essas alternativas se apresentam como perfeitamente factíveis e objetivas.

V.1 - DETERMINAÇÃO DO ESFORÇO DE DESENVOLVIMENTO

Baseado nas especificações já definidas, o tamanho provável do software deve ser calculado, conforme descrito na seção IV.2. Com esse tamanho em mãos e, tendo por base os dados registrados conforme itens IV.3 e IV.4, podem ser calculados três possíveis esforços para o projeto. São:

Esforço-Histórico - Considerando o tipo e as características do projeto deve ser examinado qual o índice (custo) de produtividade médio registrado para projetos similares. Se esse índice, é por exemplo, 10 horas-brutas para cada ponto por função e

se o tamanho calculado é 300 pontos por função (PF), então o esforço-histórico é 3.000 horas-brutas.

Esforço-Atividade - Este tempo é determinado tendo por base a necessidade de cada atividade. Assim, examinando projetos similares já concluidos, é calculada a média do tempo gasto em cada uma das atividades, relacionadas em IV.3.1, por PF de cada projeto. Essa média é expressa da seguinte forma: para cada ponto por funçao são necessarias X horas-brutas de trabalho para codificação, Y horas-brutas de trabalho para análise de dados, Z horas-brutas de trabalho para conserto de erros, e assim sucessivamente. Dessa forma, se o tamanho calculado é 300 PF, basta multiplicar, X,Y,Z...., por 300 e temos a quantidade de horas de cada atividade. A soma delas é o esforço necessário.

Esforço-Fórmula - O esforço calculado com base nas alternativas anteriores pode ser melhor estabelecido se forem considerados os fatores de produtividade, presente em cada desenvolvimento, e que, normalmente, mudam a cada novo projeto. Para isso, é proposta uma fórmula, na qual todos os elementos influenciadores estejam refletidos. Este tipo de esforço é calculado utilizando-se uma fórmula padrão e tendo seu resultado ajustado em função de fatores ligados à produtividade, ou seja, fatores que aumentam ou diminuem o tempo

necessário, conforme seja seu grau de influência. Mais adiante será apresentada a fórmula padrão referida.

O Esforço Total necessário para o desenvolvimento do software poderá ser um dos três ou, então, aquele que demonstrar maior consistência. Poderá também ser a média entre os dois com base em históricos, ou ainda, a média entre os três esforços encontrados. O resultado deverá ser expresso em meses-brutos.

Esse Esforço Total poderá depois ser usado em uma das fórmulas de Tempo Total presentes na literatura. Esse Tempo Total, como já definido, é o tempo previsto para decorrer entre o início e o término do desenvolvimento. Entre as mais aceitas está a de BOEHM(8).

Uma outra abordagem para calcular esse Tempo Total, seria utilizar o mesmo enfoque utilizado para calcular o Esforço-Histórico e o Esforço-Atividade. Ou seja, entre projetos similares, se verificaria o quanto, para cada PF, decorreu em Tempo Total e então, com base nos PFs estimados para o software em desenvolvimento, pode ser calculado o seu tempo total.

Da mesma forma, entre projetos similares, se verificaria o quanto, para cada PF do projeto, decorreu de tempo para cada uma das fases e, com base nos PFs estimados para o software em desenvolvimento, pode-se calcular o Tempo Total de cada fase.

O Tempo Total decorrido é medido em função do registro feito no formulário do anexo-5 (Registro Diário do Esforço do Trabalho), ou seja, o Tempo Total é aquele decorrido entre o registro da primeira e da última atividade em cada fase ou projeto.

V.1.1 - CÁLCULO DO ESFORÇO-HISTÓRICO

Para calcular o Esforço-Histórico são necessários os seguintes passos:

- Escolher no mínimo três projetos similares já concluidos e calcular o índice (custo) médio de produtividade entre eles. Por similares devem, entender-se projetos que se assemelhem, no mínimo quanto a:

  - linguagem usada;
  - tamanho aproximado;
  - experiência da equipe;
  - classe do sistema, ou seja.
    - uso pessoal
    - uso na empresa
    - domínio público
    - comercialização externa
    - contratos especiais
    - software de sistema, etc;
  - tipo do sistema, ou seja,
    - batch, tempo real, controle de processos,

" on line ", processamento de imagens, etc;

- natureza do trabalho.
  - . desenvolvimento
  - . expansão
  - . manutenção

- nível do software
  - . projeto ou sistema
  - . aplicação ou sub-sistema
  - . rotina ou módulo
  - . programa

. O índice de produtividade de cada projeto, calculado conforme IV.3 e IV.4, é aquele que consta no formulário do anexo-7 (Relatório de Produtividade - Resumo Final).

. Uma vez obtido o índice médio ele deve ser multiplicado pelo tamanho estimado que já deve ter sido calculado.

. Do produto anterior devem ser diminuidas as horas-brutas já empregadas no projeto.

. O resultado anterior deve ser convertido para meses-brutos.

V.1.2 - CÁLCULO DO ESFORÇO-ATIVIDADE

Para calcular o Esforço-Atividade são necessários os seguintes passos:

. escolher no mínimo três projetos similares já concluidos

e dividir a quantidade de horas gastas em cada atividade pelo tamanho final do projeto. Assim, para cada projeto, saberemos quantas horas por PF, foram necessárias em cada atividade;

- para cada atividade, calcular a média de horas por PF entre os três projetos;
- multiplicar a média de horas por PF pelo tamanho estimado do software em desenvolvimento;
- converter o resultado para horas brutas e abater as horas brutas já empregadas no projeto;
- o resultado deve ser convertido para meses-bruto;
- por projetos similares deve ser entendido o mesmo já considerado em IV.1.1, e
- a quantidade de horas gastas em cada atividade por projeto deve ser obtida do formulário anexo-6/Resumo do Esforço do Trabalho.

V.1.3 - CÁLCULO DO ESFORÇO-FÓRMULA

A expressão mais simples de uma fórmula para encontrar o esforço seria a seguinte:

$$ET = (PF * IP) - HE \qquad (k)$$

onde

ET = Esforço total necessário expresso em horas-brutas;

PF = Tamanho, em Pontos por Função, estimado para o software;

IP = Índice (custo) de produtividade média registrado na instalação para projetos similares (x horas brutas para cada PF);

HE = Horas brutas já empregadas no projeto.

Acontece, que o IP da fórmula varia em função dos fatores de produtividade que permeiam cada projeto.

Conforme descrição no capítulo VI, pode existir diferentes graus de influência para cada um dos fatores de produtividade.

Conceitualmente o IP da fórmula (k) pode ser definido como sendo o pior IP possível na instalação, menos o incremento devido a fatores de produtividade. O pior IP possível é aquele que seria obtido quando todos os fatores de produtividade estivessem na condição mais negativa. Esse IP será identificado por PHPF, ou seja, Pior Resultado em Horas por PF. Nesse caso, a fórmula para calcular o IP seria a seguinte:

IP = PHPF - (PHPF * FP) (1)

onde

PHPF = Significa o pior resultado possível de ser obtido, isto é, quando todos os fatores de produtividade estiverem na situação mais negativa possível. Os fatores de

produtividade e os diversos graus de influência de cada um, estão descritos no capítulo VI.

FP = Significa o grau de influência dos diversos fatores de produtividade que estão associados ao projeto. O grau de influência de cada um varia de 0,01 a 0,99, dependendo de como ele se apresenta como fator influenciador do desenvolvimento. A maneira de determinar esse grau de influência esta no capítulo VI.

Substituindo o IP na equação (k) temos:

$$ET = PF\ [PHPF - (PHPF * FP)] - HE \quad \text{ou}$$

$$ET = [(PF * PHPF) - (PF * PHPF * FP)] - HE \quad \text{ou}$$

$$ET = [PHPF * PF(1 - FP)] - HE \quad (m)$$

Ocorre, que esse PHPF, na prática, é dificil de ser encontrado e a literatura apresenta discrepâncias muito grandes quando aborda a questão.

Em (2) o pior caso considerado é de 100 horas por PF. Em (29) o pior caso encontrado foi de 85,5 horas por PF. Em (55) há indicação de que os menores coeficientes de produtividade situam-se entre 34 e 86 horas por PF. Em (25) os três piores casos estudados foram 106,6; 96,0 e 53,6 horas por PF.

Não é viável portanto mantê-lo na fórmula.

Tome-se a equação (l) e isole-se o PHPF:

$$IP = PHPF - (PHPF * FP) \qquad (l)$$

$$IP = PHPF\ (1 - FP)$$

$$PHPF = \frac{IP}{1 - FP} \qquad (n)$$

Com essa fórmula aplicada a projetos similares desenvolvidos anteriormente na instalação é possível encontrar o PHPF específico daquela instalação. Assim, aplicando a equação (n) na equação (m) temos:

$$ET = \left[\frac{IPsimilar}{1 - FPsimilar} * PF\ (1 - FP) \right] - HE \qquad (p)$$

onde

IPsimilar = é o índice de produtividade obtido para projetos similares;

FPsimilar = é o fator de produtividade encontrado em projetos similares;

PF = é o tamanho, em Pontos por Função do software que está sendo medido;

FP = é o fator de produtividade presente no software que esta sendo medido;

HE = horas brutas já empregadas.

Para que seja encontrado o FP da fórmula devem ser seguido os seguintes passos:

- Usando o formulário do anexo-10 (Fatores de Produtividade-Estimativa de Influência, registrar o grau de influência de cada fator.
  O grau de influência de cada fator é obtido com base na descrição do capítulo VI.

- Os graus estimados são multiplicados pelos seus respectivos pesos e os resultados somados;

- O total dessas somas é dividido pela somatória dos pesos e o resultado é o FP da fórmula.

- O IPsimilar é obtido através do formulário do anexo-7 (Relatório de Produtividade-Resumo Final) dos projetos que foram tidos como similares;

- O FPsimilar é obtido através do formulário do anexo-10 (Fatores de Produtividade-Estimativa de Influência), dos projetos que foram tidos como similares.

O peso associado à cada fator foi definido com base no potencial influenciador que cada um tem em relação aos demais. O potencial influenciador foi inferido a partir das tabelas do capítulo VI, porém não há garantia de consistência para seus valores. A definição é simples proposta sem fundamentação sólida.

Depois de calculado, o ET, deve ser transformado em meses-brutos.

A estimativa obtida com apoio neste capítulo é, obviamente, discutível. Não foram feitos testes exaustivos na prática porque o objetivo fundamental deste trabalho é outro. Entretanto, há neste capítulo, recursos simples e claros, que permitem assegurar a existência de estimativas, senão totalmente reais, pelo menos consistentemente aceitáveis.

CAPITULO VI

FATORES SIGNIFICATIVOS DE PRODUTIVIDADE

Tal como definido em IV.1.3, existem dois tipos de fatores permeando as considerações sobre grandeza de software. Um deles engloba os fatores do tamanho que tem a ver com o software em si, independente das condições externas sob as quais o software será produzido. O outro engloba os fatores de produtividade que tem a ver, exatamente, com as condições externas propiciadoras de aumento ou diminuição na produtividade. Os fatores referentes ao primeiro tipo foram discutidos em IV.2 e os fatores referentes ao segundo tipo serão discutidos neste capítulo.

Os fatores de produtividade tratados aqui são aqueles que entram para o cálculo do tempo baseado em fórmula, tal como discutido em V.1.3. Também será baseado neles que se farão as análises de produtividade sugeridas em IV.4.

Cada fator de produtividade considerado neste capítulo será apresentado em conjunto com seu conceito e possíveis graus de influência que pode assumir na produção de software. Esses possíveis graus foram estabelecidos com base em analogia ao impacto de cada fator, conforme apresentado na literatura. Para cálculo da influência geral dos fatores em conjunto, deve ser usado o formulário anexo-10/Fatores de Produtividade - Estimativas de Influência.

Este capítulo foi escrito com base em JONES(9), especialmente quanto a seleção dos fatores significativos, que nele encontram-se plenamente analisados e justificados. Na análise de cada fator, serão adicionados outras fontes que o tratam, caso haja.

Uma observação adicional às seções que se seguem é que, o esforço empregado em manutenção, refere-se a um período de cinco anos.

VI.1- LINGUAGEM DE CODIFICAÇÃO

O surgimento de muitas Linguagens de Programação, a existência de linguagens específicas para certos tipos de atividade, o advento de linguagens de quarta geração e as ditas não-procedurais, tem dificultado a mensuração da influencia que a Linguagem de Programação exerce sobre a produtividade.

A literatura tem tratado exaustivamente a questão, mas os resultados apresentados são quase sempre ofuscados pelo paradoxo tratado no Cap.III com exeção para ALBRECHT e GAFFNEY (12), BEHRENS(31) e JONES(55) que situam adequadamente essa influência.

A verdade é que as linguagens de alto nível produzem influências que são claras e facilmente comprováveis, bem como, influências sutis que embora não estensivamente comprováveis, são também significantes.

Entre as primeiras existem:

- A óbvia diminuição de código para completar determinada função;
- Menos probabilidade de erros, devido a menor quantidade de código a escrever;
- As estruturas sintática e semântica de determinadas linguagens, dada a especificidade a que se destinam, já trazem embutidos recursos que diminuem o esforço de codificação. Como exemplo temos LISP para inteligência artificial e APL para Manipulação de matrizes.

Entre as segundas incluem-se a diminuição de esforço para projetar e desenvolver programa quando se usa Linguagens Interpretativas (APL, BASIC, FORTH, ETC), posto que produzem protótipos imediatos de determinadas funções; as linguagens com recurso de protótipação generalizadas diminuem o esforço no desenvolvimento do projeto (LINC); a existência, em determinadas linguagens, de recursos para manipular Banco de Dados; a existência, em determinadas linguagens, de funções para documentação, etc.

Dos estudos que JONES(9) fez, sobre um software com as mesmas funções mas escritos em linguagens diferentes, observam-se os seguintes resultados:

FATORES	LINGUAGENS			
	ASSEMBLER	COBOL	APL	PLANILHA
01-Tamanho em LCF	20.000	6.500	1.600	400
02-Tamanho em PF	62	62	62	62
03-Custo Total US$	335.000	145.000	57.500	21.200
04-Esforço Total (mês/hom)	65	27	9,5	2,2
05-Tempo total(mês)	10	7	4	1,2
06-Custo por LCF US$	16.75	22.31	35.94	52.50
07-Custo por PF US$	5403	2.338	927	338
08-LCFs por Mês	308	241	168	181
09-PF por Mês	0,95	2,4	6,5	9,5
10-Esforço Manut. (mês/hom.)	2	2	2	2

FIGURA 7: O reflexo da Linguagem na construção de software

Assim, dependendo da linguagem em que o software for codificado pode haver uma maior ou menor produtividade no seu desenvolvimento. Essa diferença e' devida ao potencial de expressão que as linguagens foram adiquirindo ao longo do tempo, diminuindo o esforço empregado na fase de codificação com reflexo também em outras fases como manutenção, teste, depuração e outros.

Os diversos graus de influência foram organizados por grupos como abaixo:

GRUPOS	GRAU DE INFLUÊNCIA
GRUPO 1: São Linguagens de Baixo Nivel, tidas como de Primeira Geração. Neste grupo incluem-se: o ASSEMBLER BASICO, MACRO ASSEMBLER e outras derivações do ASSEMBLER.	0,01 - 0,09
GRUPO 2: São Linguagens com potencial de expressão de até 4 em relação ao ASSEMBLER. Neste grupo incluem-se o ALGOL, COBOL, C, CHILL, FORTRAN, JOVIAL, PASCAL, etc.	0,10 - 0,39
GRUPO 3: São Linguagens com potencial de expressão de 4 a 7 em relação ao ASSEMBLER. Neste grupo incluem-se RPG, PL-1, MODULA-2, ADA, PROLOG, LISP, FORTH, BASIC, LOGO, etc.	0,40 - 0,65

GRUPOS	GRAU DE INFLUÊNCIA
GRUPO 4:	
São Linguagens com potencial de expressão de 8 a 30 em relação ao ASSEMBLER. Neste grupo incluem-se APL, SMALLTALK, OBJECTIVE-C, DBASE, BANCO DE DADOS (quarta geração), LINGUAGENS DE QUARTA GERACAO, LINC, LINGUAGENS DE CONSULTA, STRATEGEM, etc	0,66 -0,79
GRUPO 5:	
São Linguagens com potencial de expressão acima de 30 em relação ao ASSEMBLER. Neste grupo incluem-se sobretudo as PLANILHAS, LINGUAGENS NÃO PROCEDURAIS, etc.	0,80 - 0,99

VI.2 - TAMANHO DO SOFTWARE

"Produtividade diminui a medida que o tamanho do software aumenta, não porque o incremento no tamanho do código requeira esforço mais significativo, e sim por causa das diversas atividades periféricas que são severamente afetadas, como documentação, integração, planejamento, teste, remoção de

erros, etc" (9).

Com enfoque similar, outras fontes como BOEHM(8) e BROOKS(35), são uniformes em afirmar que a medida que o software aumenta de tamanho ele se torna mais caro e mais demorado para ser produzido.

Em um projeto grande há mais necessidade de documentação, de comunicação entre os membros do grupo, o volume e a complexidade dos dados aumenta, interfaceamento entre módulos e programas requer mais atenção, atividades de interação, teste e revisão ganham significado preponderante.

Dos estudos em JONES(9), feitos sobre três softwares de mesmo tipo mas tamanhos diferentes, os seguintes resultados emergem:

FATORES	TIPOS DE SOFTWARE		
	FINANÇAS EMPRESARIAIS	FOLHA DE PAGAMENTO	ORÇAMENTO
01- Linguagem	COBOL	COBOL	COBOL
02- Tamanho em LCF	1.000	10.000	100.000
03- Tamanho em PF	9,5	95	950
04- Custo Total US$	40.000	410.000	7.825.000
05- Custo por LCF US$	40	41	78,25
06- Custo por PF US$	2.894	3.631	5.710
07- Esforço mês/homem	5,5	69	1.085
08- LCF por mês	182	145	92
09- PF por mês	1,7	1,4	0,9
10- Esforço Manutenção	2.5	13	480
11- Primeira Atividade mais cara	Codificação	Codificação Cons. defeito	Correçao defeito
12- Segunda Atividade mais cara	Correćão defeitos	Teste	Documentação
13- Terceira Atividade mais cara	Documentação	Gerência	Codificação

FIGURA 8: O reflexo do Tamanho do Software na construção de software.

Na tabela acima, qualquer fator que expresse produtividade, fica severamente afetado a medida que o software aumenta de tamanho. Por essa razão, e necessário que o grau de influência exercido sobre a produtividade seja adequadamente refletido. Para esse fim foi desenvolvida a fórmula abaixo, que embora reflita apenas linearmente a influência do tamanho, possibilita tê-lo em consideração. A fórmula é a seguinte:

$$GI = 1 - \frac{0.99\ Tx}{Tmax}$$

ONDE:

GI = Grau de Influência devido ao tamanho do software.

Tx = Tamanho previsto do software em questão, expresso em pontos por função.

Tmax = Tamanho máximo, expresso em ponto por função, que um software pode ter naquela particular instalação. Cada organização deve estabelecer seu próprio Tmax, uma vez que o grau de influência devido ao tamanho, variará em função das possibilidades e experiência que a organização tem em produzir grandes softwares.

VI.3 EXPERIÊNCIA DA EQUIPE.

Como em qualquer atividade humana, a produção de software é influenciada pelas experiências e habilidades daqueles que estiverem envolvidos.

Pessoas com pouca experiência não possuem domínio sobre os recursos ao seu dispor, não estão certos do caminho a tomar, não conhecem com profundidade as nuances implícitas e não

guardam segurança sobre suas possibilidades próprias. Esse quadro gera incertezas e desconfiança junto a equipe e aos usuários envolvidos, dificultando ainda mais as atividades de desenvolvimento.

Pessoas experientes, ao contrário, não perdem tempo explorando alternativas inviáveis, não ficam indecisos ante caminhos diferentes, ficam plenamente a vontade diante dos recursos disponíveis e são capazes de tomar decisões com base em percepções sutis que só são possíveis como fruto da experiência. Um quadro desse estimula os circunstantes, transmite segurança e o grau de incerteza diminui ou se anula.

A literatura registra o estudo dessa questão, como em BOEHM(8), CURTIS(36) e outros, indicando este fator como um dos que mais impacto produz na construção de um software, embora sem equacionar questões como área de experiência (técnica, funcional, administrativa, etc) e fase e atividades nas quais a experiência tem impacto.

Com referência a áreas em que a experiência deve ser vista, podem ser consideradas as seguintes:

. LINGUAGEM E FERRAMENTAS

Trata-se da experiência que a equipe tem na utilização da linguagem de programação na qual o software esta sendo construido e na utilização das ferramentas automatizadas de apoio ao desenvolvimento;

. MÉTODO E TÉCNICAS

Trata-se da experiência que a equipe possui no uso da metodologia de desenvolvimento que esta sendo utilizada na instalação;

. ÁREA DE APLICAÇÃO

Trata-se da experiência que a equipe reune, relacionada com a área de aplicação que esta sendo desenvolvida.

Com esse enfoque, e considerando o estudo que JONES(9) fez para o mesmo software desenvolvido por quatro equipes com diferentes experiências, os seguintes resultados foram observados:

F A T O R E S	CASO 1	CASO 2	CASO 3	CASO 4
1- Tamanho em LCF-COBOL	1000	1000	1000	1000
2- Experiência em:				
2.1 Ling/Ferram	ALTA	BAIXA	ALTA	BAIXA
2.2 Métodos/Técnicas	ALTA	BAIXA	ALTA	BAIXA
2.3 Área Aplicação	ALTA	ALTA	BAIXA	BAIXA
3- Custo Total (US$)	29.000	35.000	42.500	70.000
4- Custo por LCF	29,00	35,00	42,5	70,00
5- Esforço(mês/homem)	4	5	6	10
6- LCF por mês/homem	250	200	167	111
7- Esforço manutenção	1,8	2,0	2,5	4,0

FIGURA 9: A influência da Experiência na construção de software.

