Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Modelos de Estimativas de Custo de Software
COCOMO & COCOMO II

Waine Teixeira Júnior
Rosely Sanches

N$^{\underline{0}}$ 106

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos

ABRIL/2000

SUMÁRIO

Lista de Abreviaturas ..... **i**
Lista de Equações ..... **iii**
Lista de Figuras ..... **iv**
Lista de Quadros ..... **v**
Lista de Tabelas ..... **vi**
Capítulo 1 O Modelo COCOMO **1**
1.1 ESTIMATIVAS DE PROJETO DE SOFTWARE 1
1.2 O MODELO COCOMO 81 3
1.2.1 Equações do modelo COCOMO *4*
1.2.2 Direcionadores de custo *5*
1.2.3 Exemplo de estimativa de esforço e prazo do modelo COCOMO *7*
1.2.4 Distribuição de esforço e prazo – COCOMO Básico e Intermediário *7*
1.2.5 Exemplo de distribuição do quantitativo de pessoal por fase *9*
1.3 FUNDAMENTOS DO MODELO COCOMO 11
Capítulo 2 O modelo COCOMO II **13**
2.1 O PROJETO COCOMO II 13
2.1.1 Programação de Usuário Final *14*
2.1.2 Infra-estrutura *14*
2.1.3 Geradores de Aplicações *15*
2.1.4 Composição de Aplicações *15*
2.1.5 Integração de Sistemas *15*
2.2 RESULTADOS DO PROJETO COCOMO II 16
Capítulo 3 COCOMO II – Application Composition **17**
3.1 APLICAÇÃO DO MODELO 17
3.2 CONTAGEM DE PONTOS DE OBJETO 17
Capítulo 4 COCOMO II – Early Design **21**
4.1 APLICAÇÃO DO MODELO 21
4.2 PONTOS DE FUNÇÃO 21
4.2.1 Contagem de Pontos de Função *22*
4.3 DIRECIONADORES DE CUSTO 24
4.3.1 Direcionador de Custo RCPX – Confiabilidade e a Complexidade do Produto *25*
4.3.2 Direcionador de Custo RUSE – Reuso Requerido *26*
4.3.3 Direcionador de Custo PDIF – Dificuldade de Plataforma *26*
4.3.4 Direcionador de Custo PERS – Capacidade de Pessoal *26*
4.3.5 Direcionador de Custo PREX – Experiência do Pessoal *27*
4.3.6 Direcionador de Custo FCIL – Facilidades *27*
4.3.7 Direcionador de Custo SCED – Cronograma *28*
4.4 FATORES DE ESCALA 28
4.4.1 Fatores de Escala Precedência (PREC) e Flexibilidade de Desenvolvimento (FLEX) *29*
4.4.2 Fatores de Escala Arquitetura e Resolução de Risco (RESL) *29*
4.4.3 Fator de Escala Coesão de Equipe (TEAM) *31*
4.4.4 Fator de Escala Maturidade do Processo (PMAT) *31*
4.5 EQUAÇÕES DO MODELO *EARLY DESIGN* 34
4.6 EQUAÇÃO PARA ESTIMAR CRONOGRAMA DE DESENVOLVIMENTO 36
Capítulo 5 COCOMO II – Post-Architecture **37**
5.1 APLICAÇÃO DO MODELO 37
5.2 REGRAS DE CONTAGEM DE LINHAS DE CÓDIGO 37
5.3 DIRECIONADORES DE CUSTO 38
5.4 CATEGORIA DE FATORES PRODUTO 40
5.4.1 Confiabilidade Requerida do Software (RELY) *40*
5.4.2 Tamanho da Base de Dados (DATA) *40*
5.4.3 Complexidade do Produto (CPLX) *40*
5.4.4 Reusabilidade Requerida (RUSE) *41*

5.4.5 Documentação Adequada às Necessidades do Projeto (DOCU) *41*
5.5 A CATEGORIA DE FATORES PLATAFORMA 41
5.5.1 Restrição de Tempo de Execução (TIME) *41*
5.5.2 Restrição de Armazenamento Principal (STOR) *42*
5.5.3 Volatilidade da Plataforma (PVOL) *42*
5.6 A CATEGORIA DE FATORES PESSOAL 42
5.6.1 Capacidade do Analista (ACAP) *42*
5.6.2 Capacidade dos Programadores (PCAP) *43*
5.6.3 Rotatividade de Pessoal (PCON) *43*
5.6.4 Experiência com a Aplicação (AEXP) *43*
5.6.5 Experiência com a Plataforma de Desenvolvimento (PEXP) *43*
5.6.6 Experiência com Linguagens e Ferramentas de Desenvolvimento (LTEX) *43*
5.7 CATEGORIA DE FATORES PROJETO 44
5.7.1 Uso de Ferramentas de Software (TOOL) *44*
5.7.2 Desenvolvimento Distribuído ou Multisite (SITE) *44*
5.7.3 Cronograma de Desenvolvimento Requerido (SCED) *44*
5.7.4 Confiabilidade Requerida do Software (RELY) *45*
5.7.5 Tamanho da Base de Dados (DATA) *45*
5.7.6 Complexidade do Produto *45*
5.7.7 Reusabilidade Requerida (RUSE) *46*
5.7.8 Documentação Adequada às Necessidades do Projeto (DOCU) *46*
5.7.9 Restrição de Tempo de Execução (TIME) *46*
5.7.10 Restrição de Armazenamento Principal (STOR) *46*
5.7.11 Volatilidade da Plataforma (PVOL) *47*
5.7.12 Capacidade do Analista (ACAP) *47*
5.8 EQUAÇÕES DO MODELO *POST-ARCHITECTURE* 47
Capítulo 6 Ferramentas Automatizadas 49
6.1 FERRAMENTAS QUE IMPLEMENTAM COCOMO 49
6.1.1 Revic *49*
6.1.2 COSMOS *50*
6.1.3 COSTAR. *51*
6.1.4 USC COCOMO 2000.0 ..... *54*
6.1.5 Construx Estimate *54*
6.1.6 COST XPERT 2.1 ..... *56*
Capítulo 7 Conclusão 61
Capítulo 8 Bibliografia 63

Lista de Abreviaturas

3GL	Linguagem de Terceira Geração
AA	Porcentagem de esforço de reuso devido à avaliação e à assimilação
AAF	Fator de Ajuste de Adaptação
AAM	Multiplicador de Ajuste de Adaptação
ACAP	Capacidade do Analista
ACT	Trânsito de Mudança Anual
AEXP	Experiência com Aplicações
ASLOC	Linhas de Código Adaptadas
AT	Tradução Automática
BRAK	Quebra. A quantidade de mudança controlada permitida em um desenvolvimento de software antes que os requisitos sejam "congelados"
CASE	Engenharia de Software Auxiliada por Computador
CM	Porcentagem de código modificado durante o reuso
CMM	Modelo de Maturidade de Capacidade
COCOMO	Modelo de Custo Construtivo
COTS	Software Comercial de Prateleira
CPLX	Complexidade do Produto
CSTB	Conselho de Ciência da Computação e Telecomunicações
DATA	Tamanho da Base de Dados
DBMS	Sistema Gerenciador de Banco de Dados
DI	Grau de Influência
DM	Porcentagem de projeto modificado durante o reuso
DOCU	Documentação necessária aos requisitos do projeto
EDS	Sistemas de Dados Eletrônicos
ESLOC	Linhas de Código Equivalentes
FCIL	Facilidades
GFS	Software Fornecido pelo Governo
GUI	Interface Gráfica para Usuário
ICASE	Ambiente Integrado de Software Auxiliado por Computador
IM	Porcentagem de integração refeita durante o reuso
KSLOC	Milhares de linhas de código adaptadas
KESLOC	Milhares de linhas de código equivalentes
LEXP	Experiência com Linguagem de Programação
LTEX	Experiência com Linguagem e Ferramentas
MODP	Práticas Modernas de Programação
NIST	National Institute of Standards and Technology
NOP	Novos Pontos de Objeto
OS	Sistema Operacional
PCAP	Capacidade do Programador
PCON	Continuidade de Pessoal
PDIF	Dificuldade de Plataforma
PERS	Capacidade de Pessoal
PEXP	Experiência com Plataforma

PL	Linha de Produto
PM	Pessoas Mês
PREX	Experiência de Pessoal
PROD	Taxa de Produtividade
PVOL	Volatilidade de Plataforma
RCPX	Confiabilidade e Complexidade do Produto
RELY	Confiabilidade Requerida do Software
RUSE	Reusabilidade Requerida
RVOL	Volatilidade dos Requisitos
SCED	Cronograma de Desenvolvimento Requerido
SECU	Aplicação de Segurança Sigilosa
SEI	Software Engineering Institute
SITE	Operação Multi-site
SLOC	Linhas de Código Fonte
STOR	Restrição de Armazenamento Principal
SU	Porcentagem de Esforço de Reuso devido à compreensão do software
T&E	Teste e Avaliação
PF	Pontos de Função
TIME	Restrição de Tempo de Execução
TOOL	Uso de Ferramentas de Software
TURN	Tempo de Rotatividade de Computador
UNFM	Não Familiaridade do Programador
USA/ESD	U.S. Air Force Electronic Systems Division
VEXP	Experiência com Máquina Virtual
VIRT	Volatilidade de Máquina Virtual
VMVH	Volatilidade de Máquina Virtual: *Host*
VMVT	Volatilidade de Máquina Virtual: *Target*

Lista de Equações

Equação 1 Estimativa de Esforço de Desenvolvimento 4
Equação 2 Estimativa de Tempo de Desenvolvimento 5
Equação 3 Interpolação Aritmética 10
Equação 4 Pontos de Objeto e Reuso 19
Equação 5 Application Composition – Produtividade dos Desenvolvedores 19
Equação 6 Esforço e Pontos de Objeto 20
Equação 7 Esforço – Early Design & Post-Architecture 28
Equação 8 Maturidade do Processo (*PMAT*) 33
Equação 9 Esforço de Desenvolvimento – *Early Design* 34
Equação 10 Cálculo do coeficiente *b* 34
Equação 11 Porcentagem de Quebra – BRAK 34
Equação 12 Reuso de Componentes 35
Equação 13 Fatores para Equação de Reuso de Componentes (1) 35
Equação 14 Fatores para Equação de Reuso de Componentes (2) 35
Equação 15 Manutenção de Projeto 35
Equação 16 Estimativa de Prazo 36
Equação 17 Tamanho da Base de Dados 45
Equação 18 Modelo Post Architecture – Estimativa de Esforço 47

Lista de Figuras

Figura 1 Base de dados históricos para estimativas [Fernandes, 1995] 2
Figura 2 Componentes de Pontos de Função [Dekkers, 1998] 22
Figura 3 REVIC – Tela de abertura da ferramenta 49
Figura 4 COSMOS – Tela de estimativa e distribuição por fases.....50
Figura 5 COSTAR – Tela de entrada de pontos de função.....51
Figura 6 COSTAR – Seleção da Linguagem de desenvolvimento 51
Figura 7 COSTAR – Tela de ajuste dos direcionadores de custo 52
Figura 8 COSTAR – Menu principal 52
Figura 9 COSTAR – Relatório de Estimativa por Fase 53
Figura 10 COSTAR – Relatório de Estimativa por Atividade.....53
Figura 11 USC COCOMO – Menu principal do sistema.....54
Figura 12 CONSTRUX – Menu principal 55
Figura 13 CONSTRUX – Ajuste dos direcionadores de custo 56
Figura 14 Cost Xpert – Menu Principal 57
Figura 15 Cost Xpert – Estimativa de Tamanho 57
Figura 16 Cost Xpert – Direcionadores de Custo 58
Figura 17 Cost Xpert – Outros Fatores Direcionadores de Custo.....58
Figura 18 Cost Xpert – Estimativas Parciais.....59
Figura 19 Cost Xpert – Gráfico de Distribuição 59
Figura 20 Cost Xpert – Gráfico de Gantt 60

Lista de Quadros

Quadro 1 Atributos Direcionadores de Custo do COCOMO [Conte, 1986] 6
Quadro 2 Futuro das Práticas de Software [COCOMOII, 2000] 14
Quadro 3 Tipos de Função 23
Quadro 4 Direcionadores de Custo [COCOMOII, 2000] 25
Quadro 5 Escala de Classificação do Direcionador *RUSE* [COCOMOII, 2000].....26
Quadro 6 Fatores de Escala do COCOMO original [COCOMOII, 2000] 30
Quadro 7 Componentes da escala do Fator TEAM [COCOMOII, 2000].....31
Quadro 8 Áreas Chave de Processo do CMM [PAULK, 1992].....32
Quadro 9 Avaliação das Práticas do Modelo CMM (COCOMOII, 2000].....33

Lista de Tabelas

Tabela 1 Parâmetros do COCOMO – Estimativa de Esforço [Conte, 1986] 4
Tabela 2 Parâmetros do COCOMO – Estimativa de Prazo [Fernandes, 1995] 5
Tabela 3 Direcionadores de Custo do COCOMO [Conte, 1986].....6
Tabela 4 Multiplicadores do COCOMO 81 [Conte, 1986].....6
Tabela 5 Tabela Padrão de Tamanho de Software do COCOMO [Fernandes, 1995] ...7
Tabela 6 Distribuição de Esforço [Fernandes, 1995].....8
Tabela 7 Distribuição de Prazo [Fernandes, 1995] 8
Tabela 8 Distribuição do esforço para o exemplo [Fernandes, 1995].....9
Tabela 9 Distribuição do prazo para o exemplo [Fernandes, 1995].....9
Tabela 10 Distribuição de pessoal por fase para o exemplo [Fernandes, 1995] 9
Tabela 11 Classificação de Pontos de Objeto para Telas [COCOMOII, 2000].....18
Tabela 12 Pesos de Complexidade de Pontos de Objeto [COCOMOII, 2000].....18
Tabela 13 Produtividade de Desenvolvedores [COCOMOII, 2000] 19
Tabela 14 Classificação dos Tipos de Função [COCOMOII, 2000].....23
Tabela 15 Complexidade dos Tipos de Função [COCOMOII, 2000].....24
Tabela 16 Conversão de PF em Linhas de Código [COCOMOII, 2000] 24
Tabela 17 Escala de Classificação do Direcionador *RCPX* [COCOMOII, 2000] 25
Tabela 18 Escala de Classificação do Direcionador *PDIF* [COCOMOII, 2000] 26
Tabela 19 Escala de Classificação do Direcionador *PERS* [COCOMOII, 2000] 27
Tabela 20 Escala de Classificação do Direcionador *PREX* [COCOMOII, 2000].....27
Tabela 21 Escala de Classificação do Direcionador *FCIL* [COCOMOII, 2000].....28
Tabela 22 Escala de Classificação do Direcionador *SCED* [COCOMOII, 2000] 28
Tabela 23 Fatores de Escala Calibrados [COCOMOII, 2000].....29
Tabela 24 Componentes da escala do Fator RESL [COCOMOII, 2000] 30
Tabela 25 Post Achitecture – Direcionadores de Custo [COCOMOII, 2000].....38
Tabela 26 Direcionadores de Custo Calibrados [COCOMOII, 2000] 39

Capítulo 1 O MODELO COCOMO

1.1 ESTIMATIVAS DE PROJETO DE SOFTWARE

Um dos processos fundamentais do gerenciamento de projetos de software é o *planejamento*, um conjunto de atividades que visam estimar tempo e recursos necessários ao projeto, definir tarefas e preparar uma estrutura para o gerenciamento [Humphrey, 1989]. Para que um projeto de software possa ser efetivamente planejado, *estimativas* do custo, em termos de esforço, prazo e orçamento, devem ser derivadas logo no início, através de um processo bem definido – o *processo de estimativa de custo de software* [Paulk et. al., 1993].

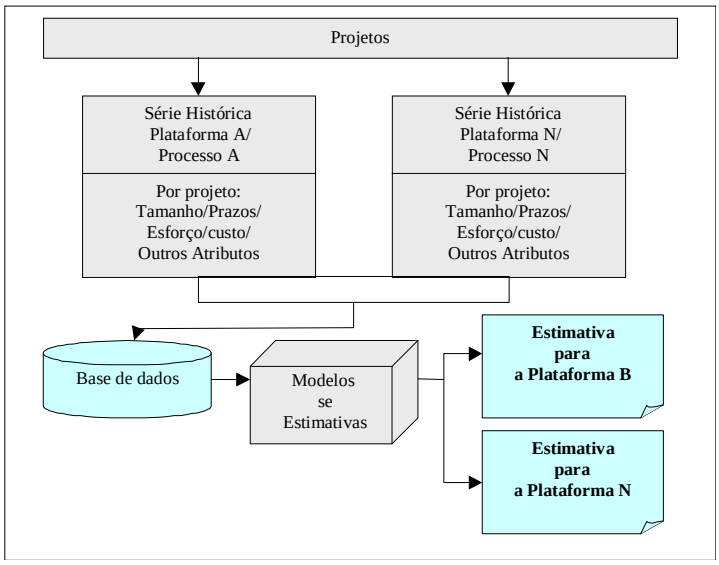
Os processos utilizados pelas organizações para estimar projetos podem ser enquadrados em duas categorias: *processos baseados em modelo* e *processos baseados em analogia*.

O *processo baseado em analogia* compara projetos novos com projetos anteriormente desenvolvidos pela organização para estimar custo. Uma das formas mais comuns de processos baseados em analogia é o uso da *experiência acumulada* com o desenvolvimento de sistemas ao longo do tempo, algo muito comum nas organizações de pequeno porte que trabalham com pequenos projetos e força de trabalho estável [Vigder, 1994].

Para aumentar a precisão das estimativas, um histórico dos projetos passados deve ser mantido em uma base de dados. A base de dados é alimentada com a coleta de dados sobre prazos, esforço e custo efetivamente realizados em projetos anteriormente desenvolvidos (Figura 1), considerando-se o tipo de plataforma de hardware/software e o tipo de processo de desenvolvimento, bem como a coleta de dados sobre outros atributos, como esforço de retrabalho [Fernandes, 1995].

Em grandes projetos ou em grandes organizações, estimar projetos baseando-se nas memórias dos indivíduos como base de dados histórica é insuficiente. Normalmente os projetos são de natureza mais complexa, cujas informações são antigas e volumosas e o conhecimento acumulado ao longo do tempo é distribuído em um grande número de pessoas.

Figura 1 Base de dados históricos para estimativas [Fernandes, 1995]



O *processo baseado em modelo* faz uso de um modelo de custo de projeto baseado nas *características do sistema que será construído*, no *processo por meio do qual ele será construído* e *no ambiente de desenvolvimento*. O modelo de custo pode ser *informal*, apresentado na forma de um conjunto de diretrizes utilizadas pelos estimadores, ou *formal*, descrito matematicamente.

