

**UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**



GERENCIAMENTO DE PROJETOS BASEADO EM MÉTRICAS

THIAGO JUNQUEIRA VILARINHO

**DEZEMBRO
2006**

GERENCIAMENTO DE PROJETOS BASEADO EM MÉTRICAS

THIAGO JUNQUEIRA VILARINHO

Trabalho de Projeto Final de Curso apresentado por Thiago Junqueira Vilarinho à Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Ciências da Computação.

Professor André Luiz Alves

Orientador

Professor José Luiz de Freitas Júnior,
Dr.
Coordenador de Projeto Final de Curso

AGRADECIMENTOS

Ao professor e orientador André Luiz, por todo o auxílio dado para a realização deste trabalho, pela paciência em me atender nas horas indevidas e pela bronca dada na hora certa para que este projeto se concretizasse.

Às empresas em que trabalhei e à empresa na qual trabalho e aos colegas pela inestimável experiência que me foi passada e da importância que tiveram para a escolha deste tema.

À Universidade Católica de Goiás pela sabedoria em escolher sua equipe de professores, que foram tão importantes durante minha vida universitária.

Por último, à minha família, pelos momentos de carinho, apoio, brigas e problemas.

Dedico este projeto à minha família e amigos... Pelas brigas, diversão e apoio!

RESUMO

Estudo de técnicas de métricas de apoio à atividade de gerenciamento de projetos. As técnicas para medições e métricas utilizadas no projeto é a Análise de Pontos de Função e o COCOMO que auxiliam os gerentes e sua equipe a conhecerem o tamanho do trabalho a ser realizado, a fim de poderem estimar o tempo para a conclusão dos trabalhos e os recursos a serem consumidos durante o projeto. O resultado deste projeto servirá como apoio aos gerentes de futuros projetos iniciarem os trabalhos conhecendo o tamanho do produto final e, sendo assim, o tempo e a quantidade de recursos a serem consumidos durante os trabalhos.

Palavras-chave: Gerenciamento de projetos, Análise de Pontos de Função, COCOMO.

ABSTRACT

Study of techniques of metrics to support activities of project management. The techniques to measure and metrics used in this project is Function Point and COCOMO that help managers and team to know the size of work to be done, to be possible to estimate the time to conclude works and the resources to be consumed during the project. The result of this project will help managers of future projects start the work knowing size of the final product, and on this way, time and resources to be consumed during the work.

Palavras-chave: *Project Management, Function Point Analysis, COCOMO.*

GERENCIAMENTO DE PROJETOS BASEADO EM MÉTRICAS

SUMÁRIO

<u>LISTA DE FIGURAS</u>	IX
<u>LISTA DE TABELAS</u>	XI
<u>LISTA DE ABREVIATURAS E SIGLAS</u>	XIII
<u>Introdução</u>	1
<u>Software e Engenharia de Software</u>	2
2.1. <u>História</u>	2
2.2. <u>Modelos de desenvolvimento</u>	3
2.2.1. <u>Modelo cascata</u>	3
2.2.2. <u>Prototipação</u>	3
2.2.3. <u>Modelo Incremental</u>	4
2.2.4. <u>Modelo RAD (<i>Rapid Application Development</i>)</u>	5
2.2.5. <u>Modelo espiral</u>	5
2.3. <u>Paradigmas</u>	6
2.3.1. <u>Análise estruturada clássica</u>	6
2.3.2. <u>Análise Estruturada Moderna</u>	7
2.3.3. <u>Análise orientada a objetos</u>	8
2.4. <u>Crise do <i>software</i></u>	9
<u>Gerência de projetos de software</u>	12
3.1. <u>Aspectos gerenciais de um projeto de <i>software</i></u>	12
3.1.1. <u>Pessoas</u>	12
3.1.2. <u>Produto</u>	13
3.1.3. <u>Processo</u>	13
3.1.4. <u>Projeto</u>	13
3.2. <u>Atividades do gerente de projeto</u>	14
3.2.1. <u>Gerência da qualidade</u>	14
3.2.2. <u>Gerência de mudanças</u>	15
3.2.3. <u>Gerência de configuração</u>	16
3.2.4. <u>Melhoria do processo</u>	18
<u>Qualidade do produto a partir de métricas</u>	23
<u>Métricas</u>	26
5.1. <u>Modelos de métricas</u>	26
5.1.1. <u>Análise</u>	26
5.1.2. <u>Projeto</u>	27
5.1.3. <u>Desenvolvimento</u>	28
5.1.4. <u>Teste [14]</u>	28
5.1.5. <u>Manutenção</u>	29
5.2. <u>Análise por Pontos de Função</u>	29
5.3. <u>Constructive Cost Model (COCOMO)</u>	34
5.3.1. <u>Delivery Source Instructions (DSI)</u>	34
5.3.2. <u>Modelos</u>	34
5.3.3. <u>Tipos Fundamentais</u>	35
5.3.4. <u>Estimando com COCOMO</u>	35
5.4. <u>Estimativas</u>	40
5.5. <u>Medindo um projeto de <i>software</i></u>	41
<u>Ferramenta para gerenciamento de projetos baseado em métricas</u>	43

<u>6.1. Visão geral</u>	43
<u>6.2. Necessidades</u>	43
<u>6.3. Estrutura</u>	43
<u>6.4. Funcionamento</u>	44
<u>6.5. Domínio da informação</u>	44
<u>6.6. Ambiente para trabalho</u>	45
<u>6.6.1. Ambiente do administrador</u>	45
<u>6.6.2. Tarefas internas do ambiente do administrador</u>	46
<u>6.6.3. Ambiente para o analista e o gerente</u>	46
<u>6.7. Requisitos</u>	47
<u>6.7.1. Requisitos para administradores</u>	47
<u>6.7.2. Requisitos para analistas</u>	67
<u>6.7.3. Requisitos para Operadores</u>	79
<u>6.8. Projeto</u>	81
<u>6.8.1. Diagrama de entidade-relacionamento</u>	82
<u>6.8.2. Diagrama de classe da camada de persistência</u>	83
<u>6.8.3. Diagrama de classe da camada de modelo</u>	84
<u>6.8.4. Diagrama de classe da camada de negócios</u>	85
<u>6.8.5. Diagrama de classes da camada de apresentação</u>	86
<u>6.9. Validação</u>	87
<u>Conclusão</u>	88
<u>REFERÊNCIAS BIBLIOGRÁFICAS</u>	89
<u>GLOSSÁRIO</u>	90
<u>APÊNDICES</u>	91
<u>APÊNDICE A – Medindo o <i>software</i></u>	91

LISTA DE FIGURAS

<u>Figura 1 – Modelo de desenvolvimento em cascata</u>	3
<u>Figura 2 – Modelo de desenvolvimento por protótipos</u>	3
<u>Figura 3 – Modelo de desenvolvimento incremental</u>	4
<u>Figura 4 – Modelo de desenvolvimento espiral</u>	5
<u>Figura 5 – Diagrama de caso de uso para funções do Administrador</u>	45
<u>Figura 6 – Diagrama de caso de uso com as operações permitidas para os Administradores</u>	46
<u>Figura 7 – Diagrama de caso de uso para as funções do analista</u>	46
<u>Figura 8 - Protótipo da tela de associação para fator de ajuste do COCOMO</u>	47
<u>Figura 9 - Diagrama de sequência para requisito F0001</u>	48
<u>Figura 10 - Protótipo da tela de Associação de Rayleigh</u>	49
<u>Figura 11 - Diagrama de sequência para o requisito F0002</u>	50
<u>Figura 12 - Protótipo da tela de Cadastro de linguagens</u>	51
<u>Figura 13 - Diagrama de sequência para o requisito F0003</u>	51
<u>Figura 14 - Protótipo da tela de cadastro de níveis de influência</u>	52
<u>Figura 15 - Diagrama de sequência para o requisito F0004</u>	52
<u>Figura 16 - Protótipo da tela de cadastro de tipos fundamentais</u>	53
<u>Figura 17 - Diagrama de sequência para o requisito F0005</u>	53
<u>Figura 18 - Protótipo da tela de cadastro de atributos de pontos de função</u>	54
<u>Figura 19 - Diagrama de sequência para o requisito F0006</u>	55
<u>Figura 20 - Protótipo da tela de cadastro de características de pontos de função</u>	56
<u>Figura 21 - Diagrama de sequência para o requisito F0007</u>	56
<u>Figura 22 - Protótipo da tela de cadastro de características do COCOMO</u>	57
<u>Figura 23 - Diagrama de sequência para o requisito F0008</u>	58
<u>Figura 24 - Protótipo da tela de cadastro de complexidades de pontos de função</u>	59
<u>Figura 25 - Diagrama de sequência para o requisito F0009</u>	60
<u>Figura 26 - Protótipo da tela de cadastro de escalas do COCOMO</u>	61
<u>Figura 27 - Diagrama de sequência para o requisito F0010</u>	61
<u>Figura 28 - Protótipo da tela de cadastro de fases</u>	62
<u>Figura 29 - Diagrama de sequências do requisito F0011</u>	62
<u>Figura 30 - Protótipo da tela de cadastro de modelos do COCOMO</u>	63
<u>Figura 31 - Diagrama de sequência para o requisito F0012</u>	63
<u>Figura 32 - Protótipo da tela de cadastro da regra de esforço</u>	64
<u>Figura 33 - Diagrama de sequência para o requisito F0013</u>	64
<u>Figura 34 - Protótipo da tela de cadastro de tamanho-padrão</u>	65
<u>Figura 35 - Diagrama de sequência para o requisito F0014</u>	65
<u>Figura 36 - Protótipo da tela de cadastro de tipos de distribuição</u>	66
<u>Figura 37 - Diagrama de sequência para o requisito F0015</u>	66
<u>Figura 38 - Diagrama de sequência para o requisito F0017</u>	68
<u>Figura 39 - Diagrama de sequência para o requisito F0022</u>	71
<u>Figura 40 - Diagrama de sequência para o requisito F0023</u>	73
<u>Figura 41 - Protótipo da tela de cadastro de projetos</u>	74
<u>Figura 42 - Diagrama de sequência para o requisito F0024</u>	74
<u>Figura 43 - Protótipo da tela de cadastro de atributos do projeto</u>	75
<u>Figura 44 - Diagrama de sequência para o requisito F0025</u>	75
<u>Figura 45 - Protótipo da tela de cadastro de características de PF do projeto</u>	76
<u>Figura 46 - Diagrama de sequência para o requisito F0026</u>	76
<u>Figura 47 - Protótipo da tela de cadastro de linguagens do projeto</u>	77

<u>Figura 48 - Diagrama de seqüência do requisito F0027</u>	77
<u>Figura 49 - Protótipo da tela de registro de características para COCOMO</u>	78
<u>Figura 50 - Diagrama de seqüência para o requisito F0028</u>	78
<u>Figura 51 - Diagrama de entidades-relacionamento</u>	82
<u>Figura 52 - Diagrama de classe para a camada de persistência</u>	83
<u>Figura 53 - Diagrama de classe da camada de modelo</u>	84
<u>Figura 54 - Diagrama de classe da camada de negócio</u>	85
<u>Figura 55 - Diagrama de classes da camada de apresentação</u>	86

LISTA DE TABELAS

<u>Tabela 1 - Nível de complexidade para arquivos lógicos internos ou de interface externa</u>	31
<u>Tabela 2 - Nível de complexidade para entradas externas</u>	31
<u>Tabela 3 - Nível de complexidade para saídas externas</u>	32
<u>Tabela 4 - Nível de complexidade para consultas externas</u>	32
<u>Tabela 5 - Cálculo da quantidade de pontos por função não ajustados</u>	32
<u>Tabela 6 - Quantidade de instruções por ponto de função</u>	34
<u>Tabela 7 - Cálculo do esforço</u>	36
<u>Tabela 8 - Determinação do nível de cada atributo</u>	37
<u>Tabela 9 - Determinação do nível para o atributo complexidade</u>	38
<u>Tabela 10 – Fatores de ajuste por atributo do projeto</u>	39
<u>Tabela 11 - Cálculo do prazo</u>	39
<u>Tabela 12 - Distribuição percentual do esforço e prazo pelas fases do projeto</u>	40
<u>Tabela 13 - Equações para estimativa de esforço e prazo</u>	41
<u>Tabela 14 - Requisito F0001 (Associar Escala e Características do COCOMO)</u>	47
<u>Tabela 15 - Requisito F0002(Associação para Rayleigh)</u>	49
<u>Tabela 16 - Requisito F0003 (Cadastrar Linguagem)</u>	51
<u>Tabela 17 - Requisito F0004 (Cadastrar Níveis de Influência)</u>	52
<u>Tabela 18 - Requisito F0005 (Cadastrar Tipos Fundamentais)</u>	53
<u>Tabela 19 - Requisito F0006 (Cadastrar atributos)</u>	54
<u>Tabela 20 - Requisito F0007 (Cadastrar características de Pontos de Função)</u>	56
<u>Tabela 21 - Requisito F0008 (Cadastrar características de COCOMO)</u>	57
<u>Tabela 22 - Requisito F0009 (Cadastrar complexidades)</u>	59
<u>Tabela 23 - Requisito F0010 (Cadastrar escalas)</u>	61
<u>Tabela 24 - Requisito F0011 (Cadastrar fases)</u>	62
<u>Tabela 25 - Requisito F0012 (Cadastrar modelo)</u>	63
<u>Tabela 26 - Requisito F0013 (Cadastrar regra de esforço)</u>	64
<u>Tabela 27 - Requisito F0014 (Cadastrar tamanho-padrão)</u>	65
<u>Tabela 28 - Requisito F0015 (Cadastrar tipos de distribuição de Rayleigh)</u>	66
<u>Tabela 29 - Requisito F0016 (Calcular o Fator de Ajuste COCOMO)</u>	67
<u>Tabela 30 - Requisito F0017 (Calcular tamanho em Pontos de Função)</u>	67
<u>Tabela 31 - Requisito F0018 (Carregar dados para estimar com COCOMO)</u>	69
<u>Tabela 32 - Requisito F0019 (Carregar dados para medir em Pontos de Função)</u>	69
<u>Tabela 33 - Requisito F0020 (Calcular somatória de PF)</u>	69
<u>Tabela 34 - Requisito F0021 (Calcular fator de ajuste dos PF)</u>	70
<u>Tabela 35 - Requisito F0022 (Estimar usando o COCOMO)</u>	70
<u>Tabela 36 - Requisito F0023 (Fazer distribuição de Rayleigh)</u>	72
<u>Tabela 37 - Requisito F0024 (Manter Projetos)</u>	74
<u>Tabela 38 - Requisito F0025 (Registrar atributos do projeto)</u>	75
<u>Tabela 39 - Requisito F0026 (Registrar Características de Pontos de Função)</u>	76
<u>Tabela 40 - Requisito F0027 (Registrar as linguagens utilizadas pelo projeto)</u>	77
<u>Tabela 41 - Requisito F0028 (Registrar características do COCOMO para o projeto)</u>	78
<u>Tabela 42 - Requisito F0029 (Registrar histórico de tamanho de projeto)</u>	79
<u>Tabela 43 - Requisito F0030 (Alteração de registro)</u>	79
<u>Tabela 44 - Requisito F0031 (Exclusão de registro)</u>	80
<u>Tabela 45 - Requisito F0032 (Inclusão de registros)</u>	80
<u>Tabela 46 - Requisito F0033 (Manipular registro)</u>	81
<u>Tabela 47 - Medindo a complexidade dos ALI</u>	91

<u>Tabela 48 - Medindo a complexidade das EE</u>	91
<u>Tabela 49 - Medindo a complexidade das SE</u>	92
<u>Tabela 50 - Medindo a complexidade das CE</u>	92
<u>Tabela 51 - Medindo o total de Pontos de Função não ajustados</u>	92
<u>Tabela 52 - Cálculo do Fator de Ajuste Total</u>	93
<u>Tabela 53 - Cálculo do Fator de Ajuste Total</u>	94
<u>Tabela 54 - Distribuição do esforço durante o projeto</u>	95

LISTA DE ABREVIATURAS E SIGLAS

AIE	Arquivos de interface Externa
ALI	Arquivos lógicos internos
CE	Consulta Externa
CMMI	<i>Capability Marurity Model Integrated</i>
COCOMO	<i>Constructive Cost Model</i>
DD	Dicionário de Dados
DER	Diagrama de entidade-relacionamento
DFD	Diagrama de fluxo de dados
DSI	<i>Delivery Source Instructions</i>
DTE	Diagrama de transição de estados
EE	Entrada Externa
GQM	<i>Goal Question Metric</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IFPUG	<i>International Function Poing Users Group</i>
ISO	<i>International Organization for Standardization</i>
KDSI	1000 * DSI
MPS.BR	Melhoria do processo de Software brasileiro
RAD	<i>Rapid Application Development</i>
SE	Saída Externa
SEI	<i>Software Engineering Insitute</i>
TED	Tipo de elemento de dados
TER	Tipo de elemento de registro
TAR	Tipo de arquivo referenciado
UML	<i>Unified Modeling Language</i>

GERENCIAMENTO DE PROJETOS BASEADO EM MÉTRICAS

CAPÍTULO I

Introdução

Um dos maiores problemas que enfrentamos ao desenvolver *software* é a dificuldade em criar sem que cronograma, orçamento e qualidade definidos inicialmente sofram constantes alterações.

Estes problemas geraram um preconceito contra a área de informática, mesmo sendo de grande importância para toda e qualquer organização. Esse preconceito vem dos constantes problemas enfrentados pelos: usuários, que estão sempre esperando uma correção; clientes, que estão sempre pagando a mais por um produto que nunca termina; equipe de desenvolvimento, que não sabe o que fazer; gerente de projeto, que não sabe como está o andamento do projeto.

Este projeto tem o objetivo de criar ferramentas para solucionar alguns problemas da gerência, de forma que esta tenha mecanismos para controlar sua equipe e planejar a divisão e execução das tarefas de acordo com o andamento do projeto.

Os mecanismos a serem utilizados para a atividade de gerência de projeto são as métricas. Estas métricas ajudarão o gerente de projeto a definir o custo e o prazo para que possa fechar um contrato mais próximo do real, sem que os envolvidos sofram prejuízos.

O gerente de projeto, depois da fase de análise poderá medir, a partir de técnicas como Análise por Pontos de Função, o tamanho do *software*. Depois que o gerente conhece o tamanho do trabalho a ser realizado, medirá a quantidade de esforço prevista para realizar o projeto, utilizando a técnica chamada *Constructive Cost Model* (COCOMO). Com base nessas informações, poderá prever informações como o prazo, quantidade de pessoas para a equipe, custo do projeto ou muitas outras.

O objetivo é criar uma ferramenta automática, de forma que a medição ocorra automaticamente e totalmente transparente para o usuário. Fazendo isso, a consistência e integridade dos dados das medições serão confiáveis, de forma que os clientes saibam quanto o projeto custa para empresa e para o cliente.

CAPÍTULO II

Software e Engenharia de Software

2.1. História

Software, ao mesmo tempo, é um produto que permite o usuário utilizar o potencial computacional embutido do *hardware* sendo utilizado como meio de distribuição de produtos para utilização, o *hardware* no qual está o *software*. A combinação entre *hardware* e *software* fornece o produto mais importante dessa época, a informação. *Softwares* são capazes de transformar grupos de pequenas informações em informações de grande utilidade para as organizações ou indivíduos enquanto o *hardware* auxilia o *software* a receber e mostrar os dados.

O papel do *software* nos últimos 50 anos se tornou muito mais vital para a sociedade quando o desempenho, capacidade e variedade dos *hardwares* aumentaram e as tecnologias se sofisticaram. Esse aprimoramento dos componentes de *hardware* foi alcançado graças à microeletrônica que reduziu o tamanho dos componentes, reduzindo então o tamanho dos equipamentos e ainda fez com que o preço se tornasse acessível à população.

Com a popularização, passou-se a ter uma maior dependência dos *hardwares* e *softwares*. Na década de 70 e 80 diversas publicações sugeriram o impacto dessas novas tecnologias para a cultura mundial, como sendo uma nova revolução industrial ou então, a troca de uma sociedade industrial por uma sociedade da informação. Na década de 90, a estrutura de poder, baseada nas estruturas governo, educação, indústria, economia e forças armadas, foram substituídos pela democratização do conhecimento.

Com o surgimento e explosão da rede mundial de computadores, a função do programador passou a ter maior importância para as organizações, sendo substituídos por equipes de especialistas em desenvolvimento de *software*, onde cada subgrupo se tornava responsável por uma tecnologia utilizada na criação do sistema.

A formação dessas equipes aumentou a disseminação do conhecimento e a agilidade do processo, no entanto, problemas que eram enfrentados persistiam. *Softwares* continuavam sendo entregues com atraso, extrapolavam o orçamento inicial, tinham muitos erros que eram descobertos apenas pelos usuários, não era possível gerenciar o desenvolvimento, estimar o

crescimento ou a demanda no mercado. Esses problemas culminaram na criação de modelos para desenvolvimento que buscavam garantir a qualidade do projeto.