Na tabela acima verifica-se que conforme o grau de experiência em cada uma das áreas, o impacto na produtividade é diferente. Para determinar o grau de influência exercido pela experiência, adotou-se o seguinte roteiro:

		FAIXA / PESO						
		nenhu X1	baixa X2	média X5	alta X8	compl X10	TOTAL	MÉDIA
Linguagem e Ferramentas	1							
	2							
Métodos e Técnicas	3							
	4							
Área de Aplicação	5							
	6							
Total	7							
	8							

FIGURA 10: Matriz de experiência de equipe

(a) As linhas 1,3 e 5 são preenchidas com a quantidades de membros que estão situados em cada uma das faixas de experiência;

(b) As linhas 2,4 e 6 são preenchidas com a multiplicação da quantidade pelo peso respectivo da faixa ;

(c) Para situarmos cada membro na sua respectiva faixa de experiência, utiliza-se o seguinte:

. Para experiência com Linguagem e Ferramenta/Métodos e Técnicas:

Nenhuma = membros que possuem de 0 a 1 ano de experiência;

Baixa = membros que possuem de 1 a 3 anos de experiência;

Média = membros que possuem de 3 a 6 anos de experiência;

Alta = membros que possuem de 6 a 10 anos de experiência;

Completa = membros que possuem acima de 10 anos de experiência;

. Para experiência com área de aplicação:

Nenhuma = membros que nunca tiveram contato anterior com o assunto da aplicação;

Baixa = membros que só participam de um desenvolvimento anterior no mesmo assunto;

Média = membros que já participaram de no mínimo 3 desenvolvimentos anteriores;

Alta = membros que possuem formação profissional sobre o assunto ou que já participaram de no mínimo 5 desenvolvimentos anteriores;

Completa = membros que possuem formação profissional sobre o assunto e/ou que já participaram em acima de 6

desenvolvimentos anteriores.

(d) A coluna média é obtida atráves da divisão nas dois totais relacionados à cada área de experiência (2/1, 4/3 6/5 e 8/7);

(e) Em seguida para obter-se o real grau de influência na produtividade, aplica-se a fórmula abaixo:

GI = 0,099 EXPx

onde

GI = é o grau de influência a ser calculado

EXPx = é a expressão da experiência média geral existente na equipe envolvida com o projeto. Está na intercessão entre a última linha e última coluna da Matriz de Experiência de Equipe (figura 10).

VI.4 - MÉTODOS ESTRUTURADOS.

No ínicio das atividades de produção de software, a ótica utilizada e difundida era de que, desenvolver software, representava um esforço artesanal, gerado com base na arte de gênios privilegiados. Com o amadurecimento desse ramo do conhecimento, suas atividades tiveram maior disciplinamento e a matéria envolvida foi sistematizada, transformando-se em ciência.

Depois da identificação de um ciclo claro de desenvolvimento, as pesquisas voltaram-se para o disciplinamento e a sistematização de cada uma das fases envolvidas.

Daí o surgimento de uma série de métodos e técnicas

tratando, básicamente, de como estruturar as atividades envolvidas em cada fase. Como as primeiras tentativas vieram adjetivadas com a palavra "estruturada", surgiram uma infinidade de abordagens que se apresentavam como estruturadas: análise estruturada, requisitos estruturados, documentação estruturada, manutenção estruturada, projeto estruturado, programação estruturada, etc.

Considera-se, entretanto, como estruturadas somente aquelas técnicas e métodos que estejam embasados em algum fundamento teórico, devidamente formalizado. Portanto, o conceito de estruturado aqui é no sentido de estar sistematizado, fundamentado e adequadamente disciplinado em visão global envolvendo todos os aspectos da questão.

Com esse enfoque, a literatura só registra estruturação para as fases de definição de requisitos como em WARNIER(40), JACKSON(41) e GANE(42); de projeto como em MYERS(37), STEVENS(38) e CONSTANTINE(39); e de codificação como WIRTH(44) e DIJKSTRA(43).

A verdade é que, o advento dessas abordagens trouxe um grande incremento na produtividade do desenvolvimento de softwares e como afirma JONES(9) " os métodos estruturados, somente por eles, não resolvem todos os problemas do desenvolvimento, nem são capazes de trasformar amadores em profissionais. Entretanto, quando rigorosamente aplicados, eles registram sólidos progressos ".

Dos estudos em JONES(9), feitos com cinco sistemas de

mesmo tamanho mas desenvolvidos com métodos de diversos níveis de estruturação, obseva-se o seguinte:

FATOR	CASO 1	CASO 2	CASO 3	CASO4	CASO 5
1-Especificação	N.Estrut.	Estrut.	N.Estrut.	N.Estrut.	Estrut.
2-Projeto	N.Estrut.	N.Estrut.	Estrut.	N.Estrut.	Estrut.
3-Código	N.Estrut.	N.Estrut.	N.Estrut.	Estrut.	Estrut.
4-Tamanho(LCF)	10.000	10.000	10.000	10.000	10.000
5-Linguagem	COBOL	COBOL	COBOL	PASCAL	PASCAL
6-Esforço(m/h)	62	59	58	54	45
7-Cust.Tot. US$	460.000	355.000	355.000	400.000	260.000
8-Custo por LCF	46	35	35	40	26
9-LCF/(mês/hom)	161	169	169	185	222
10-Esforço/manut	30	12	13	26	7

FIGURA 11: A influência dos Métodos Estruturados na construção de software

Em função disso, a utilização de métodos estruturados deve ter seu grau de influência destacado, quando se estiver analisando a produtividade. Para isto, propõe-se o seguinte:

0,01 - 0,08 - Desenvolvimento sem utilização de qualquer método.

0,09 - 0,29 - Desenvolvimento com utilização de método estruturado apenas na codificação.

0,30 - 0,49 - Desenvolvimento com utilização de método estruturado para projeto e codificação.

0,50 - 0,69 - Desenvolvimento com utilização de método

estruturado para definição de requisitos, projeto e codificação.

0,70 - 0,89 - Desenvolvimento total feito com o uso de métodos, porém com abordagem independente por fase.

0,90 - 0,99 - Desenvolvimento total feito com base em uma única metodologia estruturada da definição de requisitos até a implantação.

VI.5 - AMBIENTE E FERRAMENTAS.

À medida em que se avança no conhecimento aprofundado das atividades de produção de software, a medida que esse conhecimento se torna mais sistematizado, e padronizado surge, também, a possibilidade de se ter essas atividades todas automatizadas.

Ferramentas automatizadas para várias dessas atividades já são uma realidade. Trabalhos anteriores na UFRJ/COPPE podem ser citados, entre eles, AGUIAR(45), BLASCHECK(46), SANTOS(47), ROCHA e SOUZA(48)e (50).

Certo é, que a automatização dessas atividades produz um impacto significativo na produtividade. As razões são diversas, mas principalmente por causa de:

. Rapidez - devido a natureza das ferramentas, realizar certas atividades se torna menos cansativo, mais rápido e menos suscetível de erros. Daí diminuir o tempo de sua produção;

. Revisão - revisar e consertar atividades já

concluidas é sempre complexo, por causa de reflexos e conexões que elas guardam entre si e, principalmente, com fases e atividades posteriores;

. Padronização - O uso de ferramentas automatizadas força a uniformização de procedimentos, o acesso a fonte sempre atualizada e disciplina a condução do andamento.

Dos estudos de JONES(9), feitos com três softwares de mesmo tamanho mas desenvolvidos em ambientes com ferramentas em diversos graus de automatização, observa-se o seguinte:

FATORES	MELHOR AMBIENTE	MÉDIO AMBIENTE	PIOR AMBIENTE
1-Tamanho em LCF de C	100.000	100.000	100.000
2-Esforço(mês/homem)	790	1.367	1.953
3-Custo Total US$	4.045.000	7.140.000	10.330.000
4-Manutenção(mes/homem)	19	61	113
5-Custo por LCF	40,45	71,40	103,30
6-LCF por (mês/homem)	126	73	51

FIGURA 12: A influência do Ambiente e das Ferramentas na construção de softaware

Para que esse fator tenha sua influência considerada propõe-se o seguinte:

1- DEFINIÇÃO DE REQUISITOS.

1.1- Existência de metodologia

1.2- Metodologia usada manualmente

1.3- Ferramenta para gráficos

1.4- Ferramenta para textos

1.5- Ferramenta integrada a fase seguinte

2- PROJETO CONCEITUAL.

2.1- Existência de metodologia

2.2- Metodologia usada manualmente

2.3- Ferramenta para gráficos

2.4- Ferramenta para textos

2.5- Ferramenta integrada à fase seguinte

3- PROJETO DETALHADO.

3.1- Existência de metologia

3.2- Metologia usada manualmente

3.3- Ferramentas para gráficos

3.4- Ferramenta para texto

3.5- Ferramenta integrada a fase seguinte

4- DOCUMENTAÇÃO.

4.1- Método para documentar

4.2- Estrutura básicas dos manuais automatizados

4.3- Editores configurados conforme estrutura

4.4- Disponibilidade on-line

4.5- Manutenção automatizada

5- BIBLIOTECAS TÉCNICAS

5.1- Existência de definições de dados

5.2- Existência de código reutilizavel

5.3- Bibliotecas mantidas e documentadas

6- CONDIÇÕES TRABALHO.

6.1- Terminais para utilização

6.2- Moveis, espaço fisico e materiais acessórios

6.3- Silêncio e isolamento

6.4- Apoio logístico

6.5- Disponibilidade de Hardware

7- FERRAMENTAS DE GERÊNCIA

7.1- Ferramenta para previsão de tempo/custo totais

7.2- Ferramenta para previsão por fase e tarefas

7.3- Geração a manutenção automatizadas de planos

7.4- Acompanhamento automatizado do projeto

7.5- Avaliação automatizada de produtividade

A cada um dos itens acima deve ser atribuido um valor que vai de 0,00 a 0,03 comforme seja o grau de excelência demonstrado.

A soma dos valores anotados para cada item produzirá o grau de influência do fator "AMBIENTE E FERRAMENTAS" na produtividade do desenvolvimento.

VI.6 - QUALIDADE DE EXPANSÃO.

A literatura tem sido unânime em afirmar que expandir um software é menos produtivo que desenvolvê-lo totalmente novo.

Isso ocorre porque em uma expansão há fatores adicionais que aumentam a necessidade de trabalho. Por exemplo:

- O software a ser expandido tem que ser profundamente estudado e compreendido e nem sempre há documentação adequada disponível;

- Mesmo que as mudanças sejam pequenas no software original há necessidade de recompilar e testar todos os programas,de atualizar manuais e documentaçao em geral.

- As expansões sempre utilizam técnicas, métodos e ferramentas novas disponíveis. Entretanto, para isso, é preciso atualizar o software original, e fazê-lo adequado a receber a expansão.

Essas circunstâncias, entretanto, são a natureza mesmo da expansão e fazem parte do trabalho que é desenvolvido. Seja qual for o ambiente, a linguagem, a experiência, o equipamento ou o fabricante sempre haverá esse tipo de trabalho.

Por essa razão, conforme afirmado na seçao III.1, não se pode comparar a produtividade obtida em uma expansão com a produtividade obtida em um desenvolvimento totalmente novo.

O trabalho adicional necessário a um projeto de expansão é, pois, parte do "produto do trabalho", ele é um fator de tamanho e não de produtividade.

A produtividade obtida em uma expansão deve ser comparada a outras obtidas também em expansões. Para isso, entretanto, o produto do trabalho de uma expansão deve ser

calculado de forma diferente daquela utilizada para calcular o produto do trabalho de desenvolvimento totalmente novo.

Essa distinção foi considerada na seçao IV.4.2.2, onde formas diferentes são apresentadas para o cálculo de um e outro.

Apesar do anteriormente considerado, há projeto de expansão que é mais produtivo do que outro, mesmo que se mantenham iguais outros fatores. Parece que a diferença existente origina-se em circunstâncias peculiares à natureza da expansão. JONES(9) identificou dois tipos de razões básicas:

- Abrangência da conexão entre o código novo e o existente.
- Status da qualidade do código do existente.

Do estudo apresentado por ele, feito com três produtos de software de mesmo tamanho, adicionados a software de tamanhos iguais mas com qualidade e conexões diferentes, observam-se os seguintes resultados:

FATORES	CASO 1	CASO 2	CASO 3
1- Tamanho Existente(LCF)	30.000	30.000	30.000
2- Tamanho Adicionado(LCF)	10.000	10.000	10.000
3- Linguagem	COBOL	COBOL	COBOL
4- Abrangência Conexão	apenas novos módulos	novos módulos e mudanças no existente	novos módulos e mudanças no existente
5- Qualidade Existente	estável	estável	problemático
6- Esforço (mês/homem)	71,8	89,8	98,0
7- Manutenção(mês/homem)	21,0	27,0	30,0
8- Custo Total (US$)	455.000	585.000	640.000
9- Custo por LCF(US$)	45,50	58,50	64,00
10-LCF por mês	143	111	102

FIGURA 13: A influência da abrangência da conexão e da qualidade do software existente nos projetos de expansão.

Face a isso, esses dois aspectos devem ter suas influências consideradas nos projetos de expansão. Propõe o seguinte:

ABRANGÊNCIA DA CONEXÃO

0,35 - 0,49 - O código novo será adicionado atráves de módulos independentes sem grandes conexões com o código existente;

0,20 - 0,34 - Parte do código novo será adicionado atráves de módulos independentes e outra parte atráves de mudanças no código existente;

0,01 - 0,19 - O novo código será adicionado integralmente atráves de mudanças no código original.

STATUS DA QUALIDADE

0,40 - 0,49 - Código existente estável. Número de defeitos por ano situa-se, no máximo, em 0,5 por cada 1000 LCF;

0,30 - 0,39 - Código existente, em processo de estabilização. Volume de defeitos anuais situa-se entre 0,6 e 3,0 por cada 1000 LCF;

0,20 - 0,29 - Código existente é instável. Volume anual de defeitos situa-se entre 3,1 e 6,0 por cada 1000 LCF;

0,10 - 0,19 - Código existente é propenso a erros. Volume anual de erro situa-se entre 6,1 e 10,0 por cada 1000 LCF;

0,01 - 0,09 - Código existente é extremamente problemático. Volume anual de erros situa-se acima de 10,0 por cada 1000 LCF.

OBS.: Quando se tratar de desenvolvimento novo este fator deve ser considerado como 0,99.

VI.7 - REUTILIZAÇÃO DE CÓDIGO.

A consciência de que boa parte do código desenvolvido para integrar determinados softwares, pode também ser utilizada

para integrar outros softwares, levou programadores e analistas a iniciarem, espontaneamente,um movimento no sentido da utilização racional dessa técnica.

A padronização de algoritmos, rotinas comuns, módulos de cálculos padrões, etc, tem sido uma prática adotada em todas as empresas que apresentam altos índices de produtividade.

Reutilização é uma tecnologia emergente que aos poucos ganha definição clara. UM exemplo disto são as novas linguagens como ADA, MODULA-2, FORTH, ETC, que já trazem em sua essência um direcionamento para reutilização de códigos.

Na seção IV.2.3.2, quando foram descritas as características gerais de um software em desenvolvimento, foi incluida a característica reutilizabilidade como fator influenciador do tamanho. Lá, a ótica utilizada, é no sentido de mensurar a influência que existe quando se desenvolve um software visando reutilizar suas partes componentes.

Aqui, a ótica utilizada, é no sentido de mensurar a produtividade que existe quando, no desenvolvimento de um software, utiliza-se código já existente previamente, ou seja, quando se faz uso de reutilização de código.

A influência desse fator é significativa, conforme se observa na tabela abaixo. Nela, que foi tirada dos estudos de JONES(9), são considerados quatro softwares de mesmo tamanho, mas construidos com percetagens diferentes de código reutilizado.

A tabela é a seguinte:

FATORES	SEM REUSO	25% REUSO	50% REUSO	75% REUSO
1- Tamanho Total (LCF - COBOL)	10.000	10.000	10.000	10.000
2- LCF REUSADO	0	2.500	5.000	7.500
3- Esforço(mês/homem)	60,5	38,0	27,0	12,0
4- Manutenção(mês/homem)	21,0	7,0	5,0	3,0
5- Custo Total US$	407.500	225.000	160.000	75.000
6- Custo p/LCF US$	40,75	22,50	16,00	7,50
7- LCF por mês	165	263	370	833

FIGURA 14: A influência da Reutilização de Código na construção de software.

Para refletir essa influência na expectativa de produtividade, propõe-se que se estabeleça, em termos percentuais, o quanto do código que irá compor o novo sistema, será oriundo de reutilização.

Estabelecida a reutilização prevista divide-se por 100 e o resultado será o grau de influência da fórmula.

VI.8 - CIRCUNSTÂNCIAS GEOGRÁFICAS.

O desenvolvimento de um software pode ser grandemente afetado pela questão da separação geográfica dos locais que servirão de base para as equipes que o construirão.

Por separação geográfica entende-se aqui a não localizaçao contígua fisicamente dos orgãos, empresas ou filiais que precisarão estabelecer comunicações mútuas. Sob essa ótica,

locais situados em uma mesma cidade, mas distantes fisicamente, já estão separados geograficamente.

A medida que cresce a existência de equipes separadas aumenta também a necessidade de coordenação, de comunicação, de trâmite de documentos. Nessa situação as respostas demoram, as decisões custam a ser tomadas e a introdução de modificações ou consertos se torna problemática.

A verdade é que essas circunstâncias diminuem a produtividade conforme se observa na tabela abaixo.

Nessa tabela, que foi extraida dos estudos de JONES(9), são consideradas três situações para um software de mesmo tamanho e características. As diferenças devidas as questões geográficas são as seguintes:

CASO 1 - Software desenvolvido por uma mesma equipe em uma mesma localidade.

CASO 2 - Software desenvolvido cooperativamente entre equipes de seis localidades diferentes, sendo duas nos Estados Unidos e quatro na Europa. Todas as localidades estão ligadas entre si por recursos de teleconferência, transferência de dados e programas, redes de tele-processamento, etc.

CASO 3 - Similar ao caso 2 sendo que sem a existência de redes de tele-processamento ou qualquer outro recurso de comunicação. Os contatos são feitos através de viagens ou reuniões periódicas.

FATORES	CASO 1 Mesmo Local	CASO 2 6 locais com comunicação	CASO 3 6 locais sem comunicação
1- Tamanho em LCF -COBOL	250.000	250.000	250.000
2- Esforço(mês/homem)	4.100	4.860	6.175
3- Manutenção(mês/homem)	1.780	1.780	1.780
4- Custo Viagens US$	50.000	250.000	3.750.000
5- Custo Total US$	29.450.000	33.450.000	43.525.000
6- Custo por LCF	117,60	133,80	174,10
7- LCF por mês	61	51	40

FIGURA 15 :- A influência da Separação Geográfica na construção de software.

Para refletir essa influência, propõe-se a fórmula abaixo:

$$G\ I = 1 - \frac{QT^2}{100}$$

ONDE:

QT é a quantidade de diferentes locais que participam do desenvolvimento.

Pela fórmula acima considera-se que o ideal é quando existe apenas um local, pois nesse caso o grau de influência será 0,99.

VI.9 - MÉTODOS DE REMOÇÃO DE ERROS.

A literatura é abundante na descrição dos efeitos que o uso sistemático de métodos para previnir e remover erros produz no desenvolvimento de software.

A existência desses métodos desde as fases iniciais do desenvolvimento não só minimizam os problemas com manutenção, como também diminui o esforço necessário para desenvolver as fases seguintes.

Embora a adoção desses métodos gere um custo adicional, o resultado que eles produzem está acima de qualquer comparação. Mais impacto na produtividade eles têm quanto mais cedo forem introduzidos e quanto mais sistemática for sua aplicação.

Nos estudos de JONES(9), foram considerados dois softwares de mesmo tamanho com diferentes abordagens quanto aos métodos para remoção de erros. São as seguintes as características:

CASO 1 - Trata-se de um software desenvolvido para uso interno e que usou os seguintes métodos: teste pessoal de mesa, grupo informal para revisão do projeto, teste de unidade, teste de função e teste de integração.

CASO 2 - Trata-se de software para ser comercializado e que adotou maior rigor na prevenção e remoção de erros. Os métodos utilizados foram: teste pessoal de mesa, inspeções formais de projeto, inspeções formais de código, inspeção do plano de teste, inspeção da documentação do usuário, teste de unidade, teste de função, teste de integração e teste de campo.

Os principais resultados obtidos foram:

FATORES	SOFTWARE INTERNO	SOFTWARE PARA COMERCIALIZAR
1- Tamanho (LCF - COBOL)	250.000	250.000
2- Esforço(mês/homem)	2.041	2.564
3- Manutenção(mês/homem)	1.408	460
4- Custo Total US$	17.245.000	15.120.000
5- Custo por LCF US$	68,98	60,48
6- LCF por mês	122	97

FIGURA 16:- A influência dos Métodos de Remoção de Erros na construção de software.

Dessa forma, o grau de influência desse fator precisa estar mensurado e, para tal, propõe-se que seja atribuído um determinado valor para cada um dos principais métodos dependendo da fase ou atividade em que forem aplicados.

A soma desses valores fornecerá o grau de influência total devido a esse fator.

Para a obtenção do proposto segue-se a tabela abaixo:

MÉTODOS	REQUISITOS	PROJETO	CÓDIGO	DOCTAÇÃO
1- Grupo Informal de Revisão	0,04	0,04	0,02	0,01
2- Revisão Estruturada	0,04	0,04	0,03	0,01
3- Revisão Garantia Qualidade	0,06	0,06	0,04	0,03
4- Grupos Formais de Inspeção	0,04	0,04	0,03	0,03
5- Revisão Individual	0,01	0,01	0,01	0,01
6- Provas de Correção	-	-	0,08	-
7- Teste de Mesa	-	-	0,03	
8- Código Lido p/ Dupla	-	-	0,01	
9- Revisão pelo Lider	-	-	0,01	
10- Modelagem/Prototipação	-	-	0,01	
11- Teste de Unidade/Módulo	-	-	0,01	
12- Teste de Função	-	-	0,01	
13- Teste de Integração	-	-	0,01	
14- Simulação	-	-	0,04	
15- Teste de Performance	-	-	0,02	0,01
16- Teste de Campo	-	-	0,06	0,01
17- Teste de Aceitação	0,02	0,02	0,04	0,01

FIGURA 17: - Tabela de influências motivadas pelos Métodos de Remoção de Erros.