Modelos *formais* quantificam os dados de entrada do processo de estimativa de custo para depois aplicar um conjunto de equações que calculam as saídas do processo. As equações são desenvolvidas através de análise de dados históricos e devem ser ajustadas para cada ambiente de desenvolvimento.

Segundo Conte [Conte, 1986], os modelos formais mais conhecidos são: o *Modelo de Alocação de Recursos*[1] [Putnam, 1992]; o *Modelo de Pontos de Função* [Albrecht, 1983]; o *Modelo RCA PRICE S*[2]; e o COCOMO (*Constructive Cost Model*) [Boehm, 1981], o modelo mais conhecido, mais completo e documentado de todos os modelos de estimativa.

---

[1] O modelo de Putnam é baseado na curva de Rayleigh que pressupõe uma distribuição de esforço ao longo da existência de um projeto de desenvolvimento de software[Pressman, 1997].

[2] O modelo PRICE S utiliza as variáveis tamanho, tipo e complexidade de projeto como atributos primários para produzir estimativas do custo de funções de sistemas para cada fase do projeto, através do uso da abordagem de cima para baixo ou *top-down* [Conte, 1986].

1.2 O MODELO COCOMO 81

Apresentado em 1981 por Boehm, o COCOMO é um modelo desenvolvido para estimar esforço, prazo, custo e tamanho da equipe para um projeto de software. Todas as referências ao COCOMO encontradas na literatura publicada até 1995 são citações desse modelo.

O COCOMO é apresentado na forma de um conjunto de modelos divididos hierarquicamente em três níveis: *Básico*, *Intermediário* e *Avançado*.

O Modelo 1 – *COCOMO Básico* – calcula o esforço do desenvolvimento de software em função do tamanho estimado do programa expresso em linhas de código estimadas. O Modelo 2 – *COCOMO Intermediário* – calcula o esforço de desenvolvimento de software em função do tamanho do programa e de um conjunto de direcionadores[3] de custo, alternativamente chamados atributos ou fatores de software, que incluem avaliações subjetivas do produto, do hardware, do pessoal e dos atributos do projeto. O Modelo 3 – *COCOMO Avançado* – incorpora todas as características da versão intermediária, incluindo a avaliação do impacto dos atributos do software e da equipe desenvolvedora em cada passo do processo de engenharia de software.

Depois da análise dos requisitos funcionais do software, o tamanho[4] da aplicação deve ser estimado em milhares de linhas de código (KLOC) e o projeto deve ser classificado em um dos três modos de desenvolvimento, identificados por Boehm: *orgânico*, *embutido* ou *semidestacado*.

No *modo orgânico*, equipes relativamente pequenas desenvolvem sistemas num ambiente altamente "familiar", *in-house*. Nesse modo de desenvolvimento, a maior parte das pessoas engajadas no projeto tem experiência prévia com sistemas similares na organização e entendimento completo do sistema. Outras características do modo orgânico são: ambiente estável de desenvolvimento com pouca necessidade de inovação e inexistência requisitos de entrega rígidos; uso de algoritmos simples; e tamanho relativamente pequeno, com projetos na faixa de até 50.000 linhas de código.

---

[3] Um direcionador de custo é uma característica de desenvolvimento de software que tem efeito aumentativo ou diminutivo na quantidade de esforço de desenvolvimento final do projeto, como por exemplo, a experiência da equipe de projeto, ou ainda, a confiabilidade requerida do software [COCOMOII, 2000].

[4] Determinar o tamanho no início do projeto é uma das limitações do método. Uma alternativa viável é a utilização da técnica de contagem de *Pontos de Função*, por ser facilmente efetuada logo no início do projeto. Pontos de função podem ser convertidos em linhas de código [Fernandes, 1995].

O principal fator que distingue um projeto de software de *modo embutido* ou *restrito* é a necessidade de seguir restrições rigorosas. O produto a ser desenvolvido deverá operar dentro de um contexto complexo de hardware, software e regras e procedimentos operacionais. São projetos de software caracterizados por serem relativamente grandes, com muita necessidade de inovação, que demandam altos custos de verificação e validação. Exemplos de projetos do modo embutido são: projeto de sistema de transferência eletrônica de fundos e projeto de sistema de controle de tráfego aéreo.

O *modo semidestacado* é aplicado em projetos de software com características situadas entre os modos orgânico e embutido. Suas características básicas são: a equipe mescla grande e pouca experiência com a aplicação e com a tecnologia e o tamanho do software pode chegar a 300.000 linhas de código.

1.2.1 Equações do modelo COCOMO

As equações de estimativa de *esforço* de desenvolvimento são da forma apresentada na Equação 1, onde *S* é o *tamanho* do software expresso em milhares de linhas de código, excluídos os comentários, e *m(X)* é um *multiplicador* composto que depende de 15 direcionadores de custo. No COCOMO *Básico*, *m(X) = 1* para cada direcionador de custo; no COCOMO *Intermediário* devem ser atribuídos valores específicos para cada atributo.

Equação 1 Estimativa de Esforço de Desenvolvimento

$$E = a_i S^{b_i} m(X)$$

Os modos de desenvolvimento e os valores de a_i e b_i para os níveis *Básico* e *Intermediário* são apresentados na Tabela 1.

Tabela 1 Parâmetros do COCOMO – Estimativa de Esforço [Conte, 1986]

	Básico		***Intermediário***	
Modo	*ai*	*bi*	*ai*	*bi*
Orgânico	2.4	1.05	3.2	1.05
Semidestacado	3.0	1.12	3.0	1.12
Embutido	3.6	1.20	2.8	1.20

Além de estimar o esforço, o COCOMO também apresenta equações para estimar o *prazo de desenvolvimento nominal do projeto* em meses e a *divisão do esforço* por fases e atividades do projeto.

Os modos de desenvolvimento *Básico* e *Intermediário* utilizam as mesmas equações para determinar o prazo. As equações de estimativa de prazo de desenvolvimento são da forma apresentada na Equação 2, onde ***T*** é o tempo de desenvolvimento e ***E*** o esforço já calculado.

Equação 2 Estimativa de Tempo de Desenvolvimento

$$T = a_i E^{b_i}$$

Os modos de desenvolvimento e os valores de a_i e b_i para os níveis *Básico* e *Intermediário* são apresentados na Tabela 2.

Tabela 2 Parâmetros do COCOMO – Estimativa de Prazo [Fernandes, 1995]

Modelos Básico e Intermediário		
Modo	*ai*	*bi*
Orgânico	2.5	0,38
Semidestacado	2.5	0,35
Embutido	2.5	0,32

1.2.2 Direcionadores de custo

Direcionador de custo é uma característica de desenvolvimento de software que tem efeito aumentativo ou diminutivo na quantidade de esforço de desenvolvimento final do projeto. Boehm definiu 15 direcionadores de custo para o COCOMO que, segundo ele, provocam impacto significativo na produtividade e nos custos do projeto.

O fator *m(X)* da equação de estimativa de esforço de desenvolvimento (Equação 1) é nada mais que o produto dos 15 direcionadores de custo. Os 15 atributos de software do modelo COCOMO *Intermediário* estão agrupados dentro de 4 categorias, apresentadas no Quadro 1.

Boehm usou abordagem heurística para determinar o efeito dos direcionadores de custo. O efeito dos direcionadores foi quantizado em escalas de classificação que variam de cinco a seis pontos (Muito Baixo, Baixo, Nominal, Alto, Muito Alto e Extra Alto), dependendo do atributo. Valores numéricos foram atribuídos a cada ponto da escala.

Quadro 1 Atributos Direcionadores de Custo do COCOMO [Conte, 1986]

Categoria	Atributos	
Produto	RELY	*Required Software Reliability (Confiabilidade Requerida do Software)*
	DATA	*Data Base Size (Tamanho da Base de Dados)*
	CPLX	*Product Complexity (Complexidade do Software)*
Computador	TIME	*Execution Time Constraint (Restrição de Tempo de Execução)*
	STOR	*Main Storage Constraint (Restrição de Memória Principal)*
	VIRT	*Virtual Machine Volatility (Volatilidade da Máquina Virtual)*
	TURN	*Computer Turnaround Time (Tempo de Rotatividade do Computador)*
Pessoal	ACAP	*Analist Capability (Capacidade dos Analistas)*
	AEXP	*Applications Experience (Experiência com a Aplicação)*
	PCAP	*Programmer Capability (Capacidade dos Programadores)*
	VEXP	*Virtual Machine Experience (Experiência com Máquina Virtual)*
	LEXP	*Programming Language Experience (Experiência com Linguagem de Programação)*
Projeto	MODP	*Modern Programming Practices (Prática com Programação Moderna)*
	TOOL	*Use of Software Tools (Uso de Ferramentas de Software)*
	SCED	*Required Development Schedule (Prazo Requerido para o Desenvolvimento)*

Definições completas dos direcionador de custo estão descritas em Boehm [Boehm, 1981]. Exemplos de definições e *multiplicadores* para os atributos RELY, STOR e AEXP estão apresentados na Tabela 3 e 4.

Tabela 3 Direcionadores de Custo do COCOMO [Conte, 1986]

Classificações	***RELY*** *(natureza da perda)*	***STOR*** *(% de armazenamento disponível)*	***AEXP*** *(experiência)*
Muito Baixa	Desconforto superficial	.....	< ou = 4 meses
Baixa	Perda facilmente recuperável	.....	1 ano
Nominal	perda moderada	< ou = 50%	3 anos
Alta	Alta perda financeira	70%	6 anos
Muito Alta	Risco de vidas humanas	85%	12 anos
Extra Alta	.....	95%	.....

Tabela 4 Multiplicadores do COCOMO 81 [Conte, 1986]

Direcionador de Custo	***Valores***					
Atributo de Produto	*Muito Baixo*	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
RELY	.75	.88	1.00	1.15	1.40	.....
STOR	.....	.....	1.00	1.06	1.21	1.56
AEXP	1.29	1.13	1.00	.91	.82	....

1.2.3 Exemplo de estimativa de esforço e prazo do modelo COCOMO

A seguir é apresentado um exemplo de uso do modelo COCOMO Intermediário para estimativa de esforço e prazo de um projeto de software.

Foi solicitado o desenvolvimento de um sistema de banco de dados para um projeto de automação de escritório. Segundo o documento de requisitos, o sistema deverá ser composto de quatro módulos, cujo tamanho foi estimado em: *módulo de entrada de dados* – 0.6 KLOC, *módulo de atualização de dados* – 0.6 KLOC, *módulo de consulta* – 0.8 KLOC e *módulo de relatórios* – 1.0 KLOC. O tamanho nominal do sistema é 3.0 KLOC. O modo de desenvolvimento do projeto foi considerado *orgânico* (a = 3.2 e b = 1.05). O gerente avaliou os 15 direcionadores de custo e chegou ao seguinte resultado: *Complexidade Alta* (1.15), *Armazenamento Alto* (1.06), *Experiência Baixa* (1.13) e *Capacidades dos Programadores Baixa* (1.17). Os outros atributos foram considerados nominais.

$$\mathbf{E = a_i S^{b_i} m(X)} \qquad E = 3.2 * (3.0)^{1.05} * (1.15 * 1.06 * 1.13 * 1.17)$$
$$E = 3.2 * 3.17 * 1.61$$
$$\mathbf{E = 16.33 \text{ pessoas/mês}}$$

$$\mathbf{T = 2.5\ E^{38}} \qquad T = 2.5 * 16.33^{0.38}$$
$$\mathbf{T = 7.23 \text{ meses}}$$

1.2.4 Distribuição de esforço e prazo – COCOMO Básico e Intermediário

O COCOMO emprega uma tabela-padrão de tamanho de software (Tabela 5) para cálculo da distribuição de esforço e prazo em projetos de software.

Tabela 5 Tabela Padrão de Tamanho de Software do COCOMO [Fernandes, 1995]

Tamanho do Projeto	*Linhas de Código Entregues (KDSI)*
Pequeno	2
Intermediáio	8
Médio	32
Grande	128
Muito Grande	512

Boehm definiu tabelas (Tabela 6 e 7) baseadas na distribuição de Rayleigh[5], para associar a distribuição percentual de esforço e prazo aos tamanhos de projetos, de acordo com as fases.

---

[5] A distribuição de Rayleigh permite realizar estimativas aproximadas da distribuição de esforço, pessoal necessário, previsão de defeitos, etc.[Fernandes, 1995].

Tabela 6 Distribuição de Esforço [Fernandes, 1995]

Distribuição de Esforço (%)		*Tamanho*				
Modo	***Fase***	***2KDSI***	***8KDSI***	***32KDSI***	***128KDSI***	***512 KDSI***
Orgânico	Planos e Requisitos	6	6	6	6	.....
	Projeto do Produto	16	16	16	16	.....
	Programação	68	65	62	59	.....
	Projeto Detalhado	26	25	24	23	.....
	Codificação	42	40	38	36	.....
	Integração e teste	16	19	22	25	.....
Semidestacado	Planos e Requisitos	7	7	7	7	7
	Projeto do Produto	17	17	17	17	17
	Programação	64	61	58	55	52
	Projeto Detalhado	27	26	25	24	23
	Codificação	37	35	33	31	29
	Integração e teste	19	22	25	28	31
Embutido	Planos e Requisitos	8	8	8	8	8
	Projeto do Produto	18	18	18	18	18
	Programação	60	57	54	51	48
	Projeto Detalhado	28	27	26	25	24
	Codificação	32	30	28	26	24
	Integração e teste	22	25	28	31	34

Tabela 7 Distribuição de Prazo [Fernandes, 1995]

Distribuição de Prazo (%)		*Tamanho*				
Modo	***Fase***	***2KDSI***	***8KDSI***	***32KDSI***	***128KDSI***	***512 KDSI***
Orgânico	Planos e Requisitos	10	11	12	13	.....
	Projeto do Produto	19	19	19	19	.....
	Programação	63	59	55	51	.....
	Integração e teste	18	22	26	30	.....
Semidestacado	Planos e Requisitos	16	18	20	22	24
	Projeto do Produto	24	25	26	27	28
	Programação	56	52	48	44	40
	Integração e teste	20	23	26	29	32
Embutido	Planos e Requisitos	24	28	32	36	40
	Projeto do Produto	30	32	34	36	38
	Programação	48	44	40	36	38
	Integração e teste	22	24	26	28	30

1.2.5 Exemplo de distribuição do quantitativo de pessoal por fase

O procedimento para distribuir esforço e prazo em cada fase do projeto é dado pelo exemplo a seguir [Fernandes, 1995].

Um determinado projeto, cujo tamanho inicialmente foi estimado em 32 KDSI (milhares de linhas de código entregues) e será desenvolvido no modo orgânico. De acordo com a tabela-padrão (Tabela 5), trata-se de um projeto de tamanho médio. A aplicação da equação do esforço resultou em 122 homens/mês. O fator de ajuste (derivado dos multiplicadores) foi calculado em 1.19, e o total de homens/mês foi ajustado para *145 homens/mês*. O prazo foi estimado em *16,5 meses*.

A distribuição de esforço e prazo por fases é obtida pela multiplicação do total de homens/mês e do prazo (Tabelas 8 e 9), respectivamente pelos percentuais das Tabelas 6 e 7.

Tabela 8 Distribuição do esforço para o exemplo [Fernandes, 1995]

Modo Orgânico	Planos e Requisitos	0.06 * 145 = 8.7
	Projeto do Produto	0.16 * 145 = 23.20
	Programação	0.62 * 145 = 89.90
	Integração e Teste	0.22 * 145 = 31.90

Tabela 9 Distribuição do prazo para o exemplo [Fernandes, 1995]

Modo Orgânico	Planos e Requisitos	0.12 * 16.5 = 1.98
	Projeto do Produto	0.19 * 16.5 = 3.14
	Programação	0.55 * 16.5 = 9.08
	Integração e Teste	0.26 * 16.5 = 4.29

A estimativa do quantitativo de pessoal necessário para cada fase é obtida dividindo-se a distribuição dos totais de esforço pelos totais de prazo (Tabela 10).

Tabela 10 Distribuição de pessoal por fase para o exemplo [Fernandes, 1995]

Modo Orgânico	Planos e Requisitos	8.7 ÷ 1.98 = 4.39 ou 4
	Projeto do Produto	23.20 ÷ 3.14 = 7.39 ou 7
	Programação	89.90 ÷ 9.08 = 9.90 ou 10
	Integração e Teste	31.90 ÷ 4.29 = 7.44 ou 7

Para tamanhos de software diferentes dos tamanhos da tabela padrão, mas dentro dos limites de 2 a 512, pode-se empregar a interpolação aritmética para determinar a distribuição de esforço e prazos por fases.

A interpolação aritmética é uma forma de determinar valores “eqüidistantes” entre dois pontos na tabela. Em geral, se existirem dois pontos numa tabela (X_0 ,Y_0) e (X_1 ,Y_1), pode-se encontrar o valor de Y correspondente a um ponto X entre X_0 e Y_1 por interpolação utilizando a seguinte fórmula:

Equação 3 Interpolação Aritmética

$$Y = Y_0 + \frac{X - X_0}{X_1 - X_0}\left(Y_1 - Y_0\right)$$

Para determinar a estimativa de esforço para a fase de integração e teste, supondo uma estimativa de tamanho de 6 KDSI, que fica entre 2 e 8 KDSI (Tabela Padrão) e os valores de estimativa de esforço para a fase de integração e teste, 16 e 19 (Tabela de Distribuição de Esforço), as variáveis da equação assumem os seguintes valores: $Y_0 = 16$, $Y_1 = 19$, $X = 6$, $X_0 = 2$ e $X_1 = 8$.

$$Y = 16 + \frac{6-2}{8-2}\left(19 - 16\right) = 18$$

1.3 FUNDAMENTOS DO MODELO COCOMO

As equações do COCOMO foram derivadas a partir do estudo de uma base de dados de 63 projetos realizados ao longo de um período de 15 anos, de 1964 até 1979, em sua maior parte na empresa *TRW Systems, Inc.*. Os projetos foram implementados em várias linguagens diferentes, entre elas as linguagens Assembly, FORTRAN, COBOL, PL/1 e Jovial e variavam, em tamanho, desde 2.000 até cerca de 1.000.000 de linhas de código (comentários excluídos) e pertenciam a vários domínios de aplicação: negócios; aplicações científicas; sistemas de controle; e sistemas operacionais [Conte, 1986].

A taxa de produtividade nos projetos, em linhas de código, variava de baixa até alta, entre 28 e 1.250 pessoas/mês. Os valores dos atributos direcionadores de custo foram elaborados para cada projeto, através da avaliação de peritos e do uso da técnica Delphi[6].