2.2. Modelos de desenvolvimento

Toda engenharia é baseada em processos, métodos e ferramentas. Para atingir a qualidade, os processos de engenharia de *software* devem ser fundamentados sobre modelos de desenvolvimento. O modelo a ser adotado por cada projeto poderá variar de acordo com a necessidade e com o perfil dos profissionais envolvidos no projeto. Entre os modelos de desenvolvimento de software os principais são:

2.2.1. Modelo cascata

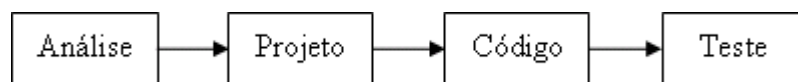


Figura 1 – Modelo de desenvolvimento em cascata

O modelo cascata é o paradigma mais antigo da engenharia de *software* e alguns dos problemas encontrados em sua utilização é que projetos reais raramente seguem um fluxo seqüencial, já que dificilmente um analista consegue extrair todas as informações dos usuários do sistema de uma só vez. Outro problema é que os usuários devem ter muita paciência para esperar para usar o *software* já que este só estará disponível para uso no final do projeto.

2.2.2. Prototipação

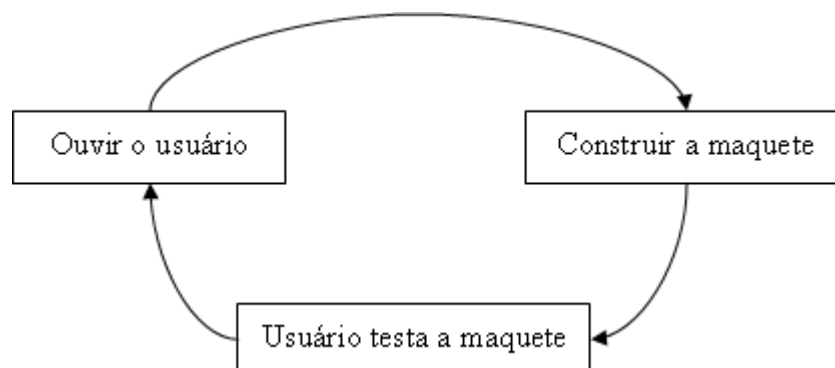


Figura 2 – Modelo de desenvolvimento por protótipos

Esse paradigma é muito utilizado quando o usuário define apenas o objetivo geral do *software*, sem definir quais serão as entradas, saídas ou processamentos ou então, quando o programador não está seguro da eficácia do algoritmo, do funcionamento com um sistema operacional ou da interface homem-máquina.

O problema deste modelo está no fato do protótipo não poder ser usado como produto final, por não assegurar os critérios de qualidade. Portanto, a equipe terá que refazê-lo, gerando um retrabalho e a insatisfação do cliente que já utilizou o programa e não quer esperar pela nova versão.

Caso a equipe entregue o protótipo como produto final, poderá estar entregando um *software* com algoritmos ineficientes, já que para que o protótipo funcione, a implementação é feita rapidamente, sem se preocupar com a execução, plataforma ou a manutenibilidade do programa.

2.2.3. Modelo Incremental

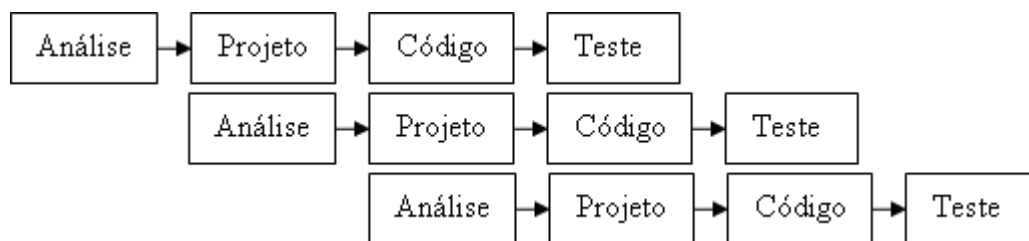


Figura 3 – Modelo de desenvolvimento incremental

Este modelo é uma combinação de elementos do modelo linear com a filosofia da prototipagem. O processo consiste em aplicar seqüências lineares de incrementos, onde ao final de cada incremento, o usuário terá uma parte do *software* totalmente funcional.

A vantagem em utilizar o desenvolvimento incremental está no fato do usuário já começar a utilizar o produto antes do final do projeto e evita atrasos ao tentar conciliar a integração de algumas partes do software com outro *software* ou *hardware* que ainda não esteja disponível. Assim, a equipe continua o desenvolvimento do projeto focando em outros requisitos até que a parte que falta fique pronta e a integração com o módulo possa ser feita.

O problema do modelo incremental está no fato de que as atividades são realizadas de forma incremental, de forma que toda a equipe esteja sempre alocada. No entanto, caso os incrementos não sejam bem planejados, um incremento que seja menor ou mais simples que

seu antecessor, fará com que a o incremento atrase porque a equipe responsável pela atividade ainda estará alocada para a atividade do incremento anterior.

2.2.4. Modelo RAD (*Rapid Application Development*)

O modelo RAD é o modelo incremental que enfatiza ciclos de desenvolvimento extremamente curtos. O rápido desenvolvimento é conseguido pelo uso de componentes no desenvolvimento, o que reduz o trabalho dos desenvolvedores, já que diversas rotinas já estarão prontas para o uso.

O problema do modelo RAD está em:

- Definir o tamanho correto das equipes RAD;
- Comprometer toda a equipe e clientes com o desenvolvimento ágil;
- Construir componentes para projetos que não sejam modularizados;
- Adaptar-se a projetos de alto risco técnico.

2.2.5. Modelo espiral

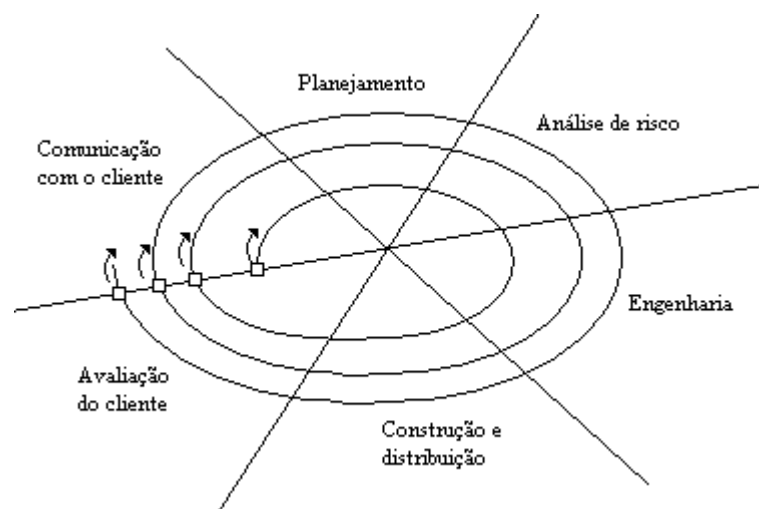


Figura 4 – Modelo de desenvolvimento espiral

O modelo espiral é uma evolução do modelo incremental, onde cada espiral entrega um dos subprodutos do projeto e a cada interação na espiral, o gerente do projeto pode ajustar prazos, orçamento e número de interações a serem cumpridas até o final do projeto, já que a cada interação, é feita uma avaliação do trabalho realizado.

Com essa avaliação, a equipe e o cliente terão maior conhecimento sobre o produto e reagirão melhor aos riscos à medida que o projeto progride.

O problema nesse modelo está no fato de necessitar muita experiência em medir os riscos de cada nível para conseguir o sucesso no projeto, já que risco não gerenciado ou descoberto certamente trará problemas para o projeto.

2.3. Paradigmas

2.3.1. Análise estruturada clássica

Antes da análise estruturada clássica, a análise dos sistemas era feita em forma de uma narrativa que descrevia todas as funções e necessidades dos usuários. Essa forma de documentação da análise, apesar de se assemelhar à forma como os programas eram escritos, é problemática, já que para entender a análise do sistema é necessário ler todo o documento, o que é um problema quando se deseja entender apenas uma parte do problema.

Uma narrativa do problema muito extensa gera vários pontos problemáticos:

- Inconsistência do documento: Alteração em um ponto da narrativa que estiver ligada a outro ponto, uma manutenção de uma parte pode levar a outra a ficar inconsistente;
- Ambigüidade: diversas pessoas poderiam ter diversas interpretações do mesmo documento;
- Dificuldade na manutenção do documento: um documento muito extenso tornaria a sua manutenção impraticável, fazendo com que o documento ficasse obsoleto antes do final do projeto.

Devido a esses problemas, DeMarco [1], na década de 70, criou a análise estruturada para representar o problema graficamente, particionada e sem redundância. Essa forma de análise dos problemas ainda representava a forma como os programas eram escritos, mas facilitava a compreensão da documentação do projeto.

Na análise estruturada, os objetivos do sistema são inicialmente definidos junto ao cliente. Com essa definição dos objetivos, o diagrama de contexto é criado, delimitando toda a fronteira entre o sistema e os usuários.

Com base nos objetivos do sistema e do diagrama de contexto, a lista de eventos a serem executados pelo sistema deverá ser criada, descrevendo assim todos os eventos a que o sistema deverá responder.

Com base no diagrama de contexto e na lista de eventos, o analista passa então a representar o problema graficamente. Para isso, são utilizados diagramas de fluxos de dados (DFD) e dicionários de dados (DD).

No DFD inicial, é mostrado o diagrama de contexto, com toda a interação dos usuários com o sistema. Então a partir da lista de eventos do sistema, o DFD inicial é refinado e os eventos são apresentados e detalhados, até o nível desejado.

Depois, monta-se o DD, que é a representação dos dados em uma forma textual, mostrando os dados levantados nos DFD para cada depósito de dados.

Ao refinar os DFDs do problema, cria-se uma estrutura de análise particionada, já que cada parte do problema estará representada e os seus refinamentos deverão estar relacionados, facilitando assim a completa manutenção de um requisito.

Como cada requisito pode ser visto e alcançado facilmente, fica mais difícil criar redundância dos problemas, dessa forma, a análise do problema fica mais enxuta, reduzindo o tamanho dos documentos, o tempo gasto para entender o problema e para fazer alterações nos requisitos.

Dessa forma, passamos a ter toda a especificação do problema representada graficamente, nas devidas partes e sem redundância de informações.

2.3.2. Análise Estruturada Moderna

Depois de alguns anos de uso da análise estruturada, alguns problemas começaram a ser verificados, até que na década de 90, Yourdon escreveu um livro propondo mudanças na análise estruturada, surgindo então, a análise estruturada moderna.

Na análise estruturada moderna, Yourdon [2] sugere aos usuários evitar a construção dos modelos físicos e lógicos da forma como o sistema atual está funcionando, já que serão descontinuados. Dessa forma, sobrarão tempo no projeto para coisas mais importantes e que ao final do projeto serão efetivamente utilizados.

A distinção entre modelo lógico e físico na análise estruturada era vaga, já que não estabelecia até onde o modelo lógico iria, sem demonstrar as tecnologias que eram utilizadas e eram demonstradas nos modelos físicos do sistema.

Outro problema era representar sistemas de tempo-real. Para isso, foi introduzido o diagrama de transição de estados (DTE), que modela os estados e as transições por que passam para se transformarem em outro estado. Nesse diagrama, era apresentado o porquê iria mudar de estado e como seria o novo estado.

A visualização dos dados e o entendimento dos relacionamentos entre cada entidade do depósito de dados no dicionário de dados também era complicada, por ter uma forma textual. O diagrama de entidade-relacionamento (DER) então foi adicionado para substituir o DD, já que é uma forma mais fácil de visualizar todos os dados e os relacionamentos entre entidades.

O diagrama de contexto também sofreu mudanças, já que passou a ser subdividido por eventos, para facilitar a visualização de cada evento do sistema.

2.3.3. Análise orientada a objetos

A análise orientada a objetos, criada na década de 90, é baseada nos princípios adotados pela análise estruturada moderna, mas, passou a utilizar os recursos de desenvolvimento da orientação a objetos. A orientação a objetos é um mecanismo muito mais robusto para desenvolvimento de aplicações e aprendizagem do problema, já que representa melhor os problemas, apresentando-os da mesma forma como aprendemos quando crianças, determinando o que são os objetos, quais são as características entre os objetos, o todo e as partes de cada problema.

Com base nessa visão, diversos autores começaram a definir a análise orientada a objetos conforme a nova forma de desenvolvimento. No entanto, cada autor deu a sua forma e complexidade à análise do problema, criando então diversas formas de se resolver o mesmo problema pelo mesmo paradigma. Até que Grady Booch, James Rumbaugh e Ivar Jacobson juntaram e criaram a *Unified Modeling Language* (UML) [3] que foi a união das melhores práticas de cada método e definição de diversos diagramas para representação do problema.

Definindo como fazer a análise orientada a objetos e a linguagem a ser utilizada na definição dos problemas, teremos todas as ferramentas necessárias para a análise dos problemas reais para o desenvolvimento das soluções.

A análise orientada a objetos separa cada tipo de visão do problema e então agrupa os diversos diagramas que representem à mesma forma de visualização. Essas visões são:

- Visão do usuário: Utiliza o diagrama de caso de uso, um diagrama simples para que os técnicos possam demonstrar a solução para qualquer nível de usuário;
- Modelo estrutural: Diversos diagramas são associados para formar esse modelo. A intenção dessa classe de modelos é definir os dados e as funcionalidades vista de dentro do sistema;
- Modelo comportamental: Os modelos desta classe são utilizados para demonstrar os aspectos dinâmicos e comportamentais do sistema. Podem ser associados aos diagramas estruturais;
- Modelo de implementação: Representa como os modelos estruturais e comportamentais do sistema serão implementados no produto a ser desenvolvido;
- Modelo ambiental: Representa os aspectos estruturais e comportamentais do ambiente no qual o sistema implementado estará sendo utilizado.

A principal característica desse paradigma está na reutilização do trabalho, para que novos projetos possam aproveitar o trabalho de projetos passados. Essa reutilização de trabalho vem da aplicação de técnicas criadas pela orientação a objetos tais como a herança e o polimorfismo.

Para que uma mesma classe possa ser reutilizada, o analista deverá garantir o domínio do problema que a classe em questão soluciona. Para que isso possa ser assegurado, o analista, no trabalho de levantar os dados para a classe, deverá verificar não só o que o usuário diz, mas também literaturas técnicas, aplicações existentes, experiências passadas, prever requisitos futuros e diversos ambientes. Ao final do levantamento das informações para a especificação da classe, o analista deverá então criar todo o glossário de termos associados ao requisito, os padrões para reuso, os modelos funcionais e a linguagem do domínio. Fazendo todo esse processo, a equipe poderá utilizar as classes da biblioteca em diversos projetos, já que uma mesma classe poderá ser utilizada em diversos ambientes.

2.4. Crise do *software*

A “crise do *software*” foi caracterizada por problemas com: entrega do *software* no tempo e orçamento estipulados inicialmente; controlar o desenvolvimento do *software*; medir a qualidade do produto final; gerenciar toda a equipe de desenvolvimento.

A principal causa está no imediatismo dos profissionais que para ganharem tempo no projeto, ao invés de realizar uma análise correta e completa do problema, projetar o sistema e prever futuras manutenções, vão direto para a construção do *software* podendo assim, reinventarem a roda, não planejarem futuras manutenções ou o crescimento do produto e não conseguir medir o progresso do projeto.

Essa cultura imediatista de desenvolvimento de *software* criou diversos mitos, que como todo mito, pode ser traiçoeiro, mas, com base na experiência das fontes, são tidos como verdadeiros.

Mitos dos gerentes do projeto podem ser:

1 – Possuir um livro com os padrões e procedimentos de construção de *software*, a construção do projeto estará garantida.

Se não for efetivamente utilizado pela equipe, não for atualizado e os procedimentos não forem eficientes, não irá servir para nada ter um guia desses.

2 – Os profissionais usarem computadores de última geração.

Possuir ferramentas de apoio para o desenvolvimento é mais importante que computadores potentes, que, normalmente, nem são completamente utilizados.

3 – Adicionar pessoas para adiantar um projeto atrasado.

Se não estiverem muito bem sintonizados ao projeto, adicioná-los, só irá atrasar ainda mais o projeto, já que o restante da equipe gastará um tempo ensinando o projeto aos novatos.

4 – Terceirizar um serviço para não precisar se preocupar com ele.

Deverá ter um maior controle sobre a empresa terceirizada para que essa realize o projeto com sucesso.

Mitos do cliente podem ser:

1 – Propósito geral do sistema é suficiente para começar o programa.

Os detalhes do projeto são essenciais para seu sucesso.

2 – Os requisitos são inconstantes, mas *software* é flexível e fazer mudanças é fácil.

Uma mudança pode impactar em todo o *software*, trazendo um grande prejuízo para o projeto.

Mitos da equipe do projeto podem ser:

1 – O trabalho termina quando o *software* é entregue.

A maior parte do trabalho em um *software* está na manutenção.

2 – Antes que o programa funcione, não é possível assegurar a qualidade.

A qualidade de um projeto é medida desde o início do projeto.

3 – O único produto a ser entregue é o *software*.

Além do próprio *software*, entregar a documentação também é fundamental.

4 – Engenharia de *software* cria uma volumosa e desnecessária documentação que atrasará o projeto.

A engenharia visa à qualidade, o que reduz o retrabalho. Sem retrabalho, o produto será entregue muito mais rápido e melhor.

O foco desse projeto é criar uma ferramenta de apoio ao gerente de projeto, para que possa planejar e monitorar suas atividades do projeto, sem recair sobre os mitos acima. Dessa forma, poderá:

- Controlar o bom uso dos padrões e procedimento da empresa;
- Estimar o tamanho do projeto;
- Estimar o tamanho da equipe;
- Verificar projetos passados como base de estimativas para novos projetos;
- Controlar os requisitos do projeto;
- Controlar a equipe do projeto em suas atividades;
- Agrupar a documentação do projeto.

CAPÍTULO III

Gerência de projetos de software

3.1. Aspectos gerenciais de um projeto de *software*

Uma gerência efetiva de projetos de *software* deve focar nos quatro pontos principais, que são: pessoas, produto, processo e o projeto. O gerente que não dê a devida atenção a todos esses pontos, nessa ordem, tende a ver o seu projeto não ser bem-sucedido.

Esses quatro pontos são interligados e a importância deles está no fato do processo de engenharia do produto ser um trabalho basicamente humano e sem a devida atenção à equipe, os profissionais tendem a não dar importância ao projeto; sem estimular a clara comunicação entre clientes, usuários e a equipe de desenvolvimento, o gerente corre o risco de controlar um projeto de um belo produto para o problema errado; sem estimular e controlar o processo definido para o projeto, boas ferramentas e métodos não servirão em nada; e, por último, aquele gerente que não tenha um plano de projeto bem definido, não conseguirá terminar o projeto com sucesso.

3.1.1. Pessoas

A importância do fator pessoas dentro de um projeto é tão grande que o *Software Engineering Institute* (SEI) desenvolveu o modelo de maturidade em gerenciamento de pessoas (PM-CMM). As áreas-chave para a equipe do projeto estão ligadas ao recrutamento, seleção, gerenciamento do desempenho, treinamentos, crescimento profissional, organização e desenvolvimento em time.

Manter pessoas motivadas e com alto nível técnico ajuda a garantir um projeto bem-sucedido, mas a equipe deve possuir confiança mútua tanto entre os integrantes quanto com os níveis superiores. As estrelas do time devem ser evitadas para que o time não se desintegre, já que um time unido é muito mais eficiente que partes do time, rivalizando-se com as demais.

Para manter os integrantes da equipe motivados, a gerência do projeto deverá garantir que as atividades atribuídas aos integrantes da equipe estão de acordo com as capacidades técnicas de cada um, de forma a não frustrar um bom profissional com atividades simples ou

um profissional menos capacitado com uma atividade que não conseguirá exercer completamente.

3.1.2. Produto

Para que o produto final a ser entregue ao cliente seja aquilo que o cliente realmente queria, no início do projeto, a equipe de desenvolvimento, o cliente e os usuários devem estabelecer quais serão os objetivos a serem alcançados com o produto. A partir dessa informação é que o esforço, o custo, os riscos, a divisão de tarefas e o calendário poderão ser definidos.

Depois que a equipe e o cliente definem o objetivo do produto e o escopo do projeto é que a análise poderá ser feita, pois só nesse ponto é que as funcionalidades poderão ser definidas e avaliadas.

Com a definição do objetivo, escopo e as funcionalidades desejadas, a equipe de desenvolvimento e o cliente poderão encontrar alternativas para as restrições que o projeto tenha, tais como: tempo, custo, disponibilidade de pessoal e interfaces técnicas.

3.1.3. Processo

A base de trabalho para um projeto é o processo que for adotado pela equipe.

Existem vários modelos de processo de desenvolvimento de *software*, onde organizam as atividades de desenvolvimento de *software* de uma forma diferente. No entanto, nenhum modelo é o melhor ou o pior. Cada modelo tem seus prós e contras e o modelo a ser escolhido para um projeto, nem sempre será o mesmo para outro projeto.

Todos os processos são divididos em diversas atividades e para a execução dessas atividades, existem diversas metodologias que irão conduzir o andamento da atividade em questão.

3.1.4. Projeto

Todos os projetos precisam ser planejados, executados e controlados. Para que o projeto termine com sucesso, o esforço, tempo e custo envolvidos precisam ser estimados, marcos de controle de qualidade e as ferramentas precisam ser definidos, e todas as fases

precisam ser monitoradas sempre, para que em caso de necessidade, possam ser redefinidas sem grandes perdas para o projeto.