VI.10 - RECURSOS DE CODIFICAÇÃO.

Na seção VI.1 foi abordada a influência que a linguagem de codificação em si, exerce sobre a produtividade. Lá, examinou-se cada grupo de linguagens sob a ótica do potencial de expressão que possuiam em relação a uma unidade básica.

Na seção VI.5 foi abordada a influência que o Ambiente e Ferramentas exercem sobre a produtividade. Nessa seção, entretanto, não foi considerado o ambiente de codificação, pois ele, por si só, tem um impacto diferenciado na produtividade.

Nesta seção, está sendo considerado como fator independente a questão dos Recursos de Codificação.

Por Recursos de Codificação entende-se o conjunto de facilidades que o ambiente de codificação pode propiciar para o desenvolvimento do código a ser gerado.

Incluem-se nessas facilidades: as linguagens, a simplicidade do uso , as ferramentas de apoio, a integração do ambiente, a padronização de conceitos e técnicas e outros.

Em (51) há uma pesquisa feita em 600 instalações usuárias de ambientes UNISYS e IBM, conduzida por uma instituição independente chamada de "Customer Satisfations Research Institute".

Nessa pesquisa foi encontrada uma produtividade total de até 50% para um dos ambientes.

Na questão específica sobre facilidades de uso foi detectada também vantagens significativas para o ambiente mais produtivo.

Em SIMPSON e RUDOLPH (30) há uma pesquisa sobre recursos de codificação na qual o ambiente LINC é considerado como apresentando ganhos de produtividade que situam-se entre 10:1 e 80:1 quando comparado a um ambiente tradicional tipo COBOL.

Os fatores que conduzem a isso segundo a pesquisa são:

envolvimento de usuários no processo de programação; a unicidade, simplicidade e integração da linguagem; o apoio conceitual em uma metodologia sistematizada; e recursos de prototipação inclusos.

Em VALE(52) e SCHARER (53) há uma clara referência sobre o impacto que a prototipação causa no desenvolvimento como um todo.

Há outros recursos no ambiente de codificação que causam impacto na produtividade tais como geradores de aplicação ou programas, linguagem integrada ao banco de dados utilizado,geradores automáticos de teste, etc.

Das pesquisas de JONES(9) retiraremos três exemplos sobre a influência que esses recursos exercem na produtividade.

No primeiro exemplo temos o caso de um mesmo software desenvolvido através de recursos convencionais e através de prototipação, tal como abaixo:

FATORES	CONVENCIONAL	PROTOTIPAÇÃO
1- Tamanho (LCF - COBOL)	100.000	100.000
2- Esforço(mês/homem)	1.085	945
3- Manutenção(mês/homem)	480	400
4- Custo Total US$	7.825.000	6.725.000
5- Custo por LCF US$	78,25	67,25
6- LCF por mes	92	100

FIGURA 18:- A influência da prototipação na construção de software.

No segundo exemplo temos o caso de uma aplicação com mesmas funções mas desenvolvidas através de recursos

convencionais e com geradores de programa.

FATORES	COBOL	GERADOR DE PROGRAMA
1- Tamanho em Linhas - Fonte	10.000	2.000
2- Esforço em homem/mês	60	25,5
3- Esforço em Manut. homem/mês	21	8
4- Custo Total US$	407.500	167.500
5- Custo por Linha US$	40,70	83,75
6- Linhas por mês	165	78
7- Tamanho em pontos por função	95	95
8- Custo por PF US$	3.184	1.342
9- PFs por mês	1,6	3,7

FIGURA 19:- A influência de geradores de programa na construção de software.

No terceiro exemplo temos o caso de uma aplicação desenvolvida para extrair informações de um banco de dados. Em uma situação ela foi desenvolvida usando os recursos convencionais do COBOL e em outra utilizando uma linguagem de quarta geração.

FATORES	COBOL	LING DE 4 GERAÇÃO
1- Tamanho em Linhas - Fonte	1.000	50
2- Esforço em homem/mês	3	0,4
3- Esforço em Manut. homem/mês	0	0
4- Custo Total US$	16.500	2.000
5- Custo por Linha US$	16,50	40,00
6- Linhas por mês	303	125
7- Tamanho em pontos por função	10	10
8- Custo por PF US$	1.650	200
9- PFs por mês	3	25

FIGURA 20: - A influência das linguagens de quarta geração na construção de software.

Para refletir o grau de influência que os recursos de codificação geram na produtividade propõe-se que seja atribuido um determinado valor a cada um dos recursos existentes, conforme seja seu grau de excelência, nos termos abaixo:

FATORES	excelente	média	pouco
1 - Existe ferramenta para projeto de programa	0,05	0,03	0,01
2 - Existe ferramenta para geração de teste.	0,05	0,03	0,01
3 - Existe ferramenta para depuração de programa	0,05	0,03	0,01
4 - Existe ferramenta para documentação de programa	0,05	0,03	0,01
5 - Existe linguagem de quarta geração ou gerador de programa/aplicação	0,35	0,15	0,09
6 - A linguagem de manipulação do BD esta integrada a linguagem hospedeira	0,09	0,05	0,03
7 - Existe recurso de prototipagem	0,25	0,12	0,07
8 - Existe simplicidade/facilidade nos recursos de codificação	0,10	0,05	0,03

FIGURA 21: - Tabela de influências motivadas pelos Recursos de Codificação.

OBS: Na ausência de todos os fatores considerar o grau de influência como 0,00.

VI.11 - ORGANIZAÇÃO DA EQUIPE.

A literatura tem considerado a organização da equipe de desenvolvimento como um dos fatores significativos na construção

de software.

Entre as fontes mais conhecidas há BROOKS(35) com o clássico The Mythical Man-Month em que a forma dada à estrutura da equipe é tida como fundamental.

Considera-se que a estrutura hierárquica, na qual cada membro reporta-se à uma única chefia, é fator de incremento na produtividade.

A estrutura matricial, na qual cada membro pode estar respondendo simultaneamente à vários lideres dependendo do projeto ao qual estiver vinculado, é, ao contrário, fonte de dificuldades. As razões apontadas são várias, porém, as que mais se destacam são:

- inexistência de uma pessoa especifica como a responsável pelo sucesso do projeto como um todo;

- existência de conflitos entre os limites de autoridade do gerente do projeto e o gerente da instalação.

- aumento significativo das interações entre os membros da equipe, surgindo os problemas de comunicação.

Dos estudos de JONES (9) foram retirados os dados abaixo que referem-se à um mesmo software desenvolvido por equipes organizadas de maneiras diferentes.

FATORES	HIERÁRQUICO	MATRICIAL
1- Tamanho (LCF - COBOL)	100.000	100.000
2- Esforço(mês/homem)	1.085	1.230
3- Manutenção(mês/homem)	480	500
4- Custo Total US$	7.825.000	8.650.000
5- Custo por LCF US$	78,25	86,50
6- LCF por mes	92	81

FIGURA 22: - A influência da Organização da Equipe na construção de software.

Para refletir essa influência, propõe-se que seja determinada, no inicio do desenvolvimento (fase de projeto), a quantidade projetos não concluidos e com base na tabela abaixo, determinar o grau de influência.

ORGANIZAÇÃO DA EQUIPE	QUANTIDADE PROJETOS EM DESENVOLVIMENTO SIMULTÂNEO	VALORES	OBSERVAÇÕES
HIERÁRQUICA	até 01	0,99 - 0,90	A variação
HIERÁRQUICA	de 02 a 05	0,89 - 0,80	dentro de cada
MATRICIAL	até 02	0,79 - 0,70	faixa de valores
MATRICIAL	de 02 a 04	0,69 - 0,60	podera ser feita
HIERÁRQUICA	de 06 a 10	0,59 - 0,40	com base na
HIERÁRQUICA	acima de 10	0,39 - 0,25	quantidade de
MATRICIAL	de 05 a 10	0,24 - 0,15	projetos e no
MATRICIAL	acima de 10	0,14 - 0,01	tamanho que possuem

FIGURA 23: - Tabela de influência motivadas pela Organização da Equipe

VI.12 - SATISFAÇÃO DA EQUIPE.

Tal como em (54) tem havido recentemente uma preocupação em mensurar a influência que a moral da equipe de desenvolvimento exerce sobre a produtividade.

Por moral entende-se a motivação da equipe, a forma como é recompensada em seus esforços, a satisfação com o que fazem, a forma de remuneração, etc.

Essa influência é quase óbvia, mesmo porque, esse tipo de fator não afeta somente a produção de software, mas todo tipo de atividade humana.

JONES (9) estudou o assunto e, de suas conclusões, surgem dados significativos. Para dois software iguais, porém desenvolvidos por equipes com diferentes níveis de motivação, tem-se os seguintes resultados:

FATORES	AMBIENTE MOTIVADO	AMBIENTE SEM MOTIVAÇÃO
1- Tamanho (LCF - COBOL)	100.000	100.000
2- Esforço(mês/homem)	1.085	1.400
3- Manutenção(mês/homem)	480	560
4- Custo Total US$	8.607.500	8.725.000
5- Custo por LCF US$	86,08	86,25
6- LCF por mes	92	75

FIGURA 24: - A influência da Satisfaçao da Equipe na construção de software.

Para refletir essa influência sugere-se a adoção da tabela abaixo, da qual devem ser tirados os valores de cada um dos fatores que propiciam uma menor ou maior satisfação da equipe.

A soma dos valores de cada fator será o grau de influência devido pela satisfação da equipe.

FATORES	NÍVEIS		
	EXCELENTE	MÉDIO	RUIM
1- REMUNERAÇÃO	0,21-0,30	0,11-0,20	0,01-0,10
2- BENEFÍCIOS SOCIAIS	0,15-0,20	0,09-0,15	0,01-0,08
3- CONCEITO PÚBLICO DA EMPRESA	0,08-0,10	0,04-0,07	0,01-0,03
4- PLANO DE CARREIRA	0,08-0,10	0,04-0,07	0,01-0,03
5- INCENTIVO A EDUCAÇÃO	0,08-0,10	0,04-0,07	0,01-0,03
6- CRIATIVIDADE DO AMBIENTE	0,08-0,10	0,04-0,07	0,01-0,03
7- SEGURNÇA ECONOM. DA EMPRESA	0,08-0,09	0,04-0,07	0,01-0,03

FIGURA 25: - Tabela de influência motivadas pelas Satisfação da Equipe

VI.13 - OUTRAS CONSIDERAÇÕES SOBRE FATORES DE PRODUTIVIDADE.

Existem identificados na literatura uma série de outros fatores tidos como de produtividade. Entretanto, segundo a caracteristica "isolabilidade" e "expressividade" descritas na seção III.6, bem como, considerando a definição de conceitos feita na seção IV.1.3, esses fatores são associados ao tamanho do software e não à produtividade no seu desenvolvimento. Ou seja,

são "fatores de tamanho" e não fatores de produtividade.

Este é o caso, por exemplo, de complexidade, documentação, classe de software, etc.

Também existe uma série de outros fatores que são indicados na literatura como de produtividade os quais, entretanto, ainda não tiveram sua influência constatada e medida.

Este é o caso, por exemplo, de: centros de informação, centros de desenvolvimento, desenvolvimento por usuários,tamanho da empresa, tamanho da equipe de desenvolvimento, restrição de tempo e recursos, métodos padrões formalizados, treinamento sistemático do pessoal envolvido, etc.

CAPÍTULO VII

GERÊNCIA PRODUÇÃO DE SOFTWARE - VISÃO GERAL DE UM

VII.1 - INTRODUÇÃO

Nos capítulos anteriores foram abordados aspectos que, de forma geral, compõem o panorama com o qual a gerência da produção de software tem que se envolver ao longo de um desenvolvimento ou manutenção.

Assim, embora considerando que o objeto principal deste trabalho esgotou-se no capítulo anterior, seria conveniente a apresentação concatenada de todos esses elementos e aspectos, ensejando de forma organizada, uma visão geral do que seria um possível ambiente de gerência da produção de software.

Neste capítulo, esse possível ambiente será descrito em suas características principais. Para embasar essa descrição serão tomados como referência dois tipos de modelos gráficos. Um desses modelos, incluso no anexo 11, visa tão somente apresentar uma hierarquização concatenada dos componentes desse ambiente. O outro modelo, incluso no anexo 12, objetiva apresentar uma visão geral do fluxo dos dados e servirá de base para a implementação automatizada do ambiente. Esse modelo, será utilizado para a descrição do ambiente na seção VII.2, oportunidade em que, cada componente seu receberá um maior aprofundamento.

Os componentes de cada modelo são correspondentes e a base conceitual de todos eles está vinculada aos assuntos já tratados nos capítulos anteriores.

VII.2 - DESCRIÇÃO DO AMBIENTE

Esse ambiente foi dividido em cinco sub-sistemas

distintos, sendo que cada sib-sistema subdivide-se em módulos. Cada módulo divide-se em unidades e cada unidade subdivide-se em sub-unidades. Os cinco sub-sistemas são a seguir descritos.

VII.2.1 - DESCRIÇÃO DO SUB-SISTEMA DE CÁLCULAR O TAMANHO

Este sub-sistema (Figura 26), com base na metodologia Pontos por Função, descrita no capítulo IV, tem por objetivo determinar o tamanho do software a ser desenvolvido. Ele possui três módulos que são:

Módulo de Identificar as Propriedades do Software (1.1)

Com base no que foi descrito nas seções IV.2, IV.2.3 e VI, este módulo tem por objetivo identificar as diversas propriedades que o software possui e que são significativas para o cálculo do tamanho. Possui quatro unidades:

Unidade de Definir as Funções (1.1.1)

Esta unidade define todas as funções que o software possui conforme descrito nas seções IV.2.2.1 a IV.2.2.6. Possui uma sub-unidade para Identificação das Funções e outra para Criação e Manutenção das Funções.

Unidade de Converter Funções para início Expansão/Manutenção (1.1.2)

Esta unidade tem por objetivo fazer a conversão de funções de um desenvolvimento encerrado. As funções associadas a um desenvolvimento encerrado podem ser convertidas em funções de manutenção. Também as funções de um software em manutenção, podem ser convertidas para funções de desenvolvimento expansão.

Unidade de Definir as Características Gerais (1.1.3)

Com base na seção IV.2.3.2 esta unidade tem por objetivo definir as características gerais do software.

Possui uma sub-unidade para Identificação e outra para Criação e Manutenção.

Unidade de Definir os Fatores de Produtividade (1.1.4)

Com base no capítulo VI esta unidade tem por objetivo definir os graus de influência dos fatores de produtividade presentes no projeto. Possui uma sub-unidade para Identificação e outra para Criação e Manutenção.

Módulo de Calcular o Produto do Trabalho (1.2)

Com base nas seções IV.2.4, IV.2.2. e IV.2.3, este módulo calculará o produto do trabalho do software em refêrencia. Possui duas unidades:

Unidade de Calcular o Processamento Padrão (1.2.1)

Com base no que foi descrito no ítem IV.2.2, este módulo tem por objetivo a determinação dos Pontos por Função Brutos ou Processamento Padrão Associado da aplicação.

Unidade de Calcular o Fator de Ajuste (1.2.2)

Com base no conteúdo do ítem IV.2.3, este módulo tem por objetivo a determinação do Processamento Geral Associado ou Fator de Ajuste.

Unidade para Calcular Pontos por Função (1.2.3)

Com base no resultado das duas unidades anteriores serão calculados os Pontos por Função da Aplicação.

Módulo de Fornecer Apoio Metodológico (1.3)

Este módulo objetiva fornecer apoio àqueles que trabalharem no ambiente. Por apoio entende-se o conjunto de definições, conceitos e informações contidos neste trabalho.

SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE
1 - SUB-SISTEMA DE CALCULAR O TAMANHO
DIAGRAMA DE FLUXO DE DADOS

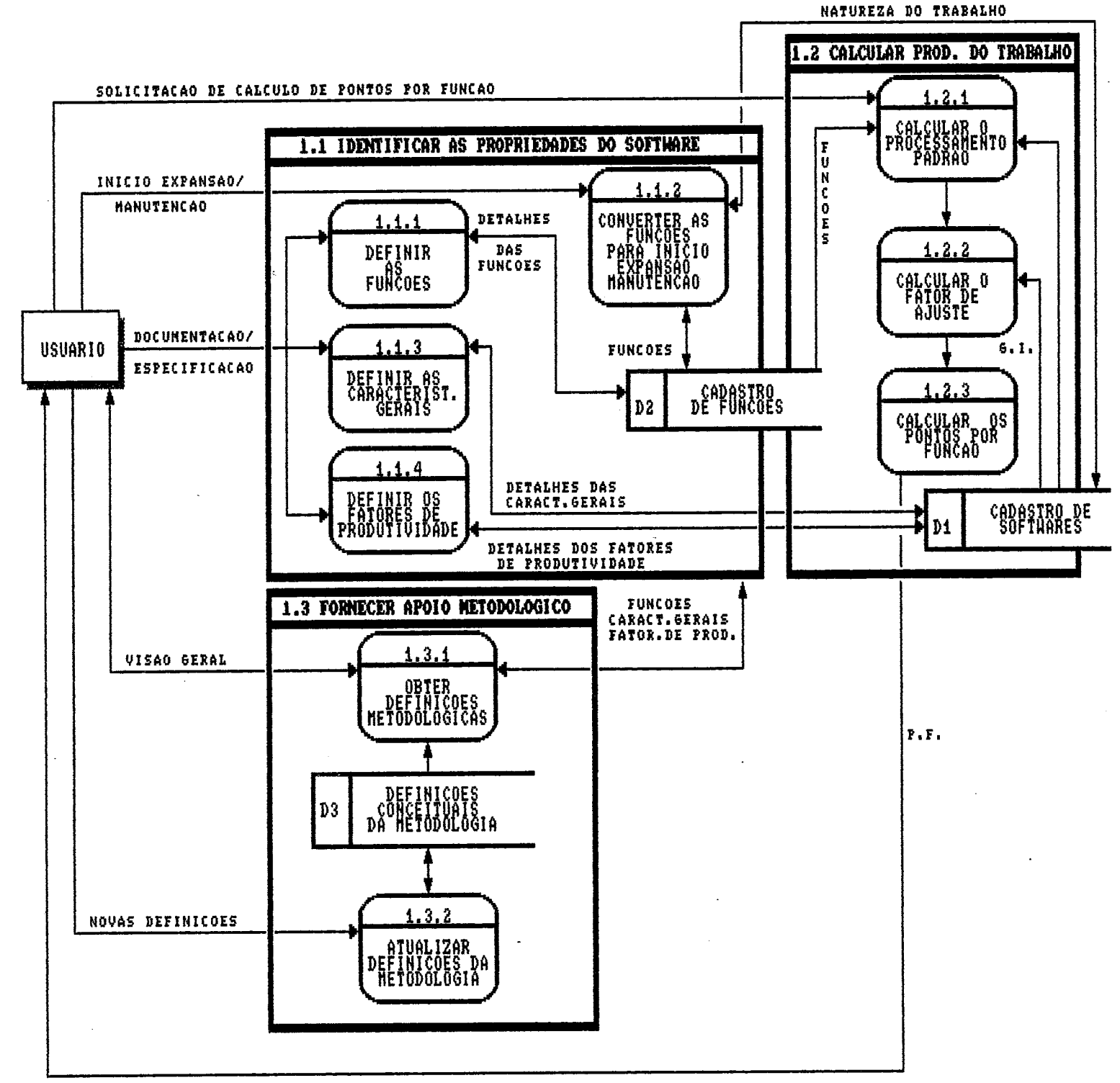


FIGURA 26: D.F.D. DO SUB-SISTEMA DE CALCULAR O TAMANHO

VII.2.2 - DESCRIÇÃO DO SUB-SISTEMA DE CÁLCULAR A PRODUTIVIDADE

Este sub-sistema (Figura 27), tal como definido no ítem IV.4, é responsável pelo cálculo da produtividade a nível de programa, módulo, projeto e, anualmente, a nível da empresa tanto para manutenção quanto para desenvolvimento em geral. Possui três módulos:

Módulo de Registrar o Esforço do Trabalho (2.1)

Com base na seção IV.3.1, este módulo tem por objetivo controlar o adequado registro e consistência do esforço real registrado diariamente para cada projeto. Possui uma unidade para Registrar o Esforço Diário (2.1.1) e outra para Consistir o Esforço Diário(2.1.2).

Módulo de Encerrar o Desenvolvimento (2.2)

Com base nas informações de encerramento do desenvolvimento, este módulo tem por objetivo fechar os registros em aberto, verificar a consistência global e garantir que todas as informações pertinentes foram fornecidas. Tem uma unidade encarregada de Finalizar o Desenvolvimento (2.2.1), e outra para analizar Dados Históricos (2.2.4).

Módulo de Apurar a Produtividade (2.3)

Tem por objetivo determinar a produtividade final para cada um dos níveis estabelecidos, isto é, ele calcula

respectivamente, a produtividade a nível de programa, de módulo, de projeto, de manutenção ao longo de um ano e desenvolvimento geral ao longo de um ano.

SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE
2 - SUB-SISTEMA DE CALCULAR A PRODUTIVIDADE
DIAGRAMA DE FLUXO DE DADOS

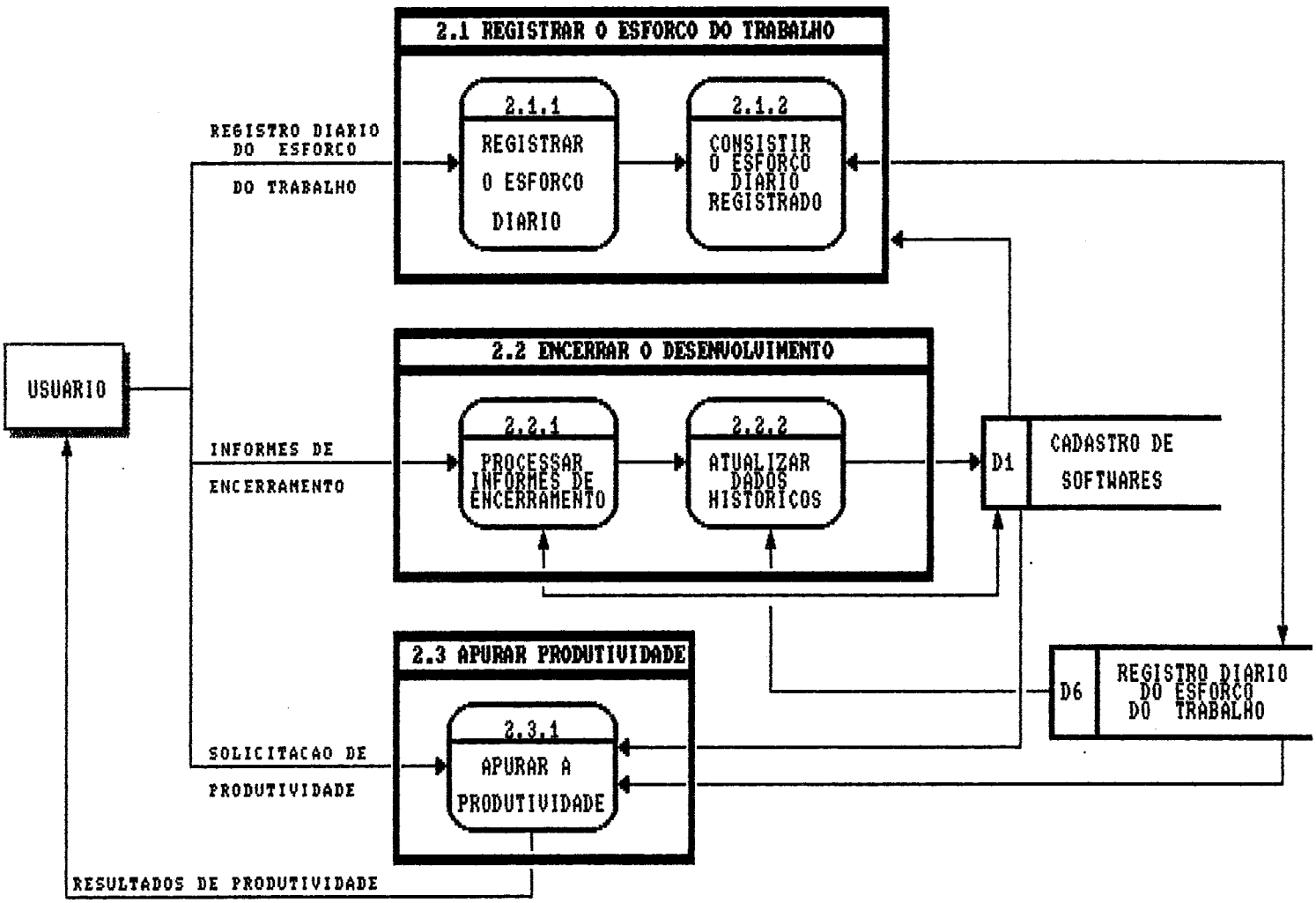


FIGURA 27: D.F.D. DO SUB-SISTEMA DE CALCULAR A PRODUTIVIDADE

VII.2.3 - DESCRIÇÃO DO SUB-SISTEMA DE ATUALIZAR AS TABELAS

Este sub-sistema (Figura 28) é responsável por fazer a manutenção cadastral dos diversos arquivos e tabelas usados pelo sistema.

SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE

3 - SUB-SISTEMA DE ATUALIZAR AS TABELAS

DIAGRAMA DE FLUXO DE DADOS

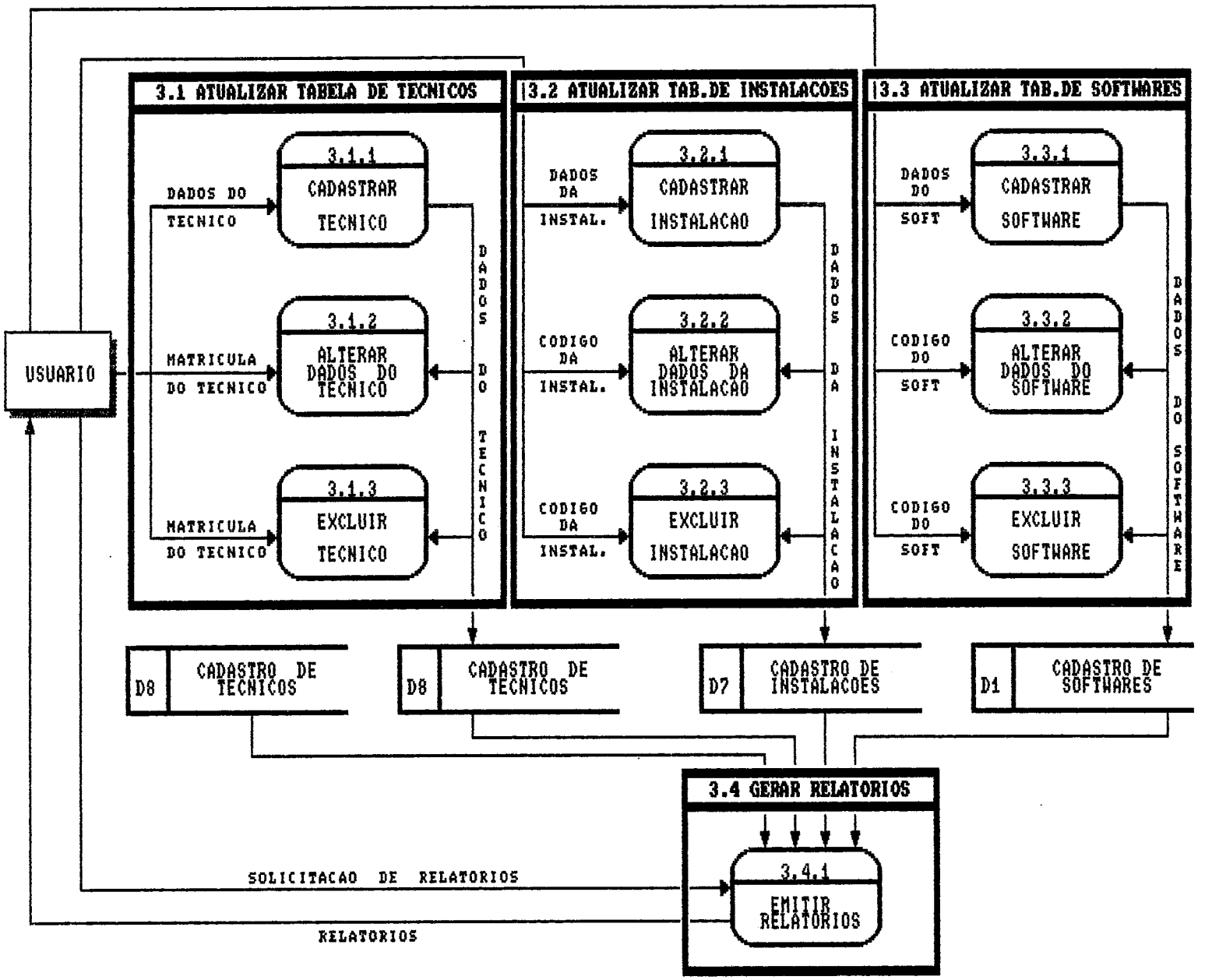


FIGURA 28: D.F.D. DO SUB-SISTEMA DE ATUALIZAR AS TABELAS

VII.2.4 - DESCRIÇÃO DO SUB-SISTEMA DE FAZER ESTIMATIVAS DO DESENVOLVIMENTO

Considerando as alternativas descritas no capítulo V, este sub-sistema (Figura 29) tem por objetivo determinar o esforço e o tempo total do projeto, das fases, das tarefas e das atividades. Este componente possui seis módulos que são:

Módulo de Calcular o Esforco Total (4.1)

Tal como descrito em V.1, este módulo objetiva determinar o Esforço Total necessário ao projeto. Possui uma unidade para Calcular o Esforço-Histórico (4.1.1), outra para Calcular o Esforço-Atividade (4.1.2) e outra para Calcular o Esforço-Fórmula (4.1.3).

Módulo de Calcular o Tempo Total (4.2)

Este módulo possui objetivo similar ao anterior sendo que voltado para Tempo Total, tal como já definido (4.2.1 a 4.2.3).

Módulo de Calcular a Demora das Fases (4.3)

Este módulo possui duas unidades. Uma para Calcular o Tempo-Total necessário a cada fase (4.3.1) e outra para Calcular o Esforço-Total necessário para cada fase (4.3.2).

Módulo de Calcular a Demora das Tarefas (4.4)

Este módulo possui duas unidades. Uma para Calcular o

Tempo-Total necessário a cada tarefa (4.4.1) e outra para Calcular o Esforço-Total necessário para cada tarefa (4.4.2).

Módulo de Calcular a Demora das Atividades (4.5)

Cada atividade, ao longo do desenvolvimento, também precisa ser prevista e avaliada, independente da fase na qual vai ocorrer. Este módulo tem por objetivo, através de uma unidade, Calcular o Tempo Total para cada atividade (4.5.1) e, através de outra unidade, Calcular o Esforço-Total para cada atividade (4.5.2).

Módulo de Emitir as Estimativas Calculadas (4.6)

Este módulo tem por objetivo emitir as Estimativas que foram calculadas no sub-sistema.

SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE

4 - SUB-SISTEMA DE FAZER ESTIMATIVAS DE DESENVOLVIMENTO

DIAGRAMA DE FLUXO DE DADOS

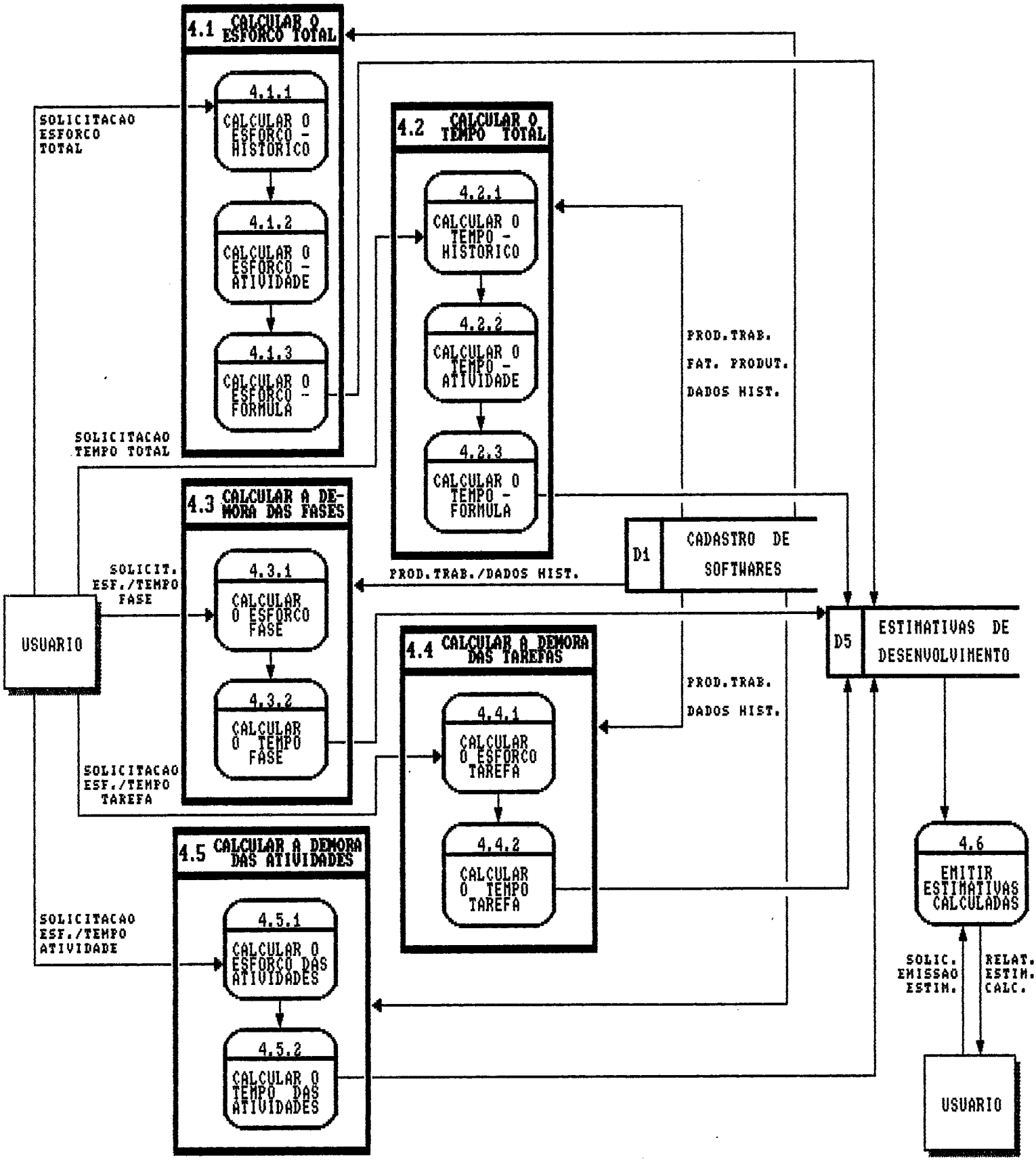


FIGURA 29: D.F.D. DO SUB-SISTEMA DE FAZER ESTIMATIVAS DE DESENVOLVIMENTO

VII.2.5 - DESCRIÇÃO DO SUB-SISTEMA DE FAZER O ACOMPANHAMENTO DO DESENVOLVIMENTO

O objetivo deste sub-sistema (Figura 30) é o de controlar o andamento do projeto à luz dos planos que foram gerados, dos tempos que foram previstos e dos marcos que foram estabelecidos. Baseado no que foi descrito em IV.3 ele possui três módulos:

Módulo de Produzir os Planos do Desenvolvimento (5.1)

Este módulo tem por objetivo gerar e gerenciar todos os planos de desenvolvimento do projeto. Possui uma unidade para Gerar os Planos do Projeto (5.1.1), para Agregar Informações Adicionais (5.1.2) e outra para Emitir esses esses Planos (5.1.3) e outra para Associar Informações Não Padrões (5.1.4)

Módulo de Avaliar o Andamento Geral (5.2)

Baseado no esforço real registrado este módulo se ocupa em comparar e avaliar os desvios do projeto com relação ao que foi planejado. Em uma de suas unidades há a comparação e Avaliação das Fases (5.2.1), em outra há a Avaliação das Tarefas (5.2.2) e, na última, há a Comparação e Avaliação das Atividades (5.2.3).

SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE
5 - SUB-SISTEMA DE ACOMPANHAR O DESENVOLVIMENTO
DIAGRAMA DE FLUXO DE DADOS

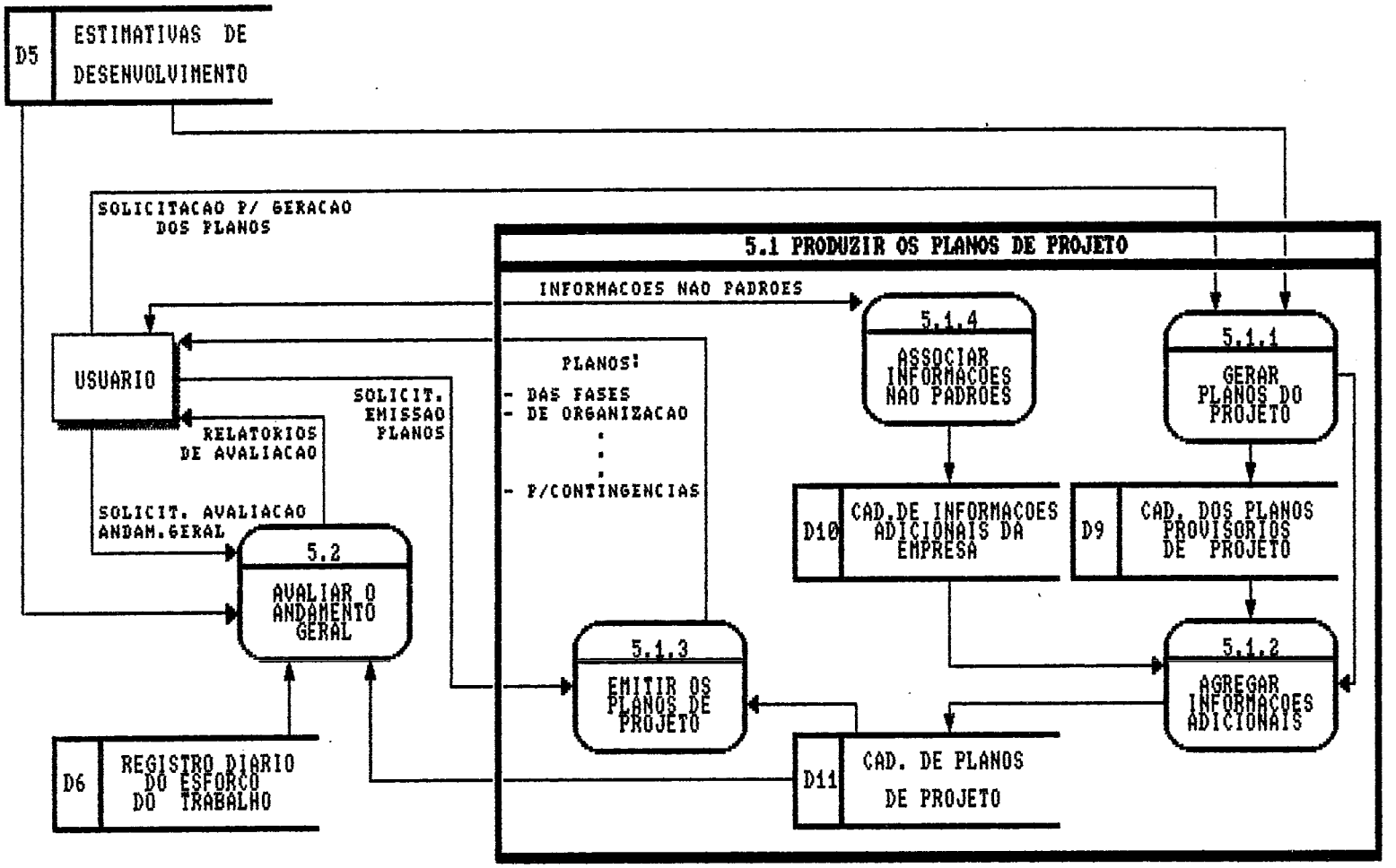


FIGURA 30: D.F.D. DO SUB-SISTEMA DE ACOMPANHAR O DESENVOLVIMENTO

VII.3 - IMPLEMENTAÇÃO DO AMBIENTE

VII.3.1 - INTRODUÇÃO

O ambiente descrito neste capítulo pode perfeitamente ser implementado em sua totalidade, gerando assim uma ferramenta útil para gerenciar o ambiente de produção de software .

Os sub-sistemas 1, 2 e 3 possuem todos seus requisitos descritos e completamente embasados neste trabalho. Por essa razão, e por possuirem objetivos associados aos do trabalho, foram implementados, estando as especificações e documentação disponíveis para interessados. O cálculo do tamanho da parte implementada, está incluido, a título de exemplo, no ANEXO-13.

Para os sub-sistemas 4 e 5 já existe conhecimento sistematizado na literatura e pode também ser objeto de implementação futura como continuidade deste trabalho.

VII.3.2 - CARACTERISTICAS DA IMPLEMENTAÇÃO FEITA

Não obstante estar fora do escopo deste trabalho, descrevemos nesta seção as principais características da implementação realizada.

a) METODOLOGIA UTILIZADA

A implementação foi feita utilizando-se uma metodologia de desenvolvimento intitulada SIM-SOFTWARE IMPLEMENTATION METODOLOGY (33)cuja características marcantes são:

- possue fases que se dividem em tarefas; tarefas que se dividem em sub-tarefas; sub-tarefas que se dividem em passos;

- possue abordagem estruturada, com refinamentos sucessivos e técnicas especificas associadas a cada atividade prevista, e
- conduz a uma documentação intensiva desde as fases iniciais, propiciando documentos completos ao final de cada fase executada.

b) DOCUMENTAÇÃO RESULTANTE

A documentação que resultou da implementação, decorre de impositivo da metodologia utilizada e constitui-se de:

- DEFINIÇÃO DE REQUISITOS - contendo o mapeamento de todas as características definidas no trabalho, como sendo necessárias ao ambiente;
- PROJETO CONCEITUAL - contendo uma abordagem global embasada no diagrama de fluxo dos dados do ambiente, incluso no anexo 12 e descrito neste capítulo;
- PROJETO DETALHADO - contendo especificações a nível de detalhe de todos os componentes do software, e
- CÓDIGO FONTE - contendo a listagem de todos os programas, módulos e procedimentos utilizados.

Não há ainda, nesta versão da implementação, manuais de usuários, de implantação e outros relacionados com a utilização em série.

c) LINGUAGEM, TAMANHO E AMBIENTE

Optou-se implementar o projeto detalhado em uma linguagem bastante difundida, utilizável em ambiente de micro-computação e que possuisse recurso de manipulação de base de dados. Por essa razão, utilizou-se o CLIPPER versão SUMMER 87 (1).

O tamanho do software, que foi calculado com a própria metodologia, encontra-se demonstrado no ANEXO-13.Inicialmente, na Definição de Requisitos,obteve-se uma grandeza de 197 pontos por função. Posteriormente, no projeto conceitual, chegou-se a um novo tamanho de 234 pontos por função.

Essa variação de expectativa de tamanho (19%), além do que seria aceitável com uso da metodologia (15%) segundo a característica "previsibilidade" da seção III.6, é explicável em face da falta de experiência no seu uso.

O código executável ficou em 356 KB e pode ser utilizado em IBM-PC e compatíveis com pelo menos 10 MB de discos magnéticos para armazenamento.

O software foi implementado com interface ostensivamente amigável, incluindo recursos de janelas, ajudas, filtros e navegação para acesso em estrutura de dados.

O esforço empregado na implementação foi de 1280 horas brutas redundando nos seguintes índices de produtividade:

. 31,62 pontos por função por mes-bruto/homem

1-CLIPPER é marca registrada da Nantucket, Inc.

. custou 5,30 horas-brutas para desenvolver cada ponto por função.