Para obter as equações do COCOMO, Boehm fez uso da combinação de experiência, obtida com resultados de outros modelos de estimativa de custo, com a opinião subjetiva de gerentes de software experientes, utilizando também tentativa e erro, para obter parâmetros do modelo inicial baseados em um subconjunto da base de dados inteira. Os valores dos parâmetros iniciais foram posteriormente refinados, e calibrados com projetos adicionais da base de dados.

Os 15 fatores de software do COCOMO representam uma redução substancial dos 29 fatores considerados por Walston e Felix. Segundo Conte [Conte, 1986], a redução do número de atributos foi obtida em parte pela combinação de alguns atributos que parecem ser fortemente correlacionados, e em parte pela omissão de outros fatores.

Alguns fatores omitidos por Boehm são *Tipo da Aplicação*, *Nível da Linguagem*, *Qualidade do Gerenciamento*, *Complexidade da Interface com o Cliente*, *Quantidade de Documentação*, *Segurança* e *Restrições de Privacidade*. A justificativa para a omissão desses fatores é dada em [Boehm, 1981].

---

[6] Na técnica *Delphi*, várias pessoas preparam estimativas independentemente. Em seguida todos se reúnem e informam como suas estimativas se comparam umas às outras. Após a primeira reunião, as pessoas recebem permissão para alterar suas estimativas. Existe um processo de iteração na técnica *Delphi*, no qual muitas estimativas gradualmente convergem a um valor mais próximo da realidade e a estimativa mais adequada pode então ser escolhida [Humphrey, 1989].

Ainda segundo Conte [Conte, 1986], atributo que parece provocar maior efeito na produtividade é CPLX – "*Product Complexity* (Complexidade do Produto)". Isso implica que, para um produto de software classificado com complexidade "Baixa", a produtividade pode ser até 236 % maior que um produto classificado com complexidade "Alta". Outros fatores relevantes que juntos podem resultar em ganho de produtividade da ordem de mais de 400% são ACAP – "*Analist Capability* (Capacidade dos Analistas)" e PCAP – "*Programmer Capability* (Capacidade dos Programadores)",

Capítulo 2 O MODELO COCOMO II

2.1 O PROJETO COCOMO II

Para atender às necessidades de um modelo de custo de projeto de software mais adequado às práticas de ciclo de vida de software dos anos 1990 e 2000, teve início, em 1994, o projeto COCOMO II, com o objetivo de desenvolver um modelo de custo mais atualizado que os modelos antecessores, o COCOMO original e o Ada COCOMO[7].

A maioria das referências ao COCOMO encontradas na literatura publicadas a partir de 1995 referem-se ao COCOMO II. Se os termos utilizados no contexto da discussão no modelo COCOMO forem: *Básico*, *Intermediário* ou *Detalhado*, para nomes dos modelos; *Orgânico*, *Semidestacado* ou *Embutido*, para "modo de desenvolvimento", o modelo apresentado refere-se ao COCOMO original, apresentado em 1981. Todavia, se os nomes mencionados dos modelos forem *Application Composition*, *Early Design*, ou *Post-architecture*; ou ainda, se forem mencionados "fatores de escala", o modelo discutido é o COCOMO II.

O projeto COCOMO II foi uma iniciativa da *University of Southern California*, contando inicialmente com a colaboração das empresas *Bellcore*, da *Texas Instruments* e da *Xerox Corporation* como membros afiliados e posteriormente com os seguintes membros: Air Force Cost Analysis Agency, Allied Signal, AT&T, Bellcore, EDS, E-Systems, Hughes, IDA, Litton, Lockheed, Martin, Loral, MCC, MDAC, Motorola, Northrop Grumman, Rational, Rockwell, SAIC, SEI, SPC, SUN, TI, TRW, USAF Rome Lab, US Army Reserach Labs e Xerox.

Para direcionar os trabalhos do projeto COCOMO II, foi levado em consideração um estudo da estimativa do futuro das práticas e dos setores de software para a próxima década de desenvolvimento (Quadro 2).

---

[7] O modelo Ada COCOMO é usado para estimar projetos de sistemas onde a linguagem de programação Ada será implementada. O uso dessa linguagem facilita o desenvolvimento de sistemas altamente confiáveis e o gerenciamento de aplicações complexas. Os direcionadores de custo RELY e CPLX possuem definições próprias [Softstar, 2000].

Quadro 2 Futuro das Práticas de Software [COCOMOII, 2000]

Programação de Usuário Final		
(55 milhões de praticantes nos EUA)		
Geradores de Aplicações e Auxiliares de Composição	***Composição de Aplicações***	***Integração de Sistemas***
600.000	700.000	700.000
Infra-estrutura		
750.000		

Segundo o estudo, por volta do ano 2005 haverá uma grande participação do setor de *Programação de Usuário Final*, com aproximadamente 55 milhões de praticantes nos Estados Unidos e um setor básico, o setor de *Infra-estrutura*, que contará com cerca de 750 mil praticantes.

Três setores serão intermediários: *Geradores de Aplicações e Auxiliares de Composição* (600 mil praticantes); *Composição de Aplicações* (700 mil praticantes); e *Integração de Sistemas de Grande Escala* e/ou *Sistemas de Software Embutidos* (700 mil praticantes).

2.1.1 Programação de Usuário Final

O setor de *Programação de Usuário Final* será impulsionado pelo grande aumento na literatura de computação e pelas pressões competitivas para a elaboração de soluções rápidas e flexíveis voltadas a sistemas de informação.

O projeto COCOMO II considerou que o setor de Programação de Usuário Final não necessita de um modelo de estimativas, visto que as aplicações típicas desse setor são desenvolvidas em horas ou dias, de forma que uma simples estimativa das atividades poderia ser suficiente.

2.1.2 Infra-estrutura

Produtos típicos do setor de *Infra-estrutura* estarão presentes em áreas de sistemas operacionais, sistemas de banco de dados e sistemas de redes de computadores. Segundo o estudo, os *desenvolvedores de infra-estrutura* possuirão bom domínio da ciência da computação e relativamente pouco conhecimento do domínio das aplicações, em contraste com *desenvolvedores usuários finais*, que

possuirão bom conhecimento do domínio de suas aplicações e pouco conhecimento de ciência da computação,

Os praticantes dos três setores intermediários precisarão de bom conhecimento de ciência da computação, além de conhecer um ou mais domínios de aplicações.

2.1.3 Geradores de Aplicações

O setor de *Geradores de Aplicações* criarão muitos elementos pré-elaborados para facilitar o trabalho dos usuários programadores. As linhas de produto desse setor possuirão componentes reutilizáveis e necessitarão de conhecimento adequado para serem bem utilizados.

2.1.4 Composição de Aplicações

O setor de *Composição de Aplicações* estará direcionado a lidar com aplicações por demais diversificadas para serem abrangidas por soluções pré-elaboradas, porém são suficientemente simples para serem compostas a partir de componentes interoperáveis.

Entre esses componentes estarão Sistemas Construtores de *Interface Gráfica para Usuário* – GUI *Graphic User Interface*, gerenciadores de objetos, gerenciadores de bancos de dados, *middleware* para processamento distribuído, *middleware* para processamento de transações, e componentes de domínio específico das áreas de finanças, medicina e processos industriais.

2.1.5 Integração de Sistemas

O setor de *Integração de Sistemas* estará voltado a sistemas de grande escala e altamente embutidos. Partes desses sistemas podem ser desenvolvidos com capacidades de *Composição de Aplicações*, porém seus requisitos necessitarão de grande quantidade de engenharia de sistemas e desenvolvimento de software customizado.

Empresas aeroespaciais e grandes desenvolvedoras de produtos e serviços de telecomunicações, industria automotiva e produtos eletrônicos são exemplos de empresas que operam nesse setor.

2.2 RESULTADOS DO PROJETO COCOMO II

Três modelos de estimativa de custo foram apresentados como resultado do projeto COCOMO II. O modelo *Application Composition*, projetado para estimar projetos desenvolvidos com o uso de ferramentas GUI modernas, baseando-se na contagem de Pontos de Objeto, o modelo *Early Design*, para estimativas preliminares de custo e duração de projetos, antes que a arquitetura completa do sistema tenha sido projetada, utilizando um pequeno conjunto de direcionadores de custo, e o modelo *Post-Architecture*, o modelo mais detalhado, para estimar projetos após a definição completa da arquitetura do sistema.

Capítulo 3 COCOMO II – APPLICATION COMPOSITION

3.1 APLICAÇÃO DO MODELO

O modelo *Application Composition* foi projetado especificamente para o setor *Composição de Aplicações*, cujas aplicações são desenvolvidas por equipes pequenas em poucas semanas ou meses.

O modelo foi projetado para prever *esforço* de prototipação envolvido no uso de ambientes integrados de composição rápida de aplicações de software auxiliados por computador ou ICASE – *Integrated Computer Aided Software Environment* (ambientes construtores de GUI, ferramentas de desenvolvimento de software, etc.).

O modelo prevê a contagem de *Pontos de Objeto*, uma nova abordagem de medição de tamanho de software para estimar custo e prazo de projetos. Pontos de Objeto são um tipo de contagem funcional de telas, relatórios e de módulos de linguagem de terceira geração, onde cada um dos elementos contados recebe uma classificação em níveis de complexidade (simples, média e alta).

Segundo os projetistas do COCOMO II, estimativas de Pontos de Objeto foram comparadas a estimativas de Pontos de Função e mostraram resultados mais adequados aos tipos de aplicação a que se destina o modelo *Application Composition* [COCOMOII, 2000].

3.2 CONTAGEM DE PONTOS DE OBJETO

O procedimento para contagem de Pontos de Objeto apresentado no COCOMO II para estimativa de esforço referente a projetos de composição de aplicações e prototipação é uma síntese do procedimento efetuado por Kauffman e Kumar e de dados de produtividade de 19 projetos [COCOMOII, 2000].

A seguir é apresentado o método de contagem de Pontos de Objeto sugerido pelo COCOMO II.

Para a contagem de Pontos de Objeto, as seguintes etapas deverão ser seguidas:

1. Efetuar *contagens de objetos*: estimar o número de telas, relatórios e componentes de 3GL que constituirão a aplicação. Deve-se assumir as definições padrões desses objetos no ambiente ICASE utilizado.

2. Classificar cada instância de objeto segundo sua *complexidade* (nível *simples*, *médio* ou *difícil*), dependendo dos valores das dimensões características. A Tabela 11 deve ser utilizada.

Tabela 11 Classificação de Pontos de Objeto para Telas [COCOMOII, 2000]

Telas			
Número de Visões contidas	***# e fontes de tabelas de dados***		
	Total < 4	*Total < 8*	*Total 8 ou+*
	(< 2 srvr e < 3 clnt)	*(2-3 srvr e 3-5 clnt)*	*(> 3 srvr e > 5 clnt)*
< 3	simples	simples	média
3 a 7	simples	média	difícil
> 8	média	difícil	difícil
Relatórios			
Número de Seções Contidas	***# e fontes de tabelas de dados***		
	Total < 4	*Total < 8*	*Total 8 ou+*
	(< 2 srvr e < 3 clnt)	*(2-3 srvr e 3-5 clnt)*	*(> 3 srvr e > 5 clnt)*
0 ou 1	simples	simples	média
2 ou 3	simples	média	difícil
4 ou +	média	difícil	difícil

NOP *New Object Points*. Novos Pontos de Objeto contados, ajustados para reuso.
Srvr Número de servidores (*mainframe* ou equivalente) de tabelas de dados usados em conjunção com *TELA* ou *RELATÓRIO*.
Clnt Número de clientes (estação de trabalho pessoal) de tabelas de dados usados em conjunção com *TELA* ou *RELATÓRIO*.
%reuso Porcentagem de telas, relatórios e módulos 3GL (linguagem de terceira geração) reusados a partir de aplicações prévias.

3. Atribuir um número para cada atributo utilizando-se a Tabela 12. Os pesos refletem o esforço relativo necessário à implementação.

Tabela 12 Pesos de Complexidade de Pontos de Objeto [COCOMOII, 2000]

Tipo de Objeto	***Peso da Complexidade***		
	Simples	***Média***	***Difícil***
Telas	1	2	3
Relatórios	2	5	8
Componentes 3GL	.....	.....	10

4. Determinar os Pontos de Objeto. Somar todos os pesos para obter o total de Pontos de Objeto.

5. Estimar a porcentagem de reuso esperada no projeto que está sendo contado. Computar os novos Pontos de Objeto (NOP) que serão desenvolvidos utilizando-se a Equação 4. O modelo para cálculo de reuso está detalhado em [COCOMOII, 2000].

Equação 4 Pontos de Objeto e Reuso

$$NOP = \frac{(PontosdeObjeto) - (100 - \%reuso)}{100}$$

6. Determinar a taxa de produtividade utilizando-se a Equação 5. A taxa de produtividade é estimada analisando-se a média subjetiva da *experiência dos desenvolvedores* e da *maturidade com o uso de ICASE* (Tabela 13).

Equação 5 Application Composition – Produtividade dos Desenvolvedores

$$PROD = \text{NOP/Pessoas-mês}$$

Tabela 13 Produtividade de Desenvolvedores [COCOMOII, 2000]

Experiência e Capacidade dos Desenvolvedores / ***Maturidade e Capacidade de ICASE***	***Muito Baixa***	***Baixa***	***Nominal***	***Alta***	***Muito Alta***
Produtividade	4	7	13	25	50

7. No modelo COCOMO II, o esforço é expresso em pessoas/mês[8] (PM). O esforço deve ser computado utilizando-se a Equação 6.

Equação 6 Esforço e Pontos de Objeto

$$PM = \frac{NOP}{PROD}$$

[8] Uma pessoa mês é a quantidade de tempo correspondente ao gasto de uma pessoa trabalhando em um projeto de desenvolvimento de software em um mês. Esse número não inclui feriados ou férias e considera tempo livre nos finais de semana. O número de pessoas mês é diferente do tempo necessário para completar um projeto; um projeto pode ser estimado em 50 PM de esforço e 11 meses de cronograma [COCOMOII, 2000].

Capítulo 4 COCOMO II – EARLY DESIGN

4.1 APLICAÇÃO DO MODELO

As etapas posteriores às fases iniciais de projetos ou ciclos de vida espirais podem requerer a pesquisa de arquiteturas alternativas ou estratégias de desenvolvimento incremental. O modelo *Early Design* do COCOMO II foi desenvolvido para elaborar estimativas iniciais e apoiar essas atividades.

O nível de detalhes que o modelo utiliza é consistente com o nível de informações disponíveis e com o nível de precisão necessário nessa fase. As principais variáveis de entrada para o modelo são: o *tamanho estimado do sistema* e os *direcionadores de custo*.

O *tamanho* da aplicação que vai ser desenvolvida é a principal variável de entrada para cálculo de esforço e tempo. Derivado da estimativa do tamanho dos módulos de software que irão constituir a aplicação ou programa, o tamanho deve ser expresso em unidade de milhares de linhas de código (KLOC).

O modelo *Early Design* permite que o tamanho da aplicação possa ser estimado em pontos de função não ajustados, posteriormente convertidos em linhas de código pelo modelo. Pontos de função podem ser utilizados como boas estimativas porque são baseados em informações que estão disponíveis logo no início do ciclo de vida do projeto.

4.2 PONTOS DE FUNÇÃO

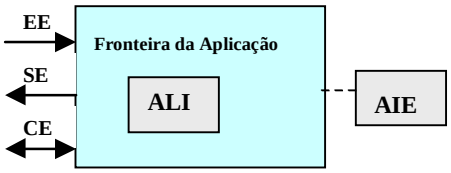
Pontos de função medem o tamanho funcional[9] de um projeto de software através da quantificação da informação da funcionalidade de processamento da aplicação. Para contar pontos de função, cinco tipos de funções ou componentes lógicos da aplicação (Figura 2) devem ser avaliados: Arquivos Lógicos Internos

---

[9] Tamanho funcional é uma medida do tamanho de um software baseada em uma avaliação padronizada dos requisitos lógicos sob o ponto de vista do usuário da aplicação [Dekkers, 1998].

(ALI), Arquivos de Interface Externa (AIE), Entradas Externas (EE), Saídas Externas (SE) e Consultas Externas (CE).

Figura 2 Componentes de Pontos de Função [Dekkers, 1998]



Depois de identificadas, cada uma das funções deve ser classificada quanto à sua *complexidade funcional* relativa: *simples*, *média* ou *complexa*[10]. Em seguida, calcula-se os *pontos de função brutos* ou *não ajustados* – métrica de tamanho utilizada pelo COCOMO II – através da aplicação de pesos de acordo com tabelas específicas para cada tipo de função.

O processo usual de contagem de pontos de função prevê o ajuste dos pontos de função contados [Braga, 1996], aplicando-se um fator obtido na avaliação do grau de influência de 14 características de projeto de software, tais como volume de transações, performance e atualização *on-line* do sistema.

O COCOMO II utiliza apenas os *pontos de função não ajustados* para estimar o tamanho da aplicação porque o grau de contribuição das 14 características do projeto no esforço estimado é considerado inconsistente com a experiência do COCOMO.

A contagem de pontos de função não ajustados do COCOMO II deve ser realizada de acordo com o processo descrito a seguir. Esse processo pode ser utilizado nos modelos *Early Design* e *Post-Architecture*.

4.2.1 Contagem de Pontos de Função

1. Determinar os tipos de função. A contagem de pontos de função não ajustados deve ser efetuada por uma pessoa treinada na técnica de pontos de função, baseando-se na informação disponibilizada no documento de requisitos do software e nos documentos do projeto disponíveis (modelo de entidade-relacionamento, *layouts* de telas, relatórios, etc.). A descrição dos cinco tipos de função está apresentada no Quadro 3.

[10] As regras para determinação do nível de complexidade são definidas pelo [IFPUG, 2000].

Quadro 3 Tipos de Função

Tipo	**Identificação**	**Exemplo**
Arquivo Lógico Interno (ALI)	Grupo de dados logicamente relacionados, identificado pelo usuário, mantido e modificado dentro da fronteira da aplicação[11] que está sendo contada.	Arquivo de clientes.
Arquivo de Interface Externa (AIE)	Grupo de dados, logicamente relacionados, utilizado no sistema que está sendo analisado, mas que não é mantido nem modificado dentro da fronteira da aplicação que está sendo contada.	Informações de acesso ao sistema principal, que se encontram em outra aplicação, num suposto arquivo *senhas*.
Entrada Externa (EE)	Função que processa dados ou informações de controle geradas por uma fonte externa à aplicação que está sendo contada. São transações que efetuam manutenção nos dados armazenados no sistema.	Funções de *inclusão*, *alteração* e *exclusão* em um sistema.
Saída Externa (SE)	Função que fornece dados ou informações de controle para fora da aplicação que está sendo contada. São transações que extraem informações do sistema para outros aplicativos.	Função de cálculo de folha de pagamento baseado na soma dos totais de horas trabalhadas de funcionários.
Consulta Externa (CE)	Transação que combina transações de entrada e saída, resultando em recuperação de dados.	Relatório com os dados de um cliente.