3.2. Atividades do gerente de projeto

A gerência de projetos de software relaciona diversas atividades que um gerente de projeto deve garantir para que tudo ocorra de forma segura, controlada e mensurável.

Entre os pontos que devem ser controlados pelo gerente, estão:

- Gerência da qualidade;
- Gerência de mudanças;
- Gerência de configuração;
- Melhoria do processo.

3.2.1. Gerência da qualidade

A atividade de garantia da qualidade deve ser adotada pelo gerente do projeto para evitar que o produto entregue ao cliente possua defeitos ou problemas que não tenham sido identificados durante o processo de desenvolvimento do software. Assim, evita que o cliente se sinta insatisfeito ou inseguro com o uso do *software*.

O problema é quando a equipe deve garantir atributos de qualidade do software. Atributos como manutenibilidade, portabilidade e eficiência não são definidas explicitamente, mas afetam toda a qualidade do projeto.

Bons gerentes de projetos devem estimular a qualidade como uma cultura para produção dentro da organização, estimulando os integrantes da equipe para que se sintam responsáveis não só por garantir a qualidade do produto, mas por melhorar até mesmo o processo de controle de qualidade.

Apesar da gerência de qualidade fazer parte do projeto, a equipe que deve verificar se os padrões de qualidade estipulados estão sendo seguidos, não pode fazer parte da mesma equipe de desenvolvimento, para que dessa forma, não esteja envolvida nos problemas de tempo e custo e assegure imparcialmente a qualidade do produto.

3.2.2. Gerência de mudanças

O desenvolvimento de qualquer sistema, independente do tamanho, haverá manutenções a serem feitas durante a vida do *software*. Para que as mudanças impactem o mínimo possível no sistema e no trabalho da equipe de desenvolvimento, a qualidade deverá ser assegurada para reduzir o esforço gasto nas manutenções.

Existem três tipos básicos de manutenção em um sistema durante a vida:

- Manutenção de requisito: Quando o requisito começa a ser utilizado ou em algum ponto de sua vida, é alterado para que atenda às novas necessidades do usuário. Pode ser uma modificação simples, já que irá mexer apenas nos componentes associados ao requisito;
- Transformação arquitetural: Essa manutenção é mais radical que uma simples manutenção de requisito. Mudanças significativas podem ser feitas na arquitetura do *software*. Por exemplo, passar um sistema de monousuário para cliente/servidor;
- Reengenharia do *software*: Nada será acrescentado ao *software*. O *software* será modificado para que se torne mais simples de entender ou manter. Na reengenharia do *software* acontecem algumas mudanças arquiteturais, mas normalmente, não são mudanças muito grandes.

Essas mudanças podem acontecer simultaneamente. Adição de novos componentes ou modificação de requisitos poderá levar a uma reengenharia do *software*.

Entretanto, alguns sistemas precisam de tantas mudanças, que comprar algum outro *software* ou refazê-lo, se torna uma opção mais fácil e/ou barata. Às vezes, refazer um *software* que a mesma empresa produziu pode ser uma vantagem, já que diversos programas que já estão em uso, poderão ser reutilizados, de forma a ganhar tempo e diminuir o custo do projeto.

Mudanças no *software* são muito caras para o projeto. Se forem freqüentes ou prejudiciais ao ambiente do sistema, causará o descontentamento dos usuários, além do custo do projeto aumentar.

Estimativas de manutenções de *software* dizem que as manutenções de um *software* de negócio variam de 50 a 80% do esforço e custo do projeto, enquanto em sistemas de tempo-real podem chegar até quatro vezes o valor de desenvolvimento do *software*.

A única maneira de diminuir os gastos com manutenções é fazendo um bom trabalho de análise, projeto e desenvolvimento. E mais ainda, prever futuras mudanças e criar mecanismos de controle para que alguma mudança nesses requisitos cause o mínimo de impacto sobre o sistema.

Alguns aspectos que aumentam o custo e o esforço de uma mudança no sistema estão relacionados com a estabilidade dos integrantes da equipe de desenvolvimento, já que depois que um projeto é concluído, a equipe tende a se desmanchar e os integrantes passam a outros projetos e no caso de manutenção, novos integrantes deverão entender todo o projeto antes de começar a fazer as alterações.

Outro aspecto que influencia no custo de uma manutenção são contratos de manutenção feitos com outra empresa, que não a que produziu o *software*. O que pode acontecer é a empresa que desenvolveu o *software* não ter se preocupado com a qualidade do produto, dificultando o trabalho da empresa que irá dar manutenção.

Por último, um aspecto muito importante são as habilidades da equipe de manutenção, que podem não ser as mesmas da equipe que desenvolveu o sistema. Com isso, a estrutura, linguagem utilizada e os conhecimentos necessários do sistema podem se tornar um problema para o esforço gasto na manutenção do sistema.

A manutenção que consome o maior número de esforço e dinheiro é uma evolução arquitetural. Com as novas tecnologias, organizações que utilizam sistemas centralizados, a um alto custo de hardware, passam a querer que seus sistemas funcionem de maneira distribuída para que o custo com hardware diminua. No entanto, fazer mudanças nesse nível em um sistema é extremamente dispendioso e o que o gerente de projeto deverá avaliar é se o custo e o esforço gasto em uma manutenção como esta é viável ou se desenvolver um novo projeto que use essas tecnologias se torna a melhor opção.

O interessante na gerência de modificação é o trabalho que o gerente e sua equipe passam a ter, de forma a reduzir o retrabalho com manutenções não previstas, diminuir o custo e o esforço gasto no projeto e ainda, melhorar a qualidade do produto de forma que o cliente não se sinta inseguro ao utilizar o *software* desenvolvido.

3.2.3. Gerência de configuração

A gerência de configuração é a adoção de padrões e procedimentos de controle de sistemas que estejam evoluindo. A necessidade em monitorar a evolução do *software* é que o

mesmo *software* pode ser distribuído para diferentes *hardwares* ou sistemas operacionais e é interessante controlar o ambiente de cada usuário para que seja distribuída a melhor versão do produto.

Esse controle é visto como uma parte do controle de qualidade do *software*, já que o mesmo gerente pode cuidar das duas áreas simultaneamente. Sendo uma parte da qualidade do produto, a equipe de desenvolvimento passa a nova versão para o controle de qualidade e quando esta é aprovada, o próprio controle de qualidade se responsabiliza por manter uma cópia dessa versão.

Existem diversos problemas relacionados ao processo de *software* e a gerência de configuração. Por exemplo, o modelo cascata de desenvolvimento, entrega o *software* para a área de qualidade quando todo o teste tiver sido feito e o *software* estiver pronto. Dessa forma, a gerência de configuração armazenará apenas uma versão, sem que tenha o histórico da manutenção ou experiências adquiridas.

Algumas organizações utilizam a gerência de configuração para construir uma versão do *software* diariamente. Essa prática exige que os desenvolvedores passem as alterações até determinado horário, para que a gerência possa criar a versão e distribuir à equipe de teste. Neste caso, cria-se um problema com a equipe de desenvolvimento, que passa a ser forçada a entregar os componentes feitos ou alterados antes mesmo de concluir os testes. Assim, o produto final pode ter problemas com o componente não testado e o desenvolvedor se torna responsável pelo fracasso da versão.

O banco de dados da gerência de configuração deve seguir determinados padrões para que assegure o impacto de mudanças e prover informações gerenciais sobre cada versão do *software*.

As informações que a gerência de configuração deve possuir estão relacionadas com os clientes que utilizam cada versão, os requisitos necessários do ambiente e dos usuários, o número de versões criadas, as versões que serão modificadas ao se alterar determinado componente do *software*, o número de modificações feitas em uma versão e o número de erros relatados de cada versão.

Essas informações servirão como base para que o gerente do projeto estime futuras manutenções, meça a qualidade do produto entregue, o esforço e o custo de cada versão e inúmeras outras informações de grande valor para o sucesso de um projeto.

A gerência de configuração é muito importante para todo o processo, já que é ela que irá manter todo o histórico de versões, mudanças, problemas e pedidos de mudanças do

software. Portanto, passa a ser um banco de conhecimento da organização tanto para projetos atuais quanto para projetos futuros. Deverá ainda administrar o banco de bibliotecas de componentes da organização, já que eles possuem informações do que cada componente faz como faz e onde é utilizado, de forma a reduzir o tamanho, esforço e custo dos projetos.

Essa área é responsável por decidir a versão e o momento que será distribuída aos clientes. Encarregam-se ainda de preparar a forma de distribuição, manuais e documentos relacionados à versão.

Ao distribuir uma versão do software, o gerente de configuração deve garantir o envio dos arquivos de configuração, de dados, de instalação, documentação e o material promocional do produto. No entanto, o cliente nem sempre irá instalar a versão sugerida, por isso, o gerente também deverá se preocupar em não enviar uma versão que dependa de outra versão antecessora.

Portanto, a função do gerente de configuração é extremamente importante para a conclusão com sucesso de todo o processo de produção do software, já que é ele quem deverá assegurar o histórico, versões sendo utilizadas e as novas versões a serem utilizadas pelos clientes.

3.2.4. Melhoria do processo

O assunto mais falado nos últimos anos na comunidade de engenharia de *software* é a melhoria do processo de desenvolvimento. A melhoria de processo visa melhorar a qualidade dos produtos com redução do custo e do tempo de produção. Mas para se ter uma melhoria de processos, deve-se entendê-lo a fundo para que seja possível perceber o que pode e o que deve ser melhorado.

Para implantar processos, nem sempre basta implantar um processo que tenha sido bem-sucedido em outra organização, já que cada organização possui visões, métodos, políticas, produtos e pessoas diferentes. Por isso, cada organização deve encontrar os métodos que formem melhor seu processo de desenvolvimento. O que pode (e deve) ser feito é se basear em práticas de sucesso das outras organizações para a criação do seu próprio processo.

Implantação de um processo ou melhoria dos processos existentes são atividades que demandam muito tempo e exigem suporte da alta diretoria da organização e disponibilidade de recursos.

O suporte da alta diretoria é essencial à melhoria do processo para que não faltem recursos para as atividades necessárias e para que a equipe participe e esteja realmente interessada no projeto.

Melhoria de processos deve executar os cinco passos fundamentais, que são:

1. **Análise do processo:** É necessária para criar a documentação de como a produção é feita atualmente para que tenha material para documentar e entender como o processo é executado antes que mude. Análises quantitativas antes e depois da melhoria do processo são necessárias, para medir a eficácia do trabalho realizado;
2. **Definição da melhoria:** Nesse ponto, pode-se usar a análise do processo para identificar os gargalos de cronograma, orçamento e qualidade dos projetos em execução e definir o que deverá ser feito para melhorar;
3. **Introduzir as mudanças:** Introduzir as atividades, métodos e ferramentas no ambiente de desenvolvimento, integrá-los com o que já existe e garantir que a integração seja compatível com o que existe;
4. **Treinar:** Não existem mudanças de processos sem treinamento dos envolvidos. Dessa forma, só com um bom treinamento das pessoas envolvidas nas novas atividades, procedimentos e ferramentas é que levará as melhorias a serem bem-sucedidas.
5. **Afinar as mudanças:** Depois da implantação do novo processo, deve-se ir afinando os procedimentos e atividades com o restante do ambiente de produção até que seja totalmente efetivo.

Atualmente existem diversos modelos consagrados de processos de desenvolvimento como o *Capability Maturity Model Integrated* (CMMI) [4] e a Melhoria de Processos de Software Brasileiro (MPS.BR). Esses modelos são baseados na melhoria incremental dos processos organizacionais, medindo cada evolução em níveis. Por exemplo, no CMMI existem cinco níveis de maturidade enquanto no MPS.BR [5] existem 7 níveis.

Os níveis de maturidade do CMMI são:

- **Inicial:** Todas as organizações que não possuem procedimentos formais de gerenciamento ou de controle dos projetos que desenvolvem. O custo, cronograma e qualidade dos projetos dessas organizações são imprevisíveis;
- **Repetido:** A organização já possui gerenciamento formal, controle da qualidade e gerência de configuração, o que faz com que consiga repetir os resultados nos projetos desenvolvidos;

- Definido: Nesse nível de maturidade, a empresa já possui um processo formal definido e já começa a visar a melhoria do processo de qualidade dos produtos produzidos;
- Gerenciado: A organização já possui dados suficientes para criar métricas dos projetos que produz e faz estimativas melhores para os novos projetos;
- Otimizado: Nesse último nível de maturidade, a melhoria dos processos já faz parte da melhoria da organização, que já está de acordo com a melhoria dos processos.

Os níveis do MPS.BR são:

- G (Parcialmente gerenciado): A organização possui uma gerência de projeto e de requisitos bem definidos;
- F (Gerenciado): A gerência de configuração e a garantia da qualidade passam a existir. A partir desse nível, o nível repetido do CMMI (nível 2) é satisfeito, pois a organização também contempla o nível G;
- E (Parcialmente definido): Começa-se a definir os processos para treinamento, melhoria dos processos organizacionais, o processo é definido e a gerência dos projetos se adaptam ao novo processo;
- D (Largamente definido): Definem-se processos para desenvolvimento de requisitos, soluções técnicas, integração entre produtos, instalação, liberação, verificação e validação final;
- C (Definido): A organização garante o nível definido do CMMI (nível 3), garantindo a análise de decisão e resolução e gerência de riscos. Nesse ponto, o processo contempla os níveis E e D e o processo está institucionalizado;
- B (Gerenciado Quantitativamente): A organização estará no nível gerenciado do CMMI (nível 4). Nesse nível, o processo organizacional passa a ser medido e uma gerência quantitativa do projeto é feita;
- A (Em otimização): Corresponde ao nível otimizado do CMMI (nível 5). Garante a inovação e implantação na organização, faz uma análise e resolução de causas dos projetos e está em melhora contínua do processo.

As idéias de melhoria de processo começaram a ser discutidas pelo engenheiro americano W. E. Deming que trabalhou para a melhoria da qualidade das indústrias japonesas depois da Segunda Guerra Mundial.

Deming introduziu a idéia do controle estatístico da qualidade, onde o número de defeitos do produto são medidos e relacionados com os problemas do processo. Dessa forma, Deming conseguiu uma melhora no processo introduzido através da redução do número de defeitos dos produtos que eram produzidos.

A análise do processo começa de forma qualitativa, onde o analista irá determinar quais são as características do modelo. Mais tarde, o processo passa a ser analisado mais detalhadamente através das métricas coletadas. Por último, o processo será descrito usando o modelo do processo.

A medição quantitativa de um processo deverá medir se o processo é eficiente ou não para a organização. No entanto, apenas dados concretos podem ser estimados, portanto, utiliza-se o tempo ou orçamento gasto em uma atividade específica para testar o quanto necessária são as atividades implantadas e até que ponto é aplicável.

As medições dos produtos criados e as do processo em questão podem ser associadas, de forma a criar métricas que garantam informações para que a equipe aperfeiçoe cada atividade especificamente e o projeto seja bem sucedido.

As métricas para processos podem ser classificadas como:

1. Tempo gasto para o processo completar;
2. Recursos gasto em um processo particular;
3. Número de ocorrências de determinado evento.

A dificuldade em medir processos ou produtos está em saber o que medir. Para ajudar nesse problema, Basili e Rombach em 1988 propuseram o *Goal Question Metric* (GQM) que auxilia na definição de medições a serem tomadas. O GQM é:

- *Goal* (Objetivo): O que a organização procura alcançar com o processo;
- *Questions* (Questões): Os refinamentos dos objetivos de uma área específica que não tenha objetivos definidos;
- *Metrics* (Métricas): Conjunto de dois ou mais tipos de medições que servem para analisar quando as melhorias no processo estarão conquistadas.

Para uma eficiente análise do processo, todos os envolvidos na área deverão participar e, então, aprovar as melhorias dos processos. Ao definir as medições do processo, têm-se as métricas. Essas métricas do processo podem (e devem) ser associadas às medições do produto, de forma a melhorar tanto o produto quanto o processo.

O interesse deste projeto em melhorar os processos de desenvolvimento de *software* está em utilizar as métricas de *software* como forma de avaliar quantitativamente o que foi melhorado no processo a cada novo projeto.

CAPÍTULO IV

Qualidade do produto a partir de métricas

O foco principal de qualquer disciplina de engenharia é a qualidade do produto a ser desenvolvido. Assim como os engenheiros, os clientes e a empresa que produz o produto também têm interesse em obterem produtos de qualidade.

Produtos de qualidade, produzidos sob processos de qualidade normalmente consomem menos recursos, menos tempo, são entregues com menos defeitos, o cliente se sente confiante e satisfeito com o produto adquirido e a empresa desenvolvedora passa a ter vantagens em futuros projetos.

Em outras engenharias é possível estimar o projeto e assegurar a qualidade do produto, pois o produto é real, visível e mensurável. Com *software*, não é possível fazer análises e estimativas da mesma forma por se tratar de algo virtual, que não pode ser medido ou avaliado. No entanto, foram estudados indicadores quantitativos para medir esses atributos.

Para que esses indicadores sejam obtidos, é necessário que uma ou mais métricas sejam estabelecidas. Métricas é o relacionamento entre duas ou mais medições, que são a contagem de dados pontuais do *software*.

Existem várias formas de se extrair medidas, mas, medidas individuais não são relevantes. A equipe só se beneficia dessas medidas quando associadas a outras, formando métricas. Mesmo as métricas devem ser bem planejadas e devem ser consideradas dentro de cada fator de qualidade.

A norma *International Organization for Standardization* (ISO) 9126 identificou alguns fatores de qualidade para *software*. São eles:

- Funcionalidade: O nível de qualidade, interoperabilidade, conformidade e segurança com que o *software* realiza as funcionalidades. Um exemplo de métrica a ser adotada é defeito por quantidade de linhas de código;
- Confiabilidade: O tempo necessário para que o software esteja pronto para ser utilizado. Indicado pelo nível de maturidade da organização e pela tolerância às falhas. Um exemplo de métricas pode ser quantidade de defeitos por versão;
- Usabilidade: A facilidade com que o usuário consegue entender, aprender e operar o *software*. Métricas aplicáveis a esse fator podem ser: tempo necessário para começar a utilizar o sistema ou através de questionários com os usuários;

- Eficiência: O nível de otimização dos recursos do sistema e do tempo. Um exemplo de métrica é a análise dos recursos consumidos por operação;
- Manutenibilidade: A facilidade em realizar correções no *software*. Indicado por fatores de análises, mudança, estabilidade e teste. Uma métrica que pode ser usada para esse fator é o tempo médio gasto por mudança;
- Portabilidade: A facilidade em mudar o ambiente do *software*. Indicado pela adaptabilidade, instalação, conforme e troca. Uma métrica que pode ser aplicada para esse fator é o tempo gasto para mudar o ambiente do sistema.

A necessidade em se garantir esses fatores é para que a equipe, de acordo com a ISO 9126, durante o projeto, possa: verificar ou prever o tempo e recursos consumidos até determinado momento; a quantidade a ser gasto até outro ponto; garantir a qualidade do produto a ser entregue; definir o cronograma para as atividades; diminuir o número de defeitos reclamados; garantir a conclusão dos requisitos solicitados; facilitar futuras correções ou manutenções.

Outra necessidade em analisar quantitativamente os projetos é para a melhoria do processo de desenvolvimento. Como o CMMI estabelece, os processos amadurecem a cada nível, e para que esse amadurecimento aconteça, é necessário que cada projeto seja documentado, medido, avaliado e, ao final os pontos de sucesso e fracasso sejam revisados.

O processo de medição e métricas sugerido por Roche [6] contém cinco atividades:

1. Formulação: Derivar medidas e métricas apropriadas para a representação do *software*;
2. Coleção: Acúmulo de informações sobre as medições de outros projetos para ajudar no trabalho de derivação dessas métricas;
3. Análise: Computação das métricas;
4. Interpretação: Avaliação dos resultados das métricas para entender a qualidade representada;
5. Retroalimentação: Recomendações obtidas da interpretação das métricas e transmitidas à equipe.

Os princípios associados à criação das métricas, ainda segundo Roche são:

1. Estabelecer os objetivos das medições antes de começar a coletar os dados;
2. Cada métrica deve ser definida de forma clara;

3. As métricas podem ser derivadas com base no domínio da aplicação;
4. Métricas devem ser acomodadas a produtos e processos específicos;
5. Automatizar o processo de coleta dos dados de medição;
6. Recomendações sobre cada métrica devem ser feitos.

Métricas podem servir para dois propósitos. Não sendo mantidas e utilizadas de forma eficaz, atrapalham o processo. Quando empregadas corretamente, melhora o processo de desenvolvimento, a qualidade do produto e ainda o trabalho e o relacionamento entre a equipe e os clientes.

Para que os processos de medição sejam realizados corretamente, é necessário um suporte gerencial forte sobre a equipe de coleta das medidas e sobre aqueles que geram ou analisam as métricas. A gerência do projeto dando a devida atenção a essas atividades, os envolvidos no projeto ganham.