Esse significativo índice de produtividade registrado está associado com a presença de quatro fatores que, conforme tratado no capítulo VI são fortes influenciadores da produtividade. São eles:

- Linguagem de codificação;
- Tamanho do software;
- Métodos estruturados, e
- Recursos de codificação (foi usado como auxílio um gerador de código)

CAPITULO VIII

CONCLUSÃO

Conforme GILB(34) qualquer técnica de estimativa deve refletir um grande número de complexos e dinâmicos fatores e, porisso, nenhum modelo estático atenderia esse requisito, sobretudo porque, quase sempre a história dos trabalhos passados é ignorada com prejuizo para as estimativas presentes, uma vez que, " nós podemos usar o passado como um guia " testando e modificando suas assunções a cada novo desenvolvimento.

Entretanto, conforme CROSSMAN(21) se não tivermos interesse em medir produtividade, acabaremos por encontrar muitas razões, possivelmente válidas, que nos convencerão ser impossível. Entretanto, se houver interesse, embora com alguma dificuldade, acabaremos por conseguir.

Foi disso que este trabalho tratou embora apresentando modelos estáticos quanto à forma, foi possível refletir uma série de fatores dinâmicos, bem como, aproveitar-se dos históricos para projetar o futuro.

Além disso, todas as questões relativas à produtividade foram examinadas e colocadas sob uma visão global, ensejando assim, uma compreensão organizada de todos os aspectos que a permeiam.

Nesse particular aspecto, releva-se útil destacar o capítulo VII - Gerência da Produção de Software - Visão Geral de um Ambiente. Naquele capítulo, acredita-se estarem inclusos todos os aspectos relativos ao gerenciamento de desenvolvimento de software.

É possível, entretanto, que na visão geral apresentada, faltem ainda outros componentes de relevo que ainda não foram identificados. Essas necessidades surgirão à medida que a idéia básica ganhar outros desdobramentos e receber aprofundamentos em outros trabalhos.

Conforme referido no capítulo VII, três componentes dessa visão geral foram implementados. A implementação dos outros dois (Estimativas do Desenvolvimento e Acompanhamento do Desenvolvimento) ensejará a completa automatização do gerenciamento da produção de software.

Embora com todo esse escopo, o trabalho manteve uma rigorosa simplicidade, possibilitando assim, ferramentas efetivamente úteis e utilizáveis para gerenciar o processo de desenvolvimento e a medição da produtividade na produção de software.

REFERÊNCIAS

1 - ALBRECHT, Allan J.; " Measuring Application Development Productivity "; Proceedings of the Joint Share/Guide/IBM Application Development Symposium, Monterey, CA; October-1979; pp.83-92

2 - BURROUGHS ELETRONICA; Análise por Pontos de Funções - Manual do Estudante/BRDS - 140; Rio de Janeiro; Novembro-1985

3 - IBM CORPORATE INFORMATION SYSTEMS AND ADMINISTRATION; AD/M Productivity Measurement and Estimate Validation CIS & A Guideline 313; Purchase, NY; November-1984

4 - HALSTEAD; Maurice H.; Elements of Software Science; Elsevier North-Holland; New York; 1977

5 - JONES, Capers; Programming Productivity - Issues For the Eighties; IEEE Press; Silver Spring, MD; Cat.No. EHO 186-7; 1981; pp.221-224

6 - BELADY, L. and LEHMAN, M. M.; " A Model of Large Program Development "; IBM System Journal; Vol.15, NO.3, 1976; pp.225-252

7 - JONES, T. C.; " Measuring Programming Quality and Productivity "; IBM Systems Journal, Vol.17, NO.1, 1978; pp.39-63

8 - BOEHM, B.W.; Software Engineering Economics; Prentice-Hall, Inc.; Englewood Cliffs, NJ; 1981

9 - JONES, Capers; Programming Productivity; McGraw-Hill; New York; 1986

10 - CHRISTENSEN, K., FITSOS, G. P, and SMITH, C.P.; "A Perspective on Software Science "; IBM System Journal, Vol.20, No.4, 1981; pp.3-18

11 - FITSOS, G. P.; " Software Science Counting Rules and Tuning Methodology "; Technical Report Tr 03.025, IBM Santa Teresa Laboratory, 555 Bailey Avenue, P.O. Box 50020, San Jose, CA 95150 (SEP - 1979)

12 - ALBRECHT, Allan J., and GAFFNEY, John E.; " Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation "; IEEE Transaction on Software Engineering, Vol.SE-9, No.6, November-1983, pp.639-648

13 - BOHRER, Robert; " Halstead's Criterion and Statistical Algorithms "; Proceedings of the Eight Annual Computer Science/Statistics Interface Symposium, Los Angeles; February-1975, pp.262-266

14 - ELSHOFF, James L.; " Measuring Commercial PL/I Programs Using Halstead's Criteria "; General Motors Research Publication, GMR-2012; November-1975

15 - GORDON, R. D.; and HALSTEAD, M. H.; "An Experiment Comparing Fortran Programming Times with the Software Physics Hypothesis "; Computer Science Department Technical Report 16Z; Purdue University; October-1975

16 - FITSOS, G. P.; " Vocabulary Effects in Software Science "; Thecnical Report IR 03.082; IBM Santa Teresa Laboratory, 555 Bailey Avenue, P.O. Box 50020, San Jose, CA 95150 (January-1980)

17 - SMITH, C. P.; " A Software Science Analysis of IBM Programming Products "; Iecnical Report IR 03.081; IBM Santa Teresa Laboratory, 555 Bailey Avenue, P.O. Box 50020, San Jose, CA 95150 (January - 1980)

18 - CURTIS, Bill; " Software Metrics: Guest Editor's Introduction "; IEEE Iransaction on Software Engineering, Vol.SE-9, No.6, November-1983; pp.637-638

19 - McCABE, T.; " A Software Complexity Measure "; IEEE Iransactions on Software Engineering, Vol.SE-2; December-1976; pp.308-320

20 - PRESSMAN, R.; Software Engineering - A Practioner's Approach; McGraw-Hill; New York; 1982

21 - CROSSMAN, Trevor; " Taking the Measure of Programmer Productivy "; Datamation, May-1979; pp.144-147

22 - WALTSON, C. E. and FELIX, C.P.; " A Method of Programming Measurement and Estimation "; IBM System Journal, Vol.16, No.1 (1977); pp.54-73

23 - RUDOLPH, E.; " Productivity in Computer Application Development "; Working Paper No9 - Depto of Management Studies - Auckland University, New Zeland; March-1983

24 - RUDOLPH, E.; " Software Development Productivity - How to Measure it "; NZ Interface, April-1983; pp.46-49

25 - VACCA, John; " Fuction Points: Uma Nova Medida de Software", Data News, 18.03.86; pp.28-34

26 - KUNKLER, J. E.; " A Cooperative Industry Study: Software Development Productivity "; Xerox Corporation, NY; May-1983

27 - ALBRECHT, Allan J.; " Pontos de Função Auxiliam Gerentes na Avaliação de Avaliações "; Computer World - 26.08.85

28 - DRUMMOND, Steve; " Measuring Applications Development Performance; Datamation, FEB. 15, 1985; pp.102-108

29 - RUDOLPH, E.; " Measuring Software Development Productivity "; VIII Conferencia NZ de

Computadores; September-1983

30 - SIMPSON, G. and RUDOLPH, E.; " Evaluation of a Fourth Generation Language "; VIII Conferencia NZ de Computadores, September-1983

31 - BEHRENS, Charles A.; " Measuring the Productivity of Computer Systems Development Activities with Function Points "; IEEE Iransaction on Software Engineering, Vol.SE-9, No.6, November 1983 pp.648-652

32 - LAMBERT, G. N.; " A Comparative Study of System Response Time on Program Developer Productivity "; IBM Systems Journal, Vol.23, No.1, 1984; pp.36-43

33 - BURROUGHS CORPORATION; Software Implementation Methodology (SIM), Reference Manual; Proprietary Data; Detroit, Michigan- USA; September 1985.

34 - GILB, Tom; " Estimating Software Attributes: Some Unconventional Points of View "; ACM SIGSOEI SOEIWARE ENGINEERING NOIES, Vol.11, No.1; Jan-1986; pp.49-59.

35 - BROOKS, F.P.JR; Mythical Man-Month; Addison Wesley; Reading; MA ; 1982.

36 - CURTIS, B; " Substantiating Programmer Varability"; Proceeding of the IEEE, vol.69, No.7, July - 1981.

37 - MYERS, G.J; Composite/Structured Design; Van Nostrand Reinhold; New York; 1978.

38 - STEVENS, W.J; Using Structured Design; Willey; New York; 1981.

39 - CONSTANTINE, L.L. e YOURDON, E; Structured Design;Prentice - Hall; Englewood Cliffs, NJ; 1979.

40 - WARNIER, J.D; Logical Construction of Systems; Van Nostrand Reinhold; New York; 1981.

41 - JACKSON, Michael A; Systems Development; Englewood Cliffs, NJ; Prentice-hall; 1983.

42 - GANE, C.e SARSON, T; Structured Systems Analysis: Tools and Techniques; Prantice - Hall; Englewood Cliffs, NJ;1979.

43 - DIJKSTRA, E.;"Goto Statements Considered Harmful"; Comunication of the ACM, vol 11, No 9, novembro 1968; PP.147-148.

44 - WIRTH, N.; Algoritms + Data Structures = Programs;Prentice Hall; Englewood Cliffs, NJ; 1976.

45 - AGUIAR, T.C.; Ferramentas de Apoio a Análise Estruturada; Tese de Mestrado; IME - Rio de Janeiro; fevereiro 1986.

46 - BLASCHECK, J.R.S; Dicionário de Dados Automatizado para Análise Estruturada; Tese de Mestrado; IME - Rio de Janeiro; fevereiro - 1987.

47 - SANTOS, C.J.; Um Ambiente de Apoio Automatizado para o Desenvolvimento de Software Básico; Tese de Mestrado; COPPE/UFRJ Rio de Janeiro; 1987.

48 - ROCHA, ANA R.C.,e SOUZA, Jano M.;"Taba: Uma Estação de Trabalho para o Engenheiro de Software"; Retatório Técnico ES-145/88;COPPE/UFRJ - Rio de Janeiro; 1988.

50 - ROCHA, ANA R.C.; e SOUZA, Jano M.; (editores) Ambientes de Desenvolvimento de Software e Projeto TABA; Relatorio de Seminario ES - 179/88; COPPE/UFRJ - Rio de Janeiro; dezembro - 1988.

51 - UNISYS CORPORATION; " The Productivity Advantage "; EUA; 1988.

52 - VALE, M.A. de Oliveira; " Um Estudo sobre o Uso de Prototipos Rápidos no Desenvolvimento de Projetos de Software"; Monografia de Engenharia de Software apresentada na UFRJ/COPPE/PROGRAMA DE ENG. DE SISTEMA E COMPUTAÇÃO; 1986.

53 - SCHARER; L.L.; " Prototyping in a Production Environment "; ITT - Conference on Programing Productivity,junho-83; pp.440-455.

54 - INTERNATIONAL DATA CORPORATION. Systems Development Productivity; Waltham,MA;janeiro - 1981;PP. 56-60.

55 - JONES,CAPERS; " Pontos por Função: Nova Visão das Linguagens de Programação "; Data News, Ano XIII, N. 450; 03.04.89; pp. 12-14.

56 - LEITE, Julio Cesar S.P.; Contabilizaçao de Custos no Desenvolvimento de Software; Dissertação de Mestrado; PUC-Rio de Janeiro; 1979.

ANEXO - 1

CÁLCULO DE PONTOS POR FUNÇÃO **TIPO DE FUNÇÃO**

IDENTIFICAÇÃO DE FUNÇÕES ____ ______________

SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	NOME

CÓDIGO FUNÇÃO	DESCRIÇÃO	COMPLEX INICIAL	FORMA INCLUSÃO	SITUAÇÃO ATUAL	COMPLEX APÓS	OBSERVAÇÕES

REALIZADO	REVISADO
EM ___/___/___ POR ______________	EM ___/___/___ POR ______________

ANEXO - 2

CÁLCULO DE PONTOS POR FUNÇÃO

SOFTWARE EXISTENTE

SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	NOME

CÓDIGO DO TIPO	TIPO DE FUNÇÃO	COMPLEXIDADE ASSOCIADA			TOTAL
		SIMPLES	MÉDIA	COMPLEXA	
EE	ENTRADA EXTERNA	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	___
SE	SAÍDA EXTERNA	___ x 4 = ___	___ x 5 = ___	___ x 7 = ___	___
AI	ARQ. LÓG. INTERNO	___ x 7 = ___	___ x 10 = ___	___ x 15 = ___	___
IE	INTERFACE EXTERNA	___ x 5 = ___	___ x 7 = ___	___ x 10 = ___	___
CE	CONSULTA EXTERNA	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	___
				PONTOS POR FUNÇÃO BRUTOS (PFB) =	___

COD.	CARACTERÍSTICA GERAL	GRAU DE INFLUÊNCIA	COD.	CARACTERÍSTICA GERAL	GRAU DE INFLUÊNCIA
C1	COMUNICAÇÃO DE DADOS	___	C8	ATUALIZAÇÃO ON-LINE	___
C2	FUNÇÕES DISTRIBUÍDAS	___	C9	PROCESSAMENTO COMPLEXO	___
C3	OBJETIVOS DE PERFORMANCE	___	C10	REUTILIZABILIDADE	___
C4	CARGA DA CONFIGURAÇÃO	___	C11	FACILIDADE INSTALAÇÃO	___
C5	VOLUME DE TRANSAÇÕES	___	C12	FACILIDADE OPERAÇÃO	___
C6	ENTRADA DE DADOS ON-LINE	___	C13	MULTIPLICIDADE LOCAIS	___
C7	EFICIÊNCIA USUÁRIO FINAL	___	C14	FACILIDADE MUDANÇA	___
				SOMA DOS GRAUS DE INFLUÊNCIA (SGI) =	___

FATOR DE AJUSTE EXISTENTE (FAE) = 0,65 + (0,01 x SGI) = ___

REALIZADO	REVISADO
EM ___/___/___ POR ___	EM ___/___/___ POR ___

ANEXO - 3

CÁLCULO DE PONTOS POR FUNÇÃO						PRODUTO DO TRABALHO DO DESENVOLVIMENTO
SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	N O M E

	COMPLEXIDADE	ENTRADAS EXTERNAS	SAÍDAS EXTERNAS	ARQ. LÓGICOS INTERNOS	INTERFACES EXTERNAS	CONSULTAS EXTERNAS
INCLUÍDOS	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS INCLUÍDOS (PFB_{INC}) = _____

	COMPLEXIDADE	ENTRADAS EXTERNAS	SAÍDAS EXTERNAS	ARQ. LÓGICOS INTERNOS	INTERFACES EXTERNAS	CONSULTAS EXTERNAS
ALTERADOS (DEPOIS CONCLUSÃO)	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS ALTERADOS (DEPOIS CONCLUSÃO) ($PFB_{ALT-DEP}$) = _____

	COMPLEXIDADE	ENTRADAS EXTERNAS	SAÍDAS EXTERNAS	ARQ. LÓGICOS INTERNOS	INTERFACES EXTERNAS	CONSULTAS EXTERNAS
EXCLUÍDOS (ANTES INÍCIO)	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS EXCLUÍDOS (PFB_{EXC}) = _____

CÓD.	CARACTERÍSTICA GERAL	GRAU INFL. ANT.	GRAU INFL. DEP.	CÓD.	CARACTERÍSTICA GERAL	GRAU INFL. ANT.	GRAU INFL. DEP.
C1	COMUNICAÇÃO DE DADOS	___	___	C8	ATUALIZAÇÃO ON-LINE	___	___
C2	FUNÇÕES DISTRIBUÍDAS	___	___	C9	PROCESSAMENTO COMPLEXO	___	___
C3	OBJETIVOS DE PERFORMANCE	___	___	C10	REUTILIZABILIDADE	___	___
C4	CARGA DA CONFIGURAÇÃO	___	___	C11	FACILIDADE INSTALAÇÃO	___	___
C5	VOLUME DE TRANSAÇÕES	___	___	C12	FACILIDADE OPERAÇÃO	___	___
C6	ENTRADA DE DADOS ON-LINE	___	___	C13	MULTIPLICIDADE LOCAIS	___	___
C7	EFICIÊNCIA USUÁRIO FINAL	___	___	C14	FACILIDADE MUDANÇA	___	___

REALIZADO:

EM ____/____/____

POR _____

REVISADO:

EM ____/____/____

POR _____

SOMA DOS GRAUS DE INFLUÊNCIA-ANTES (SGI_{ANT}) = _____

SOMA DOS GRAUS DE INFLUÊNCIA-DEPOIS (SGI_{DEP}) = _____

FATOR DE AJUSTE EXISTENTE-ANTES (FAE_{ANT}) :
$FAE_{ANT} = 0,65 + (0,01 \times SGI_{ANT})$ = _____

FATOR DE AJUSTE EXISTENTE-DEPOIS (FAE_{DEP}) :
$FAE_{DEP} = 0,65 + (0,01 \times SGI_{DEP})$ = _____

PRODUTO DO TRABALHO DO DESENVOLVIMENTO
$PFB = (PFB_{INC} + PFB_{ALT-DEP}) \times FAE_{DEP} + (PFB_{EXC} \times FAE_{ANT})$ = _____

ANEXO - 4

CÁLCULO DE PONTOS POR FUNÇÃO	PRODUTO DO TRABALHO DE MANUTENÇÃO

SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	NOME

COMPLEXIDADE		ENTRADAS EXTERNAS	SAÍDAS EXTERNAS	ARQ. LÓGICOS INTERNOS	INTERFACES EXTERNAS	CONSULTAS EXTERNAS
INCLUÍDOS	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS INCLUÍDOS (PFB_{INC}) = ___

ALTERADOS (ANTES INÍCIO)						
	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS ALTERADOS (ANTES INÍCIO) ($PFB_{ALT-ANT}$) = ___

ALTERADOS (DEPOIS CONCLUSÃO)						
	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS ALTERADOS (DEPOIS CONCLUSÃO) ($PFB_{ALT-DEP}$) = ___

EXCLUÍDOS (ANTES INÍCIO)						
	SIMPLES	___ x 3 = ___	___ x 4 = ___	___ x 7 = ___	___ x 5 = ___	___ x 3 = ___
	MÉDIA	___ x 4 = ___	___ x 5 = ___	___ x 10 = ___	___ x 7 = ___	___ x 4 = ___
	COMPLEXA	___ x 6 = ___	___ x 7 = ___	___ x 15 = ___	___ x 10 = ___	___ x 6 = ___
	TOTAL	___	___	___	___	___

PONTOS POR FUNÇÃO BRUTOS EXCLUÍDOS (PFB_{EXC}) = ___

CÓD.	CARACTERÍSTICA GERAL	GRAU INFL. ANT.	GRAU INFL. DEP.	CÓD.	CARACTERÍSTICA GERAL	GRAU INFL. ANT.	GRAU INFL. DEP.
C1	COMUNICAÇÃO DE DADOS	___	___	C8	ATUALIZAÇÃO ON-LINE	___	___
C2	CONJUNTOS DISTRIBUÍDOS	___	___	C9	PROCESSAMENTO COMPLEXO	___	___
C3	DESEMPENHO	___	___	C10	REUTILIZABILIDADE	___	___
C4	CARGA DA CONFIGURAÇÃO	___	___	C11	FACILIDADE INSTALAÇÃO	___	___
C5	VOLUME DE TRANSAÇÕES	___	___	C12	FACILIDADE OPERACIONAIS	___	___
C6	ENTRADA DE DADOS ON-LINE	___	___	C13	FACILIDADE MUDANÇA	___	___
C7	EFICIÊNCIA USUÁRIO FINAL	___	___	C14		___	___

REALIZADO:
EM ___/___/___
POR ___

REVISADO:
EM ___/___/___
POR ___

SOMA DOS GRAUS DE INFLUÊNCIA-ANTES (SGI_{ANT}) = ___
SOMA DOS GRAUS DE INFLUÊNCIA-DEPOIS (SGI_{DEP}) = ___

FATOR DE AJUSTE EXISTENTE-ANTES (FAE_{ANT}) :
$FAE_{ANT} = 0,65 + (0,01 \times SGI_{ANT})$ = ___

FATOR DE AJUSTE EXISTENTE-DEPOIS (FAE_{DEP}) :
$FAE_{DEP} = 0,65 + (0,01 \times SGI_{DEP})$ = ___

PRODUTO DO TRABALHO DE MANUTENÇÃO :
$PFS = PF_{ANT} + (PFB_{INC} + PFB_{ALT-DEP}) \times FAE_{DEP} - (PFB_{EXC} + PFB_{ALT-ANT}) \times FAE_{ANT}$ = ___

ANEXO - 5

CÓDIGO DA INSTALAÇÃO = ______	REGISTRO DIÁRIO DO ESFORÇO DO TRABALHO	DATA
MATRIC. FUNCIONÁRIO = ______		___/___/___
NOME DO FUNCIONÁRIO = ______		

PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PRO-GRAMA	NATU-REZA	FASE	TA-REFA	ATIVI-DADE	HORA INÍCIO	HORA FIM	OBSERVAÇÕES	VISTO DA CHEFIA

ANEXO – 6

NÍVEL = ________

TIPO = ________

RESUMO DO ESFORÇO DO TRABALHO

INSTALAÇÃO ________

DATA DE INÍCIO ___/___/___

DATA DE CONCLUSÃO ___/___/___

ANO-BASE ________

SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	NOME

ATIVIDADE	FASES										HORAS	HORAS	MESES
	001	002	003	004	005	006	007	008	009	010	LÍQUIDAS	BRUTAS	BRUTOS
TOTAL HORAS LÍQ.													
TOTAL HORAS BRUTAS													
TOTAL MESES BRUTOS													

ANEXO - 7

RELATÓRIO DE PRODUTIVIDADE - RESUMO FINAL

NÍVEL = _____

TIPO = _____

SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	N O M E

DATA DE INÍCIO: ___/___/___

DATA DE CONCLUSÃO: ___/___/___

MATRÍCULA	R E S P O N S Á V E L

ANO BASE

PRODUTO DO TRABALHO - TAMANHO FINAL	ESFORÇO DO TRABALHO		
	HORAS LÍQUIDAS	HORAS BRUTAS	MESES BRUTOS

RESULTADO DE DESENVOLVIMENTO

LOCAIS IMPLANTADOS : _____

$$\text{PRODUTIVIDADE} = \frac{\text{PRODUTO DO TRABALHO}}{\text{ESFORÇO DO TRABALHO (MESES BRUTOS)}} =$$

" A PRODUTIVIDADE NO DESENVOLVIMENTO DESTE SOFTWARE FOI DE _____ PONTOS POR FUNÇÃO POR MES-BRUTO/HOMEM ".

$$\text{CUSTO} = \frac{\text{ESFORÇO DO TRABALHO (HORAS BRUTAS)}}{\text{PRODUTO DO TRABALHO}} =$$

" CUSTOU _____ HORAS-BRUTAS/HOMEM PARA DESENVOLVER CADA PONTO POR FUNÇÃO DESTE SOFTWARE ".