2. Determinar o nível de complexidade das funções contadas. Classificar cada função em um nível de complexidade (Baixa, Média ou Alta), dependendo do número de elementos de dados contidos e do número de tipos de arquivos referenciados. A Tabela 14 deve ser utilizada.

Tabela 14 Classificação dos Tipos de Função [COCOMOII, 2000]

ALI e AIE			
Elementos de Registro	*Elementos de dados*		
	1 a 19	*20 a 50*	*51 ou +*
1	Baixa	Baixa	Média
2 a 5	Baixa	Média	Alta
6 ou +	Média	Alta	Alta
SE e CE			
Tipos de Arquivos	*Elementos de dados*		
	1 a 5	*6 a 19*	*20 ou +*
0 ou 1	Baixa	Baixa	Média
2 a 3	Baixa	Média	Alta
4 ou +	Média	Alta	Alta
EE			
Tipos de Arquivos	*Elementos de dados*		
	1 a 4	*5 a 15*	*16 ou +*
0 ou 1	Baixa	Baixa	Média
2 a 3	Baixa	Média	Alta
3 ou +	Média	Alta	Alta

3. Aplicar pesos de acordo com a complexidade. Um número deve ser atribuído para cada tipo de função (Tabela 15).

[11] A fronteira da aplicação é a linha que separa o projeto que está sendo contado de outras aplicações ou sistemas da organização [Braga, 1996].

Tabela 15 Complexidade dos Tipos de Função [COCOMOII, 2000]

Tipo de Função	*Peso da Complexidade*		
	Baixa	*Média*	*Alta*
ALI	7	10	15
AIE	5	7	10
EE	3	4	6
SE	4	5	7
CE	3	4	6

4. Computar os pontos de função não ajustados. Totalizar todas as contagens dos tipos de função para obter o total de pontos de função não ajustados.

Para determinar a quantidade de pessoas/mês, os pontos de função não ajustados devem ser convertidos em linhas de código, de acordo com a linguagem de implementação escolhida (Assembly, linguagem de Quarta-Geração, etc.). Os modelos *Early Design* e *Post-Architecture* utilizam uma tabela para converter pontos de função não ajustados na quantidade equivalente de linhas de código (Tabela 16), como sugerido em [Jones, 1991][COCOMOII, 2000] .

Tabela 16 Conversão de PF em Linhas de Código [COCOMOII, 2000]

Linguagem	*SLOC/UFP*	*Linguagem*	*SLOC/UFP*
Ada	71	ANSI Cobol 85	91
AI Shell	49	Fortran 77	105
APL	32	Forth	64
Assembly	320	Jovial	105
Assembly (Macro)	213	Lisp	64
ANSI/Quick/Turbo Basic	64	Modula2	80
Basic Compilado	91	Pascal	91
Basic Interpretado	128	Prolog	64
C	128	Report Generator	80
C++	29	Spreadsheet	6

4.3 DIRECIONADORES DE CUSTO

O modelo *Early Design* utiliza um conjunto reduzido de direcionadores de custo. Esses direcionadores de custo foram obtidos através da combinação dos direcionadores de custo do modelo *Post-Architecture* (Quadro 4).

As escalas correspondentes e os multiplicadores de esforço dos direcionadores de custo do modelo *Early Design* mantém relacionamento consistente com os direcionadores combinados do modelo *Post-Architecture*, para assegurar a consistência das estimativas dos dois modelos.

Quadro 4 Direcionadores de Custo [COCOMOII, 2000]

Modelo Early Design	***Modelo Post-Architecture***
Direcionadores de Custo	*Direcionadores de Custo combinados*
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

O mapeamento dos direcionadores de custo e escalas de classificação do modelo *Post-Architecture* para o modelo *Early Design*, foi feito com o uso e combinação de equivalentes numéricos do níveis de classificação.

Os valores numéricos dos direcionadores de custo do modelo *Post-Architecture* (*Muito Baixa* corresponde a uma classificação numérica de 1, *Baixa* é 2, *Nominal* é 3, *Alta* é 4, *Muito Alta* é 5 e *Extra Alta* é 6) foram somados e os totais resultantes foram distribuídos em uma escala de classificação expandida, que varia de *Extra Baixo* até *Extra Alto*.

4.3.1 Direcionador de Custo RCPX – Confiabilidade e a Complexidade do Produto

O direcionador de custo *RCPX* (Tabela 17) combina quatro direcionadores de custo: *Confiabilidade Requerida do Software* (RELY), *Tamanho da Base de Dados* (DATA), *Complexidade do Produto* (CPLX) e *Documentação adequada às necessidades do ciclo de vida* (DOCU).

Tabela 17 Escala de Classificação do Direcionador *RCPX* [COCOMOII, 2000]

	Extra Baixo	***Muito Baixo***	***Baixo***	***Nominal***	***Alto***	***Muito Alto***	***Extra Alto***
Total de RELY, DATA, CPLX E DOCU	5,6	7,8	9 a 11	12	13 a 15	16 a 18	19 a 21
Ênfase na Combiabilidade, Documentação	Muito Pouco	Pouco	Alguma	Básico	Forte	Muito Forte	Extrema
Complexidade do Produto	Muito Simples	Simples	Alguma	Moderada	Complexa	Muito Complexa	Extramamente Complexa
Tamanho da Base de Dados	Pequena	Pequena	Pequena	Moderado	Grande	Muito Grande	Muito Grande

4.3.2 Direcionador de Custo RUSE – Reuso Requerido

O direcionador de custo RUSE é o mesmo direcionador do modelo *Post-Architecture*. O sumário dos níveis de classificação desse direcionador é dado no Quadro 5.

Quadro 5 Escala de Classificação do Direcionador *RUSE* [COCOMOII, 2000]

	Muito Baixo	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
RUSE		nenhum	Projeto	Programa	Linha de produto	Múltiplas linhas de produto

4.3.3 Direcionador de Custo PDIF – Dificuldade de Plataforma

O direcionador PDIF (Tabela 18) combina três direcionadores de custo: *Tempo* (TIME), *Restrição de Armazenamento Principal* (STOR) e *Volatilidade de Plataforma* (PVOL). As escalas de TIME e STOR variam de *Nominal* a *Extra Alto* e PVOL varia de *Baixo* até *Muito Alto*. Os totais numéricos da soma de suas classificações varia de 8 (N, N, Baixo) até 17 (Extra Alto, Extra Alto, Muito Alto).

Tabela 18 Escala de Classificação do Direcionador *PDIF* [COCOMOII, 2000]

	Baixo	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
Total de TIME, STOR E PVOL	8	9	10 a 12	13 a 15	16 a 17
Tempo e restrição de armazenamento	50%	50%	65%	80%	90%
Volatilidade da plataforma	Muito estável	Estável	Um tanto volátil	Volátil	Altamente Volátil

4.3.4 Direcionador de Custo PERS – Capacidade de Pessoal

O direcionador de custo *PERS* – Capacidade de Pessoal ou Equipe Desenvolvedora (Tabela 19) combina os direcionadores de custo *Capacidade do Analista* (ACAP), *Capacidade do Programador* (PCAP) e *Continuidade de Pessoal* (PCON). Cada um deles possui uma escala de classificação que varia de *Muito Baixo* (= 1) até *Muito Alto* (= 5).

Tabela 19 Escala de Classificação do Direcionador *PERS* [COCOMOII, 2000]

	Extra Baixo	***Muito Baixo***	***Baixo***	***Nominal***	***Alto***	***Muito Alto***	***Extra Alto***
Total de ACAP, PCAP, PCON	3,4	5,6	7,8	9	10,11	12,13	14.15
Porcentagem combinada de ACAP e PCAP	20%	39%	45%	55%	65%	75%	85%
Rotatividade Anual de Pessoal	45%	30%	20%	12%	9%	5%	4%

4.3.5 Direcionador de Custo PREX – Experiência do Pessoal

O direcionador de custo PREX (Tabela 20) combina três direcionadores de custo: *Experiência com a Aplicação* (AEXP), *Experiência com a Plataforma* (PEXP) e *Experiência com a Linguagem e Ferramentas* (LTEX). Cada um desses direcionadores varia de *Muito Baixo* a *Muito alto*.

Tabela 20 Escala de Classificação do Direcionador *PREX* [COCOMOII, 2000]

	Extra Baixo	***Muito Baixo***	***Baixo***	***Nominal***	***Alto***	***Muito Alto***	***Extra Alto***
Total de AEXP, PEXP E LTEX	*3,4*	*5,6*	7,8	9	10,11	12,13	14,15
Experiêcia com aplicações, plataforma, linguagem e ferramentas	*3 meses*	*5 meses*	9 meses	1 anos	2 anos	4 anos	6 anos

4.3.6 Direcionador de Custo FCIL – Facilidades

O direcionador de custo FCIL (Tabela 21) combina dois direcionadores: Uso de Ferramentas de Software (TOOL) e Desenvolvimento *Multisite* (SITE). A escala do direcionador TOOL varia de *Muito Baixo* a *Muito Alto* e SITE de *Muito Baixo* a *Extra Alto*. A soma dessas classificações varia de 2 (Muito Baixo, Muito Baixo) até 11 (Muito Alto, Extra Alto).

Tabela 21 Escala de Classificação do Direcionador *FCIL* [COCOMOII, 2000]

	Extra Baixo	*Muito Baixo*	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
Total de TOOL e SITE	2	3	4,5	6	7,8	9,1	11
Ferramenta de apoio	Mínimo	Alguma	Ferramenta CASE simples	Ferramentas básicas de ciclo de vida	Bom; moderado	Forte; moderado	Forte; bem integrada
Condições de desenvto. multisite	Fraco apoio ao desenvto. multisite complexo	Algum apoio ao desenvolv. multisite complexo	Algum apoio ao desenvto. multisite moderadte. complexo	Apoio básico ao desenvto. multisite moderadte. complexo	Forte apoio ao desenvto. multisite moderadte. complexo	Forte apoio ao desenvto. multisite simples	Apoio muito forte ao desenvto. multisite combinado ou simples

4.3.7 Direcionador de Custo SCED – Cronograma

O direcionador de custo SCED é o mesmo do modelo *Post-Architecture*. Os níveis de classificação desse direcionador de custo estão apresentados na Tabela 22.

Tabela 22 Escala de Classificação do Direcionador *SCED* [COCOMOII, 2000]

	Muito Baixo	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
SCED	75% do nominal	85%	100%	130%	160%	

4.4 FATORES DE ESCALA

Modelos de custo de software geralmente possuem um fator exponencial para levar em consideração “economias” e “deseconomias” de escala encontradas em diferentes tamanhos de projeto. O expoente B da equação de estimativa de esforço (Equação 7) é utilizado para capturar esses efeitos.

A Equação 7 é básica para cálculo de esforço de desenvolvimento de projeto nos modelos *Early Design* e *Post-Architecture*. As variáveis de entrada são o *tamanho (S)*, a constante *a* e o fator escalar *b*.

Equação 7 Esforço – Early Design & Post-Architecture

$$PM_{Nominal} = a * (S)^{b}$$

Se B < 1.0, o projeto apresenta economia de escala. Se o tamanho do produto for dobrado, o esforço necessário ao projeto ainda é menor que o dobro. A produtividade do projeto, nesse caso, aumenta à medida que o tamanho do produto aumenta.

Se B = 1.0, as economias e deseconomias de escala estão balanceadas. Esse modelo linear é geralmente usado para estimar custo de pequenos projetos e é utilizado pelo modelo Application Composition.

Se B > 1.0, o projeto apresenta deseconomia de escala. Isto é normalmente devido a 2 fatores principais: crescimento de taxas de comunicação interpessoal e de integração de grandes sistemas.

Os fatores de escala do COCOMO II substituem os modos de desenvolvimento do modelo COCOMO original *Orgânico*, *Semidestacado* e *Embutido*, refinando também os quatro fatores de escala do modelo Ada COCOMO. Cada direcionador escalar possui níveis de classificação que variam de *Muito Baixo* até *Extra Alto*. Os respectivos valores estão apresentados na Tabela 23.

Tabela 23 Fatores de Escala Calibrados [COCOMOII, 2000]

Fatores de Escala	*Muito Baixo*	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
Precedência	4,05	3,24	2,43	1,62	0,81	0
Flexibilidade de desenvolvimento	6,07	4,86	3,64	2,43	1,21	0
Arquitetura/Resolução de Risco	4,22	3,38	2,53	1,69	0,84	0
Coesão da Equipe	4,94	3,95	2,97	1,98	0,99	0
Maturidade do Processo	4,54	3,64	2,73	1,82	0,91	0

4.4.1 Fatores de Escala Precedência (PREC) e Flexibilidade de Desenvolvimento (FLEX)

Esses dois fatores de escala procuram condensar as diferenças existentes nos modos de desenvolvimento *Orgânico*, *Semidestacado* e *Embutido* do COCOMO original. O Quadro 6 apresenta o mapeamento das características do projeto nesses dois fatores.

4.4.2 Fatores de Escala Arquitetura e Resolução de Risco (RESL)

O fator de escala RESL combina dois fatores do modelo Ada COCOMO. A Tabela 24 apresenta as classificações do modelo Ada COCOMO utilizados para formar uma definição mais compreensiva dos níveis de classificação do fator RESL.

Quadro 6 Fatores de Escala do COCOMO original [COCOMOII, 2000]

Característica	***Muito Baixo***	***Nominal/Alto***	***Extra Alto***
Precedência			
Compreensão organizacional dos objetivos do produto	Geral	Considerável	Completo
Experiência no trabalho com sistemas de software relacionados	Moderado	Considerável	Extensivo
Desenvolvimento simultâneo de novos hardware associados e procedimentos operacionais	Extensivo	Moderado	Algum
Necessidade de novas arquiteturas de processamento de dados e algoritmos	Considerável	Algum	Mínimo
Flexibilidade de desenvolvimento			
Necessidade de conformidade do software com requisitos pré-estabelecidos	Total	Considerável	Básica
Necessidade de conformidade do software com especificações de interface externa	Total	Considerável	Básica
Prêmio por término antecipado	Alto	Médio	Baixo

Tabela 24 Componentes da escala do Fator RESL [COCOMOII, 2000]

Característica	***Muito Baixo***	***Baixo***	***Nominal***	***Alto***	***Muito Alto***	***Extra Alto***
O Plano de Gerenciamento de Risco identifica todos os itens de risco críticos, estabelece marcos de referência como solução através de PDR - Revisão do Projeto do Produto	Nenhum	Pouco	Algum	Geralmente	Na maioria das vezes	Plenamente
Cronograma, orçamento e marcos internos através de PDR compatível com o Plano de Gerenciamento de Risco	Nenhum	Pouco	Algum	Geralmente	Na maioria das vezes	Plenamente
Porcentagem do cronograma de desenvolvimento voltado à instituição da arquitetura, dados os objetivos gerais do produto	5	10	17	25	33	40
Porcentagem de projetistas disponíveis para o projeto	20	40	60	80	100	120
Ferramenta de apoio disponível para resolução de itens de risco, desenvolvimento e verificação de incertezas da arquitetura	Nenhum	Pouco	Algum	Bom	Forte	Pleno
Nível de incertezas nos direcionadores chave da arquitetura: missão, interface do usuário, COTS, hardware, tecnologia, performance	Extrema	Significante	Considerável	Algum	Pouco	Muito Pouco
Número e criticidade de itens de risco	> 10 Crítico	5 a 10 Crítico	2 a 4 Crítico	1 Crítico Crítico	> 5 Não Crítico	< 5 Não Crítico

4.4.3 Fator de Escala Coesão de Equipe (TEAM)

O fator de escala TEAM diz respeito à turbulência e a entropia do projeto. Entropia é a medida da desordem ou ocorrências ao acaso devido às dificuldades na sincronização dos elementos que tem parte ou são interessados no projeto (*stakeholders*): usuários, clientes, desenvolvedores, pessoal de manutenção, interfaces, e outros. Essas dificuldades podem aparecer com as diferenças nos objetivos e na cultura dos *stakeholders* com as dificuldades na reconciliação dos objetivos individuais e com a falta de experiência e familiaridade na atuação dos elementos como uma equipe.

O Quadro 7 indica detalhes das definições nos níveis do fator de escala TEAM.

Quadro 7 Componentes da escala do Fator TEAM [COCOMOII, 2000]

Característica	*Muito Baixo*	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
Consistência dos objetivos e da cultura dos stakeholders	Pouco	Alguma	Básica	Considerável	Forte	Plena
Habilidade, disposição dos stakeholders para incorporar os objetivos dos outros stakeholders	Pouco	Alguma	Básica	Considerável	Forte	Plena
Experiência dos stakeholders na atuação como equipe	Nenhum	Pouco	Pouco	Básica	Considerável	Extensivo
Sinergia dos stakeholders para obter visão partilhada e comprometimento	Nenhum	Pouco	Pouco	Básica	Considerável	Extensivo

4.4.4 Fator de Escala Maturidade do Processo (PMAT)

O procedimento para determinação do fator de escala PMAT está baseado no modelo de maturidade CMM[12] da SEI.

---

[12] CMM – *Capability Maturity Model* é um modelo de desenvolvimento para aplicação específica ao software, dentro do contexto de *Qualidade Total* no âmbito de uma organização. O CMM provê uma estrutura para avaliação e melhoria, baseado principalmente na experiência da comunidade de software da indústria americana, com a utilização de conceitos de gerenciamento de processos de autores consagrados, cujos modelos foram amplamente aplicados em outras áreas. Segundo o modelo CMM, o melhoramento contínuo do processo é baseado em passos evolucionários pequenos ao invés de inovações revolucionárias. O CMM oferece um *framework* para organizar passos evolucionários dentro de cinco níveis de maturidade que aplicam sucessivos fundamentos para a melhoria contínua do processo [PAULK et. al. 1993].

Há duas formas para classificar o nível de maturidade do processo. A primeira delas é utilizar o resultado da avaliação do CMM efetuada na organização. A organização pode ser classificada em cinco níveis (Quadro 8): Nível 1 – (médio baixo), Nível 1 – (médio alto), Nível 2, Nível 3, Nível 4 e Nível 5.