O suporte gerencial sobre o processo de coleta de métricas deve ser focado sobre os seguintes atributos:

1. Simples: As métricas devem ser simples de entender;
2. Intuitivo: Fornecer ao engenheiro noções intuitivas dos atributos em questão;
3. Objetivo: Obtido de forma clara para que outras pessoas ou projetos, com os mesmos dados, cheguem aos mesmos resultados;
4. Consistente: Não fazer combinações complicadas de unidades de medidas;
5. Independente de linguagem: Métricas são baseadas apenas na análise, projeto ou estrutura do software, evitando assim, análises incorretas das métricas;
6. Ser um mecanismo eficaz de alta qualidade: Deve prover ao gerente informações para garantir a alta qualidade do produto final.

Mesmo que esses atributos sejam essenciais, os modelos de métricas não conseguem cumprir todos eles. Nesse caso, a equipe deve definir quais são os atributos mais importantes para o projeto.

Este projeto final de curso deverá garantir que o processo de medição, para formação das métricas adotadas para os projetos, seja o mais automático possível, de forma a diminuir o número de tarefas da equipe de desenvolvimento e, mesmo assim, suprir os gerentes e líderes de projeto com as informações necessárias para garantirem a qualidade do produto.

CAPÍTULO V

Métricas

Para obter as métricas no processo de desenvolvimento de *software*, existem diversos modelos de medições. Cada modelo é utilizado em uma atividade da engenharia de *software*. Medições podem ser obtidas nas atividades de análise, projeto, desenvolvimento, teste e manutenção. Portanto, no início do projeto, a equipe deverá definir as medições e métricas a serem utilizadas.

5.1. Conceitos

5.1.1. Medida

Uma medida é o padrão para o tamanho de uma característica do software.

Em projetos de software, servem para comparar uma característica do software com alguma notação real.

Alguns exemplos de medida podem ser: Ponto de Função (PF), Linha de Código (LOC) ou *Delivery Source Instructions* (DSI).

Para se compreender melhor uma medida, pode-se fazer a comparação com as medidas de peso: grama (G) ou libras; temperatura: graus Celsius ou graus Fahrenheit; distância: metros ou jardas.

5.1.2. Medição

Uma medição é o ato de calcular o tamanho da característica desejada do software, comparando com o padrão de medida desejado.

O uso de uma medição em projetos de software visa à obtenção de dados numéricos para o tamanho de determinada característica do software.

Um exemplo de medição, é o ato de medir o tamanho do software em Pontos de Função (PF).

5.1.3. Métrica

Uma métrica é o resultado da comparação entre dois ou mais resultados de medições realizadas.

O objetivo de uma métrica é apresentar a quem interesse, estimativa do projeto, afim de avaliar o tempo a ser consumido para cada funcionalidade, esforço necessário para a realização do projeto ou outras estimativas necessárias.

Exemplos de métricas podem ser Pontos de Função por Tamanho da Equipe, Homens por mês (H/M) e Pontos de Função por mês.

5.1.4. Indicador

Um indicador é a comparação entre uma métrica e o resultado esperado para o projeto.

~~An-~~eEntre outras funcionalidades, um indicador serve para avaliar a situação atual do projeto e indicar numericamente o andamento do projeto.

Exemplos de indicadores pode ser a quantidade de Pontos de Função concluídos pela quantidade total medida para o sistema, quantidade de dias consumidos para realizar uma quantidade de Pontos de Função pela quantidade de dias estimados para o projeto ou o tamanho total medido do projeto.

5.1. Modelos de métricas

Para cada atividade de engenharia de *software*, existem os seguintes modelos:

5.1.1. Análise

Nesta atividade, o trabalho técnico do projeto de *software* é iniciado, levantando os requisitos e definindo a base do projeto.

Métricas nesta atividade apenas estimam o tamanho final do *software*.

5.1.1.1. Pontos de função

Utiliza como base para medir o tamanho do *software*, a quantidade de entradas, saídas e consultas com o usuário e o número de arquivos e interfaces externas.

A quantidade de pontos de função é então ajustada de acordo com 14 fatores de ajuste. Essa pontuação ajustada será o tamanho do *software* em pontos de função.

De posse dessa informação, o gerente de projeto, ainda na fase de análise, poderá estimar o tamanho do software a ser construído. Tendo uma base histórica de projetos, será capaz de estimar o tempo, custo e uso de recursos no projeto.

5.1.1.2. Bang

Criado por DeMarco [7], o modelo Bang segue o mesmo princípio do modelo de Pontos de Função. A diferença é que o analista deverá avaliar um pequeno conjunto de primitivas e não por funções que depois serão ajustadas.

As principais primitivas são:

1. Primitivas funcionais: Número de transformações do nível mais baixo do DFD;
2. Elementos de dados: Número de atributos de um objeto de dados;
3. Objetos: Número de objetos de dados;
4. Relacionamentos: Número de conexões entre objetos de dados;
5. Estados: Número de estados observáveis na transição entre estados;
6. Transições: Número de estados transitórios no diagrama de estados.

Outra característica do modelo Bang é dividir o *software* nos domínios: Funcional e Dados. Onde *softwares* voltados para a engenharia ou aplicações científicas são funcionais enquanto sistemas de informação são guiados por dados.

5.1.1.3. Especificação de qualidade

O modelo de métricas pela especificação da qualidade, criado por Davis [8], propõe uma lista com características de qualidade para o modelo de análise. Essas características são: não ambíguo, integridade, exatidão, entendimento, comprovação, consistente, executável, conciso, rastreável e modificável.

Mesmo a maior parte dessas características sendo qualitativas, o modelo sugere que possam ser medidas através de uma ou mais métricas.

5.1.2. Projeto

5.1.2.1. Arquitetural

Focado nas características da arquitetura e na efetividade dos módulos, não precisam de nenhum conhecimento do trabalho interno do *software*.

Esse modelo foi criado por Card e Glass [9], onde definiram três medidas de complexidade do projeto. Complexidade estrutural, complexidade de dados e complexidade do sistema.

5.1.2.2. Nível de componente

O modelo de métricas no nível de componentes diferentemente do nível arquitetural, foca nas características internas dos componentes. As medidas baseiam-se na coesão [10], acoplamento [11] e complexidade [12].

A partir dessas medidas, o engenheiro poderá julgar, no nível de componente, a qualidade do produto.

5.1.3. Desenvolvimento

A teoria de Halstead [13] propõe a ciência do *software*, que são as primeiras leis analíticas para os *softwares*.

Essa ciência do *software* associa leis quantitativas para desenvolver programas usando um conjunto de medidas primitivas que podem ser derivadas depois que o código é gerado ou estimado uma vez que o projeto estiver completo.

As medidas primitivas são usadas para calcular o tamanho do programa, o volume potencial dos algoritmos e o volume atual, nível do programa em relação à complexidade do programa, o nível da linguagem e outros esforços de desenvolvimento como tempo ou falhas.

5.1.4. Teste [14]

As métricas para testes são focadas no processo de teste e, os testadores de software deverão utilizar as métricas calculadas nos processos de análise, projeto e desenvolvimento a fim de guiar as atividades de projeto e execução dos testes.

Métricas de análise ajudam os testadores de *software* a estimarem, a partir de base histórica de dados, a quantidade de erros que devem esperar encontrar durante o teste.

As métricas para análise arquitetural do software irão auxiliar o testador a prever a complexidade em integrar o sistema ao ambiente de trabalho e os direcionam a executar testes mais exaustivos nos componentes que possuem maior complexidade.

A partir desse conjunto de métricas, será possível que a equipe de testes verifique o quanto já progrediu na atividade para cada projeto, já que deverão avaliar todos os requisitos de cada projeto, tendo a medição da quantidade de requisitos, poderão saber quanto trabalho ainda falta antes de terminarem o projeto.

5.1.5. Manutenção

Todas as métricas citadas até agora estarão presentes nas métricas das atividades de manutenção, no entanto, o *Institute of Electrical and Electronics Engineers* (IEEE), na norma 982.1-1998 [15] sugere um índice da maturidade do software, que será medido pela estabilidade do produto.

Para medir essa estabilidade, deve-se medir o número de módulos da versão, a quantidade de módulos alterados para a versão, quantidade de módulos adicionados à versão e a quantidade de módulos removidos para a versão.

Ao calcular a maturidade, a partir da fórmula sugerida pela norma do IEEE e com a análise das versões anteriores do software, pode-se estimar a estabilidade do produto durante o uso.

A partir dessa quantidade de métricas disponíveis, é possível verificar que a gerência do projeto possui diversos controles para o processo de construção do produto a ser entregue ao cliente. Ao utilizar corretamente esses recursos, o gerente de projeto se torna pró-ativo para mudar o rumo das atividades em andamento e cumprir os prazos e custos do projeto em questão.

O problema que prejudica o uso correto e completo das métricas das atividades do projeto é a falta de disciplina em manter o processo de construção de software da forma correta e como foi descrito. Seguindo de maneira livre, o gerente de projeto estará medindo os indicadores pelas atividades. Como são coletados manualmente pelo operador do programa, os dados não poderão ser confiáveis, já que métricas não confiáveis não melhoram o processo.

5.2. Análise por Pontos de Função

A análise por Pontos de Função utiliza uma medida de quantidade de funcionalidades distribuídas pelo *software* como uma unidade. Como a quantidade de funcionalidades não é possível de ser medida diretamente, outras contagens são feitas.

A análise por pontos de função foi proposta por Albrecht em 1979 [16] que sugeriu uma medida chamada ponto de função. Depois, a métrica foi evoluída por Carper Jones [17] e pelo *International Function Point Users Group* (IFPUG) [18].

Segundo o *Counting Practice Manual* (CPM), manual mantido pelo IFPUG, a contagem de pontos por função pode ser feita de três maneiras:

- Contagem de pontos de função de projetos de desenvolvimento;
- Contagem de pontos de função de projetos de melhoria;
- Contagem de pontos de função de aplicações em produção.

Para começar a contagem dos pontos de função, é necessário determinar as fronteiras do *software* com o usuário e com o ambiente em que está integrado. Ao determinar as fronteiras do *software*, determinam-se quais serão os pontos a serem contados.

A contagem de pontos de função não ajustada deverá ser feita depois que toda a fronteira do *software* tiver sido determinada. A contagem dos pontos de função não ajustados é a contagem das funcionalidades a serem efetivamente entregues ao usuário.

Essas funcionalidades podem ser de dois tipos: Funções de dados e Funções transacionais. As funções de dados atendem aos requisitos de dados do usuário e do sistema em que o *software* estará ambientado. As funções transacionais atendem aos requisitos de processamento dos dados para o *software*.

Subdividindo as funções de dados, tem-se:

- Arquivos Lógicos Internos (ALI): Agrupamento lógico de dados que pode ser um arquivo ou uma parte de um banco de dados do sistema. Podem ser ALI: arquivos de

configuração, tabelas de dados para o *software*, mensagens de auxílio ou erro para o usuário;

- Arquivos de Interface Externa (AIE): Agrupamento lógico de dados que servem para controlar ou suprir dados a outros *softwares*. Podem ser AIE: tabela de dados, mensagens de auxílio ou erro para outros *softwares*.

Subdividindo as funções transacionais, tem-se:

- Entradas externas (EE): Dados vindo de fora da fronteira da aplicação. Podem ser EE: entrada on-line de usuário ou processamento de arquivos batch;
- Saídas externas (SE): Dados enviados para fora da fronteira da aplicação. Podem ser SE: Relatórios, relatórios on-line, envio de dados para outra aplicação;
- Consultas externas (CE): Saída de dados provenientes de um processamento do *software* que tiver sido disparado por uma entrada externa. Podem ser CE: Recuperação de dados, consultas, mensagens de ajuda.

Depois que a quantidade de funcionalidades a ser entregue ao usuário for definida, o nível de complexidade de cada funcionalidade deverá ser definida entre os níveis baixa, média e alta.

Para determinar o nível de complexidade de cada funcionalidade de dados, deve-se determinar a quantidade de tipos de elementos de dados (TED) e tipos de elementos de registros (TER) associados aos ALI e AIE.

Um TED é um campo de dados de um arquivo lógico interno ou de interface externa, enquanto um TER é um subgrupo de dados reconhecidos pelo usuário dentro de um arquivo lógico interno ou de interface externa.

Para que o nível de complexidade das funcionalidades transacionais seja determinado, é necessário determinar a quantidade de tipos de arquivos referenciados (TAR) e tipos de elementos de dados (TED).

Um TAR é um ALI lido ou mantido por uma função ou então, um AIE lido pelo *software*. Um TED é um campo reconhecido pelo usuário e tratado pela função transacional.

Para definir o nível de complexidade de cada funcionalidade de dados do *software*, deve-se relacionar a quantidade de TER e TED dentro desta funcionalidade de acordo com a tabela:

Tabela 1 - Nível de complexidade para arquivos lógicos internos ou de interface externa

Quantidade de TER	Quantidade de TED		
	1 a 19	20 a 50	Acima de 51
1	Baixa	Baixa	Média
2 a 5	Baixa	Média	Alta
Acima de 5	Média	Alta	Alta

Para definir o nível de complexidade de cada funcionalidade transacional do *software*, deve-se relacionar a quantidade de TAR e TED para cada funcionalidade de acordo com as tabelas abaixo:

Tabela 2 - Nível de complexidade para entradas externas

Quantidade de TAR	Quantidade de TED		
	1 a 4	5 a 15	Acima de 15
0 a 1	Baixa	Baixa	Média
2	Baixa	Média	Alta
Acima de 2	Média	Alta	Alta

Tabela 3 - Nível de complexidade para saídas externas

Quantidade de TAR	Quantidade de TED		
	1 a 5	6 a 19	Acima de 19
0 a 1	Baixa	Baixa	Média
2 a 3	Baixa	Média	Alta
Acima de 3	Média	Alta	Alta

Tabela 4 - Nível de complexidade para consultas externas

Entradas			
Quantidade de TAR	Quantidade de TED		
	1 a 4	5 a 15	Acima de 15
0 a 1	Baixa	Baixa	Média
2	Baixa	Média	Alta
Acima de 2	Média	Alta	Alta
Saídas			
Quantidade de TAR	Quantidade de TED		
	1 a 5	6 a 19	Acima de 19
0 a 1	Baixa	Baixa	Média
2 a 3	Baixa	Média	Alta
Acima de 3	Média	Alta	Alta

Depois que a contagem das funcionalidades é feita e o nível de complexidade de cada funcionalidade é determinado, deve-se preencher a tabela 5 para obter a somatória por Pontos de Função não ajustados (SPFNA).

Tabela 5 - Cálculo da quantidade de pontos por função não ajustados

Função	Quantidade	Complexidade	Peso	Resultado
Arquivos Internos		Baixa	3	
		Média	4	
		Alta	6	
Total				
Arquivos de Interface Externa		Baixa	4	
		Média	5	
		Alta	7	
Total				
Entradas Externas		Baixa	7	
		Média	10	
		Alta	15	
Total				
Saídas Externas		Baixa	5	
		Média	7	
		Alta	10	
Total				
Consultas Externas		Baixa	3	
		Média	4	
		Alta	6	
Total				
SPFNA				

Para calcular o fator de ajuste para os pontos de função do *software*, é necessário atribuir uma pontuação de 0 (não aplicável) a 5 (absolutamente essencial) para as seguintes perguntas:

1. O sistema requer *backup* e *restore* confiável?
2. Comunicação de dados é requerida?
3. Existe função de processamento distribuído?
4. O desempenho é crítico?
5. O sistema será executado em um ambiente operacional existente e altamente utilizado?
6. O sistema requer entrada de dados on-line?
7. A entrada de dados on-line requer que a transação de entrada seja construída sobre várias telas ou operações?
8. Os arquivos principais são atualizados on-line?
9. As entradas, saídas, consultas e arquivos são complexos?
10. O processamento interno é complexo?
11. O código é projetado para ser reusado?
12. O projeto inclui conversão e instalação?
13. O sistema é projetado para instalações múltiplas entre diferentes organizações?
14. A aplicação é projetada para facilitar as mudanças e o fácil uso pelo usuário?

Depois que o nível de influência de cada critério for definido, soma-se todos os níveis de influência e calcula-se o fator de ajuste de acordo com a seguinte fórmula, onde FA é o fator de ajuste e SNI é a somatória dos níveis de influência:

$$FA = 0,65 + 0,01 * SNI \quad (1)$$

Agora temos a quantidade de pontos por função não ajustada e o fator de ajuste, portanto, para termos o total de pontos por função líquida para o *software*, devemos seguir a seguinte fórmula, onde PF significa pontos de função, SPFNA é a somatória dos pontos de função não ajustados e FA é o fator de ajuste:

$$PF = SPFNA * FA \quad (2)$$

O problema da análise por Pontos de Função é não ser suficiente para que o gerente calcule as métricas do projeto. Para que isso seja possível, o gerente deverá ter uma base histórica de projetos anteriores e relacionar os dados com a quantidade de Pontos de Função.

5.3. *Constructive Cost Model (COCOMO)*

Ponto de função não permite aos gerentes de projetos calcularem as métricas do projeto, sendo necessário ainda uma base histórica de projetos, de forma que possam comparar os tamanhos dos projetos e relacionar os dados.

A partir desse problema, Barry Boehm [19] desenvolveu o COCOMO, para que seja possível medir o custo, esforço, tamanho da equipe e prazo do projeto.

5.3.1. *Delivery Source Instructions (DSI)*

O COCOMO é baseado na unidade de medida *Delivery Source Instructions (DSI)*. A partir do tamanho dos projetos em DSI, Boehm desenvolveu uma tabela de estimativas a partir de 63 projetos nas áreas de: negócio, controle, ciência, suporte e sistema operacional.

O problema que surge está em relacionar pontos de função com DSI. Para isso, Caper Jones e Albrecht [17], na IBM, normalizaram o nível das linguagens a partir de número de instruções equivalentes em Assembler para criar um ponto de função.

O relacionamento entre ponto de função e DSI para algumas linguagens está relacionado na tabela 6.

Tabela 6 - Quantidade de instruções por ponto de função

Linguagem	Quantidade de Instruções
Assembler	320
C	180
Banco de dados	40
Orientação a objetos	29
Quinta geração	4

Para montar as estimativas com o COCOMO a partir do total de pontos de função ajustados, multiplica-se esse valor pela quantidade de instruções, e têm-se o total de DSI produzidos. Caso o projeto utilize mais de uma linguagem, basta somar os totais de DSI de cada linguagem e ao final têm-se o total de DSI do projeto.

5.3.2. Modelos

O COCOMO é dividido em três modelos, Básico, Intermediário e Avançado.

O modelo Básico é aplicável aos projetos de tamanho pequeno e médio. O problema desse nível é que desconsidera detalhes técnicos como restrições de *hardware*, experiência da equipe ou o uso de técnicas e ferramentas modernas.

O modelo Intermediário trabalha como o modelo Básico, com a diferença de considerar os fatores não abordados por aquele modelo.

O modelo Avançado é capaz de estimar em nível de módulo, subsistema e sistema, individualizando os atributos do projeto a cada fase.

Para o objetivo deste projeto, estaremos trabalhando com o modelo Intermediário, que realizará estimativas no nível necessário para atender às necessidades da maioria dos projetos.

5.3.3. Tipos Fundamentais

Depois que o modelo do COCOMO tiver sido escolhido, devemos definir qual é o tipo do projeto. Para o COCOMO, existem três tipos fundamentais. Orgânico, Difuso e Restrito.

Para projetos do tipo Orgânico, a equipe toda deverá ter conhecimento sobre o problema e saber como o sistema trabalha para os objetivos da empresa. Nesse tipo de projeto, todas as pessoas da equipe cooperam desde o início do projeto, não existe grande fluxo de

comunicação e todos sabem como é a divisão do trabalho dentro do projeto. Projetos desse tipo têm ambientes de desenvolvimento estável, algoritmos simples e, em média, possuem 50 KDSI.

Projetos Difusos são projetos intermediários entre o Orgânico e o Restrito, portanto, a equipe é mesclada em nível de conhecimento do problema e tecnologias. Projetos desse tipo possuem até 300 KDSI.

Projetos Restritos são realizados dentro de níveis de restrições muito grandes. Algumas dessas restrições podem ser: *hardware*, *software*, procedimentos e regras. Portanto, requisitos de projetos do tipo Restrito dificilmente são alterados. Por fim, projetos Restritos têm alto custo com testes e validações, o que faz com que o projeto tenha alto custo e exija prazos longos para desenvolvimento.

5.3.4. Estimando com COCOMO

De posse da quantidade de DSI do projeto, o gerente poderá medir o esforço, prazo e quantidade de pessoal para o projeto.

Para que o gerente de projeto meça o esforço a ser gasto com o projeto, deverá utilizar uma das fórmulas a seguir, de acordo com o modelo (básico ou intermediário) e o tipo fundamental (orgânico, difuso ou restrito) escolhido. Para o cálculo do esforço, o gerente de projeto deverá ter a quantidade total de DSI a ser produzido pelo projeto.