RESULTADO DE MANUTENÇÃO/SUPORTE

LOCAIS IMPLANTADOS : _____

$$\text{PRODUTIVIDADE} = \frac{\text{PRODUTO DO TRABALHO}}{\text{ESFORÇO DO TRABALHO (HORAS BRUTAS)}} =$$

" A PRODUTIVIDADE NA MANUTENÇÃO DESTE SOFTWARE FOI _____ PONTOS POR FUNÇÃO POR HORA-BRUTA/HOMEM NO PERÍODO DE UM ANO ".

$$\text{CUSTO} = \frac{\text{ESFORÇO DO TRABALHO (HORAS BRUTAS)}}{\text{PRODUTO DO TRABALHO}} =$$

" CUSTOU _____ HORAS-BRUTAS/HOMEM PARA MANTER CADA PONTO POR FUNÇÃO DESTE SOFTWARE NO PERÍODO DE UM ANO ".

_____ FEITO POR

_____ REVISTO POR

OBSERVAÇÃO IMPORTANTE :

VIDE NO VERSO A ANÁLISE DOS FATORES QUE JUSTIFICAM OS RESULTADOS ACIMA.

NATUREZA:	ANÁLISE INDIVIDUALIZADA DA PRODUTIVIDADE	ANO BASE
DESENVOLVIMENTO [ ]		______
EXPANSÃO [ ]		NÍVEL
MANUTENÇÃO [ ]		______

SOFTWARE		RESPONSÁVEL	VIDE OBSERVAÇÃO ABAIXO	
CÓDIGO	NOME		PRODUTIVIDADE	CUSTO

______
FEITO POR

______
REVISTO POR

OBS.: Os resultados de desenvolvimento devem ser lidos como PFs POR MES-BRUTO/HOMEM e HORAS-BRUTAS/HOMEM POR CADA PF. Os resultados de manutenção como PFs POR HORA-BRUTA/HOMEM e HORAS-BRUTAS/HOMEM POR CADA PF, no ano-base.

ANEXO - 10

FATORES DE PRODUTIVIDADE - ESTIMATIVA DE INFLUENCIA

SOFTWARE	PROJETO OU SISTEMA	APLICAÇÃO OU SUB-SIST.	ROTINA OU MÓDULO	PROGRAMA	NATUREZA	NOME

FATOR DE PRODUTIVIDADE		INFLUENCIA		
CÓDIGO	NOME	PREVISTA	PESO	TOTAL
001	LINGUAGEM DE CODIFICAÇÃO		5	
002	TAMANHO DO SOFTWARE		5	
003	EXPERIENCIA DA EQUIPE		4	
004	MÉTODOS ESTRUTURADOS		3	
005	AMBIENTE E FERRAMENTAS		4	
006	QUALIDADE DA EXPANSÃO		3	
007	REUTILIZAÇÃO DE CÓDIGO		1	
008	CIRCUNSTÂNCIAS GEOGRÁFICAS		4	
009	MÉTODOS DE REMOÇÃO DE ERROS		2	
010	RECURSOS DE CODIFICAÇÃO		3	
011	ORGANIZAÇÃO DA EQUIPE		1	
012	SATISFAÇÃO DA EQUIPE		2	

$$\text{Fator de Produtividade (FP)} = \frac{\text{TOTAL DAS INFLUENCIAS}}{\text{SOMA DOS PESOS}}$$

OBS.: Quando for desenvolvimento novo o item 006 deve ser tido como 0,99.

FEITO POR	REVISTO POR	TOTAL		

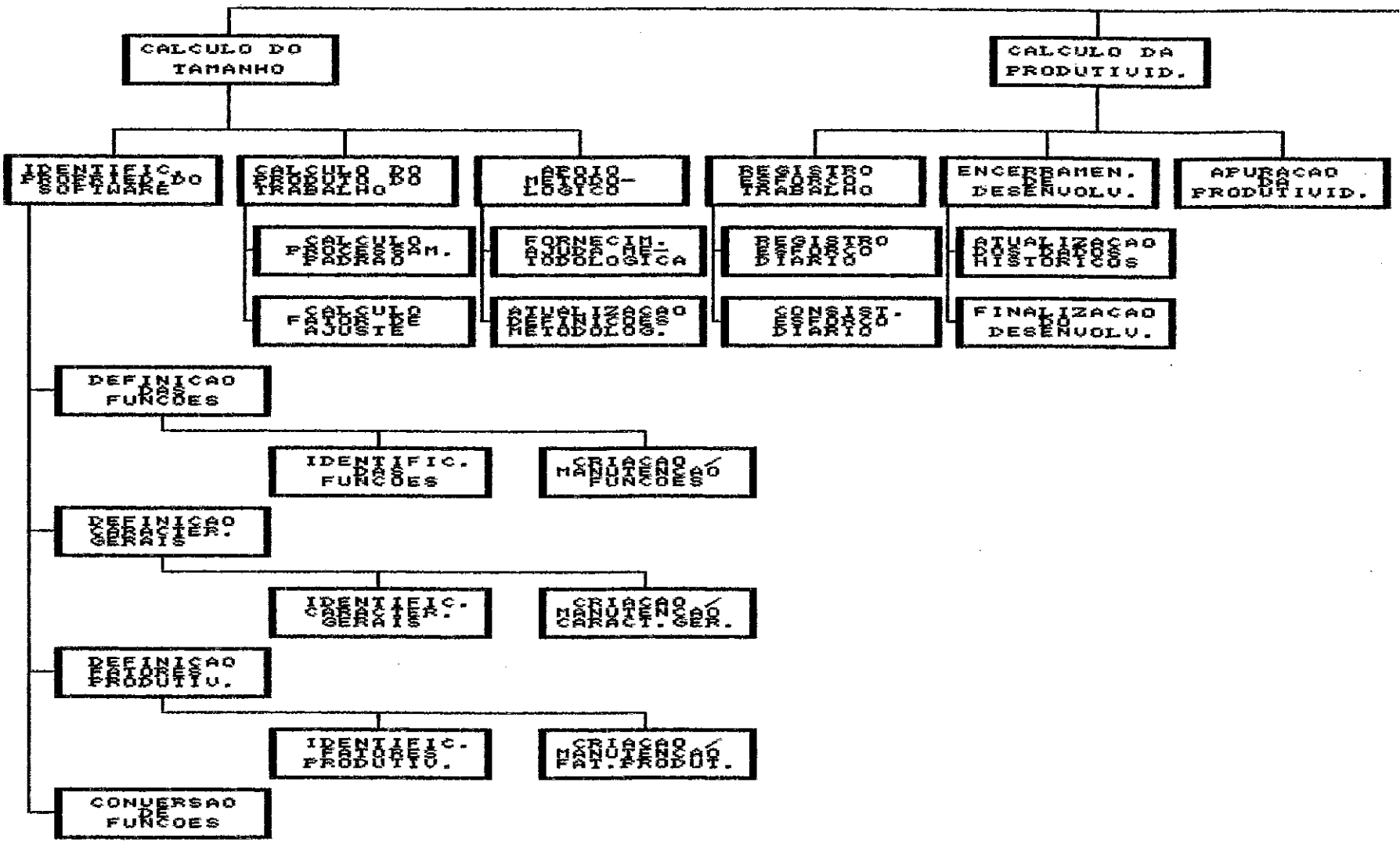
ANÁLISE EVOLUTIVA GLOBAL DA PRODUTIVIDADE

ANO	DESENVOLVIMENTO NOVO		EXPANSÃO		MANUTENÇÃO	
	PRODUTIVIDADE	CUSTO	PRODUTIVIDADE	CUSTO	PRODUTIVIDADE	CUSTO

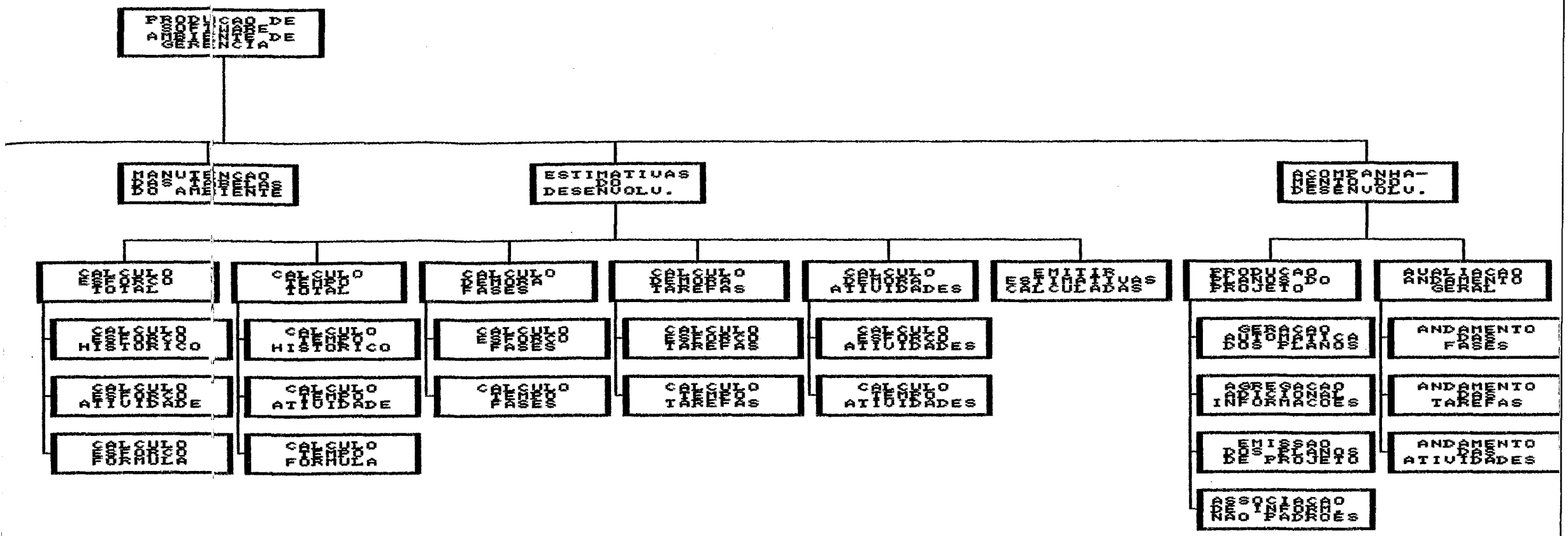
OBS.: Os resultados de desenvolvimento devem ser lidos como PFs POR MES-BRUTO/HOMEM e HORAS-BRUTAS/HOMEM POR CADA PF. Os resultados de manutenção como PFs POR HORA-BRUTA/HOMEM e HORAS-BRUTAS/HOMEM POR CADA PF, no ano-base.

GERÊNCIA DA PRODUÇÃO DE SOFTWARE

VISÃO GERAL DO AMBIENTE



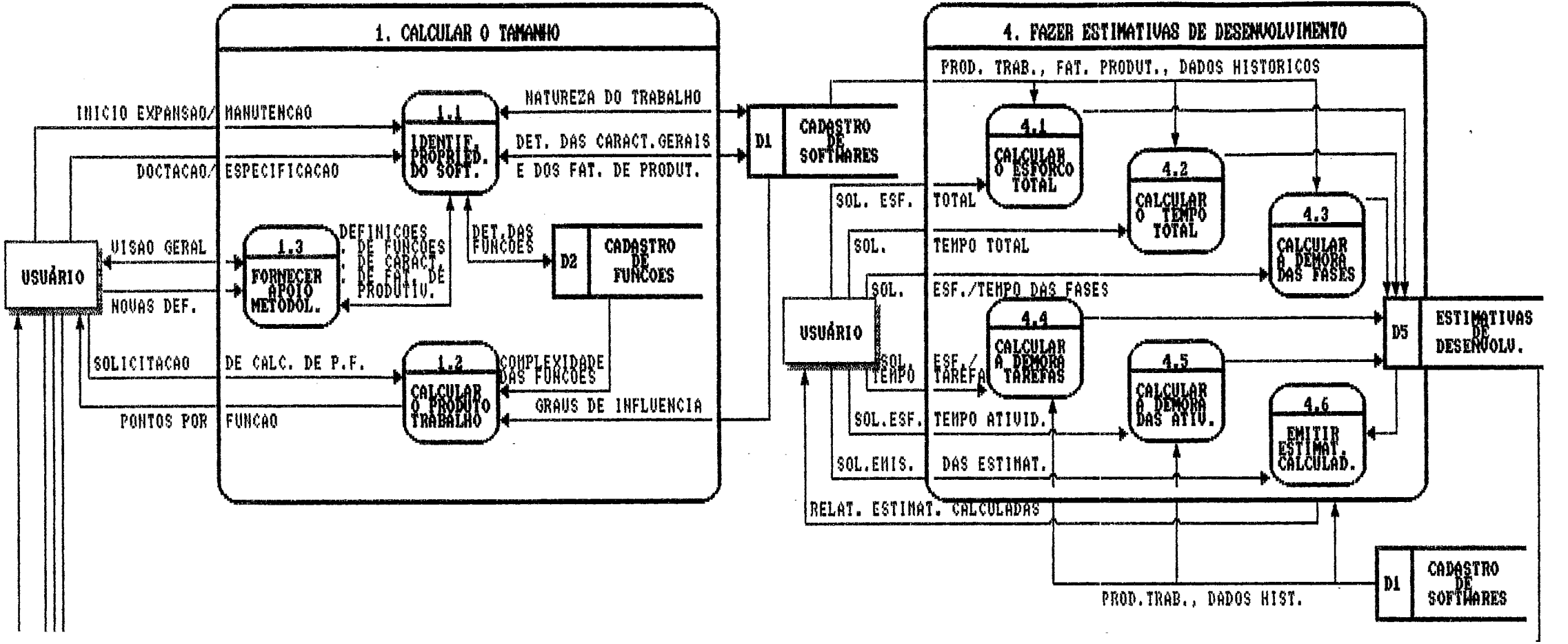
ANEXO - 11

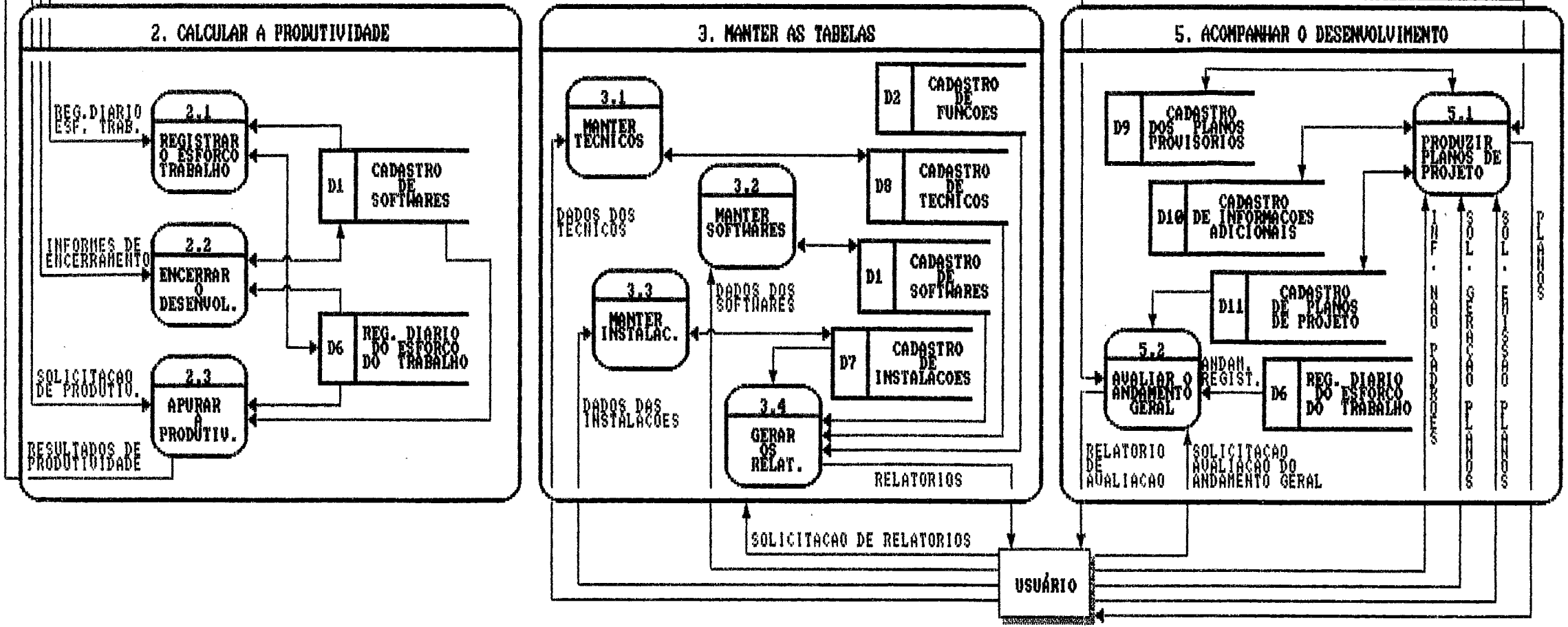

PRODUCAO DE SOFTWARE DE AMBIENTE DE GERENCIA
MANUTENCAO DAS TABELAS DO AMBIENTE
ESTIMATIVAS DO DESENVOLV.
ACOMPANHA-MENTO DO DESENVOLV.
CALCULO ESFORCO TOTAL
CALCULO TEMPO TOTAL
CALCULO PENOBA FASES
CALCULO PENOBA TAREFAS
CALCULO PENOBA ATIVIDADES
EMITIR ESTIMATIVAS CALCULADAS
CALCULO ESFORCO HISTORICO
CALCULO ESFORCO ATIVIDADE
CALCULO ESFORCO FORMULA
CALCULO TEMPO HISTORICO
CALCULO TEMPO ATIVIDADE
CALCULO TEMPO FORMULA
CALCULO ESFORCO FASES
CALCULO TEMPO FASES
CALCULO ESFORCO TAREFAS
CALCULO TEMPO TAREFAS
CALCULO ESFORCO ATIVIDADES
CALCULO TEMPO ATIVIDADES
EFORUCAO PLANOS DO PROJETO
AVALIACAO ANDAMENTO GERAL
GERACAO AUTOMATICA DOS PLANOS
AGREGACAO APLICACOES INFORMACOES
EMISSAO DOS PLANOS DE PROJETO
ASSOCIACAO DE INFORM. NAO PADROES
ANDAMENTO DAS FASES
ANDAMENTO DAS TAREFAS
ANDAMENTO DAS ATIVIDADES

ANEXO-12

SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE

DIAGRAMA GERAL DO FLUXO DOS DADOS




2. CALCULAR A PRODUTIVIDADE
REG.DIARIO ESF. TRAB.
2.1 REGISTRAR O ESFORCO TRABALHO
D1 CADASTRO DE SOFTWARES
INFORMES DE ENCERRAMENTO
2.2 ENCERRAR O DESENVOL.
D6 REG. DIARIO DO ESFORCO DO TRABALHO
SOLICITACAO DE PRODUTIV.
2.3 APURAR A PRODUTIV.
RESULTADOS DE PRODUTIVIDADE
3. MANTER AS TABELAS
3.1 MANTER TECNICOS
D2 CADASTRO DE FUNCOES
D8 CADASTRO DE TECNICOS
DADOS DOS TECNICOS
3.2 MANTER SOFTWARES
D1 CADASTRO DE SOFTWARES
DADOS DOS SOFTWARES
3.3 MANTER INSTALAC.
D7 CADASTRO DE INSTALACOES
DADOS DAS INSTALACOES
3.4 GERAR OS RELAT.
RELATORIOS
SOLICITACAO DE RELATORIOS
5. ACOMPANHAR O DESENVOLVIMENTO
D9 CADASTRO DOS PLANOS PROVISORIOS
5.1 PRODUZIR PLANOS DE PROJETO
D10 CADASTRO DE INFORMACOES ADICIONAIS
D11 CADASTRO DE PLANOS DE PROJETO
5.2 AVALIAR O ANDAMENTO GERAL
ANDAN. REGIST.
D6 REG. DIARIO DO ESFORCO DO TRABALHO
RELATORIO DE AVALIACAO
SOLICITACAO AVALIACAO DO ANDAMENTO GERAL
INF. NAO PADROES
SOL. GERACAO PLANOS
SOL. EMISSAO PLANOS
PLANOS
USUÁRIO

A N E X O - 1 3

EXEMPLO DE DETERMINAÇÃO DE PONTOS POR FUNÇÃO

1 - INTRODUÇÃO

Com objetivo de demonstrar a determinação Pontos por Função sera apresentado neste anexo um exemplo real de sua utilização.

No capítulo VII foi apresentado um ambiente para gerência da produção de software. Parte desse ambiente foi implementada com base nas especificações e diagramas incluídos naquele capítulo.

O cálculo do tamanho da parte implementada foi incluido, de forma detalhada, na documentação (Projeto Conceitual) do software resultante.

Neste anexo, o cálculo acima sera demonstrado. Entretanto, conforme a seção IV.2.1, esse cálculo deve ser feito com base nas especificações e, como a especificação da parte implementada não esta incluida neste trabalho, a identificação e classificação das funções podem não parecer tão claras.

De qualquer maneira, cada função identificada e classificada, será associada aos elementos componentes do DFD incluso no capítulo VII.