A Segunda forma é através da avaliação das 18 *Áreas Chave de Processo* (KPAs[13]) do modelo CMM pela organização. O procedimento para determinar o fator de escala PMAT sugerido pelo modelo COCOMO II está apresentado a seguir.

A organização deve realizar uma avaliação e decidir a porcentagem de adequação de suas práticas dentro de cada uma das KPAs.

Quadro 8 Áreas Chave de Processo do CMM [PAULK, 1992]

Nível		Foco	Área chave de processo (KPA) executadas
5	OTIMIZADO	Melhoria contínua do processo	3- Gerenciamento da Mudança no Processo
			2- Gerenciamento da Mudança Tecnológica
			1- Prevenção de Defeito
4	GERENCIADO	Qualidade do produto e do processo	2- Gerenciamento da Qualidade de Software
			1- Gerenciamento Quantitativo do Processo
3	DEFINIDO	Processos de engenharia e suporte organizacional	7- Revisões (peer review)
			6- Coordenação entre grupos
			5- Engenharia de Produto de Software
			4- Gerenciamento de Software Integrado
			3- Programa de Treinamento
			2- Definição do Processo da Organização
			1- Foco no Processo da Organização
2	REPETÍVEL	Processos de Gerenciamento de Projetos	6- Gerenciamento da Configuração de Software
			6- Gerenciamento da Configuração de Software
			5- Garantia da Qualidade de Software
			4- Gerenciamento de Subcontrato de Software
			3- Acompanhamento de Projeto de Software
			2- Planejamento de Projeto de Software
1	INICIAL	Pessoas competentes e heróicas	1- Gerenciamento de Requisitos

O nível de adequação é determinado através de um julgamento das práticas em relação às metas de cada KPA (Quadro 9).

[13] Cada nível de maturidade do modelo CMM é composto de várias Áreas-Chave de Processo (KPAs – *Key Process Areas*). As KPAs identificam as questões que devem ser discutidas para alcançar um nível de maturidade, agrupando um conjunto de atividades relacionadas que, quando executadas coletivamente, fazem com que a organização atinja um conjunto de objetivos considerados importantes para aumentar a capacidade do processo [PAULK et. al. 1993].

As KPAs são avaliadas de acordo com o seguinte critério:

- Marcar ❶ *Quase Sempre* quando os objetivos da KPA são consistentemente obtidos e estão bem estabelecidos em procedimentos de operação padronizados (mais que 90% das vezes).
- Marcar ❷ *Freqüentemente* quando os objetivos da KPA são geralmente obtidos, mas algumas vezes são omitidos, devido a circunstâncias difíceis (de 60 a 90% das vezes).
- Marcar ❸ *Cerca de Metade* quando os objetivos da KPA são atingidos cerca de *metade das vezes* (de 40 a 60% das vezes).
- Marcar ❹ *Ocasionalmente* quando os objetivos da KPA são *algumas vezes atingidos*, porém menos que geralmente (de 10 a 40% das vezes).
- Marcar ❺ *Raramente* se alguma vez os objetivos são *dificilmente obtidos* (menos que 10% das vezes).
- Marcar ❻ *Não se Aplica* quando a organização possui conhecimento necessário sobre o projeto ou organização e a KPA, porém nota-se que a KPA não se aplica às circunstâncias.
- Marcar ❼ *Não Sabe* quando há incertezas a respeito de como avaliar a KPA.

Quadro 9 Avaliação das Práticas do Modelo CMM (COCOMOII, 2000]

	Áreas Chave de Processo (KPAs)	***1***	***2***	***3***	***4***	***5***	***6***	***7***
1	*Gerenciamento de Requisitos*							
2	*Planejamento do Projeto de Software*							
3	*Acompanhamento e Verificaçã do Projeto*							
4	*Gerenciamento de Subcontratados*							
5	*Garantia de Qualidade de Software*							
6	*Gerenciamento de Configuração*							
7	*Foco no Processo Organizacional*							
8	*Definicação do Processo Organizacional*							
9	*Programa de Treinamento*							
10	*Gerenciamento Integrado de Software*							
11	*Engenharia de Produto de Software*							
12	*Coordenação Integrupo*							
13	*Revisões*							
14	*Gerenciamento Quantitativo do Processo*							
15	*Gerenciamento da Qualidade do Software*							
16	*Prevenção de Defeitos*							
17	*Gerenciamento de Mudança de Tecnologia*							
18	*Gerenciamento de Mudança do Processo*							

Depois que cada os níveis das KPAs foram avaliados, determina-se o fator PMAT utilizando-se a Equação 8.

Equação 8 Maturidade do Processo (*PMAT*)

$$5 - \left[\sum_{i=1}^{18} \left(\frac{KPA\%_i}{100} * \frac{5}{18} \right) \right]$$

4.5 EQUAÇÕES DO MODELO *EARLY DESIGN*

A Equação 9 é utilizada para calcular o esforço de desenvolvimento em projetos estimados pelo modelo *Early Design*.

Equação 9 Esforço de Desenvolvimento – *Early Design*

$$PM = a * [Tamanho']^b * \prod_{i=1}^{7} EM_i + PM_M$$

Onde:

PM Esforço estimado em Pessoas/Mês

a Constante, provisoriamente ajustada em 2.5

Tamanho Tamanho estimado, expresso em milhares linhas de código (KLOC)

b Fator escalar

EM Multiplicadores de esforço: RCPX, RUSE, PDIF, PERS, PREX, FCIL, SCED

PM_M Estimativa de esforço de manutenção.

O coeficiente *b* é calculado com a Equação 10.

Equação 10 Cálculo do coeficiente *b*

$$b = 1.01 + 0.01 * \sum_{j=1}^{5} SF_j$$

SF Fatores de escala: PREC, FLEX, RESL, TEAM, PMAT

O COCOMO II utiliza o fator ou variável *porcentagem de quebra* BRAK (Equação 11) para ajustar o tamanho efetivo do produto de software. Essa variável reflete a porcentagem de código descartado devido à volatilidade dos requisitos. Por exemplo, em um projeto que produz 100.000 instruções e descarta o equivalente a 20.000 instruções, BRAK equivale a 20 %. O fator BRAK não é utilizado no modelo *Application Composition*.

Equação 11 Porcentagem de Quebra – BRAK

$$Tamanho' = Tamanho * \left(1 + \frac{BRAK}{100}\right)$$

BRAK Quebra: porcentagem de código descartado devido à volatilidade dos requisitos. É a quantidade de mudança controlada permitida em um desenvolvimento de software antes que os requisitos sejam “congelados”

O modelo de reuso de componentes, código e os detalhes das equações (Equações 12, 13 e 14) utilizadas para o cálculo de estimativas são apresentados e detalhados no manual do modelo COCOMO II [COCOMO II, 2000].

Equação 12 Reuso de Componentes

$$Tamanho = KNSLOC + KASLOC * \left(\frac{100 - AJ}{100} \right) * (AAM)$$

KNSLOC Tamanho dos componentes adaptados, expressos em milhares de linhas de código adaptadas.
KASLOC Tamanho dos componentes, expressos em milhares de novas linhas de código

Equação 13 Fatores para Equação de Reuso de Componentes (1)

$$AAM = \left\{ \frac{AA + AAF * (1 + 0/02(SU)(UNFM)}{100}, AAF \leq 0.05 \right.$$

ou

$$AAM = \left\{ \frac{AA + AAF + (SU) * (UNFM)}{100}, AAF > 0.05 \right.$$

AA Avaliação e assimilação
AAF Produtividade de tradução automática
SU Compreensão do software (zero se DM = 0 e CM = 0)
UNFM Desconhecimento (não está familiarizado) do programador em relação ao software

Equação 14 Fatores para Equação de Reuso de Componentes (2)

$$AAF = 0.4(DM) + 0.3(CM) + 0.3(IM)$$

DM Porcentagem de projeto modificado
CM Porcentagem de código modificado
IM Porcentagem de integração e teste modificado

O COCOMO II utiliza um modelo de reuso para cálculo de manutenção. A Equação 15 é utilizada para calcular a estimativa de esforço de manutenção.

Equação 15 Manutenção de Projeto

$$PM_M = \frac{ASLOC\left(\frac{AT}{100}\right)}{ATPROD}$$

ASLOC Tamanho dos componentes adaptados expresso em linhas de código
AT Porcentagem dos componentes que são automaticamente traduzidos
ATPROD Produtividade de tradução automática

4.6 EQUAÇÃO PARA ESTIMAR CRONOGRAMA DE DESENVOLVIMENTO

Para estimar o cronograma de desenvolvimento de projetos, os modelos de estimativa do COCOMO II utilizam a Equação 16.

Equação 16 Estimativa de Prazo

$$TDEV = \left[a * PM^{(0.33+0.2*(b-1.01))}\right] * \frac{SCED\%}{100}$$

A Constante, provisoriamente ajustada em 3.0

SCED% Porcentagem de compreensão/expansão do multiplicador de esforço SCED

PM Estimativa de Pessoas/mês

b Fator escalar

TDEV Tempo para desenvolvimento

Capítulo 5 COCOMO II – POST-ARCHITECTURE

5.1 APLICAÇÃO DO MODELO

Quando um projeto se apresenta pronto para ser desenvolvido, já deve possuir uma arquitetura de ciclo de vida que possa fornecer informações mais precisas para as entradas dos direcionadores de custo, proporcionando estimativas mais precisas.

Para apoiar esse estágio, o COCOMO II projetou o modelo *Post-Architecture*, um modelo comprometido com a forma de desenvolvimento de software atual e também de manutenção de produtos de software. Esse modelo prevê a utilização de linhas de código e/ou pontos de função para estimar o tamanho inicial do projeto, reuso de software, 17 direcionadores de custo e 5 fatores de escala de projeto.

Para estimar pontos de função para o modelo *Post-Architecture*, utiliza-se o processo de conversão de pontos de função não ajustados em linhas de código sugerido pelo modelo *Early Design* (Capítulo 4). O COCOMO II permite que componentes possam ter seus tamanhos estimados em pontos de função e outros, onde pontos de função não podem ser bem descritos, tais como aplicações em tempo real e computação científica, em linhas de código.

5.2 REGRAS DE CONTAGEM DE LINHAS DE CÓDIGO

O objetivo da contagem de linhas de código é medir a quantidade de trabalho intelectual necessário para desenvolver um programa. Uma das dificuldades de contagem aparece quando se tenta definir medições consistentes em diferentes linguagens. Definir o que é uma linha de código é difícil devido às diferenças conceituais envolvidas no processo de contagem de declarações executáveis e declarações de dados em diferentes linguagens.

O modelo COCOMO II sugere o uso de um *checklist* desenvolvido pela SEI para apoiar as medições. O *checklist* completo esta disponibilizado no manual do COCOMO II [COCOMOII, 2000].

Para assegurar a uniformidade de dados coletados no projeto COCOMO II, uma ferramenta de coleta de métricas automatizada chamada *Amadeus* foi utilizada [COCOMO II, 2000].

5.3 DIRECIONADORES DE CUSTO

O modelo *Post-Architecture* utiliza 17 direcionadores de custo para ajuste do esforço nominal (Tabela 25), agrupados em 4 categorias: *Produto*, *Plataforma*, *Pessoal* e *Projeto*.

Tabela 25 Post Achitecture – Direcionadores de Custo [COCOMOII, 2000]

Fatores de Produto						
	Muito Baixo	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
RELY	levemente inconveniente	perdas facilmente recuperáveis	moderado, perdas facilmente recuperáveis	alta perda financeira	risco de vidas humanas	
DATA		Bytes da DB/Pgm LOC < 10	10 D/P<100	100 D/P < 100	D/P 1000	
RUSE		Nenhum	Projeto	Programa	Linha de Produto	Múltiplas Linhas de Produto
DOCU	Muita necessidade ao projeto	Pouca necessidade ao projeto	Bem adequado às necessidades do projeto	Excessiva para as necessidades do projeto	Muito excessiva para as necessidades do projeto	
Fatores de Plataforma						
	Muito Baixo	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
TIME			50% de uso do tempo de execução disponível	70%	85%	95%
STOR			50% de uso do armazenamento disponível	70%	85%	95%
PVOL		maior mudança a cada 12 meses; menor mudanças a cada mês	maior: 6 meses menor: 2 semanas	maior: 2 meses menor: 1 semana	maior: 2 semanas menor: 2 dias	
Fatores de Pessoal						
	Muito Baixo	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
ACAP	15 pontos percentuais	35	55	75	90	
PCAP	15 pontos percentuais	35	55	75	90	
PCON	48% ao ano	24% ao ano	12% ao ano	6% ao ano	3% ao ano	
AEXP	2 meses	6 meses	1 ano	3 anos	6 anos	
PEXP	2 meses	6 meses	1 ano	3 anos	6 anos	
LTEX	2 meses	6 meses	1 ano	3 anos	6 anos	

Tabela 25 (Cont.) Resumo dos Direcionadores de Custo do modelo Post Achitecture [COCOMOII, 2000]

Fatores de Projeto						
	Muito Baixo	*Baixo*	*Nominal*	*Alto*	*Muito Alto*	*Extra Alto*
TOOL	edição, código, *debug*	simples, frontend, backend CASE, pouca integração	ferramentas básicas depojeto, integração moderada	ferramentas fortes e maduras de projeto, integração moderada	ferramentas fortes e maduras de projeto proativas, bem integradas com processos, métodos e reuso	
SITE: Combinação	internacional	entre cidades e multi-companhia	Muiti-cidade ou multi-companhia	mesma cidade ou area metropolitana	Mesmo edifício ou instalação	plenamente combinado
SITE: Comunicações	telefone, correio	telefone individual, FAX	email, banda curta	comunicação de banda larga	comunicação de banda larga, algum uso de video conferência	multimídia interativa
SCED	75% do nominal	85%	100%	130%	160%	

A categoria de fatores *Plataforma* diz respeito à complexidade de hardware e de software de infra-estrutura (anteriormente conhecida no COCOMO original como máquina virtual).

Os valores para cálculo dos fatores do COCOMO II são o resultado da calibragem em 83 projetos e estão apresentados na Tabela 26.

Tabela 26 Direcionadores de Custo Calibrados [COCOMOII, 2000]

Direcionador de Custo	***Classificação***					
	Muito Baixo	***Baixo***	***Nominal***	***Alto***	***Muito Alto***	***Extra Alto***
RELY	0,75	0,88	1	1,15	1,39	
DATA		0,93	1	1,09	1,19	
CPLX	0,75	0,88	1	1,15	1,3	1,66
RUSE		0,91	1	1,14	1,29	1,49
DOCU	0,89	0,95	1	1,06	1,13	
TIME			1	1,11	1,31	1,67
STOR			1	1,06	1,21	1,57
PVOL		0,87	1	1,15	1,3	
ACAP	1,5	1,22	1	0,83	0,67	
PCAP	1,37	1,16	1	0,87	0,74	
PCON	1,24	1,1	1	0,92	0,84	
AEXP	1,22	1,1	1	0,89	0,81	
PEXP	1,25	1,12	1	0,88	0,81	
LTEX	1,22	1,1	1	0,91	0,84	
TOOL	1,24	1,12	1	0,86	0,72	
SITE	1,25	1,1	1	0,92	0,84	0,78
SCED	1,29	1,1	1	1	1	

Este anexo apresenta os 17 direcionadores de custo do modelo *Post-Architecture*, utilizados para ajustar o esforço nominal. Esses direcionadores são agrupados em 4 categorias: *Produto*, *Plataforma*, *Pessoal* e *Projeto*. A Tabela F.1 apresenta os critérios de classificação de cada um dos fatores.

5.4 CATEGORIA DE FATORES PRODUTO

Categoria *Produto* é composta por cinco fatores: RELY, DATA, RUSE, e DOCU.

5.4.1 Confiabilidade Requerida do Software (RELY)

O fator *Confiabilidade Requerida do Software* é a medida da amplitude em dado período de tempo, que o software deve executar sua devida função. Se o efeito de uma falha do software for apenas levemente inconveniente, o fator RELY é considerado *Baixo*. Por outro lado, se uma falha da aplicação envolver risco de perda de vidas humanas, o fator RELY deve ser considerado *Muito Alto*.

5.4.2 Tamanho da Base de Dados (DATA)

O fator *Tamanho da Base de Dados* procura captar o efeito dos requisitos de dados de grande porte no desenvolvimento do produto. A classificação é determinada pela Equação F.1. A razão pela qual o tamanho da base de dados é algo importante para ser considerado é devido ao esforço necessário para gerar dados de teste para execução do programa. DATA é classificado como *Baixo* se D/P for menor que 10 e *Muito Alto* se for maior que 1000.

$$\frac{D}{P} = \frac{TamanhodaBasedeDados}{TamanhodoSoftware(I} \qquad \text{(F.1)}$$

5.4.3 Complexidade do Produto (CPLX)

O fator Complexidade do Produto é uma nova escala de classificação do COCOMO II. A complexidade é dividida em 5 áreas: operações de controle, operações computacionais, operações dependentes de algum tipo de máquina, operações de gerenciamento de dados e operações de gerenciamento de interface de

usuário. Deve-se selecionar a área ou combinação de áreas que caracterizam o produto ou subsistema do produto. A classificação de complexidade é a média dessas áreas julgada subjetivamente.

5.4.4 Reusabilidade Requerida (RUSE)

O fator *Reusabilidade Requerida* procura medir a quantidade de esforço adicional necessário para construir componentes destinados a futuro reuso em outros projetos. Esse esforço é despendido na criação de projetos de software mais genéricos, documentação mais elaborada e realização de extensivos testes para assegurar que esses componentes estejam adequados para o uso em outras aplicações.

5.4.5 Documentação Adequada às Necessidades do Projeto (DOCU)

Muitos modelos de custo de software possuem um direcionador de custo para o nível de documentação requerida do projeto.

No COCOMO II, a escala de classificação para o fator DOCU é avaliada em termos de adequação da documentação do projeto com os requisitos do ciclo de vida. A escala de classificação varia de *Muito Baixo* até *Muito Alto*.

5.5 A CATEGORIA DE FATORES PLATAFORMA

A categoria de fatores *Plataforma* diz respeito à complexidade de hardware e de software de infra-estrutura (anteriormente conhecida no COCOMO original como máquina virtual).

5.5.1 Restrição de Tempo de Execução (TIME)

O fator *Restrição de Tempo de Execução* é uma medida da restrição do tempo de execução imposto em um sistema de software. A escala é expressa em termos de porcentagem de tempo de execução disponível que deve ser utilizado pelo sistema ou subsistema. A escala de classificação varia de *Nominal*, menos de 50% de recursos de tempo de execução utilizado, até *Extra Alto*, 95% dos recursos de execução de tempo é consumido.