Tabela 7 - Cálculo do esforço

	Básico	Intermediário
Orgânico	$H/M = 2,4 * KDSI^{1,05}$	$H/M = 3,2 * KDSI^{1,05}$
Difuso	$H/M = 3,0 * KDSI^{1,12}$	$H/M = 3,0 * KDSI^{1,12}$
Restrito	$H/M = 3,6 * KDSI^{1,20}$	$H/M = 2,8 * KDSI^{1,20}$

Para que o gerente de projeto meça o esforço pelo tipo Intermediário, os quinze fatores de ajuste deverão ser levados em consideração. Esses fatores de ajuste são divididos entre quatro atributos. Os fatores de ajuste são:

- Produto
 - RELY – Confiabilidade;
 - DATA – Tamanho da base de dados por Instruções fonte;
 - CPLX – Complexidade.

- Computacional
 - TIME – Disponibilidade do tempo de máquina;
 - STOR – Restrições quanto ao uso de memória principal;
 - VIRT – Mudanças no ambiente do *software*;
 - TURN – Tempo de resposta.
- Equipe
 - ACAP – Capacidade dos analistas;
 - AEXP – Experiência dos analistas;
 - PCAP – Capacidade dos programadores;
 - VEXP – Experiência no ambiente de *hardware*;
 - LEXP – Experiência na linguagem de programação.
- Projeto
 - MODP – Técnicas modernas de programação;
 - TOOL – Uso de ferramentas;
 - SCED – Prazo requerido pra o desenvolvimento.

Conhecendo os fatores de ajuste, o gerente de projeto deverá então classificar o nível de influência de cada um sobre o projeto em questão. Para a definição do nível de influência, o gerente de projeto deverá seguir a tabela 8.

Tabela 8 - Determinação do nível de cada atributo

Atributo	Muito Baixo	Baixo	Nominal	Alto	Muito Alto	Altíssimo
RELY	Sem perda	Baixa	Moderada	Perdas financeiras	Risco à vida	
DATA		DT < 10	$10 \leq DT < 100$	$100 \leq DT < 1K$	$DT \geq 1K$	
TIME			$\leq 50\%$	70%	85%	95%
STOR			$\leq 50\%$	70%	85%	95%
VIRT		12 meses	6 meses	2 meses	2 semanas	
TURN		Imediato	< 4 horas	4-12 horas	> 12 horas	
ACAP	15%	35%	55%	75%	90%	
AEXP	≤ 4 meses	1 ano	3 anos	6 anos	12 anos	
PCAP	15%	35%	55%	75%	90%	
VEXP	≤ 1 mês	4 meses	1 ano	3 anos		
LEXP	≤ 1 mês	4 meses	1 ano	3 anos		
MODP	Não utiliza	Pouco	Algum uso	Generalizado	Rotineiro	
TOOL	Micro	Mini	Mainframe	Programação e teste	Análise, Projeto e Gerência	
SCED	75%	85%	100%	130%	160%	

Para determinar o nível de influência para o atributo complexidade (CPLX), o gerente de projeto deverá considerar o nível de influência com a maior quantidade de atributos de acordo com a tabela 9, onde: QA é *Quality Assurance* (Garantia da Qualidade); CM é *Configuration Management* (Gerência de configuração); PDR é *Product Design Review* (Revisão de projeto do projeto); IV & V é *Independent Verification & Validation* (Verificação independente e Validação).

Tabela 9 - Determinação do nível para o atributo complexidade

	Muito baixo	Baixo	Nominal	Alto	Muito Alto	Extra Alto
Controle de operações	Código simples; Não usa Programação estruturada	Operadores com uso de programação estruturada	Uso de tabelas de decisão; Algum controle entre módulos.	Controle entre módulos; controle de filas e pilha; grande nº de operadores.	Código reentrante e recursivo; manuseio de interrupções.	Controle de microcódigo ; programação dinâmica de recursos.
Operações computacionais	Avaliação de expressões simples	Avaliação de expressões moderadas	Uso de rotinas matemáticas e estatísticas	Equações diferenciais	Equações diferenciais parciais	Análise de dados stocásticos
Operações dependentes de máquina	Instruções simples de leitura e gravação	Tratamento de I/O	Processamento de I/O; escolha do periférico; detecção de erro	Operações a nível físico de I/O	Rotinas de diagnóstico; comunicação	Operações em micro-programação; controle de tempos.
Operações de gerência de dados	Arrays simples na memória principal	Arquivos simples; sem arquivos intermediários	Entradas para múltiplos arquivos	Reestruturação complexa de dados ao nível de registro	Rotina parametrizada para dados; otimização de pesquisa	Estrutura relacional
Projeto de requisitos e do produto	Pouco detalhe; pouca verificação sem QA, CM	QA e CM básicos; planos de testes	V & V	Verificação detalhada; padrões de QA, CM, PDR, planos de teste.	Planos de teste muito detalhados	
Projeto detalhado	Informação de projeto mínima; mínimo QA e CM	Detalhe moderado; QA e CM básicos, plano de teste	V & V	Verificação detalhada; padrões de QA e CM; plano de teste.	Inspeções detalhadas; planos de testes muito detalhados; I V & V	
Teste de código e Unidade	Sem teste; mínimo QA e CM	Procedimento de teste mínimo; QA e CM básicos	V & V	Procedimento de teste detalhado; QA e CM	Inspeções detalhadas de código; I V & V	
Integração e teste	Muitos requisitos sem teste; mínimo QA e CM	Requisitos frequentemente e sem testes; QA e CM básico	V & V	Teste detalhado; QA e CM; documentação	Teste muito detalhados; QA e CM; I V & V	

Depois de definir o nível de influência de cada fator de ajuste, o gerente de projeto deverá seguir a tabela 10 para definir o fator de ajuste total para o esforço do projeto.

Tabela 10 – Fatores de ajuste por atributo do projeto

Atributos	Pesos					
	Muito Baixo	Baixo	Nominal	Alto	Muito Alto	Extra Alto
RELY	.75	.88	1.00	1.15	1.40	
DATA		.94	1.00	1.08	1.16	
CPLX	.70	.85	1.00	1.15	1.30	1.65
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		.87	1.00	1.15	1.30	
TURN		.87	1.00	1.07	1.15	
ACAP	1.46	1.19	1.00	.86	.71	
AEXP	1.29	1.13	1.00	.91	.82	
PCAP	1.42	1.17	1.00	.86	.70	
VEXP	1.21	1.10	1.00	.90		
LEXP	1.14	1.07	1.00	.95		
MODP	1.24	1.10	1.00	.91	.82	
TOOL	1.24	1.10	1.00	.91	.83	
SCED	1.23	1.08	1.00	1.04	1.10	

Seguindo o tipo Intermediário, o gerente de projeto deverá multiplicar a estimativa de esforço por cada fator de ajuste para encontrar o esforço total do projeto.

Depois que o esforço do projeto for calculado, pelo tipo Básico ou Intermediário, o gerente de projeto poderá medir o prazo do projeto, de acordo com a tabela 11, seguindo o mesmo modelo utilizado para o cálculo do esforço.

Tabela 11 - Cálculo do prazo

Orgânico	$P = 2,5 * H/M^{0,38}$
Difuso	$P = 2,5 * H/M^{0,35}$
Restrito	$P = 2,5 * H/M^{0,32}$

Depois que o esforço e o prazo forem medidos, o gerente de projeto medirá a quantidade de pessoas que farão parte da equipe do projeto através da equação 2.

$$\text{Equipe} = (H/M) / (P) \text{ [pessoas]} \quad (3)$$

Agora, o gerente de projeto possuirá as medidas para esforço, prazo e tamanho da equipe segundo as técnicas do COCOMO.

O uso do COCOMO pode ser exemplificado da seguinte forma.

5.4. Estimativas

Depois que o tamanho, esforço, prazo e quantidade de pessoal do projeto forem medidos, o gerente de projeto desejará estimar em qual parte do projeto os recursos serão gastos.

Baseado na distribuição de Rayleigh, Boehm criou a tabela 12 que estabelece uma distribuição percentual do esforço e prazo consumidos durante as fases do projeto de *software*.

As fases para um projeto de *software*, segundo o COCOMO, são: Planos e Requisitos, Projeto, Programação e Integração e Teste.

Tabela 12 - Distribuição percentual do esforço e prazo pelas fases do projeto

		Tamanho (KDSI)				
Modo	Fase	2	8	32	128	512
Esforço						
Orgânico	Planos e Requisitos	6	6	6	6	
	Projeto	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	
Difuso	Planos e Requisitos	7	7	7	7	7
	Projeto	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
Restrito	Planos e Requisitos	8	8	8	8	8
	Projeto	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
Prazo						
Orgânico	Planos e Requisitos	10	11	12	13	
	Projeto	19	19	19	19	
	Programação	63	59	55	51	
	Integração e teste	18	22	26	30	
Difuso	Planos e Requisitos	16	18	20	22	24
	Projeto	24	25	26	27	28
	Programação	56	52	48	44	40
	Integração e teste	20	23	26	29	32
Restrito	Planos e Requisitos	24	28	32	36	40
	Projeto	30	32	34	36	38
	Programação	48	44	40	36	38
	Integração e teste	22	24	26	28	30

Para utilizar essa distribuição, o gerente de projeto deverá multiplicar o esforço em H/M pela porcentagem associada à fase do projeto que escolher.

5.5. Medindo um projeto de *software*

Supondo que temos um projeto a ser realizado em C com um total de 100 pontos de função. O total de DSI para o projeto seria:

$$DSI = 180 * 100 = 18.000 \quad (4)$$

O próximo passo agora é calcular o esforço bruto do projeto, sendo ele difuso.

$$H/M = 3 * 18^{1.12} = 76.39 \quad (5)$$

A complexidade do projeto sendo de nível Alto para todos os atributos de ajuste, o esforço total líquido é calculado.

$$H/M = 76.39 * 1.02 = 80.98 \quad (6)$$

A partir do esforço total do projeto, o prazo deverá ser calculado.

$$P = 2.5 * 80.98^{0.35} = 11.64 \quad (7)$$

Por fim, o tamanho da equipe a participar do projeto é calculado.

$$E = 80.98 / 11.64 = 6.95 \quad (8)$$

Para estimar o consumo de esforço e prazo em cada fase do projeto, usamos as porcentagens da tabela 12.

Tabela 13 - Equações para estimativa de esforço e prazo

	Esforço		Prazo	
	%	Total	%	Total
Planos e Requisitos	7	5.67	18	2.1
Projeto	17	13.77	25	2.91
Programação	61	49.40	52	6.05
Projeto Detalhado	26	21,05		
Codificação	35	28,34		
Integração e teste	22	17.82	23	2.68

Se tivéssemos o custo mensal para o projeto, agora, o valor total para o desenvolvimento do projeto poderia ser calculado. Bastaria multiplicar o esforço de cada fase pelo custo mensal da atividade e, então, multiplicar pelo prazo estimado. Ao final, totalizar os custos estimados para ter o custo total do projeto.

Depois que todas essas medições e estimativas são feitas, é possível acompanhar o cronograma e o consumo de recursos, antecipando possíveis problemas com o custo e o prazo do projeto.

O objetivo para este projeto é criar uma ferramenta que realize todos os cálculos necessários para essas medições e guarde essas informações como base histórica, de forma que futuramente, possa confrontar os cálculos teóricos com os resultados práticos da organização. Fazendo isso, a cada projeto, as estimativas serão mais precisas.

CAPÍTULO VI

Ferramenta para gerenciamento de projetos baseado em métricas

6.1. Visão geral

O problema enfrentado pelos gerentes de projeto atualmente é conseguir medir o tamanho do produto a ser desenvolvido por sua equipe. Sem conhecer o tamanho do produto, é muito difícil para que o gerente consiga estimar o prazo, custo, consumo de recursos e cronograma de atividades.

De acordo com esse problema, esse projeto visa criar uma ferramenta que sirva para que a equipe de análise alimente os dados do tamanho do produto a ser desenvolvido, de acordo com a técnica de análise por pontos de função e essa ferramenta deverá fazer os cálculos necessários para fazer as estimativas de prazo, custo, esforço e tamanho da equipe utilizando a análise por pontos de função e o COCOMO.

6.2. Necessidades

O problema que este projeto pretende eliminar é a complexidade em realizar todos os cálculos necessários para as técnicas de análise por pontos de função e COCOMO.

Portanto, a ferramenta deverá possuir uma entrada de dados para o cálculo automático do tamanho do *software* e posterior saída com os resultados desses cálculos.

6.3. Estrutura

A estrutura da ferramenta será baseada na entidade projeto. Dessa forma, todos os cálculos ficarão associados ao projeto correspondente.

Para resolver o problema do gerente de projeto não conseguir acompanhar a tendência de crescimento do *software*, a ferramenta deverá armazenar a data e o tamanho a cada medição feita para o projeto.

Como a ferramenta deverá realizar os cálculos, é necessário que a ferramenta possua tabelas com as informações sobre complexidade, linguagem, modelos, tipos fundamentais,

esforço, prazo, fatores de ajuste e notas para pontos de função e fatores de ajuste e pesos para COCOMO.

6.4. Funcionamento

O funcionamento da ferramenta se iniciará pelo cadastro do projeto e, então, a equipe de análise poderá iniciar o cadastro dos atributos do projeto (ALI, AIE, EE, SE, CE, fatores de ajuste de APF e COCOMO).

O gerente, deverá medir o software sempre que julgar necessário e, poderá visualizar as estimativas do projeto através das listagens e relatórios disponíveis no programa.

6.5. Domínio da informação

As informações necessárias para desenvolver esta ferramenta estão contidas nesse trabalho de projeto final de curso. As fórmulas dispostas no capítulo V são aquelas necessárias para a realização dos cálculos de Análise de Pontos de Função e COCOMO.

6.6. Ambiente para trabalho

6.6.1. Ambiente do administrador

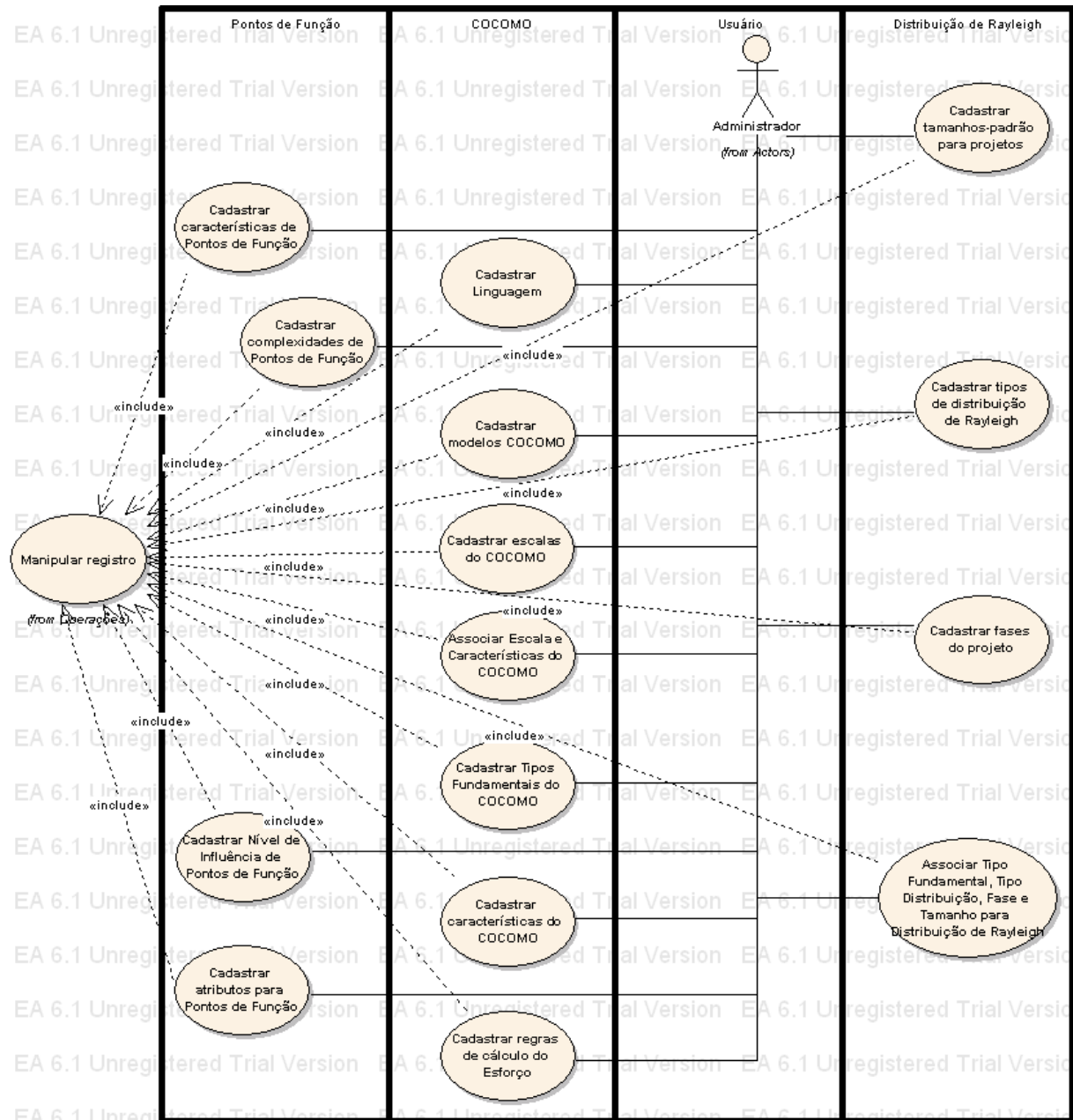


Figura 5 – Diagrama de caso de uso para funções do Administrador

6.6.2. Tarefas internas do ambiente do administrador

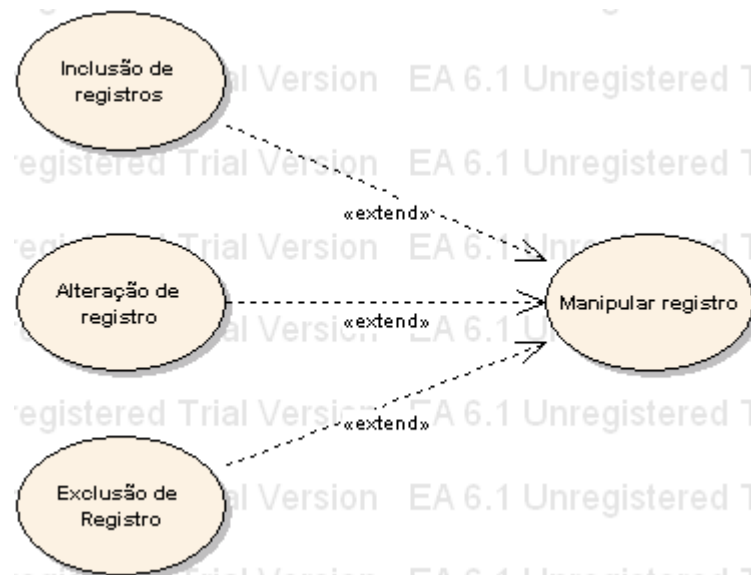


Figura 6 – Diagrama de caso de uso com as operações permitidas para os Administradores

6.6.3. Ambiente para o analista e o gerente

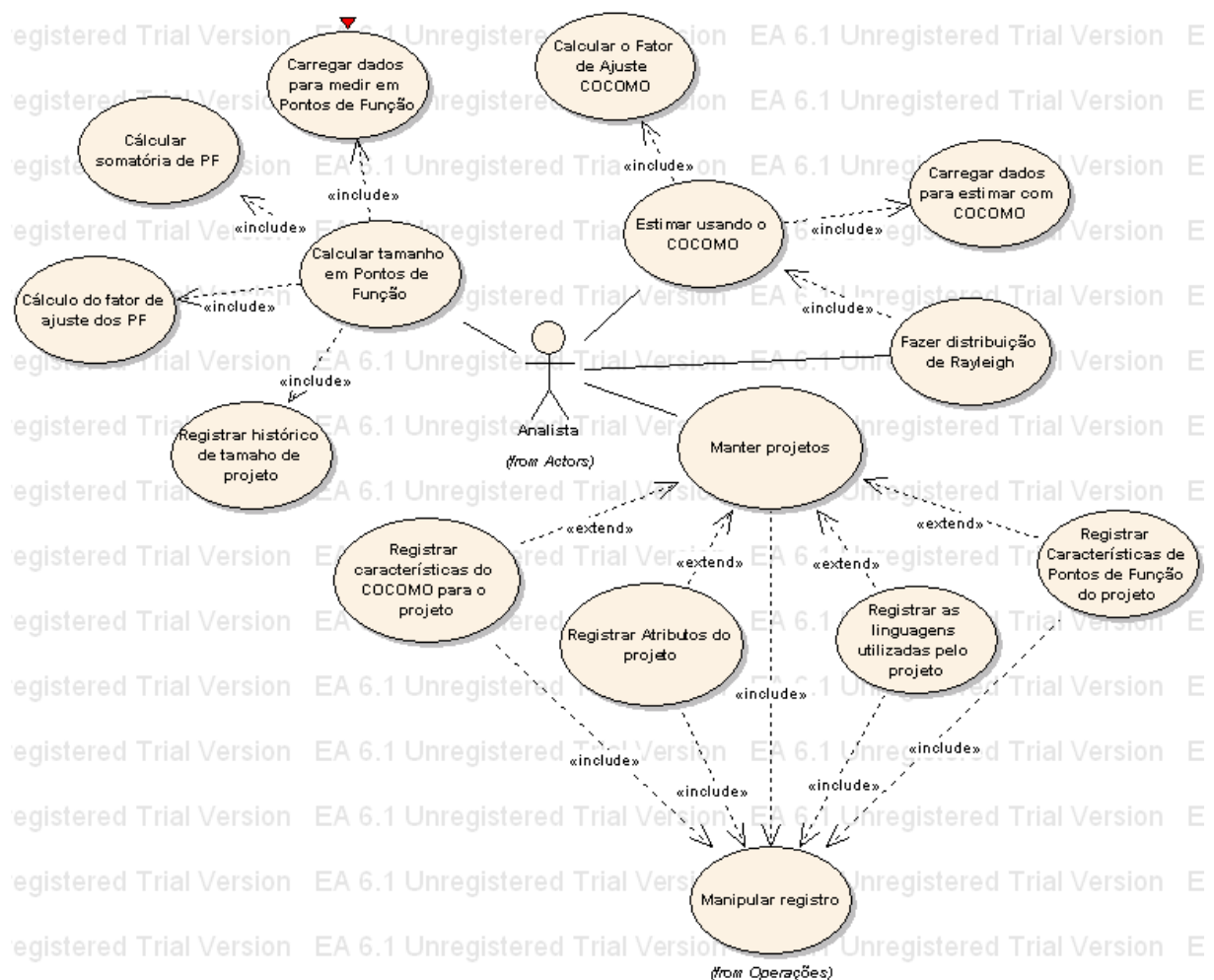


Figura 7 – Diagrama de caso de uso para as funções do analista

6.7. Requisitos

6.7.1. Requisitos para administradores

6.7.1.1. Associar Escala e Característica

Tabela 14 - Requisito F0001 (Associar Escala e Características do COCOMO)