2 - IDENTIFICAÇÃO E CLASSIFICAÇÃO DAS ENTRADAS

Foram revisadas as seguintes entradas do sistema, identificadas na Definição de Requisitos, com seus respectivos propósitos, atributos e complexidade:

a) **NOME DA ENTRADA:** Cadastrar funções

I) Propósito: Alimentar o sistema com as funções do software.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
CODIGO DO TIPO DE FUNÇÃO	02
CODIGO DA TRANSAÇÃO	08
NOME DA TRANSAÇÃO	20
DETALHE DE COMPLEX. DEPOIS	01

III) Complexidade: MÉDIA
5 Campos
2 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Funções).

IV)Fonte da Identificação: Processo 1.1.1

b) **NOME DA ENTRADA:** Alterar funções

I) Proposito: Possibilitar ao usuário a alteração dos atributos da função identificada.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
CODIGO DO TIPO DE FUNÇÃO	02
CODIGO DA TRANSAÇÃO	08
NOME DA TRANSAÇÃO	20
DETALHE DE COMPLEX. DEPOIS	01

III) Complexidade: MÉDIA
5 Campos
2 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Funções)

IV) Comentarios: Através do acesso ao Cadastro de Funções, a partir dos campos-chave, todo o conteudo do registro e editado, computando-se como Entrada Externa os dados a serem alterados.

V)Fonte da Identificação: Processo 1.1.1

c) **NOME DA ENTRADA:** Excluir funções

I) Propósito: Permitir ao usuário a exclusão de funções.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
CODIGO DO TIPO DE FUNÇÃO	02
CODIGO DA TRANSAÇÃO	08

III) Complexidade: SIMPLES
3 Campos
2 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Funções)

IV)Fonte da Identificação: Processo 1.1.1

d) **NOME DA ENTRADA:** Alterar características gerais

I) Propósito: Possibilitar ao usuário a alteração dos graus de influência das diversas características gerais do software.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
GRAU INFL. CARACTERÍSTICA	01

III) Complexidade: SIMPLES
15 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

IV) Comentarios: Através do acesso ao Cadastro de Softwares, a partir do campo-chave, computa-se como Entrada Externa os dados passiveis de alteração.

V)Fonte da Identificação: Processo 1.1.3

e) **NOME DA ENTRADA:** Alterar fatores de produtividade

I) Propósito: Permitir ao usuário a alteração dos graus de influência dos fatores de produtividade identificados.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
GRAU INFLUÊNCIA DO FATOR	04

III) Complexidade: SIMPLES
13 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

IV) Comentarios: Efetivado o acesso ao Cadastro de Softwares, computa-se como Entrada Externa o conjunto de graus de influência passiveis de alteração.

V)Fonte da Identificação: Processo 1.1.4

f) **NOME DA ENTRADA:** Converter funções

I) Propósito: Possibilitar ao usuário o informe de uma nova natureza do trabalho.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
NOVA NATUREZA	02
DATA INICIO	08

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Funções)

IV)Fonte da Identificação: Processo 1.1.2

g) **NOME DA ENTRADA:** Obter definições metodológicas

I) Propósito: Permitir ao usuário fazer a solicitação para acesso ao Cadastro de Definições Conceituais da Metodologia.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
CODIGO DA DEFINIÇÃO	03

III) Complexidade: SIMPLES
2 Campos
1 Arquivo referenciado
(Cadastro de Definições Conceituais da Metodologia)

IV) Comentarios: O atributo codigo somente será solicitado quando o usuário estiver em uma das funções abaixo:

1. Criação e/ou Manutenção das Características Gerais
2. Criação e/ou Manutenção dos Fatores de Produtividade
3. Efetuar Registro de Identificação das Funções no Cadastro.

V)Fonte da Identificação: Processo 1.3.1

h) **NOME DA ENTRADA:** Solicitar atualização das definições conceituais da metodologia

I) Propósito: Possibilitar ao usuário efetuar modificações nas Definições Conceituais da Metodologia.

II) Atributos:

NOME	TAM.
CODIGO DA DEFINIÇÃO	03

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Funções)

IV) Comentarios: O atributo codigo somente será solicitado quando o usuário estiver em uma das funções abaixo:

1. Criação e/ou Manutenção das Características Gerais
2. Criação e/ou Manutenção dos Fatores de Produtividade
3. Efetuar Registro de Identificação das Funções no Cadastro.

V)Fonte da Identificação: Processo 1.3.2

i) **NOME DA ENTRADA:** Solicitar cálculo do tamanho

I) Propósito: Possibilitar ao usuário a solicitação do cálculo dos pontos por função (existentes, de desenvolvimento ou de manutenção) de um determinado software.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22

III) Complexidade: SIMPLES
1 Campo
1 Arquivo referenciado
(Cadastro de Softwares)

IV)Fonte da Identificação: Processo 1.2

j) **NOME DA ENTRADA:** Registrar esforco diario do trabalho

I) Propósito: Possibilitar ao usuário informar os dados referentes ao Registro Diario do Esforco do Trabalho.

II) Atributos:

NOME	TAM.
CODIGO DA INSTALAÇÃO	03
MATRICULA DO TÉCNICO	08
DATA EXECUÇÃO ATIVIDADE	08
CODIGO DO SOFTWARE	22
CODIGO DA FASE	03
CODIGO DA TAREFA	02
CODIGO DA ATIVIDADE	04
HORA-INICIO	05
HORA-FIM	05

III) Complexidade: COMPLEXA
9 Campos
4 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Instalações, Cadastro de Tecnicos, Cadastro do Registro Diario do Esforco do Trabalho)

IV)Fonte da Identificação: Processo 2.1

k) **NOME DA ENTRADA:** Alterar registro diario do esforco do trabalho

I) Propósito: Possibilitar ao usuário realizar alterações nos registros fornecidos.

II) Atributos:

NOME	TAM.
CODIGO DA INSTALAÇÃO	03
MATRICULA DO TÉCNICO	08
DATA DO REGISTRO	08
CODIGO DO SOFTWARE	22
CODIGO DA FASE	03
CODIGO DA TAREFA	02
CODIGO DA ATIVIDADE	04
HORA-INICIO	05
HORA-FIM	05

III) Complexidade: COMPLEXA
9 Campos
4 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Instalações, Cadastro de Tecnicos, Cadastro do Registro Diario do Esforco do Trabalho)

IV) Comentarios: Acessado o registro solicitado no Cadastro, computa-se como Entrada Externa o conjunto de dados passiveis de alteração.

V)Fonte da Identificação: Processo 2.1

l) **NOME DA ENTRADA:** Excluir registro diario do esforco do trabalho

I) Propósito: Possibilitar ao usuário a exclusão do Registro Diario do Esforco do Trabalho.

II) Atributos:

NOME	TAM.
MATRICULA DO TÉCNICO	08
DATA DO REGISTRO	08
HORA-INICIO	05

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro do Registro Diario do Esforco do Trabalho)

IV)Fonte da Identificação: Processo 2.1

m) **NOME DA ENTRADA:** Informar encerramento

I) Propósito: Possibilitar ao usuário a entrada de informações referentes ao encerramento de um software.

II) Atributos:

NOME	TAM.
MATRICULA DO TÉCNICO	08
CODIGO DO SOFTWARE	22
DATA DE ENCERRAMENTO	08
TIPO DE ENCERRAMENTO	01

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

IV)Fonte da Identificação: Processo 2.2

n) **NOME DA ENTRADA:** Solicitar produtividade

I) Propósito: Possibilitar ao usuário requisitar relatórios de produtividade (ANEXO-6 e ANEXO-7) de um determinado software.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22

III) Complexidade: SIMPLES
1 Campo
1 Arquivo referenciado(CADASTRO DE SOFTWARES)

IV)Fonte da Identificação: Processo 2.3

o) **NOME DA ENTRADA:** Cadastrar técnicos

I) Propósito: Permitir ao usuário o cadastramento de técnicos.

II) Atributos:

NOME	TAM.
MATRICULA DO TÉCNICO	08
NOME DO TÉCNICO	35
FUNÇÃO DO TÉCNICO	15

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Técnicos)

IV)Fonte da Identificação: Processo 3.1

p) **NOME DA ENTRADA:** Alterar dados do técnico

I) Propósito: Permitir ao usuário realizar modificações dos dados fornecidos dos técnicos

II) Atributos:

NOME	TAM.
MATRICULA DO TÉCNICO	08
NOME DO TÉCNICO	35
FUNÇÃO DO TÉCNICO	15

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Técnicos)

IV) Comentarios: Inicialmente, serão solicitados os campos-chave (matricula do técnico) para efetivar o acesso ao Cadastro de Técnicos, e então os demais campos serão editados.

V)Fonte da Identificação: Processo 3.1

q) **NOME DA ENTRADA:** Excluir técnico

I) Propósito: Permitir ao usuário efetuar a exclusão lógica do técnico do Cadastro de Técnicos.

II) Atributos:

NOME	TAM.
MATRICULA DO TÉCNICO	08

III) Complexidade: SIMPLES
1 Campo

1 Arquivo referenciado
(Cadastro de Técnicos)

IV)Fonte da Identificação: Processo 3.1

r) **NOME DA ENTRADA:** Cadastrar instalações

I) Propósito: Possibilitar ao usuário o cadastramento das diversas instalações onde serão implantados os sistemas.

II) Atributos:

NOME	TAM.
CODIGO DA INSTALAÇÃO	03
IDENTIF. DA INSTALAÇÃO	20
ENDERECO DA INSTALAÇÃO	
Rua	15
No/Complemento	10
Bairro	15
Cidade	25
Estado	02
Pais	20
Cep	05

III) Complexidade: SIMPLES
9 Campos
1 Arquivo referenciado
(Cadastro de Instalações)

IV)Fonte da Identificação: Processo 3.2

s) **NOME DA ENTRADA:** Alterar dados da instalação

I) Propósito: Possibilitar ao usuário a alteração dos dados referentes as instalações onde serão implantados os sistemas.

II) Atributos:

NOME	TAM.
CODIGO DA INSTALAÇÃO	03
IDENTIF. DA INSTALAÇÃO	20
ENDERECO DA INSTALAÇÃO	
Rua	15
No/Complemento	10
Bairro	15
Cidade	25
Estado	02
Pais	20
Cep	05

III) Complexidade: SIMPLES
9 Campos
1 Arquivo referenciado

(Cadastro de Instalações)

IV) Comentarios: Efetivado o acesso ao Cadastro, conta-se como Entrada Externa o conjunto de dados a serem alterados.

V)Fonte da Identificação: Processo 3.2

t) **NOME DA ENTRADA:** Excluir instalações

I) Propósito: Possibilitar ao usuário a exclusão lógica de instalações cadastradas.

II) Atributos:

NOME	TAM.
CODIGO DA INSTALAÇÃO	03

III) Complexidade: SIMPLES
1 Campo
1 Arquivo referenciado
(Cadastro de Instalações)

IV)Fonte da Identificação: Processo 3.2

u) **NOME DA ENTRADA:** Cadastrar software

I) Propósito: Permitir ao usuário a inclusão de novos softwares a serem gerenciados.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	30
LINGUAGEM USADA	10
CLASSE	02
QTDE LOCAIS IMPLANTADOS	03
DATA DE INICIO	08
MATRICULA DO TÉCNICO	08
CODIGO DA INSTALAÇÃO	03

III) Complexidade: COMPLEXA
8 Campos
3 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Técnicos, Cadastro de Instalações)

IV)Fonte da Identificação: Processo 3.3

v) **NOME DA ENTRADA:** Alterar dados do software

I) Propósito: Possibilitar ao usuário a alteração dos dados fornecidos para o software.

II) Atributos:

NOME	TAM.
CODIGO DO SOFTWARE	22
LINGUAGEM USADA	10
CLASSE	02
QTDE LOCAIS IMPLANTADOS	03
DATA DE INICIO	08
MATRICULA DO TÉCNICO	08
CODIGO DA INSTALAÇÃO	03

III) Complexidade: COMPLEXA
7 Campos
3 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Técnicos, Cadastro de Instalações)

IV) Comentarios: Acessado o registro solicitado, computa-se como Entrada Externa o conjunto dos dados passiveis de alteração.

V)Fonte da Identificação: Processo 3.3

x) **NOME DA ENTRADA:** Solicitar analise individualizada da produtividade.

I) Propósito: Possibilitar ao usuário requisitar o relatório de Analise Individualizada da Produtividade (ANEXO-8) para um determinado ano-base.

II) Atributos:

NOME	TAM.
NIVEL	01
NATUREZA DO TRABALHO	02
DATA-BASE	08

III) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Funções)

IV)Fonte da Identificação: Processo 2.3

3 - IDENTIFICAÇÃO E CLASSIFICAÇÃO DAS SAIDAS

Foram revisadas as seguintes saidas do sistema, identificadas na Definição de Requisitos, com seus respectivos propósitos, frequência, volume, complexidade e atributos:

a) **NOME DA SAIDA:** Resumo do Esforco do Trabalho (Anexo-6)

I) Propósito: Apresentar um resumo do esforco do trabalho das atividades, em cada fase do ciclo de vida do software.

II) Frequência: Ao termino de cada desenvolvimento e, eventualmente, no decorrer do mesmo, assim como ao final de cada ano de manutenção de um software.

III) Volume: Um para cada gerência, um para cada chefia e um para cada analista responsavel.

IV) Complexidade: MÉDIA
19 Campos
2 Arquivos referenciados
(Cadastro de Softwares, Cadastro do Registro Diario do Esforco do Trabalho)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	35
DATA DE INICIO	08
DATA DE CONCLUSÃO	08
ANO-BASE	08
NOME DA ATIVIDADE	60
CODIGO DA INSTALAÇÃO	03
NIVEL	01
TIPO	01
TEMPO DAS ATIVIDADES POR FASE (N x 10)	04
HORAS LIQUIDAS	06
HORAS BRUTAS	06
MESES BRUTOS	02
TOTAL HORAS LIQUIDAS	06
TOTAL HORAS BRUTAS	06
TOTAL MESES BRUTOS	02
TOTAL HORAS LIQUIDAS GLOBAIS	06
TOTAL HORAS BRUTAS GLOBAIS	06
TOTAL MESES BRUTOS	02

IV) Comentarios: Este relatório será ordenado pelo campo "NOME DA ATIVIDADE".

V)Fonte da Identificação: Processo 2.3.1/anexo-6

b) **NOME DA SAIDA:** Relatório de Produtividade para Desenvolvimento - Resumo Final (Anexo 7)

I) Propósito: Apresentar a produtividade e custo para desenvolvimento de um software especifico.

II) Frequência: Apos o termino do desenvolvimento de um software e sempre que solicitado pela chefia ou gerência.

III) Volume: Em média, 1 relatório por software.

IV) Complexidade: MÉDIA
- 16 campos
- 2 arquivos referenciados (Cadastro de Softwares, Cadastro de Técnicos)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	35
MATRICULA DO TÉCNICO	08
NOME DO TÉCNICO	35
DATA DE INICIO	08
DATA DE CONCLUSÃO	08
ANO-BASE	08
PRODUTO DO TRABALHO	05
ESFORCO DO TRABALHO	
HORAS LIQUIDAS	06
HORAS BRUTAS	06
MESES BRUTOS	02
QUANTIDADE DE LOCAIS IMPLANTADOS	02
RESULTADO DO DESENVOLVIMENTO	
PRODUTIVIDADE	06
CUSTO	06
NIVEL	01
TIPO	01

VI) Comentarios: Este relatório podera conter uma opção para impressão de um conjunto de softwares.

VII)Fonte da Identificação: Processo 2.3.1/Anexo-7

c)**NOME DA SAIDA:** Relatório de Produtividade para Manutenção/Suporte-Resumo Final (Anexo 7)

I) Propósito: Apresentar a produtividade e custo para manutenção/suporte de um software especifico.

II) Frequência: Apos o termino do desenvolvimento de um software e sempre que solicitado pela chefia ou gerência.

III) Volume: Em média, 1 relatório por software.

IV) Complexidade: MÉDIA
- 16 campos
- 2 arquivos referenciados (Cadastro de Softwares, Cadastro de Técnicos)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	35
MATRICULA DO TÉCNICO	08
NOME DO TÉCNICO	35
DATA DE INICIO	08
DATA DE CONCLUSÃO	08
ANO-BASE	08
PRODUTO DO TRABALHO	05
ESFORCO DO TRABALHO	
HORAS LIQUIDAS	06
HORAS BRUTAS	06
MESES BRUTOS	02
QUANTIDADE DE LOCAIS IMPLANTADOS	02
RESULTADO DO DESENVOLVIMENTO	
PRODUTIVIDADE	06
CUSTO	06
NIVEL	01
TIPO	01

VI) Comentarios: Este relatório podera conter uma opção para impressão de um conjunto de softwares.

VII)Fonte da Identificação: Processo 2.3.1/Anexo-7

d) **NOME DA SAIDA:** Relatório de Analise Individualizada da Produtividade (Anexo - 8)

I) Propósito: Apresentar a produtividade e custo para desenvolvimento e manutenção de softwares concluidos durante um ano.

II) Frequência: Ao final de cada ano-base ou a qualquer momento, através da solicitação feita pelo usuário.

III) Volume: Em média, 4 relatórios. Um para a gerência e um para cada chefia.

IV) Complexidade: COMPLEXA:
- 23 Campos
- 3 Arquivos referenciados

(Cadastro de Softwares, Cadastro de Técnicos, Cadastro de Dados Historicos)

V) Atributos:

NOME	TAMANHO
DATA-BASE	08
CODIGO DO SOFTWARE	22
NATUREZA DO TRABALHO	02
NOME DO SOFTWARE	35
NOME DO TÉCNICO	35
PRODUTIVIDADE DO SOFTWARE	06
CUSTO DO SOFTWARE	06
NIVEL	01

VI) Comentarios: Este relatório será ordenado pelo campo "NOME DO TÉCNICO".

VII)Fonte da Identificação: Processo 2.3.1/anexo-8

e) NOME DA SAIDA: Relatório de Analise Evolutiva Global da Produtividade (Anexo- 9)

I) Propósito: Apresentar a evolução da produtividade da empresa como um todo, através da produtividade e custo para desenvolvimento e manutenção de todos os softwares concluidos nos diversos anos-base.

II) Frequência: Anual, desde o primeiro ano de registro ate o ultimo ano-base.

III) Volume: Em média, 4 relatórios. Um para a gerência e um para cada chefia por instalação.

IV) Complexidade: MÉDIA:
8 Campos
2 Arquivos referenciados
(Cadastro de Dados Historicos, Cadastro de Instalações)

V) Atributos:

NOME	TAMANHO
DATA-BASE	08
IDENTIFICAÇÃO DA INSTALAÇÃO	20
DESENVOLVIMENTO	
PRODUTIVIDADE	06
CUSTO	06
EXPANSÃO	
PRODUTIVIDADE	06
CUSTO	06
MANUTENÇÃO	
PRODUTIVIDADE	06
CUSTO	06

VI) Comentarios: Este relatório será ordenado pelo campo "ANO-

BASE".

VII)Fonte da Identificação: Processo 2.3.1/anexo-9

f) **NOME DA SAIDA:** Relatório dos Fatores de Produtividade de um Sistema (Anexo - 10)

I) Propósito: Mostrar os diversos fatores de Produtividade, seus respectivos graus de influência e, consequentemente, o fator de produtividade global de um software.

II) Frequência: Um para cada sistema.

III) Volume: Um para a gerência e um para cada chefia.

IV) Complexidade: SIMPLES
15 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	30
GRAUS DE INFLUÊNCIA DOS FAT. DE PRODUTIVIDADE	04
FATOR DE PRODUTIVIDADE GLOBAL	02

VI) Comentarios: Este relatório podera conter uma opção para impressão de um conjunto de softwares.

VII)Fonte da Identificação: Processo 2.3.1/anexo-10

g) **NOME DA SAIDA:** Relatório das Funções Identificadas

I) Propósito: Mostrar as funções identificadas com seus respectivos atributos.

II) Frequência: Durante toda a fase de desenvolvimento e manutenção, dependendo das necessidades do analista responsavel.

III) Volume: Um para cada analista responsavel.

IV) Complexidade: SIMPLES
12 Campos
1 Arquivo referenciado
(Cadastro de Funções)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	35

CODIGO DO TIPO	02
COMPLEXIDADE ASSOCIADA	
Antes	01
Apos	01
FORMA DE INCLUSÃO	01
SITUAÇÃO ATUAL	01
TOTAL POR TIPO DE FUNÇÃO (5 x)	03

VI) Comentarios: Este relatório será ordenado por pelos campos "TIPO DE FUNÇÃO e por CODIGO DA TRANSAÇÃO".

VII)Fonte da Identificação: Processo 1.1

h) **NOME DA SAIDA:** Relatório das Características Gerais do Sistema

I) Propósito: Mostrar as diversas características gerais, seus respectivos graus de influência e, consequentemente, o fator de ajuste para um determinado software.

II) Frequência: Durante toda a fase de desenvolvimento e manutenção, dependendo das necessidades do analista responsavel.

III) Volume: Um para cada analista responsavel por um software.

IV) Complexidade: SIMPLES
18 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
NOME DO SOFTWARE	35
GRAUS DE INFLUÊNCIA	01
SOMA DOS GRAUS DE INFLUÊNCIA	02
FATOR DE AJUSTE EXISTENTE	04

VI) Comentarios: Este relatório podera conter uma opção para impressão de um conjunto de softwares.

VII)Fonte da Identificação: Processo 1.1

i) **NOME DA SAIDA:** Calculo dos pontos por função

I) Propósito: Exibir os principais valores referentes ao cálculo de pontos por função de determinado software.