5.5.2 Restrição de Armazenamento Principal (STOR)

O fator *Restrição de Armazenamento Principal* representa o grau de restrição de armazenamento principal imposto ao sistema ou subsistema. Dado um notável aumento do tempo de execução disponível do processador e armazenamento principal, pode-se questionar se essas variáveis de restrição ainda são importantes. A escala de classificação varia de *Nominal*, menos que 50%, até *Extra Alto*, 95%.

5.5.3 Volatilidade da Plataforma (PVOL)

O termo plataforma é uma referência à complexidade de hardware e software (Sistema Operacional, Sistema Gerenciador de Banco de Dados, etc.) que a aplicação necessita para executar suas tarefas.

Se o software a ser desenvolvido é um sistema operacional, a plataforma é o hardware do computador. Se o software a ser desenvolvido é um sistema gerenciador de banco de dados, a plataforma é o hardware do computador e o sistema operacional. Se o software a ser desenvolvido é um *browser* de texto para redes, então a plataforma é a rede, o hardware do computador e os repositórios de informação distribuídos.

A escala de classificação desse fator varia de *Baixa*, onde existe uma grande mudança a cada 12 meses, até *Muito Alta*, onde há grande mudança a cada duas semanas.

5.6 A CATEGORIA DE FATORES PESSOAL

A categoria *Pessoal* é composta por 6 fatores: ACAP, PCAP, PCON, AEXP, PEXP e LTEX.

5.6.1 Capacidade do Analista (ACAP)

O fator *Capacidade do Analista* é o primeiro dos fatores agrupados na categoria Pessoal. O analista é a pessoa que trabalha com requisitos de alto nível e/ou com requisitos detalhados. Os principais atributos que devem ser considerados são: a habilidade de análise e projeto, a eficiência e competência, e a habilidade de comunicação e cooperação.

Não deve ser levado em consideração o nível de experiência do analista, pois esse atributo é considerado pelo fator AEXP. A escala de classificação varia de *Muito Baixo*, 15%, até *Muito Alto*, 95%.

5.6.2 Capacidade dos Programadores (PCAP)

A avaliação deve ser baseada na capacidade dos programadores de trabalhar em equipe, não individualmente. Os principais fatores que devem ser considerados na classificação da habilidade são: habilidade; eficiência e competência; habilidade de comunicação e cooperação. A experiência individual do programador não deve ser considerada nesse fator; ela é classificada no fator AEXP.

5.6.3 Rotatividade de Pessoal (PCON)

A escala de classificação do fator PCON é graduada em termos de rotatividade anual de pessoal.

5.6.4 Experiência com a Aplicação (AEXP)

Essa classificação está relacionada com o grau de experiência da equipe de desenvolvimento no sistema ou subsistema que será desenvolvido. As classificação estão definidas em termos de nível de experiência da equipe com o tipo de aplicação. O menor nível de classificação é 2 meses para pouca experiência 6 anos ou mais para o grau Muito Alto.

5.6.5 Experiência com a Plataforma de Desenvolvimento (PEXP)

O modelo *Post-Architecture* considera a influência do fator PEXP no desenvolvimento de sistemas. Esse fator procura determinar o grau de compreensão de plataformas de desenvolvimento mais poderosas, incluindo ambientes *Graphic User Interface*, bancos de dados, redes de computadores, e capacidades de *middleware* distribuídas..

5.6.6 Experiência com Linguagens e Ferramentas de Desenvolvimento (LTEX)

Esse fator procura medir o grau de experiência da equipe com o uso da linguagem e de ferramentas de desenvolvimento para o projeto do sistema ou subsistema. O desenvolvimento de software pode incluir o uso de ferramentas nas tarefas de análise de requisitos, gerenciamento de configuração, preparação de documentação, etc..

O menor grau de classificação para esse fator é menos que 2 meses e o maior é 6 anos ou mais (Muito Alto).

5.7 CATEGORIA DE FATORES PROJETO

A categoria *Projeto* é composta pelos fatores: TOOL, SITE e SCED.

5.7.1 Uso de Ferramentas de Software (TOOL)

O grau de classificação para as ferramentas de desenvolvimento variam de Muito Baixo, para ferramentas que ofereçam apenas a edição e codificação, até Muito Alto, para ferramentas de gerenciamento integradas no ciclo de vida.

5.7.2 Desenvolvimento Distribuído ou Multisite (SITE)

Esse fator foi incluído no COCOMO II por causa do considerável aumento na freqüência de desenvolvimento em ambientes fisicamente distribuídos e nas indicações de que seus efeitos no desenvolvimento global do sistema são significantes.. Para a classificação desse direcionador de custo, deve-se avaliar dois fatores: distribuição geográfica e comunicações de apoio (uso de aplicativos de correio eletrônico a comunicações multimídia).

5.7.3 Cronograma de Desenvolvimento Requerido (SCED)

Essa classificação procura avaliar restrições de cronograma impostas à equipe de desenvolvimento. Os graus de classificação são definidos em termos de porcentagem do tempo de desenvolvimento em relação ao valor nominal estimado para o projeto. Cronogramas acelerados tendem a necessitar de maior esforço nas fases finais de desenvolvimento.

Um cronograma restrito a 74% do tempo de desenvolvimento nominal é classificado Muito Baixo. Um cronograma A extenso necessita de mais esforço nas fases iniciais de desenvolvimento, pois existe mais tempo para planejamento, especificação e validação. Um cronograma previsto para 160% do valor nominal do tempo de desenvolvimento do projeto extenso é classificado como Muito Alto.

5.7.4 Confiabilidade Requerida do Software (RELY)

O fator *Confiabilidade Requerida do Software* é a medida da amplitude em dado período de tempo, que o software deve executar sua devida função.

Se o efeito de uma falha do software for apenas levemente inconveniente, o fator RELY é considerado *Baixo*. Por outro lado, se uma falha da aplicação envolver risco de perda de vidas humanas, o fator RELY deve ser considerado *Muito Alto*.

5.7.5 Tamanho da Base de Dados (DATA)

O fator *Tamanho da Base de Dados* procura captar o efeito dos requisitos de dados de grande porte no desenvolvimento do produto. A classificação é determinada pela Equação 17. A razão pela qual o tamanho da base de dados é algo importante para ser considerado é devido ao esforço necessário para gerar dados de teste para execução do programa. DATA é classificado como *Baixo* se D/P for menor que 10 e *Muito Alto* se for maior que 1000.

Equação 17 Tamanho da Base de Dados

$$\frac{D}{P} = \frac{TamanhodaBasedeDados\,(Bytes)}{TamanhodoSoftware\,(LOC)}$$

5.7.6 Complexidade do Produto

O fator *Complexidade do Produto* é uma nova escala de classificação do COCOMO II. A complexidade é dividida em 5 áreas: *operações de controle*, *operações computacionais*, *operações dependentes de algum tipo de máquina*, *operações de gerenciamento de dados* e *operações de gerenciamento de interface de usuário*. Deve-se selecionar a área ou combinação de áreas que caracterizam o produto ou subsistema do produto. A classificação de complexidade é a média dessas áreas julgada subjetivamente.

5.7.7 Reusabilidade Requerida (RUSE)

O fator *Reusabilidade Requerida* procura medir a quantidade de esforço adicional necessário para construir componentes destinados a futuro reuso em outros projetos.

Esse esforço é despendido na criação de projetos de software mais genéricos, documentação mais elaborada e realização de extensivos testes para assegurar que esses componentes estejam adequados para o uso em outras aplicações.

5.7.8 Documentação Adequada às Necessidades do Projeto (DOCU)

Muitos modelos de custo de software possuem um direcionador de custo para o nível de documentação requerida do projeto. No COCOMO II, a escala de classificação para o fator DOCU é avaliada em termos de adequação da documentação do projeto com os requisitos do ciclo de vida. A escala de classificação varia de *Muito Baixo* até *Muito Alto*.

5.7.9 Restrição de Tempo de Execução (TIME)

O fator *Restrição de Tempo de Execução* é uma medida da restrição do tempo de execução imposto em um sistema de software. A escala é expressa em termos de porcentagem de tempo de execução disponível que deve ser utilizado pelo sistema ou subsistema. A escala de classificação varia de *Nominal*, menos de 50% de recursos de tempo de execução utilizado, até *Extra Alto*, 95% dos recursos de execução de tempo é consumido.

5.7.10 Restrição de Armazenamento Principal (STOR)

O fator *Restrição de Armazenamento Principal* representa o grau de restrição de armazenamento principal imposto ao sistema ou subsistema. Dado um notável aumento do tempo de execução disponível do processador e armazenamento principal, pode-se questionar se essas variáveis de restrição ainda são importantes. A escala de classificação varia de *Nominal*, menos que 50%, até *Extra Alto*, 95%.

5.7.11 Volatilidade da Plataforma (PVOL)

O termo plataforma é uma referência à complexidade de hardware e software (Sistema Operacional, Sistema Gerenciador de Banco de Dados, etc.) que a aplicação necessita para executar suas tarefas.

Se o software a ser desenvolvido é um sistema operacional, a plataforma é o hardware do computador. Se o software a ser desenvolvido é um sistema gerenciador de banco de dados, a plataforma é o hardware do computador e o sistema operacional. Se o software a ser desenvolvido é um *browser* de texto para redes, então a plataforma é a rede, o hardware do computador e os repositórios de informação distribuídos.

O fator *Plataforma* também se refere a qualquer compilador ou a*ssembler* que apóia o desenvolvimento da aplicação. A escala de classificação desse fator varia de *Baixa*, onde existe uma grande mudança a cada 12 meses, até *Muito Alta*, onde há grande mudança a cada duas semanas.

5.7.12 Capacidade do Analista (ACAP)

O fator *Capacidade do Analista* é o primeiro dos fatores agrupados na categoria Pessoal. O analista é a pessoa que trabalha com requisitos de alto nível e/ou com requisitos detalhados. Os principais atributos que devem ser considerados são: a habilidade de análise e projeto, a eficiência e completitude, e a habilidade de comunicação e cooperação. Não deve ser levado em consideração o nível de experiência do analista, pois esse atributo é considerado pelo fator AEXP. A escala de classificação varia de *Muito Baixo*, 15%, até *Muito Alto*, 95%.

5.8 EQUAÇÕES DO MODELO *POST-ARCHITECTURE*

O esforço de desenvolvimento, segundo o modelo *Post-Architecture* é estimado com a Equação 18, diferenciando da equação do modelo *Early Design* no número de direcionadores de custo, que agora passa a ser 18.

Equação 18 Modelo Post Architecture – Estimativa de Esforço

$$PM = a * \left[Tamanho'\right]^{b} * \prod_{i=1}^{17} EM_{i} + PM_{M}$$

As demais equações utilizadas para calcular o fator escalar b, o ajuste de tamanho, porcentagem de reuso de código, etc. são iguais às equações do modelo *Early Design* (Equações 10, 11, 12, 13, 14 e 15).

Capítulo 6 FERRAMENTAS AUTOMATIZADAS

6.1 FERRAMENTAS QUE IMPLEMENTAM COCOMO

Existem várias ferramentas de estimativa automatizadas que implementam os modelos COCOMO e COCOMO II, disponíveis em versões comerciais e versões de domínio público. Ferramentas de estimativa representam um bom método para checagem com estimativas derivadas manualmente [SOFTWARE, 1996].

Os exemplos de ferramentas de estimativa de software apresentados neste relatório são: *REVIC*, *COSMOS*, *COSTAR*, *USC COCOMO 2000.0*, *CONSTRUX e Cost Xpert*..

6.1.1 Revic

REVIC – REVised Intermediate COCOMO é uma ferramenta de estimativa de domínio público disponibilizada na Internet [http://sepo.nosc.mil/revic.zip]. Trata-se de uma implementação derivada do modelo COCOMO original utilizada extensivamente pela comunidade de software do *DoD* – Departamento de Defesa dos Estados Unidos. A ferramenta também foi adotada como modelo de custo de software oficial dos desenvolvedores de software da Força Aérea americana.

Figura 3 REVIC – Tela de abertura da ferramenta

A ferramenta foi desenvolvida no ambiente DOS (Figura 3) e é relativamente fácil de ser utilizada, operável por menus apresentados ao usuário. Vários parâmetros podem ser modificados pelo usuário para simular o ambiente de desenvolvimento mais próximo da realidade da organização.

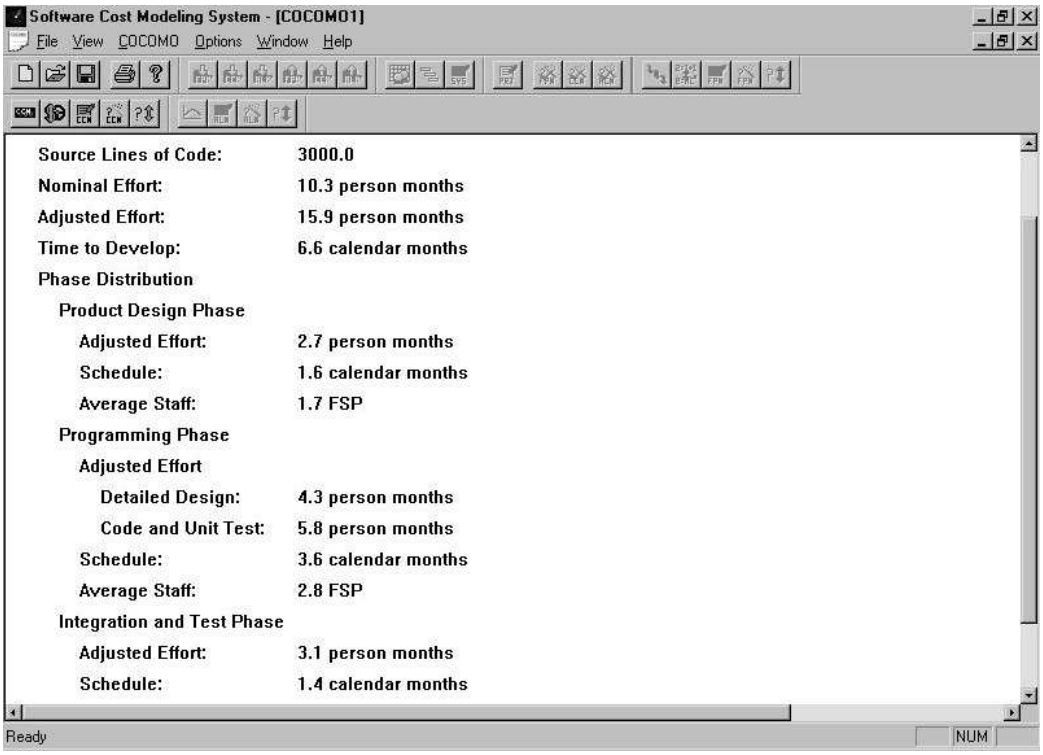
Para ilustrar a aplicação das ferramentas COSMOS, COSTAR, USC COCOMO 2000.0, CONSTRUX e Cost Xpert, foram utilizados os dados do exemplo do Capítulo 1, item 1.2.3 (3.0 Kloc, CPLX alto, STOR alto, AEXP baixo e ACAP baixo).

6.1.2 COSMOS

COSMOS – Software Cost Modeling System Version 4.1 é uma implementação *freeware* da *East Tennessee State University*, disponível na Internet [http://csvaxsrv.east-tenn-st.edu/~dsgnstudio/cosmos/download.html].

A ferramenta combina os elementos do modelo *COCOMO original* e da *Análise de Pontos de Função* para levar em conta a funcionalidade do elemento humano e dos fatores de ambiente de desenvolvimento que devem ser considerados na construção de um modelo de estimativa. O tamanho estimado do projeto pode estimado em linhas de código ou em pontos de função.

Figura 4 COSMOS – Tela de estimativa e distribuição por fases

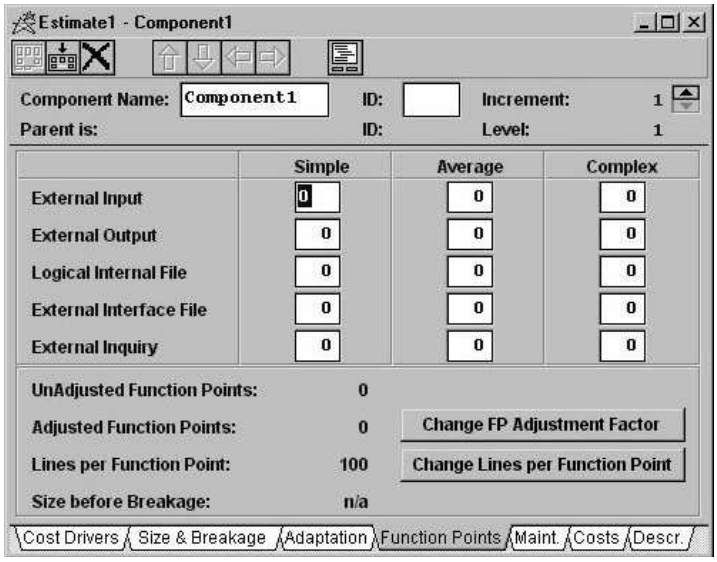


6.1.3 COSTAR

COSTAR é uma ferramenta comercial baseada no modelo COCOMO para calculo de estimativa de esforço, custo, prazo e distribuição de equipe. A versão demonstrativa pode ser encontrada na Internet [http://www.softstarsystems.com/costar5d.zip].

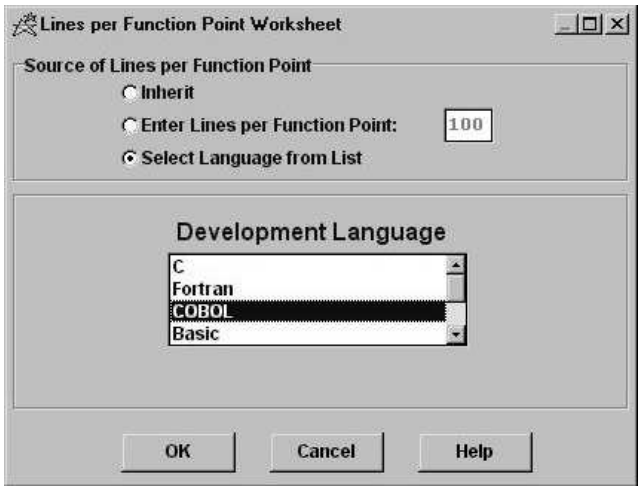
A ferramenta implementa os modelos *COCOMO original*, *Ada COCOMO* e *COCOMO II* . O tamanho do sistema pode ser expresso em de pontos de função ou em linhas de código. Os pontos de função contados são inseridos na ferramenta na tela apresentada na Figura 5.