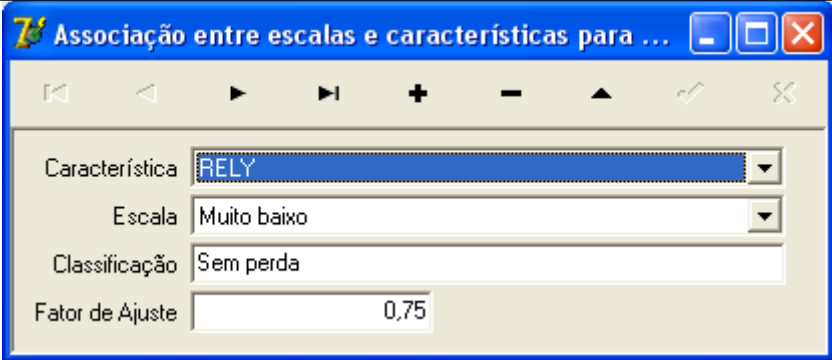
Código	F00001	Nome	Associar Escala e Características do COCOMO
Justificativa	Caso de uso responsável por manter as informações necessárias para as associações entre escala e característica de COCOMO.		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Característica, Escala, Classificação e Fator de Ajuste		
Relacionamento	F0033 – Manipular Registros		
Protótipo	 <p>Figura 8 - Protótipo da tela de associação para fator de ajuste do COCOMO</p>		
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 9 - Diagrama de seqüência para requisito F0001

6.7.1.2. Associação para Rayleigh

Tabela 15 - Requisito F0002(Associação para Rayleigh)

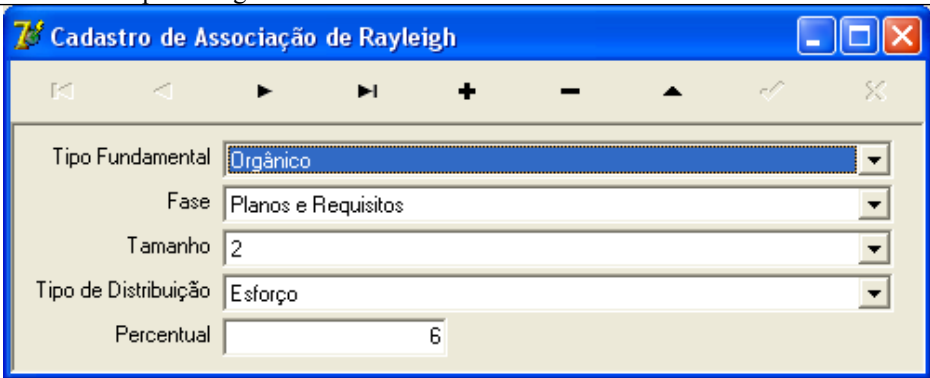
Código	F00002	Nome	Associação para Rayleigh
Justificativa	Caso de uso responsável por manter as informações relacionadas às associações entre Tipo de Distribuição, Fase e Tamanho de projeto para determinar a porcentagem a ser consumida do tipo de distribuição para cada fase do projeto de acordo com o tamanho do projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Tipo Fundamental, Fase, Tamanho, Tipo de Distribuição e Percentual		
Relacionamento	F0033 – Manipular Registros		
Protótipo	 <p>Figura 10 - Protótipo da tela de Associação de Rayleigh</p>		
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 11 - Diagrama de seqüência para o requisito F0002

6.7.1.3. Cadastrar Linguagem

Tabela 16 - Requisito F0003 (Cadastrar Linguagem)


Código	F00003	Nome	Cadastrar Linguagem
Justificativa	Caso de uso responsável por cadastrar as linguagens disponíveis para realizar a conversão da medida Pontos por Função para DSI		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Nome, Equivalente em PF		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 12 - Protótipo da tela de Cadastro de linguagens**Figura 13 - Diagrama de sequência para o requisito F0003**

6.7.1.4. Cadastrar Nível de Influência

Tabela 17 - Requisito F0004 (Cadastrar Níveis de Influência)

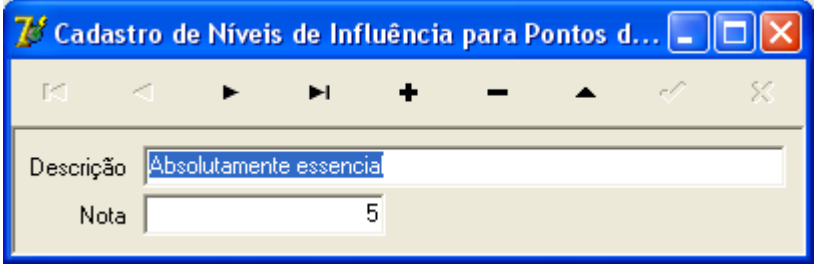
Código	F00004	Nome	Cadastrar níveis de influência
Justificativa	Caso de uso responsável por manter o cadastro de níveis de influência a ser usado para classificar cada critério de avaliação para o cálculo de Pontos de Função		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição, Nota		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados.		

Figura 14 - Protótipo da tela de cadastro de níveis de influência**Figura 15 - Diagrama de sequência para o requisito F0004**

6.7.1.5. Cadastrar Tipos Fundamentais

Tabela 18 - Requisito F0005 (Cadastrar Tipos Fundamentais)

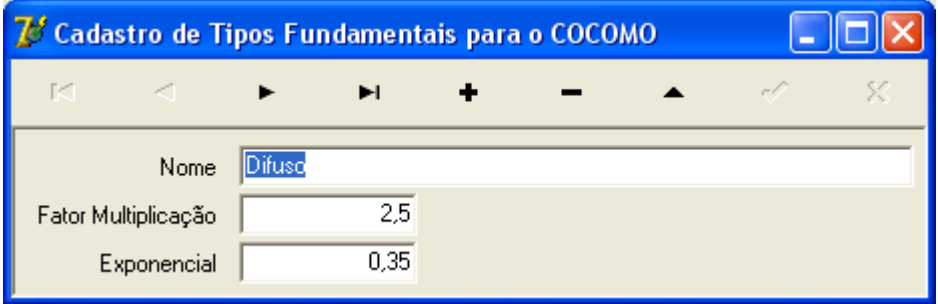
Código	F00005	Nome	Cadastrar tipos fundamentais
Justificativa	Caso de uso responsável por manter as informações sobre os tipos fundamentais do COCOMO disponíveis para os projetos		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Nome, Cálculo do Prazo (Fator Multiplicação, Exponencial)		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 16 - Protótipo da tela de cadastro de tipos fundamentais**Figura 17 - Diagrama de sequência para o requisito F0005**

6.7.1.6. Cadastrar atributos

Tabela 19 - Requisito F0006 (Cadastrar atributos)

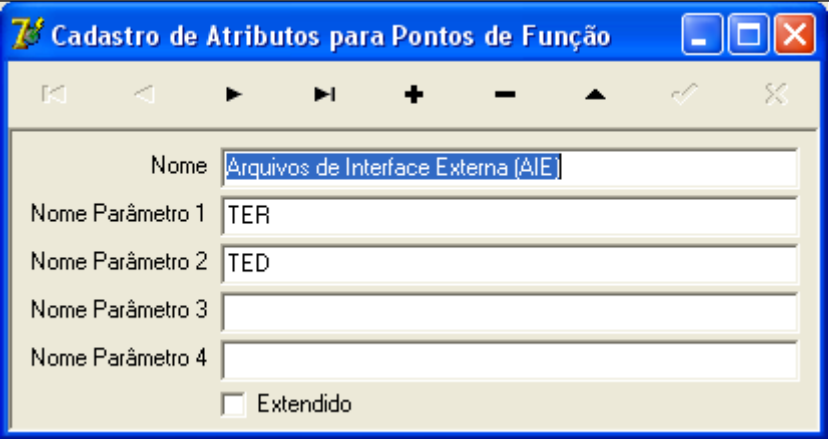
Código	F00006	Nome	Cadastrar atributos
Justificativa	Caso de uso responsável por manter as informações sobre os atributos de um projeto medido com Pontos de Função		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Nome, Nome Parâmetro 1, Nome Parâmetro 2, Nome Parâmetro 3, Nome Parâmetro 4, Estendido		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 18 - Protótipo da tela de cadastro de atributos de pontos de função

Figura 19 - Diagrama de sequência para o requisito F0006

6.7.1.7. Cadastrar características de Pontos de Função

Tabela 20 - Requisito F0007 (Cadastrar características de Pontos de Função)

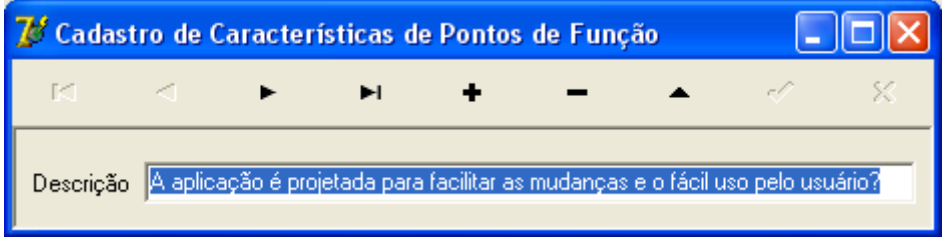
Código	F00007	Nome	Cadastrar características de Pontos de Função
Justificativa	Caso de uso responsável por manter as informações sobre as características do projeto para o ajuste dos pontos de função do projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	Figura 20 - Protótipo da tela de cadastro de características de pontos de função 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados.		

Figura 21 - Diagrama de sequência para o requisito F0007

6.7.1.8. Cadastrar características de COCOMO

Tabela 21 - Requisito F0008 (Cadastrar características de COCOMO)

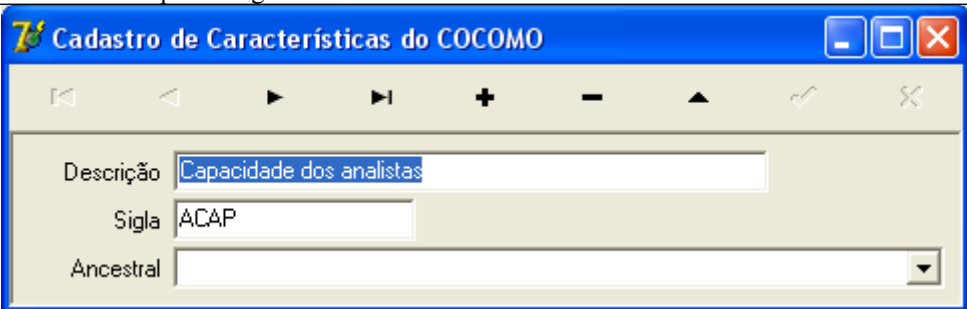
Código	F00008	Nome	Cadastrar características de COCOMO
Justificativa	Caso de uso responsável por manter as informações relacionadas às características de projeto para cálculo do ajuste para estimativas pelo COCOMO		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição, Sigla e Ancestral		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<p>Figura 22 - Protótipo da tela de cadastro de características do COCOMO</p> <ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 23 - Diagrama de seqüência para o requisito F0008

6.7.1.9. Cadastrar complexidades

Tabela 22 - Requisito F0009 (Cadastrar complexidades)

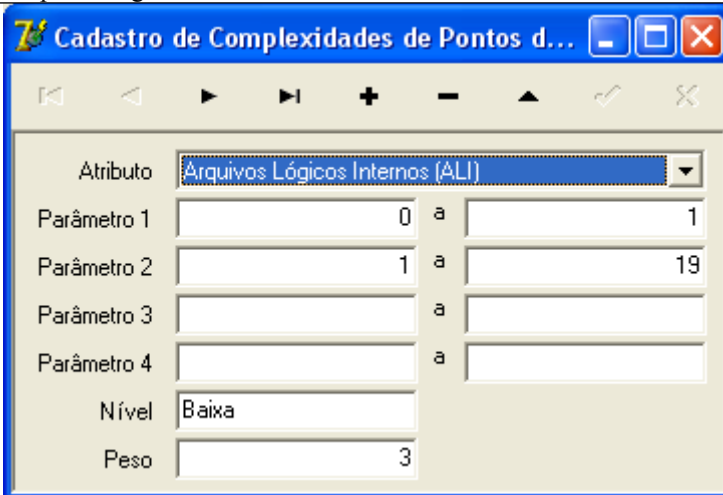
Código	F00009	Nome	Cadastrar complexidades
Justificativa	Caso de uso responsável por manter as informações sobre a complexidade para cada atributo de Ponto de Função		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Atributos, Valores mínimo e máximo para Parâmetro 1, Parâmetro 2, Parâmetro 3 e Parâmetro 4, Nível e Peso		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<p>Figura 24 - Protótipo da tela de cadastro de complexidades de pontos de função</p> <ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 25 - Diagrama de seqüência para o requisito F0009

6.7.1.10. Cadastrar escalas

Tabela 23 - Requisito F0010 (Cadastrar escalas)

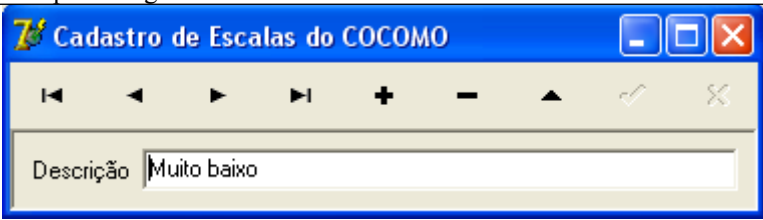
Código	F00010	Nome	Cadastrar escalas
Justificativa	Caso de uso responsável por manter as informações sobre as escalas de classificação para cada critério de ajuste do projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 26 - Protótipo da tela de cadastro de escalas do COCOMO**Figura 27 - Diagrama de seqüência para o requisito F0010**

6.7.1.11. Cadastrar fases

Tabela 24 - Requisito F0011 (Cadastrar fases)

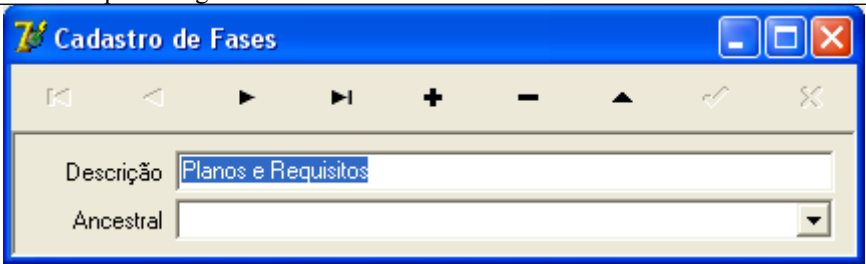
Código	F00011	Nome	Cadastrar fases
Justificativa	Caso de uso responsável por manter as informações relacionadas às fase de projeto disponíveis para realizar a distribuição de Rayleigh		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição e Ancestral		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 28 - Protótipo da tela de cadastro de fases**Figura 29 - Diagrama de seqüências do requisito F0011**

6.7.1.12. Cadastrar modelos

Tabela 25 - Requisito F0012 (Cadastrar modelo)

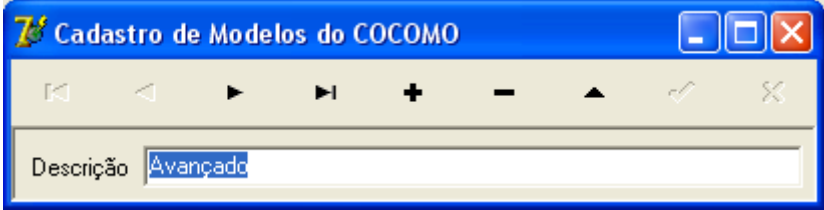
Código	F00012	Nome	Cadastrar modelo
Justificativa	Caso de uso responsável por manter as informações sobre os modelos do COCOMO disponíveis para estimar os projetos		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 30 - Protótipo da tela de cadastro de modelos do COCOMO**Figura 31 - Diagrama de sequência para o requisito F0012**

6.7.1.13. Cadastrar regras de cálculo do Esforço

Tabela 26 - Requisito F0013 (Cadastrar regra de esforço)

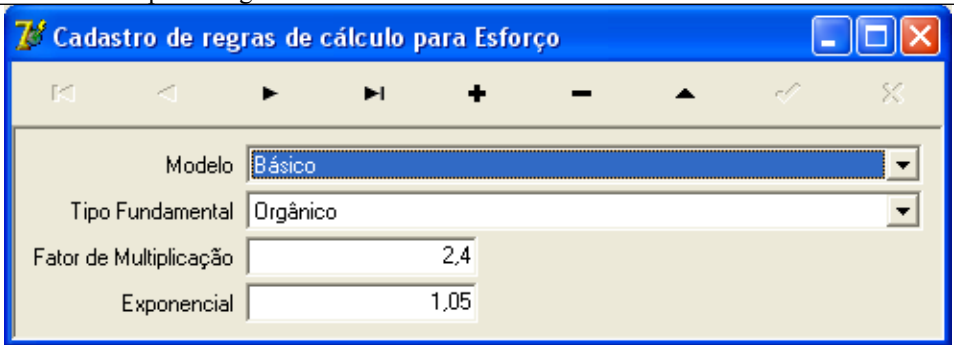
Código	F00013	Nome	Cadastrar regra de esforço
Justificativa	Caso de uso responsável por manter as informações relacionadas às regras de cálculo de estimativas de esforço pelo COCOMO.		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Modelo, Tipo Fundamental, Fator de Multiplicação e Exponencial		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados.		

Figura 32 - Protótipo da tela de cadastro da regra de esforço**Figura 33 - Diagrama de sequência para o requisito F0013**

6.7.1.14. Cadastrar tamanho-padrão

Tabela 27 - Requisito F0014 (Cadastrar tamanho-padrão)

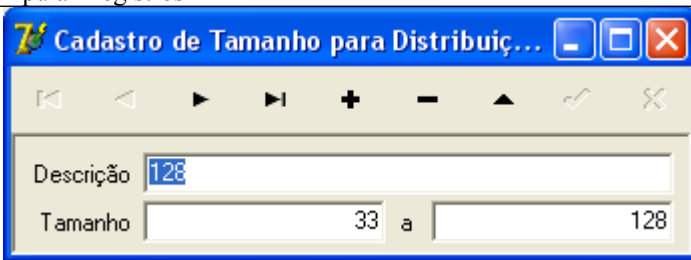
Código	F00014	Nome	Cadastrar tamanho-padrão
Justificativa	Caso de uso responsável por manter as informações relacionadas ao tamanho de projeto para estimar a distribuição do esforço e prazo segundo a Distribuição de Rayleigh		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição, Tamanho mínimo e tamanho máximo		
Relacionamento	F0033 – Manipular Registros		
Protótipo			
Validação	1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados.		

Figura 34 - Protótipo da tela de cadastro de tamanho-padrão**Figura 35 - Diagrama de sequência para o requisito F0014**

6.7.1.15. Cadastrar tipos de distribuição de Rayleigh

Tabela 28 - Requisito F0015 (Cadastrar tipos de distribuição de Rayleigh)

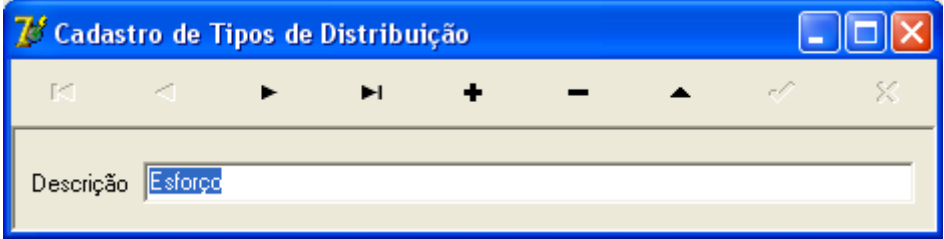
Código	F00015	Nome	Cadastrar tipos de distribuição de Rayleigh
Justificativa	Caso de uso responsável por manter as informações relacionadas aos tipos de distribuições de Rayleigh disponíveis		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros.		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Dados	Descrição		
Relacionamento	F0033 – Manipular Registros		
Protótipo	 <p>Figura 36 - Protótipo da tela de cadastro de tipos de distribuição</p>		
Validação	<ol style="list-style-type: none"> 1. Usuário deverá incluir um registro e esse deverá estar persistido no banco de dados; 2. Usuário deverá localizar o registro incluído e alterá-lo. Esta alteração deverá estar persistida no banco de dados; 3. Usuário deverá localizar o registro alterado e excluí-lo. Esta exclusão deverá estar persistida no banco de dados. 		