II) Complexidade: MÉDIA
12 Campos
2 Arquivos referenciados
(Cadastro de Softwares, Cadastro de Funções)

III) Atributos:

NOME	TAMANHO
PONTOS POR FUNÇÃO BRUTOS INCLUIDOS	05
PONTOS POR FUNÇÃO BRUTOS ALTERADOS ANTES	05
PONTOS POR FUNÇÃO BRUTOS ALTERADOS DEPOIS	05
PONTOS POR FUNÇÃO BRUTOS EXCLUIDOS	05
PONTOS POR FUNÇÃO EXISTENTES	05
PONTOS POR FUNÇÃO DE DESENVOLVIMENTO	05
PONTOS POR FUNÇÃO DE SUPORTE	05
PONTOS POR FUNÇÃO ANTERIOR	05
SOMA DOS GRAUS DE INFLUÊNCIA ANTES	02
SOMA DOS GRAUS DE INFLUÊNCIA DEPOIS	02
FATOR DE AJUSTE ANTES	04
FATOR DE AJUSTE DEPOIS	04

IV)Fonte de Identificação: Processo 1.2

j) **NOME DA SAIDA:** Relatório dos Dados Historicos

I) Propósito: Apresentar os Dados Historicos (Pontos por Função Brutos, Fatores de Produtividade, Características Gerais, Esforco Total, Tempo Total,...) de todos os softwares concluidos na instalação.

II) Frequência: Um por ano-base.

III) Volume: Um para a gerência.

IV) Complexidade: MÉDIA
22 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
CODIGO ANTERIOR	22
NOME DO SOFTWARE	35
DATA DE INICIO	08
DATA DE TERMINO	08
LINGUAGEM USADA	10
CLASSE DO SOFTWARE	02
QUANTIDADE DE LOCAIS IMPLANTADOS	03
TIPO DE ENCERRAMENTO	01
MATRICULA DO TÉCNICO	08
PONTOS POR FUNÇÃO	05
CODIGO DA INSTALAÇÃO	03
TEMPO DAS ATIVIDADES POR FASE (N x 10)	04
HORAS LIQUIDAS	06
HORAS BRUTAS	06
MESES BRUTOS	02

TOTAL HORAS LIQUIDAS	06
TOTAL HORAS BRUTAS	06
TOTAL MESES BRUTOS	02
TOTAL HORAS LIQUIDAS GLOBAIS	06
TOTAL HORAS BRUTAS GLOBAIS	06
TOTAL MESES BRUTOS	02

VI) Comentarios: Este relatório será ordenado pelo campo "CODIGO DO SOFTWARE".

VII)Fonte da Identificação: Processo 2.2

k) **NOME DA SAIDA:** Relatório de softwares

I) Propósito: Apresentar ao usuário as informações inerentes a todos os softwares da instalação.

II) Frequência: Ao inicio de cada desenvolvimento e manutenção de um software e, eventualmente, a pedido da chefia ou do analista responsavel.

III) Volume: Um para cada chefia e um para cada analista responsavel pelo software.

IV) Complexidade: MÉDIA
12 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

V) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
CODIGO ANTERIOR	22
NOME DO SOFTWARE	35
DATA DE INICIO	08
DATA DE TERMINO	08
LINGUAGEM USADA	10
CLASSE DO SOFTWARE	02
QUANTIDADE DE LOCAIS IMPLANTADOS	03
TIPO DE ENCERAMENTO	01
MATRICULA DO TÉCNICO	08
PONTOS POR FUNÇÃO	05
CODIGO DA INSTALAÇÃO	03

IV) Comentarios: Este relatório será ordenado pelo campo "CODIGO DO SOFTWARE".

V)Fonte da Identificação: Processo 3.4

l) **NOME DA SAIDA:** Relatório dos Técnicos Cadastrados

I) Propósito: Apresentar uma listagem dos dados de todos os possiveis técnicos da instalação.

II) Frequência: Sempre que um técnico sair da empresa ou um novo técnico for admitido e, eventualmente, a pedido da gerência.

III) Volume: Um para a gerência e um para cada chefia.

IV) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado
(Cadastro de Técnicos)

V) Atributos:

NOME	TAMANHO
MATRICULA DO TÉCNICO	08
NOME DO TÉCNICO	35
FUNÇÃO DO TÉCNICO	15

VI) Comentarios: Este relatório será ordenado pelo campo "NOME DO TÉCNICO".

VII)Fonte da Identificação: Processo 3.4

m) **NOME DA SAIDA:** Relatório das Instalações

I) Propósito: Apresentar os dados de todas as instalações gerenciadas.

II) Frequência: Sempre que uma instalação for desativada ou quando for ativada uma nova instalação e, eventualmente, a pedido da gerência.

III) Volume: Um para a gerência.

IV) Complexidade: SIMPLES
9 Campos
1 Arquivo referenciado
(Cadastro de Instalações)

V) Atributos:

NOME	TAMANHO
CODIGO DA INSTALAÇÃO	03
IDENTIFICAÇÃO DA INSTALAÇÃO	20
ENDERECO DA INSTALAÇÃO	
Rua	15
N./Complemento	10
Bairro	15
Cidade	25
Estado	02

Pais	20
Cep	05

VI) Comentarios: Este relatório será ordenado pelo campo "CODIGO DA INSTALAÇÃO".

VII)Fonte da Identificação: Processo 3.4

n) **NOME DA SAIDA:** Relação dos Técnicos Cadastrados (tela-paginada)

I) Propósito: Exibir os dados referentes aos diversos técnicos cadastrados.

II) Complexidade: SIMPLES
3 Campos
1 Arquivo referenciado (Cadastro de Técnicos)

III) Atributos:

NOME	TAMANHO
MATRICULA DO TÉCNICO	08
NOME DO TÉCNICO	35
FUNÇÃO DO TÉCNICO	15

IV) Comentarios: Este relatório será ordenado pelo campo "MATRICULA DO TÉCNICO".

V)Fonte da Identificação: Processo 3.4

o) **NOME DA SAIDA:** Relação das Instalações (tela-paginada)

I) Propósito: Exibir os dados referentes as diversas instalações cadastradas.

II) Complexidade: SIMPLES
9 Campos
1 Arquivo referenciado (Cadastro de Instalações)

III) Atributos:

NOME	TAMANHO
CODIGO DA INSTALAÇÃO	03
IDENTIFICAÇÃO DA INSTALAÇÃO	20
ENDERECO DA INSTALAÇÃO	
Rua	15
No/Complemento	10
Bairro	15
Cidade	25
Estado	02
Pais	20

Cep	05

IV) Comentarios: Este relatório será ordenado pelo campo "CODIGO DA INSTALAÇÃO".

V)Fonte da Identificação: Processo 3.4

p) **NOME DA SAIDA:** Relação dos softwares(tela-paginada).

I) Propósito: Exibir os dados referentes aos diversos softwares cadastrados

II) Complexidade: SIMPLES
12 Campos
1 Arquivo referenciado
(Cadastro de Softwares)

III) Atributos:

NOME	TAMANHO
CODIGO DO SOFTWARE	22
CODIGO ANTERIOR	22
NOME DO SOFTWARE	35
DATA DE INICIO	08
DATA DE TERMINO	08
LINGUAGEM USADA	10
CLASSE DO SOFTWARE	02
QUANTIDADE DE LOCAIS IMPLANTADOS	03
TIPO DE ENCERAMENTO	01
MATRICULA DO TÉCNICO	08
PONTOS POR FUNÇÃO	05
CODIGO DA INSTALAÇÃO	03

IV) Comentarios: Este relatório será ordenado pelo campo "CODIGO DO SOFTWARE".

V)Fonte da Identificação: Processo 3.4

4- IDENTIFICAÇÃO E CLASSIFICAÇÃO DAS CONSULTAS EXTERNAS

Foram revisadas as seguintes consultas externas, identificadas na Definição de Requisitos, com seus respectivos propósitos e complexidade :

a) **NOME DA CONSULTA:** Definição das Funções (Tela de Ajuda)

I) Propósito: Possibilitar ao usuário uma verificação ON-LINE a Definição das Funções.

II) Complexidade: MÉDIA.

III) Comentarios: Esta tela podera ser solicitada somente quando o usuário estiver na operação de Identificação de Funções de um determinado software.

IV)Fonte da Identificação: Processo 1.3

b) **NOME DA CONSULTA:** Definição das Características Gerais (Tela de Ajuda)

I) Propósito: Possibilitar ao usuário uma verificação ON-LINE a Definição das Características Gerais, segundo a Metodologia Pontos por Função.

II) Complexidade: MÉDIA.

III) Comentarios: Esta tela podera ser solicitada somente quando o usuário estiver na operação de Definição dos Graus de Influência das Características Gerais de um determinado software.

IV)Fonte da Identificação: Processo 1.3

c) **NOME DA CONSULTA:** Definição dos Fatores de Produtividade (Tela de Ajuda)

I) Propósito: Possibilitar ao usuário uma verificação ON-LINE a Definição dos Fatores de Produtividade, segundo a Metodologia em questão.

II) Complexidade: MÉDIA.

III) Comentarios: Esta tela podera ser solicitada somente quando o usuário estiver na operação de Definição dos Graus de Influência dos Fatores de Produtividade de um determinado software.

IV)Fonte da Identificação: Processo 1.3

d) **NOME DA CONSULTA:** Visão Geral da Metodologia Pontos por Função (Tela de Ajuda)

I) Propósito: Possibilitar ao usuário uma verificação ON-LINE da Metodologia Pontos por Função.

II) Complexidade: MÉDIA.

III) Comentarios: Esta tela podera ser solicitada em qualquer parte do sistema, exceto quando o usuário estiver em uma das tres opções abaixo:
Identificação de Funções
Identificação dos Graus de Influência das Características Gerais
Identificação dos Graus de Influência dos Fatores de Produtividade.

IV)Fonte da Identificação: Processo 1.3

e) **NOME DA CONSULTA:** Consulta ao Cadastro de Funções

I) Propósito: Possibilitar ao usuário consultar os dados a respeito de uma determinada função.

II) Complexidade: MÉDIA.

III) Comentarios: Inicialmente, serão solicitados os campos-chave (codigo do software, codigo do Tipo de Função, codigo da Transação) para efetivar o acesso ao Cadastro de Funções, então todo o conteudo do registro e editado. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2-b.

IV)Fonte da Identificação: Processo 1.1

f) **NOME DA CONSULTA:** Consulta as Características Gerais de um software

I) Propósito: Possibilitar ao usuário consultar os valores atribuidos aos graus de influência das características gerais de um software qualquer.

II) Complexidade: SIMPLES.

III) Comentarios: Inicialmente, será solicitado o campo-chave (codigo do software) para efetivar o acesso ao Cadastro de Softwares, então todo o conteudo do registro e editado. Quanto

aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2.d.

IV)Fonte da Identificação: Processo 1.1

g) **NOME DA CONSULTA:** Consulta aos Fatores de Produtividade de um software

I) Propósito: Possibilitar ao usuário consultar os graus dos fatores de produtividade de um software.

II) Complexidade: SIMPLES.

III) Comentarios: Inicialmente, será solicitado o campo-chave (codigo do Software), para efetivar o acesso ao Cadastro de Softwares, então todo o conteudo do registro e editado. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2.n.

IV)Fonte da Identificação: Processo 1.1

h) **NOME DA CONSULTA:** Consulta ao Registro Diario do Esforco do Trabalho

I) Propósito: Possibilitar ao usuário consultar os dados do Registro Diario do Esforco do Trabalho.

II) Complexidade: COMPLEXA.

III) Comentarios: Inicialmente, serão solicitados os campos-chave (matricula do técnico, data do registro, hora-inicio) para fins de identificação do registro, então todo o seu conteudo e editado. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2.j.

IV)Fonte da Identificação: Processo 2.1

i) **NOME DA CONSULTA:** Consulta aos Informes de Encerramento

I) Propósito: Possibilitar ao usuário consultar os dados dos informes de encerramento de um software.

II) Complexidade: SIMPLES.

III) Comentarios: Inicialmente, será solicitado o campo-chave (codigo do software) para identificação do registro, quando então todo o seu conteudo e editado. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2.x.

IV)Fonte da Identificação: Processo 2.2

j) **NOME DA CONSULTA:** Consulta ao Cadastro de Softwares

I) Propósito: Possibilitar ao usuário consultar os dados referentes ao software solicitado.

II) Complexidade: COMPLEXA.

III) Comentarios: Inicialmente, será solicitado o campo-chave (codigo do software) para efetivar o acesso ao Cadastro de Softwares, então todo o conteudo do registro e exibido. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2.v.

IV)Fonte da Identificação: Processo 3.3

k) **NOME DA CONSULTA:** Consulta ao Cadastro de Técnicos

I) Propósito: Possibilitar ao usuário consultar os dados referentes a um determinado técnico.

II) Complexidade: SIMPLES.

III) Comentarios: Inicialmente, será solicitado o campo-chave (matricula do Técnico) para efetivar o acesso ao Cadastro de Técnicos, então todo o conteudo do registro e editado. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2.p.

IV)Fonte da Identificação: Processo 3.1

l) **NOME DA CONSULTA:** Consulta ao Cadastro de Instalações

I) Propósito: Possibilitar ao usuário consultar os dados referentes a instalação solicitada.

II) Complexidade: SIMPLES.

III) Comentarios: Inicialmente, será solicitado o campo-chave (codigo da Instalação) para efetivar o acesso ao Cadastro de Instalações, então todo o conteudo do registro e editado. Quanto aos valores para Frequência, Volume e Atributos, assemelham-se aqueles citados no item 2-s.

IV)Fonte da Identificação: Processo 3.2

5- IDENTIFICAÇÃO E CLASSIFICAÇÃO DAS INTERFACES EXTERNAS

Foram revisadas as seguintes interfaces externas do sistema, identificadas na Definição de Requisitos, com seus respectivos propósitos, complexidade, volume e atributos :

a) **NOME DA INTERFACE:** Cadastro de Softwares

I) Propósito: Prover informações inerentes a um software para o Calculo da Produtividade.

II) Volume: Em média, dois registros por sistema.

III) Complexidade: SIMPLES
31 Campos
01 Registro Lógico

V) Atributos:
NOME
CODIGO DO SOFTWARE
NOME DO SOFTWARE
DATA DE INICIO
DATA DE TERMINO
LINGUAGEM USADA
CLASSE DO SOFTWARE
QTDE LOCAIS IMPLANTADOS
TIPO DE ENCERRAMENTO
MATRICULA DO TÉCNICO
CODIGO DA INSTALAÇÃO
QTDE EE SIMPLES
QTDE EE MÉDIA
QTDE EE COMPLEXA
QTDE SE SIMPLES
QTDE SE MÉDIA
QTDE SE COMPLEXA
QTDE CE SIMPLES
QTDE CE MÉDIA
QTDE CE COMPLEXA
QTDE IE SIMPLES
QTDE IE MÉDIA
QTDE IE COMPLEXA
QTDE AI SIMPLES
QTDE AI MÉDIA
QTDE AI COMPLEXA
CARACT. GERAL ANTES
CARACT. GERAL DEPOIS
FATOR DE PRODUTIVIDADE
ESFORCO TOTAL
HORAS LIQUIDAS
HORAS BRUTAS

MESES BRUTOS

V) Comentarios: Este cadastro não será descrito como Interface para o Sub-Sistema de Estimativas de Desenvolvimento (de acordo com o especificado na Definição de Requisitos).

Tabela de Interfaces:

DE	ARQUIVO INTERFACEADO	PARA
Sub-sistema de Calculo do Tamanho	Cadastro de Softwares	Sub-sistema de Calculo da Produtividade

VI)Fonte da Identificação: Processo 1.2

5- IDENTIFICAÇÃO E CLASSIFICAÇÃO DOS ARQUIVOS INTERNOS

Foram revisadas as seguintes necessidades de armazenamento de dados:

a) **NOME DO ARQUIVO:** CADASTRO DAS FUNÇÕES DO SOFTWARE

I) Propósito: Armazenar os dados referentes as diversas funções identificadas por software.

II) Volume: E o produto do no médio de funções por sistema pela quantidade de sistemas existentes.

III) Atributos:

NOME
CODIGO DO SOFTWARE
CODIGO DO TIPO DE FUNÇÃO
CODIGO DA TRANSAÇÃO
NOME DA TRANSAÇÃO
DETALHES DE COMP. ANTES
DETALHES DE COMP. DEPOIS
FORMA DE INCLUSÃO
SITUAÇÃO ATUAL

IV) Complexidade: SIMPLES
8 Campos
1 Registro Lógico

V)Fonte da Identificação: Deposito de Dados D2

b)**NOME DO ARQUIVO:**CADASTRO REGISTRO DIARIO ESFORCO DO TRABALHO

I) Propósito: Prover informações do esforco diario do trabalho registrado para todas as atividades executadas pelas equipes de desenvolvimento e manutenção que são necessarias para o cálculo da produtividade.

II) Volume: O produto da média das atividades executadas em um dia de trabalho pelo numero de funcionarios da instalação e pelo numero de dias uteis no ano-base nos fornecera o volume de dados para um ano-base.

III) Atributos:

NOME
CODIGO DA INSTALAÇÃO
MATRICULA DO TÉCNICO
DATA DO REGISTRO
CODIGO DO SOFTWARE
CODIGO DA FASE
CODIGO DA TAREFA

CODIGO DA ATIVIDADE
HORA INICIO
    HORA
    MINUTO
 HORA FIM
    HORA
    MINUTO

IV) Complexidade: SIMPLES
                  11 Campos
                   1 Registro Lógico

V)Fonte da Identificação: Deposito de Dados D6

c)**NOME DO ARQUIVO:** CADASTRO DE SOFTWARES

I) Propósito: Armazenar informações inerentes aos diversos softwares em desenvolvimento, expansão e manutenção na instalação.

II) Volume: Para um sistema (similar a este) com tamanho médio de 237 pontos por função, temos um volume médio de 30 softwares registrados neste cadastro durante o Desenvolvimento Novo.

III) Atributos:
       NOME
CODIGO DO SOFTWARE
NOME DO SOFTWARE
DATA DE INICIO
DATA DE TERMINO
LINGUAGEM USADA
CLASSE DO SOFTWARE
QTDE LOCAIS IMPLANTADOS
TIPO DE ENCERRAMENTO
MATRICULA DO TÉCNICO
CODIGO DA INSTALAÇÃO
QTDE EE SIMPLES
QTDE EE MÉDIA
QTDE EE COMPLEXA
QTDE SE SIMPLES
QTDE SE MÉDIA
QTDE SE COMPLEXA
QTDE CE SIMPLES
QTDE CE MÉDIA
QTDE CE COMPLEXA
QTDE IE SIMPLES
QTDE IE MÉDIA
QTDE IE COMPLEXA
QTDE AI SIMPLES
QTDE AI MÉDIA
QTDE AI COMPLEXA
CARACT. GERAL ANTES
CARACT. GERAL DEPOIS

FATOR DE PRODUTIVIDADE
   HORAS BRUTAS
   HORAS LIQUIDAS
   HORAS BRUTAS
   MESES BRUTOS

IV) Complexidade: SIMPLES
                  31 Campos
                   1 Registro Lógico

V)Fonte da Identificação: Deposito de Dados D1

d)**NOME DO ARQUIVO:** CADASTRO DE TÉCNICOS

I) Propósito: Prover informações dos técnicos disponiveis no C.P.D. para identificação das atribuições de responsabilidade dos mesmos.

II) Volume: Uma ocorrência para cada funcionario das diversas instalações gerenciadas (ativos ou não).

III) Atributos:
       NOME
MATRICULA DO TÉCNICO
NOME DO TÉCNICO
FUNÇÃO DO TÉCNICO

IV) Complexidade: SIMPLES
                  Ø3 Campos
                   1 Registro Lógico

V)Fonte da Identificação: Deposito de Dados D8

e)**NOME DO ARQUIVO:** CADASTRO DE INSTALAÇÕES

I) Propósito: Armazenar informações referentes as diversas instalações que estão sendo gerenciadas.

II) Volume: Uma ocorrência para cada instalação.

III) Atributos:
       NOME
CODIGO DA INSTALAÇÃO
IDENTIF. DA INSTALAÇÃO
ENDERECO DA INSTALAÇÃO
   RUA
   No/COMPLEMENTO
   BAIRRO
   CIDADE
   ESTADO

PAIS
C.E.P.

IV) Complexidade: SIMPLES
9 Campos
1 Registro Lógico

V)Fonte da Identificação: Deposito de Dados D7

f)**NOME DO ARQUIVO:**CADASTRO DEFINIÇÕES CONCEITUAIS DA METODOLOGIA

I) Propósito: Armazenar informações das definições conceituais da Metodologia Pontos por Função (Visão Geral da Metodologia, Definição de Funções, Características Gerais e Fatores de Produtividade).

II) Volume: A estimar.

III) Atributos:
NOME
CODIGO DA DEFINIÇÃO
DESCRIÇÃO DA DEFINIÇÃO

IV) Complexidade: SIMPLES
2 Campos
1 Registro Lógico

V)Fonte da Identificação: Deposito de Dados D3

7- CÁLCULO DO TAMANHO

FUNÇÃO	COMPLEXIDADES SIMPLES	MEDIA	COMPLEXA	TOTAL
EE-ENTRADA EXTERNA	17 * 3 = 51	2 * 4 = 8	4 * 6 = 24	79
SE-SAIDA EXTERNA	8 * 4 = 32	7 * 5 = 35	1 * 7 = 7	74
AI-ARQUIVO INTERNO	6 * 7 = 42	0 * 10 = 0	0 * 15 = 0	42
IE-INTERFACE EXTERNA	1 * 5 = 5	0 * 7 = 0	0 * 10 = 0	5
CE-CONSULTA EXTERNA	5 * 3 = 15	5 * 4 = 20	2 * 6 = 12	47

Pontos por Função Brutos (P.F.B.) =247

a.2) CARACTERÍSTICAS GERAIS

CARACTERÍSTICAS	G.I	CARACTERÍSTICAS	G.I
1.Comunicação de Dados	1	8.Atualizações On-line	3
2.Funções Distribuidas	0	9.Processamento Complexo	0
3.Objetivos de Performance	2	10.Reutilizabilidade	0
4.Carga da Configuração	2	11.Facilidade de Instalação	2
5.Volume de Transações	2	12.Facilidade de Operação	2
6.Entrada de Dados On-line	5	13.Multiplicidade de Locais	3
7.Eficiência do Usuário Final	4	14.Facilidade de Mudanca	4
Sub-Total	16	Sub-Total	14

Somatório dos Graus de Influência (S.G.I.) = 30

Fator de Ajuste Existente (F.A.E.) = 0,65 + (0,01 * S.G.I.)
= 0,65 + (0,01 * 30)
= 0,95

Pontos por Função Existentes (P.F.E.) = P.F.B. * F.A.E.
= 247 * 0,95
= 234