Figura 5 COSTAR – Tela de entrada de pontos de função



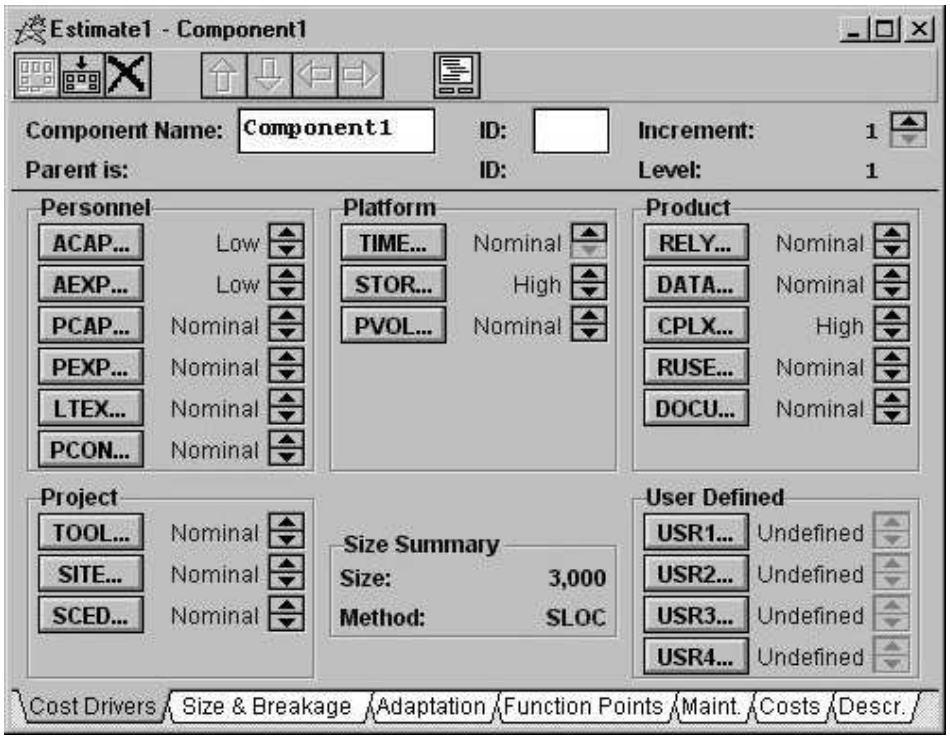
Após a inserção dos dados de pontos de função, deve-se informar a linguagem escolhida para implementação do sistema (Figura 6).

Figura 6 COSTAR – Seleção da Linguagem de desenvolvimento



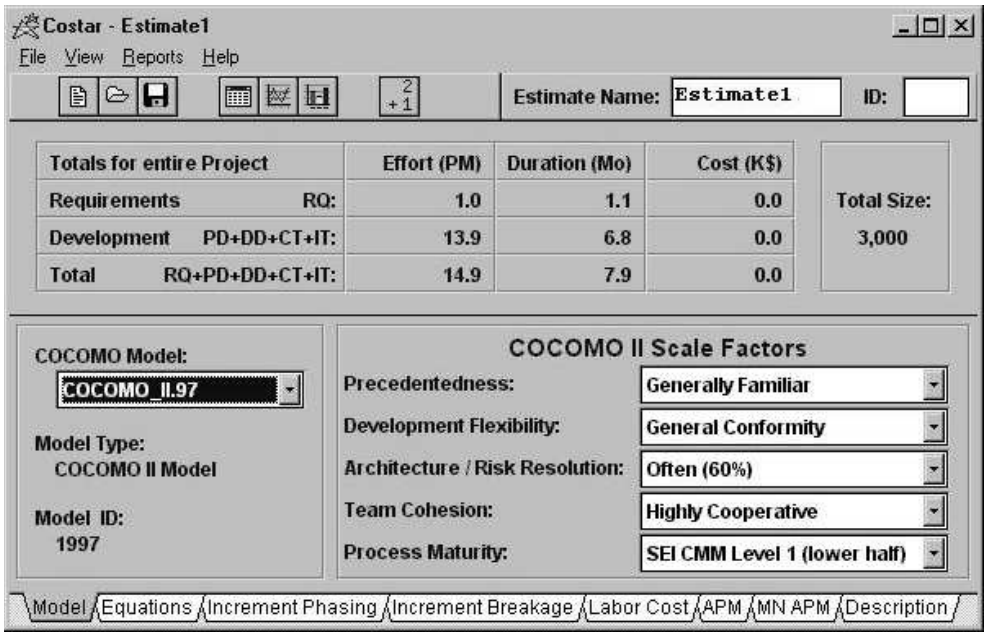
Os direcionadores de custo do modelo COCOMO II devem ser informados na tela apresentada na Figura 7.

Figura 7 COSTAR – Tela de ajuste dos direcionadores de custo



A Figura 8 apresenta o menu principal da ferramenta. Nessa tela deve-se selecionar o modelo implementado e, no caso de escolha do COCOMO II, informa-se os fatores de escala. O resultado global do cálculo das estimativas para o projeto é mostrado nessa tela.

Figura 8 COSTAR – Menu principal



A ferramenta COSTAR™ gera uma série de relatórios para estudo das estimativas calculadas. A Figura 9 apresenta o relatório em detalhes da distribuição do esforço em fases e sua duração.

Figura 9 COSTAR – Relatório de Estimativa por Fase

Estimate1 - Detail Report

Print | Save As Text | Save as Graphic | ☑ Headers | << Back | Next >>

Estimate1 - Detail Report

Costar Demo 13/04/00 17:24:07 Page: 1

Estimate Name: Estimate1 Estimate ID:
Model Name: COCOMO_II.97 Model ID: 1997 Process Model: COCOMO II

Component Name: Component1 Component ID:
Component Size: 3,000 Level: 1

Phase	Effort (Person-Months)	Cost (K$)	Duration (Months)	Staffing
RQ -- Requirements	1.0	0.0	1.1	0.9
PD -- Product Design	2.4	0.0	1.6	1.4
DD -- Detailed Design	3.7	0.0	1.6	2.3
CT -- Code & Unit Test	5.1	0.0	2.1	2.4
IT -- Integration & Test	2.7	0.0	1.4	2.0
Subtotals (PD+DD+CT+IT)	13.9	0.0	6.8	
Totals (RQ+PD+DD+CT+IT)	14.9	0.0	7.9	
MN -- Maintenance (per year)	0.0	0.0		0.0

As ferramentas que implementam o modelo COCOMO levam em consideração as atividades de *Gerenciamento de Configuração* e *Garantia de Qualidade de Software* (CM/QA) no distribuição de esforço e cálculo de custo do projeto(Figura 10).

Figura 10 COSTAR – Relatório de Estimativa por Atividade

Estimate1 - Activity Report

Print | Save As Text | Save as Graphic | ☑ Headers | << Back | Next >>

Estimate1 - Activity Report

Costar Demo 18/04/00 15:52:47 Page: 1

Estimate Name: Estimate1 Estimate ID:
Model Name: COCOMO_II.97 Model ID: 1997 Process Model: COCOMO II

	Effort in Person-Months						
Activity	RQ	PD	DD	CT	IT	Total RQ to IT	MN
Requirements	0.5	0.3	0.1	0.2	0.1	1.2	0.0
Product Design	0.2	1.0	0.3	0.4	0.1	2.0	0.0
Programming	0.0	0.3	2.1	2.9	0.9	6.2	0.0
Test Plans	0.0	0.1	0.2	0.2	0.1	0.6	0.0
V & V	0.1	0.1	0.3	0.4	0.9	1.7	0.0
Project Office	0.1	0.3	0.3	0.4	0.2	1.3	0.0
CM/QA	0.0	0.1	0.3	0.4	0.2	0.9	0.0
Manuals	0.1	0.2	0.2	0.3	0.2	1.0	0.0
Totals	1.0	2.4	3.7	5.1	2.7	14.9	0.0

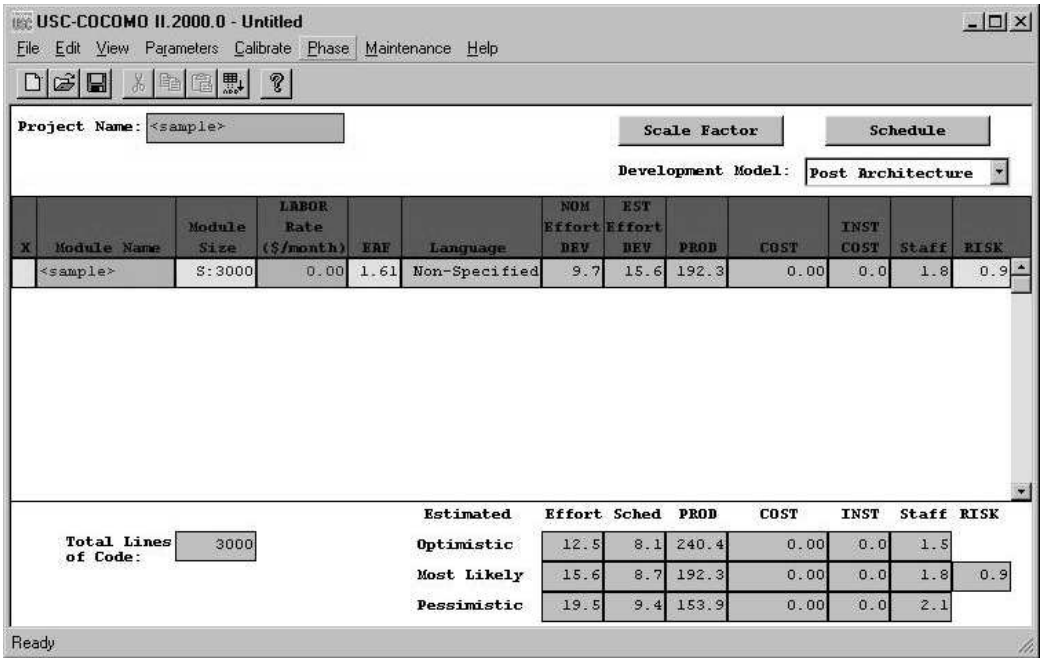
6.1.4 USC COCOMO 2000.0

A ferramenta USC COCOMO 2000.0 é uma implementação do modelo *Post Architecture* do COCOMO II, com características funcionais semelhantes às da ferramenta COSTAR.

Ela calcula esforço, prazo e distribuição do esforço e encontra-se disponível para plataformas Sun Sparc Unix, Microsoft Windows 95/NT, e plataformas que habilitam a linguagem Java. O manual de referência do modelo COCOMO II e da ferramenta de versão pública que a implementa encontram-se disponíveis na Internet [http://sunset.usc.edu/COCOMOII/cocomo.html].

Os parâmetros dos direcionadores de custo podem ser calibrados na ferramenta. Vários tipos de relatórios podem ser gerados pela ferramenta.

Figura 11 USC COCOMO – Menu principal do sistema



6.1.5 Construx Estimate

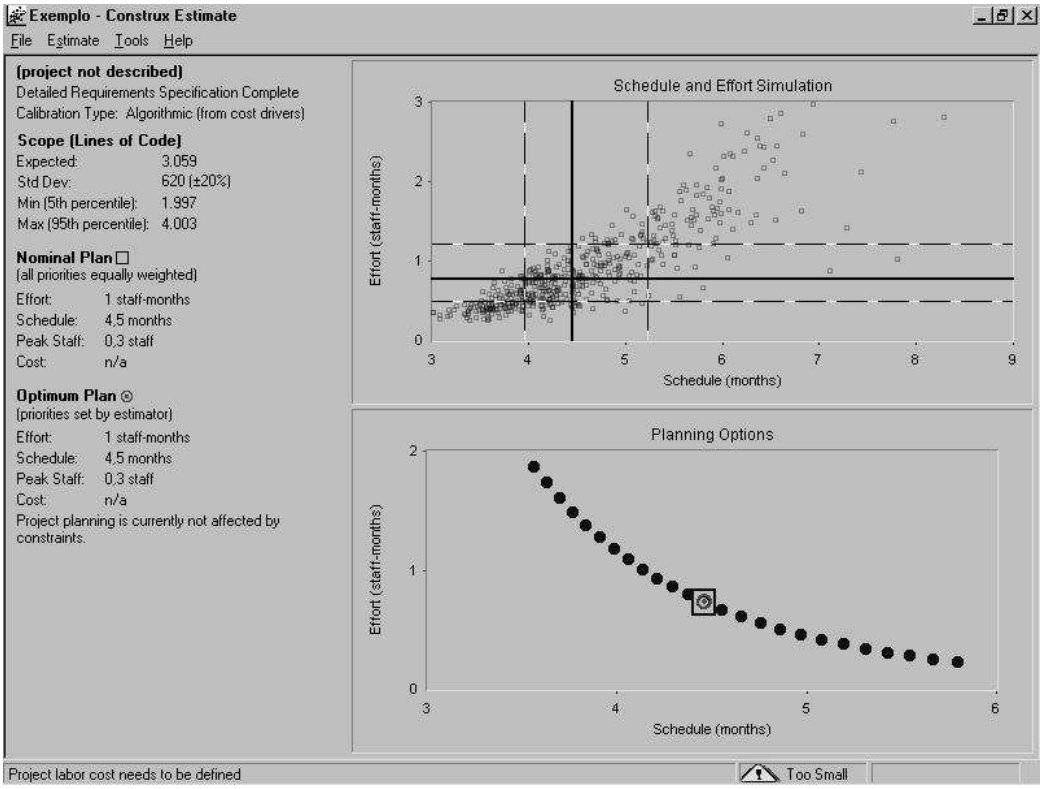
Construx Estimate é uma ferramenta comercial de estimativa cuja versão demonstrativa está disponível na Internet [http://www.construx.com/estimate/download.htm].

A ferramenta combina três modelos de estimativa: *SLIM*, *COCOMO II* e o método de simulação *Monte Carlo*.

O modelo SLIM [Putnam, 1992] foi desenvolvido por Lawrence H. Putnam no início da década de setenta. Ele está baseado no estudo e percepção de que projetos de software bem sucedidos seguem padrões bem definidos que podem ser modelados em um conjunto de equações exponenciais.

A Figura 12 apresenta o menu principal da ferramenta.

Figura 12 CONSTRUX – Menu principal

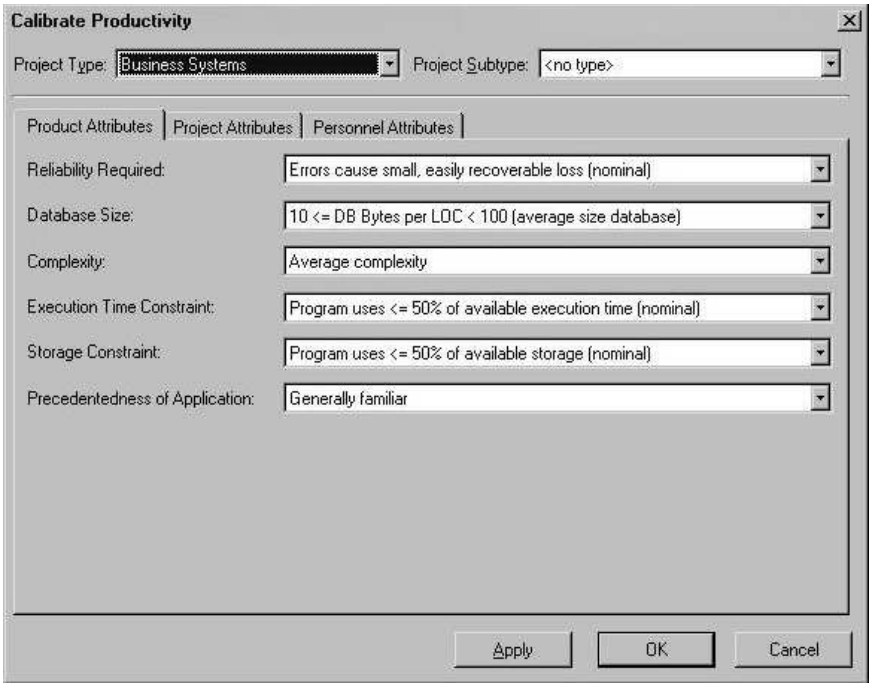


Quando estimativas são calibradas utilizando-se direcionadores de custo, Construx Estimate utiliza o modelo COCOMO II para complementar o modelo SLIM.

A ferramenta utiliza um conjunto provisório de fatores para calibrar a implementação do modelo COCOMO II, visto que o modelo ainda está sendo desenvolvido. Os desenvolvedores dessa ferramenta afirmam que ela será atualizada assim que os fatores de calibragem do modelo COCOMO II forem divulgados.

A Figura 13 apresenta a tela de seleção dos direcionadores de custo para calibragem das estimativas.

Figura 13 CONSTRUX – Ajuste dos direcionadores de custo



Construx Estimate utiliza o modelo de simulação Monte Carlo. Esse modelo simula centenas ou milhares de resultados possíveis para o projeto que está sendo estimado, baseado em dados de tamanho, produtividade, fase atual do projeto e outros parâmetros informados.

O método de Monte Carlo calcula o potencial de cada resultado e projeta níveis de risco para diferentes opções de planejamento.

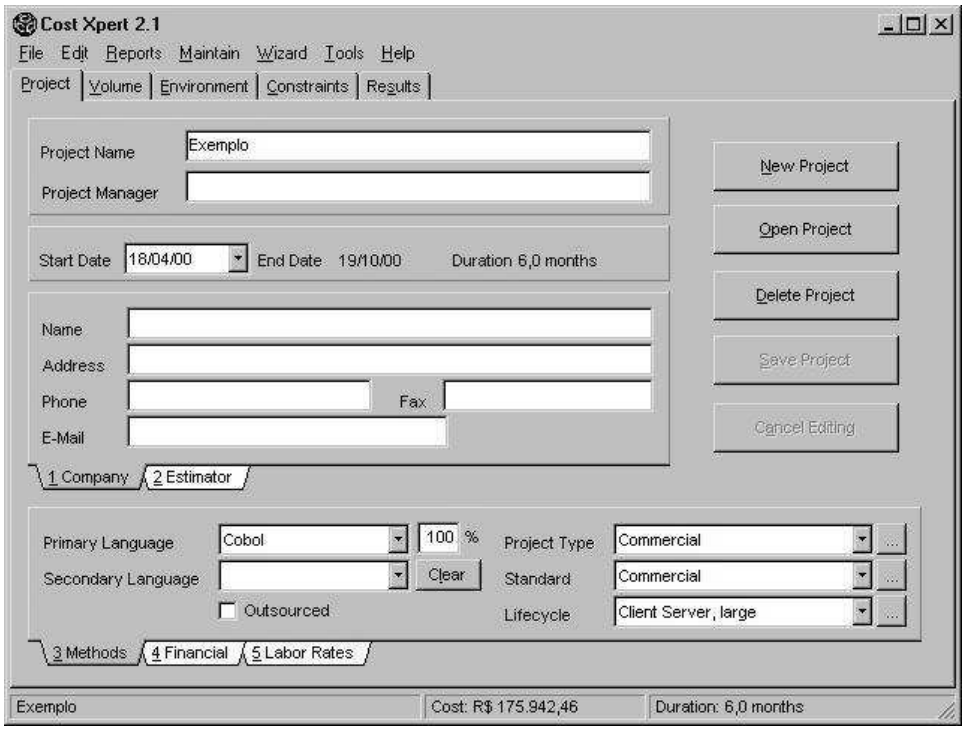
6.1.6 COST XPERT 2.1

Cost Xpert é uma ferramenta comercial que implementa vários métodos de estimativa. A versão demonstrativa da ferramenta encontra-se disponível na Internet [http://www.costxpert.com/userform.htm?cxdownload].