Figura 37 - Diagrama de sequência para o requisito F0015

6.7.2. Requisitos para analistas

6.7.2.1. Calcular o Fator de Ajuste COCOMO

Tabela 29 - Requisito F0016 (Calcular o Fator de Ajuste COCOMO)

Código	F00016	Nome	Cadastrar o Fator de Ajuste COCOMO
Justificativa	Caso de uso responsável por calcular o fator de ajuste definido para o projeto		
Escopo	O sistema deverá ser capaz de calcular o fator de ajuste para estimar pelo COCOMO		
Pré-condições	Dados do projeto e associação entre característica, escala e classificação do COCOMO		
Pós-condições	Valor do fator de ajuste para o cálculo do COCOMO		
Descrição	1 - Sistema - Multiplicar os fatores de ajuste das características simples 2 - Sistema - Verificar qual escala aparece mais para cada características compostas 3 - Sistema - Multiplicar o FA calculado no passo 1 pelo FA calculado no passo 2 4 - Sistema - Retornar o fator de ajuste calculado no passo 3		

6.7.2.2. Calcular tamanho em Pontos de Função

Tabela 30 - Requisito F0017 (Calcular tamanho em Pontos de Função)

Código	F00017	Nome	Cadastrar tamanho em Pontos de Função
Justificativa	Caso de uso responsável por executar os cálculos para medir o tamanho do projeto em Pontos de Função		
Escopo	O sistema deverá calcular o tamanho do projeto em PF baseado nos dados fornecidos para o projeto		
Pré-condições	Dados do projeto a ser medido, Complexidade e Nível de Influência		
Pós-condições	Tamanho do projeto em PF		
Descrição	1 - Sistema - Executar o caso de uso "Calcular somatória de PF" 2 - Sistema - Executar o caso de uso "Calcular o fator de ajuste para Pontos de Função" 3 - Sistema - Realizar o produto entre o SPFNA e FA 4 - Sistema - Retornar o produto resultante do passo anterior ao Usuário		
Dados			
Relacionamento	F0020 – Calcular somatória de PF F0021 – Calcular o fator de ajuste para Pontos de Função		

Figura 38 - Diagrama de seqüência para o requisito F0017

6.7.2.3. Carregar dados para estimar com COCOMO

Tabela 31 - Requisito F0018 (Carregar dados para estimar com COCOMO)

Código	F00018	Nome	Carregar dados para estimar com COCOMO
Justificativa	Caso de uso responsável por carregar os dados associados ao projeto necessários para a execução do caso de uso "Estimar usando o COCOMO"		
Escopo	Garantir que todos os dados necessários para o cálculo da estimativa pelo COCOMO estejam carregados		
Pré-condições	Dados do projeto a ser medido		
Pós-condições	Dados necessários para os cálculos estarem disponíveis		
Descrição	1 - Sistema - Carregar os dados do projeto 2 - Sistema - Carregar o valor da última medição realizada 3 - Sistema - Carregar a lista de linguagens associadas ao projeto 4 - Sistema - Carregar as informações para o cálculo do esforço do projeto 5 - Sistema - Carregar as informações para o cálculo do prazo do projeto 6 - Sistema - Carregar as características associadas ao projeto		

6.7.2.4. Carregar dados para medir em Pontos de Função

Tabela 32 - Requisito F0019 (Carregar dados para medir em Pontos de Função)

Código	F00019	Nome	Carregar dados para medir em Pontos de Função
Justificativa	Caso de uso responsável por carregar os dados associados ao projeto necessários para a execução do caso de uso "Calcular tamanho em Pontos de Função"		
Escopo	Garantir que todos os dados necessários para o cálculo do tamanho em Pontos de Função estejam carregados		
Pré-condições	Dados do projeto a ser medido		
Pós-condições	Dados necessários para os cálculos estarem disponíveis		
Descrição	1 - Sistema - Carregar os atributos associados ao projeto 2 - Sistema - Carregar a lista de características de PF		

6.7.2.5. Calcular somatória de PF

Tabela 33 - Requisito F0020 (Calcular somatória de PF)

Código	F00020	Nome	Calcular somatória de PF
Justificativa	Caso de uso responsável por realizar os cálculos para medir em PF		
Escopo	Garantir que todos os dados necessários para o cálculo do tamanho em Pontos de Função estejam carregados		
Pré-condições	- Todos os atributos associados ao projeto - Lista das complexidades de cada atributo		
Pós-condições	Tamanho do projeto em PF calculado		
Descrição	1 - Sistema - Determinar a quantidade de atributos por complexidade do projeto 2 - Sistema - Multiplicar a quantidade de atributos por complexidade pelo peso associado 3 - Sistema - Somar os resultados dos produtos calculados no passo anterior 4 - Sistema - Retornar o produto da somatória calculada no passo anterior		

6.7.2.6. Calcular fator de ajuste dos PF

Tabela 34 - Requisito F0021 (Calcular fator de ajuste dos PF)

Código	F00021	Nome	Calcular fator de ajuste dos PF
Justificativa	Caso de uso responsável por calcular o Fator de Ajuste para ajustar o tamanho bruto do projeto de acordo com as características definidas.		
Escopo	Irà calcular o fator de ajuste para o cálculo dos pontos de funções ajustados		
Pré-condições	- Lista dos níveis de influência de cada característica associadas ao projeto		
Pós-condições	Fator de ajuste para PF não ajustados		
Descrição	1 - Sistema - Somar o nível de influência selecionado para cada característica de PF 2 - Sistema - Aplicar, ao resultado do passo anterior à fórmula 1, onde [FA] é fator de ajuste para cada característica do projeto e [SNI] é somatória dos níveis de influência calculado no passo anterior		
Fórmulas	1 - $[FA = 0,65 + 0,01 * SNI]$		

6.7.2.7. Estimar usando o COCOMO

Tabela 35 - Requisito F0022 (Estimar usando o COCOMO)

Código	F00022	Nome	Estimar usando o COCOMO
Justificativa	Caso de uso responsável por executar os cálculos para realizar estimativas baseado no COCOMO para o projeto		
Escopo	Irà controlar todos os passos para realizar as estimativas pelo COCOMO		
Pré-condições	<ul style="list-style-type: none"> - Dados do Projeto - Tamanho do projeto em Pontos de Função - Dados das linguagens associadas ao projeto - Lista com os pesos das características associadas ao projeto - Valores para o cálculo do esforço - Valores para o cálculo do prazo 		
Pós-condições	Dados das estimativas		
Descrição	1 - Sistema - Executar o caso de uso F0018 2 - Sistema - Multiplicar o tamanho do projeto pelo número de instruções de cada linguagem selecionada 3 - Sistema - Somar os produtos calculados no passo anterior 4 - Sistema - Estimar o esforço em H/M (homens/mês) através da fórmula 1 5 - Sistema - Executar o caso de uso F0016 6 - Sistema - Multiplicar o esforço bruto pelo fator de ajuste COCOMO 7 - Sistema - Estimar o prazo em meses através da fórmula 2 8 - Sistema - Estimar a equipe em pessoas através da fórmula 3 9 - Sistema - Retornar ao Usuário as estimativas para o projeto		
Fórmulas	1 - $[H/M = \text{Fator_Multiplicacao} * (KDSI ^ \text{Exponencial})]$ 2 - $[P = \text{Fator_Multiplicacao} * (H/M ^ \text{Exponencial})]$ 3 - $[E = (H/M) / (P)]$		
Relacionamento	F0016 - Calcular o fator de ajuste COCOMO F0018 - Carregar dados para estimar com COCOMO		

Figura 39 - Diagrama de seqüência para o requisito F0022

6.7.2.8. Fazer distribuição de Rayleigh

Tabela 36 - Requisito F0023 (Fazer distribuição de Rayleigh)

Código	F00023	Nome	Fazer distribuição de Rayleigh
Justificativa	Caso de uso responsável por realizar o cálculo de estimativa da divisão de tipos de distribuição por fase e tamanho do projeto		
Escopo	Irá controlar todos os passos para realizar as distribuição do esforço e prazo por Rayleigh		
Pré-condições	<ul style="list-style-type: none"> - Dados do Projeto - Tamanho do projeto em Pontos de Função - Dados das linguagens associadas ao projeto - Lista com os pesos das características associadas ao projeto - Valores para o cálculo do esforço - Valores para o cálculo do prazo - Lista com os percentuais para cada fase do tamanho do projeto 		
Pós-condições	Dados da distribuição		
Descrição	1 - Sistema - Executar o caso de uso F0021 2 - Sistema - Obter os valores para H/M e P 3 - Sistema - Calcular o percentual de cada atividade para as fases dos tipos de distribuição de Rayleigh 4 - Sistema - Retornar a lista com os valores percentuais e líquidos para o Usuário		
Relacionamento	F0021 - Estimar usando o COCOMO		

Figura 40 - Diagrama de seqüência para o requisito F0023

6.7.2.9. Manter projetos

Tabela 37 - Requisito F0024 (Manter Projetos)

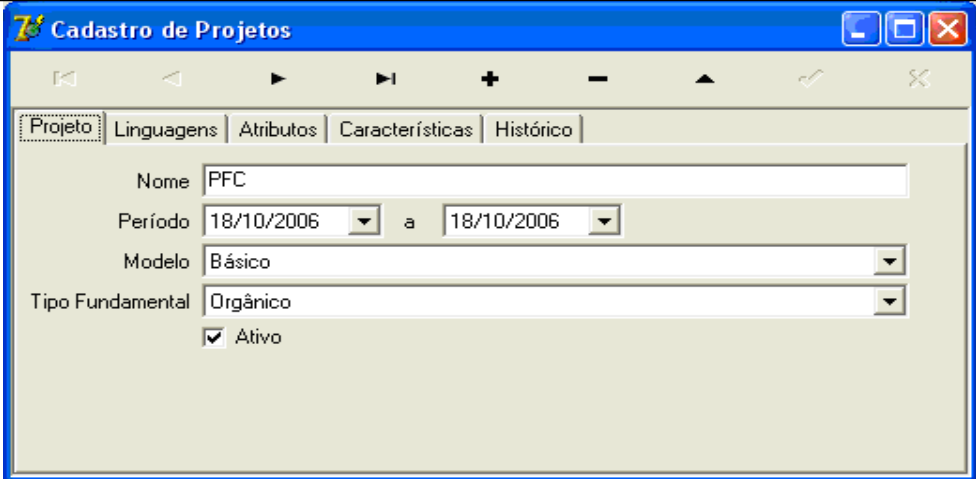
Código	F00024	Nome	Manter projetos
Justificativa	Caso de uso responsável por manter as informações do projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Protótipo			
Figura 41 - Protótipo da tela de cadastro de projetos			
Relacionamentos	F0033 – Manipular Registros		

Figura 42 - Diagrama de seqüência para o requisito F0024

6.7.2.10. Registrar Atributos do projeto

Tabela 38 - Requisito F0025 (Registrar atributos do projeto)

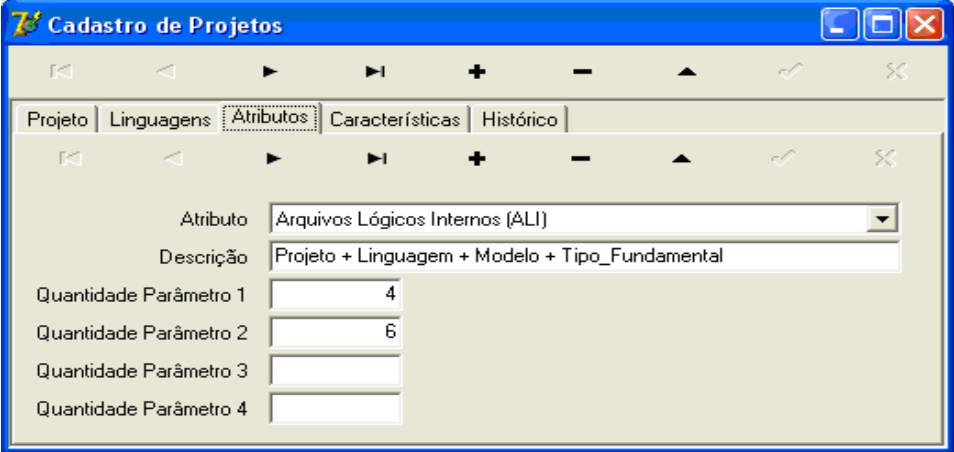
Código	F00025	Nome	Registrar atributos do projeto
Justificativa	Caso de uso responsável por manter as informações dos atributos pertencentes a cada projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Relacionamento	F0023 – Manter Projetos F0033 – Manipular Registros		
Protótipo			

Figura 43 - Protótipo da tela de cadastro de atributos do projeto**Figura 44 - Diagrama de seqüência para o requisito F0025**

6.7.2.11. Registrar Características de Pontos de Função

Tabela 39 - Requisito F0026 (Registrar Características de Pontos de Função)

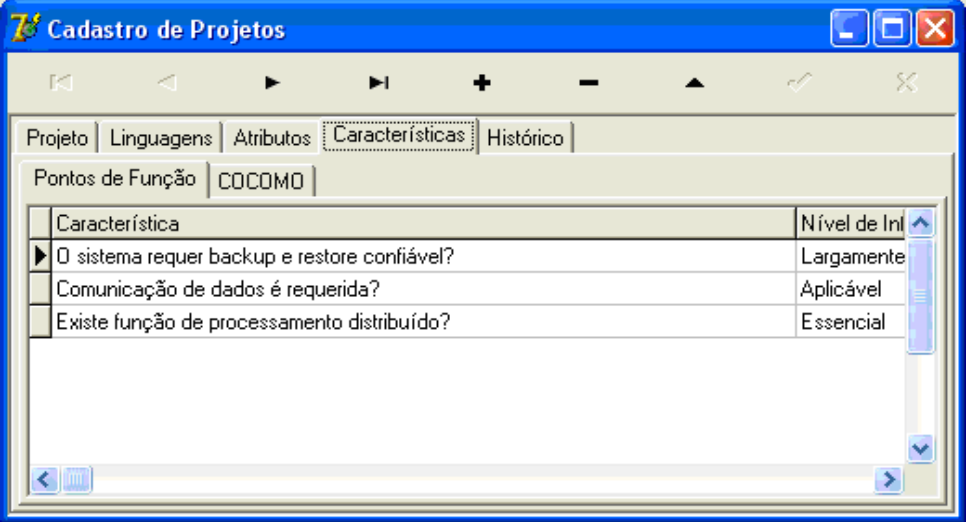
Código	F00026	Nome	Registrar características de Pontos de Função
Justificativa	Caso de uso responsável por manter a escala atribuída às características do projeto para medição em Pontos de Funções		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Relacionamento	F0023 – Manter Projetos F0033 – Manipular Registros		
Protótipo			

Figura 45 - Protótipo da tela de cadastro de características de PF do projeto**Figura 46 - Diagrama de sequência para o requisito F0026**

6.7.2.12. Registrar as linguagens utilizadas pelo projeto

Tabela 40 - Requisito F0027 (Registrar as linguagens utilizadas pelo projeto)

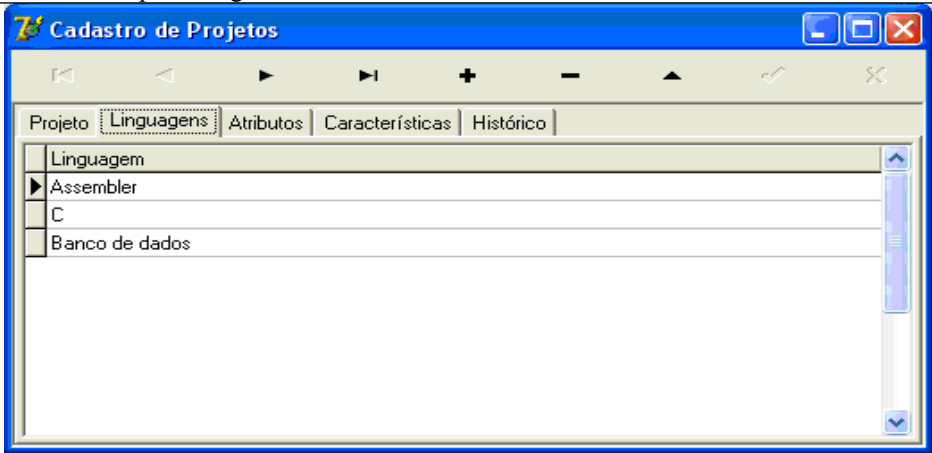
Código	F00027	Nome	Registrar as linguagens utilizadas pelo projeto
Justificativa	Caso de uso responsável por manter as informações relacionadas às linguagens que serão utilizadas no projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Relacionamento	F0023 – Manter Projetos F0033 – Manipular Registros		
Protótipo			

Figura 47 - Protótipo da tela de cadastro de linguagens do projeto**Figura 48 - Diagrama de seqüência do requisito F0027**

6.7.2.13. Registrar características do COCOMO para o projeto

Tabela 41 - Requisito F0028 (Registrar características do COCOMO para o projeto)

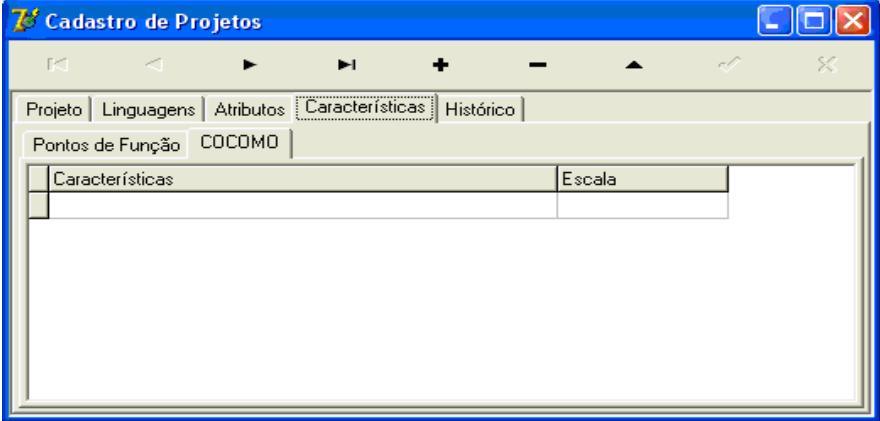
Código	F00028	Nome	Registrar características do COCOMO para o projeto
Justificativa	Caso de uso responsável por manter a escala atribuída às características do projeto para realizar as estimativas do COCOMO para o projeto		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros		
Pré-condições	Dados a serem mantidos		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Executa o caso de uso "Manipular Registros"		
Relacionamento	F0023 – Manter Projetos F0033 – Manipular Registros		
Protótipo			

Figura 49 - Protótipo da tela de registro de características para COCOMO**Figura 50 - Diagrama de sequência para o requisito F0028**

6.7.2.14. Registrar histórico de tamanho de projeto

Tabela 42 - Requisito F0029 (Registrar histórico de tamanho de projeto)

Código	F00029	Nome	Registrar histórico de tamanho de projeto
Justificativa	Caso de uso responsável por manter informações históricas sobre o tamanho do projeto a cada execução do caso de uso "Calcular tamanho em Pontos de Função"		
Escopo	O usuário deve ser capaz de incluir, alterar ou excluir registros		
Pré-condições	<ul style="list-style-type: none"> - Projeto medido - Tamanho do projeto - Data da medição 		
Pós-condições	Alterações persistidas no banco de dados		
Descrição	1 - Sistema - Armazena as informações		
Relacionamento	F0017 – Calcular tamanho em Pontos de Função		

6.7.3. Requisitos para Operadores

6.7.3.1. Alteração de registro

Tabela 43 - Requisito F0030 (Alteração de registro)