A ferramenta utiliza dados coletados de 7.000 projetos, comerciais e militares, para assegurar maior precisão nas estimativas.

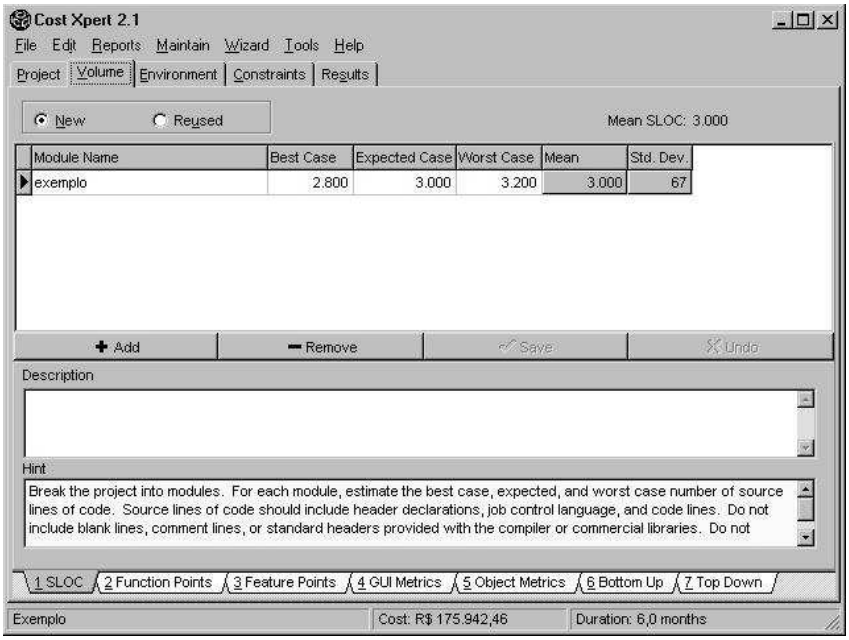
A Figura 14 apresenta o menu principal da ferramenta.

Figura 14 Cost Xpert – Menu Principal



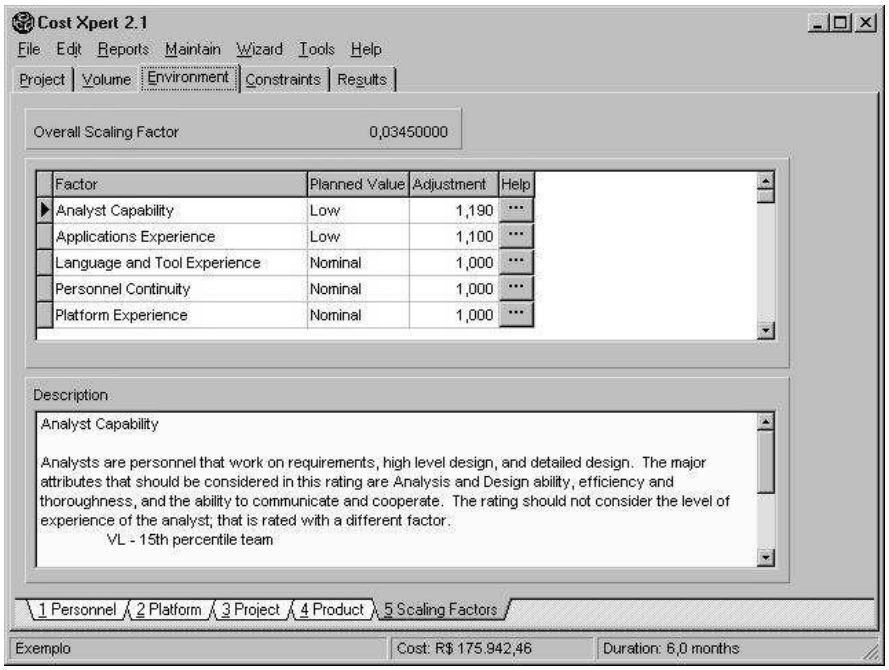
Para estimar o tamanho do projeto, a ferramenta implementa uma variedade de métodos. Pode-se utilizar linhas de código, pontos de função, pontos de característica, métricas GUI, métricas de objeto e as abordagens heurísticas *top down* e *bottom up* (Figura 15).

Figura 15 Cost Xpert – Estimativa de Tamanho



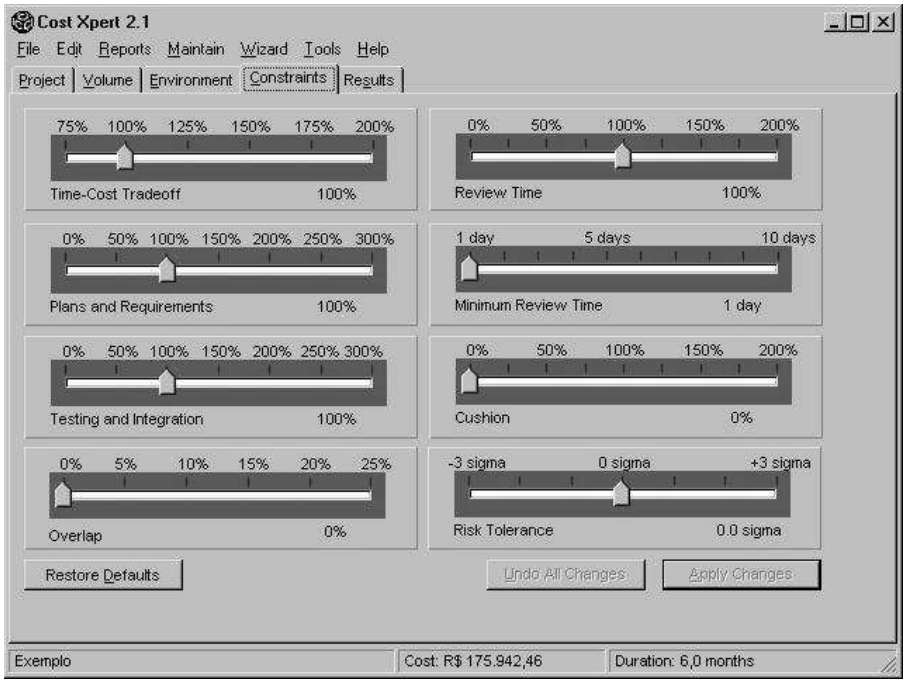
A Figura 16 apresenta a tela de seleção dos parâmetros dos direcionadores de custo do modelo COCOMO II.

Figura 16 Cost Xpert – Direcionadores de Custo



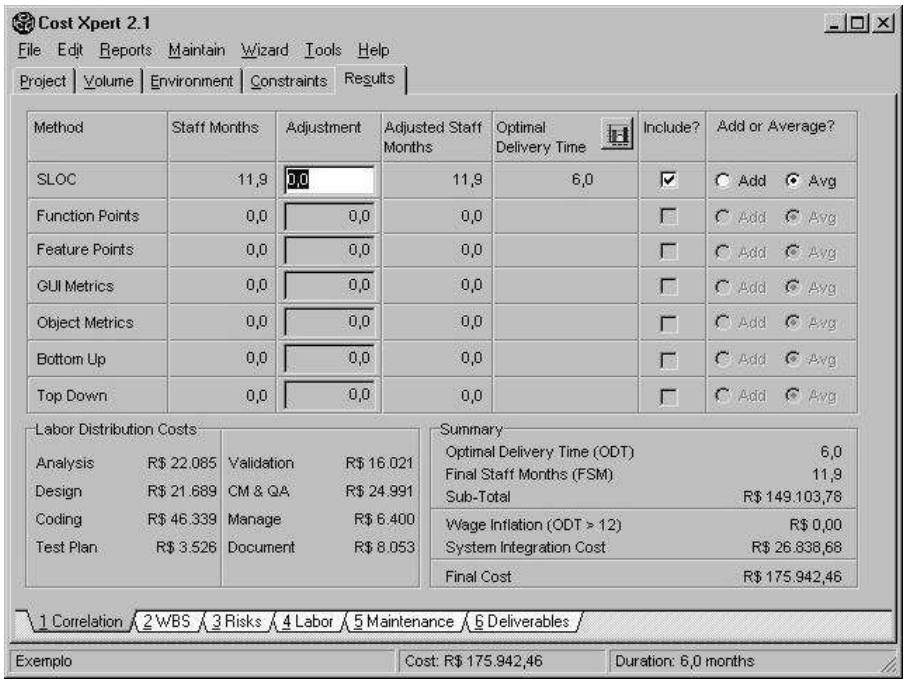
Além do ajuste dos direcionadores de custo do COCOMO II, a ferramenta possibilita um ajuste ainda mais reinado da estimativa, baseado em um conjunto de 8 fatores, apresentados na Figura 17.

Figura 17 Cost Xpert – Outros Fatores Direcionadores de Custo



A Figura 18 apresenta a tela de resultados parciais das estimativas de acordo com o método de estimativa de tamanho utilizado.

Figura 18 Cost Xpert – Estimativas Parciais



Além dos relatórios de divisão de esforço por fases e atividades, a ferramenta gera gráficos da estrutura de divisão funcional do trabalho e gráficos de distribuição do esforço (Figuras 19 e 20).

Figura 19 Cost Xpert – Gráfico de Distribuição

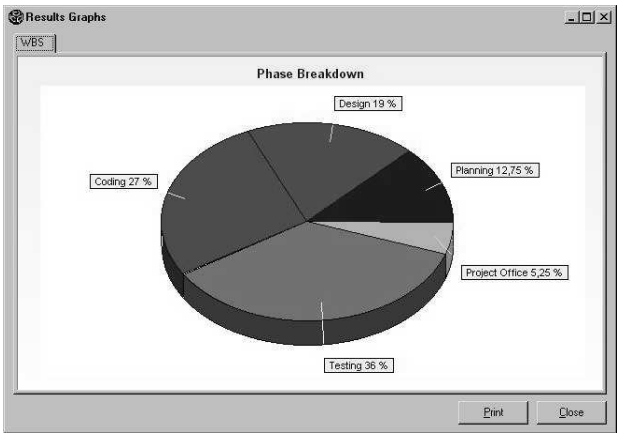
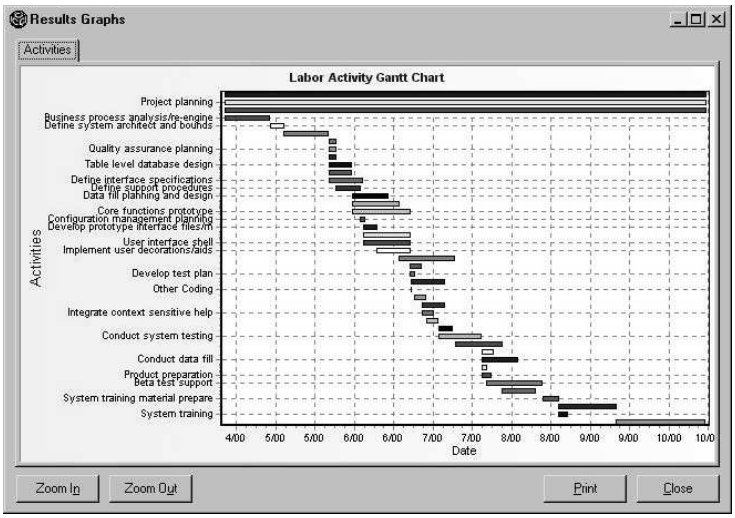


Figura 20 Cost Xpert – Gráfico de Gantt



Capítulo 7 CONCLUSÃO

Estimativas são necessárias para que projetos sejam planejados e conduzidos com sucesso. As empresas desenvolvedoras de software contam com um amplo conjunto de modelos e técnicas para estimar projetos, entre eles o COCOMO – *Constructive Cost Model* – apresentado em 1981 por Boehm. Esse modelo apresenta uma série de equações para calcular estimativas de custo e prazo de projetos, a partir da estimativa do tamanho inicial de um sistema ou dos módulos que o compõem.

Em 1994 teve início o projeto COCOMO II, uma iniciativa da USC – *University of Southern California* – com o objetivo de atualizar o modelo COCOMO original com as práticas de ciclo de vida de software modernas para estimar precisamente ciclos de vida de software espirais ou modelos evolucionários, desenvolvimento de software a partir de componentes de software disponíveis comercialmente e desenvolvimento de software orientado a objeto.

O projeto deu origem a três novos modelos de estimativa baseados em estágios específicos de desenvolvimento de projeto. Cada um deles requer variáveis de entrada específicas para as equações de cálculo das estimativas, de acordo com o nível de detalhes de informações disponíveis no estágio de desenvolvimento cuja estimativa será calculada.

O primeiro modelo, denominado *Application Composition*, é voltado para as fases iniciais de projetos e ciclos de vida espirais que envolvem o uso de prototipagem. Esse modelo utiliza *Pontos de Objeto*, uma medida de tamanho de sistemas de software similar a Pontos de Função.

Early Design é o modelo desenvolvido para estimar as atividades das demais fases de projetos ou ciclos de vida espirais que envolvem pesquisa de arquiteturas alternativas ou estratégias de desenvolvimento incremental. Nos estágios iniciais, quando há pouca informação para estimar o tamanho do sistema, utiliza-se pontos de função não ajustados como variável de entrada para o cálculo.

O modelo *Post-Architecture* foi desenvolvido para ser utilizado quando o projeto já está pronto para ser desenvolvido. Nessa fase já se encontram disponíveis informações suficientes para calcular todos os direcionadores de custo. Esse modelo

utiliza linhas de código ou, opcionalmente, Pontos de Função como variáveis de entrada para as equações de cálculo.

Os modelos do COCOMO II levam em consideração o reuso de componentes de software e reengenharia no projeto.

Existem atualmente várias ferramentas de estimativa automatizadas que implementam os modelos COCOMO e COCOMO II, disponíveis em versões comerciais e de domínio público. Exemplos de ferramentas são: REVIC, COSMOS, COSTAR, USC COCOMO II, CONSTRUX E COST XPERT.

Ferramentas de estimativa possuem as mesmas características gerais e exigem uma ou mais das seguintes categorias de dados: *estimativa quantitativa do tamanho do projeto*; *características qualitativas de projeto*; *descrição do pessoal responsável pelo desenvolvimento*. A partir desses dados as ferramentas podem calcular esforço, tempo total de desenvolvimento, distribuição de esforço em fases e atividades, custos, composição da equipe desenvolvedora e em certos casos, calculam também cronograma por fases e atividades de desenvolvimento e riscos associados ao projeto.

Capítulo 8 BIBLIOGRAFIA

[Albrecht, 1983] ALBRECHT, A. J.; GAFFNEY, J. E. JR. Software Function, Source Lines of Code, and Develpment Effort Prediction In: *IEEE Transactions on Software Engineering*, SE-9, 6, p.639-648

[Albrecht, 1983] ALBRECH, A. J.; GAFFNEY, J. E. Jr. Software Function, Source Lines of Code, and Development Effort Prediction in *IEEE Transactions on Software Engineering*, SE-9, 6, 639-648.

[Boehm, 1981] BOEHM, B. *Software Engineering Economics* Englewood Cliffs N.J.: Prentice-Hall Inc. 1981

[Braga, 1996] BRAGA, A. *Análise de pontos por função*. Rio de Janeiro: Infobook, 1996.

[COCOMOII, 2000] COCOMO II [on-line] [05/02/2000] Disponível na Internet: <http://sunset.usc.edu/index.html>

[Conte, 1986] CONTE, S.D.; SHEN V.Y. *Software Engineering Metrics and Models.* Menlo Park, California: Benjamin/Cummings Publishing, 1985.

[Dekkers, 1998] DEKKERS, C. A. *Managing (the Size of) Your Projects*. A Project Management Look at Function Points. Quality Plus Technologies, Inc. [on-line] [05/02/2000] Disponível na Internet: <www.bfgup.com.br>

[Fernandes, 1995] FERNANDES, A. A. *Gerência de Software através de métricas: garantindo a qualidade do projeto, processo e produto*. São Paulo: Atlas, 1995.

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

[IFPUG, 1994] Function Points Counting Practice Manual. Release 4.0. Westerville, Ohio, 1994 [on-line] [05/02/2000] Disponível na Internet: <www.ifpug.org >

[IFPUG, 2000] International Function Points Users Group [on-line] [05/02/2000] Disponível na Internet: <www.ifpug.org>

[Jones, 1991] JONES, C. *Applied Software Measurement: assuring productivy and quality*. New York: McGraw-Hill, 1996.

[Paulk, et. al., 1993] PAULK, M. C.;CURTIS, B.;CHRISSIS, M. B.; WEBER, C. V., *Capability Maturity Model for Software, version 1.1,* CMU/SEI-93-TR-24, fevereiro, 1993.

[Pressman, 1997] PRESSMAN, R. *Software Engineering: A Practitioner's Approach*. 4th. Ed. New York: McGraw-Hill, 1997.

[Putnam, 1992] PUTNAM, L. H. and MYERS, W. *Measures for Excellence: Reliable Software on Time, Within Budget*. Yourdon Press, Englewood Cliffs N., J.: 1992.

[Softstar, 2000] Softstar Systems [03/02/99/]. Disponível na Internet: < http://www.softstarsystems.com/advanced.htm#Ada >

[Vigder, 1994] VIGDER, M. R.; KARK, A. W. *Software Cost Estimation and Control* [on-line]. National Research Council Canada, Institute for Information Technology, 1994. [03/02/99/]. Disponível na Internet: <http://wwwsel.iit.nrc.ca/abstracts/NRC37116.abs>