Código	F00030	Nome	Alteração de registro
Justificativa	Caso de uso que registra as alterações feitas pelo Usuário em um registro		
Escopo	Caso de uso responsável por alterar o registro		
Pré-condições	Registro estar selecionado		
Pós-condições	Alteração estar persistida no banco de dados		
Descrição	<ul style="list-style-type: none"> 1 - Usuário - Selecionar o registro que deseja alterar 2 - Usuário - Alterar as informações desejadas 3 - Usuário - Executar a ação "Alterar" 4 - Sistema - Verificar se todos os campos obrigatórios foram preenchidos 5 - Sistema - Validar as informações dos campos 6 - Sistema - Registrar os dados incluídos pelo Administrador 7 - Sistema - Apresentar a mensagem "Dados alterados com sucesso!" ao Usuário 		
Fluxos Alternativos	<ul style="list-style-type: none"> 1. Para o caso de algum campo obrigatório do cadastro não ter sido preenchido pelo Usuário: <ul style="list-style-type: none"> 1.1. Sistema - Apresenta a mensagem "Campo obrigatório não preenchido!" 1.2. Sistema - Retorna ao início do Fluxo Principal 2. Para o caso do Sistema detectar alguma informação inválida entrada pelo Usuário: <ul style="list-style-type: none"> 2.1. Sistema - Apresenta a mensagem "Verifique as informações, existe dado inválido!" 2.2. Sistema - Retorna ao início do Fluxo Principal 3. Para o caso do Sistema não conseguir salvar a informação: <ul style="list-style-type: none"> 3.1. Sistema - Apresenta a mensagem "Não foi possível concluir a operação de registro" 3.2. Sistema - Retorna ao início do Fluxo Principal 		

6.7.3.2. Exclusão de Registro

Tabela 44 - Requisito F0031 (Exclusão de registro)

Código	F00031	Nome	Exclusão de registro
Justificativa	Caso de uso que registra a exclusão de um registro pelo Usuário		
Escopo	Caso de uso responsável por excluir o registro selecionado		
Pré-condições	Registro estar selecionado		
Pós-condições	Exclusão estar persistida no banco de dados		
Descrição	1 - Usuário - Seleciona o registro que deseja excluir 2 - Usuário - Executa a ação "Excluir" 3 - Sistema - Solicita uma confirmação para a ação do Usuário 4 - Usuário - Confirma a ação 5 - Sistema - Registra a exclusão 6 - Sistema - Apresenta a mensagem "Dado excluído com sucesso!" ao Usuário		
Fluxos Alternativos	1. Sistema solicita uma confirmação para o Usuário e este não confirma, o sistema irá: 1.1. Sistema - Retornar ao início do Fluxo Principal		

6.7.3.3. Inclusão de registros

Tabela 45 - Requisito F0032 (Inclusão de registros)

Código	F00032	Nome	Inclusão de registro
Justificativa	Caso de uso que registra a inclusão de um registro pelo Usuário		
Escopo	Caso de uso responsável por incluir um registro		
Pré-condições	Registro estar selecionado		
Pós-condições	Exclusão estar persistida no banco de dados		
Descrição	1 - Usuário - Preenche as informações solicitadas pelo sistema 2 - Usuário - Executa a ação "Incluir" 3 - Sistema - Verifica se todos os campos obrigatórios foram preenchidos 4 - Sistema - Valida as informações dos campos 5 - Sistema - Registra os dados incluídos pelo Usuário 6 - Sistema - Apresenta a mensagem "Dados incluídos com sucesso!" ao Usuário		
Fluxos Alternativos	1. Para o caso do Sistema detectar alguma informação inválida entrada pelo Usuário: 1.1. Sistema - Apresenta a mensagem "Verifique as informações, existe dado inválido!" 1.2. Sistema - Retorna ao início do Fluxo Principal 2. Para o caso de algum campo obrigatório do cadastro não ter sido preenchido pelo Usuário: 2.1. Sistema - Apresenta a mensagem "Campo obrigatório não preenchido!" 2.2. Sistema - Retorna ao início do Fluxo Principal 3. Para o caso do Sistema não conseguir salvar a informação: 3.1. Sistema - Apresenta a mensagem "Não foi possível concluir a operação de registro" 3.2. Sistema - Retorna ao início do Fluxo Principal		

6.7.3.4. Manipular registro

Tabela 46 - Requisito F0033 (Manipular registro)

Código	F00033	Nome	Manipular registro
Justificativa	Caso de uso responsável por executar a decisão do usuário para manipular um registro		
Escopo	Caso de uso responsável por manipular os registros		
Pré-condições	Registro estar selecionado		
Descrição	1 - Usuário - Deseja incluir uma característica, executar o caso de uso "Inclusão de Registro" 2 - Usuário - Deseja alterar uma característica, executar o caso de uso "Alteração de Registro" 3 - Usuário - Deseja excluir uma característica, executar o caso de uso "Exclusão de Registro"		
Fluxos Alternativos	1. Sistema solicita uma confirmação para a ação do Usuário e este não confirmar, o sistema irá: 1.1. Sistema - Retornar ao início do Fluxo Principal 2. Para o Usuário cancelar a ação que estiver realizando: 2.1. Usuário - Executa a ação "Cancelar" 2.2. Sistema - Solicita uma confirmação do Usuário 2.3. Usuário - Confirma a operação desejada 2.4. Sistema - Retorna ao início do Fluxo Principal		

6.8. Projeto

A seguir é apresentado o projeto da ferramenta, contendo o diagrama de entidades-relacionamento e os diagramas de classe que compõem o projeto.

6.8.1. Diagrama de entidade-relacionamento

Figura 51 - Diagrama de entidades-relacionamento

6.8.2. Diagrama de classe da camada de persistência

Figura 52 - Diagrama de classe para a camada de persistência

6.8.3. Diagrama de classe da camada de modelo

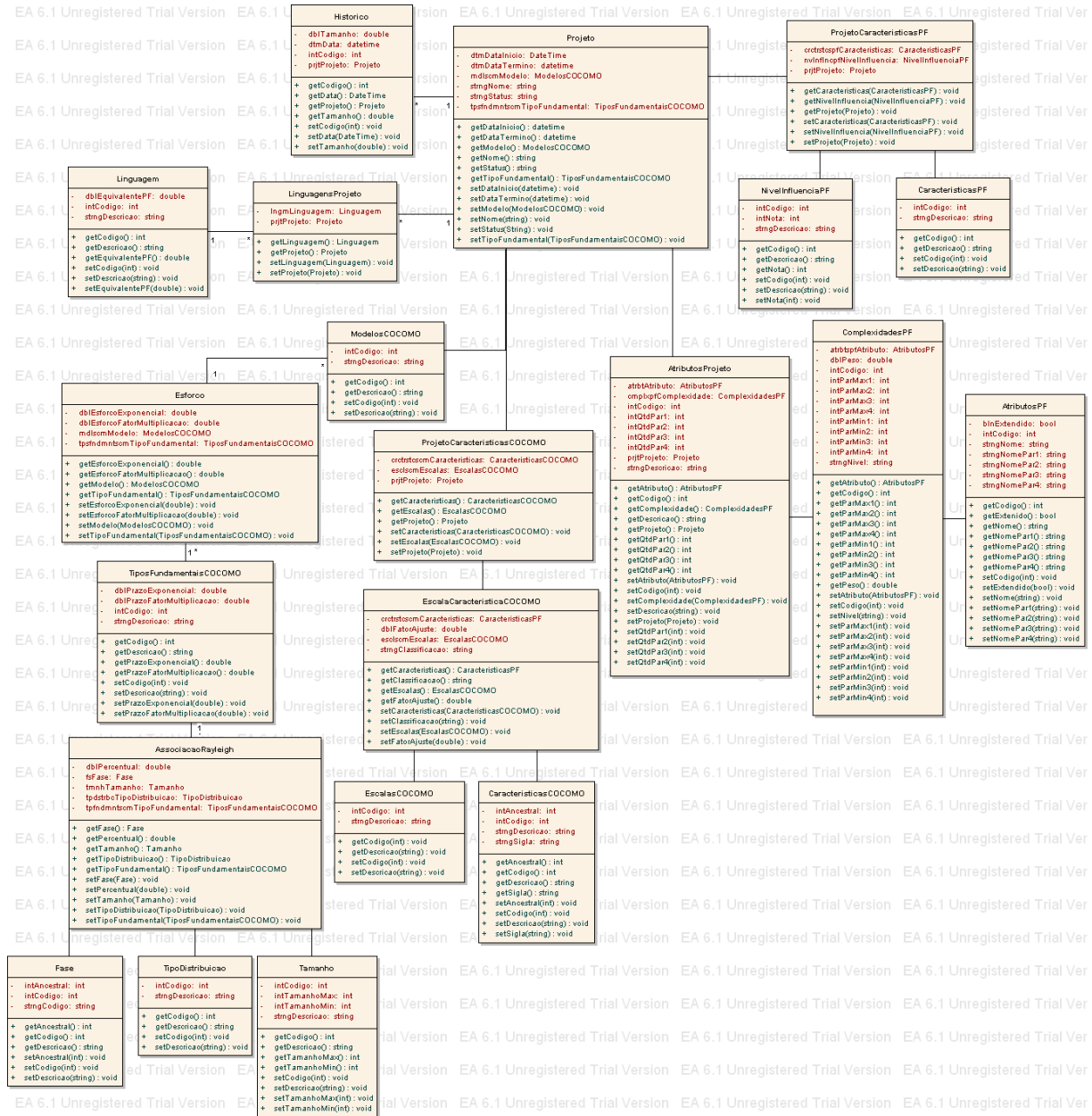


Figura 53 - Diagrama de classe da camada de modelo

6.8.4. Diagrama de classe da camada de negócios

Figura 54 - Diagrama de classe da camada de negócio

6.8.5. Diagrama de classes da camada de apresentação

Figura 55 - Diagrama de classes da camada de apresentação

6.9. Validação

Para validação das informações da ferramenta, bastará estar simulando um projeto como o exemplo contido no Apêndice A deste trabalho.

CAPÍTULO VII

Conclusão

Após o estudo de como melhorar a qualidade dos produtos e do serviço a ser prestado pela equipe de desenvolvimento, mostramos os principais pontos a serem explorados pela gerência de projeto, para que acompanhe o andamento do projeto.

Utilizando métricas, a gerência de projeto, poderá utilizar os resultados deste projeto como base para calcular o tamanho do *software* a ser desenvolvido. Dessa forma, poderá: saber o custo; saber o prazo requerido; estimular a equipe a cumprir os prazos; organizar o orçamento, evitando prejuízos.

Mostramos também que o gerente poderá visualizar a tendência de consumo do orçamento e do cronograma para cada fase do projeto, de forma a ter estimativas da quantidade de trabalho feito ou a ser feito por sua equipe.

Por fim, foram mostrados diversos pontos a serem atacados pela gerência, para que a qualidade do projeto e o atendimento sejam cada vez melhores, eficiente e menos problemáticos para os clientes.

REFERÊNCIAS BIBLIOGRÁFICAS

1. DeMarco, T. Structured Analysis and System Specification. New York: Yourdon Press, 1978.
2. Coad, P. e Yourdon, E. Object-oriented Analysis. Englewood Cliffs, NJ: Prentice-Hall, 1990
3. Booch, G. e Rumbaugh, J. et al. The Unified Modeling Language User Guide. Reading, MA: Addison-Wesley Longman
4. Sommerville, Ian, Software Engineering. Addison-Wesley, 2001
5. Associação para promoção da excelência do software brasileiro - SOFTEX, MPS.BR - Guia Geral. SOFTEX, 2005
6. Roche, J.M. Software Metrics and measurement Principles. Software Engineering Notes, ACM, 1994
7. DeMarco, T. Controlling Software Projects. Yourdon Press, 1982
8. Davis, A., et al. Identifying and Measuring Quality in a Software Requirements Specification. Proc. First Intl. Software Metrics Symposium, IEEE, Baltimore, MD, 1993.
9. Card, D.N. and R.L. Glass, Measuring Software Design Quality. Prentice-Hall, 1990
10. Bieman, J.M. and L.M. Ott, Measuring Functional Cohesion. IEEE Trans. Software Engineering, 1994
11. Dhama, H. Quantitative models of Cohesion and Coupling in Software. Journal of Systems and Software, April 1995
12. McCabe, T.J. and A.H. Watson, Software Complexity, Crosstalk. vol 7, nr 12, December 1994
13. Halstead, M., Elements of Software Science. North-Holland, 1977
14. Pressman, Roger S., Software Engineering: A practitioner's Approach. McGraw-Hill, 2001
15. Software Engineering Standards. 1994 edition, IEEE, 1994
16. Albrecht, A.J., Measuring application development productivity. SHARE/GUIDE ITM Application Development Symposium, 1979
17. Jones, Capers. Applied Software Measurement. McGraw-Hill, New York: 1991.
18. IFPUG. Function Points Counting Practice Manual - Release 4.0. Westerville, Ohio: 1994.
19. Boehm, B., Software Engineering Economics. Prentice-Hall, 1981

GLOSSÁRIO

Defeito: Problemas com o *software* encontrados e notificados pelo usuário;

Erro: Problemas com o *software* encontrados e notificados pela equipe de teste;

Escopo: Delimitação do assunto a ser abordado pelo projeto;

Esforço: Gasto de tempo de profissional com um trabalho;

Manutenção: Correção a ser realizada sobre o *software*, componente do *software* ou ambiente de produção do *software*;

Modelos de desenvolvimento: Modelos utilizados em uma tentativa de criar uma única técnica mundialmente aceitam;

Modelos de Maturidade: Modelo de melhoria de processo. Avalia as condições das equipes de desenvolvimento da empresa candidata a obter uma certificação de excelência de qualidade. Modelo de maturidade mundialmente conhecido é o CMMI;

Qualidade: Projeto de *software* onde o produto final foi entregue dentro do prazo, custo e funcionalidades atendidas conforme as empresas desejavam;

Recursos: Recurso humano ou tecnológico. Qualquer objeto que trabalhe em prol do projeto;

Versão: Coleção de todos os artefatos do projeto a ser distribuído a todos ou alguns clientes.

APÊNDICES

APÊNDICE A – Medindo o *software*

Tabela 47 - Medindo a complexidade dos ALI

ALI	TER	TED	CPLX
Projeto + Linguagem + Modelo + Tipo Fundamental	4	6	Baixa
Projeto ALI + ALI Complexidade	2	6	Baixa
Projeto AIE + AIE Complexidade	2	6	Baixa
Projeto EE + EE Complexidade	2	6	Baixa
Projeto SE + SE Complexidade	2	6	Baixa
Projeto CE + CE Complexidade	2	8	Baixa
Projeto Historico	1	3	Baixa
Projeto Fator Ajuste PF + Fator Ajuste PF + PF Nota	3	2	Baixa
Projeto Fator Ajuste COCOMO + Fator Ajuste COCOMO + COCOMO Pesos	3	3	Baixa
Esforço Modelo Tipo + Esforço + Tipo Fundamental	3	4	Baixa
Prazo Tipo + Tipo Fundamental	2	3	Baixa
ALI Complexidade	1	6	Baixa
AIE Complexidade	1	6	Baixa
EE Complexidade	1	6	Baixa
SE Complexidade	1	6	Baixa
CE Complexidade	1	10	Baixa

Tabela 48 - Medindo a complexidade das EE

EE	TAR	TED	Complex.
Incluir Projeto	1	5	Baixa
Alterar Projeto	1	5	Baixa
Excluir Projeto	1	1	Baixa
Incluir ALI	1	5	Baixa
Alterar ALI	1	5	Baixa
Excluir ALI	1	2	Baixa
Incluir AIE	1	5	Baixa
Alterar AIE	1	5	Baixa
Excluir AIE	1	2	Baixa
Incluir EE	1	5	Baixa
Alterar EE	1	5	Baixa
Excluir EE	1	2	Baixa
Incluir SE	1	5	Baixa
Alterar SE	1	5	Baixa
Excluir SE	1	2	Baixa
Incluir CE	1	7	Baixa
Alterar CE	1	7	Baixa
Excluir CE	1	2	Baixa
Incluir Fator de Ajuste PF	1	2	Baixa
Alterar Fator de Ajuste PF	1	2	Baixa
Excluir Fator de Ajuste PF	1	2	Baixa
Incluir Fator de Ajuste COCOMO	1	2	Baixa
Alterar Fator de Ajuste COCOMO	1	2	Baixa
Excluir Fator de Ajuste COCOMO	1	2	Baixa

Tabela 49 - Medindo a complexidade das SE

SE	TAR	TED	Complex.
Histórico dos Projetos	2	5	Baixa
Listagem dos atributos de Projetos	6	8	Alta
Listagem dos fatores de ajuste de PF dos Projetos	2	5	Baixa
Listagem dos fatores de ajuste do COCOMO dos Projetos	2	5	Baixa
Listagem dos esforços dos projetos	2	4	Baixa
Listagem dos prazos dos projetos	3	4	Baixa

Tabela 50 - Medindo a complexidade das CE

CE	Entrada		Saída		Complex.
	TAR	TED	TAR	TED	
Consultar Projetos por Data de Início	1	1	1	3	Baixa
Consultar Projetos por Linguagem	1	1	1	3	Baixa
Consultar Projetos por Status	1	1	1	3	Baixa
Consultar histórico por Data	1	1	1	3	Baixa
Consultar histórico por Tamanho	1	1	1	3	Baixa
Consultar Complexidade ALI	1	4	1	2	Baixa
Consultar Complexidade AIE	1	4	1	2	Baixa
Consultar Complexidade EE	1	4	1	2	Baixa
Consultar Complexidade SE	1	4	1	2	Baixa
Consultar Complexidade CE	1	4	1	2	Baixa

Tabela 51 - Medindo o total de Pontos de Função não ajustados

Tabela 31 - Tabela de Pontuação e total de Pontos de Função nas atividades				
Função	Ocorrências	Complexidade	Peso	Resultado
Arquivos Lógicos Internos	16	Baixa	3	48
	0	Média	4	0
	0	Alta	6	0
	Sub-Total			48
Entradas Externas	24	Baixa	7	168
	0	Média	10	0
	0	Alta	15	0
	Sub-Total			168
Saídas Externas	6	Baixa	5	30
	0	Média	7	0
	1	Alta	10	10
	Sub-Total			40
Consultas Externas	10	Baixa	3	30
	0	Média	4	0
	0	Alta	6	0
	Sub-Total			30
SPFNA			286	

Tabela 52 - Cálculo do Fator de Ajuste Total

Fator de Ajuste	Fator de Ajuste
Comunicação de Dados	5
Processamento de Dados Distribuído	4
Desempenho	4
Configuração Pesadamente Utilizada	3
Taxa de Transação	5
Entrada de Dados On-Line	5
Eficiência do Usuário Final	5
Atualização On-Line	5
Processamento Complexo	4
Reutilização	5
Facilidade de Instalação	5
Facilidade Operacional	3
Múltiplas Instalações	5
Facilidade de Mudança	5
SNI	63

De acordo com a equação 1, segue o cálculo para o Fator de Ajuste Líquido:

$$FA = 0,65 + (0,01 * SNI) = 0,65 + (0,01 * 63) = 1,28 \quad (9)$$

Onde:

FA = Fator de ajuste

SNI = Somatório dos níveis de influência

Segundo a equação 2, segue o cálculo do Total de Pontos de Função Ajustados:

$$PF = FA * SPFNA = 1,28 * 286 = 366 \quad (10)$$

Onde:

PF = Pontos de Função ajustados

FA = Fator de Ajuste

SPFNA = Somatória de Pontos de Função Não Ajustados

Segue o cálculo de instruções fontes a serem entregues pelo software de acordo com o total de pontos de função para um projeto implementado em linguagens de quarta geração com uma relação de 20 DSI por Ponto de Função:

$$KDSI = PF * IR = 366 * 20 = 7,32 \quad (11)$$

Onde:

$KDSI = \text{Delivery source instructions}$ (Instruções fonte entregues)

PF = Pontos de Função ajustados

IR = Instruções fontes por Pontos de Função

Segundo as equações da tabela 7, segue o cálculo de quantidade de esforço de pessoas a ser consumidas por mês para um projeto Intermediário e Orgânico:

$$H/M = 3,2 * (KDSI)^{1,05} = 3,2 * 7,32^{1,05} = 25,88 \quad (12)$$

Onde:

H/M = Quantidade de pessoas por mês

$KDSI = \text{Delivery Source Instructions}$ (Instruções fonte entregues)

Tabela 53 - Cálculo do Fator de Ajuste Total

Fator de Ajuste	Fator de Ajuste
RELY	0,75
DATA	1,16
CPLX	0,85
TIME	1,00
STOR	1,00
VIRT	0,87
TURN	0,87
ACAP	0,82
AEXP	1,29
PCAP	0,70
VEXP	0,90
LEXP	1,00
MODP	0,91
TOOL	0,91
SCED	1,00
Fator de Ajuste Total	0,31

A equação 13 ajusta o esforço de acordo com a complexidade do projeto Intermediário:

$$H/M = 25,88 * 0,31 = 8,02 \quad (13)$$

A equação 14, segue as equações da tabela 11 para o cálculo do prazo em projetos Intermediários e Orgânicos:

$$\text{Prazo} = 2,5 * 8,02^{0,38} = 5,5 \text{ meses} \quad (14)$$

A equação 15 calcula o tamanho da equipe a ser reunida:

$$\text{Equipe} = 8,02 / 5,5 = 2 \text{ pessoas} \quad (15)$$

De acordo com a distribuição de Rayleigh, a tabela 35 distribui o consumo de recursos durante o projeto.

Tabela 54 - Distribuição do esforço durante o projeto

Fase	Esforço			
	%	H/M	Prazo	Equipe
Planos e Requisitos	6	0,48	0,33	2
Projeto do Produto	16	1,28	0,88	2
Programação	68	5,45	3,74	2
Projeto Detalhado	26	2,09	1,43	
Codificação	42	3,37	2,31	
Integração e teste	16	1,28	0,88	2